Politecnico di Milano

School of Industrial and Information Engineering
Master of Science in Automation and Control Engineering

Master Thesis

Analysis of Identified Models for Quadruple-Tank Process

Supervisor
Prof. Riccardo Scattolini

Candidate
Korcan Defineci
925522

Academic Year 2020-2021

Acknowledgements

First, I am truly grateful to my parents who encouraged and supported me during my master's degree at Politecnico di Milano.

I would also like to thank Prof. Riccardo Scattolini for his guidance and patience during the progression of my thesis.

Last but not least, I am thankful to all healthcare professionals around the world who are fighting against the coronavirus pandemic.

Korcan Defineci

Contents

List of Figures

List of Tables

Abstract

The objective of this thesis is to analyze the identified models for the Quadruple-Tank Process. In this work, both linear and nonlinear model identification methods are employed by means of the MATLAB/Simulink.

In the first part of the work, the Quadruple-Tank System is described and the mathematical model of the process is stated. Then, the linearized model is obtained and the main characteristics of the corresponding system is examined by using the linearized model.

In the second part of the thesis, seven different methods are utilized so as to identify the system. Four of them belong to linear models and the remaining ones are the part of nonlinear models. The linear methods are State-Space, ARMAX, Output-Error (OE), and ARX models; the nonlinear ones are Polynomial NARX, Hammerstein-Wiener, and Neural NARX (NNARX) models. For all identified models, first, the structure of the considered models is specified. Afterwards, the models are estimated by using the estimation dataset. In the validation phase, the estimated models for each method are tested on the different datasets and the acquired models are compared based on NRMSE criterion and residual analysis in order to select one of them. After that, the parameters of the selected models are reported. Finally, the seven selected models are contrasted to decide which one is the best.

Sommario

L'obiettivo di questa tesi è analizzare i modelli identificati per il Processo Quadruple-Tank. In questo lavoro vengono utilizzati metodi di identificazione di modelli sia lineari che non lineari tramite MATLAB/Simulink.

Nella prima parte del lavoro viene descritto il Quadruple-Tank System e viene presentato il modello matematico del processo. Quindi, si ottiene il modello linearizzato e si esaminano le caratteristiche principali del sistema corrispondente utilizzando il modello linearizzato.

Nella seconda parte della tesi vengono utilizzati sette diversi metodi per identificare il sistema. Quattro di essi sono modelli lineari e i restanti sono modelli non lineari. I metodi lineari sono i modelli State-Space, ARMAX, Output-Error (OE) e ARX; quelli non lineari sono i modelli Polynomial NARX, Hammerstein-Wiener e Neural NARX (NNARX). Per tutti i modelli identificati, in primo luogo, viene specificata la struttura dei modelli considerati. Successivamente, i modelli vengono stimati utilizzando il set di dati di stima. Nella fase di validazione, i modelli stimati per ciascun metodo vengono testati sui diversi dataset e i modelli acquisiti vengono confrontati in base al criterio NRMSE e all'analisi residua per selezionarne uno. Successivamente vengono riportati i parametri dei modelli selezionati. Infine, i sette modelli selezionati vengono messi a confronto per decidere quale sia il migliore.

List of Abbreviations

MIMO	Multiple-Input and Multiple-Output
QTP	Quadruple-Tank Process
LCD	Least Common Denominator
GCD	Greatest Common Divisor
N4SID	Numerical Algorithm for the Subspace State-Space System Identification
NRMSE	Normalized Root-Mean-Square Error
SEM	Simulation Error Minimization
SISO	Single-Input and Single-Output
ARMAX	Auto-Regressive Moving-Average with Exogenous Inputs
OE	Output-Error
SQP	Sequential Quadratic Programming
ARX	Auto-Regressive Exogenous
NARX	Nonlinear Auto-Regressive Exogenous
NNARX	Neural Nonlinear Auto-Regressive Exogenous
LM	Levenberg-Marquardt
MSE	Mean-Square Error
LOOCV	Leave-One-Out Cross-Validation
NMPC	Nonlinear Model Predictive Controller

Introduction

For nonlinear dynamical systems, it is difficult to acquire a precise mathematical model; hence, identification of nonlinear systems is required for most problems encountered in the process industry [6] [7] [8] [9] [10] [11] [12] [13] [14]. In this work, the considered nonlinear system is a Quadruple-Tank, comprehensively described in Chapter 1. For the Quadruple-Tank Process (QTP), some linear models have been identified in ARX, ARMAX, and State-Space forms in [15] and [16]. Besides, several Polynomial and Neural NARX models have been used in order to identify the Quadruple-Tank System, such as in [7], [17]. Some other nonlinear model identification methods like Hammerstein and Wiener have been utilized for the considered process in [18] and [19]. The corresponding system has also been identified by using the Volterra series in [20].

This thesis is aimed at analyzing the identified models for the QTP. For the identification of the corresponding nonlinear dynamical system, both linear and nonlinear models contribute to this work.

Outline

The thesis is organized as follows:

Chapter 1 is aimed at describing the physical properties of the Quadruple-Tank Process. For this purpose, the components of the corresponding system and its working principle are specified. The nonlinear mathematical model of the system is then expressed. The linearized model is also represented and the main characteristics of the plant are determined by analyzing the multi-variable poles, invariant zeros and singular values of the linearized model. Besides, minimum and nonminimum phase modes are explained and their relation with the positions of the process valves are indicated. At the end of this chapter, the simulator of the plant is displayed and the transient responses under stationary operating conditions are shown.

Chapter 2 describes, first, the excitation signals utilized in order to identify the system. Then, the data preprocessing procedure is explained. The prediction and simulation modes are mentioned as well. Afterwards, the input-output delay is defined and how it has been chosen is clarified. Thereafter, the equation of the Normalized Root-Mean-Square Error (NRMSE) is provided and the fit percent quality metric with regard to NRMSE is reported. In Section 2.1, first, the structure of the continuous-time State-Space models is written. In the sequel, many State-Space models with different orders are estimated by employing the estimation dataset. Then, the estimated State-Space models are tested on the validation dataset and the performance of the considered models are evaluated based on NRMSE criterion, whiteness and independence tests. The obtained models are also tested on different datasets so as to attain more reliable results. After these tests, one of the estimated State-Space models is selected and its parameters are stated. In a very similar way, ARMAX, Output-Error, and linear ARX models are examined in Section 2.2, 2.3, and 2.4 respectively. At the end of this chapter, the models selected by using different linear methods are compared.

Chapter 3 initially introduces the general structure of NARX models. The Polynomial NARX models are also touched upon. In the identification phase, first, the orders of the initial NARX model are specified and how it has been acquired is mentioned. Then, the performance of the initialized NARX model is enhanced by adding some nonlinear regressors. After deciding the candidate non-polynomial NARX model, the nonlinear function of the corresponding model is discarded so as to reduce the complexity of the model structure. Subsequently, the candidate Polynomial NARX model is tested on different datasets which have never been utilized before to eliminate the risk of overfitting. Finally, the parameters and the regressors of the selected Polynomial NARX model is reported. At the beginning of Section 3.2, the structure of Hammerstein-Wiener models are demonstrated. Then, many Wiener models with different orders are estimated. The estimated models are validated by analyzing the performance of considered models on the different data sets. In the validation stage, various Hammerstein, Wiener, and Hammerstein-Wiener models are also taken into consideration for the candidate orders. Afterwards, the performance of the candidate Wiener model is improved by adjusting the lower and upper bounds of the employed saturation function. After that, the unmodified and the calibrated Wiener models are tested on different test datasets which have not been used before in order to avoid overfitting. At the end of this section, the calibrated model is selected and its equation including the parameters is written. Likewise, at the beginning of Section 3.3, the structure of Neural NARX models is encapsulated. Then, the data division method applied on the estimation dataset is explained and the impact of this method on the “early stopping” technique is mentioned. Subsequently, the regression analysis is described. In the validation stage, the trained NNARX models are tested on three different datasets. In order to get more robust results, twenty NNARX models with different weights and biases are trained for each order pairs (n_a and n_b), and then the average of their fit percent is calculated. Moreover, some other NNARX models are trained with different number of neurons in the hidden layer. Next, three candidate model structures are determined. For each model structure, two different NNARX models are trained with different weights and biases. Thus, six candidate Neural NARX models are obtained. In order to eliminate the risk of overfitting, these models are tested on the different datasets that have never been employed before. At the end of Section 3.3, the parameters of the selected NNARX model is reported. In the sequel, the selected Polynomial NARX, Wiener, and Neural NARX models are compared with regard to fit percent quality metric. After the comparison, it is seen that the selected Neural NARX (NNARX) model performs on all of the test datasets better than the other chosen models.

Finally, some conclusions are drawn for the identification of linear and nonlinear models and future developments of this work are discussed.

Chapter 1

Physical System

In this chapter, the system is described by specifying its dynamics and parameters. Then, given an operating condition, the corresponding linearized model is presented and its main characteristics are studied in terms of poles, invariant zeros and singular values.

1.1 System Description

The system is composed of four interconnected water tanks, two pumps and two valves. It is widely known as Quadruple-Tank in process industry [21]. This process is commonly used for analyzing multiple-input and multiple-output (MIMO) systems because the performance of the Quadruple-Tank Process significantly changes with respect to valve positions. A real life example of the system can be seen in the following figure.

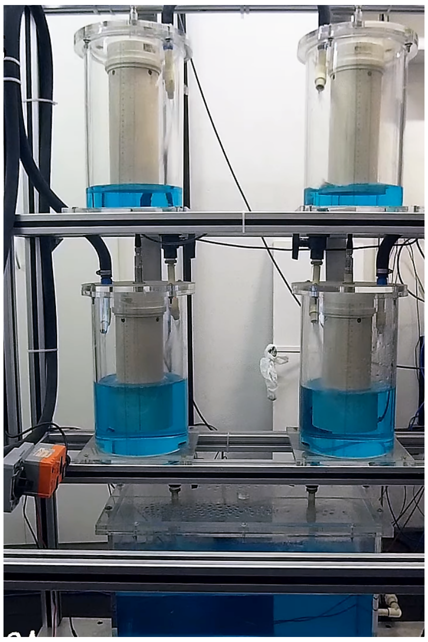

Figure 1.1: Four-Tank system used in laboratory of University of São Paulo [1]

Working Principle

In the Quadruple-Tank Process, there are four interconnected tanks, two pumps and two valves. Levels of the lower tanks can be controlled by means of the pumps. The pump on the left delivers the water to the tanks which are on the lower left and on the upper right. Likewise, the pump on the right distributes the water to the remaining tanks (lower right and upper left). The distribution of the water is done with respect to the valves' positions (see Table 1.2). Besides, because of the settlement of the tanks, the water flows down from the tanks at the top to the ones at the bottom, and the water also pours out from the lower tanks to the reservoir.

1.1.1 Nonlinear Dynamics

The nonlinear mathematical model of the Quadruple-Tank Process is represented in the following equations [21].

$$
\begin{aligned}
\frac{dh_1}{dt} &= -\frac{a_1}{S}\sqrt{2gh_1} + \frac{a_3}{S}\sqrt{2gh_3} + \frac{\gamma_a}{S}q_a \\
\frac{dh_2}{dt} &= -\frac{a_2}{S}\sqrt{2gh_2} + \frac{a_4}{S}\sqrt{2gh_4} + \frac{\gamma_b}{S}q_b \\
\frac{dh_3}{dt} &= -\frac{a_3}{S}\sqrt{2gh_3} + \frac{(1-\gamma_b)}{S}q_b \\
\frac{dh_4}{dt} &= -\frac{a_4}{S}\sqrt{2gh_4} + \frac{(1-\gamma_a)}{S}q_a
\end{aligned}
\tag{1.1}
$$

A schematic diagram of the system is demonstrated in Figure 1.2.

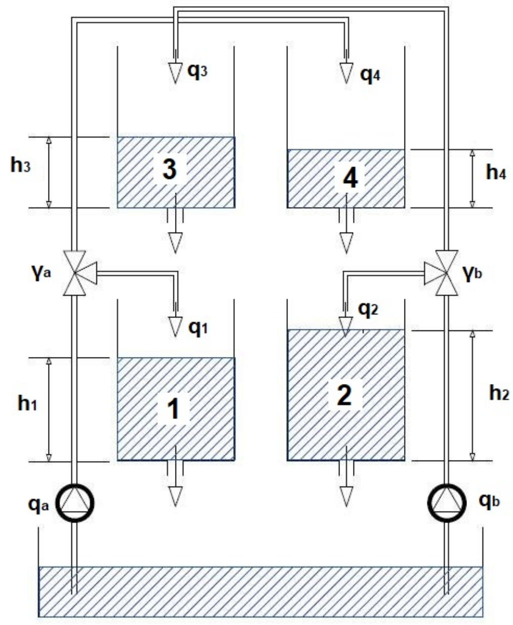

Figure 1.2: Schematic diagram of the Quadruple-Tank Process [2]

According to the system shown in Figure 1.2, the inputs are the flow rates to the both pumps and the outputs are the levels in the lower water tanks. The inputs, the outputs and the states can be expressed in the vector form as

$$\begin{aligned}
\boldsymbol{u} &= \begin{bmatrix} q_a & q_b \end{bmatrix}^T \\
\boldsymbol{y} &= \begin{bmatrix} h_1 & h_2 \end{bmatrix}^T \\
\boldsymbol{x} &= \begin{bmatrix} h_1 & h_2 & h_3 & h_4 \end{bmatrix}^T
\end{aligned} \tag{1.2}$$

and the relationship between the flow rates from each outlet pipe to the corresponding tank and the total flow from the pumps are specified by the parameters γ_a and γ_b, which depend on the valves' positions as

$$\begin{aligned}
q_1 &= \gamma_a q_a \\
q_2 &= \gamma_b q_b \\
q_3 &= (1-\gamma_b) q_b \\
q_4 &= (1-\gamma_a) q_a
\end{aligned} \tag{1.3}$$

The parameters of the plant are specified in Table 1.1

	Value	**Unit**	**Description**
h_{1max}	1.36	m	Maximum level of the tank 1
h_{2max}	1.36	m	Maximum level of the tank 2
h_{3max}	1.30	m	Maximum level of the tank 3
h_{4max}	1.30	m	Maximum level of the tank 4
h_{min}	0.20	m	Minimum level of all tanks
q_{amax}	0.0009056	m^3/s	Maximum flow rate of q_a
q_{bmax}	0.0011111	m^3/s	Maximum flow rate of q_b
q_{min}	0.00	m^3/s	Minimum flow of q_a and q_b
a_1	0.0001310	m^2	Discharge constant of tank 1
a_2	0.0001510	m^2	Discharge constant of tank 2
a_3	0.0000927	m^2	Discharge constant of tank 3
a_4	0.0000882	m^2	Discharge constant of tank 4
S	0.06	m^2	Cross-section of the tanks
γ_a	0.30		Parameter of the 3-way valve
γ_b	0.40		Parameter of the 3-way valve
h_1°	0.65	m	Linearization level of tank 1
h_2°	0.66	m	Linearization level of tank 2
h_3°	0.65	m	Linearization level of tank 3
h_4°	0.66	m	Linearization level of tank 4
q_a°	0.0004528	m^3/s	Linearization flow rate of q_a
q_b°	0.0005556	m^3/s	Linearization flow rate of q_b
g	9.81	m/s^2	Gravitational acceleration

Table 1.1: System parameters

1.1.2 Linearized Model

The linearization of the nonlinear model is required to understand the main characteristics of the system in a given operating condition.

Stationary operating points used for linearization of the nonlinear model are reported in Table 1.1. At the operating conditions, the equation 1.1 can be linearized by the help of Taylor series expansion by neglecting higher order terms [22].

Define the variables $x_i = h_i$ - h_i° (i=1,2,3,4), $u_1 = q_a$ - q_a° and $u_2 = q_b$ - q_b°. The linearized state-space equations are obtained using Jacobian matrix transformation.

$$\begin{aligned} \frac{dx}{dt} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{1.4}$$

with $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, $y \in \mathbb{R}^p$ and the matrices are

$$\begin{aligned} A &= \begin{bmatrix} -\frac{1}{T_1} & 0 & \frac{1}{T_3} & 0 \\ 0 & -\frac{1}{T_2} & 0 & \frac{1}{T_4} \\ 0 & 0 & -\frac{1}{T_3} & 0 \\ 0 & 0 & 0 & -\frac{1}{T_4} \end{bmatrix} \\ B &= \begin{bmatrix} \frac{\gamma_a}{S} & 0 \\ 0 & \frac{\gamma_b}{S} \\ 0 & \frac{1-\gamma_b}{S} \\ \frac{1-\gamma_a}{S} & 0 \end{bmatrix} \\ C &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \\ D &= \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \end{aligned} \tag{1.5}$$

where the time constants are

$$T_i = \frac{S}{a_i}\sqrt{\frac{2h_i^\circ}{g}}, \qquad i = 1, 2, 3, 4 \tag{1.6}$$

After calculating the values of the time constants with respect to the process parameters seen in Table 1.1, A and B matrices can be written as

$$\begin{aligned} A &= \begin{bmatrix} -0.005998 & 0 & 0.004244 & 0 \\ 0 & -0.006861 & 0 & 0.004007 \\ 0 & 0 & -0.004244 & 0 \\ 0 & 0 & 0 & -0.004007 \end{bmatrix} \\ B &= \begin{bmatrix} 5 & 0 \\ 0 & 6.667 \\ 0 & 10 \\ 11.67 & 0 \end{bmatrix} \end{aligned} \tag{1.7}$$

The state-space representation can be transformed to a transfer matrix $G(s)$ by taking the Laplace transform of equation 1.4. Thus,

$$\begin{aligned} sX(s) &= AX(s) + BU(s) \\ Y(s) &= CX(s) + DU(s) \end{aligned} \tag{1.8}$$

By solving the equations for $X(s)$ first, and then $Y(s)$, the following equations are attained.

$$\begin{aligned} X(s) &= (sI - A)^{-1}BU(s) \\ Y(s) &= [C(sI - A)^{-1}B + D]U(s) \end{aligned} \tag{1.9}$$

The input-output relationship is

$$Y(s) = G(s)U(s) \tag{1.10}$$

And, the transfer function matrix is

$$G(s) = C(sI - A)^{-1}B + D \tag{1.11}$$

The corresponding transfer matrix with 2 inputs and 2 outputs can be represented as

$$G(s) = \begin{bmatrix} G_{qah1}(s) & G_{qbh1}(s) \\ G_{qah2}(s) & G_{qbh2}(s) \end{bmatrix} \tag{1.12}$$

where

$$\begin{aligned} G_{qah1}(s) &= \left[\frac{\gamma_a T_1}{S(1+sT_1)}\right] \\ G_{qah2}(s) &= \left[\frac{(1-\gamma_a)T_2}{S(1+sT_2)(1+sT_4)}\right] \\ G_{qbh1}(s) &= \left[\frac{(1-\gamma_b)T_1}{S(1+sT_1)(1+sT_3)}\right] \\ G_{qbh2}(s) &= \left[\frac{\gamma_b T_2}{S(1+sT_2)}\right] \end{aligned} \tag{1.13}$$

By putting the operating point values into the transfer functions, the transfer matrix with numeric values is

$$G(s) = \begin{bmatrix} \frac{5}{s+0.005998} & \frac{0.04244}{s^2+0.01024s+0.00002546} \\ \frac{0.04675}{s^2+0.01087s+0.00002749} & \frac{6.667}{s+0.006861} \end{bmatrix} \tag{1.14}$$

The transfer matrix $G(s)$ has two finite zeros for γ_a, $\gamma_b \in (0,1)$. At least one invariant zero always has a negative real part, but one zero can be located either in left or the right half-plane. The positions of the process valves determine whether the system is minimum phase or nonminimum phase. In order to find the invariant zeros of $G(s)$, it is needed to take its determinant.

$$|G(s)| = \frac{\gamma_a \gamma_b T_1 T_2}{S^2(1+sT_1)(1+sT_2)} - \frac{(1-\gamma_b)(1-\gamma_a)T_1T_2}{S^2(1+sT_1)(1+sT_2)(1+sT_3)(1+sT_4)} \tag{1.15}$$

by multiplying the term on the left by $(1+sT_3)(1+sT_4)$, the determinant of $G(s)$ is written as

$$\det G(s) = \frac{\gamma_a \gamma_b T_1 T_2 (1+sT_3)(1+sT_4) - (1-\gamma_b)(1-\gamma_a) T_1 T_2}{S^2 (1+sT_1)(1+sT_2)(1+sT_3)(1+sT_4)} \tag{1.16}$$

After making some arrangements, the determinant is

$$\det G(s) = \frac{T_1 T_2}{S^2 \prod_{i=1}^{4} (1+sT_i) \gamma_a \gamma_b} \left((1+sT_3)(1+sT_4) - \frac{(1-\gamma_b)(1-\gamma_a)}{\gamma_a \gamma_b} \right) \tag{1.17}$$

Define a parameter η as [21]

$$\eta = \frac{(1-\gamma_a)(1-\gamma_b)}{\gamma_a \gamma_b} \tag{1.18}$$

If η is very small, then the equation 1.17 becomes

$$0 \approx (1+sT_3)(1+sT_4) \tag{1.19}$$

and the invariant zeros are close to $-1/T_3$ and $-1/T_4$; that is, both zeros have negative real part. In the event that η is very large i.e. it goes to $+\infty$, one zero tends to -∞ and the other one tends to $+\infty$. Moreover, one invariant zero is located at the origin if $\eta = 1$ because the equation 1.17 takes the form of

$$\begin{aligned} 1 &= (1+sT_3)(1+sT_4) && \text{by organizing the terms} \\ 1 &= 1 + (T_3+T_4)s + (T_3T_4)s^2 && \text{and thus} \\ 0 &= s[T_3+T_4+(T_3T_4)s] \end{aligned} \tag{1.20}$$

In this case, sum of the parameters of the valves is equal to 1 since

$$\begin{aligned} (1-\gamma_a)(1-\gamma_b) &= \gamma_a\gamma_b && \text{by distributing the terms} \\ 1-\gamma_a-\gamma_b+\gamma_a\gamma_b &= \gamma_a\gamma_b && \text{thus} \\ \gamma_a+\gamma_b &= 1 \end{aligned} \tag{1.21}$$

Under which conditions the system being minimum or nonminimum phase is specified in Table 1.2.

Condition	**Phase Mode**	**Zero Location**
$0 < \gamma_a + \gamma_b < 1$	Nonminimum phase	One zero is in the RHP
$\gamma_a + \gamma_b = 1$		One zero is at the origin
$1 < \gamma_a + \gamma_b < 2$	Minimum phase	Both zeros are in the LHP

Table 1.2: System dynamics with regard to valves' positions of QTP

Since the flow parameters of the three-way valves γ_a and γ_b are 0.3 and 0.4 respectively in the considered operating point, the process is non-minimum phase and it is expected that the dynamic performance of a control system designed for transfer matrix G(s) is limited.

Main Characteristics

The linearized model of the Quadruple-Tank Process (QTP) has multivariable zeros and poles, and one invariant zero can be located in either the left or the right half-plane by simply changing the position of the valves according to Table 1.2.

In order to find the multivariable poles, one can write the system in minimal form which is completely reachable and observable [23]. The poles of the minimal system coincide with the eigenvalues of the matrix A. Hence, the least common denominator (LCD) is equivalent to

$$\phi(s) = det(sI - A) = 0 \tag{1.22}$$

where $\phi(s)$ is the characteristic polynomial associated with a minimal realization of a system with transfer function matrix $G(s)$.

Since the matrix A is of order four and it is also a triangular matrix, the eigenvalues i.e. the multivariable poles of the system are directly located on the diagonal elements of A. Hence, it is seen in equation 1.7 that the poles are -0.005998, -0.006861, -0.004244 and -0.004007. All the poles have negative real part.

Besides, the individual transfer functions do not have any zeros according to equation 1.14; nonetheless, the transfer function matrix $G(s)$ might have invariant zeros. The system in the Laplace domain (eq. 1.8) can also be stated as

$$\begin{bmatrix} sI - A & -B \\ C & D \end{bmatrix} \begin{bmatrix} X(s) \\ U(s) \end{bmatrix} = \begin{bmatrix} 0 \\ Y(s) \end{bmatrix} \tag{1.23}$$

Let the system matrix $P(s)$ be

$$P(s) = \begin{bmatrix} sI - A & -B \\ C & D \end{bmatrix} \tag{1.24}$$

then, the normal rank of a given matrix $P(s)$ is its rank for any value of s. The multivariable zeros of a system can be considered as invariant zeros providing that these zeros are the values of s such that the rank of the system matrix $P(s)$ is lower than the normal rank r.

By means of the transfer matrix $G(s)$, the invariant zeros can also be determined. Let $z(s)$ be the polynomial of the invariant zeros of $G(s)$. Then, the polynomial $z(s)$ of the invariant zeros of $G(s)$ is the greatest common divisor (GCD) of all the numerators of all the minors of order r of $G(s)$, provided that these minors are written so that they have the polynomial $\phi(s)$ of the poles at the denominator [23].

By putting the values of the parameters from Table 1.1 into the numerator of equation 1.16, the following equation is obtained.

$$33.335s^2 + 0.273347s - 0.00142397 = 0 \tag{1.25}$$

and by solving equation 1.25, the invariant zeros are found as

$$\begin{aligned} s_1 &\approx +0.0036 \\ s_2 &\approx -0.0118 \end{aligned} \tag{1.26}$$

These invariant zeros can also be obtained by using state-space representation i.e.

directly solving the equation 1.24.

It is important to determine where the invariant zeros are because it affects the system's performance. Since one of the invariant zeros is located on the right-half-plane, the process is non-minimum phase system as already noticed by looking at the parameters γ_a and γ_b of the valves (see Table 1.2). Hence, controlling the system will be a more difficult task. The pole-zero map of $G(s)$ is shown in Figure 1.3.

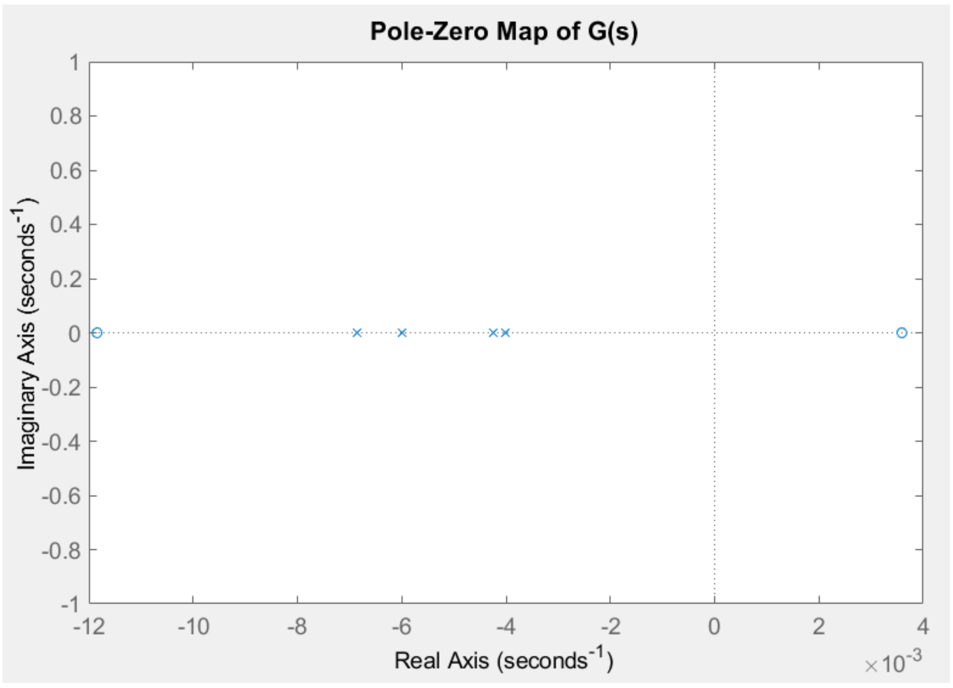

Figure 1.3: Pole-Zero Map of the overall transfer function G(s)

Besides, the singular values of the linearized system can be analysed so as to understand dynamic constraint of the QTP better. For a singular value σ and corresponding singular vectors u and v of a rectangular matrix A, define

$$\begin{aligned} Av &= \sigma u \\ A^H u &= \sigma v \end{aligned} \tag{1.27}$$

where A^H is the Hermitian transpose of A. Two orthogonal matrices U and V are generated by the corresponding singular vectors, and the following equations are obtained.

$$\begin{aligned} AV &= U\Sigma \\ A^H U &= V\Sigma \end{aligned} \tag{1.28}$$

where the diagonal entries of Σ are equal to the singular values of the matrix A. By multiplying the first equation by V^H on the right, the singular value decomposition equation is acquired.

$$A = U\Sigma V^H \tag{1.29}$$

and by means of MATLAB, considering the previous linearization point, Σ is found as

$$\Sigma = \begin{bmatrix} 0.0083 & 0 & 0 & 0 \\ 0 & 0.0078 & 0 & 0 \\ 0 & 0 & 0.0033 & 0 \\ 0 & 0 & 0 & 0.0032 \end{bmatrix} \tag{1.30}$$

The gain of the MIMO system at a given frequency ω can be defined as [23]

$$\frac{||Y(jw)||_2}{||U(jw)||_2} = \frac{||G(jw)U(jw)||_2}{||U(jw)||_2} \tag{1.31}$$

In order to estimate the value of the gain at different values of ω, equation 1.32 is written

$$\underline{\sigma}(G(jw)) \leq \frac{||G(jw)U(jw)||_2}{||U(jw)||_2} \leq \overline{\sigma}(G(jw)) \tag{1.32}$$

where $\underline{\sigma}(G(jw))$ and $\overline{\sigma}(G(jw))$ are the principal gains at that frequency. It is easier to control a system if the ratio between the maximum and minimum principal gains, which is so-called condition number, is close to 1.

The value of the principal gains (minimum and maximum singular values) is shown in Figure 1.4.

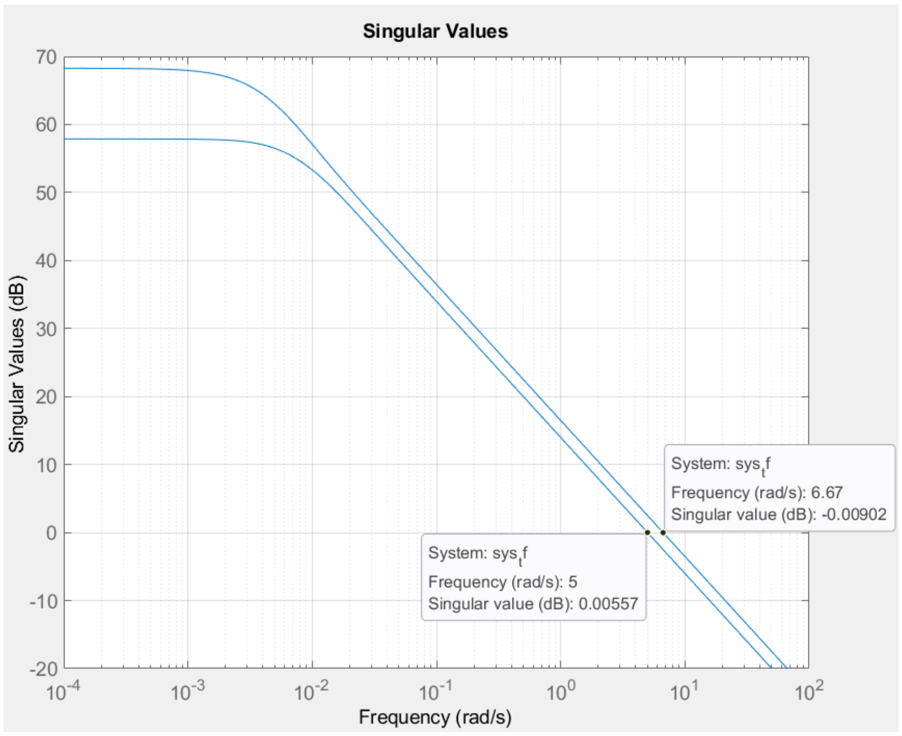

Figure 1.4: Principal gains of the system

The amplification is ensured for any input signal in the frequency band (-∞, 5]. Likewise, the attenuation is guaranteed for any input signal in the frequency band [6.67, ∞).

1.2 Plant Simulator in MATLAB/Simulink

The simulator of the plant is implemented in MATLAB/Simulink by using the equations in 1.1 and the parameters in Table 1.1. These equations and parameters are written into a MATLAB script. The script constitutes derivative of the states by using the manipulated variables q_a and q_b. It is important to mention that the states h_3 and h_4 are not measurable practically; hence, these states must be estimated in the control design step.

In this plant simulator, the first objective is to collect the outputs h_1 and h_2 under stationary operating conditions. In the following chapters, the simulator is modified for the identification and control design of the system.

The scheme of the plant can be seen in Figure 1.5. In order to make simulations, the plant simulator is designed as a subsystem. Simulation time is selected as 20000 seconds.

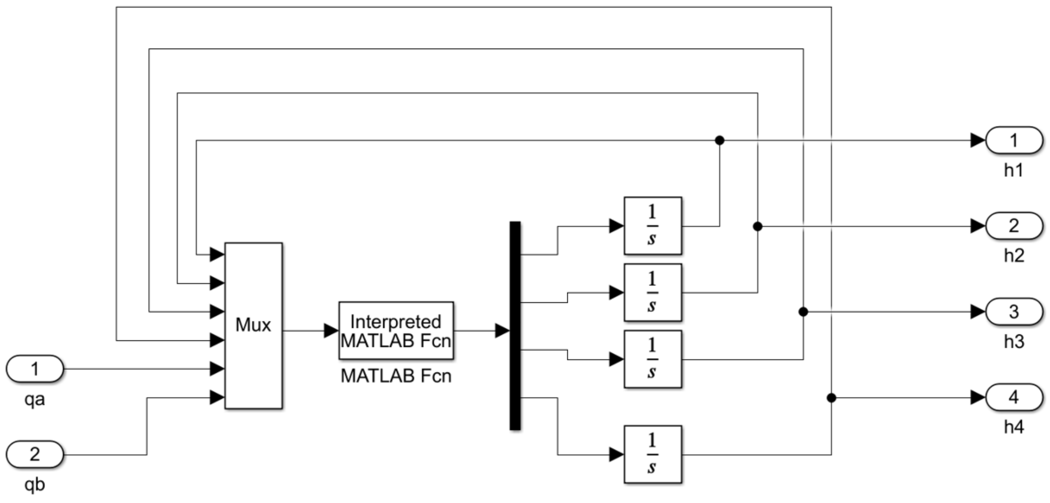

Figure 1.5: Simulator of the plant

The open loop plant is simulated on the purpose of computing the steady-state values of states h_1, h_2, h_3 and h_4. It is supposed to get nominal output values when the inputs with nominal conditions are applied as a constant at the time instant zero. The operating conditions can be seen in Table 1.1.

For a stationary operating points, the differential equations in 1.1 can be written as

$$\begin{aligned} \frac{a_3}{S}\sqrt{2gh_3^\circ} &= \frac{(1-\gamma_b)}{S}q_b^\circ \\ \frac{a_4}{S}\sqrt{2gh_4^\circ} &= \frac{(1-\gamma_a)}{S}q_a^\circ \end{aligned} \tag{1.33}$$

and thus

$$\begin{aligned} \frac{a_1}{S}\sqrt{2gh_1^\circ} &= \frac{\gamma_a}{S}q_a^\circ + \frac{(1-\gamma_b)}{S}q_b^\circ \\ \frac{a_2}{S}\sqrt{2gh_2^\circ} &= \frac{(1-\gamma_a)}{S}q_a^\circ + \frac{\gamma_b}{S}q_b^\circ \end{aligned} \tag{1.34}$$

Then, there exists a unique constant input (q_a°, q_b°) that supplies the steady-state levels of the outputs (h_1°, h_2°) provided that the following matrix

$$\begin{bmatrix} \gamma_a & (1-\gamma_b) \\ (1-\gamma_a) & \gamma_b \end{bmatrix} \tag{1.35}$$

is invertible, i.e., if and only if $\gamma_a + \gamma_b \neq 1$. In the case that $\gamma_a + \gamma_b = 1$, the transfer function matrix G(s) has an invariant zero at the origin, so there would be a derivative action.

For $\gamma_a = 0.3$ and $\gamma_b = 0.4$, the determinant of the matrix (eq. 1.35) is not equal to zero; therefore, the matrix is non-singular.

The obtained steady-state values are close to the nominal ones and transients of the states can be seen in Figure 1.6. The steady state values are

$$\overline{h}_1 = 0.6536,\ \overline{h}_2 = 0.6498,\ \overline{h}_3 = 0.6590,\ \overline{h}_4 = 0.6582 \tag{1.36}$$

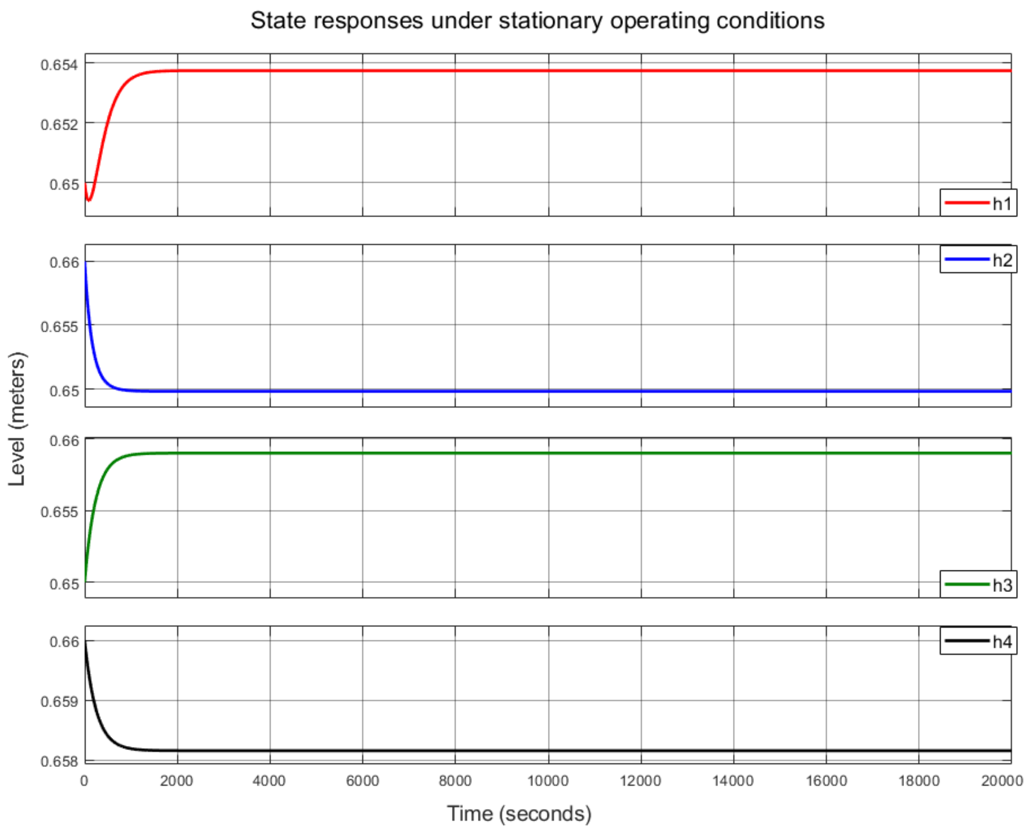

Figure 1.6: Transient responses and steady-state values for h_1, h_2, h_3 and h_4

1.3 Summary

In Section 1.1, the Quadruple-Tank Process and its working principle are described. Nonlinear model is stated with the help of equations, parameters and diagrams. Afterwards, the linearization of the system is specified. State-space representation and the corresponding transfer function matrix are determined so as to understand the main characteristics of the system including multivariable poles, invariant zeros and singular values. Furthermore, minimum and nonminimum phase modes are mentioned with regard to invariant zero location.

In Section 1.2, the simulator of the plant is explained and its scheme is shown. Then, the differential equations are represented for stationary operating conditions and invertibility of a matrix related to operating points is discussed. At the end, steady-state values and the transient responses are presented with figures.

Chapter 2

Linear Model Identification

Although, in Chapter 1, it has been pointed out that the Quadruple-Tank Process has nonlinear dynamics, linear models can be employed first for identifying the linearized system model. Linear models cannot perform as well as nonlinear models; nevertheless, it is good to realize how much a identified linear model could estimate the nonlinear system in the overall set of operating conditions.

In this chapter, three different linear models are built. First, the structures of the models are presented in each section. Then, several identification tests are performed and performance of the identified models are evaluated. After determining the candidate linear models based on estimation results, validation of these models are made. Additional tests are done in the validation step in order to acquire more reliable outcome. Last but not least, each selected state-space and input-output polynomial model is compared so as to determine which one estimates the nonlinear QTP the most.

Excitation Signal and Its Implementation

Input data has to be selected properly in order to collect output data representing the system's dynamics. It is possible to use various type of signals to identify the nonlinear model including square, chirp, sinusoidal, Gaussian white noise, pseudorandom binary sequence (PRBS).

For linear systems, a white noise (WN) is an ideal signal to excite the system because it excites all modes. However, using this kind of signal is difficult in practice. In order to meet the requirements of excitation, a PRBS signal might be a good alternative to white noise. A PRBS is a deterministic signal and it is normally periodic with a maximum period length of 2^n-1, where the integer n is the order of the PRBS. In Figure 2.1, illustration of a periodic PRBS input signal can be seen.

In this work, it has been decided to generate uncorrelated PRBS signals with also varying amplitude for two inputs q_a and q_b so as to emphasize the nonlinear behaviour of the system on the output. Maximum and minimum amplitudes of these inputs have to be close to values of demanded operating range. Choosing the sampling time is also very important because if it is too short, then the result would be considerably affected by additional high frequency noise. Also, it should not be selected very long since it may lead to aliasing problems. By taking into consideration these requirements and the dynamics of the Quadruple-Tank system,

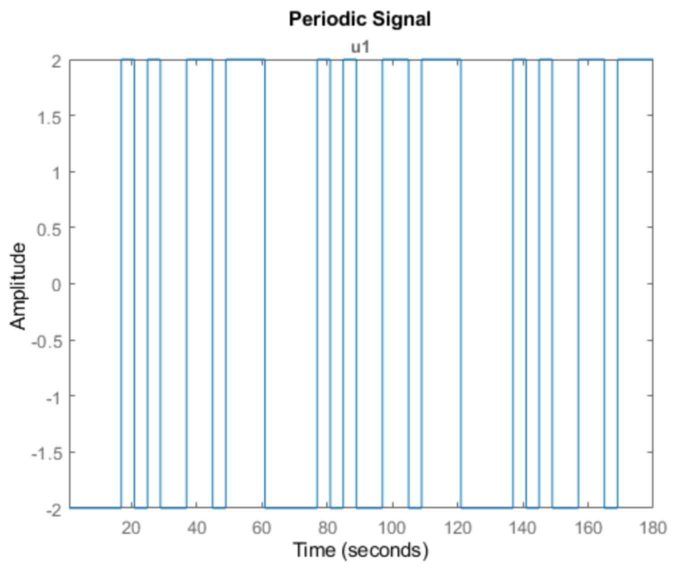

Figure 2.1: Example of a periodic PRBS input signal [3]

the sampling time has been chosen as 25 seconds.

The generated input signals for the identification can be seen in Figure 2.2. For acquisition of these signals, a MATLAB script has been created (see Appendix B).

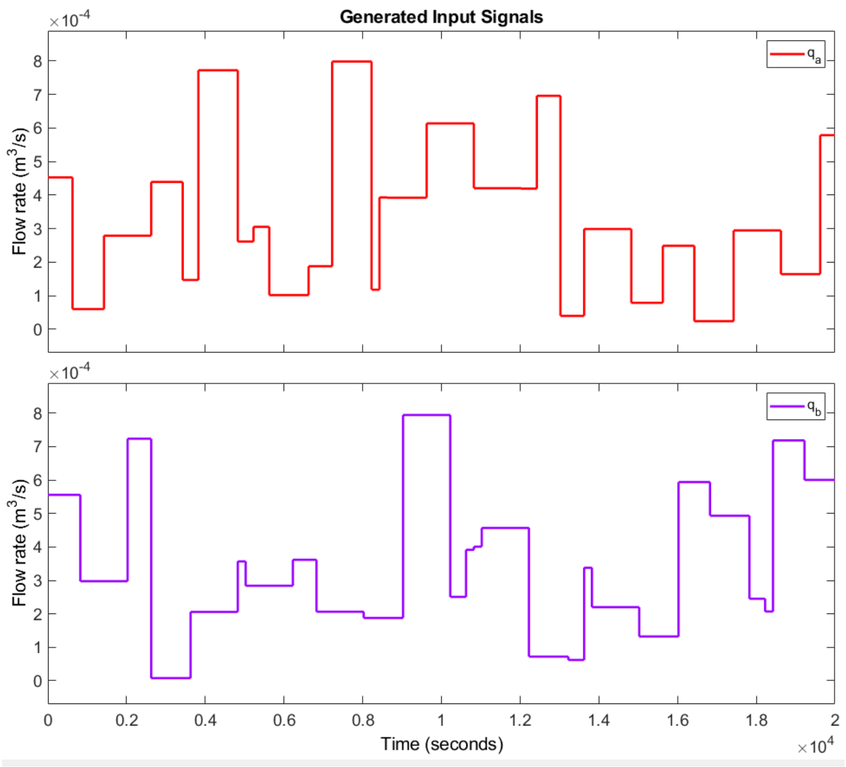

Figure 2.2: Asymmetric PRBS signals for the inputs q_a and q_b

Data Pre-processing and Estimation Mode

The asymmetric PRBS input signals shown in Figure 2.2 and the associated outputs must be splitted into two datasets; one is used for estimating the parameters and the other one is employed in order to validate the estimated model by evaluating the performance based on NRMSE criterion (see eq. 2.2). For the identification tests performed in the following sections, first 75% of the input and the output data are selected for estimation of a linear model and the remaining 25% is utilized as a validation dataset.

For linear models, normalizing the datasets, if possible, before starting estimation of a model is always suggested. Normalization of the inputs and outputs is a part of data pre-processing. For this purpose, the data sets are detrended i.e. mean value of each data is calculated and these biases are subtracted from each time-domain signal. Thus, the identified linear models can be considered as reliable in all frequency range because best-fit linear trends for which the linear models are sensitive are removed.

Two different modes can be used for error minimization. One is called as prediction mode which minimizes the 1 or k-step ahead prediction error between measured and predicted outputs during estimation. This mode uses the current and the past values of observed input and output values, as well as initial conditions. Another one is simulation mode and it minimizes the simulation error between measured and simulated outputs. Unlike the prediction, simulation mode does not use the measured output values for estimation; instead, it calculates the outputs and uses them to demonstrate the behaviour of the plant. This refers that simulation mode is more reliable and robust with compared to prediction one; hence, all linear models are calibrated in simulation mode despite the fact that it requires more computational time.

Input-Output Delay

The input-output delay of a model is crucial for the system identification; therefore, the delay for each input-output pairs has to be determined. The input-output delay is also known as dead time of the linear model. There are many ways to acquire the dead time; for example, it is possible to it by estimating the non-parametric impulse response or by looking at the step response or by using N4SID based state-space evaluation etc [3].

Numerical algorithm for the subspace state-space system identification (N4SID) method is used to estimate a linear time invariant state-space model via the non-iterative subspace method with a input-output data represented in time or frequency domain [24]. The N4SID uses developed concepts and algorithms from numerical linear algebra, such as QR-decomposition and singular value decomposition. Since the N4SID is a fast algorithm, a range of orders can be tried quickly. In this thesis, the N4SID is employed with a number of different orders to find the input-output delay of the best (proposed) model. To this end, the MATLAB System Identification Toolbox is utilized to estimate the state-space models with different orders (from 1 to 10) simultaneously from the detrended estimation data. The proposed order is 8 by the N4SID; hence, the state-space model with order of 8 is selected so as to analyze its impulse response. The logarithm of singular values related to model orders is shown in Figure 2.3. These singular values represent the harmonics in the signal. For example, the harmonics for the 8th order model is more valuable than the 9th order one according to this figure.

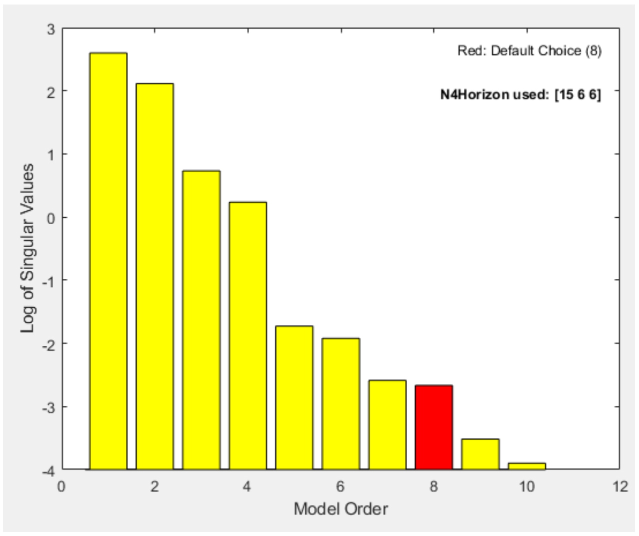

Figure 2.3: Proposed order for a state space model by N4SID method

Moreover, the confidence level is selected as 3σ, where 3σ is equal to 99.7% significance, so as to decide how many delay samples there are from inputs to outputs. The obtained impulse responses from each input to each output are demonstrated in Figure 2.4. The areas (regions) shown in this figure indicate the confidence interval

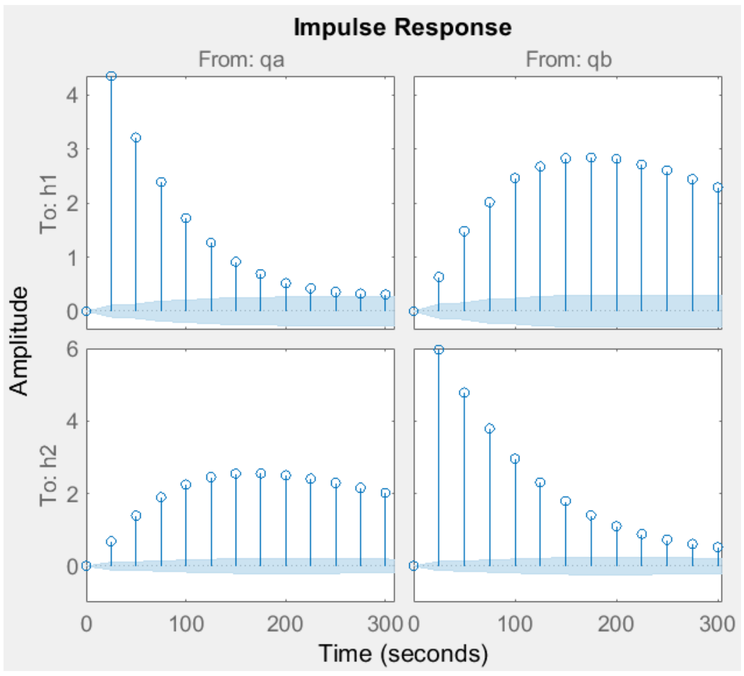

Figure 2.4: Impulse responses from inputs to outputs with 3σ confidence level

for the unimportant responses in the estimation. The amplitude of the impulse response exceeds these areas for all input-output pairs after just one sample time

(25 seconds); that is to say, the transport delay or dead time is equal to one sample time. Hence, the input-output delay for all the corresponding pairs is chosen as 1 and n_k will be 1 during the identification of all models in the following sections.

Models' Choice and Performance Criterion

The linear models implemented in the following sections are State-Space, Autoregressive Moving Average Model with Exogenous Inputs (ARMAX), Output-Error (OE), and ARX models.

In this chapter, normalized root-mean-square error (NRMSE) quality metric is used to evaluate performance of all the estimated models. The following root-mean-square error equation, which is sometimes called loss function, is related to simulation error minimization (SEM) algorithm [3] [24]. The SEM algorithm updates the parameters of an initial model to fit the estimation data.

$$RMSE = \sqrt{\frac{\sum_{t=1}^{N}(y_{obs}(t) - y_{sim}(t))^2}{N}} \tag{2.1}$$

where y_{obs} is the measured output data, y_{sim} is the simulated response of the model. Different NRMSE equations can be used for the calculation of estimation fit; in this work, the NRMSE choice is

$$NRMSE = \frac{||y_{obs} - y_{sim}||_2}{||y_{obs} - \overline{y_{obs}}||_2} \tag{2.2}$$

where $\overline{y_{obs}}$ refers to the mean value of the observed output [3]. Then, fit percentage can be written as

$$Fit\% = 100(1 - NRMSE) \tag{2.3}$$

and its range is (-∞, 100].

2.1 State-Space Models

A state-space model is a mathematical representation of a physical system as a set of input, output, and state variables related by first-order differential equations [3]. In a state space model, the output variables are originated from the state variables.

Starting a linear model identification through a state-space model is a good idea because it provides a quick estimation by requesting only one input which is the model order n. State-space models can also be represented in discrete time but only the continuous state state models are estimated in this thesis since the discrete-time state space model is similar to the ARMAX model which is explained in Section 2.2.

2.1.1 Model Structure

In continuous time, the state-space representation is

$$\begin{aligned} \dot{x}(t) &= Fx(t) + Gu(t) + Ke(t) \\ y(t) &= Hx(t) + Du(t) + e(t) \end{aligned} \tag{2.4}$$

where F is the state matrix, G is the input-to-state matrix, H is the state-to-output matrix, D is the feedthrough matrix, K is the matrix including noise component parameters and $x(t)$, $u(t)$, $y(t)$ and $e(t)$ represent the states, inputs, outputs and disturbances respectively [3]. The initial state vector $x(0)$ is required in order to estimate a state space model. The initial states can be computed by means of the MATLAB System Identification toolbox.

2.1.2 Identification

First, the performance of the detrended data used for estimation of state space models is evaluated according to NRMSE criterion so as to have an idea on the model orders. Recall that the delay for each input is selected as 1, and it is kept constant for all orders. Continuous time state space models are estimated from order 1 to order 9.

n	n_k	Fit% $[h_1;\ h_2]$
1	1	[60.16; 64.24]
2	1	[69.92; 78.69]
3	1	[74.26; 84.43]
4	1	[75.15; 85.58]
5	1	[74.39; 84.52]
6	1	[73.87; 83.64]
7	1	[73.48; 83.81]
8	1	[45.67; 78.65]
9	1	[75.20; 85.63]

Table 2.1: Fit% of different State-Space models on the estimation dataset

From Table 2.1, it is seen that the ninth, fourth, third and fifth order state space models fit the estimation data the most in the simulation mode. However, deciding the order of a model just by looking at the NRMSE results based on identification dataset is meaningless since the models with higher orders might over-fit the corresponding model. In order to avoid this, the identified models need to be tested using the validation dataset and the implementation of validation is reported in Section 2.1.3. All the estimations have been made using prediction error minimization method with simulation focus and the maximum number of iterations is selected as 50.

2.1.3 Validation

The estimated state space models with the order from 1 to 9 are validated by employing different data set and the obtained results can be seen in Table 2.2. Input dead time n_k is also set to 2 for the models which provide better fit on the validation data in simulation mode in order to be sure that the previous choice of the delay n_k is correct.

n	n_k	Fit% $[h_1; h_2]$
1	1	[67.21; 29.93]
2	1	[65.09; 54.97]
3	1	[70.86; 52.82]
4	**1**	**[77.20; 59.25]**
5	1	[80.77; 59.14]
6	1	[68.81; 48.55]
7	1	[74.52; 49.37]
8	1	[58.97; 42.80]
9	1	[78.12; 60.65]
3	2	[69.64; 52.72]
4	2	[73.15; 52.50]
5	2	[79.54; 63.34]
9	2	[73.77; 57.13]

Table 2.2: Fit% of the estimated State-Space models vs. validation data

According to Table 2.2, the state space models with order 4, 5 and 9 give the best fit percent based on NRMSE criterion. Owing to the fact that there is not a remarkable performance improvement between the order 9 and 5, the former is eliminated to get rid of unnecessary complexity. Furthermore, performance of the state space models decrease when the input-output delay n_k is assigned to 2 except the one with order 5. Its fitting percentage for the first output h_1 becomes less about 1 percent, but the performance for the second output h_2 increases approximately 4 percent. For this reason, the auto-correlation and cross-correlation of residuals for the inputs and outputs will be presented below.

Auto-correlation is a mathematical tool that estimates the correlation between observations as a function of the time lag (difference in samples) between them. The auto-correlation function measures the correlation between y_t and y_{t+k}, where

$k = 0, 1, ..., K$. Then, the auto-correlation for lag k is [25]

$$r_k = \frac{c_k}{c_0} \tag{2.5}$$

where c_0 is the sample variance and the auto-covariance function c_k is

$$c_k = \frac{1}{T} \sum_{t=1}^{T-k} (y_t - \overline{y})(y_{t+k} - \overline{y}) \tag{2.6}$$

where $\overline{y}$ is the mean of a time series y_1, ..., y_n, T is the sample size of the vector $\boldsymbol{y}$ and estimated standard error of the auto-correlation at lag $k > q$ is

$$SE(r_k) = \sqrt{\frac{1}{T}(1 + 2\sum_{j=1}^{q} r_j^2)} \tag{2.7}$$

where q is the lag coefficient beyond which the theoretical auto-correlation function is zero.

By defining the residuals as the differences between the simulated output and the measured output from the validation data set [3], the auto-correlation of the residuals for the outputs can be expressed. It is so-called whiteness test. It is also required to define confidence bounds. In this work, the confidence interval is selected as ± 3 standard errors of a Gaussian distribution in order to evaluate the performance of auto-correlation functions and if all residual values are inside this interval, then it means that the correlations are statistically unimportant and the linear model is estimated well. For all lags are greater than the number of lags in the theoretical model of the vector $\boldsymbol{y}$, the confidence bounds are $0 \pm 3\hat{\sigma}$ where $\hat{\sigma}$ is the estimated standard error of the sample auto-correlation. By contrasting the residuals with the confidence limit, it can be decided whether the residuals are white noise or not. If the residuals are white; that is, the residual auto-correlation function inside the confidence interval of the corresponding estimates, then the associated state-space model can be considered as a good model.

There is another residual analysis related to cross-correlations between the input and the residuals for each input-output pair. This is also known as independence test [3]. For a model being considered as sufficient, the residuals must be uncorrelated with past inputs. For example, the identified linear model would not be a good model in the case that the signal exceeds the confidence limits for lag k and it refers that the output produced by the input $u(t - k)$ is not properly described by the identified model. The cross-correlation function estimates the sample cross-covariance function $c_{y_1 y_2}$ for the time series y_{1t} and y_{2t} and the equation for estimating cross-covariance with the lag k is

$$c_{y_1 y_2}(k) = \begin{cases} \frac{1}{T} \sum_{t=1}^{T-k} (y_{1t} - \overline{y_1})(y_{2,t+k} - \overline{y_2}) & k = 0, 1, 2, ... \\ \frac{1}{T} \sum_{t=1}^{T+k} (y_{2t} - \overline{y_2})(y_{1,t-k} - \overline{y_1}) & k = 0, -1, -2, ... \end{cases} \tag{2.8}$$

where $\overline{y_1}$ and $\overline{y_2}$ are the sample means [25]. The cross-correlation function is

$$r_{y_1y_2}(k) = \frac{c_{y_1y_2}(k)}{s_{y_1}s_{y_2}} \tag{2.9}$$

where s_{y_1} and s_{y_2} are the sample standard deviations i.e. the square root of the variance of y_1 and y_2.

In order to compare the residuals of the identified state space models with $n = 4$ & $n_k = 1$, $n = 5$ & $n_k = 1$ and $n = 5$ & $n_k = 2$ on the validation date set, the auto-correlation and cross-correlation figures are used. As it can be seen in

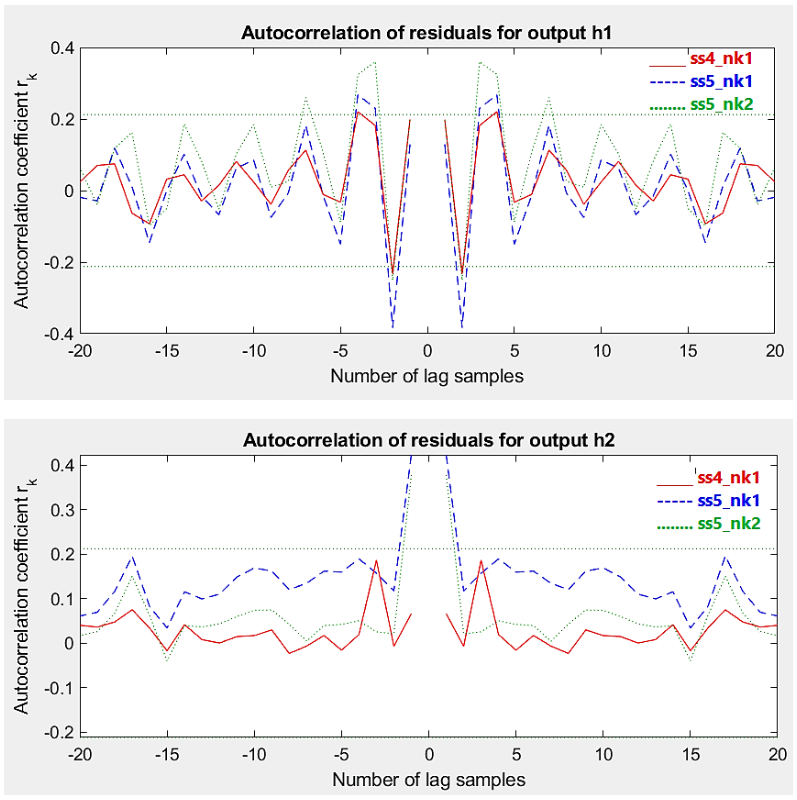

Figure 2.5: Autocorrelation of the residuals for the outputs

Figure 2.5, there is only one state space model which passes the whiteness test and it is the fourth order model with a delay of one sample time due to the fact that the fluctuations in the other models go beyond the $\pm 3\sigma$ confidence interval.

After implementing the whiteness test, both estimated linear models with the order of 5 are eliminated. Hence, the independence test is only performed for the linear model with fourth order. From Figure 2.6, it is observed that all residuals are uncorrelated with past inputs because there is no peak outside the confidence interval for any lag k, i.e. past inputs $u(t-k)$ contribute to the output $y(t)$ and this contribution is well-defined by the identified linear model. This means that the corresponding model passes the independence test as well.

In the light of the above findings, the 4th order state space model with 1 input-output delay seems as the sanest choice. The comparison of the estimated and the

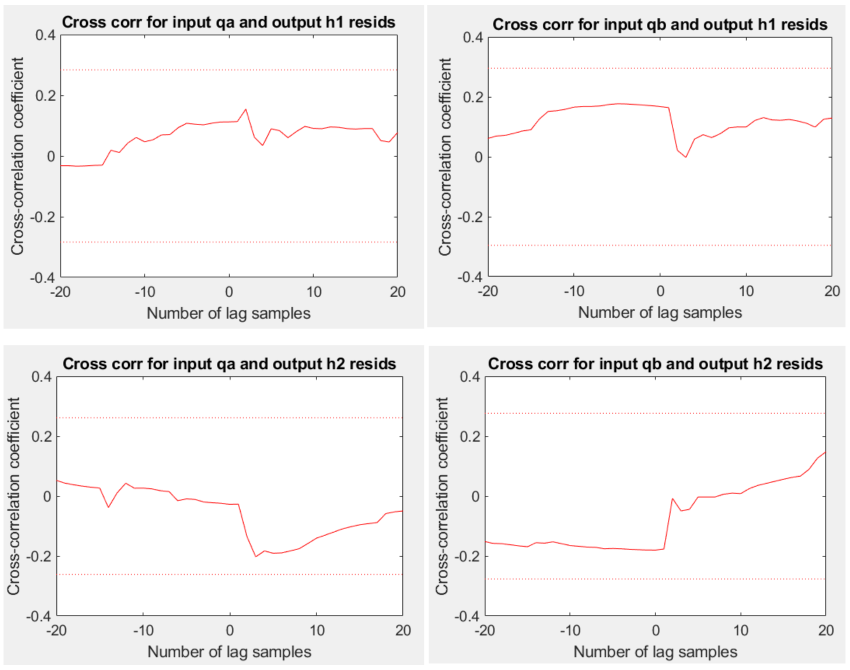

Figure 2.6: Cross-correlation for the output residuals with associated inputs

plant models can be seen in Figure 2.7.

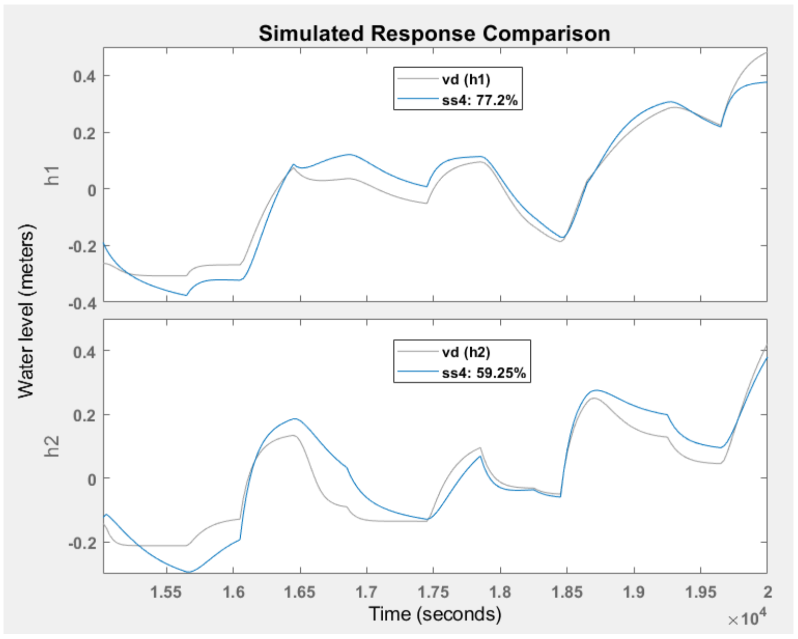

Figure 2.7: The fourth order state space model on the validation data

Even though all results imply to choose the fourth order state-space model with a delay of one sample time, additional tests have been performed in order to get more reliable results. The additional test set has been constituted by merging four different validation data sets. The PRBS input signals with also varying amplitude used for these tests are demonstrated in Figure 2.8.

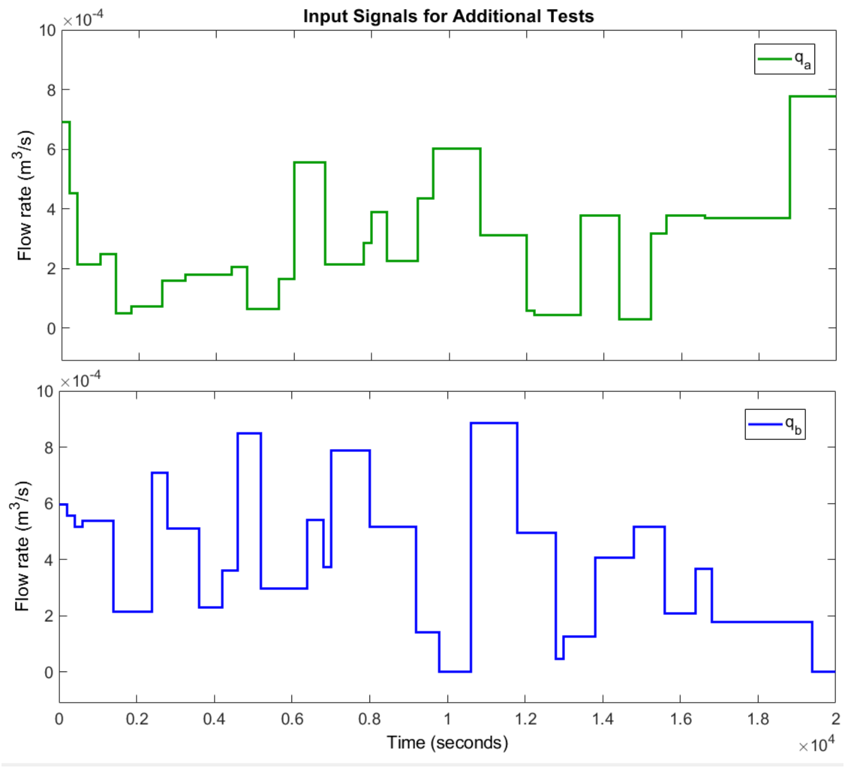

Figure 2.8: Asymmetric PRBS inputs for additional tests

These signals are detrended i.e. their mean value or offset is removed to acquire robust results while estimating linear state space models. The additional tests are made in the simulation mode and the obtained results are shown in Table 2.3.

n	n_k	**Fit% [h_1; h_2]**
1	1	[58.06; 50.80]
2	1	[75.70; 67.65]
3	1	[79.90; 73.07]
4	**1**	**[78.71; 71.53]**
5	1	[79.71; 73.26]
6	1	[79.54; 70.54]
7	1	[77.87; 71.47]
8	1	[54.88; 66.94]
9	1	[78.87; 71.68]

Table 2.3: Average Fit% of different State-Space models on the test datasets

According to the fit percentages on the test data set, the state space model with order 4 and delay 1 can be selected with peace in mind. By recalling equation 2.4, the continuous time identified state space model with the order of four and delay of one is represented as

$$\begin{aligned} \frac{dx}{dt} &= A_{id}x(t) + B_{id}u(t) + K_{id}e(t) \\ y(t) &= C_{id}x(t) + D_{id}u(t) + e(t) \end{aligned} \tag{2.10}$$

where $D_{id} = 0$ and the state-space matrices A_{id}, B_{id}, C_{id} and K_{id} are

$$\begin{aligned} A_{id} &= \begin{bmatrix} -0.008589 & -0.001322 & +0.000563 & -0.004028 \\ +0.000595 & -0.003123 & +0.000114 & +0.000131 \\ +0.010760 & +0.001627 & -0.005023 & +0.006338 \\ -0.002870 & +0.000884 & -0.001245 & -0.001621 \end{bmatrix} \\ B_{id} &= \begin{bmatrix} -2.5360 & +3.1350 \\ -0.7504 & -0.9623 \\ +3.4400 & -4.3180 \\ -0.6488 & +1.3010 \end{bmatrix} \\ C_{id} &= \begin{bmatrix} +1.3230 & -3.3600 & +1.9230 & +0.6459 \\ -0.9154 & -3.9720 & -1.4090 & +0.3498 \end{bmatrix} \\ K_{id} &= \begin{bmatrix} -0.2937 & -0.4585 \\ +0.0987 & +0.0762 \\ +0.1204 & +0.2132 \\ +0.8350 & +0.6990 \end{bmatrix} \end{aligned} \tag{2.11}$$

where K_{id} have the coefficients of the disturbance $e(t)$. The zeros and the poles of the considered state space model are

$$\begin{aligned} z_1 &= -0.0005, \quad z_2 = +0.0042 \\ p_1 &= -0.0098, \quad p_2 = -0.0048 \\ p_3 &= -0.0031, \quad p_4 = -0.0006 \end{aligned} \tag{2.12}$$

Needless to mention, these zeros and poles are different than the ones obtained in Chapter 1. However, the structure is similar i.e. all the poles are in left-half-plane and there is one invariant zero in the right-half-plane for both state space models. Thus, the continuous time identified state space model is nonminimum phase as well.

In conclusion, it is significant to note that although the identified fourth order state space model has been considered the best one for fitting the validation data among all, its fit percent regarding to NRMSE is approximately just 77% for the first output h_1 and 59% for the second output h_2 owing to the nonlinear dynamics of the plant.

2.2 ARMAX Models

The ARMAX polynomial models are widely used in the process industry. An Auto-regressive Moving Average Model with Exogenous Inputs (ARMAX) model is composed of an auto-regressive (AR), a moving average (MA) and an exogenous input (X) classes. The moving average corresponds to weighted average of white noise. It is called "moving" because the window moves during the identification process according to order of the MA part. Moreover, a stochastic process is considered as an auto-regressive (recursive) process in the event that the combination of old output values and present input is utilized so as to estimate the current output. The exogenous input is also included in order to make the system identification robust with respect to noise i.e. the noise dynamics are also taken into account in addition to the input-output system dynamics [4].

2.2.1 Model Structure

For a SISO system, the structure of an ARMAX model is

$$A(q)y(k) = B(q)u(k - n_k) + C(q)e(k) \tag{2.13}$$

where $u(k)$ and $y(k)$ are the input and the output values at sampling time k, $e(k)$ is the noise (error) term and n_k is the number of input samples that occur before the input affects the output. The variables $A(q)$, $B(q)$ and $C(q)$ are the polynomials stated in the time-shift operator q^{-1} and this operator is represented as [3] [24]

$$q^{-1}u(k) = u(k - T) \tag{2.14}$$

where T is the sampling time. Thus, the polynomials $A(q)$, $B(q)$ and $C(q)$ are

$$\begin{aligned} A(q) &= 1 + a_1 q^{-1} + ... + a_{n_a} q^{-n_a} \\ B(q) &= b_1 + b_2 q^{-1} + ... + b_{n_b} q^{-n_b+1} \\ C(q) &= 1 + c_1 q^{-1} + ... + c_{n_c} q^{-n_c} \end{aligned} \tag{2.15}$$

where n_a, n_b and n_c represent the number of poles, the number of zeros plus 1 and the number of $C(q)$ coefficients respectively. In equation 2.15, the first n_k coefficients of $B(q)$ is zero when the dynamics from $u(k)$ to $y(k)$ constitute a delay of n_k samples.

In the ARMAX model, $A(q)$ represents the common poles for the dynamic model and the noise model. Therefore, the system's dynamics $A(q)$ affects the noise term $e(k)$ at which the noise takes place in the system in early times [26]. The noise is also influenced by the polynomial of $C(q)$ which provides more flexibility for modeling the noise using its parameters. The diagram of an ARMAX structure is shown in Figure 2.9. In this figure, $y^*(k)$ is the noise-free output measurements.

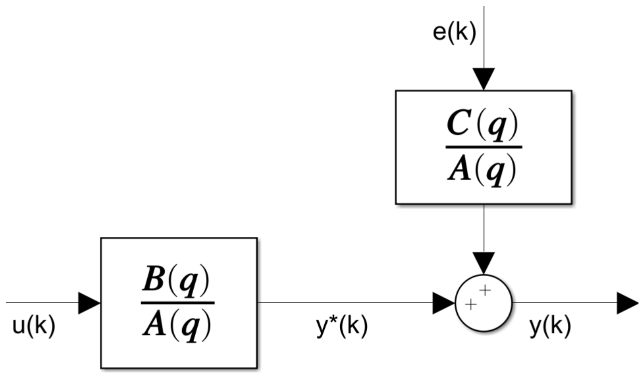

Figure 2.9: Scheme of an ARMAX model in the SISO case [4]

The equation 2.13 can be generalized for multi-input and/or multi-output systems. The generalized structure of an ARMAX model is

$$\begin{aligned} y(k) =& A_1 y(k-1) + ... + A_{n_a} y(k-n_a) + B_1 u(k-n_k) + ... \\ &+ B_{n_b} u(k-n_b-n_k+1) + e(k) + C_1 e(k-1) + ... + C_{n_c} e(k-n_c) \end{aligned} \tag{2.16}$$

where n_u and n_y are the number of inputs and outputs of the system respectively, A_1, A_2, ... $\in \mathbb{R}^{n_y,n_y}$, B_1, B_2, ... $\in \mathbb{R}^{n_y,n_u}$ and C_1, C_2, ... $\mathbb{R}^{n_y,1}$ are the coefficient matrices, $u(k) \in \mathbb{R}^{n_u}$ represents the inputs at sampling time instant k, $y(k) \in \mathbb{R}^{n_y}$ corresponds to the outputs at sampling time instant k, n_a, n_b, and n_c are the orders of the ARMAX model [3] [24].

2.2.2 Identification

The detrended estimation and validation data sets are employed for estimating ARMAX models. The input-output delay has been determined in the previous section as $n_k = 1$ for all the transfer functions of the system and it is kept constant during the estimation. A set of possible combinations of n_a, n_b and n_c is tried iteratively. More specifically, the ARMAX models with the orders $n_a = [i\ 0; 0\ i]$, $n_b = [j\ j; j\ j]$, and $n_c = [m;\ m]$ are estimated where $i,\ j,\ m\ = 1, 2, ..., 7$. In addition to this, some other models are considered in Section 2.2.3 with different orders for the sake of completeness.

For the estimated models with varying orders in the simulation mode, the Akaike Information Criterion (AIC) provides a measure of model quality of each estimated model [24]. Then, the estimated models can be compared to each other. Theoretically, the most preferable model is the one with the smallest AIC. Nonetheless, models' performance evaluation based on NRMSE criterion, whiteness test and independence test has to be made on the validation data set in addition to AIC for selecting a proper model. The AIC can also be represented in normalized form. The formulation of the normalized AIC is

$$nAIC = log\left(det\left(\frac{1}{N}\sum_{t=1}^{N}\epsilon(t,\hat{\theta}_N)(\epsilon(t,\hat{\theta}_N))^T\right)\right) + \frac{2n_p}{N} \tag{2.17}$$

where N is the number of values in the estimation data set, $\epsilon(t)$ is the n_y-by-1 vector of simulation errors, θ_N is the estimated parameters, and n_p is the number of estimated parameters [3].

In Table 2.4, the ARMAX models with acceptable estimation performance are sorted with regard to the normalized Akaike's Information Criterion (nAIC) since this criterion might give a clue for ARMAX models' whiteness test failure on the validation data set [27]. More clearly, the range of nAIC value for all the estimated models is [-23.58, -10.30] and the residuals of the models with nAIC smaller than -19 always go over the confidence limits of auto-correlation function at many lag samples with large amplitudes. In this table, the orders corresponds to arrays; for example, in the first row $n_b = 7$ is equivalent to $n_b = [7\ 7;\ 7\ 7]$, $n_c = 1$ refers to $n_c = [1;\ 1]$ etc.

n_a	n_b	n_c	n_k	Fit% $[h_1;\ h_2]$	$nAIC$
[7 0; 0 7]	7	1	1	[74.81; 85.93]	-10.30
[7 0; 0 7]	5	1	1	[74.57; 85.90]	-10.95
[5 0; 0 5]	5	1	1	[74.48; 85.76]	-10.99
[6 0; 0 6]	5	1	1	[83.43; 69.07]	-11.01
[7 0; 0 7]	5	3	1	[74.78; 86.56]	-11.21
[6 0; 0 6]	5	3	1	[84.22; 72.02]	-11.24
[7 0; 0 7]	5	2	1	[74.78; 86.11]	-11.44
[6 0; 0 6]	5	2	1	[84.18; 70.27]	-11.48
[7 0; 0 7]	7	4	1	[74.62; 86.02]	-11.60
[5 0; 0 5]	4	2	1	[70.36; 86.18]	-12.45
[6 0; 0 6]	4	2	1	[81.58; 70.34]	-12.64
[7 0; 0 7]	4	2	1	[74.34; 86.18]	-12.67
[2 0; 0 2]	5	2	1	[62.50; 83.23]	-12.76
[5 0; 0 5]	4	1	1	[68.81; 85.62]	-12.79
[6 0; 0 6]	4	1	1	[80.30; 70.12]	-12.83
[7 0; 0 7]	4	1	1	[71.77; 85.77]	-12.84
[7 0; 0 7]	7	3	1	[74.51; 85.49]	-12.86
[3 0; 0 3]	4	1	1	[66.74; 80.75]	-13.22
[4 0; 0 4]	3	1	1	[67.07; 82.62]	-13.35
[4 0; 0 4]	4	1	1	[65.56; 85.54]	-13.44

Table 2.4: Fit% and nAIC of the estimated ARMAX models with different orders

From Table 2.4, it is seen that the complexity of the models in the last 3 rows is less than the others because these three models identify the system by using less parameters; for example, the ARMAX model with the orders $n_a = [4\ 0;\ 0\ 4]$, $n_b = [3\ 3;\ 3\ 3]$, $n_c = [1;\ 1]$ has 22 parameters. However, the model order selection can only be made in the validation stage.

2.2.3 Validation

The estimated ARMAX models with different orders are validated by using a different data set. In addition to the orders mentioned in the identification phase, some models are validated by assigning the orders randomly, such as n_a = [4 2; 1 3] or n_a = [5 5; 5 5], n_b = [2 3; 3 2], n_c = [2; 1] in order to extend the estimation procedure. Despite the fact that n_k is selected as [1 1; 1 1] in the estimation phase, different n_k orders are also tested in the validation phase for the sake of completeness. The candidate models with good fit percent and negligible violations of the confidence intervals are shown in Table 2.5.

Row	n_a	n_b	n_c	n_k	Valid.%	Tests%
1	[6 0; 0 6]	[4 4; 4 4]	[2; 2]	[1 1; 1 1]	[75.96]	[74.23]
2	**[4 0; 0 4]**	**[4 4; 4 4]**	**[1; 1]**	**[1 1; 1 1]**	**[75.55]**	**[76.66]**
3	[7 0; 0 7]	[4 4; 4 4]	[1; 1]	[1 1; 1 1]	[75.34]	[74.47]
4	[6 0; 0 6]	[4 4; 4 4]	[1; 1]	[1 1; 1 1]	[75.21]	[74.40]
5	[5 0; 0 5]	[4 4; 4 4]	[1; 1]	[1 1; 1 1]	[74.50]	[74.64]
6	[4 0; 0 4]	[2 2; 2 2]	[1; 1]	[0 0; 0 0]	[71.54]	[73.47]
7	[3 0; 0 3]	[2 2; 2 2]	[2; 2]	[0 0; 0 0]	[71.18]	[73.34]
8	[6 6; 6 6]	[3 3; 3 3]	[3; 3]	[2 2; 2 2]	[70.56]	[68.94]
9	[4 0; 0 4]	[2 2; 2 2]	[2; 2]	[0 0; 0 0]	[70.53]	[73.71]
10	[4 0; 0 4]	[2 2; 2 2]	[3; 3]	[0 0; 0 0]	[70.45]	[73.57]
11	[3 3; 3 3]	[4 4; 4 4]	[2; 2]	[2 2; 2 2]	[70.42]	[76.01]
12	[4 4; 4 4]	[2 3; 3 2]	[4; 4]	[0 4; 4 0]	[69.30]	[72.58]
13	[3 3; 4 4]	[2 3; 3 2]	[2; 2]	[0 4; 4 0]	[66.90]	[73.23]
14	[4 4; 4 4]	[4 4; 4 4]	[1; 1]	[1 1; 1 1]	[65.30]	[73.82]

Table 2.5: Fit percentage of the candidate models on the validation and test datasets

Auto-correlation and cross-correlation functions have been explained in Section 2.1. All the models indicated in Table 2.5 violate the confidence bounds of the auto-correlation and/or cross-correlation functions at some lag samples; nevertheless, these are the models that provide good validation performance among hundreds of estimated models and whose confidence intervals are the least violated for the considered dead time n_k.

The minimum violations of the residuals have been observed for the models in the 12th and 13th rows of Table 2.5. However, their performance based on NRMSE criterion are considerably worse as compared to the models on the top of the table. On the other hand, the correlations of the models with $n_k = 2$ significantly violate the confidence limits by comparison with other models, so the corresponding models cannot be chosen in order to identify the plant.

The best choice might be the ARMAX model with the orders n_a = [4 0; 0 4], n_b = [4 4; 4 4], n_c = [1; 1] and n_k = [1 1; 1 1] but additional tests are also included so as to get more reliable results. The PRBS signals in Figure 2.8 have been acquired by merging four uncorrelated test data before. Here, only the second and the third portions of these signals are detrended and utilized for the additional tests. According to average performance of the models on the test data sets in Table 2.5, the model in the second row is chosen. The sum of the numbers in the order matrices n_a, n_b, and n_c gives the number of parameters. Since the orders of the selected model are $n_a = [4\ 0;\ 0\ 4]$, $n_b = [4\ 4;\ 4\ 4]$, $n_c = [1;\ 1]$, the number of coefficients are 26.

The comparison of the selected ARMAX model with the validation data set is demonstrated in Figure 2.10. The fit percent of this model on the validation data

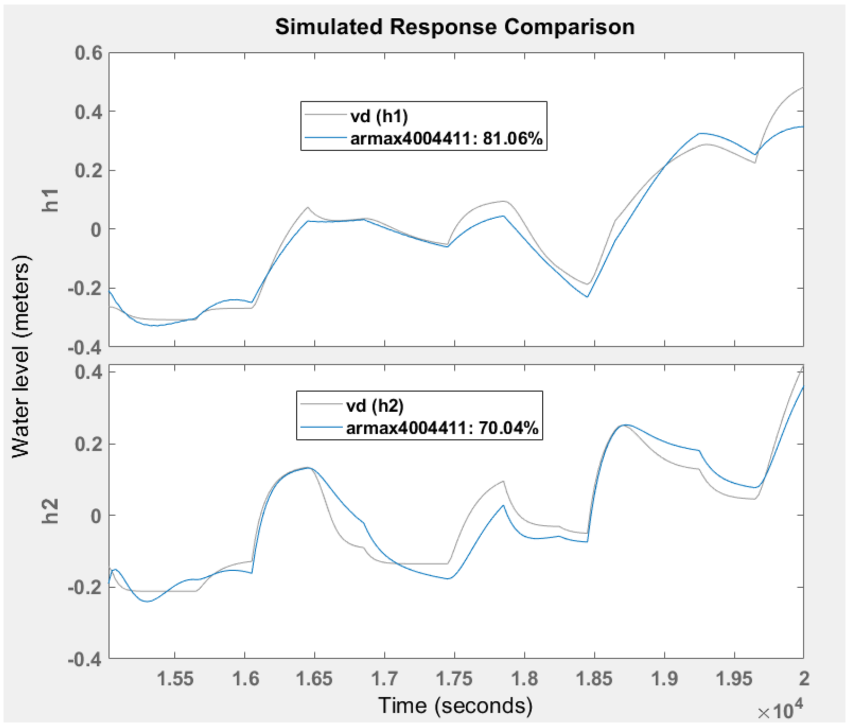

Figure 2.10: The chosen ARMAX model vs. validation data set

set is approximately 81% for the first output h_1 and 70% for the second output h_2 due to the nonlinear dynamics of the system. The performance can be increased by using nonlinear models like NARX. Identification of the Quadruple-Tank Process through nonlinear models is reported in Chapter 3.

The auto-correlations and the cross-correlations of the selected model can be seen in Figure 2.11. The residuals of the considered model exceeds $\pm 3\sigma$ confidence interval of the auto-correlation function at some lag samples close to zero; therefore, it does not pass the whiteness test completely i.e. the selected ARMAX model is not optimal. The utilization of nonlinear models for identification procedure may solve this problem.

By recalling the equation 2.16, the polynomials for the identified ARMAX model with the orders $n_a = [4\ 0;\ 0\ 4]$, $n_b = [4\ 4;\ 4\ 4]$, $n_c = [1;\ 1]$, $n_k = [1\ 1;\ 1\ 1]$ are the following:

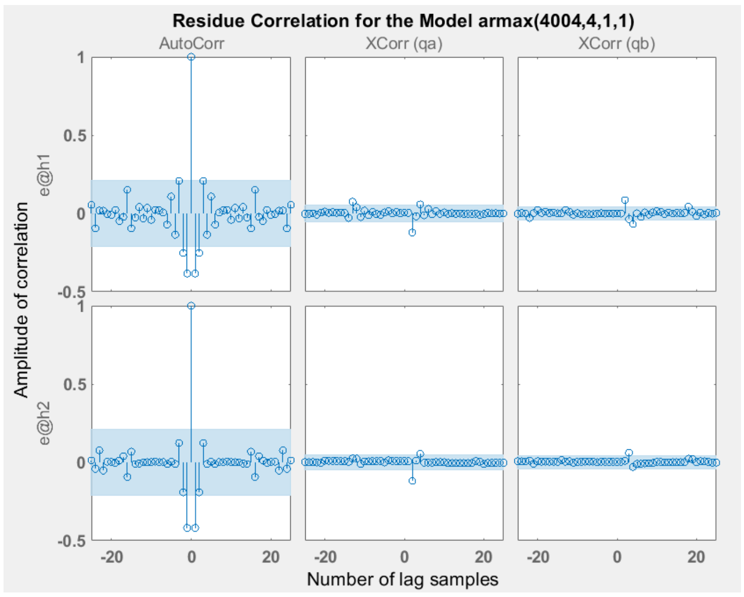

Figure 2.11: Residue correlations of the selected ARMAX model

The polynomials for the first output h_1 are

$$\begin{aligned}
A_{11}(q) &= 1 - 1.9206q^{-1} - 0.0330q^{-2} + 1.8462q^{-3} - 0.8920q^{-4} \\
A_{12}(q) &= 0 \\
B_{11}(q) &= 57.1265q^{-1} - 61.0594q^{-2} - 46.0894q^{-3} + 50.4689q^{-4} \\
B_{12}(q) &= 57.5594q^{-1} - 52.0625q^{-2} - 57.0862q^{-3} + 52.3133q^{-4} \\
C_1(q) &= 1 + 0.6889q^{-1}
\end{aligned} \tag{2.18}$$

and the polynomials for the second output h_2 are

$$\begin{aligned}
A_{21}(q) &= 0 \\
A_{22}(q) &= 1 - 3.5607q^{-1} + 4.7638q^{-2} - 2.8382q^{-3} + 0.6354q^{-4} \\
B_{21}(q) &= 12.8440q^{-1} - 13.4310q^{-2} - 7.8395q^{-3} + 8.7704q^{-4} \\
B_{22}(q) &= 143.6q^{-1} - 394.3q^{-2} + 362.2q^{-3} - 111.3q^{-4} \\
C_2(q) &= 1 + 0.1706q^{-1}
\end{aligned} \tag{2.19}$$

The zeros and the poles of the polynomial $\frac{B_{11}(q)}{A_{11}(q)}$ are

$$\begin{aligned}
\text{Zeros: } &+0.9865 \pm 0.0626i, -0.9042 \\
\text{Poles: } &+0.9710 \pm 0.0769i, -0.9804, +0.9589
\end{aligned} \tag{2.20}$$

and of the polynomial $\frac{B_{22}(q)}{A_{22}(q)}$ are

$$\begin{aligned}
\text{Zeros: } &+0.9233 \pm 0.0956i, +0.8994 \\
\text{Poles: } &+0.9208 \pm 0.0687i, +0.8595 \pm 0.0803i
\end{aligned} \tag{2.21}$$

2.3 Output-Error Models

OE models parameterizes the system's dynamics separately from the stochastic dynamics and they do not use any parameters in order to simulate the noise characteristics. For these models, it is also more important to remove offsets and linear trends as compared to other input-output polynomial models [24] [3].

2.3.1 Model Structure

For SISO systems, the structure of an Output-Error model is the following:

$$y(k) = \frac{B(q)}{F(q)} u(k - n_k) + e(k) \tag{2.22}$$

where $u(k)$ and $y(k)$ are the input and the output values at sampling time k, n_k refers to the input-output delay (dead-time) and $e(k)$ is the noise term. The first n_k coefficients of $B(q)$ become zero in the case that the dynamics from $u(k)$ to $y(k)$ comprise a delay of n_k samples. By recalling the equation 2.14, the variables $B(q)$ and $F(q)$ are

$$\begin{aligned} B(q) &= b_1 + b_2 q^{-1} + ... + b_{n_b} q^{-n_b+1} \\ F(q) &= 1 + f_1 q^{-1} + ... + b_{n_f} q^{-n_f} \end{aligned} \tag{2.23}$$

where n_b is the order of the polynomial $B(q)+1$ and n_f is the order of the polynomial $F(q)$ [3] [24].

The Output-Error model is generally used for long term simulation of dynamical systems, but not for estimating a noise model because the white noise $e(k)$ affects only the output. The diagram of an OE structure in the SISO case is demonstrated in Figure 2.12. In this figure, $y^*(k)$ refers to noise-free output measurements.

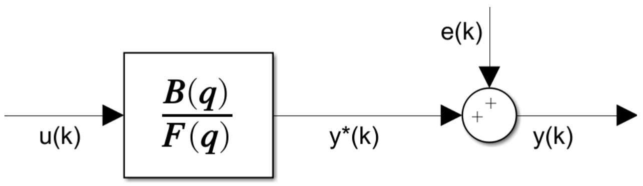

Figure 2.12: Scheme of an Output-Error model in the SISO case [5]

Then, the Output-Error model structure can be generalized as:

$$\begin{aligned} y(k) =& F_1 y(k-1) + ... + F_{n_f} y(k - n_f) + B_1 u(k - n_k) + ... \\ &+ B_{n_b} u(k - n_b - n_k + 1) + e(k) + F_1 e(k-1) + ... + F_{n_f} e(k - n_f) \end{aligned} \tag{2.24}$$

where n_u and n_y are the number of inputs and outputs of the system respectively, F_1, F_2, ... $\in \mathbb{R}^{n_y, n_u}$ and B_1, B_2, ... $\in \mathbb{R}^{n_y, n_u}$ are the coefficient matrices, $u(k) \in \mathbb{R}^{n_u}$ represents the inputs at sampling time instant k, $y(k) \in \mathbb{R}^{n_y}$ is the outputs at sampling time instant k, n_f and n_b are the orders of the OE model [3] [24].

2.3.2 Identification

The PRBS signals shown in Figure 2.2 are separated into two as an estimation and validation data sets. Then, both data sets are detrended i.e. their mean values are removed and the obtained signals are employed in order to estimate Output-Error polynomial models. Recall that the input-output delay $n_k = 1$. This delay is kept constant during estimation process. A set of possible combinations of n_b and n_f is specified. In this set, the range is $[1, 10] \in \mathbb{N}$ for both the orders n_b and n_f.

First, 100 models with different orders have been estimated iteratively by using the default solver (Gauss-Newton) of the MATLAB System Identification Toolbox. Also, the maximum number of iterations is selected as 50.

Identifying a nonlinear system via Output-Error models might be problematic. The main problem is the possible instability of the simulations during the model estimation procedure [28]. The Gauss-Newton solver pretends to fit the estimation data very well; indeed, the obtained Output-Error models have oscillations. The instability of some models with having a good fit percent for both the estimation and the validation data is demonstrated in Figure 2.13. Therefore, in the MATLAB

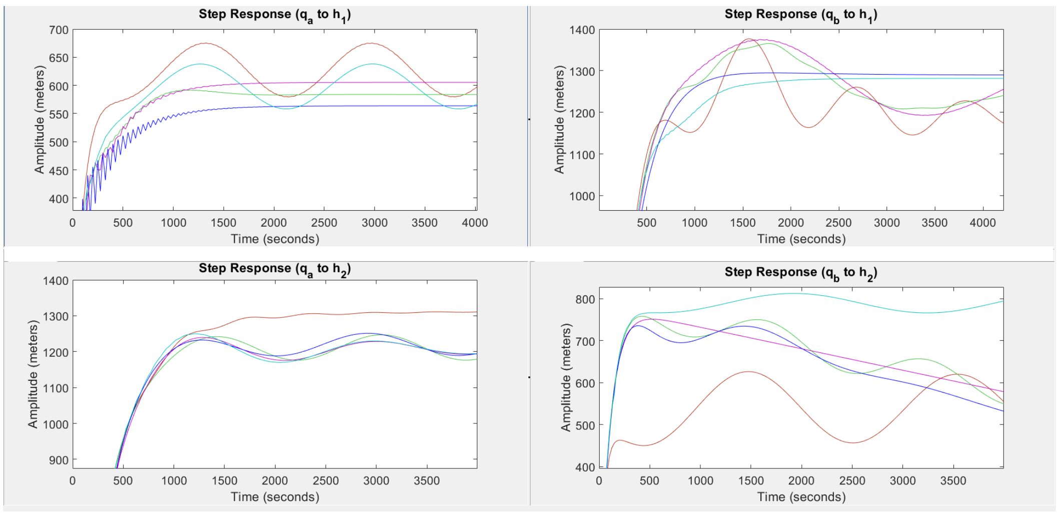

Figure 2.13: Transient response of some OE models with default solver

System Identification Toolbox, $fmincon$ solver with SQP algorithm [29] [30] [31] [32] is chosen and this solver is used with enforcing stability feature of the toolbox so as to get rid of instability issue. The OE models that provide the best fit percentage on the estimation data set can be seen in the first 15 rows of Table 2.6.

In Table 2.6, the orders refers to matrices; for example, in the first row $n_b = 7$ is equivalent to $n_b = [7\ 7;\ 7\ 7]$, $n_f = 2$ means that $n_f = [2\ 2;\ 2\ 2]$ etc. From this table, it is seen that the models oe131 and oe221 quite fit the estimation data despite the fact that the orders of these models are not large. As mentioned in previous sections, orders of a model cannot be determined just by looking at the NRMSE results based on identification dataset. The estimated models has to be tested using the validation data set and the implementation of validation is reported in Section 2.3.3. Nonetheless, analyzing the normalized Akaike's Information Criterion (nAIC) may give preliminary assessment about the validation performance because the estimated

n_b	n_f	n_k	Fit% $[h_1; h_2]$	$nAIC$
7	2	1	[75.51; 84.54]	-11.84
10	1	1	[74.40; 84.86]	-11.75
7	1	1	[74.35; 84.60]	-11.80
1	3	1	[74.11; 84.69]	-11.91
2	2	1	[74.53; 84.03]	-11.82
2	10	1	[72.30; 84.99]	-11.32
8	1	1	[74.03; 82.94]	-11.54
9	1	1	[74.19; 82.70]	-11.49
4	1	1	[73.71; 83.17]	-11.58
6	1	1	[73.63; 83.11]	-11.58
5	1	1	[73.33; 82.93]	-11.54
1	1	1	[72.07; 82.11]	-11.40
3	1	1	[72.33; 81.71]	-11.31
1	2	1	[73.56; 80.42]	-11.34
2	1	1	[71.43; 80.34]	-11.08
9	3	1	[46.15; 68.30]	-8.53
8	9	1	[66.33; 40.89]	-8.32
6	7	1	[26.64; 76.88]	-8.69
5	10	1	[73.39; 21.62]	-8.11
3	5	1	[75.77; 12.20]	-8.23

Table 2.6: Fit% of different OE models with orders n_b and n_f on the estimation dataset

models with smaller nAIC usually provide much better fit percent results on the validation data set for Output-Error models. Hence, the last 5 rows in the Table 2.6 imply that the validation performance of these models is supposed to be worse with regard to the ones on the top of the table.

2.3.3 Validation

The estimated Output-Error models with different orders are validated in the simulation mode by utilizing another data set and the models which satisfy the performance requirement the most are shown in Table 2.7.

The auto-correlation and cross-correlation have been defined in Section 2.1. According to Table 2.7, all the cross-correlations between the output residuals and the associated inputs are acceptable i.e. all the models pass the independence test. However, the auto-correlations are not taken into consideration because the results for all the models with varying orders are extremely bad. For the OE models, it is already suggested that paying less attention to the results of the whiteness test by the MATLAB System Identification Toolbox [3].

From the Table 2.7, selecting the model with $n_b = 2$, $n_f = 2$ and $n_k = 1$ seems reasonable due to the fact that its performance based on NRMSE criterion is as good as the models with higher polynomial orders. Besides, the model complexity problem can be overcome by choosing oe221.

The simulated response comparison and the cross-correlation diagrams of the

n_b	n_f	n_k	**Fit% [h_1; h_2]**	***Cross − correlation***
9	1	1	[83.48; 73.73]	Perfect
8	1	1	[82.52; 73.95]	Perfect
3	6	1	[85.36; 68.89]	Very good
2	9	1	[83.70; 69.58]	Very good
10	1	1	[84.55; 68.52]	Perfect
6	1	1	[80.61; 70.97]	Perfect
2	**2**	**1**	**[84.18; 67.15]**	**Very good**
2	10	1	[83.29; 67.92]	Very good
2	7	1	[83.26; 66.74]	Very good
2	8	1	[83.65; 65.77]	Very good
7	1	1	[82.79; 66.61]	Perfect
7	2	1	[85.18; 64.08]	Very good
5	1	1	[79.65; 68.37]	Perfect
4	1	1	[80.20; 67.75]	Perfect
2	6	1	[83.20; 64.02]	Very good
2	5	1	[82.96; 62.56]	Very good
1	3	1	[80.92; 64.57]	Perfect
7	3	1	[84.93; 58.93]	Very good
4	5	1	[83.50; 59.85]	Perfect
5	8	1	[82.11; 60.88]	Very good
1	1	1	[78.44; 63.58]	Perfect
6	5	1	[78.86; 62.84]	Very good
2	4	1	[83.09; 58.32]	Very good
3	1	1	[77.83; 63.37]	Perfect

Table 2.7: Fit% and cross-correlations of the estimated models vs. validation data

candidate Output-Error model with $n_b = 2$, $n_f = 2$ and $n_k = 1$ are demonstrated in Figure 2.14 and Figure 2.15 respectively.

As it has been done in the previous sections, additional tests are performed to make certain that the selected OE model is the right choice. The PRBS signals seen in Figure 2.8 are used for the additional tests after detrending operation. As it has been mentioned before, four different test data sets have been merged so as to constitute the signals. Here, only the second and the third portions are employed and the simulation results are represented in Table 2.8.

In Table 2.8, there is only one fit percentage value for each row since the average of both output performances is also taken. According to this table, the Output-Error polynomial model with $n_b = 2$, $n_f = 2$, and $n_k = 1$ has the second highest average fit percentage on the test data sets, so this model can be chosen.

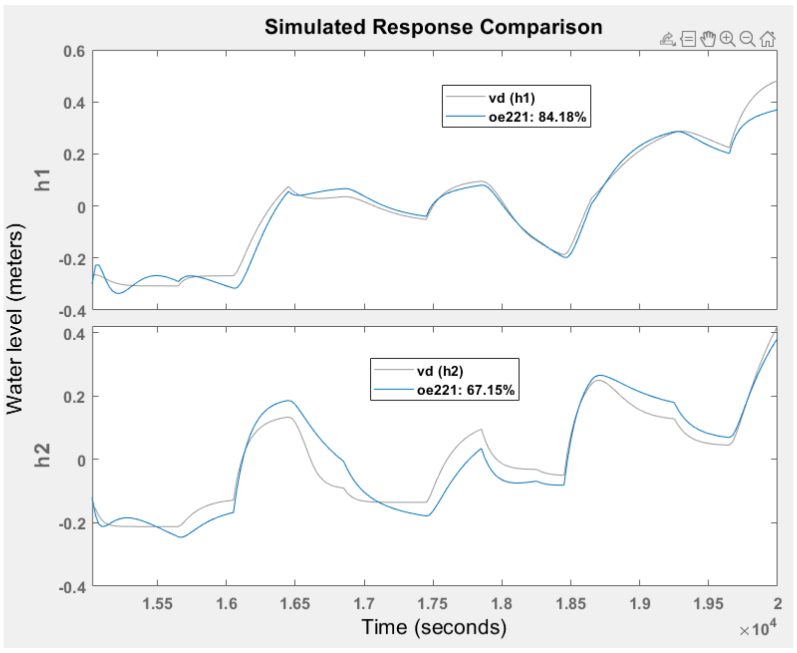

Figure 2.14: The candidate OE model (oe221) vs. validation data

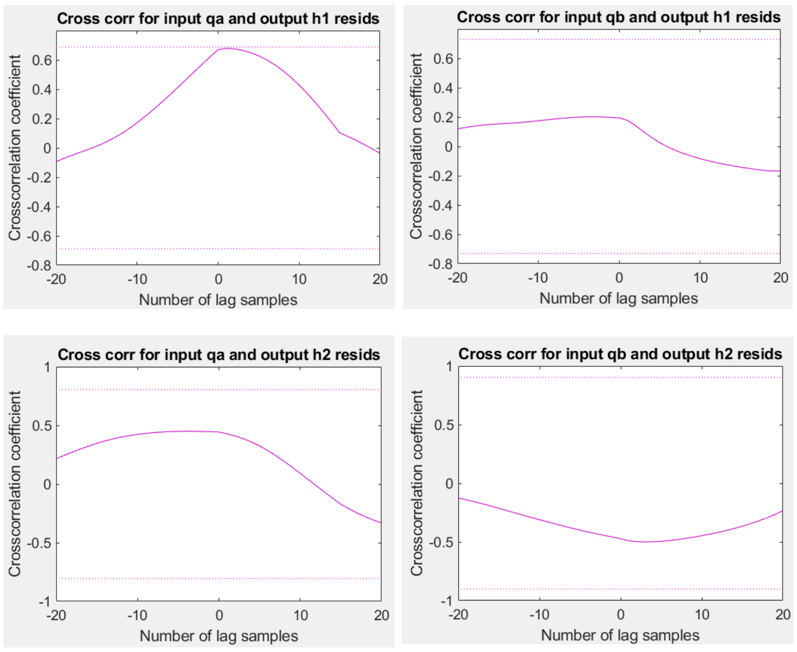

Figure 2.15: Cross-correlations of the candidate model on the validation dataset

The polynomials for the identified Output-Error model with the orders $n_b = 2$, $n_f = 2$, and $n_k = 1$ are the following:

n_b	n_f	n_k	**Average Fit%**
9	1	1	[77.80]
8	1	1	[76.82]
3	6	1	[74.67]
2	9	1	[78.53]
10	1	1	[76.89]
6	1	1	[74.29]
2	**2**	**1**	**[78.46]**
2	10	1	[77.11]
2	7	1	[77.10]
2	8	1	[77.40]
7	1	1	[75.14]
7	2	1	[77.74]
5	1	1	[73.02]
4	1	1	[74.07]
2	6	1	[75.80]
2	5	1	[73.97]
1	3	1	[75.40]
7	3	1	[69.43]
4	5	1	[68.81]
5	8	1	[75.36]

Table 2.8: Average fit% of the candidate models on the test datasets

The polynomials for the first output h_1 are

$$\begin{aligned} B_{11}(q) &= 111.5376q^{-1} - 93.3374q^{-2} \\ B_{12}(q) &= 11.5762q^{-1} + 21.6361q^{-2} \\ F_{11}(q) &= 1 - 1.5427q^{-1} + 0.5735q^{-2} \\ F_{12}(q) &= 1 - 1.6369q^{-1} + 0.6627q^{-2} \end{aligned} \tag{2.25}$$

and the polynomials for the second output h_2 are

$$\begin{aligned} B_{21}(q) &= 19.2780q^{-1} + 23.0406q^{-2} \\ B_{22}(q) &= 155.1399q^{-1} - 148.9504q^{-2} \\ F_{21}(q) &= 1 - 1.4769q^{-1} + 0.5094q^{-2} \\ F_{22}(q) &= 1 - 1.7575q^{-1} + 0.7658q^{-2} \end{aligned} \tag{2.26}$$

The zeros and the poles of the corresponding polynomials of the considered Output-

Error model are specified in the following equation.

$$
\begin{aligned}
&\text{Zero: } +0.8368, \qquad \text{Poles: } 0.9178 \text{ and } 0.6249 \text{ for } \frac{B_{11}(q)}{F_{11}(q)} \\
&\text{Zero: } -1.8690, \qquad \text{Poles: } 0.9033 \text{ and } 0.7337 \text{ for } \frac{B_{12}(q)}{F_{12}(q)} \\
&\text{Zero: } -1.1952, \qquad \text{Poles: } 0.9279 \text{ and } 0.5489 \text{ for } \frac{B_{21}(q)}{F_{21}(q)} \\
&\text{Zero: } +0.9601, \qquad \text{Poles: } 0.9585 \text{ and } 0.7990 \text{ for } \frac{B_{22}(q)}{F_{22}(q)}
\end{aligned}
\tag{2.27}
$$

In brief, the identified Output-Error polynomial model could not fit the validation data with a sufficient percentage because of the nonlinear dynamics of the system. However, the fluctuations for the outputs in the step response are removed by means of *fmincon* solver of the MATLAB System Identification toolbox. The transient responses for the selected model is shown in Figure 2.16.

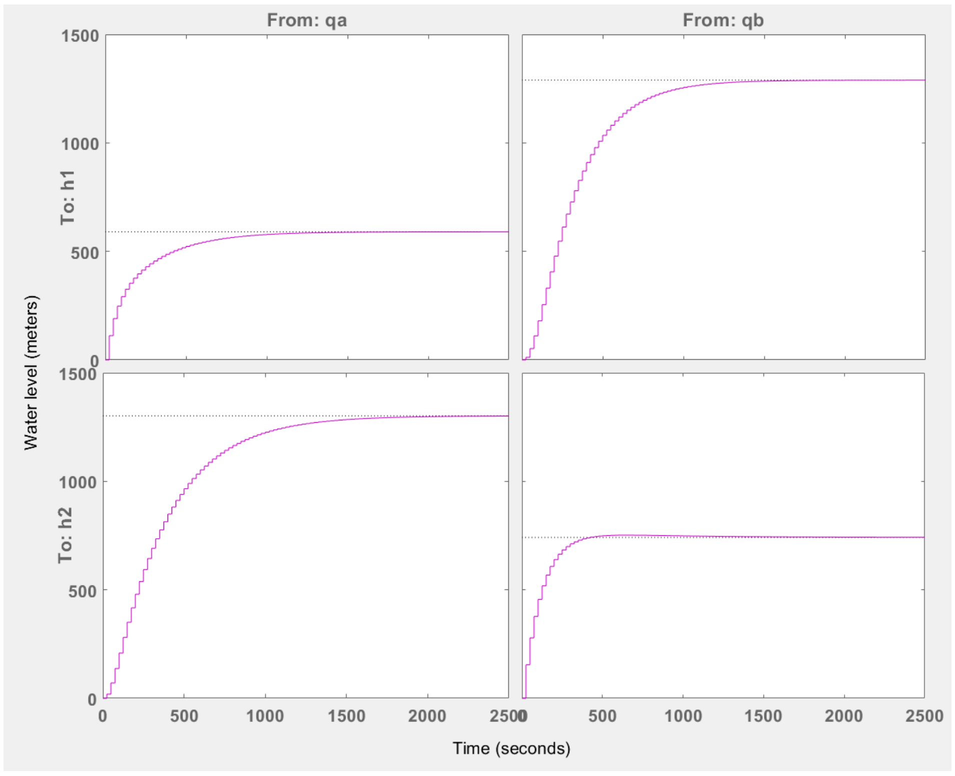

Figure 2.16: Step responses for the chosen OE model

2.4 ARX Models

The ARX model structure is an auto-regressive model subject to exogenous inputs. Regardless of the system to be either SISO or MIMO, ARX models generate linear predictors; hence, permitting the use of linear estimation methods to perform parameter estimations.

In the following parts, first, the structure of the linear ARX model is expressed and the least-square equation is stated. Then, the performance of the estimated ARX models on the validation data set is evaluated based on NRMSE quality metric and residue correlations. Afterwards, additional tests are also performed in order to get more robust results for the candidate ARX model.

2.4.1 Model Structure

The structure of an Auto-regressive Exogenous (ARX) model is

$$\begin{aligned} y(k) = A_1 y(k-1) + ... + A_{n_a} y(k-n_a) + B_1 u(k-n_k) \\ + ... + B_{n_b} u(k-n_b-n_k+1) + e(k) \end{aligned} \tag{2.28}$$

where n_u and n_y are the number of inputs and outputs of the plant respectively, A_1, A_2, ... $\in \mathbb{R}^{n_y,n_y}$ and B_1, B_2, ... $\in \mathbb{R}^{n_y,n_u}$ are coefficient matrices, $u(k) \in \mathbb{R}^{n_u}$ represents the inputs at sampling time instant k, $y(k) \in \mathbb{R}^{n_y}$ is the outputs at sampling time instant k, n_a and n_b are the orders of the ARX model and they represent the number of past output and past input terms used in order to estimate the current output respectively, n_k refers to the input-output delay and $e(k)$ is the noise (disturbance) term [24].

In Section 2.4.2, ARX models are estimated by means of the MATLAB System Identification toolbox, which solves the over-determined set of linear equations that constitutes the least-squares estimation problem [3]. The ARX model parameters vector ϑ is estimated by solving the following least-square equation:

$$\vartheta = (J^T J)^{-1} J^T y_{obs} \tag{2.29}$$

where J is the regressor matrix and y_{obs} is the measured outputs.

2.4.2 Identification and Validation

First, the ARX models are estimated by using the detrended estimation dataset which contains the input data seen in Figure 2.2 up to 15000 seconds (600 samples) and the associated outputs. As usual, the input-output delay n_k is selected as 1 sample time for all the transfer functions of the system. Various ARX models with different orders n_a and n_b are estimated from 1 to 10 i.e. the ARX models with the orders $n_a = [i\ i; i\ i]$ and $n_b = [j\ j; j\ j]$ are tried where $i,\ j\ = 1, 2, ..., 10$.

The considered linear ARX models are tested on the validation data which involves the remaining part of the input-output data shown in Figure 2.2 (last 5000 seconds or last 200 samples). Besides, for each estimated ARX model, the residual tests are performed on the validation data in order to further inspect the models. Since the estimations are made in the simulation mode, whiteness and independence tests reveal whether the simulation errors are white and uncorrelated to the input data. The results are reported in Table 2.9 and the specified orders refer to the matrices; for example, in the first row $n_a = 10$ is equivalent to n_a = [10 10; 10 10], $n_b = 9$ means that n_b = [9 9; 9 9] etc.

n_a	n_b	n_k	$Fit_{est}\%$ $[h_1;\ h_2]$	$Fit_{val}\%$ $[h_1;\ h_2]$	*Residuals*
10	9	1	[79.85; 87.13]	[84.51; 81.19]	Mediocre
10	10	1	[79.66; 88.81]	[83.72; 70.89]	Bad
3	3	1	[71.31; 78.67]	[74.58; 64.19]	Good
2	3	1	[69.02; 81.35]	[76.29; 58.59]	Good
2	**2**	**1**	**[66.29; 77.08]**	**[77.43; 52.26]**	**Outstanding**
1	2	1	[58.25; 75.41]	[62.38; 53.47]	Outstanding
1	1	1	[68.86; 77.06]	[64.92; 47.94]	Very Bad
1	10	1	[50.84; 65.58]	[47.09; 52.45]	Good
1	9	1	[48.37; 66.68]	[44.06; 52.55]	Good
3	2	1	[34.48; 70.31]	[53.46; 40.90]	Good
1	8	1	[45.41; 67.03]	[40.24; 51.67]	Good
1	7	1	[41.96; 66.85]	[35.69; 49.66]	Good
1	6	1	[37.95; 66.25]	[30.36; 46.73]	Good
1	5	1	[33.11; 64.47]	[23.97; 42.08]	Good
2	1	1	[-22.11; 9.89]	[32.98; 14.35]	Outstanding

Table 2.9: Performance of the estimated ARX models with different orders

15 different linear ARX models are added to Table 2.9 due to various reasons. The models with higher polynomial order n_a fit the validation data much better than the others; however, the residuals of these models violate more the confidence intervals of the auto-correlation and cross-correlation functions. Some of the models pass the whiteness and independence tests but their performance on the validation data are inadequate, such as $arx121$ and $arx211$. Also, the estimation fit percent of $arx211$ is very poor.

The results of the whiteness and the independence tests for the models with good validation fit percentage are demonstrated in Figure 2.17. From this figure, it is seen that all the considered models perform quite well in the independence test since only a few residuals go beyond the $\pm 3\sigma$ confidence interval of the cross-correlation function for each model. On the other hand, only the ARX model with the orders $n_a = 2$, $n_b = 2$ and $n_k = 1$ could pass the whiteness test owing to the fact that the residuals of other models violate significantly the confidence limits of the auto-correlation function at several lag samples. For this reason, the $arx221$ has been determined as the candidate model. The simulated response comparison diagram of the candidate ARX model is shown in Figure 2.18.

Additional tests are also performed before selecting the candidate model ($arx221$) in order to have more reliable results. The results are shown in Table 2.10.

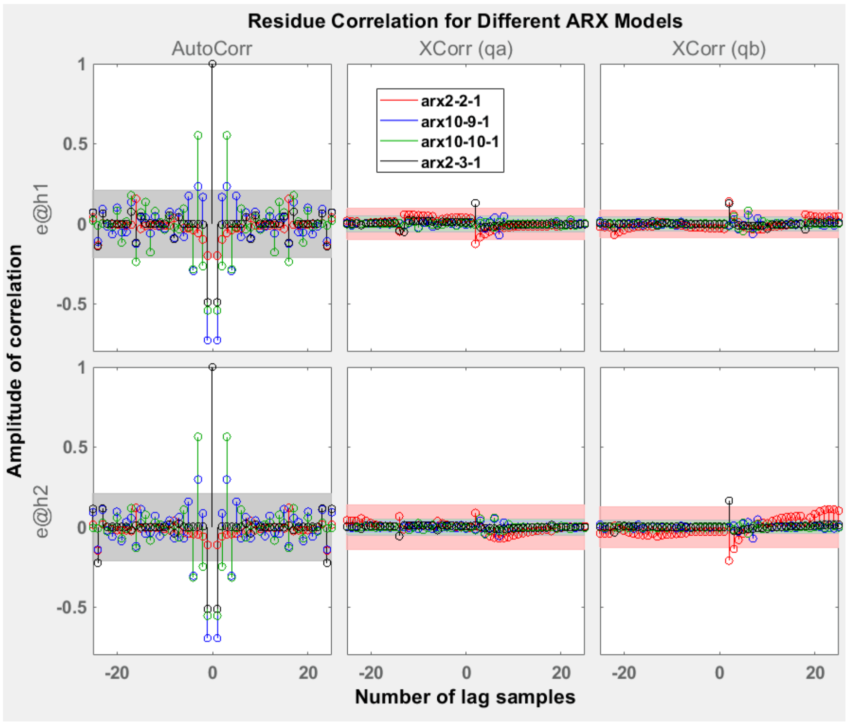

Figure 2.17: Comparison of residue correlations of different ARX models

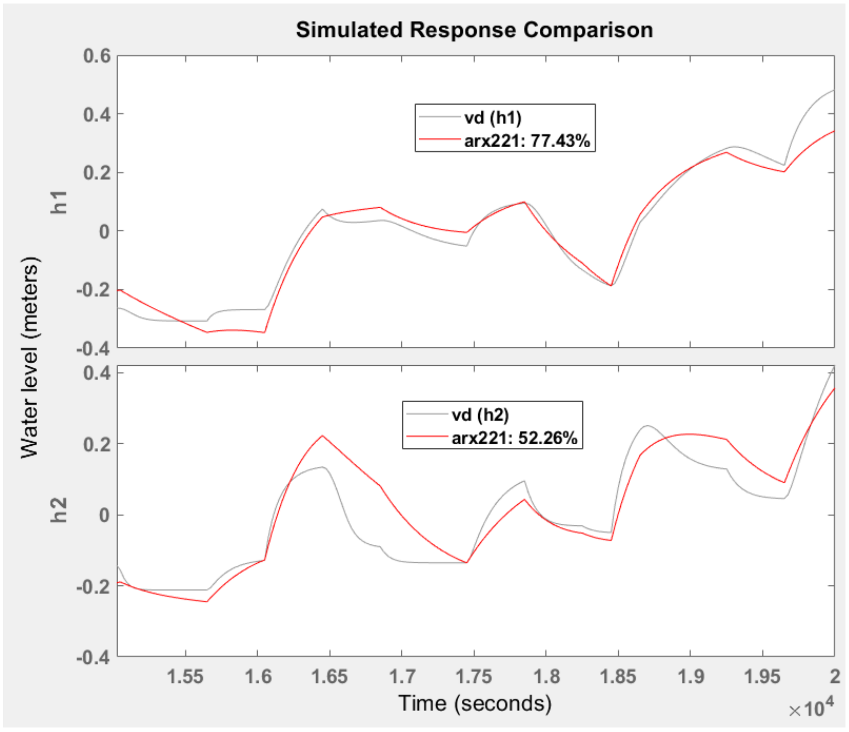

Figure 2.18: The candidate ARX model vs. validation data set

According to Table 2.10 and Figure 2.18, the candidate model could fit the test

Model Name	$\boldsymbol{Fit_{test_1}\%\ [h_1;\ h_2]}$	$\boldsymbol{Fit_{test_2}\%\ [h_1;\ h_2]}$
arx221	[72.79; 72.56]	[76.40; 69.18]

Table 2.10: Performance of the candidate ARX model on the test datasets

data sets even more than it could fit the validation data on the average. Therefore, the candidate model $arx221$ has been selected.

The polynomials for the chosen ARX model with the orders $n_a = 2$, $n_b = 2$, and $n_k = 1$ are the following:
The polynomials for the first output h_1 are

$$\begin{aligned} A_{11}(q) &= 1 - 1.9659q^{-1} + 0.9663q^{-2} \\ A_{12}(q) &= 0.0471q^{-1} - 0.0455q^{-2} \\ B_{11}(q) &= 42.1906q^{-1} - 40.0763q^{-2} \\ B_{12}(q) &= 89.4773q^{-1} - 88.2050q^{-2} \end{aligned} \tag{2.30}$$

and the polynomials for the second output h_2 are

$$\begin{aligned} A_{21}(q) &= 0.0058q^{-1} - 0.0068q^{-2} \\ A_{22}(q) &= 1 - 1.9228q^{-1} + 0.9250q^{-2} \\ B_{21}(q) &= 73.5355q^{-1} - 71.5468q^{-2} \\ B_{22}(q) &= 79.0570q^{-1} - 78.9397q^{-2} \end{aligned} \tag{2.31}$$

2.5 Comparison of the Selected Linear Models

The average fit percent of two outputs on the validation dataset for the chosen State-Space, ARMAX, OE, and ARX models are 68.23%, 75.55%, 75.67%, and 64.85% respectively. Namely, the performance of the selected ARMAX and Output-Error models based on NRMSE criterion are very close, the performance of the considered State-Space model is slightly worse than them, and the performance of the corresponding ARX model is the worst; nevertheless, there are no huge differences between the fit percent of the selected models. It is significant to emphasize that the considered State-Space and ARX models are extremely successful in the whiteness and the independence tests. Lastly, the nonlinear models can be employed in order to improve the system identification performance. The comparison of the selected models by using different methods is shown in Table 2.11.

Model Name	$\boldsymbol{Fit_{val}\%}$ $[\boldsymbol{h_1};\ \boldsymbol{h_2}]$	***Autocorr.***	***Crosscorr.***
ss4	[77.20%; 59.25%]	Outstanding	Perfect
armax4004411	[81.06%; 70.04%]	Good	Good
oe221	[84.18%; 67.15%]	Very bad	Outstanding
arx221	[77.43%; 52.26%]	Perfect	Outstanding

Table 2.11: Comparison of the obtained models with different methods

2.6 Summary

In this chapter, first, the PRBS signals used for identifying the system are defined and their figures are demonstrated. Afterwards, the pre-processing steps are explained and two different estimation modes (prediction and simulation) are discussed. Then, the input-output delay (transport delay) is described and the choice of the delay order is done by means of input-output impulse response of the estimation data. At the end, the performance evaluation criterion is mentioned and its equation is provided.

In Section 2.1, State-Space models are analyzed. Initially, the structure of a continuous-time State-Space model is reported. After that, several State-Space models are estimated with different orders. The corresponding models are validated by using a different data set. Next, the auto-correlation and cross-correlation functions are explained so as to evaluate the performance of the models on the validation dataset from a different point of view. The candidate model is tested on different data sets to get a more robust result. Hereby, the coefficients of the matrices A, B, C, K are expressed; the zeros and the poles of the selected model are stated. In the subsequent sections, input-output polynomial models are examined.

In Section 2.2 and 2.3, ARMAX models and Output-Error models have been reported respectively in a similar content by considering model structures, identification together with normalized AIC, validation by looking at fit percent and correlation functions, and additional tests. After the validation phase, the polynomials, zeros and poles of the selected models are reported.

In Section 2.4, the model structure of linear ARX model has been specified. Then, the estimated ARX models have been validated by using a different data set and the performance of the considered models has been evaluated with regard to fit percent on the validation data. In addition to this, extra tests have been performed for the candidate model so as to attain more reliable results. Ultimately, in Section 2.5, the comparison of the performance achieved with the different methods has been made.

Chapter 3

Nonlinear Model Identification

As it is seen in Chapter 2, the estimated linear models with different methods could not identify the Quadruple-Tank system sufficiently well because the corresponding system has nonlinear dynamics and wide range of operating conditions have been used in order to identify the plant.

The excitation signal employed for the identification procedure has been described in Chapter 2. Here, the input and output signals utilized for identification of nonlinear models are the same as those used in the linear identification. Two uncorrelated PRBS signals with different amplitude are used for the inputs q_a and q_b.

These asymmetric PRBS input signals and the associated outputs need to be divided into two datasets which are called estimation and validation datasets. The estimation dataset is composed of the first 75% of the input-output signals and the validation dataset consists of the remaining part of the data.

For nonlinear model identification, detrending the datasets is not required; therefore, the input signals shown in Figure 2.2 and the associated outputs are employed without removing the mean values or linear trends.

Two different estimation modes have been explained in the previous chapter. Once again, all nonlinear models are calibrated in the simulation mode, which computes the simulated outputs instead of utilizing the measured ones, due to the fact that it is more reliable and robust as compared to prediction mode. However, the simulation mode is computationally heavier than the prediction mode as expected.

Moreover, the input-output delay determined in Chapter 2 is used for nonlinear model identification. Recall that the dead-time has been chosen as 1 sample time for all the transfer functions of the system ($n_k = 1$).

In this chapter, three different nonlinear models are studied. First, the structures of the models are reported in each section. Then, many models are estimated with different orders and their performance are evaluated. After that, the estimated models are tested on the validation dataset and the results based on NRMSE quality metric and correlations of the residuals are analyzed. Additional tests are also performed in the validation phase so as to obtain more reliable results. All in all, each selected nonlinear model is compared with regard to fit percentage, model complexity in terms of orders, auto-correlations and cross-correlations of the residuals.

In the following sections, first, the Nonlinear Auto-regressive Exogenous (NARX) models are estimated. At the beginning of identification procedure in Section 3.1, the output function of the estimated models not only contain the linear function but also the nonlinear function. Then, the function that represents the non-linearity is omitted in order to reduce the model complexity and the candidate NARX model transforms into the Polynomial NARX model. Hammerstein−Wiener and Neural NARX (NNARX) models are also implemented in this chapter. The difference between the Polynomial NARX model and the Neural NARX model is that the former is linear in the parameters, whereas the latter uses a neural network and it is nonlinear in the parameters. The NNARX models are more complex, but better result might be obtained by using them.

3.1 NARX Models

The NARX model structure is a non-linear autoregressive model with exogenous inputs. The NARX models are commonly used in the process industry so as to identify nonlinear systems. The NARX models extend the linear ARX models. This extension allows to model complex nonlinear behavior of the system by utilizing output functions that combine nonlinear and linear components.

In the following parts, first, the structure of the nonlinear ARX (NARX) model is reported. Then, the estimated NARX models are validated by analyzing the performance of the corresponding models with regard to NRMSE criterion and residue correlations.

3.1.1 Model Structure

The general structure of a NARX model is

$$y(k) = F(y(k-1), ..., y(k-n_a), u(k-n_k), u(k-1-n_k), ..., u(k-n_b-n_k+1)) \tag{3.1}$$

where n_u and n_y are the number of inputs and outputs of the system respectively, $u(k) \in \mathbb{R}^{n_u}$ represents the inputs at sampling time instant k, $y(k) \in \mathbb{R}^{n_y}$ is the outputs at sampling time instant k, n_a and n_b are the orders of the NARX model and they represent the number of past output and past input terms used in order to estimate the current output respectively, n_k refers to the input-output delay, and F is the output function [3] [24]. The output function F may contain linear and/or nonlinear functions. Different mapping functions can be utilized, such as Wavelet network, Sigmoid network or Neural network for the nonlinear block.

The regressors of the NARX model can be more complex as compared to the linear ARX model because nonlinear combinations of delayed input and output variables can be included. In the case that a NARX model does not a contain nonlinear function, it is called Polynomial NARX model. In the Polynomial NARX model structure, the outputs depend linearly on the model parameters while they can depend non-linearly on the regressors. Recall that the structure of a linear ARX model

$$\begin{aligned} y(k) = A_1 y(k-1) + ... + A_{n_a} y(k-n_a) \\ + B_1 u(k-n_k) + ... + B_{n_b} u(k-n_b-n_k+1) + e(k) \end{aligned} \tag{3.2}$$

where A_1, A_2, ... $\in \mathbb{R}^{n_y, n_y}$ and B_1, B_2, ... $\in \mathbb{R}^{n_y, n_u}$ are coefficient matrices, $e(k)$ is the noise term [3] [24]. Then, the Polynomial NARX model can be represented as an extension of equation 3.2 i.e. nonlinear regressors which depend linearly on the parameters, such as $0.5y_1(k-1)^2$ or $-0.2y_2(k-2)u_1(k-3)$ can be added to this equation for any outputs.

3.1.2 Identification and Validation

The initial NARX model is obtained by using the selected ARX model as initial model for the estimation i.e the corresponding NARX model is estimated by using the orders and the parameters of the linear ARX model ($arx221$) that has been selected in Section 2.4. In this way, the orders of the linear model become the orders of the NARX models as well. Nonetheless, the unmodified estimation and validation datasets are put into operation instead of detrended ones in order to acquire the initial NARX model.

The fit percent of the corresponding NARX model on the estimation and validation data sets are [62.71%; 71.47%] and [60.24%; 67.43%] respectively. Furthermore, the additional test datasets are utilized so as to acquire more reliable results. The test datasets employed in the previous chapter are also used here. The performance of the initialized NARX model on the different test datasets are [53.60%; 69.63%] and [66.50%; 69.24%] respectively.

As it is seen from the previous paragraph, the performance of the considered NARX model is not sufficient. Therefore, after getting the initial NARX model by refining the selected ARX model in the simulation mode, the nonlinear regressors that enhance the performance are added to the outputs of the acquired NARX models iteratively by means of the MATLAB System Identification toolbox. In the regressor selection procedure, a nonlinear regressor is added to the existing NARX model and the performance of the obtained NARX model is evaluated by analyzing the fit percent on the validation and test data sets, and by looking at the correlations of the residuals. If it is thought that the overall performance increases with the addition of new regressor, the corresponding NARX model is determined as a new candidate model. During this identification procedure, the nonlinear mapping function is also included in the output function F in addition to linear function. To this end, Wavelet Network function is employed for both outputs.

In this work, some nonlinear regressors are tried for adding them to the NARX models. Since the orders of the NARX models are $n_a = 2$ and $n_b = 2$, all nonlinear regressors with only second polynomial degree could have been taken into account. However, the nonlinear regressors with second polynomial degree are not tested in the random order, whereas these regressors have been tried starting from the ones with larger "maximum value of the outputs" on the NARX model plot. An example plot is shown in Figure 3.1 for the initialized NARX model. In this figure, the relationship between the regressors $h_1(t-1)$, $h_1(t-2)$ and the first output h_1 is plotted by means of the MATLAB System Identification toolbox. It is seen that the plot looks like a plane for the corresponding regressors; hence, the relationship between the regressors and the considered output is probably linear. Notwithstanding, in this thesis, nonlinear function is also taken into account in the output function at the beginning. After a non-polynomial NARX model with sufficient performance is acquired, it will be tried to convert it to a polynomial model structure by omitting the nonlinear part.

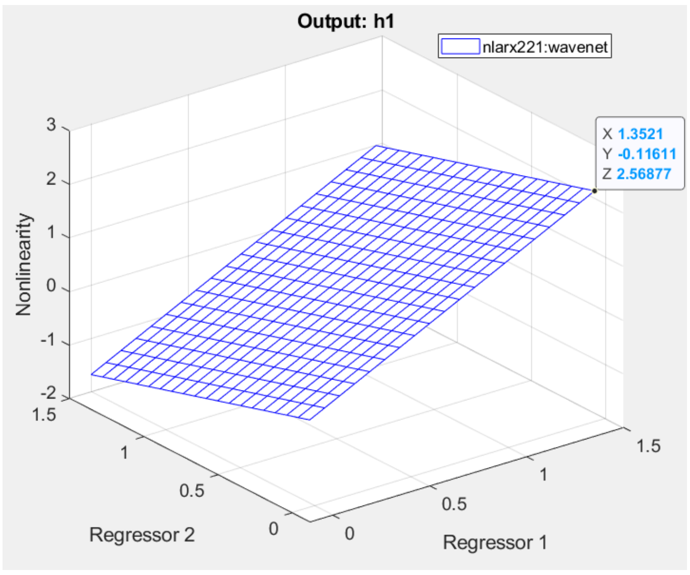

Figure 3.1: The relationship between the corresponding regressors and the first output

In Figure 3.1, the largest value of the plane on the Z axis is approximately 2.57. Owing to the fact that the orders of the initialized NARX model is $n_a = [2\ 2;\ 2\ 2]$, $n_b = [2\ 2;\ 2\ 2]$ and $n_k = [1\ 1;\ 1\ 1]$, the linear regressors of this model are $h_1(t-1)$, $h_1(t-2)$, $h_2(t-1)$, $h_2(t-2)$, $q_a(t-1)$, $q_a(t-2)$, $q_b(t-1)$, $q_b(t-2)$ for both outputs. The maximum value for each NARX plot is reported in Table 3.1 and Table 3.2.

	$h_1(t-1)$	$h_1(t-2)$	$h_2(t-1)$	$h_2(t-2)$	$q_a(t-1)$	$q_a(t-2)$	$q_b(t-1)$	$q_b(t-2)$
$h_1(t-1)$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$
$h_1(t-2)$	**2.57**	$*$	$*$	$*$	$*$	$*$	$*$	$*$
$h_2(t-1)$	2.05	1.13	$*$	$*$	$*$	$*$	$*$	$*$
$h_2(t-2)$	2.05	1.33	0.81	$*$	$*$	$*$	$*$	$*$
$q_a(t-1)$	1.97	1.25	0.73	0.73	$*$	$*$	$*$	$*$
$q_a(t-2)$	1.96	1.24	0.69	0.72	0.65	$*$	$*$	$*$
$q_b(t-1)$	1.99	1.27	0.75	0.75	0.68	0.67	$*$	$*$
$q_b(t-2)$	1.98	1.26	0.68	0.74	0.66	0.63	0.69	$*$

Table 3.1: The maximum value for the first output h_1 on the NARX plots

During the regressor selection procedure, it is observed that all of the obtained models fit the estimation data quite well and all of them also pass the independence test; therefore, the fit percent related to estimation dataset and the cross-correlation evaluation are not included in Table 3.3. In this table, $Val\%$, Rel_{val}, $Autocorr_{val}$, $Test_1\%$, $Test_2\%$, $Avg\%$ and Rel_{avg} refer to the mean value of the fit% of the outputs of the corresponding model on the validation data set, the difference between the fit% of the new candidate model and the previous one, the residue correlations on the validation data set, the mean value of the fit% of the outputs of the corresponding

	$h_1(t-1)$	$h_1(t-2)$	$h_2(t-1)$	$h_2(t-2)$	$q_a(t-1)$	$q_a(t-2)$	$q_b(t-1)$	$q_b(t-2)$
$h_1(t-1)$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$
$h_1(t-2)$	0.72	$*$	$*$	$*$	$*$	$*$	$*$	$*$
$h_2(t-1)$	1.87	1.86	$*$	$*$	$*$	$*$	$*$	$*$
$h_2(t-2)$	1.21	1.05	**2.35**	$*$	$*$	$*$	$*$	$*$
$q_a(t-1)$	0.68	0.67	1.81	1.16	$*$	$*$	$*$	$*$
$q_a(t-2)$	0.68	0.61	1.81	1.10	0.62	$*$	$*$	$*$
$q_b(t-1)$	0.69	0.68	1.83	1.17	0.64	0.63	$*$	$*$
$q_b(t-2)$	0.69	0.59	1.83	1.08	0.64	0.58	0.65	$*$

Table 3.2: The maximum value for the second output h_2 on the NARX plots

model on the first test data set, the mean value of the fit% of the outputs of the corresponding model on the second test data set, the average fit% of the corresponding model on the validation and test data sets, and the difference between the average fit% of the new candidate model and the previous one respectively.

Custom Regressor	***Added***	***Val%***	***Rel$_{val}$***	***Autocorr$_{val}$***	***Test$_1$%***	***Test$_2$%***	***Avg%***	***Rel$_{avg}$***
None	h_1, h_2	63.84		Perfect	61.62	67.87	64.44	
$h_1(t-1) \cdot h_1(t-2)$	h_1	65.66	+1.83	Very Bad	76.46	67.72	69.94	+5.50
$h_1(t-1)^2$	h_1	71.12	+5.46	Very Bad	75.78	82.29	76.40	+6.45
$h_2(t-1) \cdot h_2(t-2)$	h_2	80.45	+9.32	Very Bad	81.13	77.59	79.72	+3.32
$h_1(t-1) \cdot h_2(t-1)$	h_1	79.36	−1.08	Mediocre	82.02	87.65	83.01	+3.29
$h_2(t-2)^2$	h_2	81.93	+2.57	Mediocre	80.39	87.98	83.43	+0.42
$h_2(t-1)^2$	h_2	81.36	−0.58	Mediocre	80.91	89.54	83.93	+0.50
$h_1(t-2)^2$	h_1	82.08	+0.72	Mediocre	82.70	89.57	84.78	+0.85
$h_2(t-1) \cdot q_b(t-1)$	h_2	81.99	−0.10	Mediocre	84.83	90.05	85.62	+0.84
$q_a(t-1) \cdot q_b(t-1)$	h_1	82.35	+0.37	Mediocre	84.01	89.86	85.41	−0.21
$q_a(t-1) \cdot q_b(t-2)$	h_1	82.45	+0.10	Mediocre	85.96	89.00	85.80	+0.40
$q_a(t-2) \cdot q_b(t-1)$	h_1	82.44	−0.01	Mediocre	86.20	89.09	85.91	+0.11
$h_1(t-1) \cdot h_2(t-2)$	h_2	83.29	+0.85	Good	84.59	65.83	77.90	−8.01
$h_1(t-2) \cdot h_2(t-2)$	h_2	82.27	−1.02	Good	83.99	77.30	81.18	+3.28
$h_2(t-1) \cdot q_a(t-1)$	h_2	83.33	+1.06	Good	84.89	73.50	80.57	−0.61
$h_2(t-1) \cdot q_a(t-2)$	h_2	83.39	+0.06	Good	85.00	80.08	82.82	+2.25
$h_2(t-2) \cdot q_b(t-1)$	h_2	87.41	+4.02	Mediocre	87.24	61.86	78.84	−3.99
$q_a(t-1) \cdot q_a(t-2)$	h_2	88.88	+1.48	Mediocre	87.84	72.68	83.13	+4.30
$h_1(t-1) \cdot q_a(t-1)$	h_2	89.80	+0.91	Mediocre	86.37	74.60	83.59	+0.46
$h_1(t-1) \cdot q_a(t-2)$	h_2	90.34	+0.54	Good	87.65	53.48	77.15	−6.43
$h_1(t-1) \cdot q_b(t-1)$	h_2	90.18	−0.16	Good	87.90	61.90	79.99	+2.84
$h_1(t-1) \cdot q_b(t-2)$	h_2	89.12	−1.06	Mediocre	88.26	79.27	85.55	+5.55

Table 3.3: Performance of the obtained NARX models after adding regressors iteratively

The initial NARX model, which has been obtained by refining the selected linear

ARX model, is located on the top of Table 3.3. Even though this initialized NARX model completely passes the whiteness and independence tests, its performance based on NRMSE criterion is insufficient both on the validation and the test datasets. The average fit percent for this model is approximately 64%. Hence, custom nonlinear regressors are added. After adding $h_1(t-1) \cdot h_2(t-1)$ to the regressors of the first output h_1, the acquired NARX model fits both the validation and test datasets much better than the initial NARX model. Besides, the residuals of the corresponding model are less correlated as compared with the models in the second, the third and the fourth rows, but this model still fails the whiteness test.

Similarly, the overall performance of the NARX models are increased by adding the considered regressors one by one up to the 12th row. When the regressor $h_1(t-1) \cdot h_2(t-2)$ is appended to the regressors of the second output h_2, the fit percent of the obtained model on the validation dataset increases slightly; nevertheless, the fit% on the second test dataset decreases from 89.09% to 65.83%. It has been decided to include this regressor because the corresponding model performs better in the whiteness test and it still fits well the validation data set. The test data sets are important but the validation dataset is more significant than others in this work.

The performance of the estimated NARX models based on NRMSE quality metric is enhanced by adding other nonlinear regressors. In the event that the regressor in the 17th row is appended to the regressors of the second output h_2, the fit percent of the obtained model on the validation dataset reaches 87.41%. Although this model performs worse in the whiteness test and the performance on the second test dataset decreases considerably as compared to the previous model (from 80.08% to 61.86%), this regressor has been taken into account due to the importance of the validation data. Afterwards, the fit% on the second test data set is increased through the instrumentality of the regressor $q_a(t-1) \cdot q_a(t-2)$. Last, the regressor $h_1(t-1)$ is added to the regressors of the second output h_2 after multiplied by all input regressors. Thus, the NARX model with better performance is obtained by adding all of the nonlinear regressors in Table 3.3.

In Table 3.3, the nonlinear regressors with only second polynomial degree have been added up to this point. Now, some regressors with third and fourth polynomial degrees are also considered arbitrarily so as to improve the performance of the estimated NARX models with the orders $n_a = 2$, $n_b = 2$ and $n_k = 1$. The performance improvement of new candidate models is demonstrated in Table 3.4.

Some other regressors with higher polynomial degree are added to the obtained NARX model in addition to the ones reported in Table 3.3 in order to improve performance of the existing model on the validation and test datasets. In the first two rows of Table 3.4, the regressors $h_1(t-1)^2 \cdot h_1(t-2)^2$ and $h_2(t-1)^2 \cdot h_2(t-2)^2$ are simultaneously appended to the regressors of the outputs h_1 and h_2 respectively. After adding these regressors, the fit percentage of the corresponding NARX model on the validation dataset is increased. Then, the regressors with cubic polynomial degree ($h_1(t-1)^3$ and $h_1(t-2)^3$) are also added to the regressors of the first output h_1. In this way, the fit% of the existing model on the validation dataset becomes 91.21%. Moreover, the residuals of this model perform better in the whiteness test.

After adding the regressors with higher polynomial degree, the regressors that adversely influenced the model's performance before are taken into consideration once again because an omitted regressor depends also the regressors that is added

Custom Regressor	***Added***	***Val%***	***Rel$_{val}$***	***Autocorr$_{val}$***	***Test$_1$%***	***Test$_2$%***	***Avg%***	***Rel$_{avg}$***
$h_1(t-1)^2 \cdot h_1(t-2)^2$	h_1							
$h_2(t-1)^2 \cdot h_2(t-2)^2$	h_2	90.99	+1.87	Mediocre	80.28	83.68	84.98	−0.57
$h_1(t-1)^3$	h_1	91.06	+0.08	Good	86.95	91.76	89.92	+4.94
$h_1(t-2)^3$	h_1	91.21	+0.14	Good	87.17	92.64	90.34	+0.41
$h_1(t-1) \cdot q_a(t-1)$	h_1							
$h_1(t-1) \cdot q_a(t-2)$	h_1							
$h_1(t-1) \cdot q_b(t-1)$	h_1							
$h_1(t-1) \cdot q_b(t-2)$	h_1	89.89	−1.32	Good	86.03	87.61	87.84	−2.49
$h_2(t-1) \cdot q_b(t-1)$	h_1							
$h_2(t-1) \cdot q_b(t-2)$	h_1	92.23	+2.34	Good	88.58	87.77	89.52	+1.68
$h_2(t-2) \cdot q_a(t-2)$	h_1	91.45	−0.78	Good	86.07	88.08	88.53	−0.99
$h_2(t-1) \cdot q_b(t-2)$	h_2	90.18	−1.27	Very Good	85.38	88.29	87.95	−0.58
$h_2(t-2) \cdot q_b(t-2)$	h_2	90.98	+0.80	Very Good	86.24	87.51	88.24	+0.29
$q_a(t-1) \cdot q_b(t-1)$	h_2							
$q_a(t-1) \cdot q_b(t-2)$	h_2	**90.78**	−0.20	**Outstanding**	**87.42**	**88.33**	**88.84**	+0.60

Table 3.4: Performance of the acquired NARX models after adding nonlinear regressors with higher degree of polynomials

after it has been tried. Therefore, considering the omitted regressors might be beneficial for the existing NARX model. Starting from the fifth row in Table 3.4, four different regressors are added simultaneously to the existing nonlinear model. As it can be seen, the performance of the corresponding model based on NRMSE criterion decreases for all data sets; however, the auto-correlation of the residuals of this model are less than the previous model especially for the first output h_1.

When the nonlinear regressor $h_2(t-1) \cdot q_b(t-2)$ is appended to the regressors of the second output h_2, the acquired model performs very well on the different datasets in terms of fit percentage. Furthermore, the auto-correlation of the residuals of the corresponding model is reported as "very good" because the confidence interval of the auto-correlation function for the second output h_2 is much less violated in comparison with the previous estimated models. Finally, a model which meets all the requirements is acquired. The corresponding model fits both the validation and the test data sufficiently well and it also passes both the whiteness and the independence tests. Hence, this model, which contains both linear and nonlinear functions in its output function, has been considered as the candidate model.

In order to reduce the complexity of the candidate model, the nonlinear function block might be omitted. Thus, another NARX model which uses the same regressors as the candidate model has been estimated. Since all the corresponding regressors are used only in the linear block, the acquired model becomes a Polynomial NARX model. In the Polynomial NARX model structure, all regressors depend linearly on the model parameters. The performance evaluation of the obtained Polynomial NARX model is shown in Table 3.5.

There is no significant performance decrease for the Polynomial NARX model (approximately 2% decrease on the average). Moreover, in the event of the Polynomial NARX model, the complexity of the structure has been reduced. In the Polynomial

Model Type	***Val%***	***Autocorr$_{val}$***	***Test$_1$%***	***Test$_2$%***	***Avg%***
Non-polynomial NARX	90.78	Outstanding	87.42	88.33	88.84
Polynomial NARX	**89.97**	**Very Good**	**84.96**	**84.66**	**86.54**

Table 3.5: Performance comparison of the candidate NARX models (Polynomial vs. Non-polynomial)

NARX model, the residuals are slightly more correlated as compared to the candidate model which contains also the nonlinear function block; however, there is not much difference. The comparison of these models on the simulated response and residual correlation diagrams is shown in Figure 3.2 and Figure 3.3 respectively.

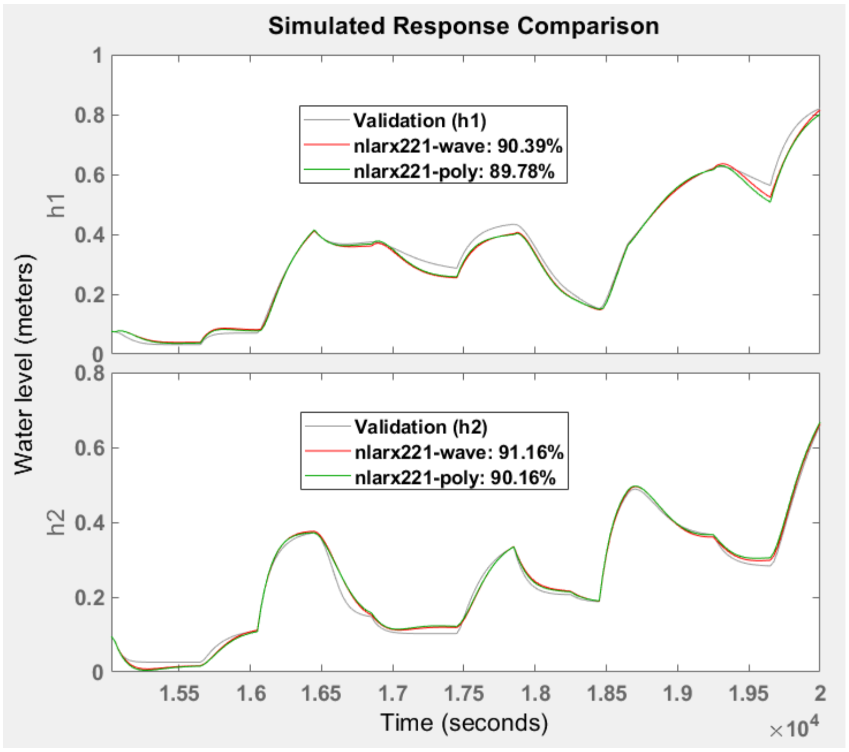

Figure 3.2: The comparison of the candidate NARX models on validation dataset

In figure 3.3, it is seen that all residuals are in the area of confidence interval of the cross-correlation function for both models. In other words, all the residuals of the considered NARX models pass the independence test. Likewise, the auto-correlation function of the residuals are in the region of confidence bounds for the first output of the non-polynomial model i.e. the corresponding residuals pass the whiteness test. Besides, there are two residuals that exceed the confidence interval of the auto-correlation function for the second output of the non-polynomial model at the lag samples -1 and +1, but these correlations can be ignored because their amplitude of correlation are very close to the confidence limit. On the other hand, the Polynomial NARX model perform slightly worse in the whiteness test; however,

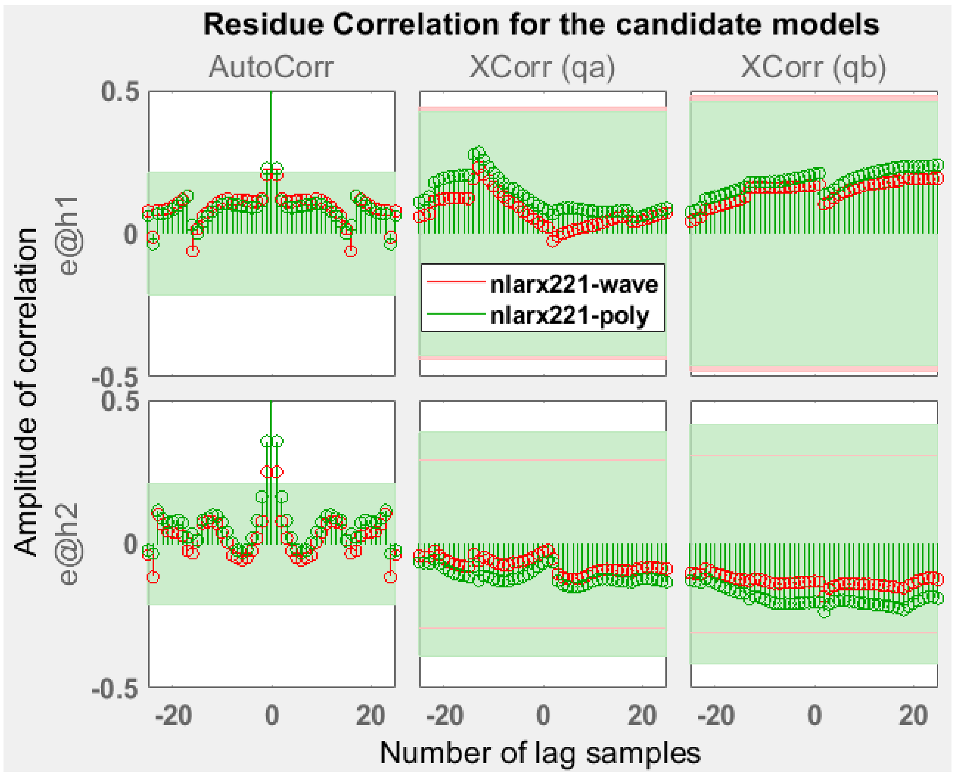

Figure 3.3: Residue correlations of the candidate NARX models

this little difference can be ignored because the complexity of the polynomial model structure is far less than the other one.

Since several nonlinear regressors have been added during the regressor selection procedure and the first and the second test data sets have been utilized in this procedure, there is a risk of overfitting effects. Other test data sets which have never been used before, now, take part in the validation phase so as to eliminate the risk of overfitting. The fit percent of the considered Polynomial NARX model on 14 different test data sets is demonstrated in Table 3.6.

$\boldsymbol{Test_3\%}$	$\boldsymbol{Test_4\%}$	$\boldsymbol{Test_5\%}$	$\boldsymbol{Test_6\%}$	$\boldsymbol{Test_7\%}$	$\boldsymbol{Test_8\%}$	$\boldsymbol{Test_9\%}$
86.21	81.35	95.56	84.78	94.60	93.27	86.94
$\boldsymbol{Test_{10}\%}$	$\boldsymbol{Test_{11}\%}$	$\boldsymbol{Test_{12}\%}$	$\boldsymbol{Test_{13}\%}$	$\boldsymbol{Test_{14}\%}$	$\boldsymbol{Test_{15}\%}$	$\boldsymbol{Test_{16}\%}$
89.55	90.79	89.82	88.87	86.64	88.44	91.65

Table 3.6: Performance of the selected Polynomial NARX model vs. test datasets never used before

According to Table 3.6, the considered model fits 89.18% the new data sets on the average. Due to the fact that the corresponding model could fit adequately each of the datasets according to the values reported in this table, this Polynomial NARX model has been selected. The structure of the selected polynomial model can be expressed as:

$$\begin{aligned}\hat{h}_1(t) &= \vartheta_1^{\mathsf{T}}\varphi_1(t-1) + d_1\\ \hat{h}_2(t) &= \vartheta_2^{\mathsf{T}}\varphi_2(t-1) + d_2\end{aligned} \tag{3.3}$$

where $\hat{h}_1(t)$ is the first estimated output, $\hat{h}_2(t)$ is the second estimated output, ϑ_1 is the parameters vector i.e. weights of the regressors for the first output, ϑ_2 is the vector of parameters for the second output, $\varphi_1(t-1)$ is the regressors vector for the first output, $\varphi_2(t-1)$ is the vector of regressors for the second output, d_1 and d_2 are the offsets of the first and the second outputs respectively.

According to equation 3.3, the vectors ϑ_1 and $\varphi_1(t-1)$ for the first output can be represented as

$$\vartheta_1 = \begin{bmatrix} -0.250760\\ +0.052640\\ -0.069129\\ +0.008187\\ -0.015926\\ -0.013793\\ -0.005396\\ +0.010468\\ -0.000814\\ +0.000142\\ -0.001009\\ +0.000162\\ -0.004072\\ -0.002668\\ +0.004720\\ +0.000304\\ -0.000187\\ -0.000716\\ -0.000468\\ +0.000188\\ +0.000015\\ -0.000797\\ -0.000065\\ +0.000009\\ -0.000839 \end{bmatrix}, \qquad \varphi_1(t-1) = \begin{bmatrix} h_1(t-1)\\ h_1(t-2)\\ h_2(t-1)\\ h_2(t-2)\\ q_a(t-1)\\ q_a(t-2)\\ q_b(t-1)\\ q_b(t-2)\\ h_1(t-1)\cdot h_1(t-2)\\ h_1(t-1)^2\\ h_1(t-1)\cdot h_2(t-1)\\ h_1(t-2)^2\\ q_a(t-1)\cdot q_b(t-1)\\ q_a(t-1)\cdot q_b(t-2)\\ q_a(t-2)\cdot q_b(t-1)\\ h_1(t-1)^2\cdot h_1(t-2)^2\\ h_1(t-1)^3\\ h_1(t-2)^3\\ h_1(t-1)\cdot q_a(t-1)\\ h_1(t-1)\cdot q_a(t-2)\\ h_1(t-1)\cdot q_b(t-1)\\ h_1(t-1)\cdot q_b(t-2)\\ h_2(t-1)\cdot q_b(t-1)\\ h_2(t-1)\cdot q_b(t-2)\\ h_2(t-2)\cdot q_a(t-2) \end{bmatrix} \tag{3.4}$$

and the parameters vector ϑ_2 and the vector of regressors $\varphi_2(t-1)$ of the selected

NARX model are

$$\vartheta_2 = \begin{bmatrix} -0.246340 \\ -0.075461 \\ +0.047468 \\ +0.003379 \\ +0.007900 \\ +0.024986 \\ +0.005304 \\ -0.006290 \\ +0.001439 \\ +0.001788 \\ +0.000671 \\ +0.004507 \\ -0.002816 \\ +0.005335 \\ -0.000849 \\ -0.001519 \\ -0.000975 \\ +0.001880 \\ -0.001269 \\ -0.000659 \\ -0.000587 \\ +0.000255 \\ -0.000583 \\ +0.000168 \\ -0.000005 \\ +0.000121 \\ +0.000380 \end{bmatrix}, \qquad \varphi_2(t-1) = \begin{bmatrix} h_1(t-1) \\ h_1(t-2) \\ h_2(t-1) \\ h_2(t-2) \\ q_a(t-1) \\ q_a(t-2) \\ q_b(t-1) \\ q_b(t-2) \\ h_2(t-1)\cdot h_2(t-2) \\ h_2(t-2)^2 \\ h_2(t-1)^2 \\ h_2(t-1)\cdot q_b(t-1) \\ h_1(t-1)\cdot h_2(t-2) \\ h_1(t-2)\cdot h_2(t-2) \\ h_2(t-1)\cdot q_a(t-1) \\ h_2(t-1)\cdot q_a(t-2) \\ h_2(t-2)\cdot q_b(t-1) \\ q_a(t-1)\cdot q_a(t-2) \\ h_1(t-1)\cdot q_a(t-1) \\ h_1(t-1)\cdot q_a(t-2) \\ h_1(t-1)\cdot q_b(t-1) \\ h_1(t-1)\cdot q_b(t-2) \\ h_2(t-1)^2\cdot h_2(t-2)^2 \\ h_2(t-1)\cdot q_b(t-2) \\ h_2(t-2)\cdot q_b(t-2) \\ q_a(t-1)\cdot q_b(t-1) \\ q_a(t-1)\cdot q_b(t-2) \end{bmatrix} \tag{3.5}$$

and the offsets of the corresponding outputs, d_1 and d_2, are 0.335608 and 0.409923 respectively.

3.2 Hammerstein-Wiener Models

The Hammerstein-Wiener model is a block-structured model where a nonlinear block both precedes and follows a linear dynamic system and this model can be used in order to identify a nonlinear system [33]. The linear block is a discrete transfer function that represents the dynamic component of the model [3]. For the SISO case, the block diagram of a Hammerstein-Wiener model is illustrated in Figure 3.4.

In the following parts, first, the structure of the Hammerstein-Wiener model is expressed. Then, many Wiener models with different orders are estimated for the identification procedure. Afterwards, the estimated models are validated by analyzing the performance of corresponding models based on the NRMSE quality metric. In the validation phase, some Hammerstein and Hammerstein-Wiener models are also tested so as to compare them with the candidate Wiener model. Finally, the performance of the candidate model is improved by modifying the lower and upper saturation limits on the outputs.

3.2.1 Model Structure

The block diagram of a Hammerstein-Wiener model is represented as

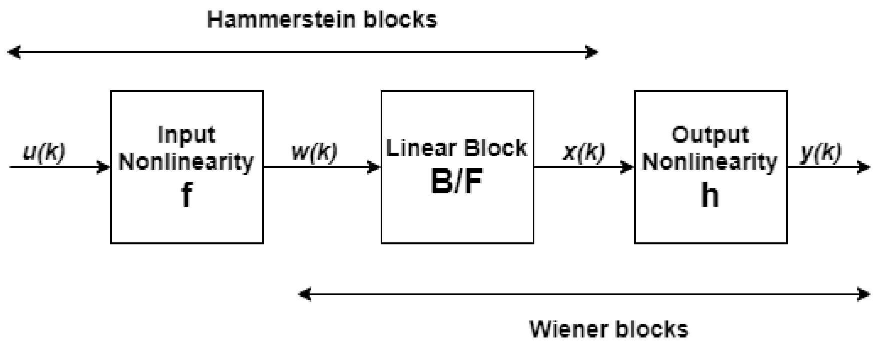

Figure 3.4: Scheme of a Hammerstein-Wiener model in the SISO case

where $u(k)$ and $y(k)$ are the input and output of the system respectively; $w(k)$ is the output of the input nonlinearity block; $x(k)$ is the input of the output nonlinearity block and $w(k)$ and $x(k)$ are called as internal variables. Also, f is the nonlinear function that transforms the input data $u(k)$ as $w(k) = f(u(k))$; h is the nonlinear function that maps the output of the linear block $x(k)$ to the system output $y(k)$ as $y(k) = h(x(k))$; B/F is the linear transfer function that transforms $w(k)$ as $x(k) = (B/F)w(k)$ [3] [24].

An identified model does not need to have both input and output non-linearities in the structure. When a model contains only the input non-linearity f, then it becomes a Hammerstein model. Likewise, in the event that the model has only the output non-linearity h, then it is called a Wiener model.

The structure of the Hammerstein-Wiener model can be generalized as

$$\begin{aligned} w(k) &= f(u(k)) \\ x(k) &= F_1x(k-1) + ... + F_{n_f}x(k-n_f) \\ &\quad + B_1w(k-n_k) + ... + B_{n_b}w(k-n_b-n_k+1) \\ y(k) &= h(x(k)) \end{aligned} \tag{3.6}$$

where n_u and n_y are the number of inputs and outputs of the system respectively; F_1, F_2, ... $\in \mathbb{R}^{n_y,n_u}$ and B_1, B_2, ... $\in \mathbb{R}^{n_y,n_u}$ are the coefficient matrices; $u(k) \in \mathbb{R}^{n_u}$ represents the inputs at sampling time instant k; $y(k) \in \mathbb{R}^{n_y}$ is the outputs at sampling time instant k; $w(k) \in \mathbb{R}^{n_u}$ and $x(k) \in \mathbb{R}^{n_y}$ are the input and output of the linear block respectively; $f \in \mathbb{R}^{n_u}$ and $h \in \mathbb{R}^{n_y}$ are the nonlinear functions that transform the input data and the output data respectively; $n_k \in \mathbb{R}^{n_y,n_u}$ is the input-output delay; n_f and n_b are the orders of the Hammerstein-Wiener model [3] [24].

3.2.2 Identification

In this section, all models are estimated in the simulation mode by using the unmodified datasets i.e. the estimation, validation and test data sets are not detrended. The models are obtained by means of the MATLAB System Identification Toolbox. The maximum number of iterations is determined as 50. In the ordinary way, the input-output delay n_k is selected as 1 sample time for all the transfer functions of the system and it is kept constant during the simulations for the sake of simplicity.

Since PRBS signals with varying amplitude are employed as input signals, input nonlinearities for the inputs q_a and q_b can be considered as *absent* or *unitgain* in the MATLAB System Identification toolbox. In other words, the input PRBS signals can be assumed as linear signals because their amplitudes do not change non-linearly with respect to time. Moreover, the non-linearity is chosen as *saturation* for both outputs in the estimations. This is an assumption and other possibilities will also be tested in Section 3.2.3. Hence, the models in the estimation phase are called Wiener models.

It is also important to remark that Output-Error models and Hammerstein-Wiener models are similar because the linear block is composed of the polynomials $B(q)$ and $F(q)$ for Hammerstein-Wiener models. In Section 2.3.3, the OE model with $n_b = 2$, $n_f = 2$, and $n_k = 1$ has been selected. Here, Wiener models could have been estimated with same orders; nonetheless, 100 Wiener models are obtained by trying different orders $n_b = [i\ i;\ i\ i]$ and $n_f = [j\ j;\ j\ j]$ iteratively where i, j = 1, 2, 3, ..., 10.

The Wiener models that provide the best fit percentage on the estimation data set are demonstrated in the first 16 rows of Table 3.7. In this table, the orders refers to matrices, such as $n_b = 2$ corresponds to $n_b = [2\ 2;\ 2\ 2]$. The Wiener models with different orders indicated in Table 3.7 are the candidates; however, as it has been mentioned in the previous sections, the model order can be decided only after the corresponding models are tested on the validation data set.

In this table, the normalized Akaike's Information Criterion (nAIC) of the considered models are also included because nAIC value gives clue about the performance

n_b	n_f	n_k	$Fit_{est}\%$ $[h_1;\ h_2]$	$nAIC$
2	2	1	[80.42; 85.78]	-12.60
9	9	1	[78.13; 84.90]	-11.88
2	3	1	[78.62; 83.55]	-12.03
4	2	1	[78.73; 83.08]	-11.86
3	2	1	[78.26; 83.27]	-12.10
9	2	1	[76.94; 83.33]	-12.11
5	8	1	[78.41; 81.00]	-11.64
5	9	1	[75.95; 83.04]	-11.55
10	1	1	[76.45; 82.51]	-11.78
4	3	1	[74.95; 83.61]	-11.65
5	3	1	[73.89; 84.48]	-11.63
3	3	1	[75.39; 82.69]	-11.79
4	6	1	[75.69; 82.14]	-11.56
9	1	1	[75.97; 81.73]	-11.78
7	7	1	[76.02; 81.31]	-11.40
5	5	1	[72.40; 84.91]	-11.64
6	8	1	[70.32; 78.94]	-10.56
9	3	1	[60.22; 76.76]	-9.84
9	8	1	[74.80; 33.10]	-8.56
1	9	1	[54.17; 35.47]	-7.62

Table 3.7: Fit% of different Wiener models on the estimation dataset

of the corresponding models on different datasets. The models with greater nAIC generally fit the data less in the validation phase. Therefore, the models in the last 4 rows of Table 3.7 are expected to have less fit percentage on the validation and test data sets.

3.2.3 Validation

The estimated Wiener models with various orders are validated by the help of different data set in the simulation mode. The residuals of all the estimated models are strongly correlated with the auto-correlation function i.e. all models fail the whiteness test. On the contrary, the residuals of all the acquired models are totally uncorrelated with the cross-correlation function i.e. all residuals are inside the confidence interval; therefore, all estimated Wiener models pass the independence test. It is seen that the residue correlation characteristics of Output-Error models and Hammerstein-Wiener models are very similar.

The models which fit the validation data the most are shown in Table 3.8. The residue correlations are not included in this table because all of the corresponding models perform same in the whiteness and the independence tests. In the table, $Fit_{val}\%$, $Fit_{test_1}\%$, and $Fit_{test_2}\%$ are the fit percentages of the outputs h_1 and h_2 of the considered Wiener models on the validation, the first test and the second test data set respectively. The obtained results become more reliable via test datasets. Furthermore, the average of $Fit_{val}\%$, $Fit_{test_1}\%$, and $Fit_{test_2}\%$ are reported as $Fit_{avg}\%$ in Table 3.8. According to this table, the obtained model with the orders $n_b = 2$, $n_f = 2$, $n_k = 1$ has the second highest fit% on the validation data set and the considered model fits the datasets the most on the average.

n_b	n_f	n_k	$Fit_{val}\%$ $[h_1; h_2]$	$Fit_{test_1}\%$ $[h_1; h_2]$	$Fit_{test_2}\%$ $[h_1; h_2]$	$Fit_{avg}\%$
4	2	1	[87.00; 67.49]	[69.99; 78.67]	[84.46; 75.75]	77.23
2	**2**	**1**	**[79.23; 70.66]**	**[84.30; 83.20]**	**[86.66; 81.14]**	**80.87**
6	7	1	[78.91; 66.14]	[69.03; 73.62]	[76.32; 65.09]	71.52
9	10	1	[84.00; 56.53]	[75.88; 78.73]	[74.09; 75.69]	74.15
9	2	1	[89.46; 44.00]	[68.57; 78.36]	[86.93; 74.37]	73.62
3	10	1	[70.91; 62.23]	[78.41; 85.58]	[69.69; 74.19]	73.50
5	10	1	[79.84; 53.00]	[72.84; 74.72]	[74.00; 75.53]	71.66
4	3	1	[75.93; 56.79]	[72.58; 80.88]	[81.98; 75.85]	74.00
7	7	1	[80.91; 51.61]	[77.79; 73.86]	[83.18; 75.20]	73.76
3	3	1	[75.13; 56.33]	[78.83; 76.61]	[81.87; 77.26]	74.34
3	2	1	[61.86; 66.04]	[73.94; 78.49]	[74.47; 80.25]	72.51
4	4	1	[74.14; 49.60]	[63.23; 74.56]	[71.51; 73.97]	67.84
5	5	1	[67.52; 54.46]	[81.32; 74.62]	[80.81; 68.31]	71.17
2	9	1	[68.48; 51.55]	[82.97; 68.19]	[78.15; 58.70]	68.01
2	10	1	[68.54; 51.17]	[77.85; 69.67]	[59.16; 56.90]	63.88

Table 3.8: Fit% of the estimated Wiener models on the validation and test datasets

Despite the fact that the Wiener model with the orders $n_b = 2$, $n_f = 2$, $n_k = 1$ and with the *saturation* nonlinearity choice for both outputs can be selected, other Wiener, Hammerstein, and Hammerstein-Wiener models are taken into account for the sake of completeness. For this reason, *unitgain*, *saturation* and *deadzone* are tried for each input and each output as an iterative process and 81 different models with the same polynomial orders ($n_b = 2$, $n_f = 2$, $n_k = 1$) are obtained by means of the MATLAB System Identification toolbox. In Table 3.9, the estimated models with better performance are reported based on the average fit percent of the corresponding models on the validation and test data sets. In this table, for each input and output, U, S, D stand for absence of non-linearity, saturation non-linearity and dead-zone non-linearity respectively.

From Table 3.9, it is seen that the performance of the previously validated model is the second best model with regard to the fit percent on the average ($Fit_{avg}\%$). The average fit percent of the corresponding model is very close to the model which provides the best performance on the average. Besides, the model in the second row performs better on the validation data set as compared to the model in the first row of Table 3.9. As it has been mentioned before, validation dataset has more importance in this work; thus, the Wiener model with saturation nonlinearities for both outputs and $n_b = 2$, $n_f = 2$, $n_k = 1$ for all the transfer functions of the system has been determined as the candidate model.

The simulated response diagram of the candidate Wiener model in comparison

q_a	q_b	h_1	h_2	$Fit_{val}\%$ $[h_1; h_2]$	$Fit_{test_1}\%$ $[h_1; h_2]$	$Fit_{test_2}\%$ $[h_1; h_2]$	$Fit_{avg}\%$
S	U	S	S	[77.43; 70.53]	[86.15; 83.28]	[87.56; 81.05]	81.00
U	**U**	**S**	**S**	[**79.23**; **70.66**]	[**84.30**; **83.20**]	[**86.66**; **81.14**]	**80.87**
U	D	S	U	[87.86; 71.06]	[81.03; 85.43]	[83.31; 73.80]	80.42
S	S	S	U	[86.78; 70.33]	[84.77; 87.55]	[81.85; 70.93]	80.37
U	S	D	U	[81.16; 73.48]	[82.02; 87.23]	[84.91; 72.90]	80.28
D	D	D	D	[80.79; 67.01]	[82.01; 81.50]	[83.36; 79.65]	79.05
U	U	S	U	[82.72; 67.56]	[80.93; 82.50]	[81.55; 72.30]	77.93
U	U	D	D	[77.07; 63.83]	[85.97; 80.91]	[82.79; 76.78]	77.79
U	U	D	U	[80.67; 66.83]	[81.27; 81.88]	[82.87; 71.88]	77.57
U	U	S	D	[87.38; 58.54]	[86.63; 69.34]	[86.81; 76.38]	77.51
U	U	U	D	[81.20; 67.70]	[76.61; 81.77]	[78.91; 78.46]	77.44
U	D	D	U	[81.24; 68.25]	[71.53; 84.18]	[81.66; 77.78]	77.44
S	S	S	S	[83.93; 70.08]	[79.44; 84.44]	[73.91; 72.49]	77.38
D	D	U	U	[82.34; 64.25]	[78.57; 79.52]	[82.72; 75.84]	77.21
U	U	U	S	[75.75; 66.45]	[78.52; 82.46]	[79.18; 80.88]	77.21

Table 3.9: Fit% of the estimated models with different input-output nonlinearities and with the orders $n_b = 2$, $n_f = 2$, $n_k = 1$

with the validation data set is demonstrated in Figure 3.5.

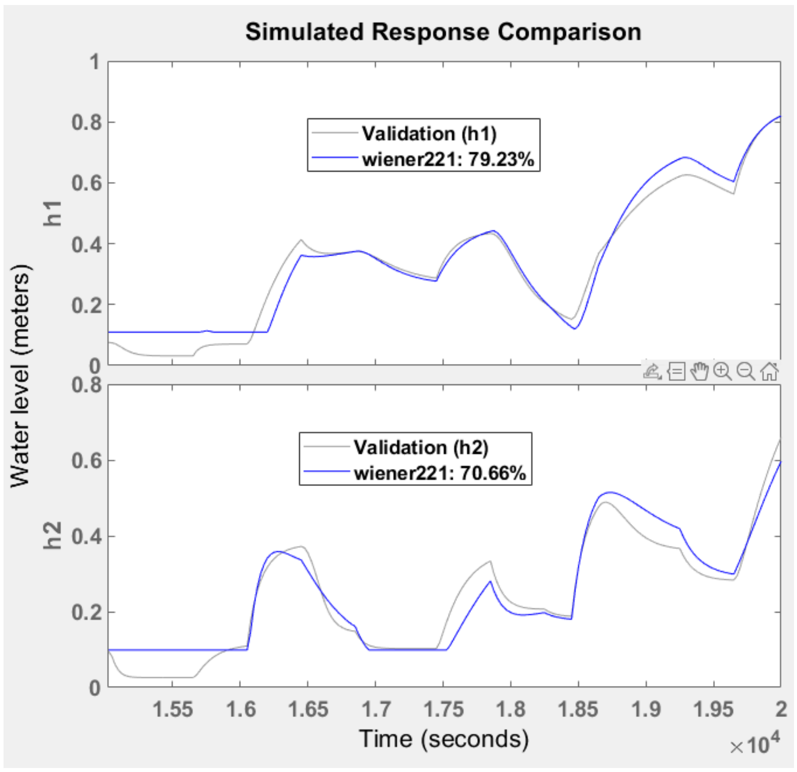

Figure 3.5: The candidate Wiener model vs. validation dataset

From Figures 2.14 and 3.5, it is seen that the performance of the candidate Wiener model on the validation dataset is slightly worse than the selected Output-Error model (75.67% for the OE model, 74.95% for the Wiener model on the average). However, the considered Wiener model fits the test data sets more than the corresponding OE model. From Tables 2.8 and 3.8, it has been found that the average fit percentages of the selected OE and the candidate Wiener models on two different test datasets are 78.46% and 83.83% respectively.

The performance of the candidate Wiener model can be enhanced by adjusting the saturation limits on both outputs. After several trials, the lower saturation limits on the first output h_1 and on the second output h_2 have been determined as 0.0636 and 0.0939 respectively. Besides, the upper saturation limits on both outputs of the candidate model have been removed for the calibrated model. The simulated response diagram of the calibrated Wiener model on the validation data set is shown in Figure 3.6. Additionally, the fit percent of the considered model on the first and the second test datasets are [80.11; 81.95] and [81.89; 80.84] respectively.

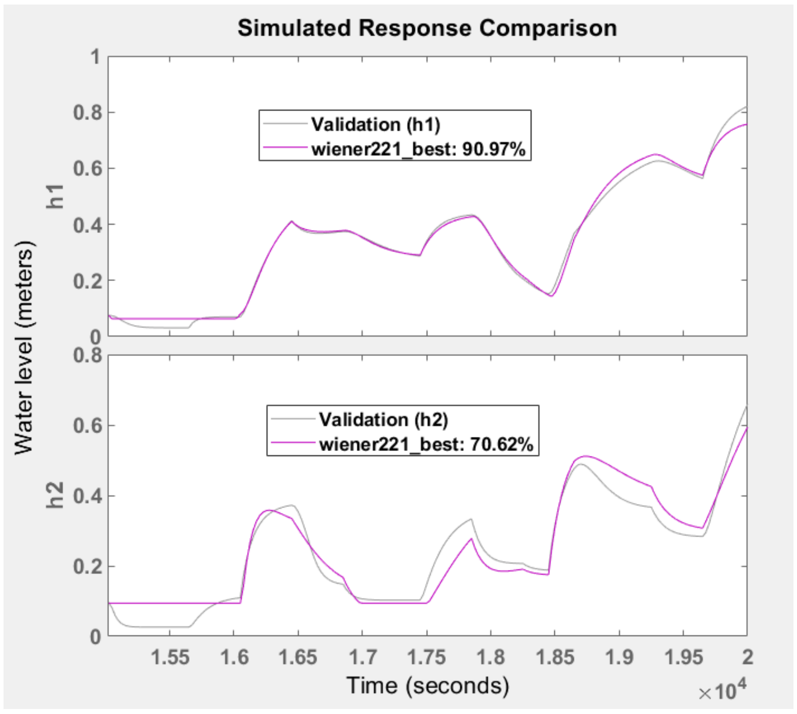

Figure 3.6: The performance of the calibrated Wiener model on the validation dataset

The results of the whiteness and the independence tests for the candidate models are shown in Figure 3.7. As it has been pointed out at the beginning of this section, the residuals of all the estimated Hammerstein-Wiener models are highly correlated with regard to the auto-correlation function; hence, any obtained models could not pass the whiteness test. On the other hand, all of the estimated models could pass the independence test owing to the fact that the residuals of the corresponding models are inside the confidence interval at any lag sample.

Since the overall performance of the candidate Wiener model has been increased (from 80.87% to 81.06% on the average) through the calibration procedure, there is a

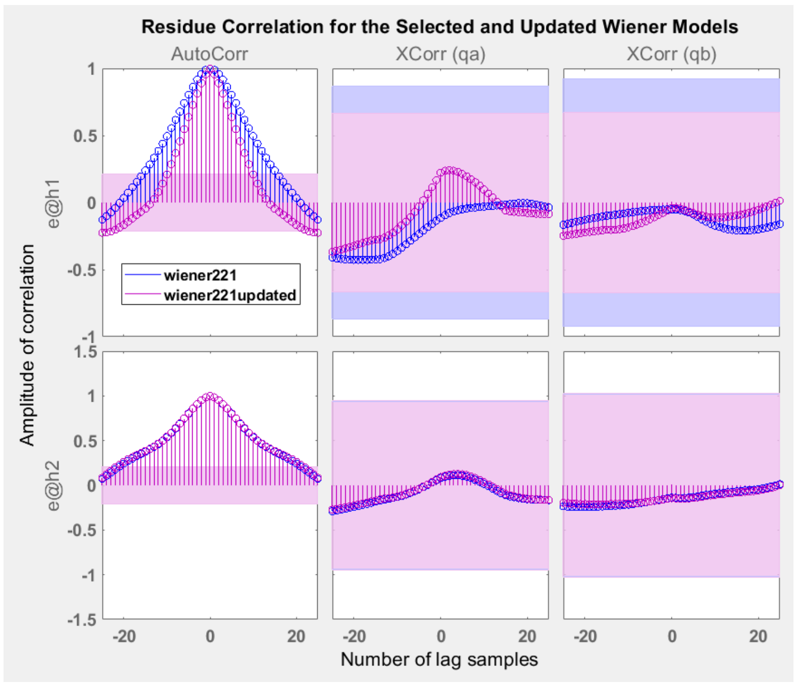

Figure 3.7: Residue correlations of the candidate Wiener models

risk of overfitting effect. Different test datasets that have not been used before, now, are employed in order to reduce the risk of overfitting. The fit percent comparison of the candidate Wiener models on 14 different test datasets is shown in Table 3.10.

Test Set No.	***Unmodified Model***	***Calibrated Model***
Data 3 Avg.Fit%	78.81%	81.81%
Data 4 Avg.Fit%	42.76%	45.00%
Data 5 Avg.Fit%	81.77%	81.91%
Data 6 Avg.Fit%	78.61%	80.40%
Data 7 Avg.Fit%	79.55%	83.15%
Data 8 Avg.Fit%	88.76%	85.77%
Data 9 Avg.Fit%	80.06%	84.77%
Data 10 Avg.Fit%	80.75%	84.04%
Data 11 Avg.Fit%	83.23%	84.99%
Data 12 Avg.Fit%	80.21%	86.36%
Data 13 Avg.Fit%	78.96%	84.16%
Data 14 Avg.Fit%	85.89%	86.17%
Data 15 Avg.Fit%	41.63%	58.05%
Data 16 Avg.Fit%	79.29%	87.30%
Overall Avg.Fit%	**75.73%**	**79.56%**
Top 12 Avg.Fit%	**81.32%**	**84.23%**

Table 3.10: Fit% comparison of the candidate Wiener models on the new test datasets

As it is seen from Table 3.10, the calibrated (updated) Wiener model fits the data more than the original candidate model for all of the test sets except one (Test set 8). The overall fit percents of the unmodified and calibrated Wiener models are 75.73% and 79.56% respectively. Both Wiener models could not fit enough the 4th and the 15th test data sets. When these two test data sets are discarded, the average fit percents of the corresponding Wiener models on the remaining datasets are 81.32% and 84.23% respectively. In brief, the calibrated Wiener model performs better on the average; therefore, the corresponding model has been selected.

By recalling the equation 3.6, the structure of the selected Wiener model can be written as:

$$\begin{aligned}
y_1(k) &= +1.8367y_1(k-1) - 0.8371y_1(k-2) + 1.6411y_2(k-1) \\
&\quad - 0.6668y_2(k-2) + 88.7083q_a(k-1) - 88.7157q_a(k-2) \\
&\quad + 4.4017q_b(k-1) + 28.8148q_b(k-2) \\
h_1(k) &= f_1(y_1(k)) \\
\\
y_2(k) &= +1.8608y_1(k-1) - 0.8685y_1(k-2) + 1.8135y_2(k-1) \\
&\quad - 0.8137y_2(k-2) + 49.8207q_a(k-1) - 40.7712q_a(k-2) \\
&\quad + 146.6556q_b(k-1) - 146.6925q_b(k-2) \\
h_2(k) &= f_2(y_2(k))
\end{aligned} \tag{3.7}$$

where f_1 and f_2 are the nonlinear functions that represent the saturation. Lower saturation limits on the outputs h_1 and h_2 are 0.0636 and 0.0939 respectively. For the selected Wiener model, no upper saturation limits have been determined. The shape of the saturation function for each output is shown in Figure 3.8.

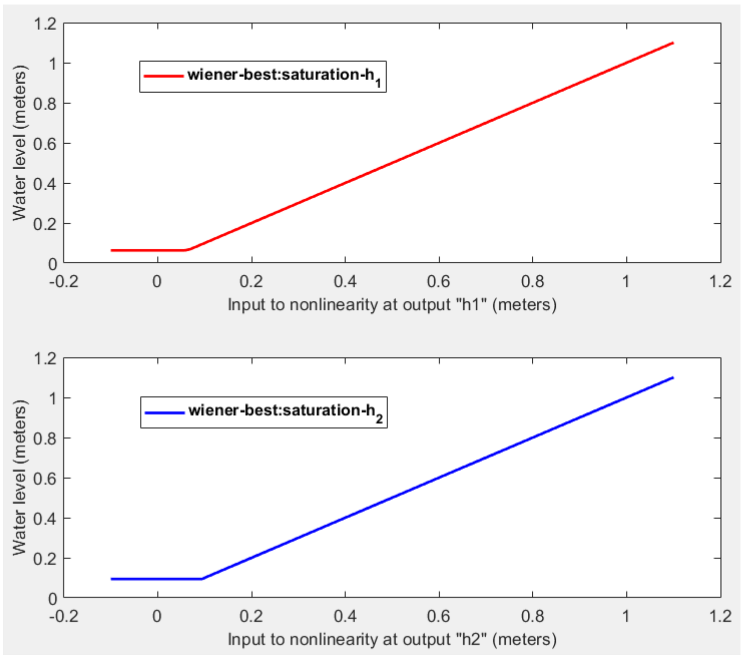

Figure 3.8: Saturation non-linearity plots for both outputs

In conclusion, it is important to note that the selected Wiener model contains only nonlinear saturation function for both outputs. During the simulation experiments, some other nonlinearity functions have also been considered for the inputs and the outputs; nevertheless, it is not possible to take into account more than one type of nonlinearity at the same time. This is a limitation of MATLAB.

3.3 Neural NARX Models

The Neural NARX (NNARX) model structure is a neural non-linear autoregressive model with exogenous inputs. The NNARX models are widely used in the process industry for identifying non-linear plants.

In the following parts of this section, the structure of the Neural NARX model is stated. Then, the training procedure is explained including the data division of the estimation dataset, the Levenberg-Marquardt algorithm, and the data preprocessing. Afterwards, an example of the simulated response diagrams for the outputs on the estimation dataset is provided, Pearson correlation coefficient is defined and then an example of regression analysis is indicated. In the validation phase, first, different orders are tried for training NNARX models with 10 basic neurons in the hidden layer. After determining the candidate orders, the considered orders are tested for the models with different number of neurons. Thereafter, some NNARX models with the candidate orders and the number of neurons are estimated. All of these models have different weights and biases. Then, performance of the obtained NNARX models are compared on the validation, the first and the second test datasets. Ultimately, the candidate NNARX models are tested on the different datasets which have never been utilized before in order to reduce the risk of overfitting and the model which provides the best performance is selected.

3.3.1 Model Structure

Recall that the general structure of a NARX model

$$y(k) = F(y(k-1), ..., y(k-n_a), u(k-n_k), u(k-1-n_k), ..., u(k-n_b-n_k+1)) \tag{3.8}$$

where n_u and n_y are the number of inputs and outputs of the system respectively, $u(k) \in \mathbb{R}^{n_u}$ represents the inputs at sampling time instant k, $y(k) \in \mathbb{R}^{n_y}$ is the outputs at sampling time instant k, n_a and n_b are the orders of the NARX model and they represent the number of past output and past input terms used in order to estimate the current output respectively, n_k refers to the input-output delay, and F is the non-linear regression function [3] [24].

The NNARX model structure is the most complicated one as compared to the model structures employed in the previous sections. In the Neural NARX models, a feedforward neural network constitutes the nonlinear regression function. Since only one hidden layer is utilized in order to estimate Neural NARX models in the following phases, the structure of the NARX neural network can be compactly written as [34] [35] [36]

$$\begin{aligned} y(k) = F_o(b_o + w_{ho}F_h(&A_1 y(k-1) + ... + A_{n_a} y(k-n_a) \\ &+ B_1 u(k-n_k) + ... + B_{n_b} u(k-n_b-n_k+1) + b_h)) \end{aligned} \tag{3.9}$$

where F_o and F_h are the functions of the output and hidden layers respectively; $b_o \in \mathbb{R}^{n_y,1}$ and $b_h \in \mathbb{R}^{N_h,1}$ are the biases of the output and hidden layers respectively; $w_{ho} \in \mathbb{R}^{n_y,N_h}$ denotes the weights from the hidden layer to the output layer; A_1, A_2, ... $\in \mathbb{R}^{N_h,n_y}$ and B_1, B_2, ... $\in \mathbb{R}^{N_h,n_u}$ correspond to the weights from the output feedback layer to the hidden layer and from the network inputs to the hidden layer respectively. Here, N_h represents the number of basic neurons in the hidden layer.

In this work, the tangent sigmoid transfer function ($tanh$) and the linear transfer function are used for the hidden and the output layers respectively. Thus, $F_h = tanh$ and $F_o = 1$. The ranges of the hidden layer and the output layer transfer functions are $[-1, +1]$ and $(-\infty, +\infty)$ respectively.

3.3.2 Identification

All models are trained in the simulation mode by using given past inputs and the simulated past outputs of the estimation dataset in order to estimate the current outputs. The models are obtained by means of the Neural Net Time Series Application in the MATLAB Deep Learning Toolbox.

In this thesis, all Neural NARX models are trained by employing the Levenberg-Marquardt (LM) algorithm. This algorithm has been designed to minimize functions which are the sums of squares of other nonlinear functions [37]. Since the performance index is selected mean-square error (MSE), this optimization algorithm is pretty convenient for training NNARX models. By implementing LM algorithm, fast and stable convergence can be achieved providing that the size of training problem is not too large [38]. In this section, the performance comparison of the estimated NNARX models is made based on "fit percent", which is related to NRMSE quality metric i.e. the fit percent value is derived from MSE value.

The maximum number of iterations for the training procedure is chosen as 1000. As it always has been, the input-output delay n_k is selected as 1 sample time and it is kept constant during the simulations for the sake of simplicity. The models are acquired by trying different orders $n_a = [i\ i;\ i\ i]$ and $n_b = [j\ j;\ j\ j]$ where $i, j = 1, 2, 3, ..., 10$. For each order pairs, the corresponding models are trained 20 times with different weights and biases so as to obtain robust results for the considered orders. During the training procedure, just one hidden layer has been used and the number of basic neurons in the hidden layer are determined as 10 initially. In the validation phase (see Section 3.3.3), some other number of neurons are also taken into account for the candidate model orders.

In order to get better results during the training phase, the input-output data are pre-processed by mapping the minimum and maximum values to -1 and $+1$ respectively. Moreover, the estimation dataset has been divided into three parts randomly by means of the MATLAB Deep Learning toolbox so as to enhance the generalization of the trained NARX neural networks. One part is composed of training points (70% of the estimation dataset) and the other ones contain validation (15%) and test points (15%) respectively. When the error on the validation set begins to increase, it means that the network starts overfitting the data. In this work, if the validation error increases for 6 consecutive iterations, then the training procedure is stopped; the weights and biases at the minimum of the validation error are determined. This technique is called "early stopping" and it is utilized to avoid overfitting [39]. An example of performance comparison of the train, validation and test data points are shown in Figure 3.9.

In this figure, it is seen that the validation error begins to rise at the epoch 270; hence the "early stopping" technique gets involved and the training procedure is stopped at this epoch.

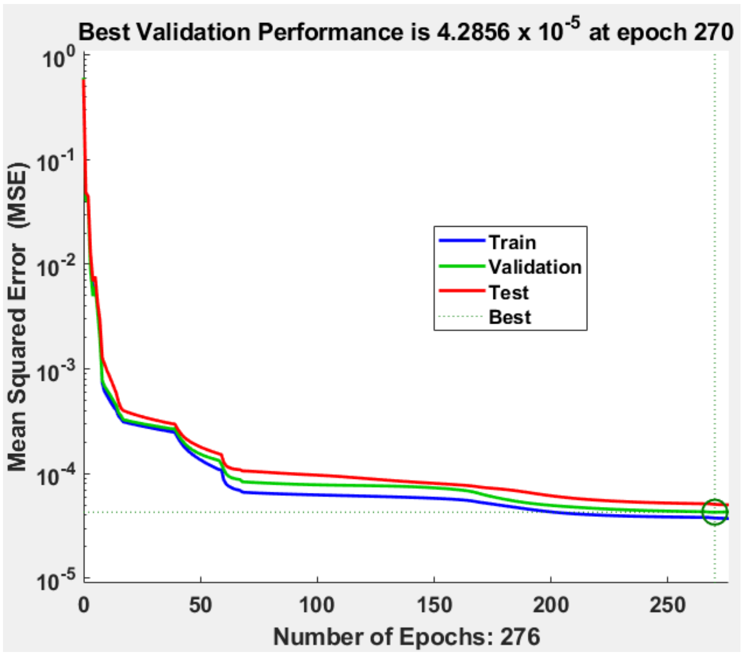

Figure 3.9: Performance comparison of the train, validation and test data points

An exemplification for the comparison of the measured and the simulated output responses on the estimation dataset are demonstrated in Figure 3.10 and Figure 3.11 respectively. In these figures, the randomly constituted train, validation and test

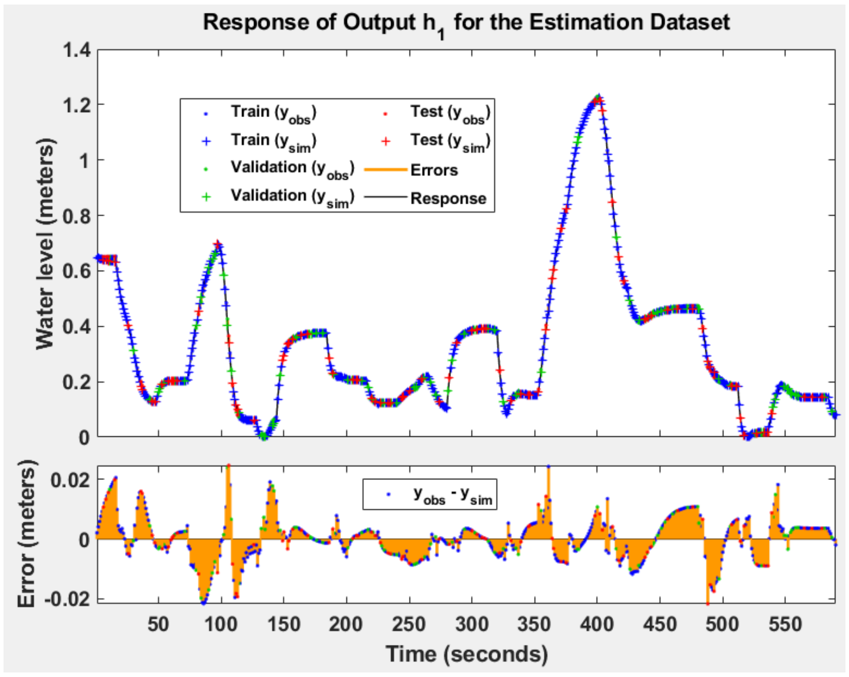

Figure 3.10: Simulated response diagram for the output h_1 on the estimation dataset

data points of the estimation dataset are shown in different colors. It is seen that the

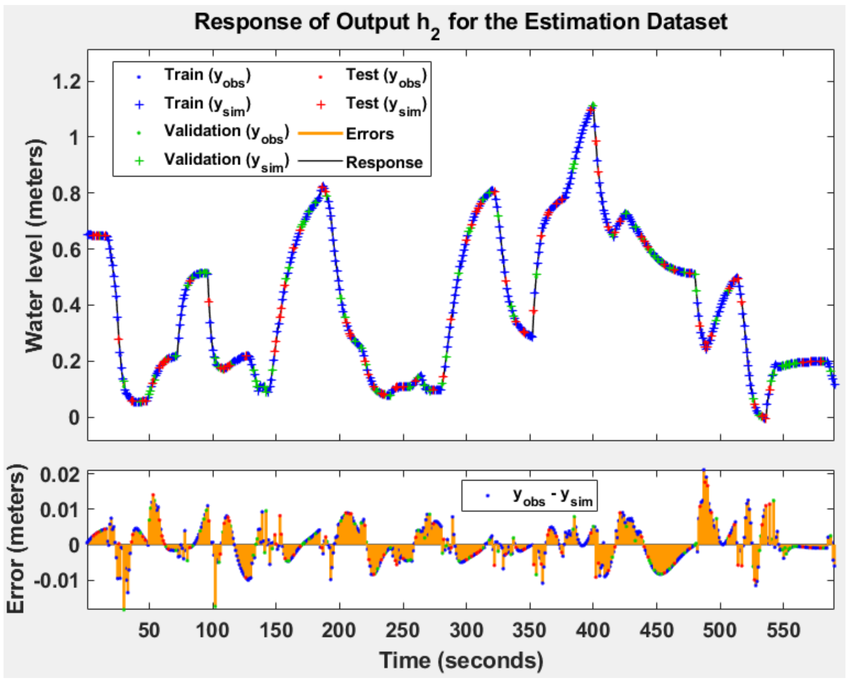

Figure 3.11: Simulated response diagram for the output h_2 on the estimation dataset

corresponding model fits very well the estimation dataset for both outputs; however, the trained NNARX models have to be tested on the different datasets and this will be done in the validation phase.

The regression analysis can also be performed in order to investigate the trained Neural NARX models in more detail. In the regression analysis, the simulated response of the trained neural network and the corresponding measured (observed) outputs are compared.

The regression analysis is related to the correlation coefficient. The correlation coefficient (R-value) of two random variables is a measure of their linear dependence [39]. In the event that R is equal to 1, there is perfect correlation between the measured and the simulated outputs. The Pearson correlation coefficient R is

$$R = \rho(y_{obs}, y_{sim}) = \frac{1}{N-1}\sum_{i=1}^{N}\left(\frac{y_{obs}(i)-\mu_{y_{obs}}}{\sigma_{y_{obs}}}\right)\left(\frac{y_{sim}(i)-\mu_{y_{sim}}}{\sigma_{y_{sim}}}\right) \tag{3.10}$$

where y_{obs} and y_{sim} are the measured and simulated outputs respectively, N is the number of observations, $\mu_{y_{obs}}$ and $\mu_{y_{sim}}$ are the mean value of the observed and simulated outputs respectively, $\sigma_{y_{obs}}$ and $\sigma_{y_{sim}}$ are the standard deviation of the measured and simulated outputs respectively. Alternatively, the equation 3.10 can be represented as

$$R = \rho(y_{obs}, y_{sim}) = \frac{cov(y_{obs}, y_{sim})}{\sigma_{y_{obs}}\sigma_{y_{sim}}} \tag{3.11}$$

where $cov(y_{obs}, y_{sim})$ is the covariance of the observed and the simulated outputs. An example of the regression analysis is shown in Figure 3.12.

In Figure 3.12, there is an indication of the good fit due to the fact that all the

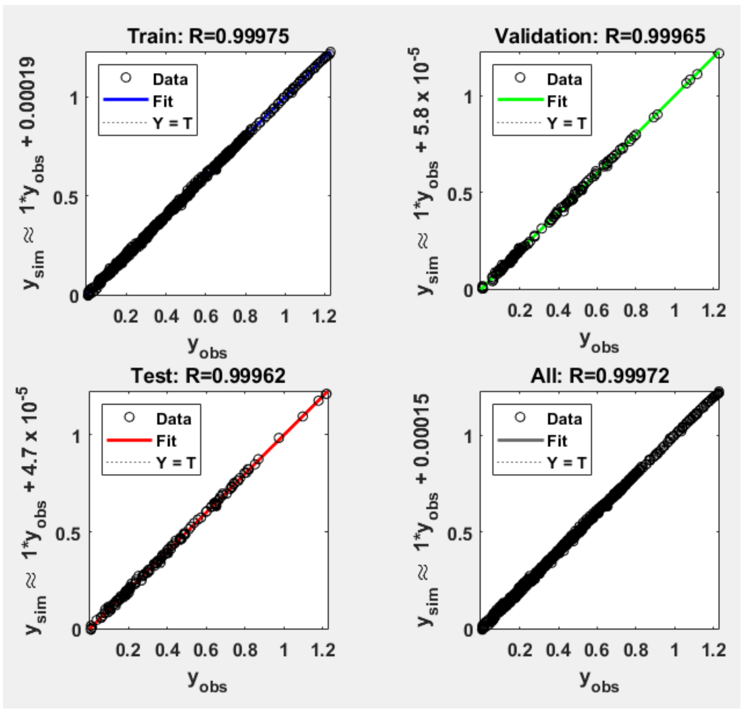

Figure 3.12: Linear regression plot of the simulated outputs on the estimation dataset

R values for the train, validation and test data points of the estimation dataset are very close to 1. In the case of training data points, for instance, the best linear fit is $y_{sim} \approx y_{obs} + 0.00019$. It means that the slope and the y-intercept of the best linear regression relating observed outputs to simulated network outputs is 1 and 0.00019 respectively.

3.3.3 Validation

The trained Neural NARX models with different orders are validated by using different datasets in the simulation mode. The 15 best performing NNARX models on the validation, the first and the second datasets are shown in the table below. For each order, 20 NNARX models with different weights and biases have been trained and then their average performance has been computed so as to gain more reliable results. In Table 3.11, $Fit_{avg}\%$ corresponds to the average fit percent of the obtained models on the validation and test datasets; $\#\ of\ Above$ 85% refers to the number of the trained NNARX models which fit validation and test data sets more than 85% on the average. For example, on the first row of this table, 11 out of 20 trained models with the orders $n_a = 4$, $n_b = 4$, $n_k = 1$ could fit the data more than 85 percent on the average.

n_a	n_b	n_k	*# of Neurons*	$Fit_{avg}\%$	*# of Above 85%*
4	4	1	10	72.88%	11
10	8	1	10	74.60%	10
7	6	1	10	69.05%	10
2	10	1	10	73.86%	9
8	7	1	10	69.95%	9
3	10	1	10	69.32%	9
6	7	1	10	67.35%	8
1	10	1	10	77.57%	7
8	10	1	10	74.34%	7
8	9	1	10	70.89%	7
10	9	1	10	68.66%	7
3	9	1	10	69.38%	6
6	6	1	10	68.84%	6
6	5	1	10	66.66%	6
7	8	1	10	66.40%	6

Table 3.11: Average Fit% of the trained NNARX models on the validation and test datasets

At the beginning of the validation phase, the models only with 10 basic neurons in the hidden layer have been trained and tested. Now, the NNARX models with the orders specified in Table 3.11 are trained and validated by employing different number of basic neurons in the hidden layer for the sake of completeness. The 20 best performing ones are demonstrated in Table 3.12.

As it can be seen from Table 3.12, the performance of the acquired models with orders and number of neurons reported on the 1st, 4th and 6th rows are quite good and their complexity are less than the other models in general. Hence, from these rows, the candidate orders and number of basic neurons in the hidden layer have been determined.

After deciding the orders of the NNARX model structures, 6 different NNARX models which have $Fit_{avg}\%$ greater than 90% are determined as the candidate models in order to compare their performance based on NRMSE quality metric. The fit percents of the trained models on the validation, the first test, and the second test data sets are shown in Table 3.13.

In Table 3.13, 6 different NNARX models with different weights and biases have been obtained. According to this table, model 6 fits the validation data the most; model 5 fits the first test data the most; model 2 performs on the second test dataset the best. Furthermore, the sixth model could fit the data the most on the average (92.63%).

n_a	n_b	n_k	*# of Neurons*	Fit_{avg}%	*# of Above* **85%**
1	**10**	**1**	**5**	85.22%	12
3	10	1	15	79.75%	11
10	9	1	15	74.80%	11
4	**4**	**1**	**10**	72.88%	11
2	10	1	5	78.11%	10
4	**4**	**1**	**5**	75.22%	10
10	8	1	10	74.60%	10
7	6	1	10	69.05%	10
2	10	1	10	73.86%	9
8	7	1	10	69.95%	9
3	10	1	10	69.32%	9
3	9	1	5	68.10%	9
8	9	1	5	70.75%	8
6	7	1	10	67.35%	8
1	10	1	10	77.57%	7
7	8	1	15	74.70%	7
8	10	1	10	74.34%	7
8	9	1	10	70.89%	7
8	7	1	15	68.90%	7
10	9	1	10	68.66%	7

Table 3.12: Average Fit% of the trained NNARX models with candidate orders and different number of basic neurons in the hidden layer

Model#	n_a	n_b	n_k	*Neuron*	Fit_{val}%	Fit_{test_1}%	Fit_{test_2}%	Fit_{avg}%
1	1	10	1	5	[92.39; 90.47]	[87.23; 93.08]	[94.37; 90.45]	91.33
2	1	10	1	5	[90.44; 89.62]	[89.43; 91.09]	[92.94; 92.74]	91.04
3	4	4	1	5	[88.73; 90.30]	[87.91; 94.51]	[90.96; 91.77]	90.70
4	4	4	1	5	[86.24; 92.15]	[89.33; 92.85]	[89.25; 92.23]	90.34
5	4	4	1	10	[94.70; 94.63]	[89.42; 93.48]	[88.41; 93.31]	92.32
6	4	4	1	10	[96.60; 96.01]	[91.87; 89.20]	[88.58; 93.53]	92.63

Table 3.13: Fit% of the candidate NNARX models with different weights and biases on the validation, the first and the second datasets

Even though the 6th model seems as the top model among them, these six NNARX models should be tried on the test data sets which have never been used before in order to eliminate the risk of overfitting and improve the neural network generalization. There is the risk because these models have been acquired according to the condition $Fit_{avg}\% > 90$ during the training phase. The average fit percent of these models on 14 new test data sets are shown in Table 3.14.

Test Set No.	***Model* 1**	***Model* 2**	***Model* 3**	***Model* 4**	***Model* 5**	***Model* 6**
Data 3 Avg.Fit%	90.16%	90.45%	89.18%	85.90%	**91.67%**	21.70%
Data 4 Avg.Fit%	78.81%	84.68%	80.97%	84.59%	84.64%	**92.05%**
Data 5 Avg.Fit%	95.46%	97.13%	96.57%	96.27%	96.41%	**97.40%**
Data 6 Avg.Fit%	85.87%	**90.57%**	85.76%	85.69%	66.16%	82.45%
Data 7 Avg.Fit%	91.75%	94.90%	93.75%	93.44%	95.84%	**97.98%**
Data 8 Avg.Fit%	94.07%	94.47%	91.52%	94.70%	86.94%	**97.18%**
Data 9 Avg.Fit%	86.63%	**90.43%**	85.93%	85.39%	74.19%	87.75%
Data 10 Avg.Fit%	92.26%	91.18%	90.76%	90.13%	**95.05%**	24.13%
Data 11 Avg.Fit%	93.18%	93.01%	91.19%	90.79%	**93.20%**	93.01%
Data 12 Avg.Fit%	91.90%	91.27%	86.13%	89.27%	**93.35%**	92.85%
Data 13 Avg.Fit%	94.33%	94.32%	93.41%	95.65%	93.81%	**96.19%**
Data 14 Avg.Fit%	91.14%	90.99%	89.21%	91.11%	**94.49%**	46.90%
Data 15 Avg.Fit%	89.25%	90.88%	91.09%	92.41%	95.50%	**95.71%**
Data 16 Avg.Fit%	91.73%	94.53%	92.08%	93.94%	**96.27%**	95.22%
Overall Avg.Fit%	90.46%	**92.06%**	89.82%	90.66%	89.82%	80.03%

Table 3.14: Average Fit% of the candidate NNARX models with different weights and biases on the new test datasets

From Table 3.14, it is seen that the second model could fit 2 out of 14 different data the most; the Model 5 and the Model 6 have been able to fit 6 out of 14 different data the most. In the case of head-to-head comparison of Model 2 and Model 5, it has been observed that the fifth model surpasses the second one for 8 different test datasets whereas the 2nd model fits 6 different datasets more than the 5th model. Nevertheless, the fifth and the sixth models could not fit some of the test datasets sufficiently well. On the contrary, the Model 2 has been able to fit all of the test datasets more than 90% except for the 4th test data set and this model could also fit the data the most on the average (92.06%). Therefore, the Model 2 with the orders $n_a = 1$, $n_b = 10$, $n_k = 1$ and 5 neurons in the hidden layer has been selected among the candidate NNARX models. The simulated response diagram of the selected NNARX model on the validation data set is demonstrated in Figure 3.13.

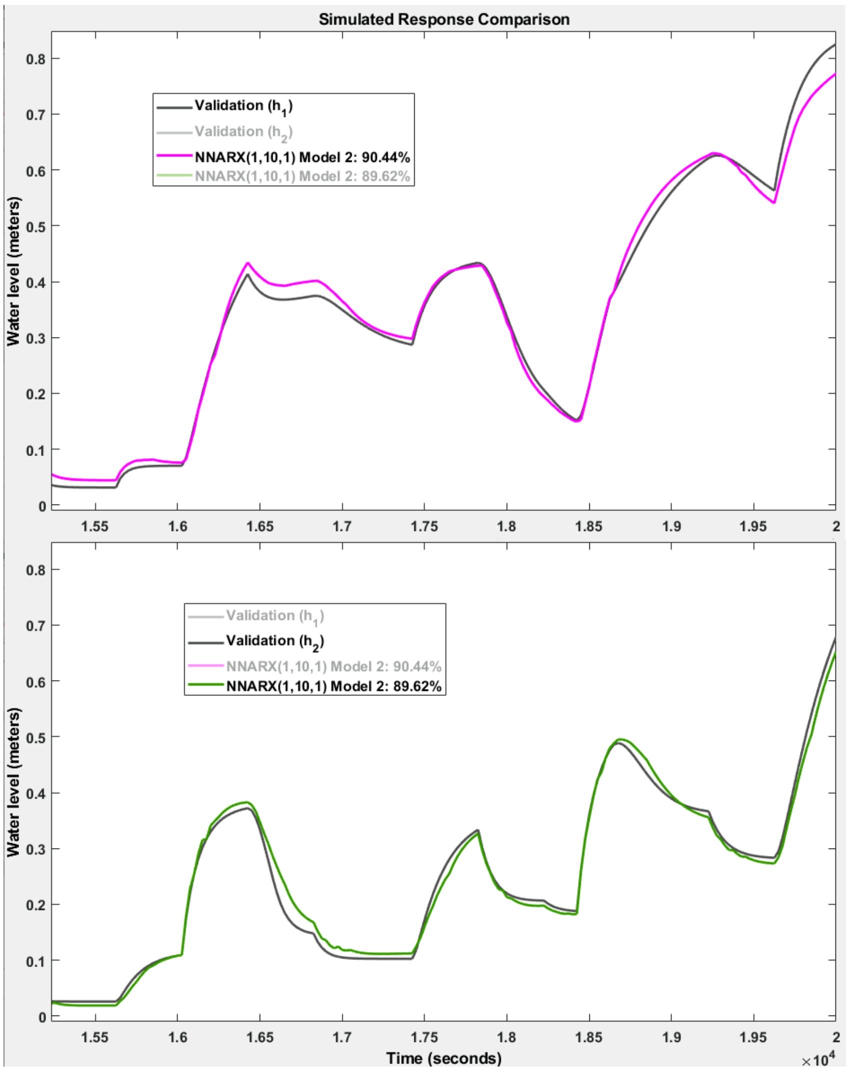

Figure 3.13: The selected Neural NARX model vs. validation dataset

By recalling the equation 3.9, the structure of the selected Neural NARX model can be written. The weight matrices $w_{oh} \in \mathbb{R}^{N_h,(n_a \times n_y)}$ for the weights going to the hidden layer from the output layer, $w_{ho} \in \mathbb{R}^{n_y,N_h}$ for the weights going to the output layer from the hidden layer, and $w_{ih} \in \mathbb{R}^{N_h,(n_b \times n_u)}$ for the weights going to the hidden layer from the network inputs are:

$$w_{oh} = \begin{bmatrix} -0.4014 & +0.4926 \\ -0.4414 & +0.0002 \\ +1.7540 & +0.0695 \\ +0.2567 & -0.4453 \\ +0.5046 & -0.3895 \end{bmatrix} \tag{3.12}$$

$$w_{ho} = \begin{bmatrix} +0.0037 & -2.1575 & +0.0690 & +0.0235 & -0.0211 \\ +0.0765 & -0.8270 & +0.0317 & -2.6866 & +0.6539 \end{bmatrix} \tag{3.13}$$

$$w_{ih}^{\mathsf{T}} = \begin{bmatrix} -0.0032 & -0.0231 & +0.4372 & -0.0229 & -0.1360 \\ +0.1867 & +0.0024 & +0.2081 & -0.0508 & -0.0388 \\ -0.3549 & -0.0014 & -0.1074 & -0.0285 & -0.0964 \\ -0.2590 & -0.0115 & +0.1856 & -0.0212 & -0.1521 \\ -0.0547 & -0.0019 & -0.1331 & -0.0203 & -0.0460 \\ -0.0810 & -0.0018 & +0.1431 & +0.0685 & +0.3338 \\ +0.0715 & +0.0010 & -0.0007 & -0.0120 & -0.0433 \\ -0.1131 & -0.0047 & +0.1584 & -0.0799 & -0.4173 \\ +0.0087 & -0.0006 & -0.0301 & -0.0005 & +0.0094 \\ -0.1573 & -0.0062 & -0.2315 & +0.0094 & +0.0430 \\ -0.0664 & +0.0026 & -0.0396 & -0.0193 & -0.0585 \\ -0.0683 & -0.0047 & +0.0753 & +0.0665 & +0.2940 \\ -0.0342 & +0.0026 & +0.1256 & -0.0016 & -0.0034 \\ +0.1582 & +0.0031 & +0.2715 & -0.0736 & -0.3347 \\ -0.1673 & +0.0017 & +0.0138 & -0.0122 & -0.0460 \\ -0.2055 & -0.0178 & -0.7877 & +0.0190 & +0.0768 \\ -0.4164 & +0.0001 & +0.0464 & -0.0112 & -0.0541 \\ -0.4472 & +0.0176 & +1.1144 & +0.0041 & +0.0047 \\ -0.8109 & +0.0018 & -0.0934 & -0.0302 & -0.0891 \\ +0.3686 & -0.0151 & -0.5483 & -0.0201 & -0.1013 \end{bmatrix} \tag{3.14}$$

And the biases of the hidden and the output layers are:

$$b_h = \begin{bmatrix} -2.5306 \\ +0.2323 \\ +1.5353 \\ +0.2650 \\ +0.5255 \end{bmatrix}, \qquad b_o = \begin{bmatrix} +0.4001 \\ +0.5891 \end{bmatrix} \tag{3.15}$$

From these matrices, one can remark that the selected NNARX model has 127 parameters (weights and biases). If the Model 4 from Table 3.14 had been chosen, the structure of the corresponding Neural NARX model would have been less complex (97 parameters). However, in this work, the Model 2 has been selected due to the fact that its performance on the new test datasets is better; in other words, the selected NNARX model has been able to fit 13 out of 14 test datasets more than 90%, whereas the fourth model could have fitted 10 out of 14 new test datasets more than 90%.

3.4 Comparison of the Selected Models

In this chapter, a Polynomial NARX, a Wiener, and a Neural NARX model have been selected after the identification and the validation phases. The average fit percent of two outputs on the validation dataset for the chosen Polynomial NARX, Wiener, NNARX models are 89.97%, 80.80%, and 90.03% respectively. That is to say, the performance of the selected Polynomial and Neural NARX models with regard to validation data are extremely close; however, the performance of the considered Wiener model is worse. In addition to the validation data set, 16 different test data sets have been utilized so as to test the estimated models and to obtain more reliable results. For all the nonlinear models obtained in this chapter, the validation, the first and the second test datasets have taken part in the identification procedure i.e. these data sets have been employed in order to improve the performance of the estimated models. And, the remaining 14 test data sets have only been used to test the acquired models. In this way, the risk of overfitting has been eliminated. The performance comparison of the selected nonlinear models is indicated in Table 3.15. In this table, *First* 3 *Avg.Fit%* represents the mean value of the fit percents of the corresponding models on the validation, the first and the second datasets. Similarly, *Last* 14 *Avg.Fit%* corresponds to the overall average fit percent of the considered models on the remaining test data sets.

Data Set No.	***Polynomial NARX***	***Wiener***	***Neural NARX***
Validation Avg.Fit%	89.97%	80.80%	**90.03%**
Test 1 Avg.Fit%	84.96%	81.03%	**90.26%**
Test 2 Avg.Fit%	84.66%	81.37%	**92.84%**
First 3 Avg.Fit%	86.54%	81.07%	**91.04%**
Test 3 Avg.Fit%	86.21%	81.81%	**90.46%**
Test 4 Avg.Fit%	81.35%	45.00%	**84.68%**
Test 5 Avg.Fit%	95.56%	81.91%	**97.13%**
Test 6 Avg.Fit%	84.78%	80.40%	**90.57%**
Test 7 Avg.Fit%	94.60%	83.15%	**94.90%**
Test 8 Avg.Fit%	93.27%	85.77%	**94.47%**
Test 9 Avg.Fit%	86.94%	84.77%	**90.43%**
Test 10 Avg.Fit%	89.55%	84.04%	**91.18%**
Test 11 Avg.Fit%	90.79%	84.99%	**93.01%**
Test 12 Avg.Fit%	89.82%	86.36%	**91.27%**
Test 13 Avg.Fit%	88.87%	84.16%	**94.32%**
Test 14 Avg.Fit%	86.64%	86.17%	**90.99%**
Test 15 Avg.Fit%	88.44%	58.05%	**90.88%**
Test 16 Avg.Fit%	91.65%	87.30%	**94.53%**
Last 14 Avg.Fit%	89.18%	79.56%	**92.06%**

Table 3.15: Performance comparison of the selected nonlinear models

As it can be seen from Table 3.15, the selected Neural NARX model outperforms for any datasets in comparison with the other chosen models. Not surprisingly, all the considered nonlinear models have been able to fit the datasets more than that of the linear ones according to Tables 2.11 and 3.15. Hence, the identified NNARX model can be employed in the controller design phase.

3.5 Summary

In Section 3.1, first, the general structure of a NARX model has been stated and the Polynomial NARX model has been described. In the identification phase, the initial NARX model has been obtained by refining the selected linear ARX model. Thus, the orders of the selected linear ARX model has become the orders of the NARX models as well. Then, the performance of the initialized NARX model has been improved by adding some nonlinear regressors. Afterwards, the nonlinear function of the obtained NARX model has been omitted in order to reduce the complexity. Owing to the fact that the validation, the 1st test, and the 2nd test datasets has been used during the regressor selection procedure, the candidate Polynomial NARX model has been tested on 14 different test datasets which had not been used before. In this way, the risk of overfitting has been elimininated. At the end of this section, the parameters and the nonlinear regressors vectors of the chosen Polynomial NARX model has been written.

At the beginning of Section 3.2, the structure of the Hammerstein-Wiener model has been expressed. Then, several Wiener models with different orders have been estimated in the identification phase. At this stage, the normalized Akaike's Information Criterion (nAIC) has also been considered. Thereafter, the estimated models have been tested by analyzing the performance of corresponding models on the validation, the 1st test, and the 2nd test data sets. In the validation phase, some Hammerstein and Hammerstein-Wiener models have been taken into account as well. After that, the performance of the candidate Wiener model has been enhanced by changing the existing lower and upper saturation limits on the outputs. In order to avoid overfitting, the unmodified and the calibrated models have been tested on 14 different test datasets which had not been utilized before. At the end, the calibrated model has been selected and its structure has been reported.

In Section 3.3, first, the structure of a Neural NARX model has been specified. In the identification phase, it has been mentioned that the estimation data had been randomly partitioned into 3 parts by means of the MATLAB Deep Learning toolbox. By doing so, the "early stopping" technique could have been implemented in the training phase. In this way, the generalization of the trained NNARX models has been improved. Afterwards, the regression analysis has been indicated and an example has been provided. In the validation phase, for each n_a and n_b order pairs, 20 Neural NARX models with same number of neurons (10) but different weights and biases have been trained and the mean of their fit percent on the validation, the 1st test, and the 2nd test datasets has been computed so as to get more robust results. Afterwards, some other NNARX models have been trained by employing 5 and 15 neurons in the hidden layer. After the performance analysis with regard to the validation, the 1st test, and the 2nd test datasets, 3 different model structures have been determined. 6 different Neural NARX models with the considered model orders and the number of neurons have been trained. The obtained NNARX models have been tested on 14 different test datasets which had not been employed before. After analyzing the results, one of them has been selected. At the end, the weights and the biases of the chosen NNARX model have been reported.

In Section 3.4, the selected Polynomial NARX, Wiener, and Neural NARX models have been compared based on the NRMSE quality metric. It has been observed that the chosen NNARX model outperformed on all of the test datasets. Finally, it has been remarked that the corresponding NNARX model might be used for designing a controller.

Conclusions

The purpose of this thesis was to analyze the performance of the identified linear and nonlinear models of the Quadruple-Tank Process (QTP). Even though the corresponding system possesses nonlinear dynamics, first, some linear models with different methods have been identified. In Chapter 2, State-Space, ARMAX, Output-Error, and linear ARX models have been employed so as to identify the plant. After evaluating the performance of the estimated models based on NRMSE criterion and residual analysis, one model has been selected for each linear method. The selected linear models have been compared in Table 2.11. In a nutshell, it has been observed that, not surprisingly, any of the selected models could not reach 80% fit on the average in the validation phase. In Chapter 3, some nonlinear models with different methods have also been identified in order to attain better performance. These are Polynomial NARX, Hammerstein-Wiener, and Neural NARX (NNARX) models. For each section of this chapter, a nonlinear model with sufficient performance has been chosen. Ultimately, the selected nonlinear models have been contrasted by using the different datasets which had not been included during the model selection procedure and it has been noticed that the selected NNARX model delivers the best performance.

During the identification of nonlinear models, extra test data sets have been used in order to eliminate the risk of overfitting and to improve the generalization of the estimated nonlinear models. Instead of bringing out new datasets, the Leave-One-Out Cross-Validation (LOOCV) technique might have been utilized.

A future development of this work could be based on designing a controller. In the controller design stage, one of the identified models, such as the selected Neural NARX (NNARX) model, might be used in the MATLAB/Simulink instead of the semi-physical model (see Figure 1.5). A Nonlinear Model Predictive Controller (NMPC) could be designed in order to control the plant owing to the fact that MPC is a powerful approach for controlling a system when a sufficiently reliable model is available and the chosen NARX models in Chapter 3 can be regarded as sufficient models. Furthermore, it is possible to explicitly include input, output, and state constraints through the MPC. These constraints might be due to limitations of the Quadruple-Tank System such as maximum allowable voltage of actuators (saturation) and design limits (height of a water tank).

Finally, a more exhaustive analysis of the properties of neural networks might be performed so as to better understand their reliability in the context of nonlinear system identification.

Bibliography

[1] F. S. Barbosa, “Quadruple tank control,” https://www.youtube.com/watch?v=bVl3kvpXG3o.

[2] R. Ranjbar, L. Etienne, E. Duviella, and J. Maestre, “Johansson’s quadruple-tank process diagram,” https://www.researchgate.net/figure/Johanssons-quadruple-tank-process-diagram-where-Sm-2-is-the-cross-section-of-all_fig1_342956995.

[3] L. Ljung, *System Identification Toolbox (User’s Guide)*, The Mathworks, Inc., Natick, Massachusetts, R2020b.

[4] S. M. Savaresi, “Lecture notes in model identification and data analysis course at politecnico di milano,” December 2019.

[5] D. Maurya, A. K. Tangirala, and S. Narasimhan, “Identification of output-error (oe) models using generalized spectral decomposition,” in *2019 Fifth Indian Control Conference (ICC)*, 2019, pp. 28–33.

[6] K. Narendra and K. Parthasarathy, “Identification and control of dynamical systems using neural networks,” *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 4–27, 1990.

[7] H. Liu and X. Song, “Nonlinear system identification based on narx network,” in *10th Asian Control Conference (ASCC)*, 2015, pp. 1–6.

[8] Í. Araújo, “Nonlinear system identification of a mimo quadruple tanks system using narx model,” *Przegląd Elektrotechniczny*, vol. 1, pp. 68–74, 2019.

[9] W. Zhao and E. Weyer, “A general convergence result for kernel-based non-parametric identification of nonlinear stochastic systems,” *IFAC-PapersOnLine*, vol. 51, no. 15, pp. 634–639, 2018, 18th IFAC Symposium on System Identification SYSID 2018.

[10] M. Bonin, V. Seghezza, and L. Piroddi, “Narx model selection based on simulation error minimisation and lasso,” *Control Theory and Applications, IET*, vol. 4, pp. 1157–1168, 08 2010.

[11] L. Piroddi and W. Spinelli, “An identification algorithm for polynomial narx models based on simulation error minimization,” *International Journal of Control*, vol. 76, pp. 1767–1781, 11 2003.

[12] M. Farina and L. Piroddi, "Simulation error minimization identification based on multi-stage prediction," *International Journal of Adaptive Control and Signal Processing*, vol. 25, pp. 389–406, 05 2011.

[13] E. Terzi, L. Fagiano, M. Farina, and R. Scattolini, "Identification of the cooling system of a large business center," *IFAC-PapersOnLine*, vol. 51, pp. 174–179, 01 2018.

[14] Y. Ma, H. Liu, Y. Zhu, F. Wang, and Z. Luo, "The narx model-based system identification on nonlinear, rotor-bearing systems," *Applied Sciences*, vol. 7, p. 911, 09 2017.

[15] K. Johansson, *Relay Feedback and Multivariable Control*, 1997.

[16] B. Bekhiti, A. Dahimene, K. Hariche, and M. A. Moustafa Hassan, "Mimo identification and digital compensator design for quadruple tank process," in *2017 5th International Conference on Electrical Engineering - Boumerdes (ICEE-B)*, 2017, pp. 1–7.

[17] N.-S. Nguyen, K. Nguyen, and T. Minh Chinh, "Black-box modeling of nonlinear system using evolutionary neural narx model," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, p. 1861, 06 2019.

[18] Z. Rayouf, C. Ghorbel, and N. Braiek, *Nonlinear PID Controller of MIMO Hammerstein model*, 07 2018.

[19] C. Ghorbel, "A new identification approach of mimo hammerstein model with separate nonlinearities," *Advances in Science Technology and Engineering Systems*, pp. 56–62, 11 2017.

[20] M. Faouzi, "Practical identification of four tank system described by volterra model," in *SENDA*, 10 2008.

[21] K. Johansson, "The quadruple-tank process: a multivariable laboratory process with an adjustable zero," *IEEE Transactions on Control Systems Technology*, vol. 8, no. 3, pp. 456–465, 2000.

[22] S. Özkan, T. Kara, and M. Arıcı, "Modelling, simulation and control of quadruple tank process," *10th International Conference on Electrical and Electronics Engineering (ELECO)*, pp. 866–870, 2017.

[23] R. Scattolini and L. Magni, *Advanced and Multivariable Control.* Pitagora, 2014, ch. 4 and 5, pp. 59–73.

[24] L. Ljung, *System Identification: Theory for the User.* Prentice Hall PTR, 1999.

[25] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control, 5th Edition.* John Wiley and Sons, 2015.

[26] C. Giller and M. Müller, "Linearity and non-linearity in cerebral hemodynamics," *Medical engineering and physics*, vol. 25, pp. 633–46, 11 2003.

[27] H. Galiana, "How can i tune the parameters (na,nb,nk...) of an arx/armax to identify a miso system, if the delays of input-output are not known?" December 2014.

[28] Q. Zhang, "Nonlinear system identification with output error model through stabilized simulation," *IFAC Proceedings Volumes*, vol. 37, no. 13, pp. 501–506, 2004.

[29] *Optimization Toolbox (User's Guide)*, The Mathworks, Inc., Natick, Massachusetts, R2021a.

[30] J. Nocedal and S. J. Wright, *Numerical Optimization.* Springer New York, 2006, pp. 529–562.

[31] L. Grüne and J. Pannek, *Nonlinear Model Predictive Control: Theory and Algorithms.* Springer New York, 2011, pp. 297–309.

[32] L. M. Fagiano, "Lecture notes in constrained numerical optimization for estimation and control course at politecnico di milano."

[33] A. Wills, T. Schön, L. Ljung, and B. Ninness, "Identification of hammerstein-wiener models," *Automatica*, vol. 49, pp. 70–81, 2013.

[34] F. Bonassi, M. Farina, and R. Scattolini, *Stability of discrete-time feed-forward neural networks in NARX configuration*, 12 2020.

[35] M. S. Hossain Lipu, M. A. Hannan, A. Hussain, M. H. Md Saad, A. Ayob, and F. Blaabjerg, "State of charge estimation for lithium-ion battery using recurrent narx neural network model based lighting search algorithm," *IEEE Access*, vol. 6, pp. 28 150–28 161, 05 2018.

[36] Q. Liu, W. Chen, H. Hu, Q. Zhu, and Z. Xie, "An optimal narx neural network identification model for a magnetorheological damper with force-distortion behavior," *Frontiers in Materials*, vol. 7, 2020.

[37] M. Hagan, H. Demuth, M. Beale, and O. De Jesús, *Neural Network Design.* Martin Hagan, 2014, ch. 12.

[38] B. Wilamowski and J. Irwin, *Intelligent Systems*, ser. Electrical engineering handbook series. CRC Press, 2018, ch. 12.

[39] M. H. Beale, M. T. Hagan, and H. B. Demuth, *Deep Learning Toolbox (User's Guide)*, The Mathworks, Inc., Natick, Massachusetts, R2021a.

Appendices

Appendix A: Implemented Code for the Quadruple-Tank System

The following MATLAB code contains the parameters of the Quadruple-Tank Process. For descriptions of these parameters, see Table 1.1.

```
h1max = 1.36;           % m
h2max = 1.36;           % m
h3max = 1.30;           % m
h4max = 1.30;           % m
hmin = 0.20;            % m (same for all tanks)

qamax = 3.26/3600;      % m^3/s
qbmax = 4/3600;         % m^3/s
qmin = 0/3600;          % m^3/s

a1 = 1.31e-4;           % m^2
a2 = 1.51e-4;           % m^2
a3 = 9.27e-5;           % m^2
a4 = 8.82e-5;           % m^2

S = 0.06;               % m^2

ga = 0.30;              % always between 0 and 1
gb = 0.40;              % always between 0 and 1

h1o = 0.65;             % m
h2o = 0.66;             % m
h3o = 0.65;             % m
h4o = 0.66;             % m

qao = 1.63/3600;        % m^3/s
qbo = 2/3600;           % m^3/s

g = 9.81;               % m/s^2

Ts = 25;                % s (Sampling Time)

T1 = S*sqrt(2*h1o/g)/a1;% 1st Time Constant
T2 = S*sqrt(2*h2o/g)/a2;% 2nd Time Constant
T3 = S*sqrt(2*h3o/g)/a3;% 3rd Time Constant
T4 = S*sqrt(2*h4o/g)/a4;% 4th Time Constant
```

The following code should be written into the interpreted MATLAB function shown in Figure 1.5 in order for the plant simulator to perform simulation experiments.

```matlab
function dh = fourtankfun(u,a1,a2,a3,a4,S,ga,gb,g)

h1 = u(1);
h2 = u(2);
h3 = u(3);
h4 = u(4);
qa = u(5);
qb = u(6);

dh1 = (-a1/S)*((2*g*h1)^(1/2)) + (a3/S)*((2*g*h3)^(1/2)) + (ga/S)*qa;
dh2 = (-a2/S)*((2*g*h2)^(1/2)) + (a4/S)*((2*g*h4)^(1/2)) + (gb/S)*qb;
dh3 = (-a3/S)*((2*g*h3)^(1/2)) + ((1-gb)/S)*qb;
dh4 = (-a4/S)*((2*g*h4)^(1/2)) + ((1-ga)/S)*qa;

dh = [dh1; dh2; dh3; dh4];
end
```

Appendix B: Code for Generating Asymmetric PRBS Input Signals

In this work, it has been decided to generate PRBS signals with also varying amplitude so as to emphasize on the output the nonlinear behavior of the system. To this end, the following MATLAB code can be utilized for producing index vectors.

```
idx = [1]; % Initial element of the index vector

for ii = 1:25
    A = randi(6,1);
    idx = [idx; idx(end) + A];
end

idx(1) = [];
idx(26) = 101;
idx_qa = idx % Index vector for the first input q_a


idx = [1]; % Initial element of the index vector

for ii = 1:25
    A = randi(6,1);
    idx = [idx; idx(end) + A];
end

idx(1) = [];
idx(26) = 101;
idx_qb = idx % Index vector for the second input q_b
```

By running the code above, index vectors with 26 elements are obtained. Then, the first element of *idx* vectors is deleted to ensure that the inputs q_a and q_b at the beginning have the nominal values q_a° and q_b° respectively. The number 101 is also appended to *idx* vectors in order to avoid the amplitude variation at the end. The reason for this is that the number of samples N are selected as 100 for both input signals (See the following code).

After obtaining index vectors, the asymmetric PRBS input signals can be created by employing the following code.

```
% PRBS_ex = idinput(N, Type, Band, Range) where N is the number ...
    of samples; Type corresponds to type of generated signal; ...
    Band represents the frequency range of generated signal; ...
    Range refers to generated input signal range.
u_qa = idinput(100,'prbs',[0 1],[0.00045278 0.0008]);

% In order to diversify the amplitude of the generated PRBS ...
    signals, the following for loop can be employed.
for ii = 1:length(idx_qa) - 1
     amp = rand;
     u_qa(idx_qa(ii):idx_qa(ii+1)-1) = amp*u_qa(idx_qa(ii));
end

u_qa = iddata([],u_qa,200);
figure; plot(u_qa)


u_qb = idinput(100,'prbs',[0 1],[0.00055556 0.0010]);

for ii = 1:length(idx_qb) - 1
     amp = rand;
     u_qb(idx_qb(ii):idx_qb(ii+1)-1) = amp*u_qb(idx_qb(ii));
end

u_qb = iddata([],u_qb,200);
figure; plot(u_qb)
```

In the previous code, 0.00045278 and 0.00055556 are the nominal values of the inputs q_a and q_b respectively. And, the 0.0008 and 0.0010 have been determined as the maximum values of the corresponding inputs. These values are close to q_{amax} and q_{bmax} but slightly smaller than them. In this code, PRBS signals with also varying amplitude are generated by means of the for loops. Lastly, the generated signals are converted to suitable forms which can be used for system identification via *iddata* command.

Appendix C: Implemented Code for Input-Output Polynomial Models (ARMAX, Output-Error, and ARX)

The following code can be used for estimating some ARMAX models with different orders.

```
load('td_vd.mat') % Load the detrended input-output data
opt = armaxOptions('Focus','simulation')

na = 1:7; % na: Ny-by-Ny matrix of nonnegative integers
nb = 1:7; % nb: Ny-by-Nu matrix of nonnegative integers
nc = 1:7; % nc: Ny column vector of nonnegative integers
nk = 1; % nk: Ny-by-Nu matrix of nonnegative integers
models = cell(1,343); % 7*7*7 = 343 models will be estimated
ct = 1;

% Nested for loops are used to estimate ARMAX models with ...
    different orders
for i = 1:7
    na_ = na(i);
    for ii = 1:7
        nb_ = nb(ii);
        for j = 1:7
            nc_ = nc(j);
            for k = 1
                nk_ = nk(k);
                models{ct} = armax(td,[na_*eye(2,2) nb_*ones(2,2) ...
                             nc_*ones(2,1) nk_*ones(2,2)],opt);
                ct = ct+1;
            end
        end
    end
end

models = stack(1, models{:}); % The generated models can be ...
    compared in a single figure after stacking them.
figure; compare(vd, models)

% The normalized Akaike Information Criterion (nAIC) of the ...
    estimated OE models are gathered by running the following code.
cc = cell(1,100);
zt = 1;
for zz = 1:100
    cc{zt} = aic(models(:,:,zz),'nAIC');
    zt = zt+1;
end
```

where *td* and *vd* are the detrended estimation and validation data sets.

The code below can be employed in order to generate Output-Error models with different orders. In this code, the estimations are made in the simulation mode as usual. Additionally, the fmincon search method is selected and the stability of the constituted OE models are guaranteed.

```matlab
load('td_vd.mat') % Load the detrended input-output data
opt = oeOptions('Focus','simulation','EnforceStability',true,...
      'SearchMethod','fmincon')
opt.SearchOptions.MaxIterations = 50;

nb = 1:10; % nb: Ny-by-Nu matrix of nonnegative integers
nf = 1:10; % nf: Ny-by-Nu matrix of nonnegative integers
nk = 1; % nk: Ny-by-Nu matrix of nonnegative integers
models = cell(1,100);
ct = 1;

for i = 1:10
    nb_ = nb(i);
    for j = 1:10
        nf_ = nf(j);
        for k = 1
            nk_ = nk(k);
            models{ct} = oe(td,[nb_*ones(2,2) nf_*ones(2,2)       ...
                             nk_*ones(2,2)],opt);
            ct = ct+1;
        end
    end
end

models = stack(1, models{:});
figure; compare(vd, models)
```

The following code can be utilized for estimating linear ARX models with different model orders.

```matlab
load('td_vd.mat') % Load the detrended input-output data
opt = arxOptions('Focus','simulation')
na = 1:10; nb = 1:10; nk = 1;
models = cell(1,100); ct = 1;

for i = 1:10
    na_ = na(i);
    for j = 1:10
        nb_ = nb(j);
        for k = 1
            nk_ = nk(k);
            models{ct} = arx(td,[na_*ones(2,2) nb_*ones(2,2)      ...
                            nk_*ones(2,2)],opt);
            ct = ct+1;
        end
    end
end

models = stack(1, models{:});
figure; compare(vd, models)
```

Appendix D: Code for Refining the Selected ARX Model to Obtain Initial NARX Model

The code below could be implemented so as to refine the existing linear ARX model and to acquire the initial NARX model.

```matlab
load('arx221.mat') % Load the selected linear ARX model

load('train_valid_test23.mat') % Load the estimation, validation, ...
    and test datasets including both input and output values

opt = nlarxOptions('Focus','simulation','SearchMethod','fmincon')

Initial_NARX = nlarx(Train, arx221, 'wavenet', opt)

compare(Validation, Initial_NARX) % Fit percent of the ...
    initialized NARX model on the validation dataset

resid(Validation, Initial_NARX) % Residual analysis
```

where $Train$ corresponds to the estimation dataset; $Validation$ is the validation dataset. After getting the initialized NARX model by refining the chosen ARX model, some nonlinear regressors are added by means of the MATLAB System Identification toolbox.

Appendix E: Code for Estimating Wiener Models and Testing Other Hammerstein-Wiener Models for the Candidate Orders n_a & n_b

The following code can be employed to estimate Wiener models with different orders and saturation nonlinearity functions on both outputs.

```
load('train_valid_test23.mat') % Load the estimation, validation, ...
    and test datasets including both input and output values
opts = nlhwOptions('InitialCondition','estimate') % Initial ...
    values are estimated by the software.
opts.SearchOptions.MaxIterations = 50
nb = 1:10; nf = 1:10; nk = 1;
models = cell(1,100); ct = 1;
for i = 1:10
    nb_ = nb(i);
    for j = 1:10
        nf_ = nf(j);
        for k = 1
            nk_ = nk(k);
            models{ct} = nlhw(Train, [nb_*ones(2,2) nf_*ones(2,2) ...
                  nk_*ones(2,2)], 'unitgain', 'saturation', opts);
            ct = ct+1;
        end
    end
end
```

In this thesis, after using the code above, the candidate orders have been determined as $n_b = 2$, $n_f = 2$, and $n_k = 1$. In the following code, these candidate orders are tested for some Hammerstein, Wiener, and Hammerstein-Wiener models.

```
% Since some words must now be altered instead of numbers, the ...
    following assignments are applied to run the code properly.
ncell_u = unitgain; ncell_s = saturation; ncell_d = deadzone;
ncell_i = [ncell_u,ncell_s,ncell_d];
ncell_j = ncell_i; ncell_k = ncell_i; ncell_m = ncell_i;

models = cell(1,81); ct = 1;
for i = 1:3
    ni_ = ncell_i(i);
    for j = 1:3
        nj_ = ncell_j(j);
        for k = 1:3
            nk_ = ncell_k(k);
            for m = 1:3
                nm_ = ncell_m(m);
                models{ct} = nlhw(Train, [2*ones(2,2) 2*ones(2,2) ...
                      1*ones(2,2)], [ni_,nj_], [nk_,nm_], opts);
                ct = ct+1;
            end
        end
    end
end
```

Appendix F: Code for Training and Simulating Neural NARX (NNARX) Models

```
close all;
load('IO_tra_val_test.mat') % Load the estimation, validation, ...
    and test datasets
% u_train - collected input time series.
% y_train - collected feedback time series.
above85 = 0; % Counter
for i = 1:20 % Run the code 20 times for each order pairs (na, nb)
    X = tonndata(u_train,false,false); % Convert input data to ...
    standard neural network cell array form
    T = tonndata(y_train,false,false); % Convert output data to ...
    standard neural network cell array form
    trainFcn = 'trainlm'; % Levenberg-Marquardt backpropagation ...
    algorithm.
    inputDelays = 1:10; % it is equivalent to nb = 10
    feedbackDelays = 1:1; % it refers to na = 1
    neurons = 5; % it corresponds to the number of neurons in the ...
    hidden layer
    net = narxnet(inputDelays, feedbackDelays, neurons, 'open', ...
    trainFcn); % Initialize NNARX model
    net.inputs{1}.processFcns = ...
    {'removeconstantrows','mapminmax'}; % Pre-processing functions
    net.inputs{2}.processFcns = {'removeconstantrows','mapminmax'};

    net = closeloop(net); % This is required to train the network ...
    in the simulation mode.
    [x,xi,ai,t] = preparets(net,X,{},T); % Prepare input and ...
    output time series data for training where x:shifted inputs, ...
    xi:initial input delay states, ai:initial layer delay states, ...
    t: shifted targets (shifted measured outputs)
    net.divideFcn = 'dividerand'; % As it has been reported in ...
    the thesis, a data division function has been used to divide ...
    the estimation dataset randomly and this is needed for ...
    implementing the "early stopping" technique.
    net.divideMode = 'time';  % Division mode for dynamic networks
    net.divideParam.trainRatio = 70/100;
    net.divideParam.valRatio = 15/100;
    net.divideParam.testRatio = 15/100;
    net.performFcn = 'mse';

    [net,tr] = train(net,x,t,xi,ai); % Train the prepared NARX ...
    Neural Network where tr is the training record
    y = net(x,xi,ai); % Simulate the trained network on the ...
    estimation dataset
    e = gsubtract(t,y); % Calculate the difference between the ...
    measured and the simulated outputs

    % Calculate the performance of training, validation and test ...
    data points
    trainTargets = gmultiply(t,tr.trainMask);
    valTargets = gmultiply(t,tr.valMask);
    testTargets = gmultiply(t,tr.testMask);
    trainPerformance = perform(net,trainTargets,y);
    valPerformance = perform(net,valTargets,y);
    testPerformance = perform(net,testTargets,y);

    % view(net) % View the NNARX model structure

    % Plots of the trained NNARX related to estimation dataset
    figure, plotperform(tr) % Performance plot of the randomly ...
    selected training, validation and test points based on MSE
    figure, plotregression(trainTargets, y, 'Train', valTargets, ...
    y, 'Validation', testTargets, y, 'Test', t, y, 'All') % ...
    Regression analysis of the estimation dataset
    figure, plotresponse(trainTargets, 'Train', valTargets, ...
    'Validation', testTargets, 'Test', y, 'outputIndex', 1)
    figure, plotresponse(trainTargets, 'Train', valTargets, ...
    'Validation', testTargets, 'Test', y, 'outputIndex', 2) % To ...
    see randomly selected points and the error of the second ...
    simulated output
    figure, ploterrcorr(e,'outputIndex',1) % Auto-correlations ...
    between the residuals of the first output
    figure, ploterrcorr(e,'outputIndex',2)
    figure, plotinerrcorr(x, e, 'inputIndex', 1, 'outputIndex',1) ...
    % Cross-correlations between the residuals of the first ...
    output and the first input
    figure, plotinerrcorr(x, e, 'inputIndex', 1, 'outputIndex',2)
    figure, plotinerrcorr(x, e, 'inputIndex', 2, 'outputIndex',1)
    figure, plotinerrcorr(x, e, 'inputIndex', 2, 'outputIndex',2)

    % Calculate the fit percent of the trained models on the ...
    estimation dataset with regard to NRMSE quality metric
    c_y = cell2mat(y);
    y_1 = c_y(1,:); % Simulated outputs
    y_2 = c_y(2,:);
    c_t = cell2mat(t);
    t_1 = c_t(1,:); % Measured (observed) outputs
    t_2 = c_t(2,:);
    nrmse_1 = norm((t_1-y_1)) / norm((t_1-mean(t_1))); % NRMSE ...
    for the 1st output
    nrmse_2 = norm((t_2-y_2)) / norm((t_2-mean(t_2)));
    fit1_train = (1-nrmse_1)*100; % Fit percent for the 1st output
    fit2_train = (1-nrmse_2)*100; % Fit percent for the 2nd output

    %% Validation dataset
    XV = tonndata(u_val,false,false); % Convert validation data ...
    to standard neural network cell array form
    TV = tonndata(y_val,false,false);

    netcv = closeloop(net);
    [xcv,xicv,aicv,tcv] = preparets(netcv,XV,{},TV); % Prepare ...
    input and output time series data for validation phase
    yv = netcv(xcv,xicv,aicv); % Simulate the trained network on ...
    the validation dataset
    c_yv = cell2mat(yv);
    ycv1 = c_yv(1,:);
    ycv2 = c_yv(2,:);
    c_tv = cell2mat(tcv);
    tcv1 = c_tv(1,:);
    tcv2 = c_tv(2,:);

    figure;plotresponse(tcv,yv,'outputIndex',1) % Performance of ...
    the first simulated output on the validation dataset
    figure;plotresponse(tcv,yv,'outputIndex',2)

    nrmse_val1 = norm((tcv1-ycv1)) / norm((tcv1-mean(tcv1))); % ...
    NRMSE for the 1st output
    nrmse_val2 = norm((tcv2-ycv2)) / norm((tcv2-mean(tcv2)));
    fit1_valid = (1-nrmse_val1)*100 % Fit percent for the 1st output
    fit2_valid = (1-nrmse_val2)*100

    %% The 1st test dataset
    XT2 = tonndata(u_test2,false,false);
    TT2 = tonndata(y_test2,false,false);

    netct2 = closeloop(net);
    [xct2,xict2,aict2,tct2] = preparets(netct2,XT2,{},TT2);
    yt2 = netct2(xct2,xict2,aict2);
    c_yt2 = cell2mat(yt2);
    yct2_1 = c_yt2(1,:);
    yct2_2 = c_yt2(2,:);
    c_tt2 = cell2mat(tct2);
    tct2_1 = c_tt2(1,:);
    tct2_2 = c_tt2(2,:);

    figure;plotresponse(tct2,yt2,'outputIndex',1)
    figure;plotresponse(tct2,yt2,'outputIndex',2)

    nrmse_test2_1 = norm((tct2_1-yct2_1)) / ...
    norm((tct2_1-mean(tct2_1)));
    nrmse_test2_2 = norm((tct2_2-yct2_2)) / ...
    norm((tct2_2-mean(tct2_2)));
    fit1_test2 = (1-nrmse_test2_1)*100
    fit2_test2 = (1-nrmse_test2_2)*100

    %% The 2nd test dataset
    XT3 = tonndata(u_test3,false,false);
    TT3 = tonndata(y_test3,false,false);

    netct3 = closeloop(net);
    [xct3,xict3,aict3,tct3] = preparets(netct3,XT3,{},TT3);
    yt3 = netct3(xct3,xict3,aict3);
    c_yt3 = cell2mat(yt3);
    yct3_1 = c_yt3(1,:);
    yct3_2 = c_yt3(2,:);
    c_tt3 = cell2mat(tct3);
    tct3_1 = c_tt3(1,:);
    tct3_2 = c_tt3(2,:);

    figure;plotresponse(tct3,yt3,'outputIndex',1)
    figure;plotresponse(tct3,yt3,'outputIndex',2)

    nrmse_test3_1 = norm((tct3_1-yct3_1)) / ...
    norm((tct3_1-mean(tct3_1)));
    nrmse_test3_2 = norm((tct3_2-yct3_2)) / ...
    norm((tct3_2-mean(tct3_2)));
    fit1_test3 = (1-nrmse_test3_1)*100
    fit2_test3 = (1-nrmse_test3_2)*100

    %% Average Fit Percent
    if fit1_valid < 0
    % If the fit percent of the corresponding NNARX model for the ...
    first output on the validation dataset is less than zero, ...
    then assign it to zero.
        fit1_valid = 0;
    end

    if fit2_valid < 0
        fit2_valid = 0;
    end

    if fit1_test2 < 0
        fit1_test2 = 0;
    end

    if fit2_test2 < 0
        fit2_test2 = 0;
    end

    if fit1_test3 < 0
        fit1_test3 = 0;
    end

    if fit2_test3 < 0
        fit2_test3 = 0;
    end

    % Compute the average fit percent for the i-th trial with ...
    same orders but different weights and biases.
    Avg_fit(i) = (fit1_valid + fit2_valid + fit1_test2 +          ...
                  fit2_test2 + fit1_test3 + fit2_test3)/6;

    % Increase the counter for each trained NNARX model which ...
    fits the data more than 85 percent on the average.
    if Avg_fit(i) > 85
        above85 = above85 + 1;
    end
end

Avg_fit % To see average fit percent of each trained NNARX model ...
    with same orders but different weights and biases

mean20_avgfit = mean(Avg_fit) % Represents the mean of 20 ...
    different Neural NARX models with same orders

above85 % This number shows that how many of the trained NNARX ...
    models have been able to fit the data related to validation, ...
    the first test and the second test datasets more than 85 ...
    percent on the average.
```

In the previous code, the orders n_a and n_b are 1 and 10 respectively. When this code is run, it creates 20 different Neural NARX models with these orders and 5 basic neurons in the hidden layer. In this work, all possible combinations of n_a and n_b from 1 to 10 and various number of neurons in the hidden layer have been taken into consideration and the obtained results have been reported in Section 3.3.3. Due to the fact that the validation, the first test, and the second test datasets had taken part in the training procedure, brand-new test datasets have been operated in order to eliminate the risk of overfitting. The code related to the new test data sets is not indicated in the appendix because the structure is very similar to the previous one. Moreover, the following code can be used to see the weights and the biases of the selected Neural NARX model.

```
load('LM1-10-5-fit91-04.mat') % Load the selected NNARX model

bh = net.b{1} %Bias vector of the hidden layer

bo = net.b{2} %Bias vector of the output layer

Input_weights = net.IW{1,1} %[5x20]
%[number of neurons x (nb*number of inputs)]

Output_weights = net.LW{1,2} %[5x2]
%[number of neurons x (na*number of outputs)]

w_ho = net.LW{2,1} %Weights from the hidden layer to the output layer
%[2x5] [number of outputs x number of neurons]

param_tot = net.numWeightElements %Total number of parameters ...
    (weights + biases)

% In order to use the selected NNARX model in the Simulink ...
    experiments, it is needed to generate a NARX Neural Network ...
    block. Write the following command for this:
gensim(net, 25) % 25 is the sampling time.
```