UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA QUÍMICA

BRENO SALDANHA SOUSA

LEVEL CONTROL OF COUPLED TANKS SYSTEM BASED ON NEURAL NETWORK TECHNIQUES

CONTROLE DE NÍVEL DE TANQUES INTERATIVOS BASEADOS EM TÉCNICAS DE REDES NEURAIS ARTIFICIAIS

CAMPINAS
2019

BRENO SALDANHA SOUSA

LEVEL CONTROL OF COUPLED TANKS SYSTEM BASED ON NEURAL NETWORK TECHNIQUES

CONTROLE DE NÍVEL DE TANQUES INTERATIVOS BASEADOS EM TÉCNICAS DE REDES NEURAIS ARTIFICIAIS

Dissertation presented to the School of Chemical Engineering of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Chemical Engineering.

Dissertação apresentada à Faculdade de Engenharia Química da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Química

Advisor: Prof. ANA MARIA FRATTINI FILETI

ESTE EXEMPLAR CORRESPONDE À VERSÃO FINAL DA DISSERTAÇÃO DEFENDIDA PELO ALUNO BRENO SALDANHA SOUSA, E ORIENTADO PELA PROFESSORA ANA MARIA FRATTINI FILETI.

Campinas
2019

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Rose Meire da Silva - CRB 8/5974

So85L

Sousa, Breno Saldanha, 1994-
Level control of coupled tanks system based on neural network techniques / Breno Saldanha Sousa. – Campinas, SP : [s.n.], 2019.

Orientador: Ana Maria Frattini Fileti.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Química.

1. Controle de processos químicos. 2. Controle preditivo. 3. Redes neurais (Computação). 4. Sistemas MIMO. I. Fileti, Ana Maria Frattini, 1965-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Química. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Controle de nível de tanques interativos baseados em técnicas de redes neurais artificiais
Palavras-chave em inglês:
Chemical process control
Predictive control
Neural network (Computer)
MIMO system
Área de concentração: Engenharia Química
Titulação: Mestre em Engenharia Química
Banca examinadora:
Ana Maria Frattini Fileti [Orientador]
Aline Carvalho da Costa
Rossana Odette Mattos Folly
Data de defesa: 18-02-2019
Programa de Pós-Graduação: Engenharia Química

Folha de Aprovação da Dissertação de Mestrado defendida por Breno Saldanha Sousa e aprovada em 18 de fevereiro de 2019 pela banca examinadora constituída pelos seguintes doutores:

_____

Profa Dra. Ana Maria Frattini Fileti

FEQ / UNICAMP

_____

Profa Dra. Aline Carvalho da Costa

FEQ / UNICAMP

_____

Profa Dra. Rossana Odette Mattos Folly

UFRJ – Escola de Química

A ATA da Defesa com as respectivas assinaturas dos membros encontra-se no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria do Programa da Unidade.

DEDICATÓRIA

“The world has changed far more in the past 100 years than in any other century in history. The reason is not political or economic but technological — technologies that flowed directly from advances in basic science.”

Stephen Hawkings (1942-2018), The Universe in a Nutshell (2001).

ACKNOWLEDGMENT

To God, for guiding me in the difficult moments.

To my parents, for emotional and financial support.

To my sister Thais, for the friendship and the life advices.

To my advisor professor Ana Frattini, for all the support, advices and suggestions in my research.

To my laboratory colleagues: Ana, Samuel, Homero, Carlos e Victor for sharing knowledge that helped me to overcome technical difficulties and for the good moments we spent together.

To my other friends: Raul, Fernando, Arthur, Fernanda, Thiago, Elisa, Thalyta and Beatriz for all the fun we had that helped me to relax in my free time.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

ABSTRACT

The level control of interactive tanks adjusting flow rates is a multiple input multiple output (*MIMO*) system that poses many challenges in the control problem, such as nonlinearities, interactions between manipulated and process variables and dead times. Therefore, conventional techniques such as the Proportional Integral Derivative (*PID*) controller might not work properly in this process. Artificial neural network (*ANN*) is a parallel processing technique that can capture highly nonlinear relationships among input and output variables. Hence, some control techniques that use *ANN* have been proposed for processes in which traditional feedback techniques may not work properly. This work aimed to test the experimental feasibility of two control techniques based on artificial neural networks applied to level control in coupled tanks: the model predictive control based on neural modeling (*MPC-ANN*) and an inverse neural network control. In the first strategy, an artificial neural network model of the process and an optimization algorithm are used to derive a satisfactory error performance. The second one is a control technique based on predicting the manipulated variables straight from the measurements of the process variables. Moreover, this work aimed to compare the performance of the two techniques mentioned with the conventional *PID*. The experiments were carried out using interactive tanks set up in of the Laboratory of Control and Automation at the University of Campinas (UNICAMP). Both levels of coupled tanks were to be controlled by manipulating the power of the two pumps that regulates output flow rates. An intermediate manual valve connected the tanks, generating nonlinearities and interaction between the levels, which made the success of *PID* control more difficult. The experimental application of the three mentioned techniques was performed with algorithm developed in MATLAB® and using a *PLC* to acquire the plant data. The comparison between the two-control neural network control techniques showed that the inverse neural control was not capable to track the set-point satisfactorily since it left an offset while the *MPC-ANN* was capable to track the set-point faster than the *PID* and it left smaller overshoots than the *PID*. The *MPC-ANN* performed better than the *PID* due to the capacity of model predictive control algorithm to minimize the deviations between the desired and predicted outputs, and the ability of artificial neural networks to deal with nonlinearities and interactions between manipulated and controlled variables. Besides, *MPC-ANN* couples feedback and feedforward strategy so it compensates model plant mismatches with the disturbance model.

RESUMO

O controle de nível de tanques interativos a partir da vazão é um sistema *MIMO* (*multiple input multiple output*), que envolve uma série de desafios como não linearidades acentuadas, interação entre as variáveis do processo e tempos mortos e, por isso, nem sempre pode ser controlado por técnicas de controle convencionais como o *PID*. Rede neurais artificiais (*RNA*) são uma técnica de processamento paralelo capaz de capturar relações bastante não lineares entre várias variáveis de entradas e várias variáveis de saídas. Dessa forma, diversas técnicas de controle utilizando RNA tem sido propostas para processos em que o controle feedback tradicional possa não funcionar satisfatoriamente. O presente trabalho visava testar a viabilidade experimental de duas técnicas de controle baseadas em redes neurais aplicadas no controle de nível em tanques interativos: o controle preditivo baseado em redes neurais (*MPC-RNA*), que consiste em utilizar um modelo neural do processo e um algoritmo de otimização para obter uma performance satisfatória; e o controle neural inverso, que é uma técnica de controle baseada na predição da variável manipulada diretamente das variáveis controladas. Além disso, o trabalho também visava comparar a performance das duas técnicas mencionadas com a performance do controlador *PID* convencional. Os experimentos foram realizados no sistema de tanques interativos do Laboratório de controle e automação (*LCAP*) na Unicamp. Ambos os níveis dos tanques acoplados eram controlados a partir da manipulação das potências das duas bombas que regulavam as vazões. Uma válvula intermediária manual conectava os tanques e gerava não linearidades, bem como interação entre os níveis, o que dificultava o controle *PID*. A aplicação experimental das três técnicas mencionadas foi feita por meio de um programa desenvolvido em MATLAB® e um CLP foi utilizado para fazer a aquisição dos dados da planta. Uma comparação entre as duas técnicas de controle baseadas em redes neurais mostrou que o controle neural inverso não foi capaz de seguir o setpoint satisfatoriamente, já que a técnica deixou um offset. Enquanto isso, a técnica *MPC-RNA* foi capaz de seguir o setpoint mais rapidamente e com menores overshoots do que o *PID*. A performance melhor do *MPC-RNA* em relação ao *PID* pode ser atribuída a capacidade do algoritmo de controle preditivo de minimizar os desvios entre a saída desejada e predita, e a habilidade das redes neurais artificiais de lidar com não linearidades e interação entre variáveis manipuladas e controladas. Além disso, o controlador *MPC-RNA* acopla a estratégia feedback e feedforward, dessa forma, compensando desvios entre o valor real e o valor predito a partir do modelo distúrbio.

Figure Index

Table Index

List of abbreviations and acronyms

ANN	-	Artificial Neural Network
ARMAX	-	Moving Average Models
CARIMA	-	Controlled Autoregressive Moving Average
DMC	-	Dynamic Matrix Control
DMPC	-	Distributed Model Predictive Control
FOPDT	-	First Order Plus Dead Time
GPC	-	Generalized Predictive Control
IAE	-	Integral of Absolute Error
IANN	-	Inverse Neural Network
IMC	-	Internal Model Control
ISE	-	Integral of Squared Error
ITAE	-	Integral of Time-weighted Absolute Error
LCAP	-	Laboratory of Control and Automation of Processes
LMPC	-	Linear Model Predictive Control
MAC	-	Model Algorithm Control
MIMO	-	Multiple Input Multiple Output
MPC	-	Model Predictive Control
MPC-ANN	-	Model Predictive Control based on Artificial Neural Network
MSEtest	-	Mean Squared Error of the test set
MSEtraining	-	Mean Squared Error of the training set
NMPC	-	Nonlinear Model Predictive Control
PI	-	Proportional Integral
PID	-	Proportional Integral Derivative
PLC	-	Programmable Logic Controller
RGA	-	Relative Gain Analysis
SISO	-	Single Input Single Output
SSE	-	Sum of Squared Errors
SSW	-	Sum of Squared Weights and Bias

Simbology

φ	–	Activation function
w_{ij}	–	Sinaptic weight of the connection between neuron “i” and neuron “j”
b_j	–	Bias
γ	–	Number of effective parameters
x_i	–	Neuron input
y_j	–	Neuron output
x_n	–	Normalized variable
x_{min}	–	Minimum value of the variable
x_{max}	–	Maximum value of the variable
y_m	–	Predicted output
y_{sp}	–	Set point
y_{sp1}	–	Set point of level 1
y_{sp2}	–	Set point of level 2
y_r	–	Reference trajectory
y_p	–	Measured output
u'	–	Calculated future input value
U	–	Current input
J	–	Objective function
N_p	–	Prediction Horizon
N_c	–	Control Horizon
w	–	Weight of the control action in the objective function
w_y	–	Weight of the control error in objective function
d_k	–	Discrepancy between the measured value of the plant and the predicted value
y^c	–	Output corrected by the disturbance model
$G_c(s)$	–	Control transfer function
$G_m(s)$	–	Sensor transfer
$G_f(s)$	–	Final element of control transfer function
$G_p(s)$	–	Process transfer function
$G_{p11}(s)$	–	Process transfer function that relates y_1 with u_1
$G_{p12}(s)$	–	Process transfer function that relates y_1 with u_2

$G_{p21}(s)$	–	Process transfer function that relates y_2 with u_1
$G_{p22}(s)$	–	Process transfer function that relates y_2 with u_2
$G_{proc}(s)$	–	Product of G_p, G_f and G_m
$G_d(s)$	–	Disturbance transfer function
$e(s)$	–	Error in the *PID* block diagram
$c(s)$	–	Control signal in the *PID* block diagram
$m(s)$	–	Manipulated variable in the *PID* block diagram
$d(s)$	–	Disturbance in the *PID* block diagram
K_c	–	Proportional gain
τ_i	–	Integral time
τ_d	–	Derivative time
λ_{ij}	–	Relative Gain between the controlled variable y_i and the manipulated variable u_j
Λ	–	Relative Gain Array
y_i	–	Level of the tank "i"
P_i	–	Power of the pump "i"
K_p	–	Gain of the process
τ_p	–	Time constant
t_d	–	Dead time
K_{cr}	–	Ultimate Gain
T_{cr}	–	Ultimate period
ρ	–	Mass Density
A_j	–	Cross Sectional Area of the tank "j"
C_{vj}	–	Valve coefficient
L_c	–	Height of the tank
T	–	Time to complete the volume of the vessel

1. Introduction

In chemical industries, certain process variables such as temperature, pressure, concentration, and level must be kept at given values, which can be achieved by manipulating some variables like the position of a valve or the power of a pump. Although these variables can be controlled by a human operator, automatic control is generally desirable because humans often make more mistakes and cannot respond as fast as a machine to a disturbance in the process.

Process control is becoming more and more important for several reasons. First, the safety and environmental regulations are very stringent nowadays. Second, due to global competition, it is important to keep a high standard of quality by reducing the variability of the process. Third, it is possible to lower variable costs of an industry by maintaining some process variables at a constant level (SEBORG; EDGAR; MELICHAMP, 2003). Besides, it is important to satisfy some operational constraints of industrial equipment. For coupled tanks, the importance of level control is usually related to equipment safety and product quality (DERDIYOK; BAŞÇI, 2013).

There are several important control purposes that must be taken into consideration in a process. A good control system must have the ability to suppress disturbances so that the control strategy makes the right change in the manipulated variables to cancel the impact of the disturbances. Besides, the control strategy must track the setpoint and optimize the control performance, by minimizing the control errors (the difference between the setpoint and the measured controlled variable), the overshoot and the response time. Finally, a good control cannot give an unstable response under any circumstance.

The level control of interactive tanks is a nonlinear problem due to the valves, so it is unlikely to be satisfactorily controlled by a conventional feedback controller (NOEL; PANDIAN, 2014). Moreover, according to Derdiyok and Başçi (2013), the problem has dead time and interactions between the manipulated and controlled variables of different loops, which makes the control by conventional techniques difficult. Roy and Roy (2016) stated that although *PID* can work properly in 80% of the situations, there are some situations that more advanced techniques should be used. For example, in a wastewater treatment plant, when it is necessary to vary the level of one tank while keeping the level of the other tank constant, it is likely that the *PID* does not work well due to very stringent restrictions. Therefore, advanced techniques should be used in situations that conventional control techniques do not work properly.

As level control of coupled tanks is not an easy task, many works have proposed different advanced techniques. For example, Roy and Roy (2016) used the fractional order *PI* along with a feedforward controller to control the level of interactive tanks. Mercangöz and Doyle (2007) used the technique of distributed model predictive control (*DMPC*) for controlling the level in a system of four tanks connected and got a better result than the control using decentralized *MPC*. Saaed, Udin, and Katebi (2010) compared a multivariable predictive *PID* controller and a simple multi-loop *PI* and derived a better result for the first technique. Vadigepalli, Gatzke and Doyle (2001) compared the performance and robustness of a *PI* decentralized control with "inner-outer" factorization-based multivariable internal model control (*IMC*) and H∞ control. The multivariable *IMC* and *H∞* provided a better performance than the *PI* controller. Qamar (2012) showed the use of sliding mode control to control interconnected tanks. Moreover, Dharamniwas *et al.* (2012) used a fuzzy controller to control coupled tanks.

Model predictive control (*MPC*) is the most used advanced control technique in the industry. Indeed, Mayne (2014) stated that *MPC* had an explosive growth in both the academy and the process industry because it proved itself very successful in comparison with other methods of multivariable control. According to Qin and Badgwell (2003), its success in the industry is related to its conceptual simplicity and its ability to handle easily and effectively complex systems with hard constraints and several inputs and outputs.

The application of neural network in the field of control of processes is promising due to its ability to deal with complex and nonlinear relationships among many inputs and outputs. Besides, neural networks can deal with noisy data (TAYYEBI; ALISHIRI, 2014). Hence, they are more representative than conventional empirical models and phenomenological models (HIMMELBLAU, 2000). Therefore, this work will study the use of two neural networks control techniques in the interactive tank system: the model predictive control based on neural networks and the inverse neural network control.

The liquid level control is one of the most common problems found in industry with several applications found in the literature, such as water and effluent treatment (ALEX *et al.*, 1999), (CONCEPCION; MENESES; VILANOVA, 2011), boilers (GAIKWAD *et al.*, 2011) (ÅSTROÖM; BELL, 2000), in chemical and pharmaceutical processing, food processing (ROY; ROY, 2016), nuclear steam generation (GAIKWAD *et al.*, 2011), (TAN, 2011), (KOTHARE *et al.*, 2000).

The next section will discuss some industrial applications of tank level control in order to show its importance in practical situations and to show some difficulties related to its control.

Section 1.2 will briefly show the theory behind artificial neural networks, its structure, how it works and the training process. Section 1.3, 1.4, 1.5 and 1.6 will discuss the three controllers that will be used in this work: the model predictive control based on neural networks, the inverse neural network control, and the *PID* controller.

1.1. Industrial Applications

This section will discuss industrial applications of level control in which the control is challenging due to nonlinearities, process variables interactions and inverse response. The purpose of this section is to illustrate the importance of designing advanced control techniques for level control problems.

1.1.1. Nuclear Steam Generation

One of the most interesting applications of level control is in steam generators of nuclear power plants because problems in the control can cause a violation of safety limits and so reactor shutdown. Besides, according to Kothare *et al.* (2000), the water level control of the steam generator is difficult because it exhibits strong inverse response, nonlinear plant characteristics, unreliable sensor measurements and hard constraints that can lead to instability.

The pressurized water reactor system showed in Figure 1 is composed of two subsystems: the steam supply system and the power conversions system. The steam supply system is composed by the reactor vessel and the steam generator. Its function is to transform the power released in the nuclear fiction reaction in thermal energy. The power conversion system is composed by the condenser, the turbine, and the electric generator. Its function is to transform the heat energy into electrical power. The manipulated variable used in this control level problem is the feedwater flow rate. The main problem is to keep the level within the allowable limits, even when the steam demand increases as a result of an increase in electrical energy.

The inverse response is caused by the “swell and shrink” effects. When the feed water flow rate is increased, the cold feed water causes a collapse in the bubbles of the tube. Hence, the liquid level decreases at the first moment. However, after some time, the effect of increasing the water mass flow surpasses the effect of collapsing the bubbles which make the level to increase. The inverse response can be also caused by the effect of the steam flow rate in the level. If the steam flow rate is increased, the level initially rises, because the pressure of the tank decreases, which causes an expansion of the water in the tube. However, the total effect of the water mass flow causes an increase in the level after some time.

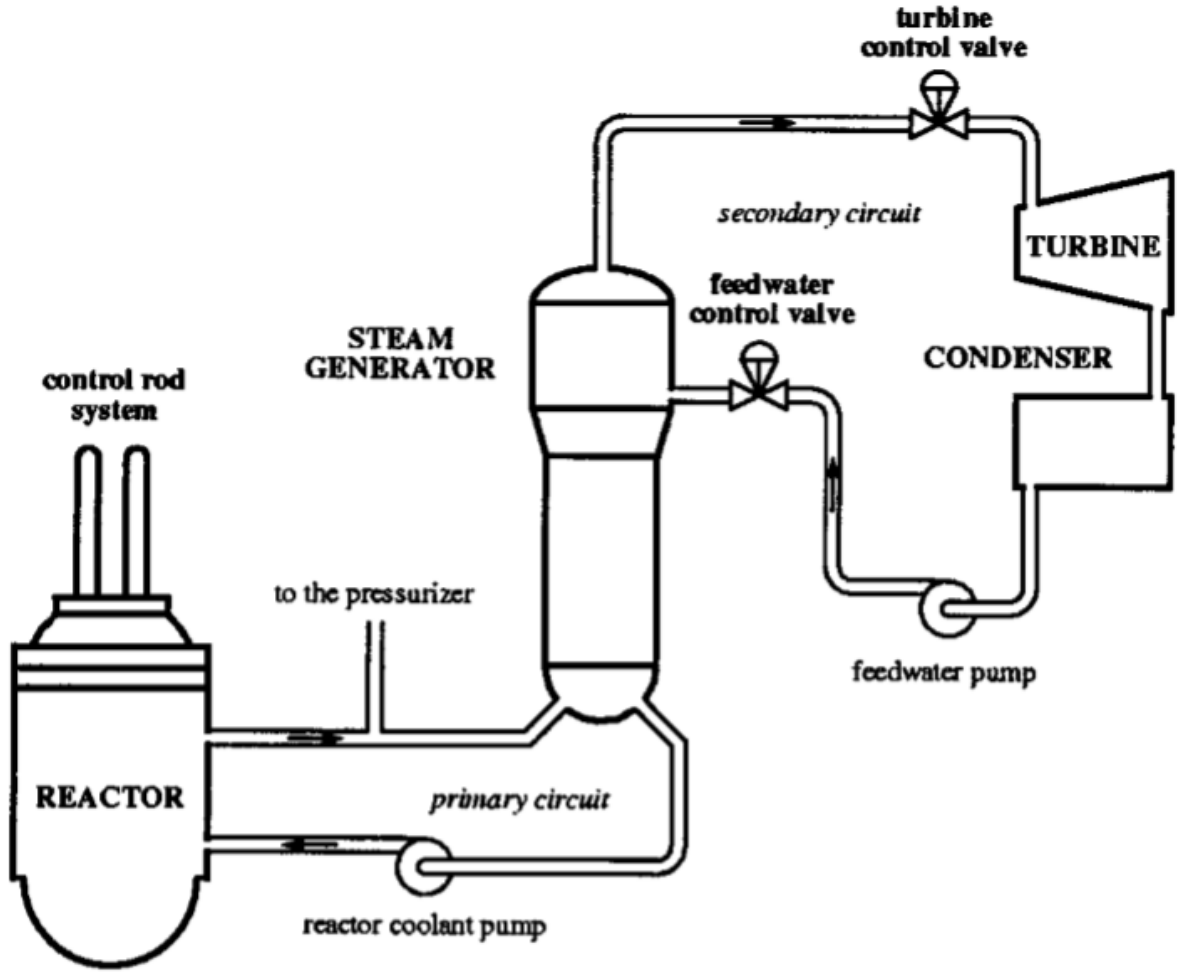

Figure 1: Layout of the pressurized water reactor system (KOTHARE et al., 2000).

1.1.2. Wastewater treatment process

Level control is extremely important in wastewater treatment process in order to minimize environmental impact on receiving water by removing pollutants to meet the strict standards imposed by authorities. Moreover, the wastewater treatment process is highly nonlinear, it is subject to large perturbations in flow, and it has uncertainties related to the composition in the incoming wastewater. These characteristics pose some difficulties to the control system, so ALEX *et al.* (1999) studied a methodology to assess control performance in a wastewater treatment plant depicted in Figure 2.

A suitable liquid level control is essential in slurry treatment to equalize the very different influent stream that can vary a lot in flow and composition. This equalization by the level control is important to achieve the highest performance in the conversion from organic matter to methane.

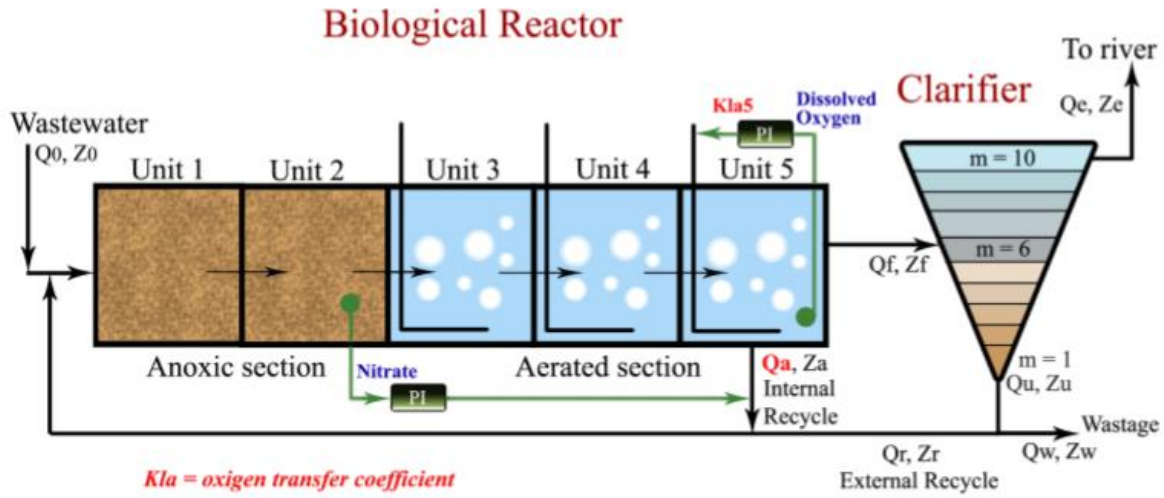

Figure 2: Wastewater treatment plant (CONCEPCION; MENESES; VILANOVA, 2011).

1.1.3. Dearation process

The dearation process is applied in power plants where it is important to remove oxygen and prevent corrosion in the boiler. The dearator is a device that is placed between the turbine and the boiler. It receives condensate from the turbine and provides feedwater to the boiler by heating to remove the oxygen. The dearator is a two-chamber pressure vessel that consists of two sections: the storage section and the deaerating section shown in Figure 3, as parts 43 and 41.

The level control is necessary for reducing the condensate flow to the dearator during the transient conditions and to avoid flooding of the column. Conventional controllers will have problem to control the system because when the pressure on the turbine falls, the level in the tank will also decrease and a simple controller will try to compensate the problem by increasing the condensate in-flow thereby putting further energy demands on the system. The increase in condensate in-flow will also increase the pressure drop which may cause the flooding of the column if the pressure drop across the equalizers (the equipment that links the two parts of the deaerator, 57 in Figure 3) exceeds the static head in the deaerating section.

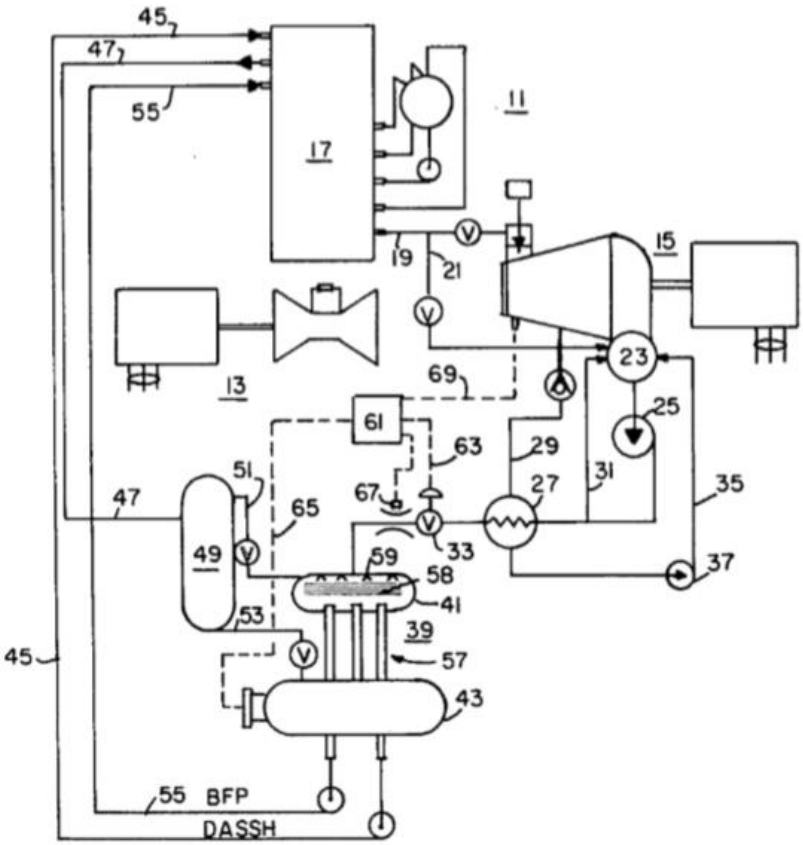

Figure 3: Diagram of the power plant (LABBE et al.).

2. Objective

2.1. Main

The main purpose of this work was to develop a neural network control based on two strategies: the inverse neural network control and the model predictive control based on neural network; and to test their performance in a coupled tanks system, which presents some difficulties in the control due to nonlinearities and interactions between the manipulated and controlled variables. Besides, the work also aimed to compare the two proposed strategies with conventional *PID* in order to understand the potential advantages and disadvantages that these strategies may have. All these strategies will be performed aiming to control two levels in a coupled tanks system.

2.2. Specific

The specific objectives of this work were:

- To develop Simulink models for multiloop *PID* control strategy and for *PID* with decouplers;

- To develop first principle models based on mass balances to simulate the process;

- To develop Matlab® algorithms for identification and control of an *MPC-ANN* and neural network control based on the inverse model for both simulation and the real process of coupled tanks system;

- To determine the values of the *MPC-ANN* tuning parameters (like the prediction horizon, control horizon, weight of the control weight and sampling time).

3. Theoretical Framework

This section will discuss the theory used in this work and it will be divided into five parts. Section 3.1 will give a brief review about artificial neural network advantages, the learning process, the neural network topology, the preprocessing and postprocessing steps. Section 3.2 will discuss about the Model predictive controller and the last three sections will discuss the three control techniques used in this work: Model predictive control based on neural networks, the inverse neural network control and the *PID* controller.

3.1. Artificial Neural Networks

Artificial Neural Networks (*ANN*) are mathematical models constituted by basic processing units called neurons, that are distributed in many layers connected by a complex network (FILETI; PACIANOTTO; CUNHA, 2006). As its own name implies, neural networks are an attempt to model the way human brain performs a particular task or function of interest. They are able to storage experimental knowledge through a learning process and interneuron connections, known as synaptic weights, used to store the acquired knowledge (HAYKIN, 2008).

However, it is important to highlight that engineering systems are considerably simpler than the human brain. Therefore, *ANN* can be seen as a nonlinear empirical model that processes the information in a parallel way and it is useful to represent input-output data, to recognize patterns, to classify data and to predict data in time series (HIMMELBLAU, 2000).

The problem solution through *ANN* are rather attractive due to its natural massive parallel distributed structure that makes neural networks capable to have a better performance than conventional models. Moreover, *ANN* are capable to generalize the learning since they can produce reasonable output prediction when inputs different from those from the training data set are presented.

According to Haykin (2008), *ANN* have several useful properties such as:

- Nonlinearity, which is an extremely important characteristic of neural networks because most of the processes are nonlinear;

- Adaptivity, since neural networks can change their parameters to adapt to changes in the surrounding environment. So, they can be retrained to deal with changes in the operating environmental conditions or they can be trained to change their parameters with time in the nonstationary environment. This adaptive structure might increase the robustness of the system;

- Evidential Response, that is, the *ANN* can not only provide information about a particular pattern to select, but also the confidence in the decision made;

- Fault Tolerance, which means that neural networks are able to perform robust computation and its performance does not worsen a lot under adverse operating conditions.

The processing structure of a neuron is shown in Figure 4. It is possible to realize that the input data (x_i) is weighted by the neural connections known as synaptic weights (w_{ji}). The result is added to a bias creating an activation state. A function called transfer function is applied to the activation state creating the output. The purpose of the transfer function is to generate a nonlinear relationship between the input and output data. The calculation described is shown in Equation 1.

$$y_j = \varphi\left(\sum_{i=1}^{n} w_{ji}x_i + b_j\right) \tag{1}$$

Theoretically, the activation function may be whatever differentiable function, such as log-sigmoid function, hyperbolic tangent function or even linear function (HIMMELBLAU, 2000). A list of the most important activation functions used in neural networks is shown in Table 1.

Table 1: Common activation functions.

Activation Function	Mathematical Description ($f(x)$=)	Output
Linear	x	[-inf; +inf]
Log-Sigmoid	$\frac{1}{1-e^{-x}}$	[0; 1]
Hyperbolic	$\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	[-1; +1]
Exponential	e^{-x}	[0; inf]
Sinoidal	$\sin(x)$	[-1; 1]
Tan-Sigmoid	$\frac{2}{1+e^{-2x}}-1$	[-1; +1]

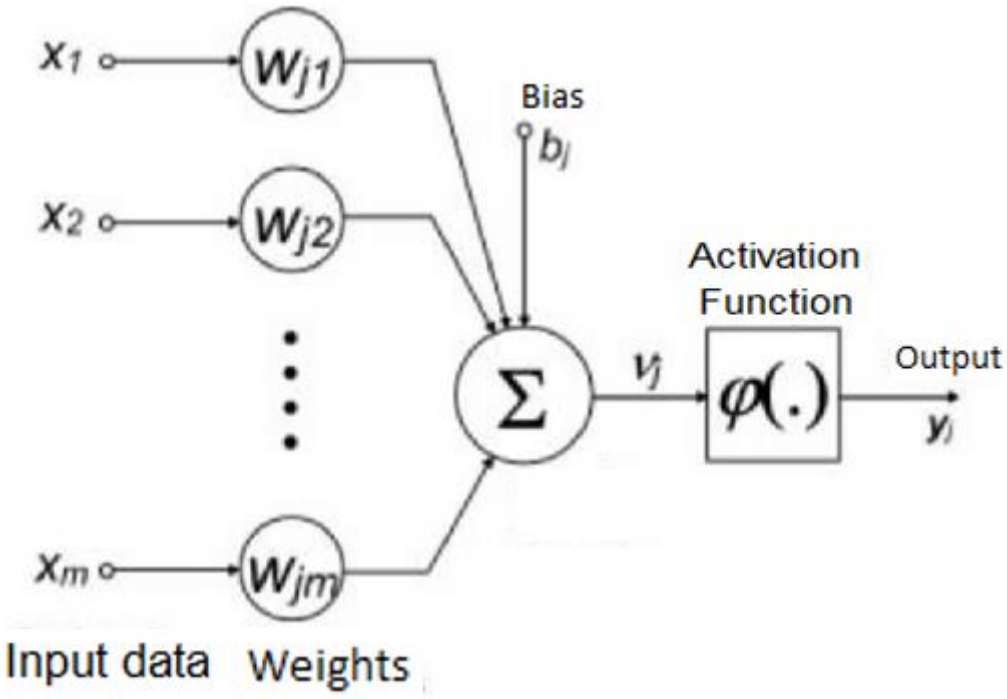

Figure 4: Structure of a single neuron, adapted (HIMMELBLAU, 2000).

It is possible to classify the neural network according to the topology as feedback or feedforward. In the feedback topology there are loops, so the output signal is not sent only from layers nearer to the input to layers nearer to output. In the feedforward structures there are no loops, so the output signal from a neuron is always sent to neurons that do not receive any information from the input. When all neurons from one layer send output signal only for the next layer, the neural network is called strictly feedforward. An illustration of a feedforward neural network is shown in Figure 5. This architecture is the most used in chemical engineering problems (FILETI; PACIANOTTO; CUNHA, 2006).

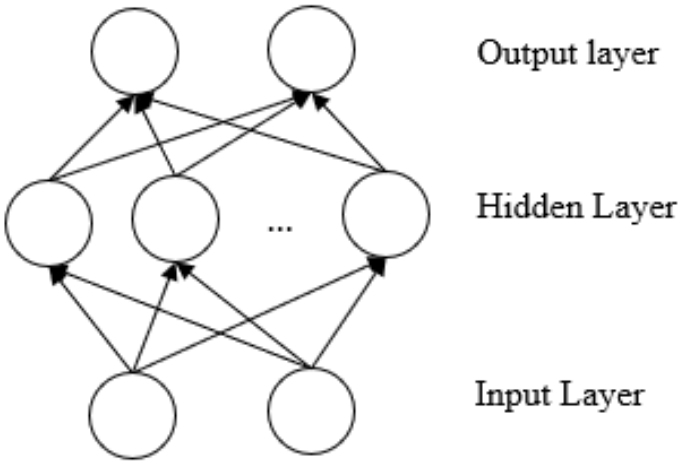

Figure 5: Feedforward neural network architecture, adapted (HIMMELBLAU, 2000).

Feedforward neural networks might have one or more hidden layers. Usually, a nonlinear function is used in the hidden layers and a linear function is used in the output layer. This allows that the neural network acts as a universal approximator, that is, it can predict from linear relationships to highly nonlinear relationships (DEMUTH; BEALE; HAGAN, 2010).

One of the most important phases of neural network is the learning or training phase. In this phase, a set of examples of input and target output data is presented to the network and the parameters of the neural network, the weights and biases are adjusted interactively to minimize the objective function. This is made by the information provided to the supervisor.

The learning may be supervised or non-supervised. In the supervised learning, there is a supervisor and the neural network learns how to imitate the supervisor from the input and output data. In the non-supervised learning, only inputs are provided to network and the supervisor acts providing labels to the groups.

In supervised learning, there are many kinds of learning. The most common is called Backpropagation Algorithm, in which weights and bias are moved to the negative gradient direction of the objective function, that is, the direction that the objective function is reduced faster. In the Backpropagation Algorithm, the new values of weights and bias are computed from the output layer to the input layer (HAGAN *et al.*, 2014).

One of the biggest problems in the neural network training is the *overfitting*, in which the training error goes to small values, but, when new values are presented to the network the error is large. This means that the neural network will not have a suitable predictive capacity, which happens when the *ANN* simply link the dots during the training phase. Overfitting is usually caused when the neural network has a great number of parameters.

To avoid overfitting, some strategies can be used. The first strategy is simply to collect more data so that the number of network parameters is much less than the number of data points in the training set. However, this methodology is not always feasible, since the amount of data is usually limited. Therefore, two methodologies are often used to avoid overshooting: the early stop and the Bayesian regularization.

In the early stop methodology, the data set is divided into three: the training set, the validation set, and the test set. The training set aims to adjust the weights and bias. The validation set aims to determine when the training will stop. At the beginning of the training, the error of both the validation and the training set decrease until a certain point when the error of validation starts to increase while the training error continues to decrease. At this moment, the network stops the training. The last set of data is the test set which aims to test the generalization ability of the neural network.

The other methodology to improve generalization is the regularization. It consists in modifying the objective function so that the sum of squared errors (SSE) is added to the term of the sum of the squared weights and bias (*SSW*). The resulting objective function is shown in the Equation 2 (HAYKIN, 2008).

$$F = \alpha SSE + \beta SSW \tag{2}$$

α and β are parameters of the objective function.

Changing this objective function decrease the number of weights and bias found in the training phase. Thus, the response will be smoother and less likely to overfit the data. One of the most important characteristics of this method is to provide the number of parameters that are being effectively used by the neural network (γ). During the training phase, it is common to increase the number of neurons of the hidden layer to improve the network. However, when a certain number of neurons is reached, a further increase in neurons will not cause a further increase in γ. According to Foresee and Hagan (1997), this number of neurons should be chosen to the hidden layer, so that the number of effective parameters will not change when the number of neurons is increased.

Some preprocessing and postprocessing steps can be applied in the input and output vector in order to make the neural network more efficient. One possible strategy is to normalize the data, so that all variables have values between -1 and 1, and neither of the variables has more importance than the others. Equation 3 shows the normalization.

$$x_n = \frac{2(x - x_{min})}{(x_{max} - x_{min})} - 1 \tag{3}$$

Where x_n is the normalized variable, x_{max} is the maximum value of the variable in the training set and x_{min} is the minimum value in the training set.

3.2. Model Predictive Control

Model predictive control (*MPC*) is one of the most used advanced control techniques in the industry and scientific community (SEBORG; EDGAR; MELICHAMP, 2003). This can be explained by the fact that *MPC* integrates techniques such as the optimal control, the stochastic control and control of processes with dead time. Moreover, *MPC* has many

characteristics of an ideal controller since it is able to handle processes with multivariable interactions, time delay, inverse response, nonlinearities, input/output constraints. Besides, *MPC* can compensate measurable and unmeasurable disturbances due to its feedback and feedforward structure and it is also able to optimize the use of control effort (OGUNNAIKE; RAY, 1994).

According to Ogunnaike and Ray (1994), not all processes have to be controlled by an *MPC* to present a good performance and a robust control. However, some processes that have complicated dynamics, with dead time, inverse response, nonlinearities and hard constraints in the input and output are more benefited by the technique.

In *MPC* applications output variables are also referred to as controlled variables and input variables are also called manipulated variables.

Model predictive control can be defined as a class of advanced control techniques that uses the current and past information of input and output variables to calculate the future output of the process through a model of the process. Then, the input variables that optimize the future performance of the system are calculated through an optimization algorithm. Figure 6 shows a block diagram of the process.

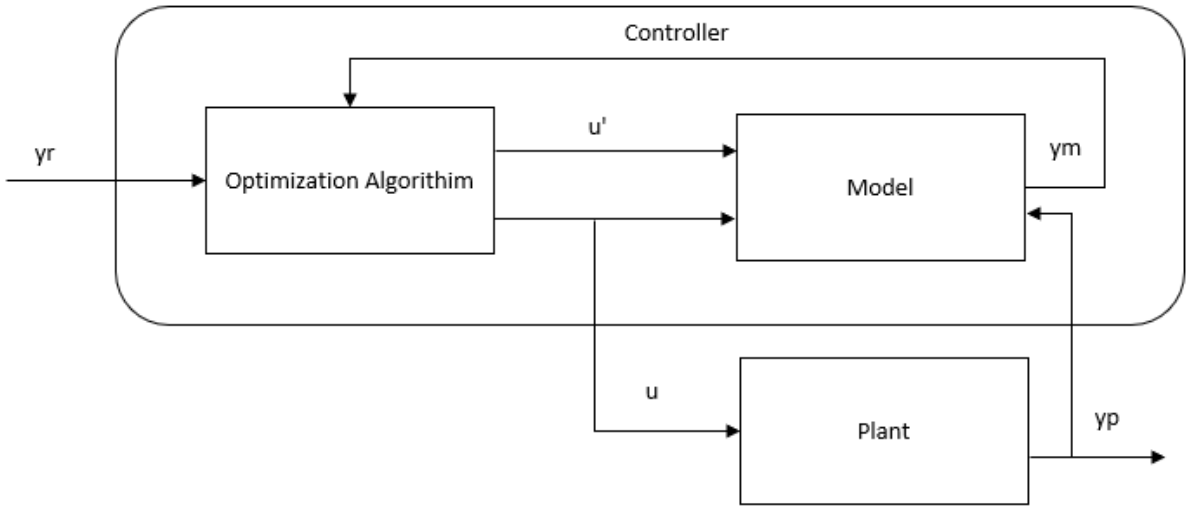

Figure 6: Block Diagram of Model Predictive Control, adapted (DEMUTH; BEALE; HAGAN, 2010).

The variable y_r represents the reference trajectory which is given by Equation 4, y_{sp} is the set point and y_p represent the measured output. The reference trajectory is the desired output trajectory, which can be either the set point (if $\alpha = 0$) or trajectory that is less abrupt than a step.

$$y_r = \alpha y_p + (1 - \alpha) y_{sp} \quad (4)$$

The variable *u'* is the calculated future input variables, y_m is the future output predicted by the model variables and *u* is the current input variable since only the first input variable is implemented.

The future controlled variables are calculated through a time interval known as the prediction horizon (N_p), while the input variables are predicted by the optimization algorithm for the control horizon (N_c). Between the control horizon and the prediction horizon, the input variables are constant and equal to the last predicted value. Besides, it is important to emphasize that, although N_c future values of manipulated variables are calculated at each sample time, only the first move is implemented, and the optimization algorithm runs every sample time. This principle is known as Receding Horizon and it is shown in Figure 7.

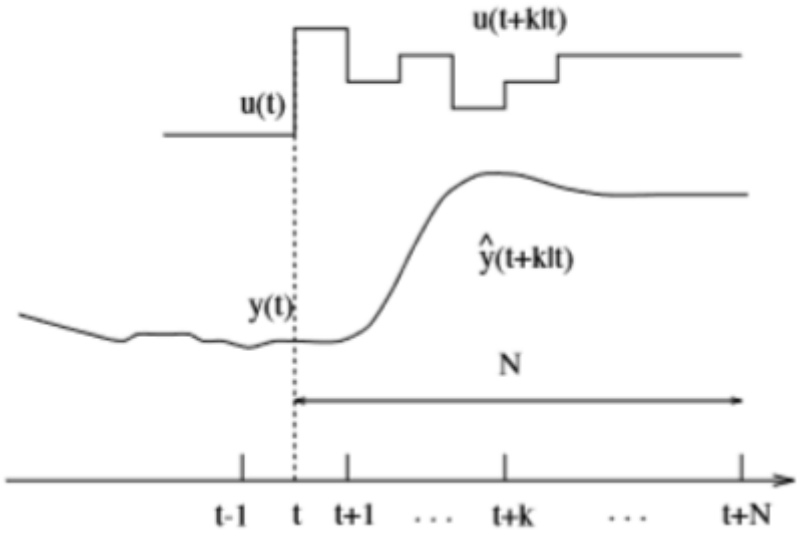

Figure 7: The receding horizon principle (CAMACHO, 2004).

The optimization algorithm commented above and shown in Figure 6 has a cost function that must be minimized which is shown in Equation 5. This cost function takes into consideration the future predicted deviation from the target trajectory over the prediction horizon; and the minimization of the expenditure of the control effort through the control horizon in order to avoid great changes in the manipulated variables that could decrease the lifetime of the actuator (OGGUNAIKE, 1994).

$$J = \sum_{j=1}^{Np} w_y\big(y_r(k) - y(k+j|k)\big)^2 + \sum_{j=1}^{Nc} w\big(u(k+j-1|k) - u(k+j-2)\big)^2 \quad (5)$$

The model of the process must be able to represent the dynamic behavior of the system accurately. It can be either linear or nonlinear depending on the complexity of the control problem and it can be updated through online identification methods in order to incorporate an adaptive scheme in the control.

As no model can represent perfectly the reality, *MPC* usually has an error prediction update technique to correct these model inaccuracies by comparing the measured output values of the plant with the values predicted by the model. A simple form of this correction is the disturbance model given by Equations 6 and 7.

$$d_y(k) = y_p(k) - y_m(k|k-1) \quad (6)$$

$$y^c(k+1) = y_m(k+1) + d_y(k) \quad (7)$$

MPC techniques have several advantages. As commented earlier, *MPC* is able to deal with complex process, with dead times, inverse responses, interactions between the controlled and manipulated variables and nonlinearities. However, *MPC* have also some disadvantages such as: it is more complex than the conventional *PID* so it needs more computational effort to run the algorithm; in the adaptive control case, all the computation must be carried out at every sampling time; it requires a prior knowledge of the process since an appropriate model of the process is necessary. It is important to emphasize that, although the *MPC* might work even with model discrepancies, an inappropriate model may affect the benefits of the technique.

There are several *MPC* techniques used in industries and they differ only in the kind of model used, and the cost function used. Some of the most known *MPC* techniques are Dynamic Matrix Control (*DMC*), Model Algorithm Control (*MAC*), state space *MPC* and Generalized Predictive Control (*GPC*).

DMC was developed by Cutler and Ramaker in 1979 and was the first *MPC* algorithm developed. The technique uses a nonparametric model based on the response to a step in the input. *MAC* is very similar to *DMC*, but its model is based on the response to an impulse in the

input. The state space *MPC* is based on the state space model while *GPC* is based on controlled autoregressive moving average (*CARIMA*) model.

The techniques cited before are all linear. Linear Model Predictive Control (*LMPC*) is well established to control multivariable processes. However, *LMPC* is inadequate for highly nonlinear processes, such as high purity distillation column, and moderately nonlinear processes which have large operation regimes, such as multi-grade polymer reactors. Therefore, Nonlinear Model Predictive Control (*NMPC*) is being more and more used in these processes due to the increasingly stringent demands on product quality (HENSON, 1998). According to Henson (1998), another reason for the increase in the use of *NMPC* is the improvement in software and hardware capabilities which make it possible to use complex algorithms.

Moreover, according to Qin (2003), *NMPC* has several advantages. First, it is able to deal with the nonlinearities of the problem and, thus, increase the profit. Second, it might be based on either first principle models or empirical models. Finally, it can efficiently deal with constraints in the manipulated variables and controlled variables.

It is important to highlight that *NMPC* has also some drawbacks. First, the optimization problems are nonconvex, which means that the solution is much more complex since the local minimum can affect the performance and stability of the control. The optimization algorithm is much more complicated since a nonlinear problem must be solved online at each sampling period, so the computational effort is rather greater than in *LMPC*. Third, the study of the robustness and stability in *NMPC* is very complex. Therefore, *NMPC* technique should be used only when the benefits of the technique are greater than the disadvantages listed before.

Nonlinear models in *NMPC* can be derived either by first principle models or empirical models. First principle models are obtained by applying transient mass, energy or momentum balance to the processes (OGUNNAIKE; RAY, 1994). Fist principle models have some advantages over empirical models since less process data is required for their development because they are highly constrained with respect to their structure, and model parameters can be estimated from laboratory experiments and routine operating data instead of plant tests. However, first principle models are difficult to derive in large-scale process and the first principle modeling approach can be too complex to be useful in *NMPC* design (HENSON, 1998).

Empirical models are usually used in large-scale and complicated processes because they do not need a deep understanding of the process. Some examples of nonlinear models that can be used in *NMPC* are Hammerstein and Wiener models, Volterra models, polynomial

autoregressive moving average models (*ARMAX* models) and artificial neural networks models. Artificial neural networks are the most popular nonlinear models used in *NMPC* techniques due to their capacity to capture highly nonlinear dynamics of multivariable processes (HERMANSSON; SYAFIIE, 2015).

3.3. Model Predictive Control based on Artificial Neural Networks

The predictive control using neural networks appeared for the first time when Willis (1992) applied the model predictive control based on artificial neural networks (*MPC-ANN*) to control a nonlinear system in a distillation column. The objective of the work was to test the performance of the new control approach and to compare its performance to the conventional *PI* controller and with the Generalized Predictive Controller.

The system was multivariable and there were three inputs: the reflux ratio, the feed rate and the vapor rate in the reboiler. Besides, there was one output: the methanol molar fraction in the bottom product. The performance criteria used was the integral of the absolute error (*IAE*). The results showed that the *MPC-ANN* presented a great improvement in the performance compared to the other two techniques used (*GPC* and *PI*) since its *IAE* was 5.7 against 62.1 of the *GPC* and 78.1 of the PI.

Willis (1992) also compared the three controllers cited before for the situation that there were two controlled variables: the molar fraction of methanol in the bottom and the molar fraction of methanol in the top of the column. In this test, the *MPC-ANN* also performed better than the other two techniques since it had *IAE* of 3.0 and 22.4 for the top and bottom composition respectively. The *PI* presented an *IAE* of 11.52 and 31.58 for the top and bottom composition respectively. According to the authors, the better performance of *MPC-ANN* was due to the fact that *ANN* had the ability to predict interactions between the top and bottom loop.

Other recent studies showed the application of *ANN* on highly nonlinear problems. Hosen, Hussain e Mjalli (2011) used the hybrid model of artificial neural networks and first principle models to control the temperature of a batch reactor in a polymerization reaction to produce styrene. The results showed that the overshoot and the response time were smaller for the *MPC-ANN* technique when compared with the performance of a *PID*. Besides, the control action was smooth. The authors point out that linear controllers are not able to capture nonlinearities in the process (HOSEN; HUSSAIN; MJALLI, 2011).

Long *et al* (2014) applied the predictive control based on neural networks to control the temperature of a reactor of methylamine removal and compared its performance with the *PID*. They also found a better performance of the *MPC-ANN*.

Afram *et al.* (2017) used the *MPC-ANN* in the heating, ventilation and air conditioning. Others researchers used the technique to control the output temperature of a heat exchanger used in the petroleum pre-heating (VASIčKANINOVÁ; BAKOŁOVÁ, 2015).

Yu, Gomm and Williams (1999) investigated the use of predictive control based on neural networks in the neutralization reactor. The process had three controlled variables: temperature of the reactor, the pH and the dissolved oxygen. According to the authors the process was very difficult to control due to the large dead time, coupling interactions and the dissolved oxygen. The controller demonstrated a satisfactory performance to track the set point and to reject disturbances. Wior *et al.* (2010) applied an approximate predictive control strategy in a neutralization tank. The model used was the multi-layer perceptron networks based on NNARX models. Draeger, Engel e Ranke (1995) used a feedforward neural network as the nonlinear prediction model in an extended *DMC*-algorithm to control the pH value in a neutralization process. The authors used actual data of the process operating with a *PI* to identify the process.

Yu *et al.* (2006) stated that, although the application of neural networks in nonlinear processes are promising, they present some disadvantages such as the great computational effort, the long time needed to train the neural network, the uncertainties of the process generated by process variations and the fact that there is no accepted theory to tuning these controllers. The authors trained the neural networks offline and then applied an online modification of the controller parameters. The results showed a better performance of the *ANN* when compared with the *PID*.

Neural networks have already been applied in many processes with complicated dynamics with large dead times, inverse response, nonlinearities, and interactions between the manipulated and controlled variables. However, no work has studied the application of neural networks for multivariable level control in coupled tanks. Therefore, this work will use the structure of *MPC-ANN* to control levels of coupled tanks process since this process has nonlinear behavior and interaction between the levels.

3.4. Inverse Neural Control

Another neural network control structure that can be used in process control is the inverse neural control. This technique uses an inverse model to predict the manipulated variables, that is, the inputs of the process are the outputs of the model and the outputs of the process are the input of the model.

The inverse neural network modeling was applied by Fileti, Pacianotto e Cunha (2006) to adjust the end blow oxygen and coolant requirement in a basic oxygen steelmaking process in order to match the temperature and carbon percentage requirement. The results showed a better performance of the neural network structure proposed than the commercial model that was being used at the moment of the study.

Moreover, Eyng and Fileti (2010) developed a feedback-feedforward controller, based on neural network inverse models aiming to keep a low concentration of ethanol and water in the effluent gas phase from an absorption column. The authors compared the *ANN* controller with the *PID* for situations under uncertainties of 5 %, 10 % and 15 % in measurements and showed that the *ANN* controller outperformed the *PID*.

A scheme of the inverse neural network controller is shown in Figure 8. It is possible to note that the manipulated variables are calculated through the controlled variables and their setpoints.

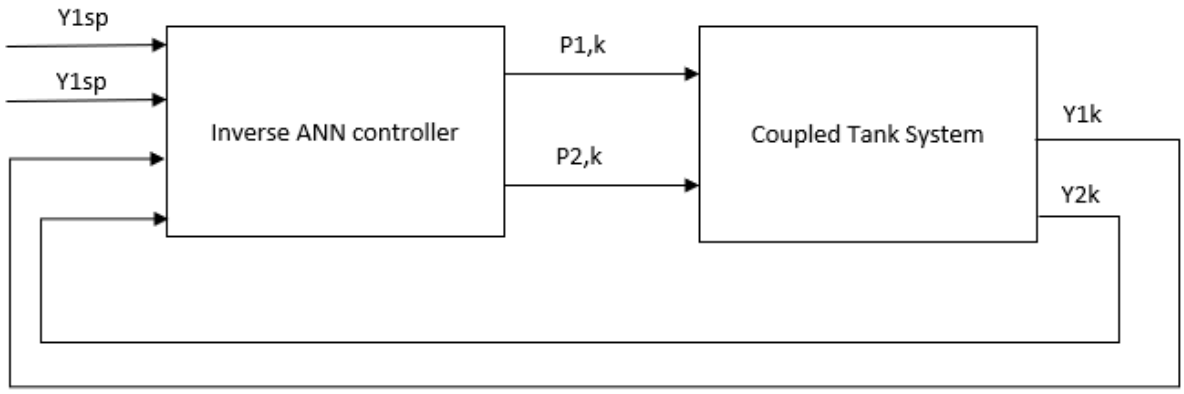

Figure 8: Block Diagram of Inverse Artificial Neural Network controller.

3.5. *PID*

Proportional Integral and Derivative (*PID*) is a feedback three-mode controller that was developed in the 1930s and became widely used in industry in 1940s. The first computer control applications were developed in the 1950s and digital *PID* has been used in industry since the 1980s.

The basic block diagram of a *PID* controller is shown in Figure 9 where the transfer functions representing each part of the process is shown. The setpoint ($y_{sp}(s)$) is compared with the measured output ($y_m(s)$) of the process (controlled variable) generating the error signal ($e(s)$). The error is the input of the *PID* controller represented by the transfer function $G_c(s)$, which generates the control signal ($c(s)$). The control signal is then transmitted to the actuator

that is represented by the transfer function $G_v(s)$, which gives the manipulated variable ($m(s)$). The manipulated variable actuates in the process that has the controlled variable as an output ($y(s)$). Finally, the controlled variable is measured by a sensor ($y_m(s)$). The disturbance is represented by "$d(s)$" and the disturbance transfer function is represented by $G_d(s)$.

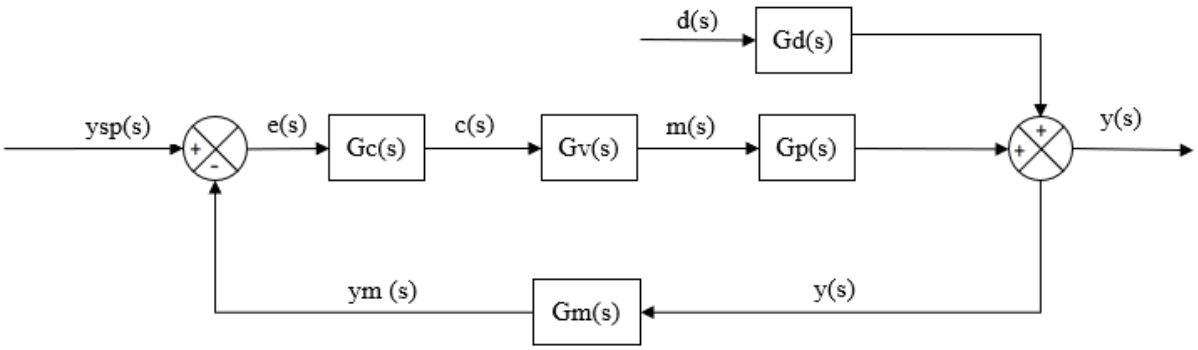

Figure 9: Block Diagram of *PID* loop.

The first control action of *PID* is the proportional action which actuates reducing the response time of the system and the final error. However, by increasing this action the system may become unstable or highly oscillatory. The second action in the *PID* controller is the integral action, which can reduce the offset to zero. Nevertheless, by increasing the integral action the system tends to oscillate and may become unstable. Therefore, the third action in *PID* is the derivative action, which can reduce oscillations and decrease the time response increasing the performance of the system. Therefore, the three actions work together to improve the transient response of the system in order to reject disturbances and track the setpoint changes. The Equation 8 shows the calculation of the control signal in the *PID* controller.

$$c(t) = u_{ss} + K_c\left(e(t) + \frac{1}{\tau_i}\int_0^t e(t)dt + \tau_d\frac{de(t)}{dt}\right)e(s) \tag{8}$$

Where K_c is the gain of the controller, τ_i is the integral time and τ_d is the derivative time of the controller.

A good performance of the control system depends on the tuning parameters choice (K_c, τ_i, τ_d). So, a small overshoot, a zero offset, and fast response can only be achieved with a reasonable choice of tuning parameters. There are some well-known methods to tune *PID* like the methods based on the response curve of the system, methods that are based on integral error criteria, methods based in frequency response like Bode Diagram and Nyquist Diagram.

In the methods based on the curve response, the control loop is "opened" and some perturbations in the manipulated variable are performed in order to get a model of the system that relates the input and output variables. This methodology is known as identification and is shown in Figure 10. The "$G_{proc}(s)$" function represents the product of the actuator, process and sensor transfer function. After the identification process, the dead time, the time constant and process gain are used to calculate the tuning parameters through correlations like Ziegler-Nichols correlation.

The *PID* controller can also be tuned using techniques that use the stability limit like the Ultimate Gain technique. In this technique, the system is kept in closed loop and the controller gain (K_c) is increased until the moment that the system reaches the stability limit.

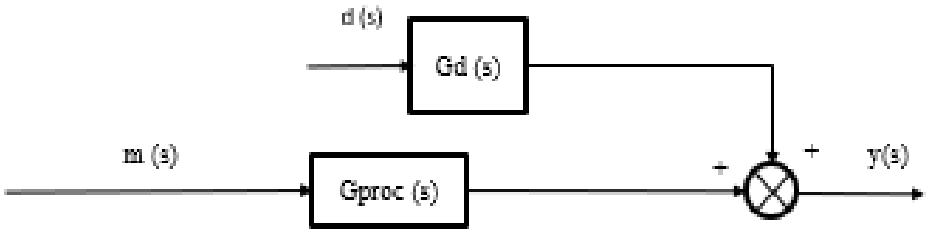

Figure 10: Block diagram of an open loop control system.

After using Ultimate Gain or Ziegler-Nichols correlation to find the initial guesses of tuning parameters, it is possible to run some simulations with models of the process (G_{proc}) and the control (G_c) in order to find out a set of control parameters that give a satisfactory performance. This proceeding is known as fine-tuning.

There are some parameters that can be used to assess the performance of the system. They are called performance criteria. According to Stephanoupoulos (1983), there are two kinds of performance criteria: the steady state performance criteria and the dynamic response performance criteria.

The most important steady state performance criteria is the offset, that is, the final error of the system or the error of the system in the steady state. A good controller will drive the offset of the system to zero.

There are two kinds of dynamic performance criteria: the ones that use only a single point of the response curve such as the overshoot and the response time and the ones that use the entire closed loop response like integral absolute error (*IAE*), integral of time-weighted

absolute error (*ITAE*) and the integral of square error (*ISE*), which are shown in Equation 9, 10 and 11 respectively.

$$IAE = \int_0^\infty |e(t)|dt \tag{9}$$

$$ITAE = \int_0^\infty e^2(t)dt \tag{10}$$

$$ISE = \int_0^\infty e^2(t)dt \tag{11}$$

Although the feedback control has many positive features, it has an inconvenient characteristic. If the controller is not well designed and the operation conditions of the system change after the implementation of the controller, the controller may lead the system to instability, that is, a limited input can lead to an unlimited output. Therefore, it is extremely important to understand under which conditions the controller may become unstable.

Equation 12 shows the relationship between the controlled variable with setpoint changes and disturbance variable. The poles of the equation or the roots of the denominator provide information about the stability of the system. If all poles have the negative real part the system is stable. Otherwise, the system is unstable.

$$Y(s) = \frac{G_c(s)G_f(s)G_p(s)}{1 + G_c(s)G_f(s)G_p(s)G_m(s)}Y_{sp}(s) + \frac{G_d(s)}{1 + G_c(s)G_f(s)G_p(s)G_m(s)}D(s) \tag{12}$$

The criteria shown above is true only for linear systems. However, this criteria provide important information for nonlinear systems that works near to the operation point.

One of the methods that can be used to determine the poles of the system is the root locus. Root locus is simply the plot of the roots of the characteristic equation as the gain K_c is varied. Therefore, it is possible to determine which values of K_c lead the system to instability.

Multiple-Input Multiple-Output (*MIMO*) problems are very usual in the modern industry since there are many variables to control. The control of these processes is usually more complicated than Single-Input Single-Output (*SISO*) problem due to process interactions

between the control loops, which means that one manipulated variable can affect all controlled variables.

One way of dealing with *MIMO* control problems is to use one feedback controller (like the *PID*) to control each control loop. This technique is known as a multiloop control variable and it raises a lot of questions like which manipulated variable should be used to control which controlled variable? Will the interactions between the control loops cause problems?

In order to answer these questions, the relative gain analysis (*RGA*) concept is usually used. The relative gains between the controlled variable y_i and the manipulated variable u_j are defined as the dimensionless ratio between the two steady-state gains. They are calculated through Equation 13.

$$\lambda_{ij} = \frac{\left(\frac{\partial y_i}{\partial u_j}\right)_u}{\left(\frac{\partial y_i}{\partial u_j}\right)_y} = \frac{open\ loop\ gain}{closed\ loop\ gain} \tag{13}$$

For a generic system with n input variables and n output variables, the relative gain array can be calculated through the matrix transfer function of the plant ($G_p(s)$) as shown in Equation 14.

$$\Lambda = G_p(0) \otimes \left(\left(G_p(0)\right)^{-1}\right)^T \tag{14}$$

For a system with two inputs and two outputs (2x2), equation 14 can be rewritten to Equation 15, and $\boldsymbol{\lambda}$ can be calculated through Equation 16.

$$\Lambda = \begin{bmatrix} \lambda & 1-\lambda \\ 1-\lambda & \lambda \end{bmatrix} \tag{15}$$

$$\lambda = \frac{K_{11}K_{22}}{|K|} \tag{16}$$

The pairing between the controlled and manipulated variable should be chosen in order to reduce the process interactions between the control loops. Therefore, the ratio between the open loop gain and closed loop gain should be as close to one as possible. Large relative gains mean that u_j does not have a great influence in y_i when the loop is closed, which could make the control difficult. Besides, $\boldsymbol{\lambda} < 0$ may lead the system to instability and must be avoided.

In order to improve the control performance of multivariable control is to use loop decoupling. This strategy, shown in Figure 11, consists of using transfer functions called decouplers to eliminate loop interactions. The decouplers transfer function is shown in Equations 17 and 18.

$$T_{21} = -\frac{G_{p21}}{G_{p22}} \tag{17}$$

$$T_{12} = \frac{G_{p12}}{G_{p11}} \tag{18}$$

The use of decouplers will not always represent an improvement in the performance of the system since the transfer functions might not successfully represent the process, especially in nonlinear systems. The ability of the decoupling technique improves the performance by eliminating the interactions between the loops of the system is limited to the accuracy of the model of the transfer function.

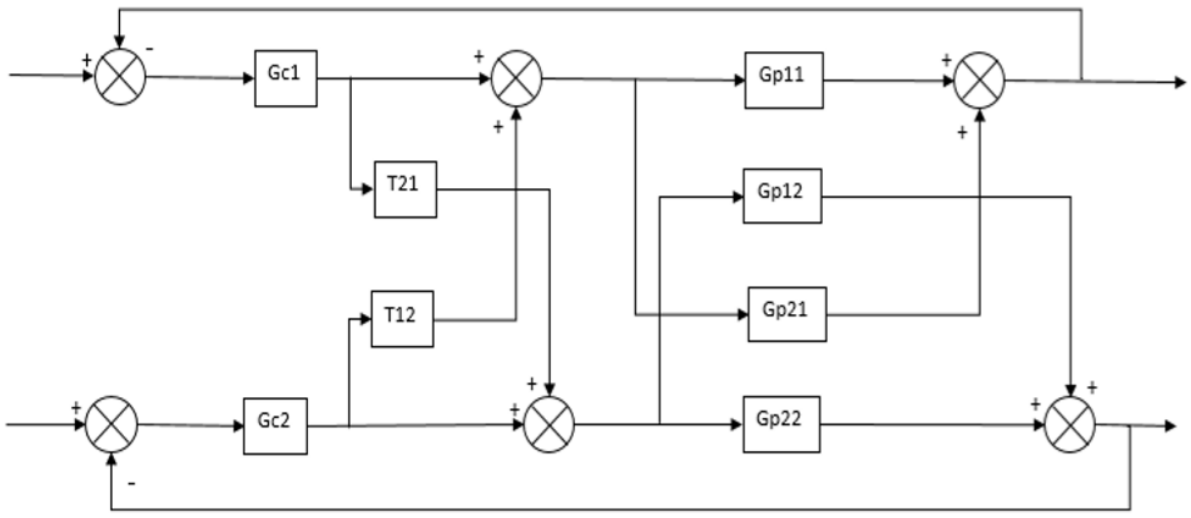

Figure 11: Multivariable control loop with loop decoupling.

4. Methodology

4.1. Process Description

The experiments were performed using an interactive tank plant placed in the Laboratory of Control and Automation of Processes (*LCAP*), which is shown in Figure 12.

Figure 12: Experimental scheme of coupled tanks.

The process has two controlled variables: the level of each upper tank. Besides, there are two manipulated variables: the power of the two pumps (P-101 and P-102). Therefore, the process is a *MIMO* process, with two inputs and two outputs.

Figure 13 shows the instrumentation diagram of the coupled tanks system used in this work. The hand valves HV-101 and HV-102 regulate the flow in the tanks V-101 and V-102. The hand valve HV-103 has the purpose of communicating two tanks, adding nonlinearity to the system and adding process interactions between the two controlled variables. The pumps P-101 and P-102 can manipulate the flow that gets in each tank (Figure 14).

The levels of the vessels were obtained by the level pressure transducers represented by LT-101 and LT-102 Siemens, series MPX5010 plugged at the bottom of the tanks (Figure 14). Delta *PLC*, model DVP20EX3 (YIC-101), shown in Figure 14, read the data and sent to the computer, where a MATLAB® program calculated the control action of both pumps sending the sign back to the plant, hence closing the loop. It is important to point out that vessel V-103 has only the purpose of recirculating the water.

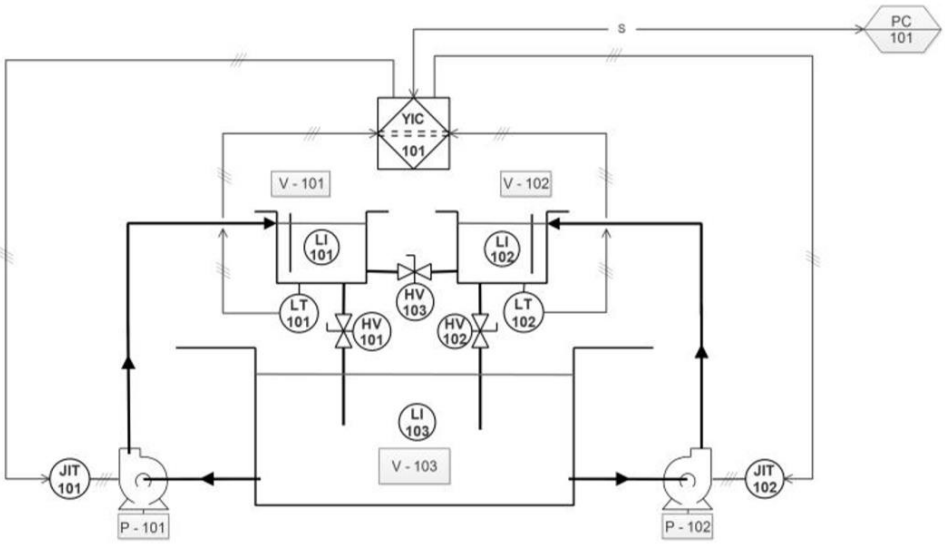

Figure 13: Instrumentation Diagram of coupled tanks.

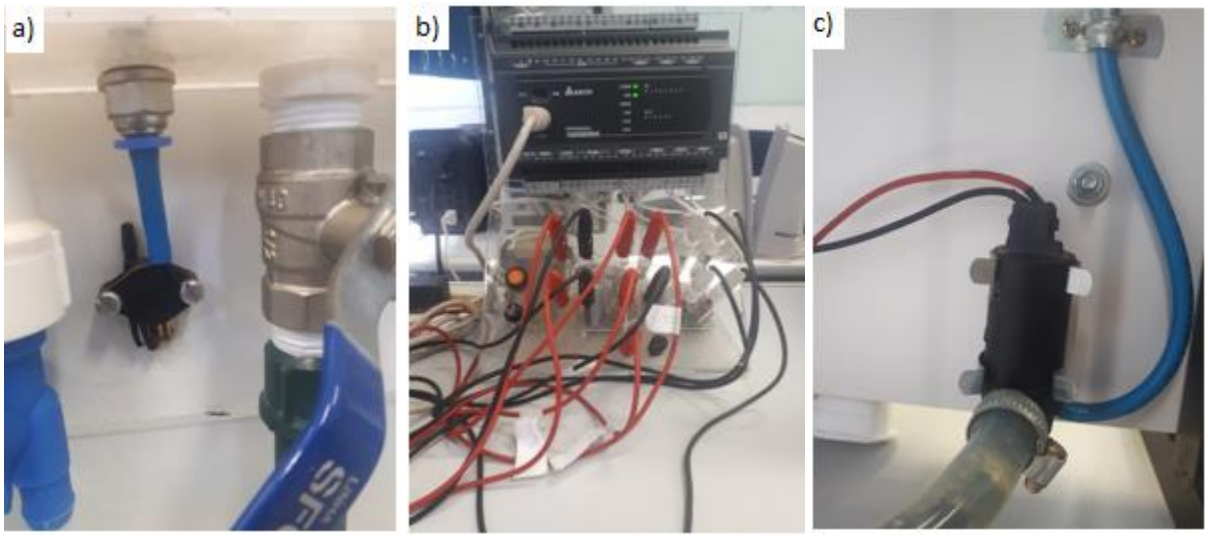

Figure 14: a) Level pressure transducers. b) *PLC*, model DVP20EX3. c) Pump used in this work.

4.2. *PID* Controller

The first step of the design of the *PID* was to construct the calibration curve in order to transform the levels given in bits from the level pressure transducers in actual level values. After that, the design of *PID* followed the following steps: identification, control and tuning using multiloop *PID* and decoupled multivariable *PID*; and the analysis of stability and linearity of the loop.

4.2.1. Process Identification

The identification procedure is the development of empirical dynamic models from input-output data of the process. Usually, this is the most time-demanding step in the application of control techniques in industrial processes. In *PID* applications, this step is important to derive initial guesses for the controller parameters from the correlations shown in Table 2. Moreover, this model can be used to derive decouplers equation through Equations 13 and 14.

For the identification process, the hand valves HV-101, HV-102 and HV-103 were kept partially opened in a fixed position since a change in the hand valve position would change the model of the plant. The hand valves HV-101 and HV-102 were opened until the level of each tank reached about 95% when the power of each pump (P-101 and P-102) was 100%. Besides, the hand valve that made the connection between the vessels (HV-103) was maintained partially opened so that the level of the tank one (LT-101) was about 65% when the power of pump P-101 (power 1) was 100% and the power of pump P-102 (power 2) was 0%. Therefore, the partial opening of HV-103 caused interactions between the controlled variables. The flow chart explaining the determination of the valves opening is shown in Figure 15.

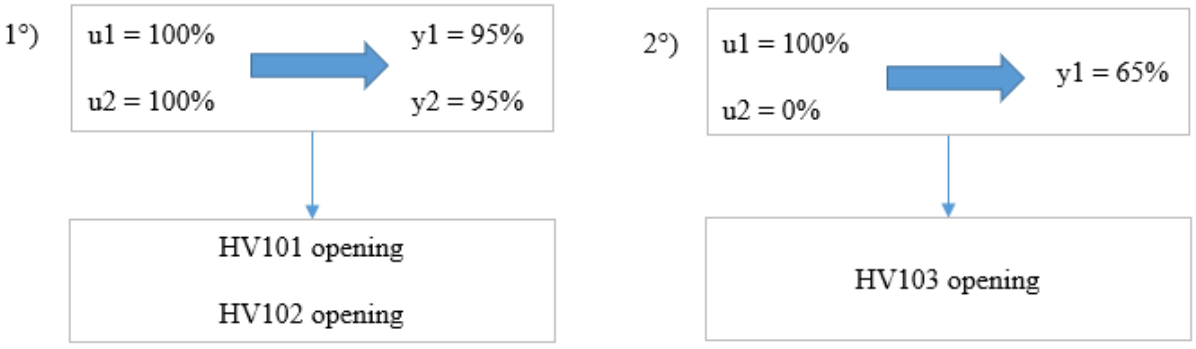

Figure 15: Flow chart used to determine the position of the valves.

One model that can be used to identify processes that have S-curve responses is the transfer function model first-order-plus-dead-time (*FOPDT*). Aiming to identify the model, the open loop configuration of Simulink® shown in Figure 16 was used. Two identification experiments were performed, the first one was to keep the power of P-102 constant while varying power of P-101 from 30% to 50% through a step. The second experiment was to vary the power of P-102 from 30% to 50% through a step change while keeping the power of P-101 constant. In the first experiment, the transfer functions $G_{p11}(s)$ and $G_{p21}(s)$ of the Figure 11 are obtained while in the second experiment $G_{p12}(s)$ and $G_{p22}(s)$ are obtained.

The step in the power of each pump was performed only after the levels had reached the steady state and the total time of the experiment was 450 seconds, which was enough for the levels to reach a new steady state.

The block "OPC write" is the block that sent the power signal to the *PLC*, while the "OPC read" is the block that read the data from the plant.

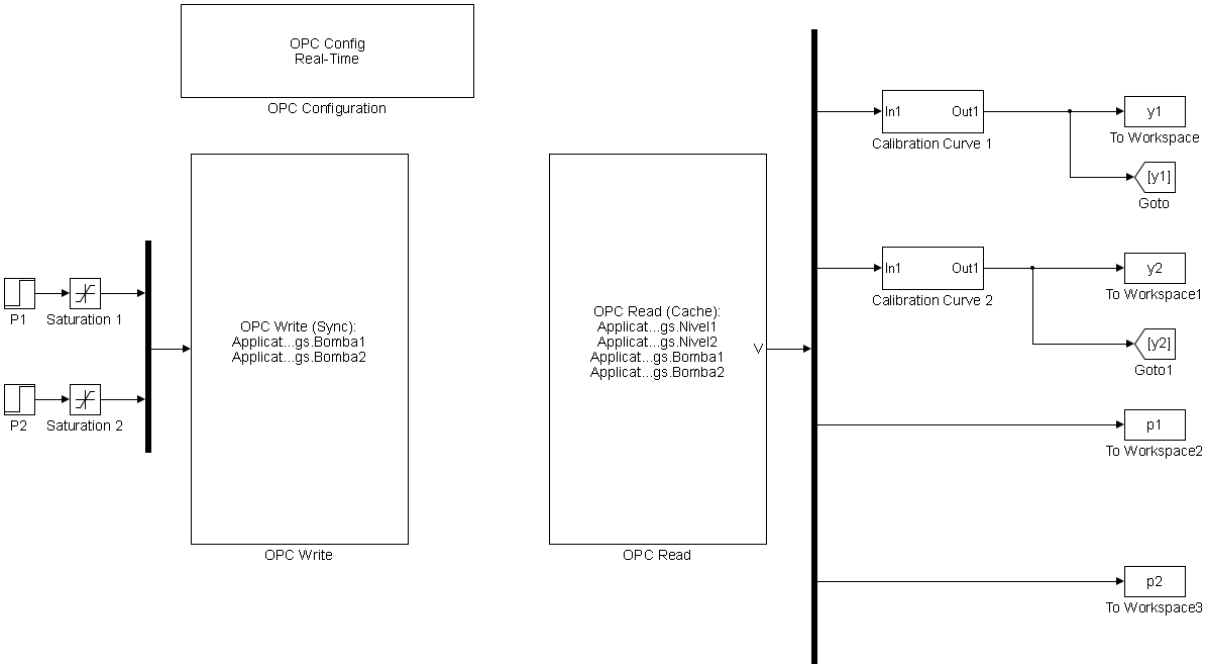

Figure 16: Simulink® Diagram of the Identification Process.

After the step in each pump, the fitting in the *FOPDT* curve can be performed using minimization of the sum of squared errors as shown in Equation 19. The "f" represents the value predicted by the *FOPDT* equation (shown in Equation 20) and "$y_{deviant}$" represents the deviation of the level data collected from the plant through a Simulink program shown in Figure 16 and the level in the steady state before applying the step.

$$F_{objective} = \sum (f - y_{deviant}) \tag{19}$$

$$f = A\left(1 - \left(1 + \frac{t}{B}\right)e^{-\left(\frac{t}{B}\right)}\right) \tag{20}$$

The next step is to calculate the three parameters of response curve (the process gain, the time constant and the dead time) using the graph of the response curve and the parameters A, B. The process gain is calculated through Equation 21.

$$K_p = \frac{A}{D} \tag{21}$$

Where D is the step in the manipulated variable (20%) and A is the parameter obtained from Equation 20. The parameter t_d (dead time) was determined through the crossing of the tangent line of the point of maximum slope with the time axis. The time constant (τ) was gotten through the subtraction between the time that the tangent line crosses the straight line of the new steady state and the dead time. This proceeding is shown in Figure 17.

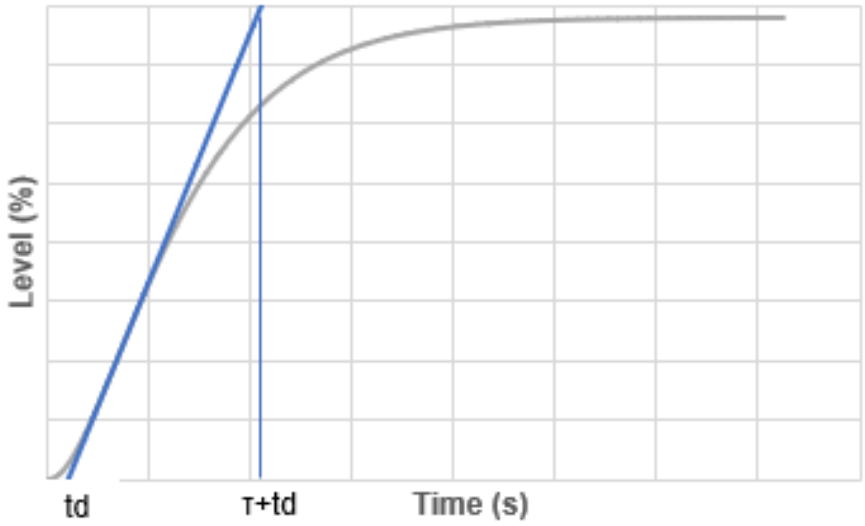

Figure 17: *FOPDT* response curve.

The final transfer function model obtained (*FOPDT*) is shown in Equation 22.

$$G_p = \frac{K_p}{\tau_p s + 1} e^{-td.s} \tag{22}$$

The parameters of the transfer function model (*FOPDT*) can be used to calculate the tuning parameters of the *PID* (K_c, τ_i, τ_d) through the Ziegler-Nichols correlation while the Ultimate gain correlation is based on the parameters obtained using the proportional contoller in limit of stability: the period of oscilation (T_c) and the ultimate gain.

Table 2: Tuning relationships for *PID*.

	K_c	τ_i	τ_d
Ziegler-Nichols	$\frac{1.2\tau}{Kt_d}$	$2t_d$	$\frac{t_d}{2}$
Ultimate Gain	$0.6\ K_{pc}$	$\frac{T_c}{2}$	$\frac{T_c}{8}$

4.2.2. *PID* Control

After the process identification, the *PID* multiloop control was performed using the block diagram shown in Figure 18. The subsystems near the y1 and y2 blocks represent the filter of data that was designed due to the noise of the sensor signal. The error subsystem was used to calculate the integral of error performance criteria shown in Equation 9, Equation 10 and Equation 11 (*IAE*, *ITAE*, and *ISE*). Besides, *PID* subsystem is shown in Figure 19.

The servo problem was performed changing the set point of level 1 and level 2 through step tests. The sequence of set points used in level 1 was 0%, 50%, 20%, 50%, 70%; while the sequence of set points used in level 2 was 0%, 60%, 40%, 30%, 70%. The steps were applied every 300 seconds and simultaneously, which means that after 300 seconds setpoint of level 1 was changed from 50% to 20% and setpoint of level 2 was changed from 60% to 40%. Therefore, the total time of the experiment was 1200 seconds. The set points of the two levels were varied to different points in order to make the problem of control more difficult due to the interactions between the two levels.

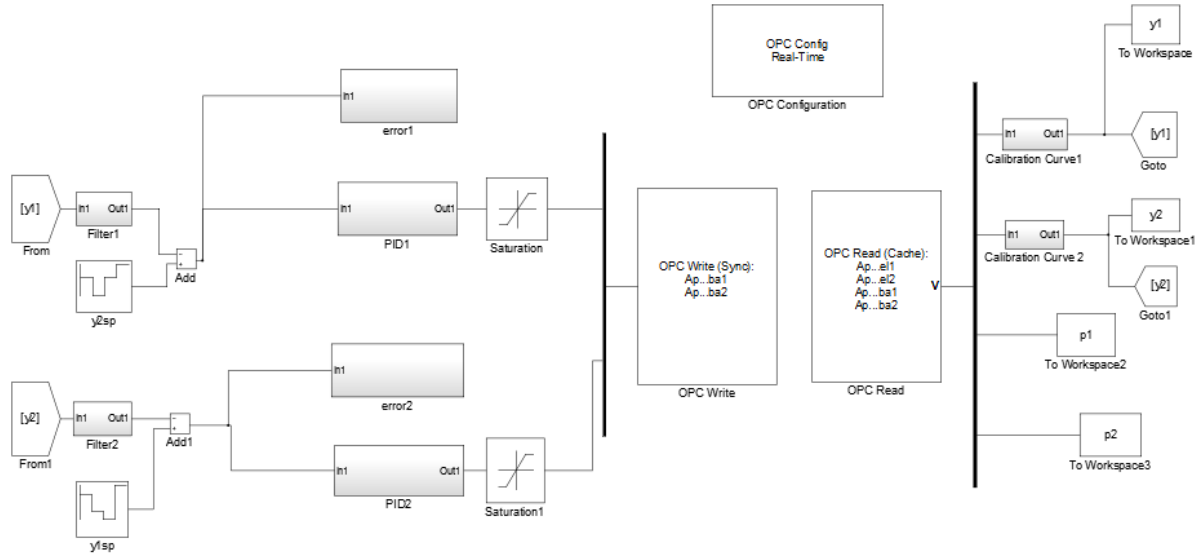

Figure 18: Block Diagram of the multiloop *PID* control.

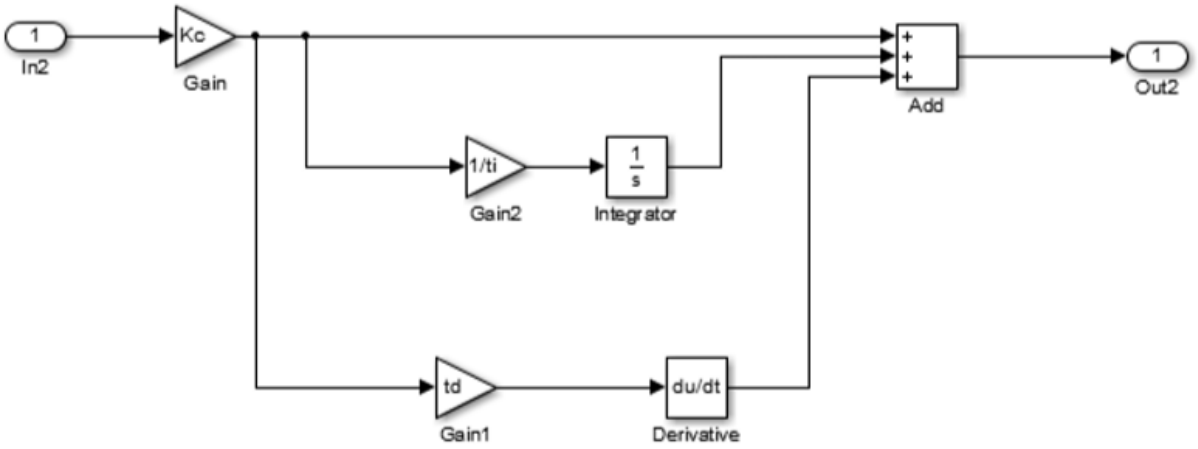

Figure 19: Block diagram of the *PID* transfer function.

Aiming to compensate process interactions between the variables, the decouplers were designed using Equations 17 and 18. The same sequence of setpoint changes that was performed for the *PID* multiloop control was also performed for the *PID* multivariable control (with decouplers) so that the performance between the two strategies could be compared. Figure 20 shows the block diagram of *PID* multivariable control using decouplers.

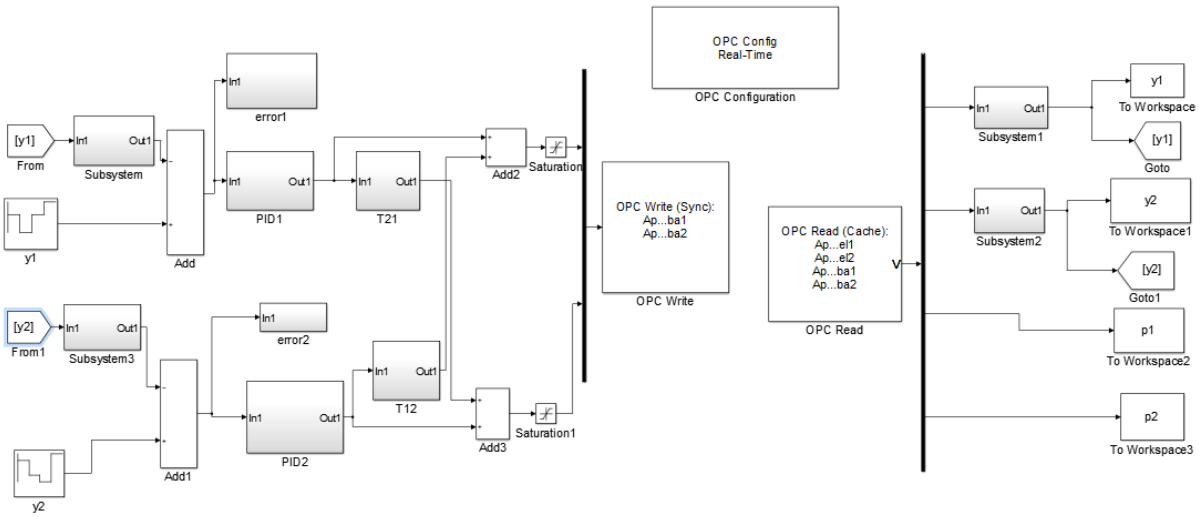

Figure 20: Block diagram of Multivariable *PID* control with decoupling technique.

4.2.3. Linearity and Stability

In order to verify the nonlinearities of the system, four linearity experiments were performed: one positive step and one negative step for the power of each pump. In the first experiment, the power of Pump P-101 was varied from 50% to 60%; while the power of pump P-102 was kept constant at 50%. In the second experiment, the power of Pump P-101 was varied from 50% to 40%; while the power of pump P-102 was kept constant at 50%. In the third experiment, the power of Pump P-102 was varied from 50% to 60%; while the power of pump P-101 was kept constant. In the fourth experiment, the power of Pump P-102 was varied from 50% to 40%; while the power of pump P-101 was kept constant.

In order to carry out the stability analysis, transfer functions found in the process identification step were used. The critical gain was then obtained through the Root Locus method and the Padé approximation, shown in Equation 23, was used to get a rational function, as shown in Equation 24.

$$e^{-tds} = \frac{{t_d}^2 s^2 - 6t_d + 12}{{t_d}^2 s^2 + 6t_d + 12} \tag{23}$$

$$G_{proc} = \frac{K_p}{\tau_p s + 1}\left(\frac{{t_d}^2 s^2 - 6t_d + 12}{{t_d}^2 s^2 + 6t_d + 12}\right) \tag{24}$$

Another way that can be used to analyze the stability is to simulate the process, replacing the *PID* block to a gain block (K_{cr} block) and increasing the gain until the limit of stability. The block diagram of this simulation is shown in Figure 21.

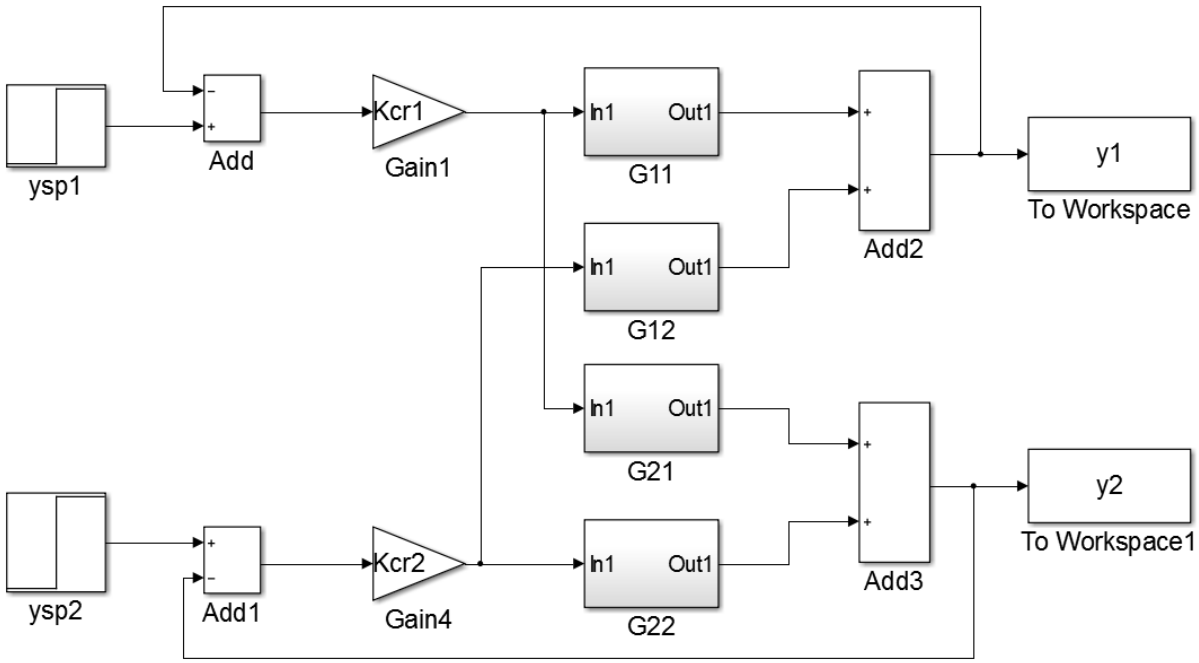

Figure 21: Stability analysis through simulation.

4.3. Model Predictive Control based on Artificial Neural Network

The *MPC-ANN* design can be divided into three parts: the definition of *ANN* setting, the identification process to obtain the model based on input/output data and the control and tuning to obtain a set of parameters of the MPC that provides a good performance.

4.3.1. *ANN* setting

According to Souza (1993), two different architectures can be used in the prediction of the long horizon. The first strategy is to predict all the N_p future values of each output in a single step. So, this neural network will have $2N_p$ ($Y_{1,k+1}$, $Y_{1,k+2}$, …, $Y_{1,k+Np,}$ $Y_{2,k+1}$, $Y_{2,k+2}$, …, $Y_{2,k+Np}$) outputs as it is shown in Figure 22.

The second strategy is to predict just output one-step-ahead using past and present output and input and then feedback this information moving forward in the time horizon, so that in the second time horizon the $Y_{1,k}$ is substituted by $Y_{1,k+1}$,$Y_{1,k+1}$ is replaced by $Y_{1,k+2}$ and so on. This recurrent proceeding is performed until all the N_p values of each output have been calculated. Figure 23a, Figure 23b, and Figure 23c shows this setting.

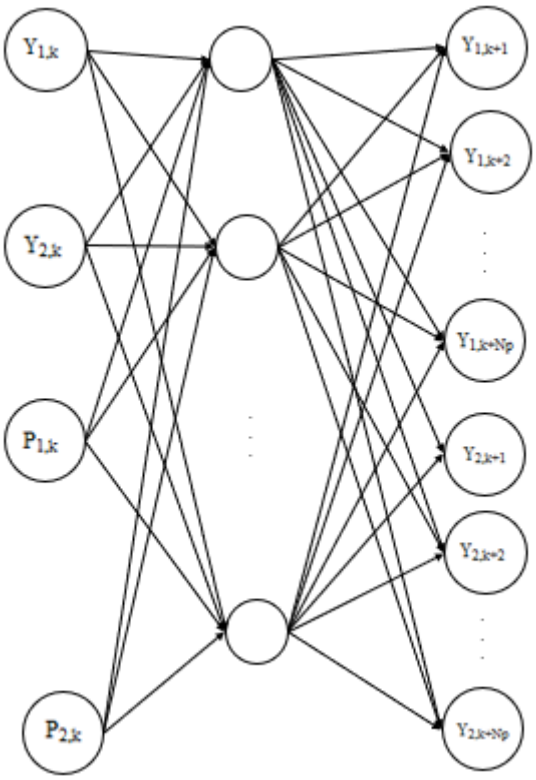

Figure 22: Neural network setting used to predict outputs in the prediction horizon.

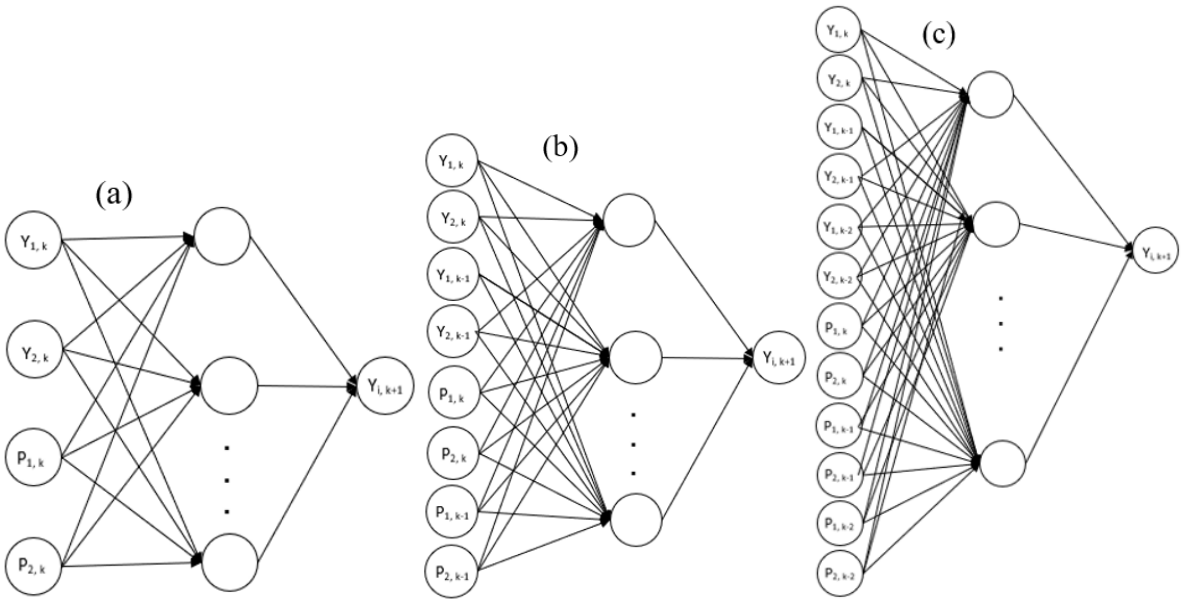

Figure 23: Architecture of the *ANN* using four inputs (a); eight inputs (b); twelve inputs (c).

According to Souza (1993), the first method has several drawbacks. The first problem is that this structure produces greater networks since its architecture is much more complex. Second, this setting is less flexible than the second one since a new training of *ANN* will have

to be performed if a new prediction horizon is picked. Third, the modeling error of this structure is greater than the second structure.

As the process has two outputs (the level of both vessels) and two inputs (the power of both pumps), three network architectures were proposed. The first option is to calculate the level one-step-ahead ($Y_{i,k+1}$) by using four network inputs: the level of each tank in the current moment ($Y_{1,k}$ and $Y_{2,k}$), and the power of each pump in the current moment ($P_{1,k}$ and $P_{2,k}$) as depicted in Figure 23a. The subscript "i" represents the level one or level two, since two networks are used, one for the prediction of each tank level. The second option is to calculate the level one-step-ahead using eight network inputs: the level of each tank in the current moment, the level one-step-delayed ($Y_{1,k-1}$ and $Y_{2,k-1}$), the power of the pump in the current moment and the power of the pump one-step-delayed ($P_{1,k-1}$ and $P_{2,k-1}$) as illustrated in Figure 23b. The third option is to use twelve inputs to calculate the level one-step-ahead ($Y_{i,k+1}$): the level of each tank at the present moment and one and two-step delayed and the power of the pump in the current moment and one and two-step delayed as shown in Figure 23c.

4.3.2. *MPC-ANN* Identification

The identification process could be divided into two phases: the build-up of the data-set and the training process aiming to obtain the best neural network architecture.

The data-set was built using the Simulink® program shown in Figure 24. Y1p, y2p represent the level 1 and 2 at the current time; y1p1 and y2p1 represent levels 1 and 2 at one-step-delayed; u1p and u2p represent the power of pumps P-101 and P-102 at the current moment; and u1p1 and u2p1 represent the power of pumps P-101 and P-102 one-step-delayed. The subsystems of the Simulink block diagram are the calibration curves.

Open loop perturbations were performed simultaneously in the power of Pump P-101 and P-102 et every 180 seconds (about three times the process time constant) and they are represented in Figure 24 by the block "repeating sequence stair". These perturbations were step tests. Random values of input were chosen from 20% to 80% of the maximum power since the actual power did not change after 80% and the real power was zero bellow 20% of maximum power. Therefore, the identification experiment aimed to use the ability of nonlinear mapping of the neural networks to model the behavior of the system in the entire range of possible levels.

One point was obtained at every Δt seconds (where Δt is the sample time). The total time of the identification process was 3600 seconds in order to get enough points to capture the dynamic behavior and to avoid overfitting. Therefore, the number of data points was about $3600/\Delta t$. Since three values of sampling time were tested in this work 3 seconds, 5 seconds and

7 seconds; the number of data points used in the neural network training was respectively 1200, 720 and 514.

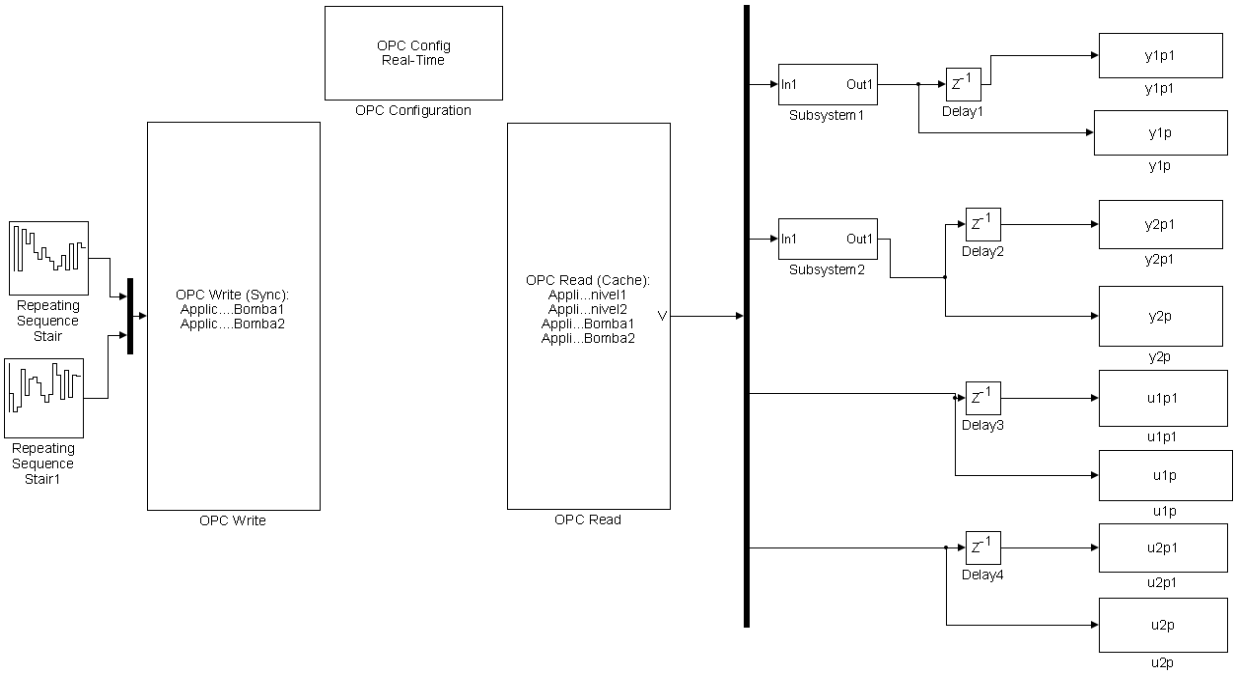

Figure 24: Simulink block diagram used to collect identification data.

The second part was to train a neural network to find the best architecture possible. In this process, the dataset collected was divided into two sets: the training set, which had 70% of the data points and the test set, which had 30% of the data points. The validation set was not necessary, because the strategy used to avoid overfitting was the Bayesian regularization, so the learning algorithm used was the Levenberg-Macquart algorithm with Bayesian regularization, that is the function *trainbr* of Matlab®.

Some *ANN* parameters were varied such as the number of hidden layers, the number of neurons in each hidden layer and the activation function used. The number of hidden layers tested was one and two, while the number of neurons ranged from one to five since a greater number of parameters did not improve the performance of *ANN*. Besides, three activation functions were tested: the tan-sigmoid function (*tansig* in Matlab®), the log-sigmoid function (*logsig* in Matlab®) and the linear function (*purelin* in Matlab®). The training algorithm is shown in Appendix A1.

The parameter used to determine the performance of the network was the mean square error of the test set (MSE_{test}).

4.3.3. *MPC-ANN* Control

The control block diagram is shown in Figure 25, in which the block "interpreted Matlab function" is the *MPC-ANN* algorithm, shown in Appendix A2. This block calculates the manipulated variables from the information of past and present input and output information and the setpoint information. The variables y_{sp1} and y_{sp2} represent the set points of level 1 and level 2; *y1p1* and *y2p1* represent the measured levels of one-step-delayed, *y1p* and *y2p* represent the value of each level at the present moment; *u1p* and *u2p* represent the power of pump P-101 and P-102 at the present moment; while *u1p1* and *u2p1* are the power of pump P-101 and P-102 one-step-delayed; *ynn1* and *ynn2* represent the level predicted by the neural network in the last step, namely $y_1(k|k\text{-}1)$, $y_2(k|k\text{-}1)$. These last values were used to correct the model inaccuracies and the effect of unmeasurable disturbances through the disturbance model like it is shown in Equations 6 and 7.

Therefore, by using this model, the discrepancy between the current measured output ($y_p(k)$) and the output predicted by the neural network model in the last step ($y_m(k|k\text{-}1)$) is calculated based on Equation 6 and the predicted output of the next step is then corrected with the discrepancy, according to Equation 7.

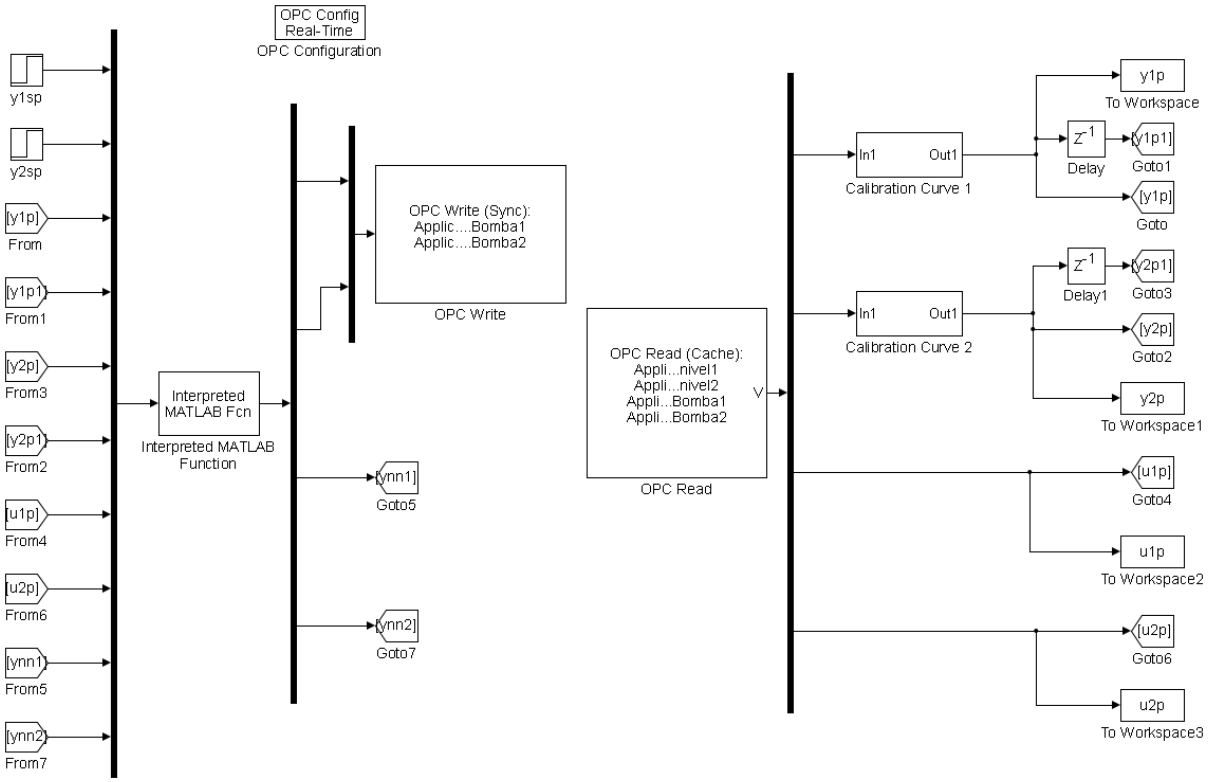

Figure 25: Simulink diagram used to control the system by *MPC-ANN*.

In order to find the set of MPC parameters that gives a reasonable performance, the closed-loop experiments were performed varying the setpoint of level 1 from 0 % to 50 % and the setpoint of level 2 from 0% to 60%. The control horizon was varied from 1 to 4 and the prediction horizon was kept at four times the control horizon ($N_p = 4N_c$). Three weight of control action (w_1, w_2) were tested; 0.01, 0.1 and 1, while the weight of control error (w_{y1}, w_{y2}) was fixed at 1. The total time used in the tuning process was 300 seconds, which was the time necessary for the system to settle down.

After the tuning part, the closed-loop experiment was performed in order to compare the *MPC-ANN* controller to the *PID* and inverse model neural network controllers. The experiment was the same of the *PID* part, in which the set point of level 1 was changed every 300 seconds and had the following values 0%, 50%, 20%, 50%, 70%; and the set point of level 2 was simultaneously varied every 300 seconds and assumed the following values 0%, 60%, 40%, 30%, 70%.

4.4. Inverse Neural Network Control

The control based on inverse neural network model calculates the manipulated variables of the process from the controlled variables. Therefore, the artificial neural network computes the power of the pumps P-101, P-102 using the value of current and past level. It is possible to divide the design of the inverse neural network control into three phases: the definition of inputs and outputs, the identification and the control.

4.4.1. Definition of Inputs and Outputs

As the Inverse Neural Network calculates the inputs from the outputs, the simplest structure possible is shown in Figure 26. The only inputs are the current level ($y1_k$ and $y2_k$) and one step delayed level ($y1_{k-1}$ and $y2_{k-1}$) and the outputs are the power of pump P-101 and P-102: P1 and P2. In this work, we used this structure because more complex structures don't make any improvement in the neural network performance.

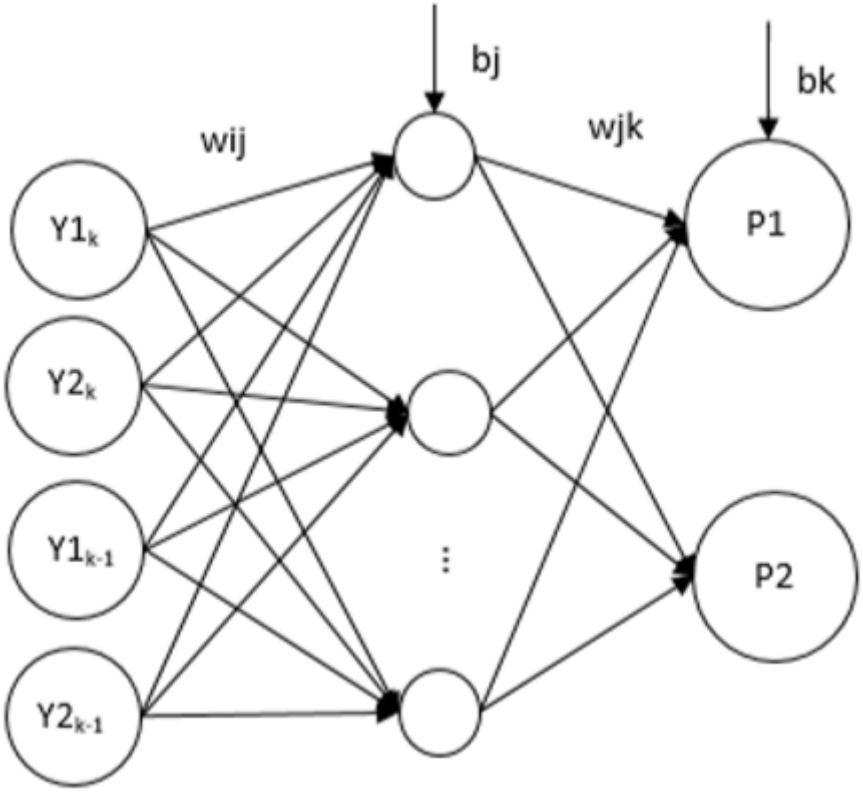

Figure 26: Inverse neural network structure.

4.4.2. Inverse Neural Network Identification

The identification process for Inverse Neural Network is similar to the identification for the *MPC-ANN*. The first step is to collect data from open-loop perturbations in the power of pumps P-101 and P-102. The hand valves (HV-101, HV-102 and HV-103) were kept constant during the build-up of the dataset. Figure 24 shows how the data was collected. The difference here is that the manipulated variables were varied at every 300 seconds instead of 180 seconds, which was the settling time.

In order to collect a satisfactory amount of data, the experiment was planned to have 33900 seconds.

Since the model accuracy is critical for this method, the training procedure is very important. The parameters varied during training were the same of *MPC-ANN* technique, that is: the activation function (tan-sigmoid, log-sigmoid or linear), the number of hidden layers (one and two) and the number of neurons in each hidden layer. The dataset was divided into only two parts: the training set and the test set since Levemberg-Macquart with regularization (*trainbr* in Matlab®) was the learning rule used in order to avoid overfitting. The criteria used to determine the best neural network structure was the minimization of mean square error of the test set. The algorithm used to train the neural network is shown in Appendix A3.

4.4.3. Inverse Neural Network Control

The implementation in Simulink of the Inverse Neural Network Control is shown in Figure 27. The block "INVERSE NEURAL CONTROL" comprises the control algorithm, shown in appendix A4, in which the control action is calculated.

The structure of the neural network employed was similar to that shown in Figure 26. The main difference here is that the inputs are anticipated one step, which means *Y1k*-1 and *Y2k-1* are replaced by *Y1k* and *Y2k*, represented as *y1* and *y2* in Figure 27. The inputs *Y1k*. and *Y2k* are replaced by inputs y_{1sp} and y_{2sp}.

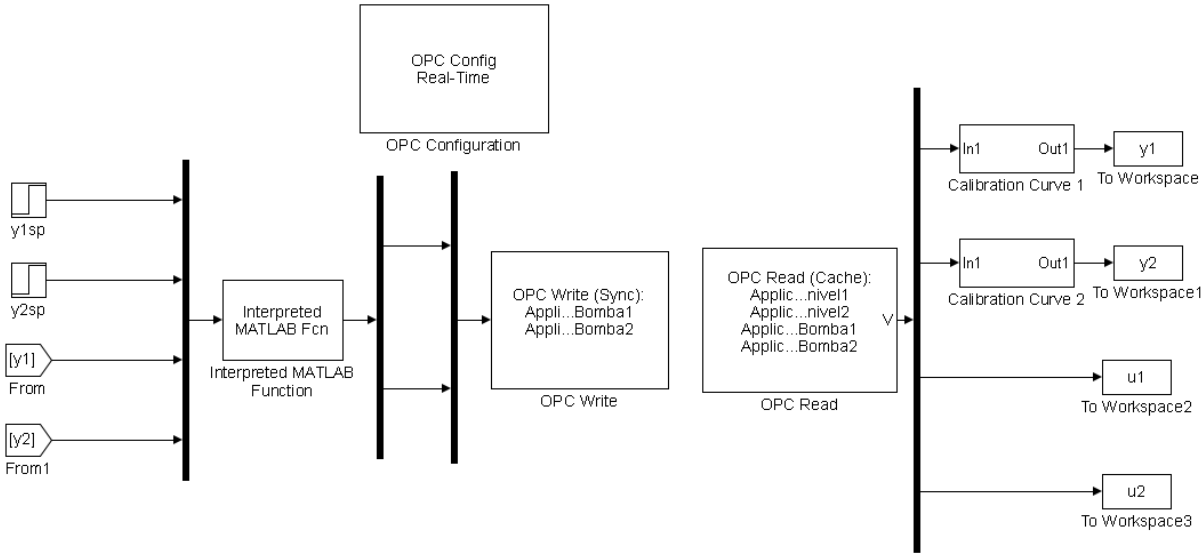

Figure 27: Simulink block diagram of the inverse neural network control.

4.5. *MPC-ANN* Modelling and Simulation

The model predictive control based on artificial neural network was also implemented in simulation. The non-linear dynamic differential equations used were based on mass balance as shown in the equation below, in which Q_i represent the flow that gets into the tanks through the pumps (P-101 and P-102) and the hand-valve HV-103; and Q_o is the flow leaving the tanks through the valves HV-101, HV-102 and HV-103. The subscript *j* represents the tank V-101 (*j* = 1) or V-102 (*j* = 2). Besides, A_j is the tank cross sectional area and ρ is the water density.

$$\rho A_j \frac{dh_j}{dt} = \rho Q_i - \rho Q_o \tag{25}$$

Equation 26 represents the total input flow rate. The first term represents the relationship between the input flowrate (in cm^3/s) and the power of the pump, u_j (in %), related to the tank (for example, P-101 for tank V-101). The signal of the second term is positive if the level of the tank "j" is smaller than the other tank, and it is negative if the level of tank "j" is larger than the level of the other tank the difference on pressure. Equation 27 represent the relationship between the output flowrate and the level of each vessel.

$$Q_i = F(u_j) \pm C_{v3}\sqrt{|h_1 - h_2|} \tag{26}$$

$$Q_o = C_{vj}\sqrt{h_j} \tag{27}$$

In which Cv_j is the valve coefficient of the hand-valve HV-10j, F is the function that relates the power of each pump ("*u*") with flow rate that get in the tank through the pumps.

The final differential equations (Equations 28 and 29) are derived through the combination of Equations 25 with Equations 26 and 27.

$$\frac{dh_1}{dt} = \frac{F_1(u_1)}{A_1} - \frac{C_{v1}\sqrt{h_1}}{A_1} \pm \frac{C_{v3}\sqrt{|h_1 - h_2|}}{A_1} \tag{28}$$

$$\frac{dh_2}{dt} = \frac{F_2(u_2)}{A_2} - \frac{C_{v2}\sqrt{h_2}}{A_2} \pm \frac{C_{v3}\sqrt{|h_1 - h_2|}}{A_2} \tag{29}$$

The input flow rate was obtained keeping all hand valves closed and measuring the time to complete the volume of the tank through Equation 30. The power of each pump was varied from 0% to 100% and a calibration curve was fitted to relate the u_j with $F(u_j)$.

$$F(u_j) = \frac{L_c A_j}{T} \tag{30}$$

In which L_c is the maximum height of the tanks, A_j is the cross-sectional areas of the tanks and T is the time to complete the vessel.

The physical parameters of the coupled tanks apparatus are shown in Table 3. The cross-sectional areas (A_1 and A_2) were calculated through the multiplication of length and width,

while the valves coefficients (C_{v1}, C_{v2}, C_{v3}) were calculated through simulation, according to the following methodology. First, the coefficient C_{v3} was kept at a value of zero to simulate the closed valve HV-103. Then, the power of pump P-101 and P-102 was kept at 100%, and the C_{v1} and C_{v2} were fitted to keep the level between 90% and 100%. After that, the C_{v3} was fitted aiming to keep the level 1 between 60% and 70% when the power of pump P-101 was 100% and pump P-102 was 0%. The Matlab® code used to simulate the process is shown in appendix A5.

Table 3: Physical parameters of the coupled tanks system.

Parameter	Value
A_1	80 cm²
A_2	80 cm²
C_{v1}	13.5 $cm^{5/2}/s$
C_{v2}	13.5 $cm^{5/2}/s$
C_{v3}	2.7 $cm^{5/2}/s$
L_c	20 cm

After discovering the valves coefficients (C_{v1}, C_{v2} and C_{v3}), the identification procedure was performed in a way similar to the procedure in the experiment shown in section 3.3.2, which means that some steps were applied in the power of the pump and the response in the level was obtained through simulation of the Equations 29 and 30. The sample time was fixed at 3 seconds, total time of the simulation was 7200 seconds, and the time between the step tests was 300 seconds. The algorithm code of the identification procedure is shown in appendix A6.

The last step was the tuning of the controller and application of the *MPC-ANN*. The sample time in this step must be the same defined in the identification, and the maximum predictive horizon is the time between the steps and the sample time, which in this work was $N_p = 100$.

5. Results and Discussion

5.1. *PID* Identification and Tuning

As explained in section 3.2.1, a first-order-plus-dead-time model (Equation 22) was used to identify the system. The hand valves (HV-101, HV-102 and HV-103) were kept partially opened and the power of each pump changed from 30% to 50%.

Transfer function G11(s) was obtained from the response of level 1 when the power of pump P-101 changed. Transfer function G12(s) was obtained from the response of level 1 when the power of pump P-102 was varied. Transfer function G21(s) was obtained from the response of level 2 when the power of pump P-101 changed. Transfer function G22(s) was obtained from the response of level 2 when the power of pump P-102 was varied.

The resulting transfer functions are shown in Table 4, while the comparison between the data points provided by the sensor (y1 and y2) and the transfer function model fitted ($y1_{FOPDT}$ and $y2_{FOPDT}$) is shown in Figure 28, Figure 29, Figure 30 and Figure 31.

Table 4: Transfer functions obtained by the identification method.

Transfer function	*Kp* (%/%)	τ_p (s)	t_d (s)
G11(s)	1.93	32.16	4.84
G12(s)	0.89	37.34	5.66
G21(s)	0.78	44.30	6.70
G22(s)	1.72	32.20	4.80

These parameters show that the total time (300 seconds) used in the identification experiments is adequate since level 1 and level 2 can reach the new steady state. Besides, the interactions between the two variables are evident. The Relative Gain Analysis, explained in the Introduction Section (item 3.5) and shown in Equation 26, not only shows that the power of Pump P-101 should control the level 1 and the power of Pump P-102 should control level 2, but it also shows that the interactions between the variables can pose challenges to the control problem because there are relative gains smaller than 0 (λ_{12} and λ_{21}) and larger than 1 (λ_{11} and λ_{22}).

$$\Lambda = \begin{bmatrix} 1.26 & -0.26 \\ -0.26 & 1.26 \end{bmatrix} \tag{31}$$

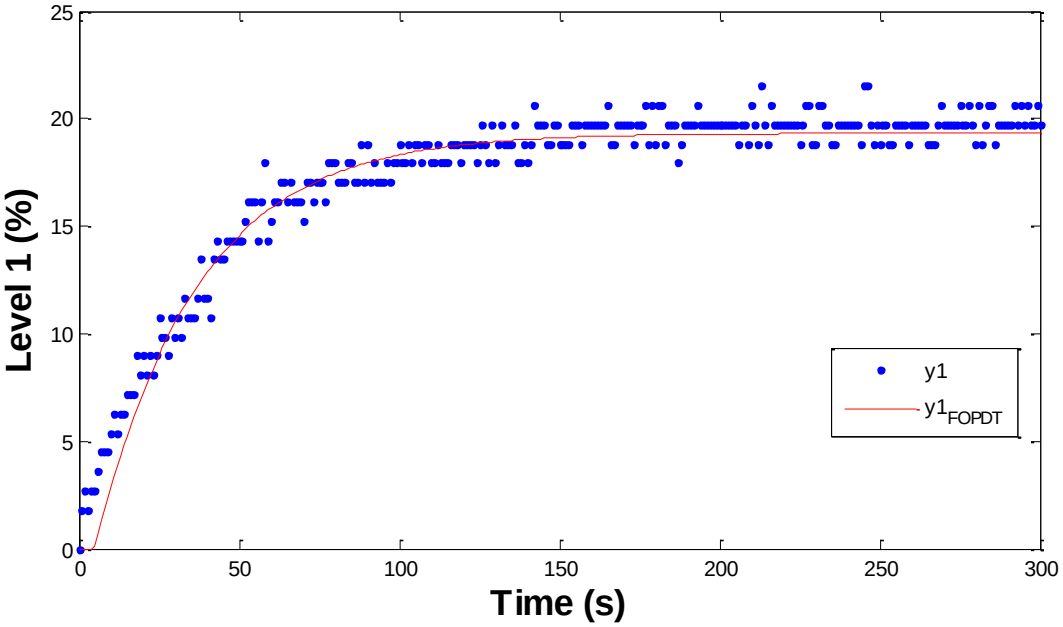

Figure 28: Identification of the process transfer function $G_{p11}(s)$.

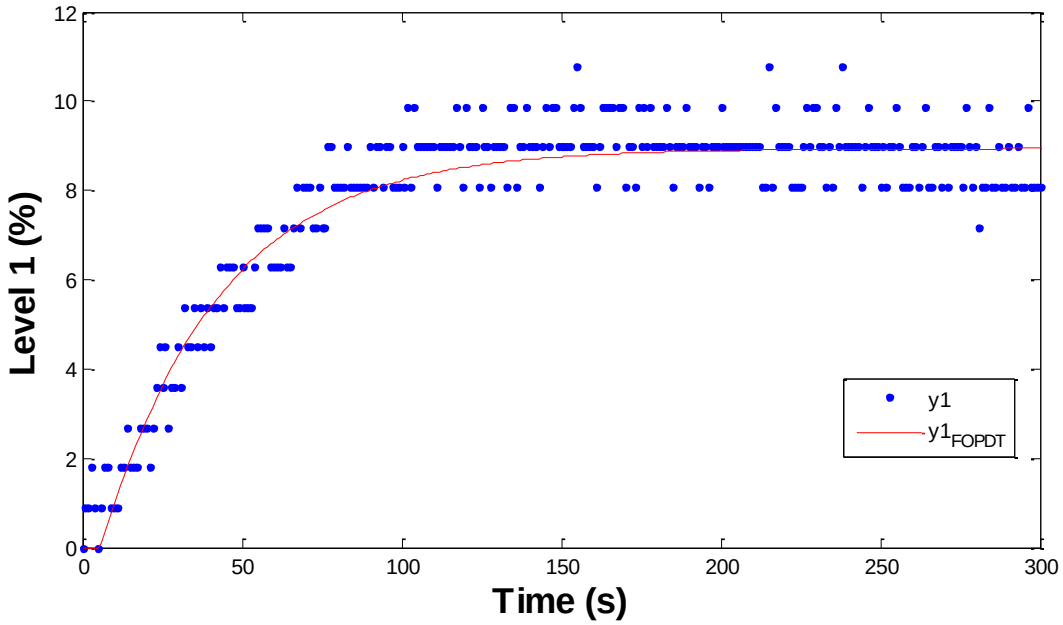

Figure 29: Identification of the process transfer function $G_{p12}(s)$.

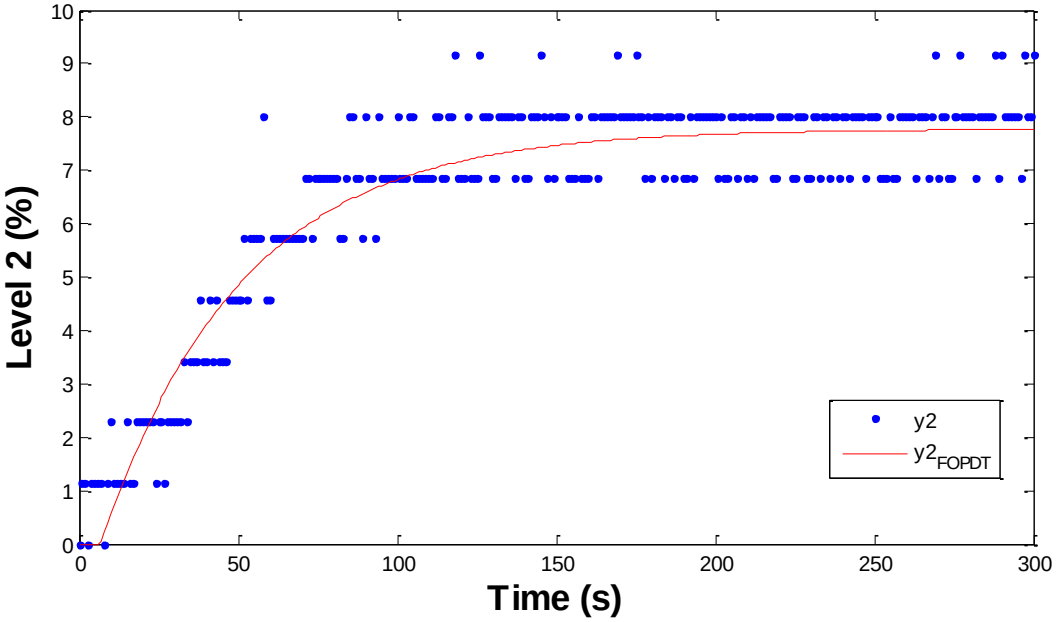

Figure 30: Identification of the process transfer function $G_{p21}(s)$.

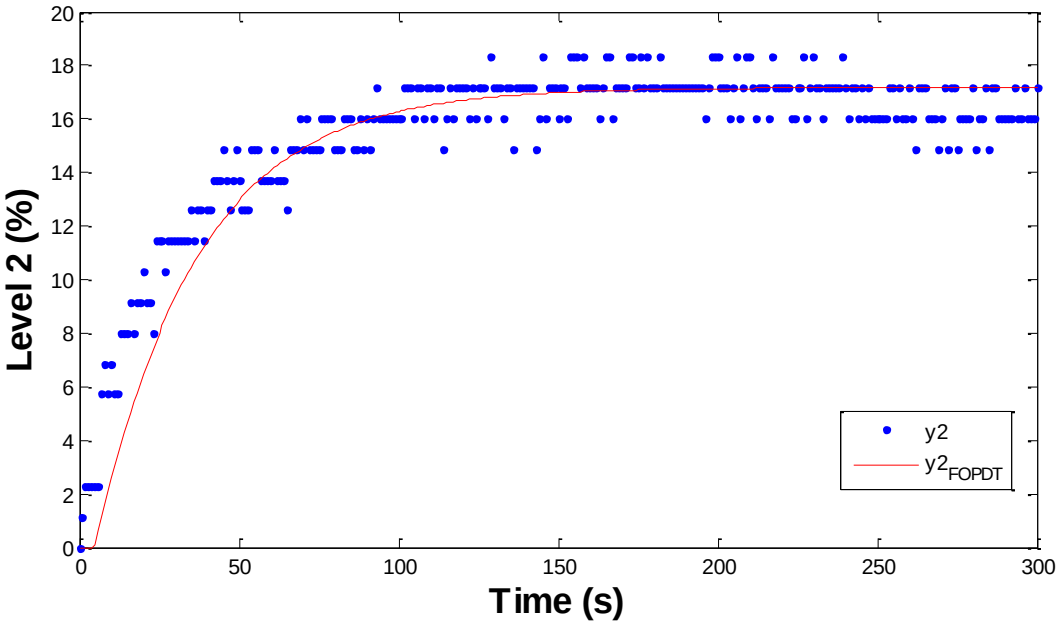

Figure 31: Identification of the process transfer function $G_{p22}(s)$.

Aiming to find the *PID* tuning parameters that result a fast response with a small overshoot and a small error, the well-known correlations of Ziegler Nichols and Ultimate gain, presented in Table 2 in the Introduction Section, were used. As shown in Table 5 and 6, both methods resulted in great overshoot (more than 50%). Therefore, a fine tuning was performed in order to reduce the overshoot. The proportional and integral action was decreased, which resulted in the highlighted tune parameters for the first controller shown in Table 5, K_{c1} = 2.50

%/%, τ_{i1} = 12.00 s, τ_{d1} = 2.29 s; and for the second controller shown in Table 6, K_{c2} = 3.00 %/%, τ_{i2} = 12.00 s, τ_{d2} = 2.28 s. Figure 32 and Figure 33 show the result of both controllers when a step test of 10% was applied in the setpoint of level 1 and level 2, respectively. Figure 34 and 34 show the control action response (power 1 and power 2) of each tuning method.

Table 5: Tuning parameters of the *PID* 1.

	K_{c1} (%/%)	τ_{i1} (s)	τ_{d1} (s)	IAE_1	$ITAE_1$	ISE_1	Overshoot$_1$
Ziegler-Nichols	4.13	9.68	2.42	311	20130	1147	0.67
Ultimate Gain	3.44	9.15	2.29	135	1701	826	0.55
Fine tuning	**2.50**	**12.00**	**2.29**	**134**	**1953**	**773**	**0.22**

Table 6: Tuning parameters of the *PID* 2.

	K_{c2} (%/%)	τ_{i2} (s)	τ_{d2} (s)	IAE_2	$ITAE_2$	ISE_2	Overshoot$_2$
Ziegler-Nichols	4.68	9.60	2.40	306	19660	1125	0.67
Ultimate Gain	3.90	9.10	2.28	134	1675	819	0.55
Fine tuning	**3.00**	**12.00**	**2.28**	**130**	**1804**	**757**	**0.24**

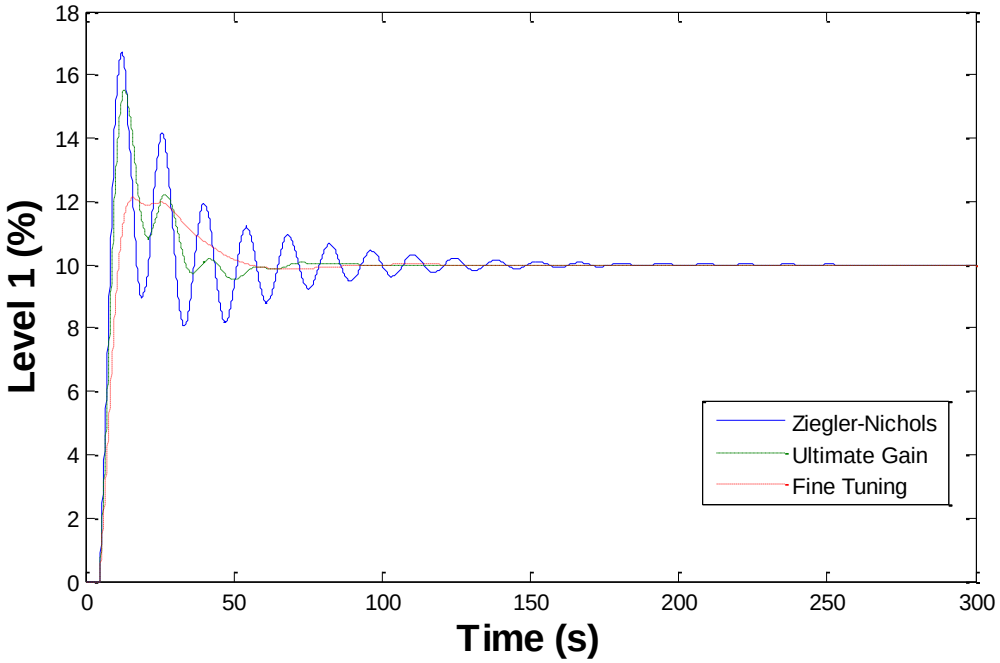

Figure 32: Comparison of the closed loop response of level 1 using different tuning methods.

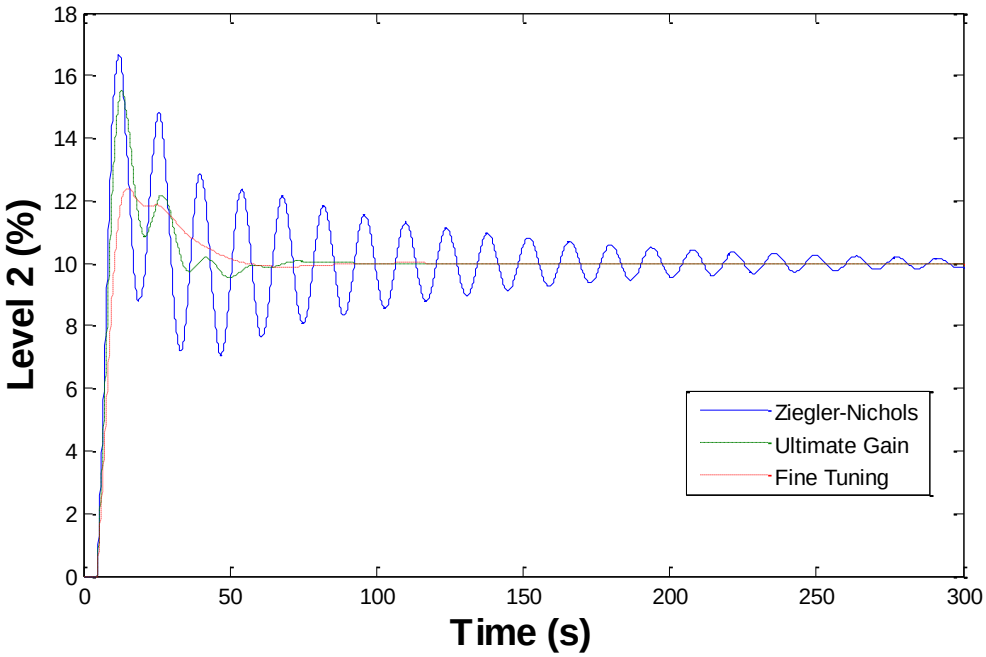

Figure 33: Comparison of the closed loop response of level 2 using different tuning methods.

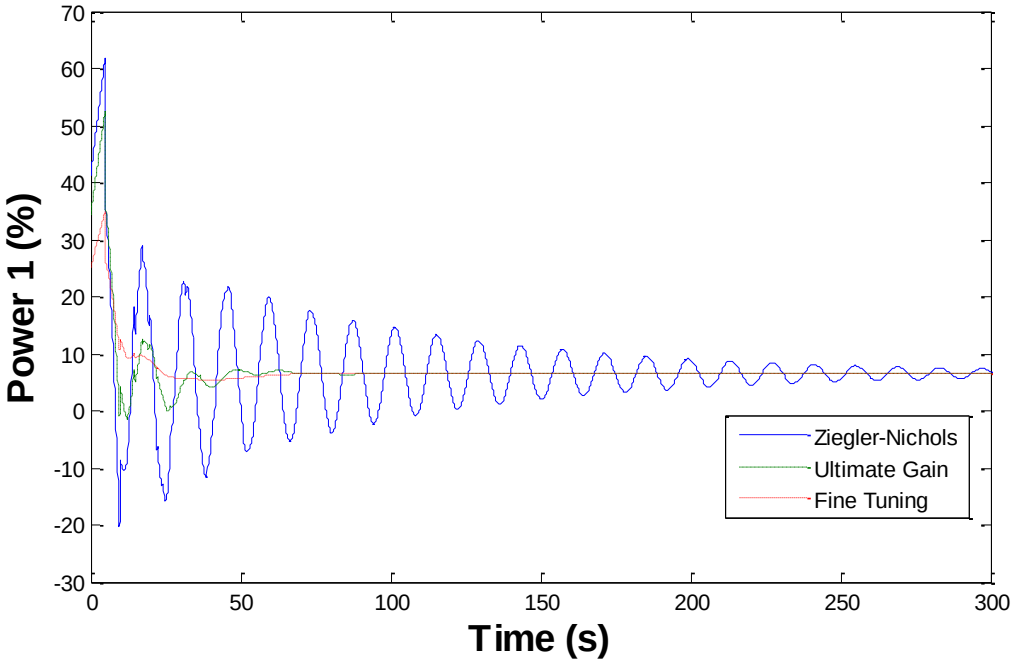

Figure 34: Comparison of the control action (Power 1) using different tuning methods.

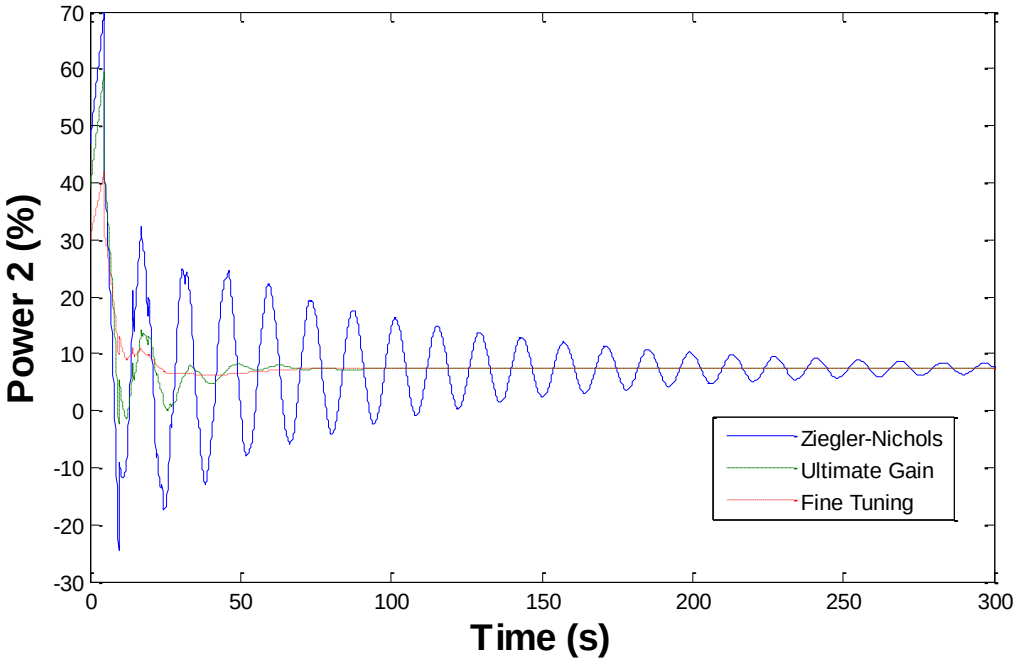

Figure 35: Comparison of the control action (Power 2) using different tuning methods.

5.2. *PID* decoupling

In order to eliminate the effect of control loop interactions, decouplers shown in Equation 17 and 18 and in Figure 11 were added to the *PID* strategy. The multivariable *PID* (with the decouplers) were then compared with the multiloop *PID* (without the decouplers). The results are shown in Figure 36 and Figure 37.

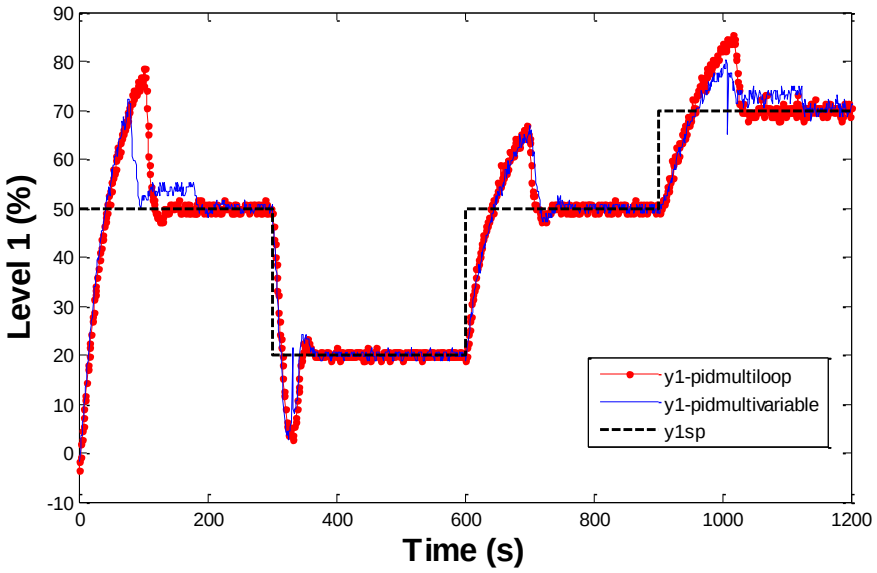

Figure 36: Comparison between multivariable *PID* strategy with multiloop *PID* strategy for level 1.

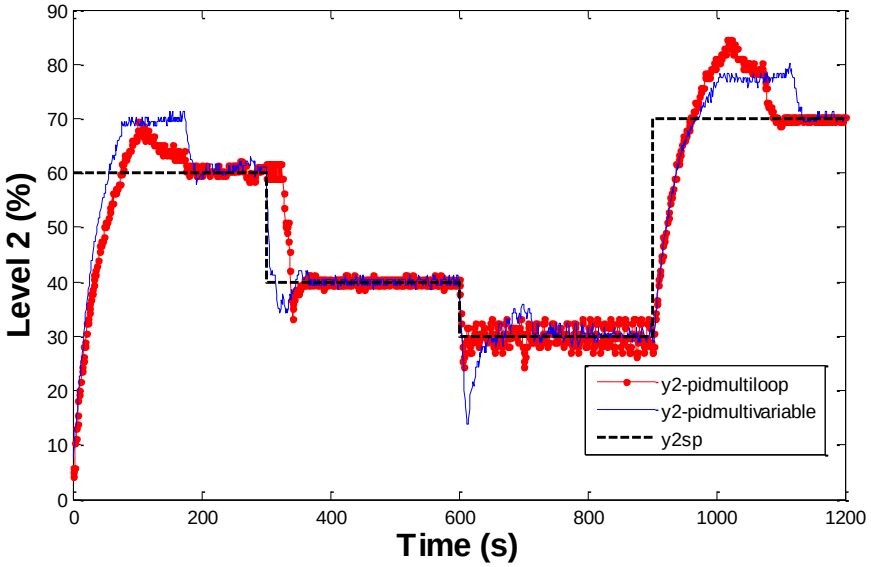

Figure 37: Comparison between multivariable *PID* strategy with multiloop *PID* strategy for level 2.

According to Seborg, Edgar and Mellichamp (2003), the decoupling control can predict and eliminate control loop interactions. However, this is possible only when the model of transfer functions can predict accurately the interactions between control loops. As it can be shown in Figure 36 and Figure 37, the gain in the performance when decoupling is performed is negligible. This can be explained by the fact that the process is nonlinear; and there are mismatches between the transfer function model and the real process, so the decoupling cannot anticipate the effect of interactions between manipulated and process variables.

5.3. *PID* Stability and Linearity Analysis

The stability analysis was performed using the Root-locus method (function *rlocus* of Matlab®), for the functions identified of the process ($G_{p11}(s)$ and $G_{p22}(s)$). Figure 38 and Figure 39 shows the result of the stability analysis. It is possible to see that there are three poles and three zeros since the second order Padé approximation was applied as shown in Equation 32. From Figure 38, the ultimate gain of controller 1 is K_c = 6.51 %/% and from Figure 39 the ultimate gain of controller 2 is K_c = 5.74 %/%. These parameters were used to determine the tuning from the Ultimate Gain method.

$$G_{OL} = 1 + G_c G_{proc} = 1 + K_{cr} \frac{Kp}{\tau_p s+1} \left(\frac{t_d^2 s^2 - 6 t_d s + 12}{t_d^2 s^2 + 6 t_d s + 12} \right) \quad (32)$$

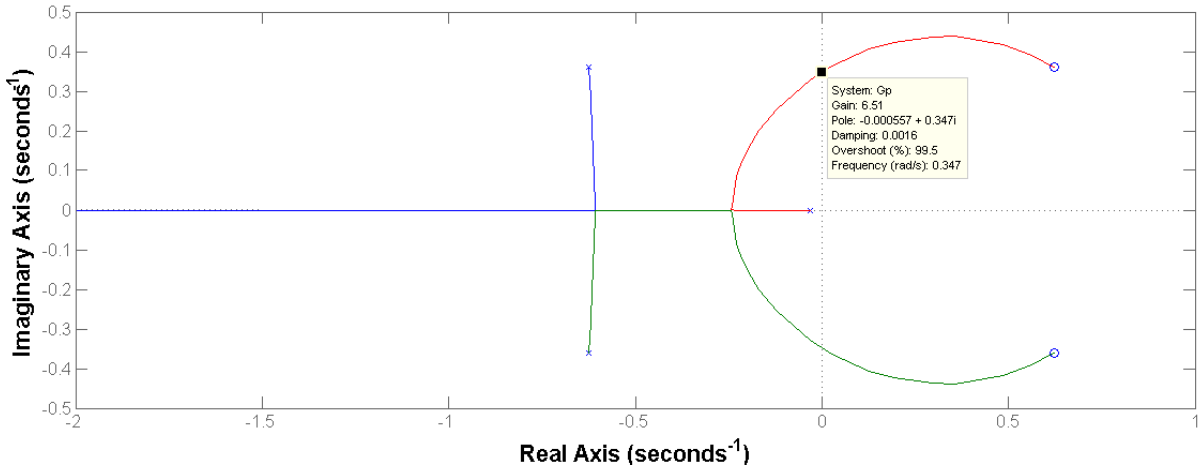

Figure 38: Ultimate gain for the control loop 1 (level 1 and power of the pump P-101).

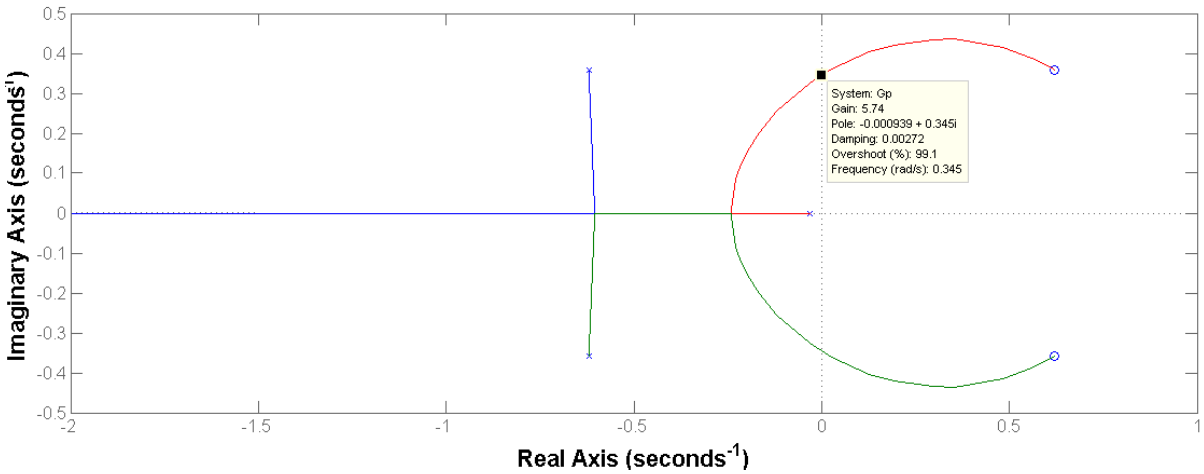

Figure 39: Critic gain for the control loop 2 (level 2 and power of the pump P-102).

In a linear system, the gain in the process variable is proportional to the variation in the manipulated variable, hence K_p assumes a constant value. Therefore, according to Figure 40 and Figure 41, the system is nonlinear, since for the same variation in the manipulated variable in module the controlled variable responds differently. Table 7 shows the process gain for the positive and negative step in the manipulated variable.

Table 7: Level variation when a positive and negative step are performed in the manipulated variable.

	Positive Step (10%)	Negative Step (10%)
AK_{p1} (%)	20.0 %	-27.8 %
AK_{p2} (%)	10.1 %	-16.0 %

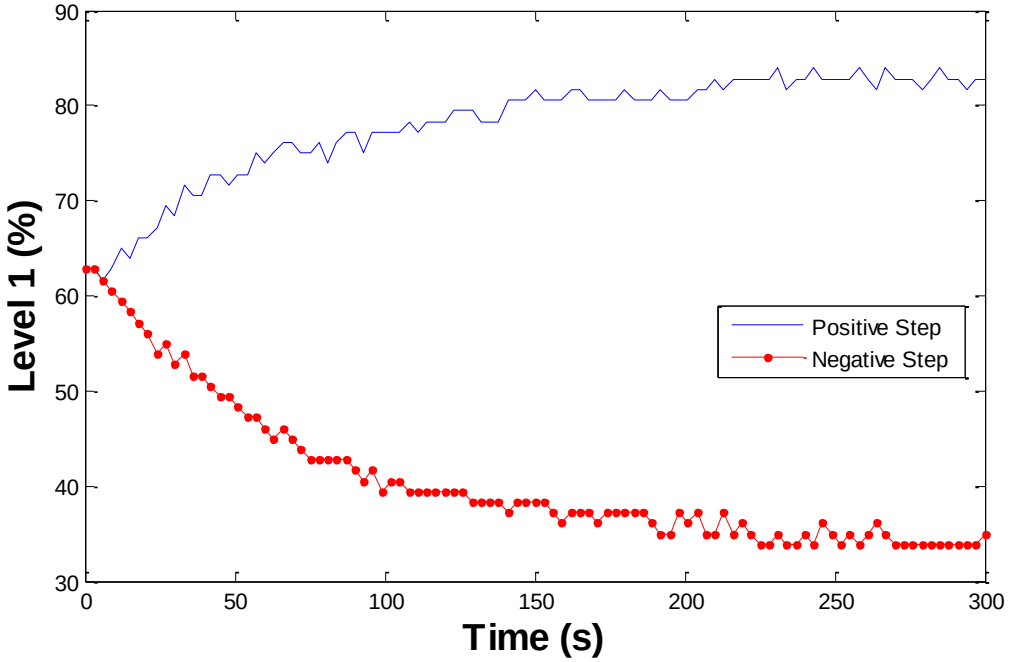

Figure 40: Linearity analysis of level 1.

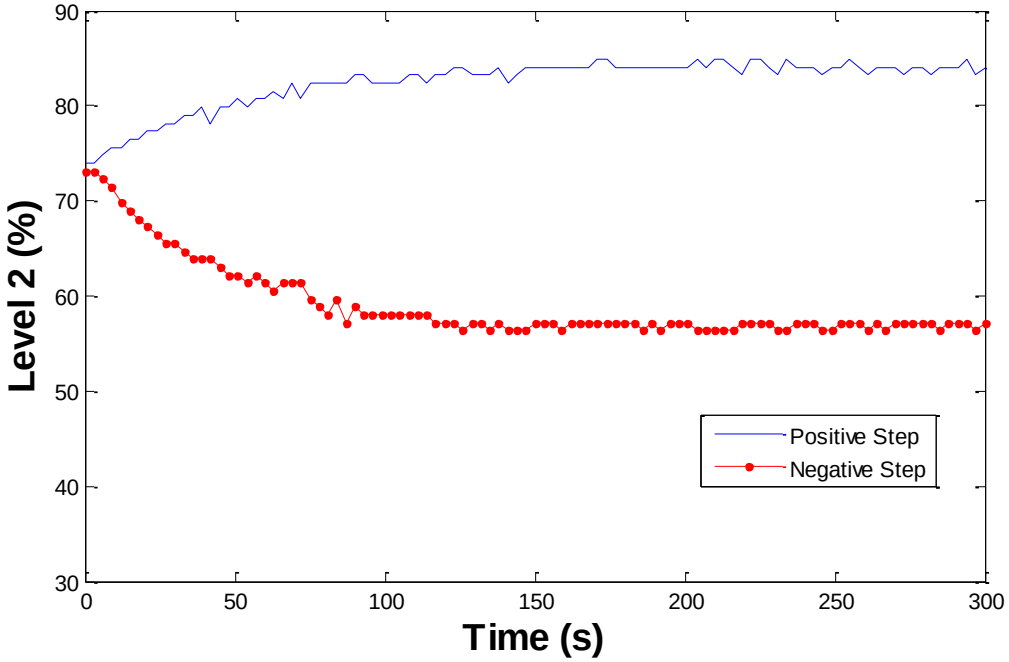

Figure 41: Linearity analysis of level 2.

Therefore, linear designed controllers, such as conventional *PID*, could not work properly when controlling the presented nonlinear experiment. Besides, large overshoots were observed in Figure 36 and Figure 37. According to the theory, the use of nonlinear techniques of control should be tested to improve the performance of the system.

5.4. *MPC-ANN* Identification

The first phase of the identification was the build-up of the dataset in order to obtain the dynamic behavior of the level of tank 1 and tank 2. The power of each pump was randomly varied through values from 20% to 80% so that the neural network could capture the entire process range. The maximum and minimum values of each controlled and manipulated variable are shown in Table 8.

Table 8: Maximum and minimum value of each variable.

Variable	Maximum Value	Minimum Value
Level 1 (%)	110	3
Level 2 (%)	100	3
Power 1 (%)	78	22
Power 2 (%)	80	24

Figure 42 and Figure 43 shows the behavior of level 1 and level 2 respectively and the division between the training set and the test set of the neural network. The first 70% of the data points were used to train the network, while the last 30% was used to test the generalization ability of the network. It is possible to imply from the figures that the test and training set cover the entire range of levels possible for the opening used in the hand valves HV-101, HV-102 and HV-103. Figure 44 and Figure 45 show the range of power of the pump imposed in the experiment and the division between the training and test sets.

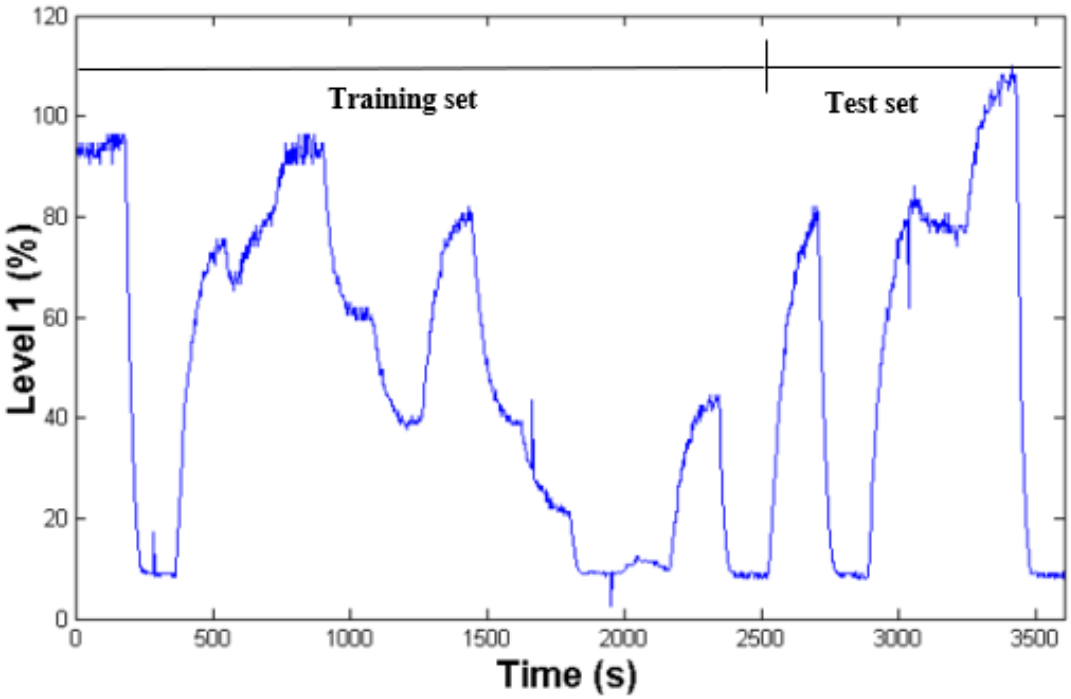

Figure 42: Dynamic Behavior of level 1 and the division between training and test set.

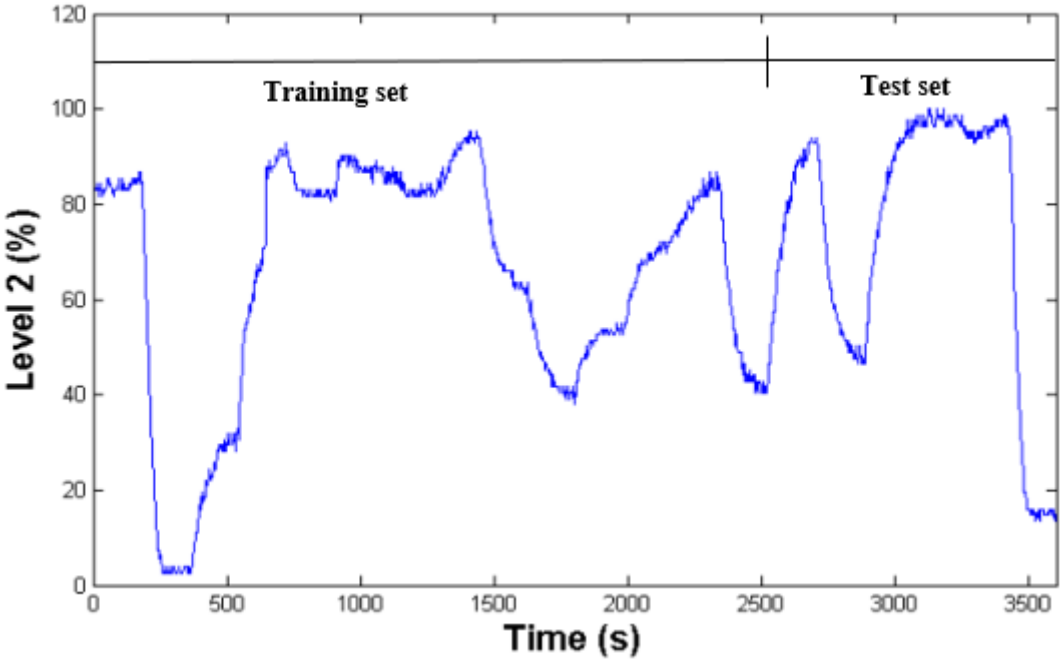

Figure 43: Dynamic Behavior of level 2 and the division between training and test set.

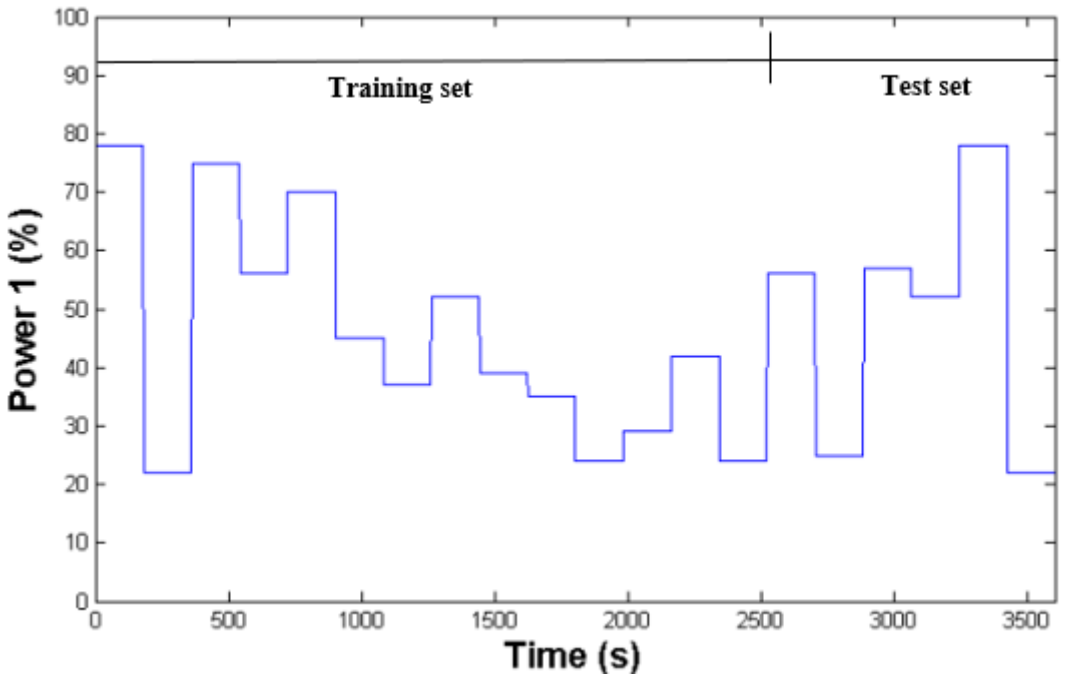

Figure 44: Dynamic Behavior of power of pump P-101 and the division between training and test set.

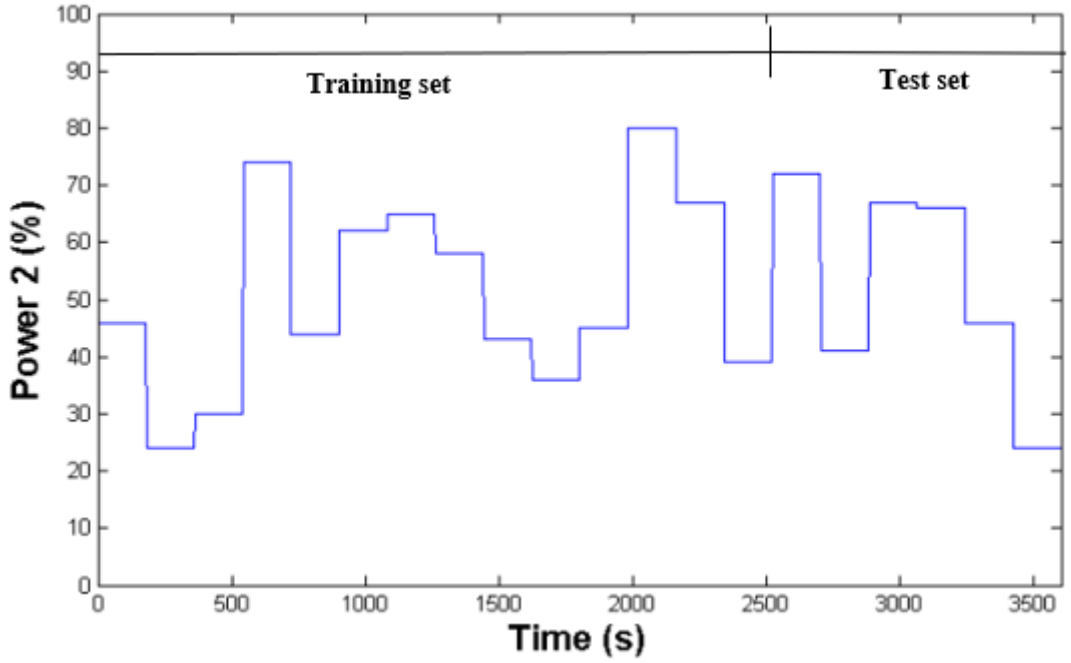

Figure 45: Dynamic Behavior of power of pump P-102 and the division between training and test set.

The second phase of identification is the choice of the best architecture of the neural network. Some architecture parameters were varied such as the number of hidden layers the number of neurons in each layer, and the activation function. Besides, the sample time was varied over 3 seconds, 5 seconds and 7 seconds. These values were based in the correlation of Astrom and Wittenmark (1997), which says that the value of the sample time should be between 1% and 5% of the time constant of the process.

It is important to point out that the choice of the sample time must balance the benefits of the identification and control design. Large sample time may lead to poor control performance, but the sample time cannot be so small due to the computational time required to process the algorithm. Moreover, small sample times can lead to numerical problems (MELEIRO, 2002). Table 9 shows that the sample time that gave the smallest prediction error was the 3 seconds. Since the computational processing time to run the *MPC-ANN* controller algorithm was about 0.4 seconds (the processor was Intel® Core™ i5-3300 CPU @ 3.00 GHz), there is no problem in using 3 seconds as the sample time of the controller.

The definition of the number of delayed inputs used in the neural network is also an important step. In the methodology section, we show that it is possible to propose a neural network that predicts all N_p outputs in one single step (Figure 22) or to predict only one output and feedback the information so that the neural network can predict the N_p future outputs (Figure 23). The second structure is more flexible and gives a smaller error, so only the second structure was used.

According to Willys (1992), a neural network with a small number of delayed inputs may not capture the dynamics of the process due to delays intrinsic of the process. However, many delayed inputs may lead to unnecessary complex neural network structure. In this work, three structures were tested: with 4 inputs (0 delayed inputs), 8 inputs (1 delayed input) and with 12 inputs (2 delayed inputs). According to Table 9, the best neural network structure was the one with only 8 inputs, which can be explained because of the small dead time of the process.

Table 9 shows the comparison between different *ANN* structural parameters. The number of neurons and the activation function presented for every sample time and *ANN* setting are the ones that minimized the error criteria. Figure 46 and Figure 47 show how the number of neurons and the activation function of Table 9 were chosen. It is possible to conclude that there is no need to use a complex network for this problem since simple structures gave a better performance.

Table 9: Effect of the neural network parameters in the performance of the network.

Sample time (seconds)	Output variable	Number of inputs	Activation function	Number of neurons in the hidden layer	MSE_{test} (x10^{-4})
3		4	logsig	2	6.00
	Y1	**8**	**logsig**	**2**	**5.71**
		12	logsig	2	6.16
		4	logsig	5	6.65
	Y2	**8**	**logsig**	**4**	**5.49**
		12	tansig	4	6.23
5		4	tansig	3	9.40
	Y1	8	logsig	3	8.09
		12	tansig	3	8.31
		4	tansig	2	11.05
	Y2	8	tansig	2	9.26
		12	tansig	2	9.30
7		4	tansig	1	13.59
	Y1	8	logsig	1	17.49
		12	logsig	1	14.32
		4	logsig	2	16.32
	Y2	8	tansig	3	6.89
		12	logsig	2	7.43

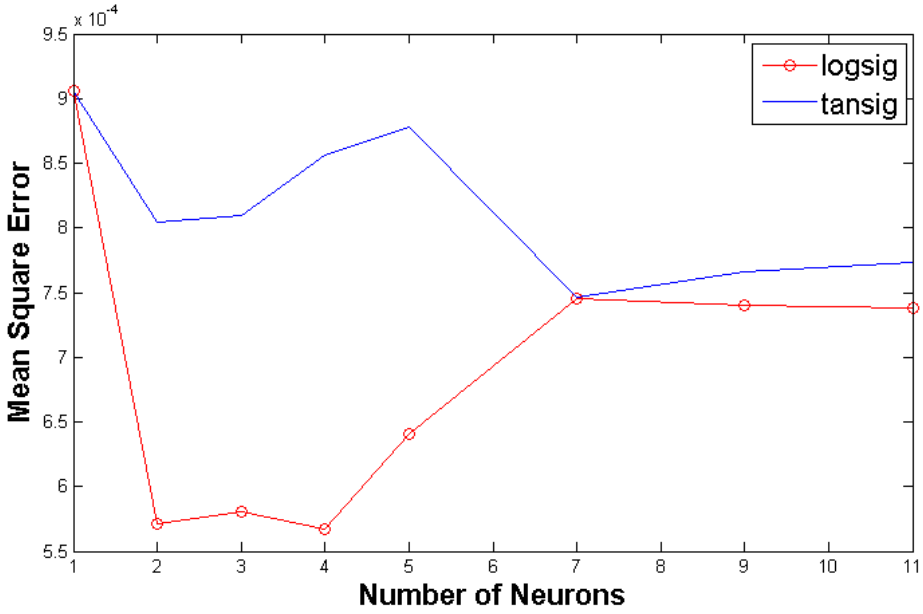

Figure 46: Definition of the number of neurons in the first neural network (prediction of level 1).

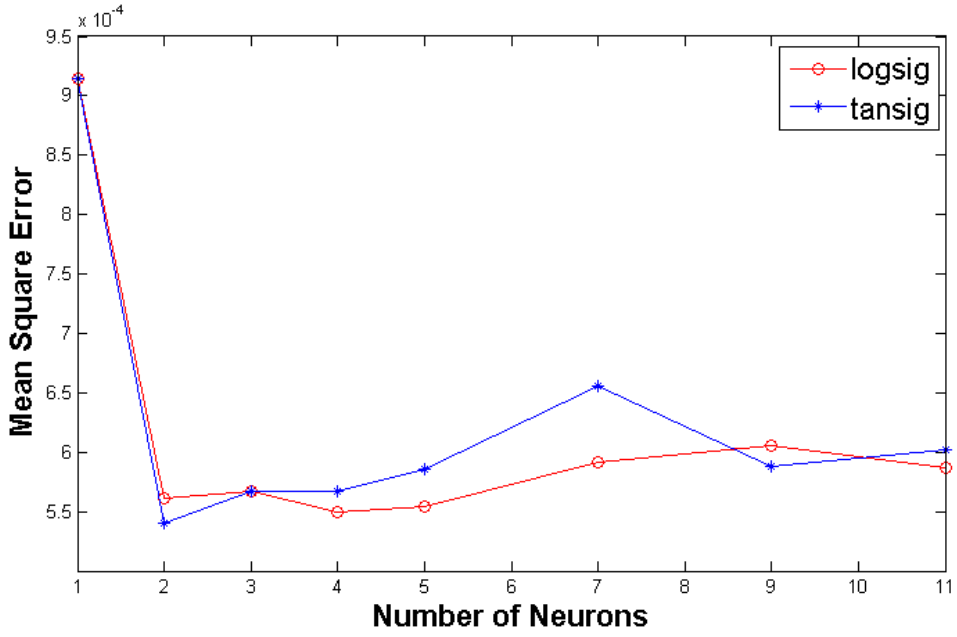

Figure 47: Definition of the number of neurons in the second neural network (prediction of level 2).

A comparison between the predicted and actual values for the best neural network architecture is shown in Figure 48 and Figure 49. The figures show the suitable prediction of level 1 and level 2, hence the neural network model can be used in the model predictive controller.

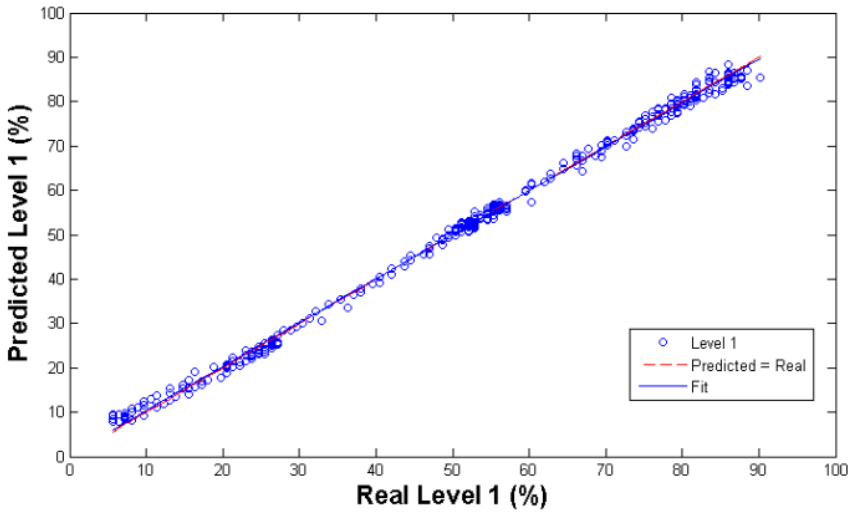

Figure 48: Accuracy of the prediction of Level 1.

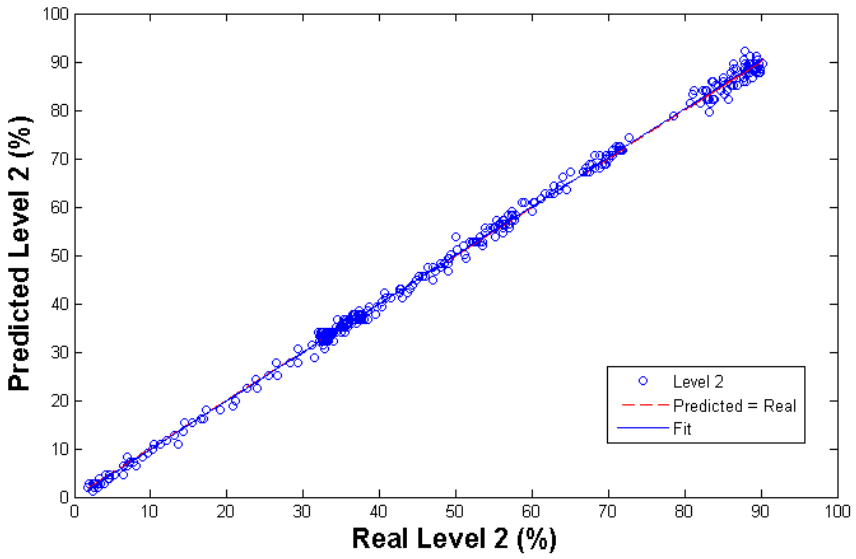

Figure 49: Accuracy of the prediction of Level 2.

5.5. *MPC-ANN* Tuning

After the identification of the *MPC-ANN* approach, the next step was to find a set of controller parameters like N_p, N_c, w, and w_y that gives a suitable performance. As explained by Seborg, Edgar and Mellichamp (2003), large control horizons can lead to instability and

aggressive responses, so N_c was kept between 1 and 4. Large predictions horizon can improve the performance since the *MPC* algorithm perform an optimization of the future performance of the system in the prediction horizon. Besides small prediction horizons make the response more aggressive.

The ratio between the prediction horizon and the control horizon was defined as four (N_p/Nc = 4), which lies in the range given by Seborg, Edgar and Mellichamp (2003). Four values of prediction horizon were tested: N_p = 4, N_p = 8, N_p = 12 and N_p = 16. The weight of the control action in the objective function was also varied. Three values were tested: w_i = 0.01, w_i = 0.1 and w_i = 1. Besides, the weight of the control error (w_{yi}) was fixed at 1.

The experimental tests were performed as explained in section 3.3.3., in which the set point was varied from 0% to 50% for the level 1 and from 0% to 60% for the level 2. The results of the effect of the tuning parameters of *MPC* in the performance criteria are summarized in Table 10.

Table 10: Effect of MPC tuning parameters on the performance criteria.

Tuning Parameters			Performance criteria			
N_p	*Nc*	*w*	*IAE*	*ITAE*	*ISE*	Overshoot
4	1	0.01	2955	133628	84426	3.9%
8	2	0.01	2811	123160	81224	9.1%
12	3	0.01	3545	268683	80911	12.6%
16	4	0.01	8227	1081411	190932	-
4	1	0.1	3430	203039	93895	11.8%
8	**2**	**0.1**	**2526**	**82928**	**78226**	**4.5%**
12	3	0.1	2965	137552	89298	7.4%
16	4	0.1	2679	132622	75708	17.8%
4	1	1	3872	248632	93601	28.6%
8	2	1	3224	175112	88502	10.9%
12	3	1	3752	262479	92901	10.1%
16	4	1	3059	237185	66665	28.2%

Based on the Table 10, the tuning parameters of *MPC-ANN* chosen were w_i = 0.1, N_p = 8 and N_c = 2. These *MPC* parameters gave the best performance since the overshoot was only 4.5 % and the integral error criteria were smaller than for other values of N_p, N_c and w_i.

Further analysis of the control response for all values of tuning parameters can be made by observing Figure 50, Figure 51 and Figure 52. It is possible to see that large *Nc* values can turn the response more oscillatory and may even lead to instability. Besides, according to Figure 53, Figure 54 and Figure 55 small weights on the control actions lead to oscillatory responses since the objective function does not punish large variations on the control action.

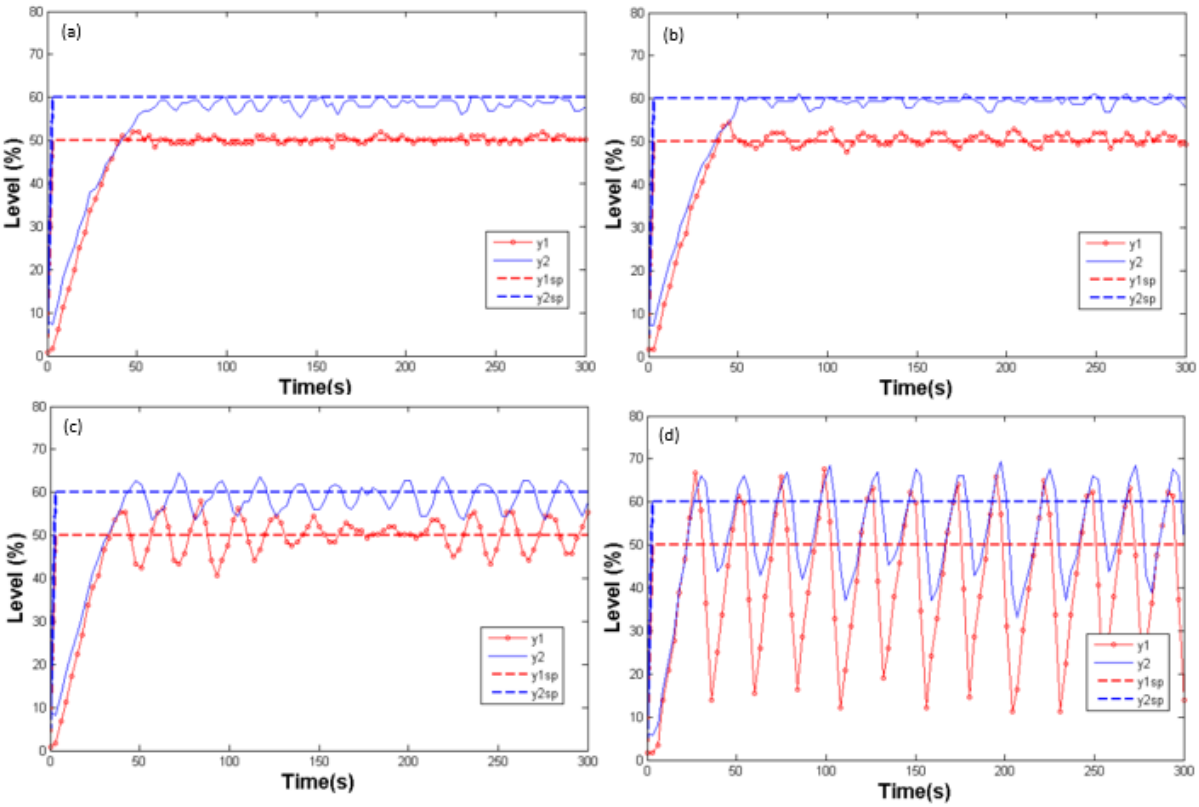

Figure 50: Effect of control and prediction horizon on the control response for $w = 0.01$. (a) $N_p = 4$, $N_c = 1$. (b) $N_p = 8$, $N_c = 2$. (c) $N_p = 12$, $N_c = 3$. (d) $N_p = 16$, $N_c = 4$.

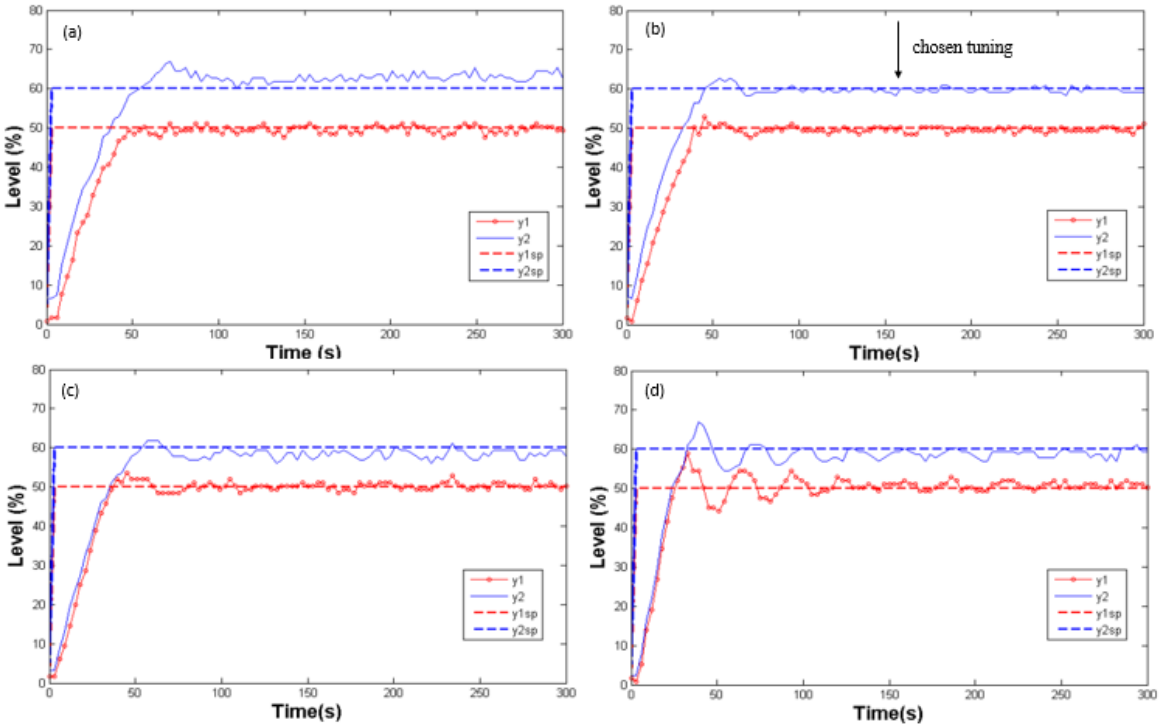

Figure 51: Effect of control and prediction horizon on the control response for w = 0.1. (a) N_p = 4, N_c = 1. (b) N_p = 8, N_c = 2. (c) N_p = 12, N_c = 3. (d) N_p = 16, N_c = 4.

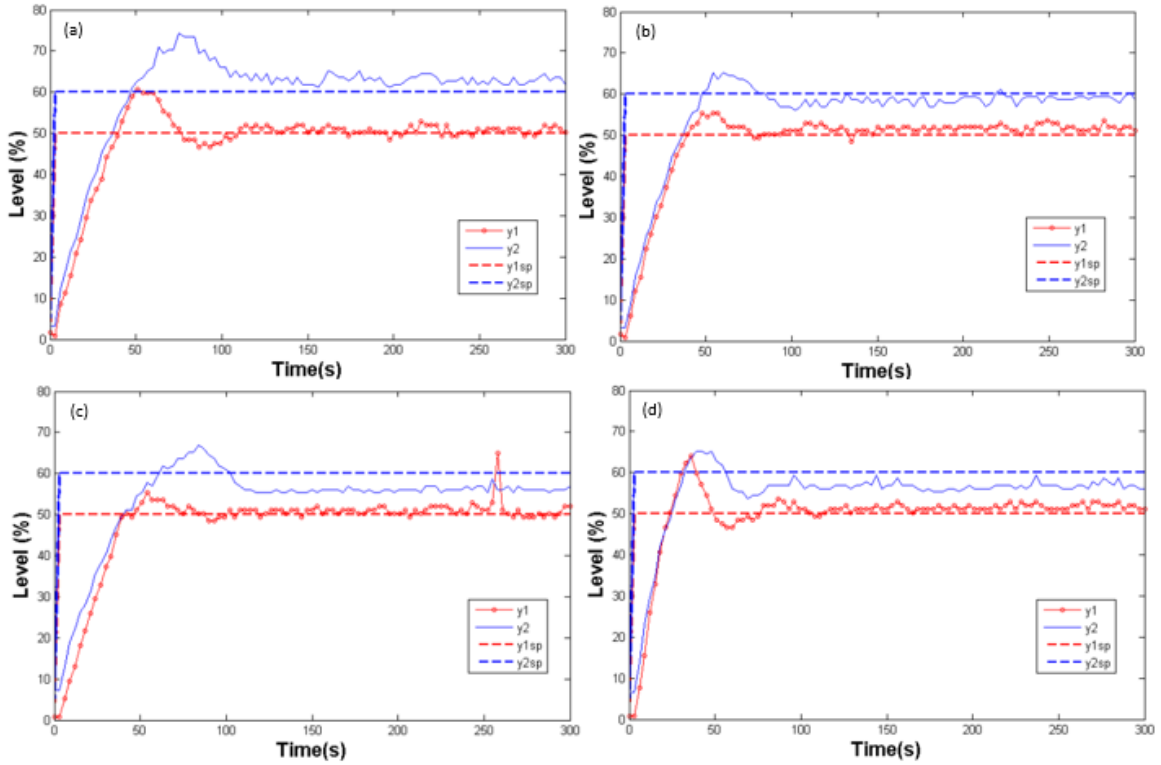

Figure 52: Effect of control and prediction horizon on the control response for w = 1. (a) N_p = 4, N_c = 1. (b) N_p = 8, N_c = 2. (c) N_p = 12, N_c = 3. (d) N_p = 16, N_c = 4.

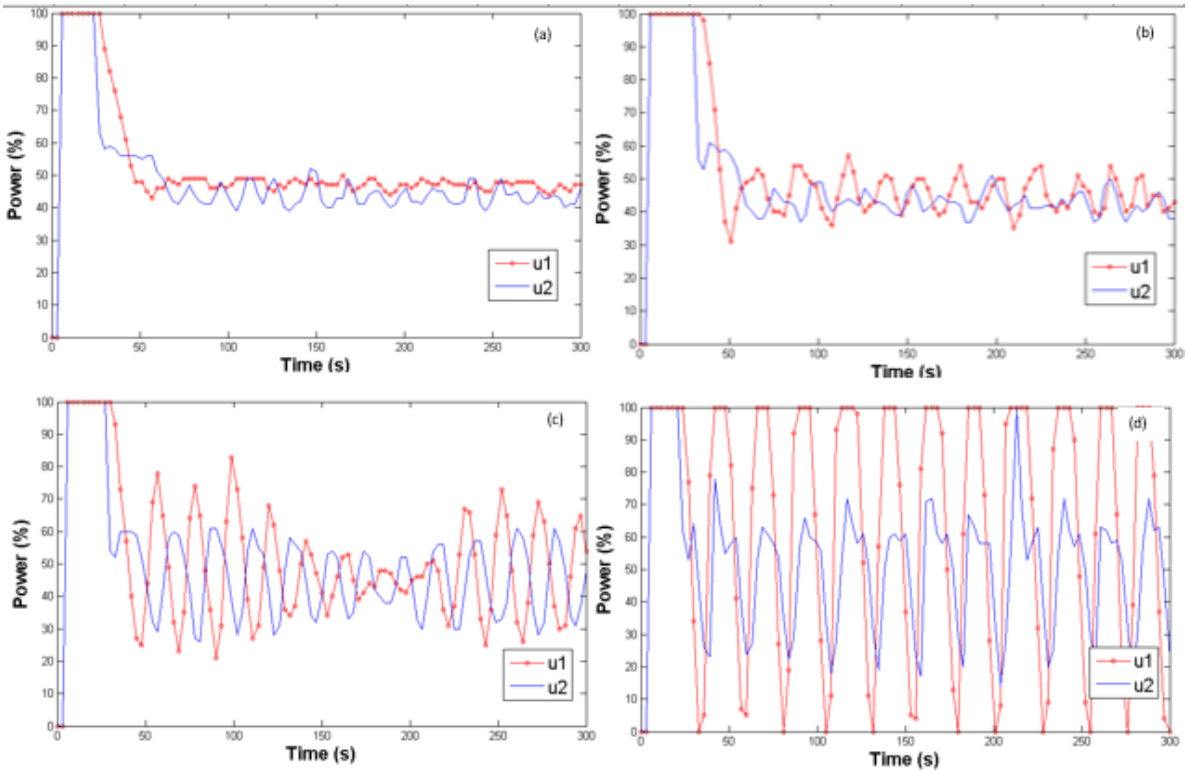

Figure 53: Effect of control and prediction horizon on the manipulated variables for w = 0.01. (a) N_p = 4, N_c = 1. (b) N_p = 8, N_c = 2. (c) N_p = 12, N_c = 3. (d) N_p = 16, N_c = 4.

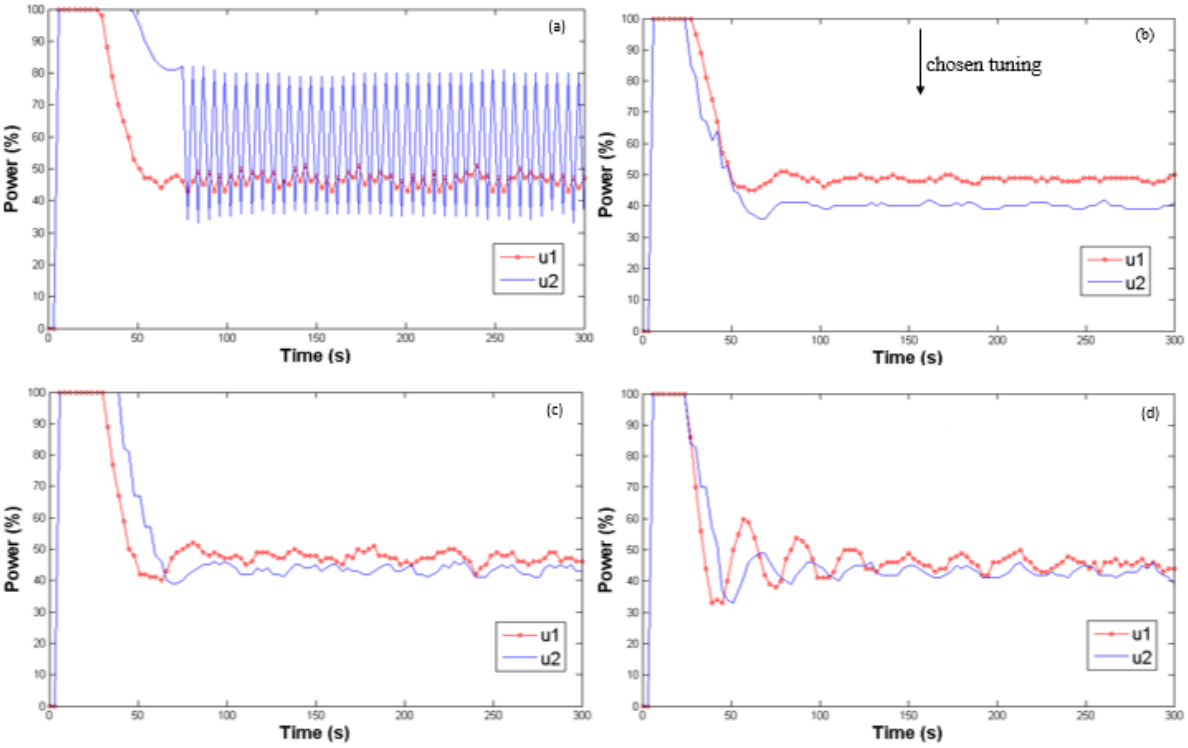

Figure 54: Effect of control and prediction horizon on the manipulated variables for w = 0.1. (a) N_p = 4, N_c = 1. (b) N_p = 8, N_c = 2. (c) N_p = 12, N_c = 3. (d) N_p = 16, N_c = 4.

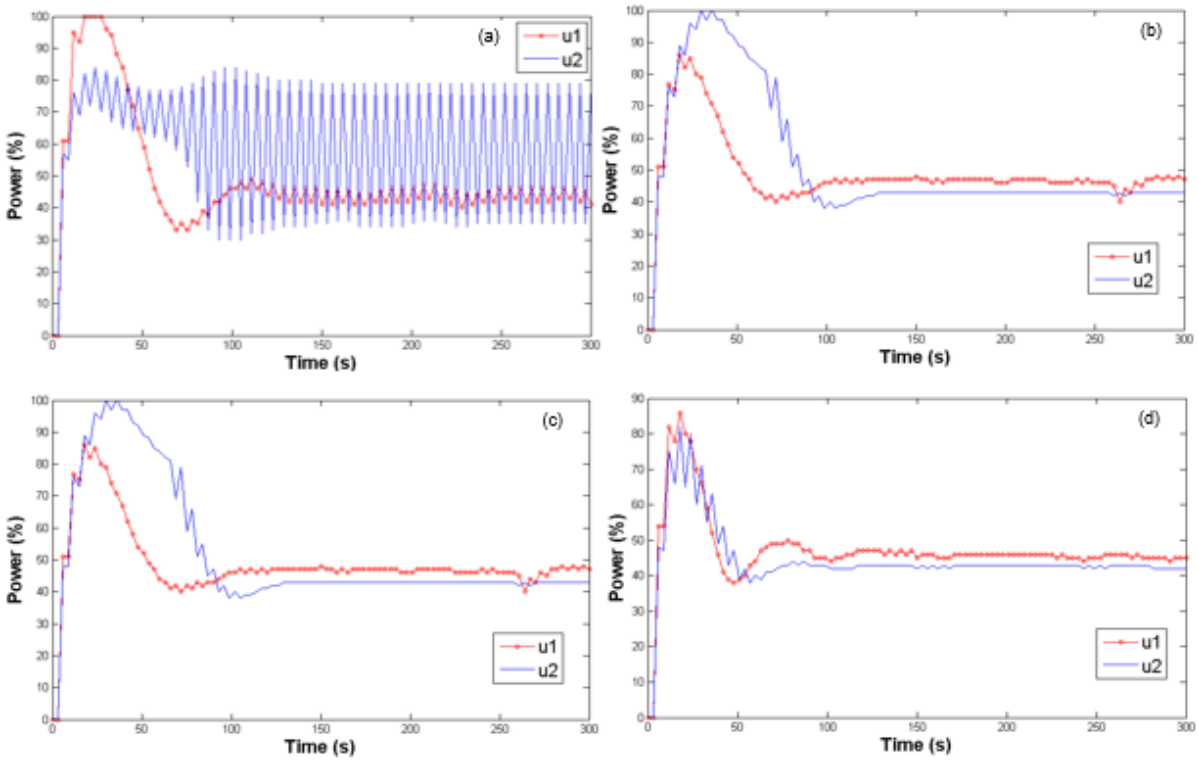

Figure 55: Effect of control and prediction horizon on the manipulated variables for w = 1. (a) N_p = 4, N_c = 1. (b) N_p = 8, N_c = 2. (c) N_p = 12, N_c = 3. (d) N_p = 16, N_c = 4.

5.6. Inverse Neural Network Identification

The identification process for the inverse neural network is also based on two steps. First, the dataset was built collecting data from open-loop perturbations in the power of Pump P-101 and the power of Pump P-102. Then, the neural network of Figure 26 was trained in order to find the best number of neurons of the hidden layer, activation function and the number of hidden layers. Figure 56 shows the behavior of the performance of the network as the number of neurons and the activation function varies. The best neural network had one hidden layer with seven neurons and with the log-sigmoid activation function.

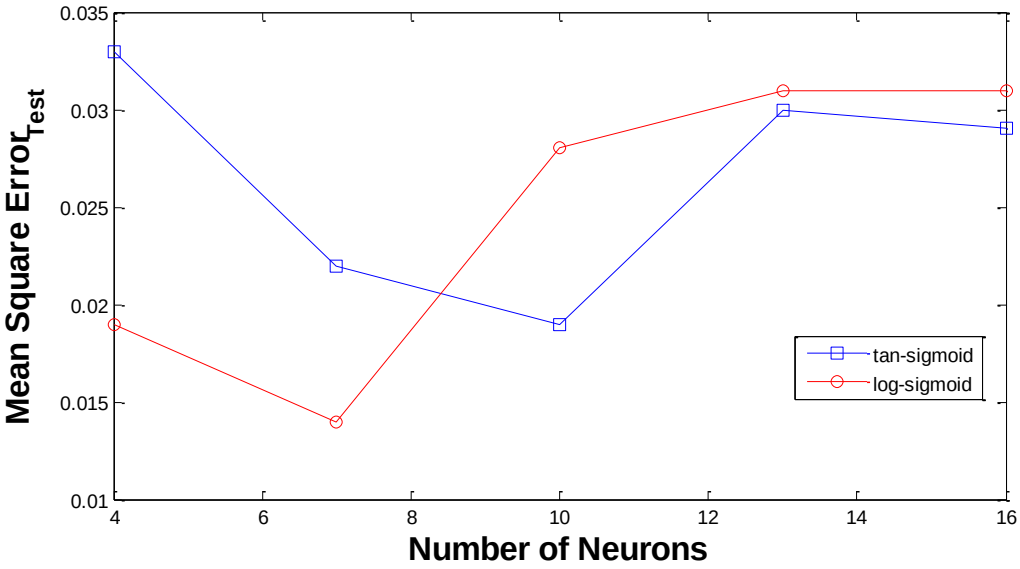

Figure 56: Performance versus number of neurons and activation function.

The *ANN* power predictions are shown in Figure 57 and Figure 58, in which it is possible to see that the prediction of the neural network model was similar to the actual values for almost every point. The angular coefficient, *m1* and *m2*, the linear coefficient, *b1* and *b2*, and the correlation coefficient, r1 and r2, are shown in Table 11. Besides, the dashed line represents the ideal prediction, that is, when the prediction of the inverse neural network is equal to the actual values of the test set. However, according to Figure 57 and Figure 58, there are mismatches between the predicted and actual power, which can cause an offset in the response, since the inverse neural network control strategy is very sensitive to model errors and small variations in the power of the pump can cause an error between the final value of the controlled values and the setpoint.

Table 11: Variation in performance with the change in the number of neurons and the activation function.

	Number of Neurons	*m1*	*b1*	*r1*	*m2*	*b2*	*r2*	MSE_{test}	$MSE_{training}$
	4	0.984	0.526	0.993	0.688	13.325	0.796	0.019	0.002
	7	**0.976**	**1.105**	**0.990**	**0.910**	**3.677**	**0.974**	**0.014**	**0.001**
Logsig	10	0.976	1.144	0.989	0.836	6.816	0.923	0.028	0.001
	13	0.977	1.151	0.989	0.779	9.406	0.890	0.031	0.001
	16	0.977	1.195	0.987	0.779	9.419	0.888	0.031	0.001
	4	0.983	1.266	0.955	0.783	9.170	0.905	0.033	0.001
	7	0.951	1.872	0.985	0.886	4.535	0.937	0.022	0.001
Tansig	10	0.974	1.752	0.966	0.896	4.246	0.960	0.019	0.001
	13	0.989	0.962	0.960	0.879	4.965	0.953	0.030	0.001
	16	0.958	2.302	0.980	0.787	9.072	0.900	0.029	0.001

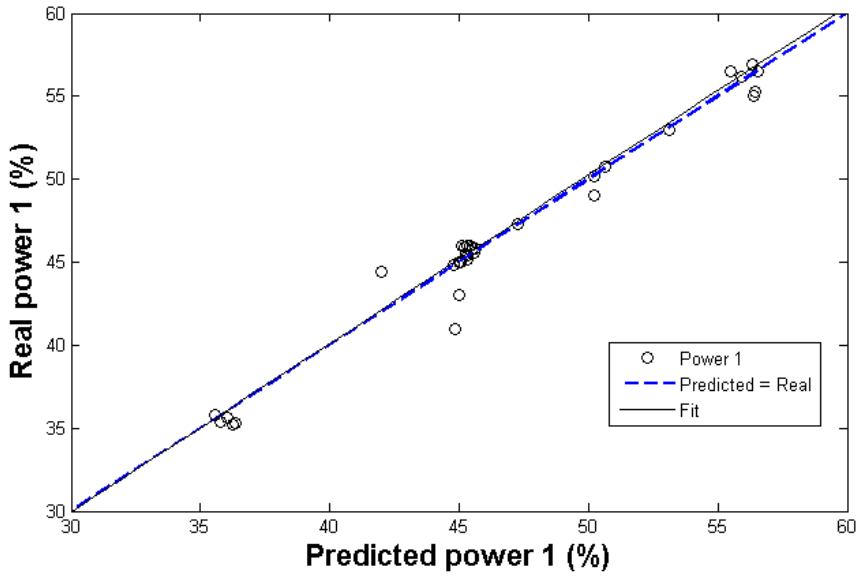

Figure 57: Accuracy of the inverse neural network prediction of power 1.

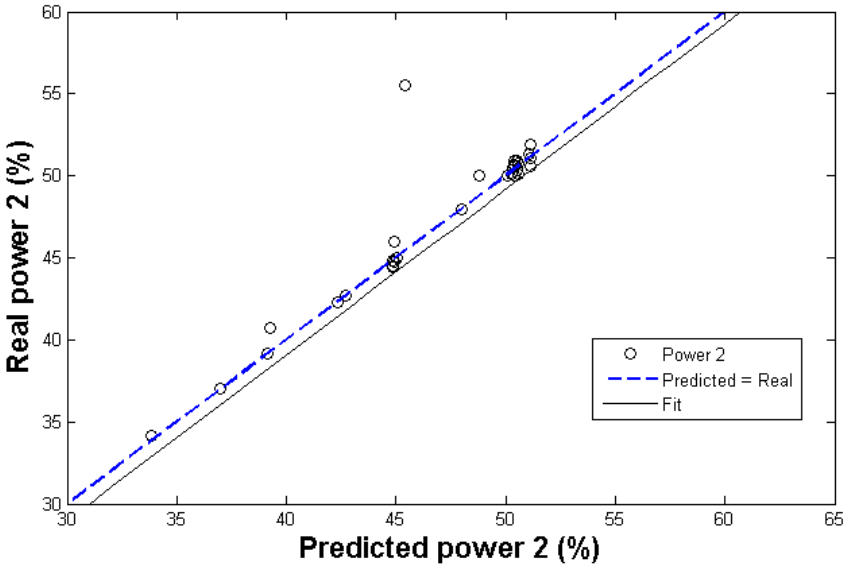

Figure 58: Accuracy of the inverse neural network prediction of power 2.

5.7. Comparison among the Controllers

After the identification and tuning steps, the three control approaches were compared based on the experiment explained in the Methodology Section, in which the setpoint of level 1 and setpoint of level 2 were varied simultaneously (0, 50, 20, 30, 70 for level 1 and 0, 60, 40, 30, 70 for level 2). The controlled variables response are shown in Figure 59 and Figure 60 and the manipulated variable response are shown in Figure 61 and Figure 62.

For the first level, the inverse neural network control could not eliminate the offset, so it was not capable of tracking the set point in the supervisory problem. The multivariable *PID* strategy presented an overshoot larger than the *MPC-ANN* strategy. Indeed, the maximum overshoot of *MPC-ANN* was 12.0 %, while the maximum overshoot of *PID* was 42.4%. Besides, the *MPC-ANN* response was faster than the *PID* response for all the perturbations in the setpoint of the supervisory problem.

For the second level, none of the control strategies had an offset and the predictive controller based on neural network had a faster response than the other two strategies. The predictive control also presented overshoot quite smaller than the *PID* controller. Moreover, the time integral performance criteria (Table 12) showed that *MPC-ANN* technique presented a better closed-loop performance than the other two techniques. The response of the inverse neural network controller was slower than the *PID* response, but the technique did not present any overshoot. It is important to emphasize that the power 2 reached the saturation in the *PID*

control as it can be seen in Figure 62 which explains the atypical shape of the closed loop response for level 2 in the *PID* control (Figure 60).

Table 12: Comparison of the performance criteria.

		IAE	*ITAE*	*ISE*
Y_1	*MPC-ANN*	4671	2642174	132309
	PID	6566	3217326	117904
	IANN	11700	6233448	209767
Y_2	*MPC-ANN*	5584	1469384	91081
	PID	6319	3322417	110379
	IANN	9147	6244135	149271

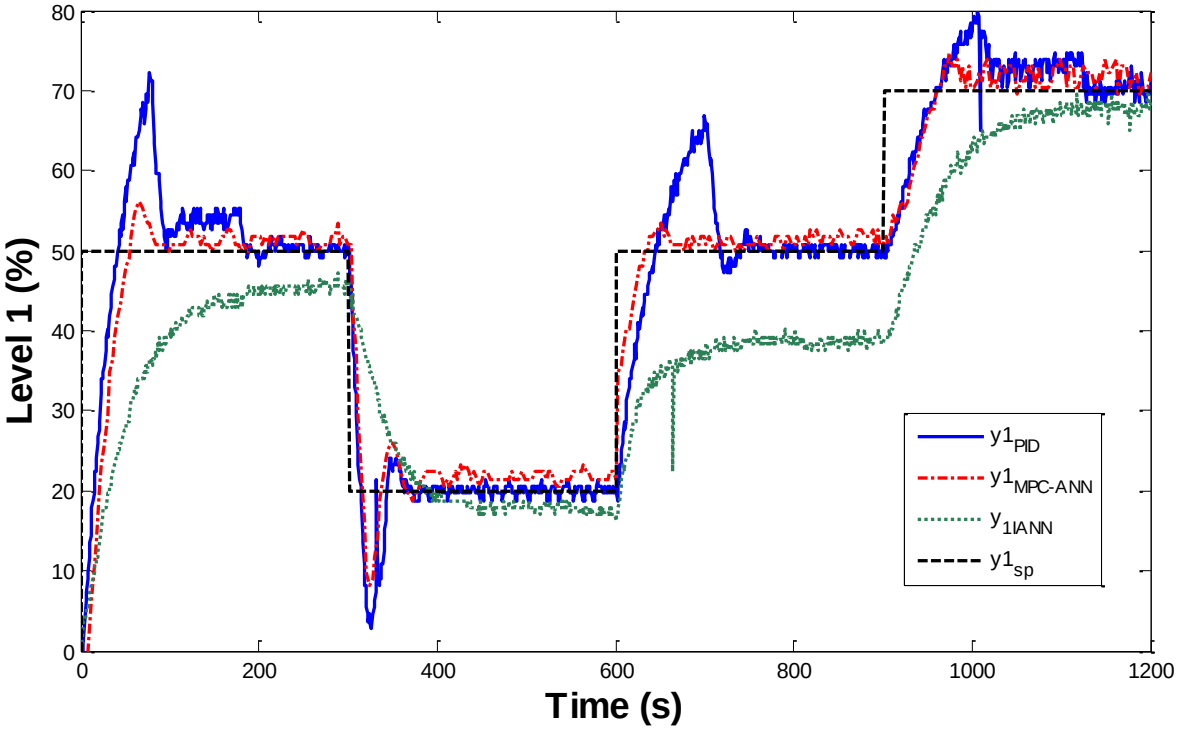

Figure 59: Performance comparison among predictive control based on neural network (*MPC-ANN*), inverse neural network (*IANN*) and conventional *PID* control for the level 1.

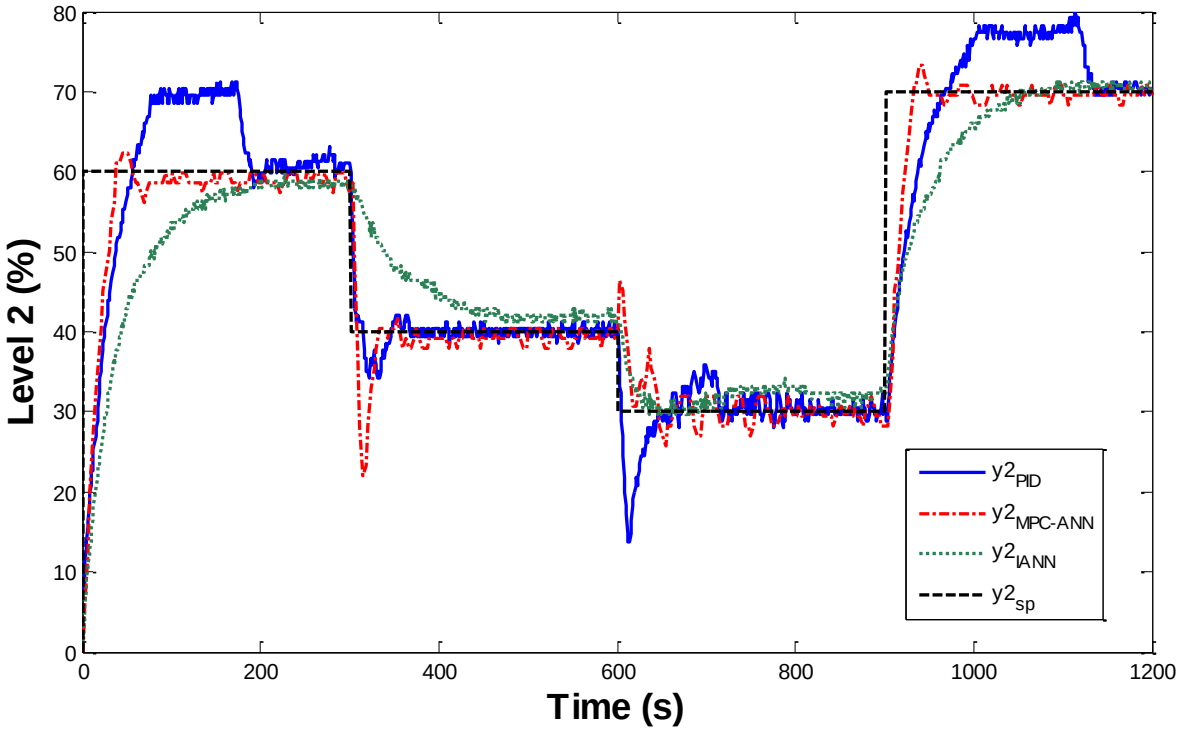

Figure 60: Performance comparison among predictive control based on neural network (*MPC-ANN*), inverse neural network (*IANN*) and conventional *PID* control for the level 2.

Figure 61 and Figure 62 shows that the control effort of the *MPC-ANN* controller was smaller than *PID* controller, so its response was smoother than the *PID* response. The use of a filter in the *PID* prevented that the response was even more aggressive, within greater variations in the manipulated variable. For the *MPC-ANN* scheme, a large control effort was prevented by choosing an ideal weight for the control action, which, in the system used in this work, was w = 0.1. For the inverse neural network control the control action was very smooth because it depended mainly on the setpoint, so the control action will quickly stabilize.

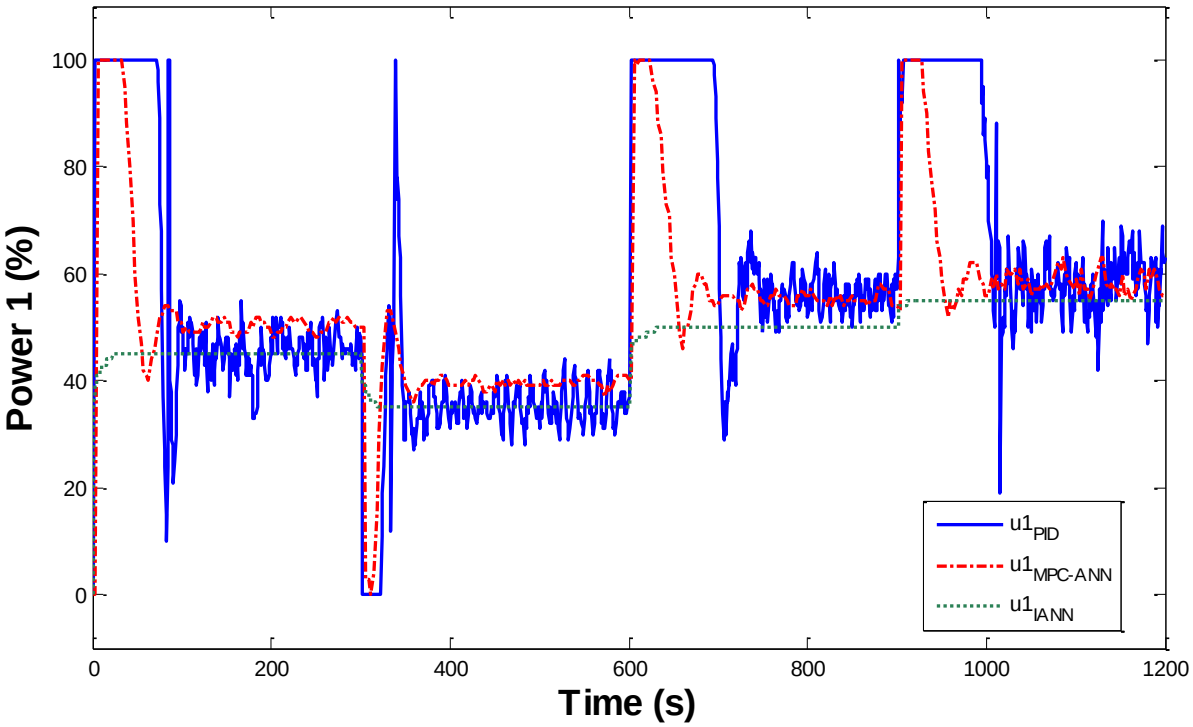

Figure 61: Control action of the power of the pump P-101.

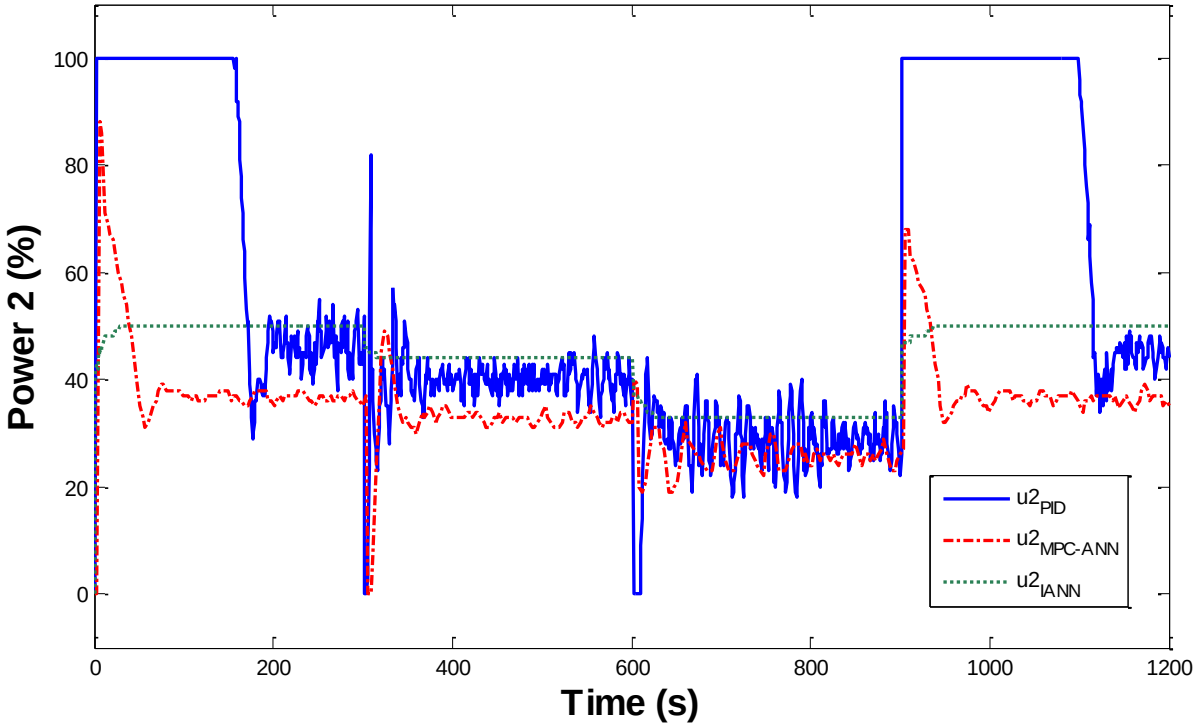

Figure 62: Control action of the power of the pump P-102.

The superior performance of the Model predictive controller based on artificial neural networks to the "well-tuned" decoupled *PID* can be explained for several reasons:

First, the decoupled *PID* is not capable to predict accurately the nonlinear interactions between the variables because it uses transfer function models, which are linear and might not give a satisfactory response for nonlinear problems as the coupled tanks process. Indeed, Seborg, Edgar and Mellichamp (2003) argues that the theoretical benefits of decouplers may not be fully realizable due to imperfect process models. On the other hand, neural network models are very accurate, and they are capable to predict complex nonlinear relationships between all inputs and outputs. Second, the ability of the model predictive control of optimizing the performance is well known. Therefore, these controllers can present a better performance, within smaller overshoots and time response.

The inverse neural network control does not perform well because this controller acts with only a feedforward action, which means it does not couple feedback with feedforward action like the *MPC-ANN*. Therefore, the performance will depend strongly on the accuracy of the model. Mismatches between the model and the real process, or unmeasured disturbances like variations in the opening of the hand valves will affect the steady state of the process variables and can lead to large offsets.

The regulatory problem was not tested in this work because the change in the position of the valves can hinder the prediction of the neural network, since the model that relates the future level with the past and current values of manipulated and controlled variables will change. Therefore, the *PID* would have a better performance than both neural network techniques for the regulatory problem since it calculates the control action based on the error.

A possible way to deal with this problem is to train the neural network offline and change the weights and biases model online as it was proposed by Muniz (2004). Another way is to use the output flow as input of the neural network so that the influence of the position of the valves in the level is captured by the neural network modelling.

5.8. *MPC-ANN* Simulation

The physical parameters shown in section 4.5 (C_{v1}, C_{v2}, C_{v3}, A_1, A_2) were used in the algorithm code (Appendix A6) in order to perform the identification of the simulated process. The open loop step tests in the flow rate of pump 1 and pump 2 are shown in Figure 63, and the response of the controlled variables are shown in Figure 64. The analysis of both figures allowed us to conclude that the total time of 7200 seconds and the time between the steps of 300 seconds was adequate since the controlled variables reached the steady state.

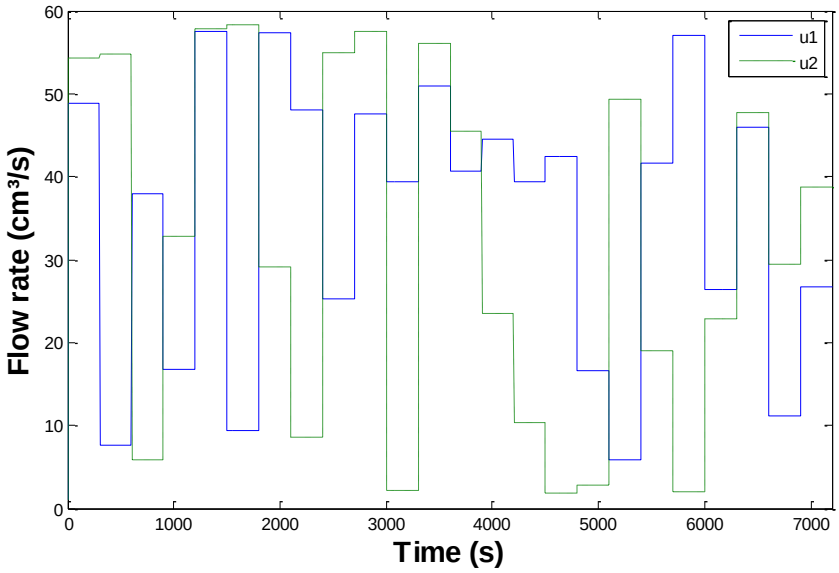

Figure 63: Step test of the flow rate in the identification process.

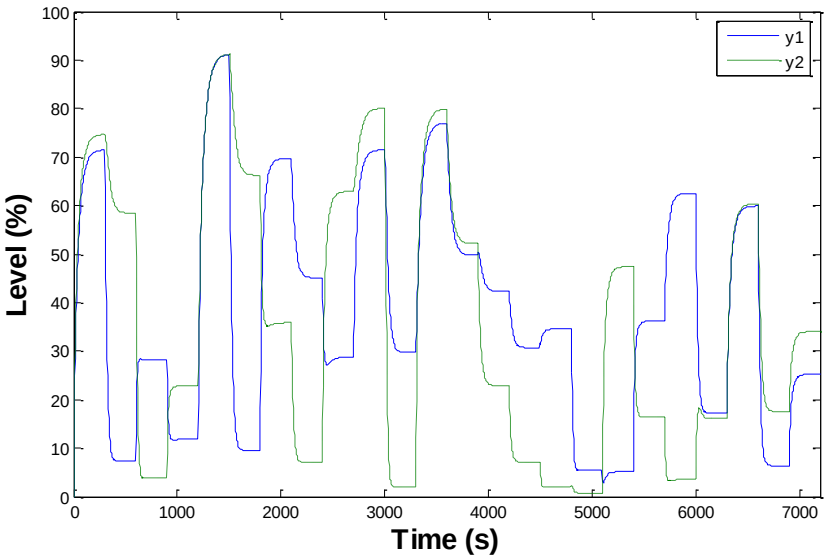

Figure 64: Open loop response in the identification process.

During the training process, the number of neurons, the number of hidden layers and the activation function was changed to obtain the best neural network structure similarly to what was performed in the section 5.4. The number of hidden layers was 1 and the number of neurons used in the hidden layer was 7 for both *ANN* (prediction of level 1 and prediction of level 2). The activation function was the tangent-sigmoid for the neural network that predicted the level 1, and log-sigmoid for the neural network that predicted the level 2.

After the training process, the controller was tuned to obtain a set of parameters (N_p, N_c, w) that provided a suitable performance similarly to what was done in the section 5.5. The

result was that the control horizon was 2, the prediction horizon was 8 and the weight of the control action was 0.5.

Finally, the model predictive controller based on artificial neural network was designed using algorithm given in appendix A7. The result of the closed loop response is shown in Figure 65 while the control action is shown in Figure 66. The maximum overshoot for the first level was 8.6 % and the maximum overshoot for the second level was 6.2 %. Besides the controller gave a fast response and it presented a smooth control action.

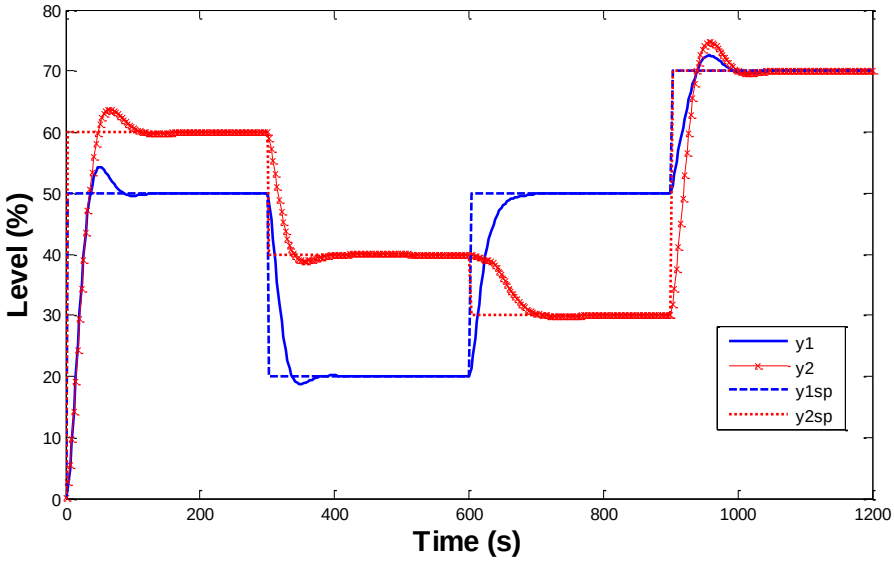

Figure 65: Simulation of the closed loop response.

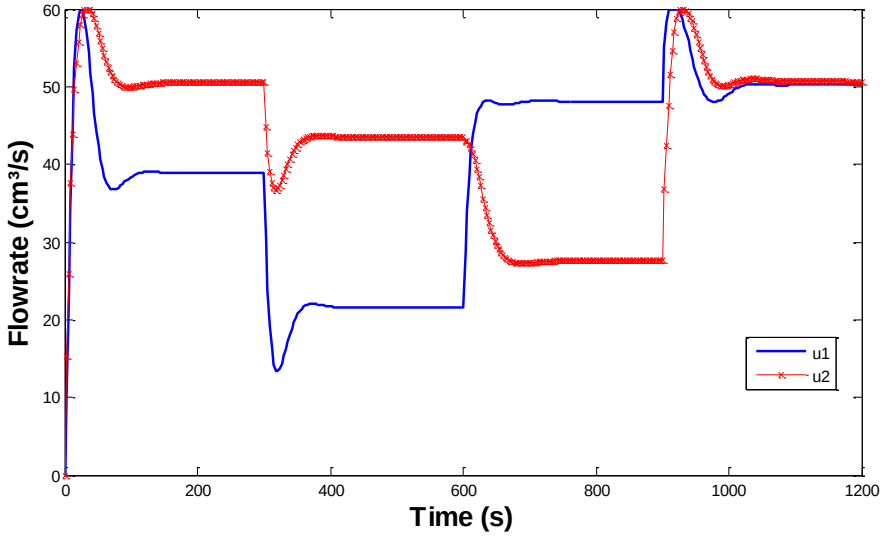

Figure 66: Control action for the simulated closed loop response.

A comparison between the results of the simulation given in this section with the closed loop response for *MPC-ANN* given in section 5.7 is shown in Figure 67 and Figure 68. Both figures prove that the model used in the simulation was adequate since the results were similar in terms of response time and overshoot. Small deviations between the simulation and the real processes can be assigned to the noises inherent of the process that were not considered in simulation.

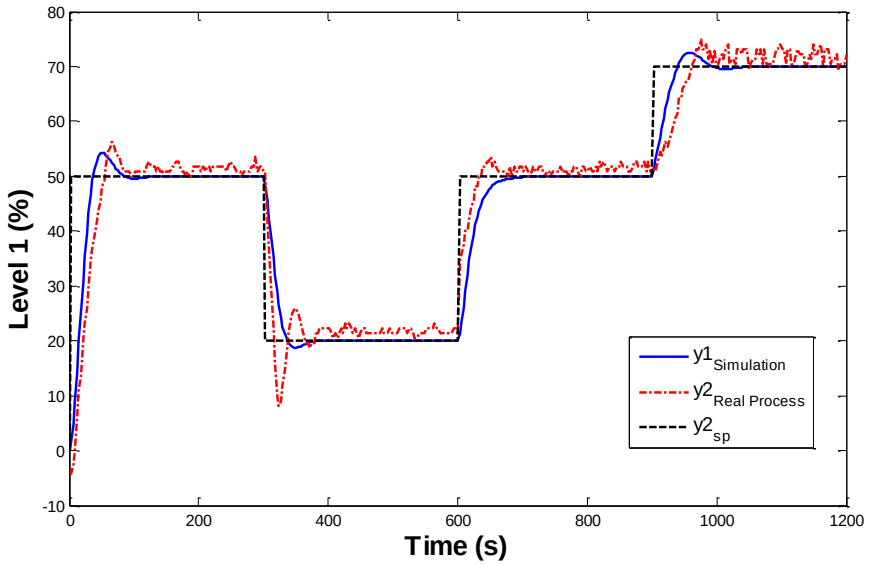

Figure 67: Comparison between simulation and real process for level 1 closed loop response using *MPC-ANN*.

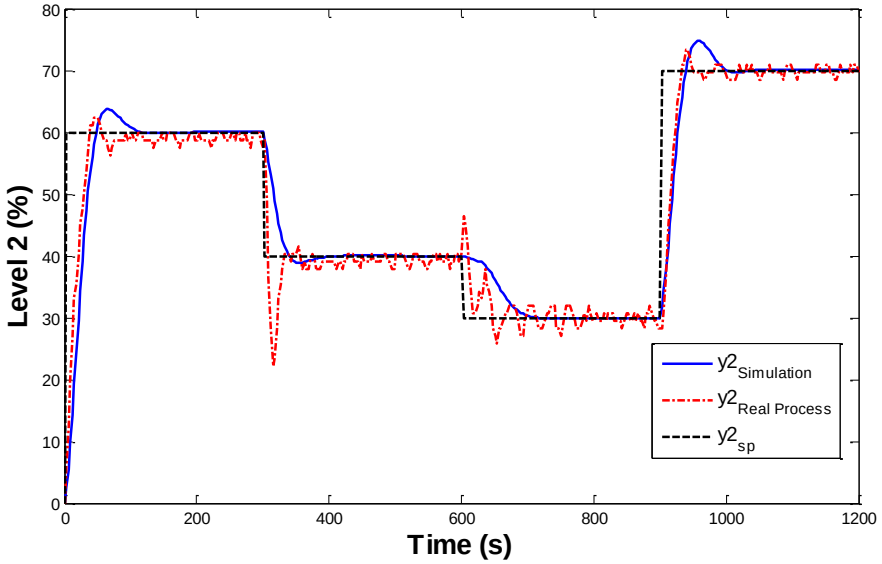

Figure 68: Comparison between simulation and real process for level 2 closed loop response using *MPC-ANN*.

6. Conclusion

In this work, three control strategies were compared: the *PID* controller, the Inverse neural network control, and the model predictive controller based on artificial neural network. The main objective was to control the level of two coupled tanks. The problem proved to be nonlinear and presented loop interactions between the manipulated variables and controlled variables, which posed challenge to the control problem.

At first, the *PID* was identified using the first order plus dead time transfer function and, then, tuned using Ziegler-Nichols relationships followed by a fine-tuning. The set of tuning parameters were: K_c = 2.5 %/%, τ_i = 12.0 seconds and τ_d = 2.29 seconds for the first control loop and K_c = 3.0 %/%, τ_i = 12.0 seconds and τ_d = 2.28 seconds for the second control loop. A decoupling control was applied using the transfer functions found in the identification process in order to eliminate loop interactions. However, the decoupling was not able to improve *PID* control performance since the process was nonlinear and the transfer functions should vary in time.

The use of a model predictive controller based on neural network was, then, proposed in order to improve control performance of a servo problem. The tuning parameters that optimize the performance were N_c = 2, N_p = 8, w_i = 0.1 and w_{yi} = 1. It was found that small values of control weight may lead to an oscillatory response. Besides, large control horizons also led to an oscillatory response.

The comparison of the three controllers was performed in a servo problem in which the level of the first and second tank were varied to different values to cause interactions between the process variables. The results showed that the *MPC-ANN* was the best control technique since it had a faster response and smaller overshoot than the other two techniques. Besides the control effort of *MPC-ANN* is smaller than the one of the *PID* controller, which could increase the lifetime of the actuator.

The first reason that explains why the predictive controller tracked the setpoint better than the other controllers was that the neural network can capture nonlinear relationships between all inputs and outputs. Second, the predictive controller couples feedforward and feedback strategy so it anticipates future process behavior and it has an adaptive structure that responds to control error and can compensate model-plant mismatches. On the other hand, inverse neural network control has only the feedforward action, so it was sensitive to model errors and unmeasured disturbances and it could not perform a satisfactory control leading to offsets.

7. Future Works

In this dissertation, the proposed strategy was used only to solve the supervisory problem because the algorithm was not capable to deal with large model mismatches caused, for example, by the change in the valve in the regulatory problem. Small model mismatches were dealt well by the disturbance model (Equations 6 and 7), but for large errors another adaptive technique is required, such as the adaptive strategy proposed by Muniz (2004), in which the neural network is trained offline and the weights and bias are changed online. Another possibility to deal with the regulatory problem is to use flow sensors in the output of the tank so that changes in the position of the valve can be captured by the neural network model as long as the flow measurements are used as input of the neural network.

The study of the stability and robustness of the *MPC-ANN* is also important and should be considered in future works. Finally, the algorithm developed in this work may be used in other control problems in chemical industry, in which the conventional feedback controller does not present a suitable performance.

8. References

AFRAM, A.; SHARIFI, F. J.; FUNG, A. S.; RAAHEMIFAR, K. Artificial neural network (*ANN*) based model predictive control (MPC) and optimization of HVAC systems: A state of the art review and case study of a residential HVAC system. **Energy And Buildings**, v. 141, p.96-113, 2017. Elsevier BV. http://dx.doi.org/10.1016/j.enbuild.2017.02.012.

ALEX, J.; BETEAU, J. F.; COPP, J. B.; HELLINGA, C.; JEPPSSON, U; MARSILI-LIBELLI, S.; PONS, M. N.; SPANJERS, H.; VANHOOREN, H. Benchmark for evaluating control strategies in wastewater treatment plants. **1999 European Control Conference (ecc)**, p.3746-3751, 1999. IEEE. http://dx.doi.org/10.23919/ecc.1999.7099914.

ASTROM, K. J.; WITTENMARK, B. **Computer Controlled Systems: Theory and design.** 3° ed. Englewood Cliffs, NJ: Prentice Hall, 1997.

CONCEPCION, H. R.; MENESES, M.; VILANOVA, R. Control strategies and wastewater treatment plants performance: Effect of controllers parameters variation. **Etfa2011**, p.1-7, 2011. IEEE. http://dx.doi.org/10.1109/etfa.2011.6059057.

DEMUTH, H., BEALE, M. HAGAN, M. **Neural Network Tollbox User's Guide for Use with MATLAB®.** USA: The Mathworks Inc., 2010. Backpropagation (Chapter 5).

DERDIYOK, A.; BASÇI, A. The application of chattering-free sliding mode controller in coupled tank liquid-level control system. **Korean Journal of Chemical Engineering**, v. 30, n. 3, p.540-545, 2012. Springer Nature. http://dx.doi.org/10.1007/s11814-012-0177-y.

DHARAMNIWAS; AHMAD, A.; REDHU, V.; GUPTA, U. Liquid Level Control by Using Fuzzy Logic. **International Journal of Advances in Engineering & Technology**, v. 4, n. 1, p. 537–549, 2012.

DRAEGER, A.; ENGEL, S.; RANKE, H. Model predictive control using neural networks. **IEEE Control Systems Magazine**, v. 15, n. 5, p.61-66, 1995. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/37.466261.

EYNG, E.; FILETI, A. M. F. Control of absorption columns in the bioethanol process: Influence of measurement uncertainties. **Engineering Applications of Artificial Intelligence**, v. 23, n. 2, p.271-282, 2010. Elsevier BV. http://dx.doi.org/10.1016/j.engappai.2009.11.002.

FILETI, A. M. F.; PACIANOTTO, T. A.; CUNHA, A. P. Neural modeling helps the BOS process to achieve aimed end-point conditions in liquid steel. **Engineering Applications of Artificial Intelligence**, v. 19, n. 1, p.9-17, 2006. Elsevier BV. http://dx.doi.org/10.1016/j.engappai.2005.06.002.

FORESEE, F. D.; HAGAN, M. T. Gauss-Newton approximation to Bayesian learning. **Proceedings Of International Conference On Neural Networks (ICNN'97)**, v. 3 p.1930-1935, 1997. IEEE. http://dx.doi.org/10.1109/icnn.1997.614194.

GAIKWAD, A. J.; VIJAYAN, P. K.; BHARTIYA, S.; KUMAR, R.; LELE, H. G.; VAZE, K. K. Selection of Steam Drum Level Control Method for Multiple Drum Interacting Loops Pressure Tube-Type BWR. **IEEE Transactions on Nuclear Science**, v. 58, n. 2, p.479-489, 2011. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/tns.2011.2108666.

HAGAN, M. T.; DEMUTH, H. B.; BEALE, M. H.; DE JESUS, O. **Neural Network Design**. 2. ed. Boston: PWS Publishing, 2014.

HAYKIN, S. **Neural Networks and Learning Machines**. 3. ed. Englewood Cliffs, NJ: Pearson Prentice Hall, 2008.

HENSON, M. A. Nonlinear model predictive control: current status and future directions. **Computers & Chemical Engineering**, v. 23, n. 2, p.187-202, 1998. Elsevier BV. http://dx.doi.org/10.1016/s0098-1354(98)00260-9.

HERMANSSON, A.W.; SYAFIIE, S. Model predictive control of pH neutralization processes: A review. **Control Engineering Practice**, v. 45, p.98-109,. 2015. Elsevier BV. http://dx.doi.org/10.1016/j.conengprac.2015.09.005.

HIMMELBLAU, D. M. Applications of artificial neural networks in chemical engineering. **Korean Journal of Chemical Engineering**, v. 17, n. 4, p.373-392, 2000. Springer Nature. http://dx.doi.org/10.1007/bf02706848.

HOSEN, M. A.; HUSSAIN, M. A.; MJALLI, F. S. Control of polystyrene batch reactors using neural network based model predictive control (NNMPC): An experimental investigation. **Control Engineering Practice**, v. 19, n. 5, p.454-467, 2011. Elsevier BV. http://dx.doi.org/10.1016/j.conengprac.2011.01.007.

KOTHARE, M. V.; METTLER, B.; MORARI, M.; BENDOTTI, P.; FALINOWER, C. M. Level control in the steam generator of a nuclear power plant. **IEEE Transactions On Control Systems Technology**, v. 8, n. 1, p.55-69, 2000. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/87.817692.

LABBE, D. E.; WOBURN, J. E.; DUTREMBLE, I. **Dearator Level Control**. US Patent 4,345,438, 1982.

LONG, L. Z.; FENG, Y.; ZHOU, K. J.; MEI, X. Control of methylamine removal reactor using neural network based model predictive control. **2014 International Joint**

Conference on Neural Networks (ijcnn), p.374-381, 2014. IEEE. http://dx.doi.org/10.1109/ijcnn.2014.6889463.

MAYNE, D. Q. Model predictive control: Recent developments and future promise. **Automatica**, v. 50, n. 12, p.2967-2986, 2014. Elsevier BV. http://dx.doi.org/10.1016/j.automatica.2014.10.128.

MERCANGÖZ, M.; DOYLE, F. J. Distributed model predictive control of an experimental four-tank system. **Journal of Process Control**, v. 17, n. 3, p.297-308, 2007. Elsevier BV. http://dx.doi.org/10.1016/j.jprocont.2006.11.003.

MELEIRO, L. A. C. **Projeto e aplicações de controladores baseados em modelos lineares, neurais e nebulosos**. Campinas: Faculdade de Engenharia Química, Universidade Estadual de Campinas, 2002. Tese (Doutorado).

MUNIZ, L. A. R. **Controle Preditivo Adaptativo Aplicado a um Reator de Pirólise Operando em Regime Semi-Batelada.** Florianópolis: Universidade Federal de Santa Catarina 2004.

NOEL, M. M.; PANDIAN, B. J. Control of a nonlinear liquid level system using a new artificial neural network based reinforcement learning approach. **Applied Soft Computing**, v. 23, p.444-451, 2014. Elsevier BV. http://dx.doi.org/10.1016/j.asoc.2014.06.037.

OGUNNAIKE, B. A.; RAY, W. H. **Process Dynamics, Modelling, and Control**. 1. ed. New York: Oxford University Press, 1994.

ROY, P.; ROY, B. K. Fractional order *PI* control applied to level control in coupled two tank *MIMO* system with experimental validation. **Control Engineering Practice**, v. 48, p.119-135, 2016. Elsevier BV. http://dx.doi.org/10.1016/j.conengprac.2016.01.002.

SAEED, Q.; UDDIN, V.; KATEBI, R. Multivariable Predictive *PID* Control for Quadruple Tank. **International Journal of Mechanical and Mechatronics Engineering**, v. 4, n.7, p. 861–866, 2010.

SEBORG, D. E.; EDGAR, T. F.; MELLICAMP, D. A. **Process Dynamics and Control**. 2. ed. New York: John Willey & Sons, 2003.

SOUZA, M. B. **Redes Neuronais Multicamadas Aplicadas a Modelagem de Processos Químicos.** Rio de Janeiro: Universidade Federal do Rio de Janeiro, 1993. Tese (Doutorado).

TAN, W. Water level control for a nuclear steam generator. **Nuclear Engineering And Design**, v. 241, n. 5, p.1873-1880, 2011. Elsevier BV. http://dx.doi.org/10.1016/j.nucengdes.2010.12.010.

TAYYEBI, S.; ALISHIRI, M. The control of MSF desalination plants based on inverse

model control by neural network. **Desalination**, v. 333, n. 1, p.92-100, 2014. Elsevier BV. http://dx.doi.org/10.1016/j.desal.2013.11.022.

QIN, S. J.; BADGWELL, T. A. A survey of industrial model predictive control technology. **Control Engineering Practice**, v. 11, n. 7, p.733-764, 2003. Elsevier BV. http://dx.doi.org/10.1016/s0967-0661(02)00186-7.

VADIGEPALLI, R.; GATZKE, E. P.; DOYLE, F. J. Robust Control of a Multivariable Experimental Four-Tank System. **Industrial & Engineering Chemistry Research**, v. 40, n. 8, p.1916-1927, 2001. American Chemical Society (ACS). http://dx.doi.org/10.1021/ie000381p.

VASICKANINOVÁ, A.; BAKOŁOVÁ, M. Control of a heat exchanger using neural network predictive controller combined with auxiliary fuzzy controller. **Applied Thermal Engineering**, v. 89, p.1046-1053, 2015. Elsevier BV. http://dx.doi.org/10.1016/j.applthermaleng.2015.02.063.

WILLIS, M. J; MONTAGUE, G. A.; DI MASSIMO, C.; THAM, M. T.; MORRIS, A. J. Artificial neural networks in process estimation and control. **Automatica**, v. 28, n. 6, p.1181-1187, 1992. Elsevier BV. http://dx.doi.org/10.1016/0005-1098(92)90059-o.

WIOR, I. R.; BOONTO, S.; ABBAS, H.; WERNER, H. Modeling and Control of an Experimental pH Neutralization Plant using Neural Networks based Approximate Predictive Control. In: **1° Virtual Control Conference,** 2010.

YU, D. L.; GOMM, J. B.; WILLIAMS, D. On-line predictive control of a chemical process using neural network models. **IFAC Proceedings Volumes**, v. 32, n. 2, p.4347-4352, 1999. Elsevier BV. http://dx.doi.org/10.1016/s1474-6670(17)56741-5.

YU, H.; ZHANG, Z. Predictive Control Based on Neural Networks of The Chemical Process. **2006 Chinese Control Conference**, p.1143-1147, 2006. IEEE. http://dx.doi.org/10.1109/chicc.2006.280528.

APPENDIX

APPENDIX A1 – Training algorithm for the neural network of the *MPC-ANN*.

```
% Script for neural network training
close all; clear all;clc;
M=load('RNA.mat');   %load datafile for training
entrada=M.RNA(1:8,:); %define inputs of the neural network (columns 1 to 8)
saida=M.RNA(9,:); % define the output of the neural network (column 9)
pause

%define maximum and minimum parameters, perform normalization. The
variables "entradan" e "saidan" are the normalized input and normalized
output respectively;
[entradan,minentrada,maxentrada,saidan,minsaida,maxsaida]=premnmx(entrada,s
aida);

% Separate the data file in training and test
ent_treina_nor=entradan(:,1:840); %input training data
sai_treina_nor=saidan(:,1:840); %output training data
ent_teste_nor=entradan(:,841:1199); %input test data
sai_teste_nor=saidan(:,841:1199); %output test data

%% Comands for neural network training
N=2; %number of neurons
net=newff(minmax(ent_treina_nor),[N,1],{'logsig','purelin'},'trainbr');
%comand to create the neural network
net.trainParam.epochs=3000;          %maximum number of steps
net.trainParam.show=50;              %update the number of steps in the
graph
net.trainParam.goal=1e-8;            %convergence goal
net.initFcn='initlay';               %initialization of weights and bias
net.performFcn='mse';                %objective function: mean square error
net.trainParam.min_grad=1e-100;      %minimum gradient
net.trainParam.mu_max=1e+100 ;       %max MU
net=init(net);                       %random initialization of the weights
and bias
[net,tr]=train(net,ent_treina_nor,sai_treina_nor); %training function

%%
Y=sim(net,ent_teste_nor);  %Y is the normalized predicted values with the
inputs of the test set
Ytreino=sim(net,ent_treina_nor);
e_nor = sai_teste_nor-Y;
etreino = sai_treina_nor-Ytreino;
mse_teste_nor = mse(e_nor);
mse_treino_nor = mse(etreino);

X=postmnmx(Y,minsaida,maxsaida);  %X is the denormalized output data

figure(1); %graph of the level 1 normalized
plot(Y(1,:),'-');
hold on
plot(sai_teste_nor(1,:),'.');
xlabel('test data');
ylabel('Level 1');
legend('Predicted Level 1 normalized','Real Level 1 normalized');
hold off;

figure(2); %
```

```
[m1,b1,r1]=postreg(X(1,:),saida(1,841:1199));    %calculation of: m1 =
angular coeficient, b1 = linear coeficient, r1 = correlation coeficient
title('TEST');
legend('Level 1','fit','predicted = real');
xlabel('real');ylabel('Predicted');
y1=X(1,:);
y1=y1';
x1=saida(1,841:1199);
x1=x1';

par=[m1,b1,r1,mse_teste_nor,mse_treino_nor];   %performance parameters of
the neural network
```

APPENDIX A2 – Algorithm of the model predictive controller based on neural networks

```
%observação 1: tirar modeloneural1 e modeloneural2
function out=controle2(in)
global ynn1 ynn2 yn1 yn2 y1sp y2sp y1p y1p1 y2p y2p1 u1p u2p settings1
settings2 settings Np Nc modeloneural1 modeloneural2
tic()
%% inputs of the MPC-ANN
y1sp=in(1); % setpoint of level 1
y2sp=in(2); % setpoint of level 2
y1p=in(3); % current value of level 1, y1(k)
y1p1=in(4); % one-step-delayed value of level 1, y1(k-1)
y2p=in(5); % current value of level 2, y2(k)
y2p1=in(6); % one-step-delyed of level 2, y2(k-1)
u1p=in(7); % current value of power 1, u1(k)
u2p=in(8); % current value of power 2, u2(k)
ynn1=in(9); % predicted value of y1 at the time k-1, y1(k|k-1)
ynn2=in(10); % predicted value of y2 at the time k-2, y2(k|k-1)
%% Tuning parameters Np = prediction horizon, Nc = control horizon
Np=16;Nc=4;

%% Load datafile with the input and output data
load ('entrada_3');
load ('saida1_3');
load ('saida2_3');

%% Normalization of input and output
[entradan,settings]=mapminmax(entrada_3);
[saida1n,settings1]=mapminmax(saida1_3);
[saida2n,settings2]=mapminmax(saida2_3);
%%
lb=0*ones(Nc,2); %lower limit of the manipulated variable
ub=100*ones(Nc,2); % upper limit of the manipulated variable

u0=50*ones(Nc,2); % initial guess

options = optimoptions('fmincon','Algorithm','sqp'); % definition of
optimization algorithm
options = optimoptions(options, 'TolFun', 1e-6, 'TolX', 1e-6); % definition
of tolerance

[Fv]=fmincon(@Fobjetivo,u0,[],[],[],[],lb,ub,[],options); % application of
the optimization algorithm
u=Fv(1,:);
```

```
ynn1=yn1;ynn2=yn2;
out=[u ynn1 ynn2]; %output of the algorithm, u = vector of manipulated
variables, ynn1 = y1(k|k-1) and ynn2 = y2(k|k-1)
toc
end
%%
function J=Fobjetivo(Fv)
global ynn1 ynn2 yn1 yn2 y1sp y2sp y1p y1p1 y2p y2p1 u1p u2p settings
settings1 settings2 Np Nc
w1=0.01;w2=0.01;wy1=1;wy2=1; % weight tuning parameters

u1(1)=u1p;u2(1)=u2p;
for j=2:Nc
    u1(j)=Fv(j-1,1);u2(j)=Fv(j-1,2);
end
u1(Nc+1:Np+2)=Fv(Nc,1);u2(Nc+1:Np+2)=Fv(Nc,2);
y1(1)=y1p1;y1(2)=y1p;y2(1)=y2p1;y2(2)=y2p;

soma1=0;soma2=0;
for j=1:Nc
    soma1=soma1+(u1(j+1)-u1(j))^2;
    soma2=soma2+(u2(j+1)-u2(j))^2;
end

soma3=0;soma4=0;
for k=3:Np+2
    x1=y1(k-1);
    x2=y1(k-2);
    x3=y2(k-1);
    x4=y2(k-2);
    x5=u1(k-1);
    x6=u1(k-2);
    x7=u2(k-1);
    x8=u2(k-2);
    ent=[x1;x2;x3;x4;x5;x6;x7;x8];
    ent=mapminmax.apply(ent,settings); % normalization of the input

x1=ent(1);x2=ent(2);x3=ent(3);x4=ent(4);x5=ent(5);x6=ent(6);x7=ent(7);x8=en
t(8);
    %neural network function
    %yt1 = prediction of level 1
    %yt2 = prediction of level 2
    yt1=(((1/(1+exp(-(x1*4.534498e-01 + x2*2.378115e-01 + x3*2.161676e-02 +
x4*5.867031e-03 + x5*4.699665e-02 + x6*3.227962e-02 + x7*1.422370e-03 +
x8*4.950159e-03 + 7.385048e-02))))*5.356746e+00 + -2.769476e+00));
    yt2=(((1/(1+exp(-(x1*4.472666e-01 + x2*-3.610939e-01 + x3*-1.036978e+00
+ x4*-5.254495e-01 + x5*-7.959184e-02 + x6*1.211340e-01 + x7*-7.538054e-01
+ x8*-3.406064e-01 + -1.592219e+00))))*-1.668303e+00 + (1/(1+exp(-
(x1*3.039242e-02 + x2*4.974226e-02 + x3*8.761524e-01 + x4*1.068648e+00 +
x5*-3.431507e-02 + x6*-1.569495e-01 + x7*5.586484e-01 + x8*3.166613e-01 + -
1.803750e+00))))*1.226841e+00 + (1/(1+exp(-(x1*-8.383311e-01 + x2*-
1.969045e-01 + x3*3.434415e-01 + x4*1.701615e-02 + x5*-2.404522e-01 + x6*-
6.234841e-01 + x7*3.184416e-01 + x8*6.341119e-01 + 6.100933e-02))))*-
3.304020e-01 + (1/(1+exp(-(x1*-4.342128e-01 + x2*6.063844e-01 + x3*-
5.450199e-01 + x4*-5.567256e-01 + x5*7.734122e-02 + x6*-5.911347e-02 +
x7*1.010859e+00 + x8*2.550675e-01 + -4.946456e-01))))*-9.124021e-01 +
6.752432e-01));
    e1=(y1p-ynn1);
    e2=(y2p-ynn2);
```

```
    y1(k)=mapminmax.reverse(yt1,settings1)+e1; %denormalization and
application of the disturbance model
    y2(k)=mapminmax.reverse(yt2,settings2)+0*e2; %denormalization and
application of the disturbance model

    soma3=soma3+(y1sp-y1(k))^2;
    soma4=soma4+(y2sp-y2(k))^2;
end

J=w1*soma1+w2*soma2+wy1*soma3+wy2*soma4; %calculation of the objective
function
yn1=y1(3);
yn2=y2(3);
end
```

APPENDIX A3 - Training algorithm for the inverse neural network.

```
% Script for neural network training
close all;clear all;clc;
M=load('RNAinv.mat');   %load datafile for training
entrada=M.RNAinv(1:4,:); %define inputs of the neural network (colunms 1 to
4)
saida=M.RNAinv(5:6,:); % define the output of the neural network (colunms 5
and 6)
pause

%Perform normalization. The variables "entradan" e "saidan" are the
normalized input and normalized output respectively;
[entradan,settings]=mapminmax(entrada);
[saida1n,settings1]=mapminmax(saida);

% Separate the data file in training and test
ent_treina_nor=entradan(:,1:138); %input training data
sai_treina_nor=saidan(:,1:138); %output training data
ent_teste_nor=entradan(:,139:198); %input test data
sai_teste_nor=saidan(:,139:198); %output test data

%% Comands for neural network training
 N=14; %number of neurons
%INÍCIO DOS COMANDOS PADRÃO PARA TREINO DA REDE
net=newff(minmax(ent_treina_nor),[N,1],{'logsig','purelin'},'trainbr');
%comand to create the neural network
net.trainParam.epochs=3000;          %maximum number of steps
net.trainParam.show=50;              %update the number of steps in the
graph
net.trainParam.goal=1e-8;            %convergence goal
net.initFcn='initlay';               %initialization of weights and bias
net.performFcn='mse';                %objective function: mean square error
net.trainParam.min_grad=1e-100;      %minimum gradient
net.trainParam.mu_max=1e+100 ;       %max MU
net=init(net);                       %random initialization of the weights
and bias
[net,tr]=train(net,ent_treina_nor,sai_treina_nor); %training function
%%
Y=sim(net,ent_teste_nor);  %Y is the normalized predicted values with the
inputs of the test set
Ytreino=sim(net,ent_treina_nor);
e_nor = sai_teste_nor-Y;
etreino = sai_treina_nor-Ytreino;
mse_teste_nor = mse(e_nor);
```

```
mse_treino_nor = mse(etreino);

X=postmnmx(Y,minsaida,maxsaida);  %X is the denormalized output data

figure(1); %graph of the level 1 normalized
plot(Y(1,:),'-');
hold on
plot(sai_teste_nor(1,:),'.');
xlabel('vetor de teste');
ylabel('Power 1');
legend('Power 1 normalizada calculado','Nivel 1 normalizada real');
hold off;

figure(2);
% power 1
[m1,b1,r1]=postreg(X(1,:),saida(1,139:198));    %calculation of: m1 =
angular coeficient, b1 = linear coeficient, r1 = correlation coeficient
title('TESTE');
legend('Power 1','fit','Predicted=Real');
xlabel('Real');ylabel('Predicted');
y1=X(1,:);
y1=y1';
x1=saida(1,139:198);
x1=x1';

figure(3);
% power 2
[m2,b2,r2]=postreg(X(2,:),saida(2,139:198));    %calculation of: m2 =
angular coeficient, b2 = linear coeficient, r2 = correlation coeficient
title('TESTE');
legend('Power 2','fit','Predicted=Real');
xlabel('Real');ylabel('Predicted');
y2=X(2,:);
y2=y2';
x2=saida(1,139:198);
x2=x2';

par=[m1,b1,r1,mse_teste_nor,mse_treino_nor]; %performance parameters of the
neural network 1
par2=[m2,b2,r2,mse_teste_nor,mse_treino_nor]; %performance parameters of
the neural network 2
```

APPENDIX A4 – Algorithm of inverse neural network.

```
function u=controleinverso(in)
x1=in(1);
x2=in(2);
x3=in(3);
x4=in(4);
load ('entrada');
load ('saida1');
load ('saida2');
[entradan,settings]=mapminmax(entrada);
[saida1n,settings1]=mapminmax(saida1);
[saida2n,settings2]=mapminmax(saida2);
ent=[x1;x2;x3;x4];
ent=mapminmax.apply(ent,settings)
x1=ent(1);x2=ent(2);x3=ent(3);x4=ent(4);
```

```
ut1=(((1/(1+exp(-(x1*8.380805e-02 + x2*-3.870490e-01 + x3*-1.012338e+00 +
x4*3.257835e-01 + -2.828167e-01))))*-1.467070e+00 + (1/(1+exp(-
(x1*9.219810e-02 + x2*1.989005e-01 + x3*5.413571e-01 + x4*-1.625672e-01 +
5.118996e-02))))*6.008189e-01 + (1/(1+exp(-(x1*3.546345e-01 + x2*-
6.870465e-01 + x3*1.473152e+00 + x4*-5.300009e-01 + -7.519748e-
01))))*1.658855e+00 + (1/(1+exp(-(x1*9.219830e-02 + x2*1.989002e-01 +
x3*5.413561e-01 + x4*-1.625667e-01 + 5.118978e-02))))*6.008177e-01 + -
3.707118e-01));
x1=x4;
ut2=(((2/(1+exp(-2*(x1*1.748690e+00 + -1.788634e+00)))-1)*1.835610e+00 +
(2/(1+exp(-2*(x1*2.374874e+00 + -9.450189e-01)))-1)*-2.029085e+00 +
(2/(1+exp(-2*(x1*-4.817538e+00 + 1.541341e+00)))-1)*-1.580950e+00 +
(2/(1+exp(-2*(x1*1.117982e+00 + 1.529792e-01)))-1)*-8.345086e-01 +
(2/(1+exp(-2*(x1*-2.878223e+00 + -5.899749e-01)))-1)*-1.038933e+00 +
(2/(1+exp(-2*(x1*1.117982e+00 + 1.529792e-01)))-1)*-8.345086e-01 +
(2/(1+exp(-2*(x1*-1.478342e+00 + -1.734772e+00)))-1)*-1.440537e+00 + -
1.725358e-01));
u(1)=mapminmax.reverse(ut1,settings1);
u(2)=mapminmax.reverse(ut2,settings2);
```

APPENDIX A5 – Simulation of the coupled tanks process.

```
%%Simulation of the neural controller
function simulacao
global U
clc; clear all; close all;
t0=0;tmax=600;h=1; %simulation of the neural training
Lc=20; %height of the tank
y0=[0,0]'; %initial value of the tank
%transformation from % to cm
y0=y0/100;
y0=y0*Lc;
[t,y]=RK_1(@modelo,h,t0,tmax,y0); %application of the runge-kutta function
plot(t,5*y(:,1),t,5*y(:,2));
legend('y1','y2');
end
function dydt=modelo(t,y)
global U
  h1=y(1);h2=y(2); % h1 e h2 are the current levels (k)
    A1=80;A2=80;Cv1=13.5;Cv2=13.5;Cv3=2.7; %parâmetros da EDO
    if t<300
        u=[0 0]; % initial power of the pump 1 and 2 respectively (in %)
    else
        u=[100 100]; % final power of the pump 1 and 2 respectively (in %)
    end

    if u(1)>20
%correlation between the power 1 (in %) and the input flowrate(cm³/s)
        F(1)=(-1.013*10^-1)+2.115*(u(1)-20)-(2.43*10^-2)*(u(1)-
20)^2+(8.80*10^-5)*(u(1)-20)^3;
    else
        F(1)=0;
    end
%correlation between the power 2 (in %) and the input flowrate(cm³/s)
    if u(2)>30
        F(2)=-1.62+2.249*(u(2)-30)-(2.70*10^-2)*(u(2)-30)^2+(1.06*10^-
4)*(u(2)-30)^3;
    else
        F(2)=0;
    end
```

```
    % differential equation
    if(h1>h2)
        dydt(1)=F(1)/A1-(Cv1/A1)*(h1)^(1/2)-(Cv3/A1)*(h1-h2)^(1/2);
        dydt(2)=F(2)/A2-(Cv2/A2)*(h2)^(1/2)+(Cv3/A2)*(h1-h2)^(1/2);
    else if(h1==h2)
        dydt(1)=F(1)/A1-(Cv1/A1)*(h1)^(1/2);
        dydt(2)=F(2)/A2-(Cv2/A2)*(h2)^(1/2);
        else
        dydt(1)=F(1)/A1-(Cv1/A1)*(h1)^(1/2)+(Cv3/A1)*(h2-h1)^(1/2);
        dydt(2)=F(2)/A2-(Cv2/A2)*(h2)^(1/2)-(Cv3/A2)*(h2-h1)^(1/2);
        end
    end
    %
    dydt=dydt';
U=[U;F];
end
```

APPENDIX A6 – Algorithm of identification of the coupled tanks process for simulation.

```
%%Simulation of the neural training
function treinamentoRK_1
clc; clear all; close all;
global F rtempos p x h
    Lc=20; %height of the tank
    h=3; %sample time of the process
    t0=0;tmax=7200; %Simulation time limit
    tspan=[t0,tmax];
    x=-h;
    rtempos=int64(300/h);
    p=rtempos;
    y0=[0,0]; %initial height
    F=[1,1]; %initial flowrate
    [t,y]=RK_1(@modelo,h,t0,tmax,(y0/100)*Lc); %aplication of the runge-
kutta function
    figure(1)
    y=real(y);
    plot(t,(y(:,1)*100/Lc),t,(y(:,2)*100/Lc));
    legend('y1','y2');xlabel('Time (s)');ylabel('Level (%)')
    figure(2)
    plot(t,F(:,1),t,F(:,2));
    legend('u1','u2');xlabel('Time (s)');ylabel('Power (%)')
    %definition of inputs and outputs of the neural network
    for k=3:length(y)
    %inputs
    entrada(1,k-2)=y(k-1,1);
    entrada(2,k-2)=y(k-2,1);
    entrada(3,k-2)=y(k-1,2);
    entrada(4,k-2)=y(k-2,2);
    entrada(5,k-2)=F(k-1,1);
    entrada(6,k-2)=F(k-2,1);
    entrada(7,k-2)=F(k-1,2);
    entrada(8,k-2)=F(k-2,2);
    %outputs
    saida1(1,k-2)=y(k,1);
    saida2(1,k-2)=y(k,2);
    end
    save entrada
```

```
    save saida1
    save saida2
end

function dydt=modelo(t,y)
global F rtempos p x h F1 F2
    h1=y(1);
    h2=y(2);
    A1=80;A2=80;Cv1=13.5;Cv2=13.5;Cv3=2.7; %Coeficients of the neural
network
    LI=0;LS=60; %Superior limit (LS) and inferior limit (LI) of the
flowrate in cm³/s (power of 100% means 60 cm³/s)
    %definition of flowrate of each pump
    if t==x+h
        if p==rtempos
            F1=LI+(LS-LI)*rand(1,1);
            F2=LI+(LS-LI)*rand(1,1);
        p=1;
        else
            F1=F(end,1);F2=F(end,2);
            p=p+1;
        end
        u=[F1,F2];
        F=[F;u];
        x=t;
    end
    %calculation of the derivatives
    if(h1>h2)
        dydt(1)=F1/A1-(Cv1/A1)*(h1)^(1/2)-(Cv3/A1)*(h1-h2)^(1/2);
        dydt(2)=F2/A2-(Cv2/A2)*(h2)^(1/2)+(Cv3/A2)*(h1-h2)^(1/2);
    else if(h1==h2)
        dydt(1)=F1/A1-(Cv1/A1)*(h1)^(1/2);
        dydt(2)=F2/A2-(Cv2/A2)*(h2)^(1/2);
        else
        dydt(1)=F1/A1-(Cv1/A1)*(h1)^(1/2)+(Cv3/A1)*(h2-h1)^(1/2);
        dydt(2)=F2/A2-(Cv2/A2)*(h2)^(1/2)-(Cv3/A2)*(h2-h1)^(1/2);
        end
    end
end
```

APPENDIX A7 – Algorithm of *MPC-ANN* for simulation.

```
%%Simulação de um controle neural em um sistema de tanques interativos
%OBSERVAÇÃO 1: o passo da equação diferencial deve ter sido o mesmo
% daquele utilizado no treinamento da rede neural
%OBSERVAÇÃO 2: a rede deve ser treinada com a função treinamentoRK_1 e
%explicitada com a função getNeuralNetExpression
function control
clc; clear all; close all;
global U h1p1 h2p1 settings settings1 settings2 ynn1 ynn2 y1sp1 y2sp1 y1sp2
y2sp2 y1sp3 y2sp3 y1sp4 y2sp4 Np Nc
load entrada
load saida1
load saida2
[entradan,settings]=mapminmax(entrada); % normalization of the input vector
[saida1n,settings1]=mapminmax(saida1); % normalization of the output vector
1 (level 1)
[saida2n,settings2]=mapminmax(saida2);% normalization of the output vector
2 (level 2)
% definition of the input variables of the numerical method
```

```
Lc=20;
h=3; % h is the sample time
y0=[0.1,0.1]'; % y0 is the initial condition
t0=0;tmax=1200; % initial (t0) and final (tmax) of the simulation
% definition of the global variables
U=[0,0]; % U is the vector of manipulated varibales (flowrate).
h1p1=y0(1);h2p1=y0(2); % h1p1 e h2p1 are the levels of  k-1
ynn1=y0(1);ynn2=y0(2); %ynn1 e ynn2 are the values predicted in k-1 moment,
y(k/k-1)
% setpoints
y1sp1=50; y2sp1=60;
y1sp2=20; y2sp2=40;
y1sp3=50; y2sp3=30;
y1sp4=70; y2sp4=70;
% controller parameters
Np=8; Nc=2;
[t,y]=RK_1(@modelo,h,t0,tmax,y0*Lc/100); % application of the Runge Kutta
method
% definition of the vector of setpoints
Y1SP(1)=0;Y2SP(1)=0;
Y1SP(2:101)=y1sp1;Y2SP(2:101)=y2sp1;
Y1SP(102:201)=y1sp2;Y2SP(102:201)=y2sp2;
Y1SP(202:301)=y1sp3;Y2SP(202:301)=y2sp3;
Y1SP(302:401)=y1sp4;Y2SP(302:401)=y2sp4;

% control graph
figure(1)
pl=plot(t,(y(:,1)*100/Lc),'b',t,(y(:,2)*100/Lc),'r',t,Y1SP,'b--',t,Y2SP,'r-
-');
set(pl(1),'linewidth',3);
set(pl(2),'linewidth',3);
legend('Nível 1','Nível 2','Y1SP','Y2SP');
ylabel('Nível (m)');
xlabel('tempo (s)');
% Flowrate graph
figure(2)
plot(t,U(:,1),t,U(:,2))
legend('u1','u2');xlabel('Time (s)');ylabel('Flowrate (cm³/s)');
end
function dydt=modelo(t,y)
global h1 h2 h1p1 h2p1 U time
    h1=y(1);h2=y(2); % h1 e h2 are the current levels (k)
    time=t;
    A1=80;A2=80;Cv1=13.5;Cv2=13.5;Cv3=2.7; %parameters of the ODE
    F=controle; %calculation of the flowrate

    % differential equations
    if(h1>h2)
        dydt(1)=F(1)/A1-(Cv1/A1)*(h1)^(1/2)-(Cv3/A1)*(h1-h2)^(1/2);
        dydt(2)=F(2)/A2-(Cv2/A2)*(h2)^(1/2)+(Cv3/A2)*(h1-h2)^(1/2);
    else if(h1==h2)
        dydt(1)=F(1)/A1-(Cv1/A1)*(h1)^(1/2);
        dydt(2)=F(2)/A2-(Cv2/A2)*(h2)^(1/2);
        else
        dydt(1)=F(1)/A1-(Cv1/A1)*(h1)^(1/2)+(Cv3/A1)*(h2-h1)^(1/2);
        dydt(2)=F(2)/A2-(Cv2/A2)*(h2)^(1/2)-(Cv3/A2)*(h2-h1)^(1/2);
        end
    end
    dydt=dydt';
    h1p1=h1;h2p1=h2; % h1p1 e h2p1 recebem os valores de nível atual para
que no próximo instante tenham os valores dos níveis em k-1
```

```
end

function F=controle
global Nc U yn1 yn2 ynn1 ynn2
%initial guess of the flowrate
F01=30*ones(Nc,1);
F02=30*ones(Nc,1);
F0=[F01 F02];
lb=zeros(Nc,2); % inferior limit
ub=60*ones(Nc,2); % superior limit
options = optimoptions('fmincon','Algorithm','sqp','Display','off');
options = optimoptions(options, 'TolFun', 1e-9, 'TolX', 1e-9);
Fv=fmincon(@Fobjetivo,F0,[],[],[],[],lb,ub); % optimization equation
ynn1=yn1;ynn2=yn2;
F=Fv(1,:); % Nc values of flowrate are predicted, but only one is
implemented (receding horizon)
U=[U;F]; % U is the vector of the of flowrate since the beggining of the
process
end

function J=Fobjetivo(Fv)
global Nc Np U h1 h2 h1p1 h2p1 settings settings1 settings2 ynn1 ynn2 yn1
yn2 y1sp1 y2sp1 y1sp2 y2sp2 y1sp3 y2sp3 y1sp4 y2sp4 time
Lc=20;
y1r1=y1sp1*Lc/100;y2r1=y2sp1*Lc/100;
y1r2=y1sp2*Lc/100;y2r2=y2sp2*Lc/100;
y1r3=y1sp3*Lc/100;y2r3=y2sp3*Lc/100;
y1r4=y1sp4*Lc/100;y2r4=y2sp4*Lc/100;
w1=0.5;w2=0.5;wy1=1;wy2=1; % weight parameters
u1(1)=U(end,1);u2(1)=U(end,2); % definition of the control actions at the
moment k-1
for j=2:Nc
u1(j)=Fv(j-1,1);u2(j)=Fv(j-1,2); % definition of the control actions from
the moment k to the moment k+Nc-2
end
u1(Nc+1:Np+2)=Fv(Nc,1);u2(3:Np+2)=Fv(Nc,2); % definition of the control
action from k+Nc-1 to k+Np (constante)
y1(1)=h1p1;y2(1)=h2p1;y1(2)=h1;y2(2)=h2; % levels of the moment k and k-1
soma1=0;soma2=0;
for j=1:Nc
soma1=soma1+(u1(j+1)-u1(j))^2;
soma2=soma2+(u2(j+1)-u2(j))^2;
end
soma3=0;soma4=0;
for k=3:Np+2
x1=y1(k-1);
x2=y1(k-2);
x3=y2(k-1);
x4=y2(k-2);
x5=u1(k-1);
x6=u1(k-2);
x7=u2(k-1);
x8=u2(k-2);
in=[x1;x2;x3;x4;x5;x6;x7;x8];
in=mapminmax.apply(in,settings); % normalization of the input of neural
network
x1=in(1);x2=in(2);x3=in(3);x4=in(4);x5=in(5);x6=in(6);x7=in(7);x8=in(8);
% explicit equations, obtained by the function getNeuralNetExpression
yt1=(((2/(1+exp(-2*(x1*2.819016e-01 + x2*1.949106e-02 + x3*-5.865203e-02 +
x4*1.482387e-01 + x5*9.494317e-04 + x6*-7.034781e-03 + x7*-1.688970e-03 +
x8*-4.399566e-02 + 8.658786e-02)))-1)*6.005468e-01 + (2/(1+exp(-
```

```
2*(x1*3.407487e-01 + x2*2.443808e-02 + x3*1.723912e-02 + x4*2.284365e-01 +
x5*-6.579133e-02 + x6*-5.289355e-04 + x7*-5.711147e-03 + x8*2.415948e-02 +
-1.103093e-01)))-1)*6.320098e-01 + (2/(1+exp(-2*(x1*2.819006e-01 +
x2*1.949049e-02 + x3*-5.865179e-02 + x4*1.482383e-01 + x5*9.502991e-04 +
x6*-7.033653e-03 + x7*-1.687004e-03 + x8*-4.399131e-02 + 8.658136e-02)))-
1)*6.005439e-01 + (2/(1+exp(-2*(x1*2.994366e-01 + x2*1.901327e-02 + x3*-
1.957330e-01 + x4*-4.358254e-02 + x5*3.819426e-02 + x6*1.314739e-02 +
x7*1.839636e-02 + x8*-3.871393e-02 + -2.683485e-01)))-1)*6.135540e-01 +
(2/(1+exp(-2*(x1*1.156525e-01 + x2*2.164519e-01 + x3*-1.802159e-01 + x4*-
2.764737e-01 + x5*1.637405e-01 + x6*2.533575e-01 + x7*-3.105567e-01 + x8*-
3.933703e-01 + -4.479905e-03)))-1)*-1.853420e-01 + (2/(1+exp(-2*(x1*-
1.560405e-01 + x2*1.402003e-01 + x3*1.624949e-01 + x4*-5.298958e-02 + x5*-
1.655863e-01 + x6*-1.351127e-01 + x7*2.013989e-01 + x8*2.252530e-01 + -
5.052635e-02)))-1)*-5.671674e-01 + (2/(1+exp(-2*(x1*-2.578701e-01 + x2*-
1.795982e-02 + x3*2.207109e-01 + x4*1.830374e-02 + x5*-9.760966e-02 +
x6*4.709090e-02 + x7*-1.943519e-01 + x8*-1.964619e-01 + -2.129366e-02)))-
1)*-5.897930e-01 + 5.459090e-02));
yt2=(((1/(1+exp(-(x1*5.306134e-02 + x2*-1.799700e-01 + x3*-5.332491e-01 +
x4*1.395022e-01 + x5*-1.604432e-01 + x6*8.489266e-02 + x7*-9.537278e-02 +
x8*-1.990108e-01 + -4.202810e-01))))*-1.024518e+00 + (1/(1+exp(-(x1*-
3.058151e-01 + x2*3.150818e-01 + x3*1.314145e+00 + x4*-1.756939e-01 +
x5*1.295084e-01 + x6*2.516001e-01 + x7*2.078133e-01 + x8*-3.738689e-01 +
1.771614e-01))))*1.580659e+00 + (1/(1+exp(-(x1*9.500831e-02 + x2*-
1.139980e-01 + x3*-6.932309e-01 + x4*4.146418e-01 + x5*5.871150e-02 + x6*-
9.551917e-02 + x7*-2.100131e-01 + x8*-2.780395e-02 + 3.612882e-01))))*-
1.184877e+00 + (1/(1+exp(-(x1*1.063424e-01 + x2*-1.524168e-01 + x3*-
5.781159e-01 + x4*8.719311e-02 + x5*2.256239e-01 + x6*2.383932e-01 +
x7*6.394211e-01 + x8*-1.046199e-01 + -3.247029e-01))))*-9.596357e-01 +
(1/(1+exp(-(x1*8.615060e-02 + x2*-1.535650e-02 + x3*-5.702494e-01 +
x4*2.991390e-01 + x5*-3.522824e-02 + x6*5.129424e-02 + x7*-4.454425e-02 +
x8*-7.157821e-02 + 1.291094e-01))))*-1.005734e+00 + (1/(1+exp(-(x1*-
8.834785e-02 + x2*4.751083e-02 + x3*6.859910e-01 + x4*-4.109392e-01 + x5*-
7.433157e-03 + x6*2.281875e-03 + x7*9.650158e-02 + x8*5.570399e-02 + -
2.527870e-01))))*1.226195e+00 + (1/(1+exp(-(x1*-3.744634e-02 + x2*-
7.822372e-02 + x3*6.821928e-01 + x4*-4.208513e-01 + x5*7.119735e-03 + x6*-
1.001318e-01 + x7*-5.182710e-02 + x8*1.083755e-01 + -3.668772e-
01))))*1.256422e+00 + 9.865168e-02));
yt1=mapminmax.reverse(yt1,settings1); % denormalization of level 1
yt2=mapminmax.reverse(yt2,settings2); % denormalization of level 2
e1=h1-ynn1;e2=h2-ynn2;
y1(k)=yt1+e1;y2(k)=yt2+e2; % application of the disturbance model to
correct the model-plant mismatches
if time<300
    soma3=soma3+(y1(k)-y1r1)^2;soma4=soma4+(y2(k)-y2r1)^2;
    elseif time<600
     soma3=soma3+(y1(k)-y1r2)^2;soma4=soma4+(y2(k)-y2r2)^2;
    elseif time<900
     soma3=soma3+(y1(k)-y1r3)^2;soma4=soma4+(y2(k)-y2r3)^2;
    else
     soma3=soma3+(y1(k)-y1r4)^2;soma4=soma4+(y2(k)-y2r4)^2;
end
if k==3
    yn1=yt1;yn2=yt2;
end
end
J=w1*soma1+w2*soma2+wy1*soma3+wy2*soma4
end
```