Neural Computing and Applications
https://doi.org/10.1007/s00521-019-04195-9

WSOM 2017

Dynamics identification and control of nonlinear MIMO coupled plant using supervised neural gas and comparison with recurrent neural controller

Iván Machón-González[1] · Hilario López-García[1] · Ignacio Bocos-Barranco[1]

Received: 1 March 2018 / Accepted: 11 April 2019

Abstract
The dynamics identification and subsequent control of a nonlinear system is not a trivial issue. The application of a neural gas network that is trained with a supervised batch version of the algorithm can produce identification models in a robust way. In this paper, the neural model identifies each local transfer function, demonstrating that the local linear approximation can be done. Moreover, other parameters are analyzed in order to obtain a correct modeling. Furthermore, the algorithm is applied to control a nonlinear multi-input multi-output system composed of tanks. In addition, this plant is a coupled system where the manipulated input variables are influencing all the output variables. The aim of the work is to demonstrate that the supervised neural gas algorithm is able to obtain linear models to be used in a state space design scenario to control nonlinear coupled systems and guarantee a robust control method. The results are compared with the common approach of using a recurrent neural controller trained with a dynamic backpropagation algorithm. Regarding the steady-state errors in disturbance rejection, reference tracking and sensitivity to simple process changes, the proposed approach shows an interesting application to control nonlinear plants.

Keywords Neural gas · Dynamics identification · Nonlinear · Transfer function · State feedback control

1 Introduction

Neural gas (NG) is an unsupervised learning algorithm [1] that produces partitions of the input data space by means of a set of prototype vectors which represent the probability density function. It has a procedure based on a cooperation-competition computation that avoids local minima problems. The batch version allows quick convergence so that the correct training is achieved in a few epochs [2]. Supervised learning versions of NG for classification have also been developed [3, 4] which have great robustness for clustering tasks.

The initial application of neural networks (NN) for nonlinear system identification and subsequent control can be considered in the early 1990s with some relevant works [5]. The modeling of nonlinear dynamic systems usually employs the nonlinear autoregressive network with exogenous inputs (NARX). The implementation allows multidimensional inputs and outputs. This is a recurrent dynamic network with feedback signals and delayed connections. The architecture of the NN is important since the components of the network must be differentiable and this type of nets has problems of overfitting. Special care must be taken when obtaining a reliable model by means of the suitable generalization procedures [6]. The control applications using NN suffer from trial-and-error methods. Firstly, the plant model is obtained using a series-parallel architecture where the true output is used for training instead of feeding back the estimated output [7]. When the plant model is obtained, the neural controller is trained by means of dynamic backpropagation because of the necessary feedback connections. This controller is trained in order to follow a specified dynamics described by a reference model. The training of recurrent networks is usually more difficult and takes much longer than the static backpropagation for feedforward networks [8]. Many

✉ Iván Machón-González
machonivan@uniovi.es
http://isa.uniovi.es/∼ivan/

[1] University of Oviedo, Edificio Departamental Oeste 2, Campus de Viesques s/n, 33204 Gijón/Xixón, Spain

Published online: 24 May 2019

approaches were based on a dynamic backpropagation algorithm to update the weights of the NN [9], and there are several versions of the algorithm implementation [10]. The algorithms for dynamic NNs have been improved using the Lyapunov synthesis method to prove the stability of the identification [11]. Although the control algorithms have been improved [12] with a strong mathematical foundation, the practical implementation lacks robustness, so not guaranteeing the necessary stability of the plant to be controlled. However, the supervised version of the NG for dynamic identification has also been proven to be robust to obtain direct dynamic models of plants [13, 14]. In addition, the algorithm is able to obtain local linear dynamic models which can be employed to control a nonlinear plant.

The first aim is to identify a simple nonlinear plant in which nonlinearity is represented by the existing switching between the linear subsystems that compose it. The subsystems are considered as first and second order. In this way, the number of local linear models (neurons) is well defined and this parameter can be omitted from the trials in contrast to other works [13, 14]. The target is to identify each local transfer function by means of each neuron. Moreover, the intention is to analyze other parameters such as the sampling time and the number of training epochs.

The second objective of this work is to identify a nonlinear plant composed of tanks that simulate an industrial process of transport and storage of liquids. The determination of the system order, its subsequent identification and the control of the plant were carried out in this paper.

An exhaustive description to tune the training parameters of the NG is outlined in Sect. 2. The local linear model of the NG is presented as an ARX model in Sect. 3. Section 4 is focused on the experimental testing using the simplest dynamic systems. The plant to be controlled is presented in Sect. 5. Section 6 deals with the estimation of the plant order and optimal clustering distribution of the data in order to establish the number of neurons and to set the delayed inputs in the neural gas model. The training of the NG and the recurrent NN are carried out in Sect. 7. The results of both controllers are presented in Sect. 8. Finally, the conclusions are presented in Sect. 9.

2 Local linear approach by neural gas

The general equation for the linearization of a multivariable function $f(v)$ at a point v_0 is indicated in (1), where v is the vector of variables, and v_0 is the linearization point of interest.

$$f(v) \approx f(v_0) + \nabla f|_{v_0} \cdot (v - v_0) \tag{1}$$

The first step is to establish the equilibrium points for linearization using the unsupervised batch training procedure. Each linear model is represented by a neuron, likewise each neuron i is associated with a prototype vector w_i, a reference value y_{w_i} and a gradient vector ∇f_i.

The updating rule of w_i is based on energy cost function (2) according to the Euclidean metric. There are more generic formulations [15] where instead of the Euclidean measure, another arbitrary metric is taken which is considered favorable for the algorithm performance.

$$E_{\mathrm{NG}} = \sum_{i=1}^{m} \sum_{j=1}^{N} h_{\sigma(v_j, w_i)} \cdot (v_j - w_i)^2, \tag{2}$$

where m is the number of neurons and N is the number of data samples v.

$$h_\sigma(v, w_i) = \exp\left(-\frac{k(v, w_i)}{\sigma(t)}\right) \tag{3}$$

Equation (3) is the typical neighborhood function with the rank function $k(v, w_i) \in \{0, \ldots, m-1\}$ representing the rank distance between prototype w_i and data vector v. The neighborhood radius $\sigma(t)$ is considered as in (4)

$$\sigma(t) = \sigma_{t_0} \cdot \left(\frac{\sigma_{t_{max}}}{\sigma_{t_0}}\right)^{t/t_{max}} \tag{4}$$

where t is the epoch step, t_{max} is the maximum number of epochs and σ_{t_0} was chosen as half the number of map units ($\sigma_{t_0} = m/2$), as in [16]. In addition, $\sigma_{t_{max}} = 10^{-5}$ in order to minimize the quantization error at the end of the training.

The adaptation of the prototype w_i is formulated in (5).

$$w_i = \frac{\sum_{j=1}^{N} h_\sigma(v_j, w_i) \cdot v_j}{\sum_{j=1}^{N} h_\sigma(v_j, w_i)} \tag{5}$$

More detailed information about the learning rule formation of the unsupervised learning algorithm can be found in [2]. The training data are composed of a set of points of the form $(f(v), v)$ where $f(v)$ is the nonlinear target function to be approximated by means of a vector of variables v. The probability distribution of input data vectors v and their corresponding target function $f(v)$ are represented by prototype vectors w and reference values y_w, respectively. Both w_i and y_{w_i} parameters can be considered as one linearization point of interest. Therefore, there are m local linear models, each of them linearized at the point defined by neuron i with its associated w_i and y_{w_i}.

At this point, Eq. (1) can be particularized, yielding (6), for the case where the linearization point is defined by the winning neuron i^* with its closest w_i to data vector v, i.e., the best matching unit (BMU). The asterisk super index denotes the winning neuron for input data vector v. The

winning neuron i^* corresponding to data vector v is defined as:

$$\begin{aligned}&(v-w_{i^*})^2 \le (v-w_k)^2 \forall k.\\ &f(v) \approx y_{w_{i^*}} + \nabla f_{i^*}\cdot(v-w_{i^*})\end{aligned} \tag{6}$$

The second step is to carry out a supervised learning in order to obtain reference value y_{w_i} and gradient vector ∇f_i, i.e., the D partial derivatives of target function f assigned to each neuron i. D is the dimension of data vector v. The energy cost function in this case is based on the square error of the function approximation [17] according to (7). The learning rules for y_{w_i} and ∇f_i are shown in (8) and (9), respectively. The discrete variant of the supervised batch approach is used here. Note that there is no classification task in this work, so the labels do not represent any class separation. In this case, the "labeling" is represented by the reference values y_w which are numerical variables. As in [18], these labeling variables do not influence the prototype w updates in the discrete version.

$$E_{\text{NGsup}} = \sum_{i=1}^{m}\sum_{j=1}^{N} h_{\sigma(v_j,w_i)}\cdot\left(f(v_j)-\hat{f}(v_j)\right)^2 \tag{7}$$

$$y_{w_i} = \frac{\sum_{j=1}^{N} h_{\sigma(v_j,w_i)}\cdot f(v_j)}{\sum_{j=1}^{N} h_{\sigma(v_j,w_i)}} \tag{8}$$

$$\Delta\nabla f_i = \frac{\sum_{j=1}^{N} h_{\sigma(v_j,w_i)}\cdot q_{ij}\cdot\left(f(v_j)-y_{w_i}-\nabla f_i\cdot q_{ij}\right)}{\sum_{j=1}^{N} h_{\sigma(v_j,w_i)}\cdot q_{ij}\cdot q_{ij}}, \tag{9}$$

where $q_{ij} = v_j - w_i$.

3 General expression of the dynamic model

Once the neural network has been trained, the dynamics of the identified system are modeled as a set of linear subsystems whose output y depends on the previous values of both output y and input u. The nonlinear autoregressive moving average (NARMA) model has been proven for nonlinear identification [19, 20] and can be expressed as

$$y_{k+d} = h(y_k, y_{k-1}, y_{k-2}, \ldots, y_{k-n+1}, u_k, u_{k-1}, u_{k-2}, \ldots, u_{k-n+1}) \tag{10}$$

where y_k is the system output at the sampling instant k, u_k is the system input at instant k, and d is the system delay. A dot product appears in (6) that can be expressed by a sum of products of the components of the data vector v and the winning prototype vector w_{i^*}. Obviously, these components are the same as indicated in (10). Considering zero delay system and substituting (10) for (6) leaves

$$\begin{aligned}y_k =& y_{w_{i^*}} + \nabla f_{i^*,1}(y_{k-1}-w_{i^*,1})\\ &+ \nabla f_{i^*,2}(y_{k-2}-w_{i^*,2}) + \cdots\\ &+ \nabla f_{i^*,n}(y_{k-n}-w_{i^*,n}) + \nabla f_{i^*,n+1}(u_k - w_{i^*,n+1})\\ &+ \nabla f_{i^*,n+2}(u_{k-1}-w_{i^*,n+2}) + \cdots\\ &+ \nabla f_{i^*,2n+1}(u_{k-n}-w_{i^*,2n+1}).\end{aligned} \tag{11}$$

The gradients are denoted as coefficients a_i when they are referred to output y_{k-i}, likewise the gradients that correspond to input u_{k-i} are denoted by b_i.

$\nabla f_{i^*,1} = a_1$, $\nabla f_{i^*,2} = a_2$, $\ldots\nabla f_{i^*,n} = a_n$ $\nabla f_{i^*,n+1} = b_0$, $\nabla f_{i^*,n+2} = b_1$, $\cdots\nabla f_{i^*,2n+1} = b_n$

and the following terms can be gathered to form variable η

$$\eta_k = y_{i^*} - \sum_{j=1}^{2n+1}\nabla f_{i^*,j}\cdot w_{i^*,j} \tag{12}$$

Denoting the polynomials with backward shift operator z^{-1} by $A(z^{-1}) = 1 - a_1 z^{-1} - a_2 z^{-2} - \ldots - a_n z^{-n}$ and $B(z^{-1}) = b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_n z^{-n}$

$$Y(z) = \frac{B(z^{-1})}{A(z^{-1})}U(z) + \frac{1}{A(z^{-1})}\eta(z)$$

which is similar to an ARX model, η is not only a zero mean independent identically distributed white noise process but also a known disturbance calculated according to (12) and it depends on the input and output values since it is obtained by BMU i^*. The internal noise of the system can be included in η. Using the z–transform, $Y(z)$ is the system output and $U(z)$ is the system input.

4 Experimental testing

4.1 Two first-order subsystems

Firstly, the proposed identification technique is tested with the simplest dynamic model: the first-order system. Figure 1 represents a simple nonlinear plant composed of two first-order systems as local linear subsystems. Linearity in both subsystems is represented by its continuous transfer function. The time constants are $T_1 = 0.1$ and $T_2 = 0.2$ s. The input signal was a square wave with random amplitude and a pulse width of 0.5 s. The switches are responsible for routing the input signal $u(t)$ to meet the constraints described in (13).

$$y(t) = \begin{cases}\mathcal{L}^{-1}\{G_1(s)\cdot U(s)\} & \text{if } 0<u(t)<1.5\\ \mathcal{L}^{-1}\{G_2(s)\cdot U(s)\} & \text{if } 1.5<u(t)<3\end{cases} \tag{13}$$

where $\mathcal{L}\{u(t)\} = U(s)$. The training data are expected to be two well-differentiated clusters in which the identification technique is able to classify each sample with its local

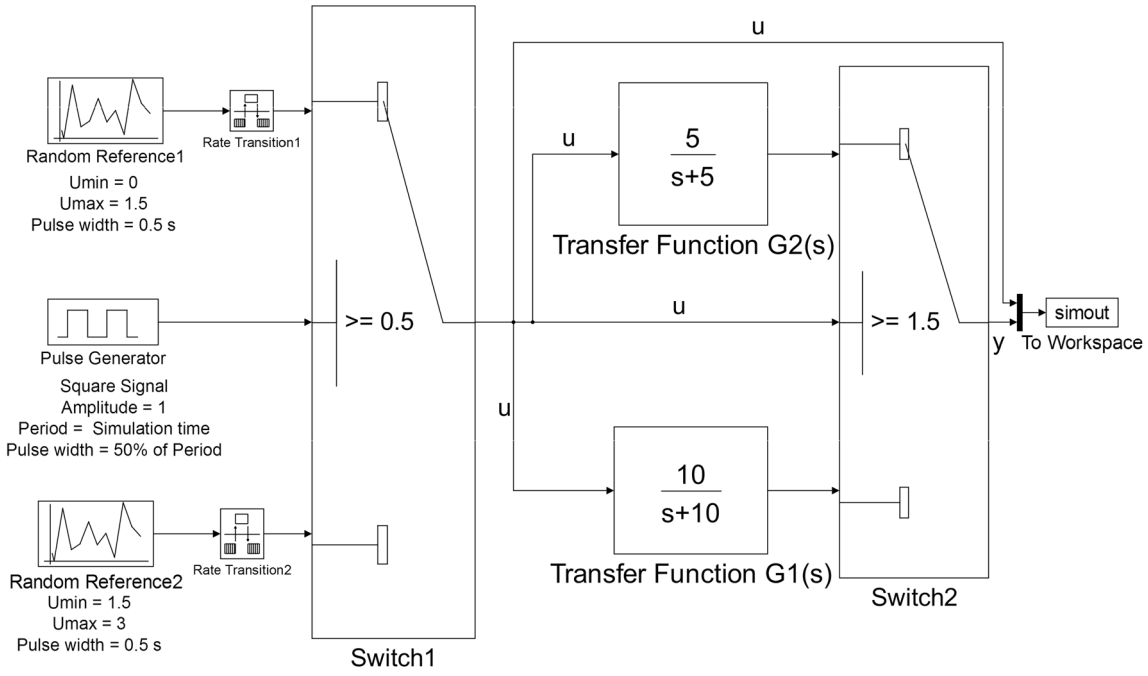


Fig. 1 First-order subsystems

linear model. Only 2 neurons ($i = 1, 2$) are used, each of which is employed to model the corresponding transfer function. Each local transfer function is obtained by means of the training data.

In order to obtain the training data, the simulation time was equal to 200 s and the sampling time T_s ranged between 0.05 and 0.01 s. Two different training datasets were formed, one for $T_s = 0.05$ and another for $T_s = 0.01$. The sampling time was chosen around $T_s = 0.1 \cdot \pi \cdot T$ that comes from the linear system theory recommendation [21] in which the sampling rate must be approximately 20 times the crossover frequency of the open-loop system.

To determine the discrete transfer function of the plant to be identified, in this case the plant is a continuous transfer function, the digitization method to be used is the matched pole–zero (MPZ). The MPZ method uses the z-transform according to $z = e^{sT_s}$, and it adds zeros at $z = -1$ in order to match the order of the numerator with that of the denominator. In addition, the low-frequency gain must be the same. Therefore, the two discrete transfer functions that must be identified considering $T_s = 0.05$ s are:

$$G_1(z) = \frac{0.3935}{z - 0.6065}; G_2(z) = \frac{0.2212}{z - 0.7788}$$

If $T_s = 0.01s$, the two discrete transfer functions are:

$$G_1(z) = \frac{0.09516}{z - 0.9048}; G_2(z) = \frac{0.0488}{z - 0.9512}$$

The coefficients of these transfer functions can be compared to the results shown in Tables 1 and 2 in order to check the correct identification of the plant. Six sets of trained models were formed for different values of sampling rate and number of epochs. Each set contains ten models trained with the same data and the same values for sampling rate and number of epochs. Ten different sets of test data were used to check the algorithm performance. The function approximation is done by each of the ten models obtained in the six different sets. Each set of models yields the same results under different test data. The same gradient vectors are obtained. The results are totally deterministic, and the convergence toward the only solution is irrefutable. The mean squared error (MSE) was calculated for each case. The mean value μ and the standard deviation σ were obtained for each of the six sets of models. Tables 3 and 4 show the results.

Tables 1 and 2 display information about the prototype vectors, the gradient vectors and the reference values that constitute the parameters of the obtained local models. The results are the mean value and the standard deviation for the different sets of models obtained by varying the sampling rate and the number of training epochs. The parameters show very low values of standard deviation, which are almost equal to zero, that confirm the almost total convergence of the algorithm. The dispersion of the results is even smaller with $T_s = 0.01$ s, and the parameters related to the linearization points are always equal. For this reason, the standard deviation is omitted and only the mean value is shown in Tables 1 and 2.

According to the results related to the first-order subsystems shown in Table 3, increasing notably the number

Table 1 First-order and sampling time = 0.05 s

	100 epochs		500 epochs		1000 epochs	
	Neuron $i = 1$	Neuron $i = 2$	Neuron $i = 1$	Neuron $i = 2$	Neuron $i = 1$	Neuron $i = 2$
a_1	0.6132	0.7786	0.6132	0.7786	0.6132	0.7786
b_1	0.3865	0.2214	0.3865	0.2214	0.3865	0.2214
y_{w_i}	0.7776	2.2992	0.7776	2.2992	0.7776	2.2992
$w_{i,1}$	0.7773	2.2990	0.7773	2.2990	0.7773	2.2990
$w_{i,2}$	0.7790	2.3001	0.7790	2.3001	0.7790	2.3001

Table 2 First-order and sampling time = 0.01 s

	100 epochs		500 epochs		1000 epochs	
	Neuron $i = 1$	Neuron $i = 2$	Neuron $i = 1$	Neuron $i = 2$	Neuron $i = 1$	Neuron $i = 2$
a_1	0.9036	0.9512	0.9048	0.9512	0.9048	0.9512
b_1	0.0962	0.0488	0.0951	0.0488	0.0951	0.0488
y_{w_i}	0.7346	2.2562	0.7346	2.2562	0.7346	2.2562
$w_{i,1}$	0.7346	2.2561	0.7346	2.2561	0.7346	2.2561
$w_{i,2}$	0.7348	2.2580	0.7348	2.2580	0.7348	2.2580

Table 3 MSE as a function of number of epochs; first-order subsystems; sampling time = 0.05 s

Total epochs	Mean squared error	
	μ	σ
100	4.519×10^{-5}	3.081×10^{-5}
500	4.521×10^{-5}	3.080×10^{-5}
1000	4.521×10^{-5}	3.080×10^{-5}

Table 4 MSE as a function of number of epochs; first-order subsystems; sampling time = 0.01 s

Total epochs	Mean squared error	
	μ	σ
100	2.778×10^{-5}	3.140×10^{-5}
500	2.640×10^{-5}	3.010×10^{-5}
1000	2.640×10^{-5}	3.010×10^{-5}

of epochs does not lead to an improvement of the MSE. As the sampling time decreases, the number of epochs is relevant. If the sampling time decreases, the number of epochs will usually need to be increased.

4.2 Two second-order subsystems

In the same way as for Sect. 4.1, two linear subsystems were considered for identification with two well-differentiated input data ranges, according to the constraints described in (14).

$$y(t) = \begin{cases} \mathcal{L}^{-1}\{G_1(s) \cdot U(s)\} & \text{if } 0.5 < u(t) < 2 \\ \mathcal{L}^{-1}\{G_2(s) \cdot U(s)\} & \text{if } -2 < u(t) < -0.5 \end{cases} \tag{14}$$

where $\mathcal{L}\{u(t)\} = U(s)$. The NG algorithm is expected to model this nonlinear plant with only two neurons. There is one-to-one correspondence between the neurons and the transfer functions. The linear subsystems are second order, and their continuous transfer functions are indicated in (15) and (16). The natural frequency of G_1 ($\omega_n = 10$ rad/s) is twice that of G_2. Therefore, G_1 has a higher bandwidth and a faster response than G_2. In addition, both subsystems have low damping factors in order to test the algorithm for identifying transient responses with high overshoot.

$$G_1(s) = \frac{100}{s^2 + 2s + 100} \tag{15}$$

$$G_2(s) = \frac{25}{s^2 + 2s + 25} \tag{16}$$

The simulation time was equal to 200 s, and the sampling time ranged from 0.05 to 0.025 s. Two different training datasets were formed, one for $T_s = 0.05$ and the other for $T_s = 0.025$. The sampling time was chosen around $T_s = 0.1 \cdot \pi/\omega_n$ for the same reason as in Sect. 4.1. Aiming to check the results of the plant dynamics identification, the discrete transfer functions obtained by the MPZ method are (17) when considering sampling time $T_s = 0.025$ s.

$$G_1(z) = \frac{0.0303(z+1)}{z^2 - 1.8906z + 0.9512}$$
$$G_2(z) = \frac{0.0076(z+1)}{z^2 - 1.9360z + 0.9512} \quad (17)$$

If sampling time T_s is equal to 0.05 s, then the discrete transfer functions of the plant are (18).

$$G_1(z) = \frac{0.1165(z+1)}{z^2 - 1.6719z + 0.9048}$$
$$G_2(z) = \frac{0.0296(z+1)}{z^2 - 1.8457z + 0.9048} \quad (18)$$

Table 5 shows the results of MSE as a function of the number of epochs using a sample time equal to 0.05s. Fifty testing datasets were used to calculate the mean value μ and the standard deviation σ of these simulations.

Table 7 indicates the gradient vectors for $T_s = 0.05$ s. The results for 4000 and 6000 epochs were omitted since their values are the same as in cases 3000 and 8000 epochs. The MSE and the gradient vectors are always the same above 3000 epochs, so there is no improvement when increasing the number of epochs in this case.

For $T_s = 0.025$, the MSE of Table 6 is mostly due to the switching between both local models at time $t = 100$ s, see Fig. 2. However, the model trained with $T_s = 0.05$ performs better at this switching time and, for this reason, the MSE of Table 5 seems better, but the dynamics plant has been modeled worse than that for $T_s = 0.025$, as shown in Fig. 3. The number of epochs seems to be an important training parameter especially if the sampling period is decreased. The higher the number of epochs, the better the dynamics identification. In all cases, the prototype vectors w and the reference values y_w converge quickly toward the same value regardless of the number of epochs. Therefore, there is a fast convergence in the task of obtaining the point of linearization. In addition, the dynamics identification, related to gradient vectors ∇f, tends to worsen for G_2 which is the subsystem with lower bandwidth or slowest transient response. There is no need to use data from the steady state. The identification is correct by means of using transient responses. For this reason, the pulse width of the random references was equal to 0.5 s (Tables 7, 8).

Table 5 MSE as a function of number of epochs; second-order subsystems; sampling time = 0.05 s

Total epochs	Mean squared error	
	μ	σ
1000	0.0043	0.0042
2000	0.0045	0.0052
3000	0.0045	0.0052
4000	0.0045	0.0052
6000	0.0045	0.0052
8000	0.0045	0.0052

Table 6 MSE as a function of number of epochs; second-order subsystems; sampling time = 0.025 s

Total epochs	Mean squared error	
	μ	σ
1000	0.0837	0.0148
2000	0.0230	0.0042
3000	0.0069	0.0029
4000	0.0049	0.0046
6000	0.0058	0.0062
8000	0.0060	0.0064

5 Description of the plant

The plant to be identified is located in the Department of Systems Engineering and Automation of the University of Oviedo. It is designed to simulate an industrial process of transport and storage of liquids (see Fig. 4a). The model plant is controlled by a programmable logic controller (PLC) connected to an OPC (OLE for Process Control) server. It consists of two lower tanks (D1 and D2) and two upper tanks (D3 and D4), which work through a cascade drain system. The lower tanks discharge their flow to a collector tank, which acts as a source for the pumping to each of the tanks by means of two pumps (B1 and B2) and two three-way valves (V1 and V2). The proposed work has the aim of identifying the operation mode named as 2×2, depicted in Figs. 4b and 5b, in which the pump B1 supplies flow to D1 and D4 tanks, proportionally distributed due to the three-way valve V1. If the setpoint is equal to 0%, all the liquid is pumped to tank D1; whereas if the setpoint is equal to 100%, then this implies that all the liquid is pumped to tank D4. Both pumps have maximum and minimum flow rates, and the capacity of the tanks is limited between top and lower levels, oscillating continuously due to the uninterrupted supply of liquid and the three fluid outlets of each tank: lower outlet, upper outlet (established as a relief) and controllable outlet due to solenoid on/off valves, which are named as LV. The lower outputs are always open. The controllable outlets by solenoid are closed for the 2×2 operation mode. The opening percentage of the three-way valve $s(t)$ and the pump power $p(t)$ are considered inputs of the system, while liquid levels in both tanks $y_1(t)$ and $y_4(t)$ are established as outputs. All these variables oscillate between 0% and 100%, and they are considered as a multiple-input multiple-output (MIMO)

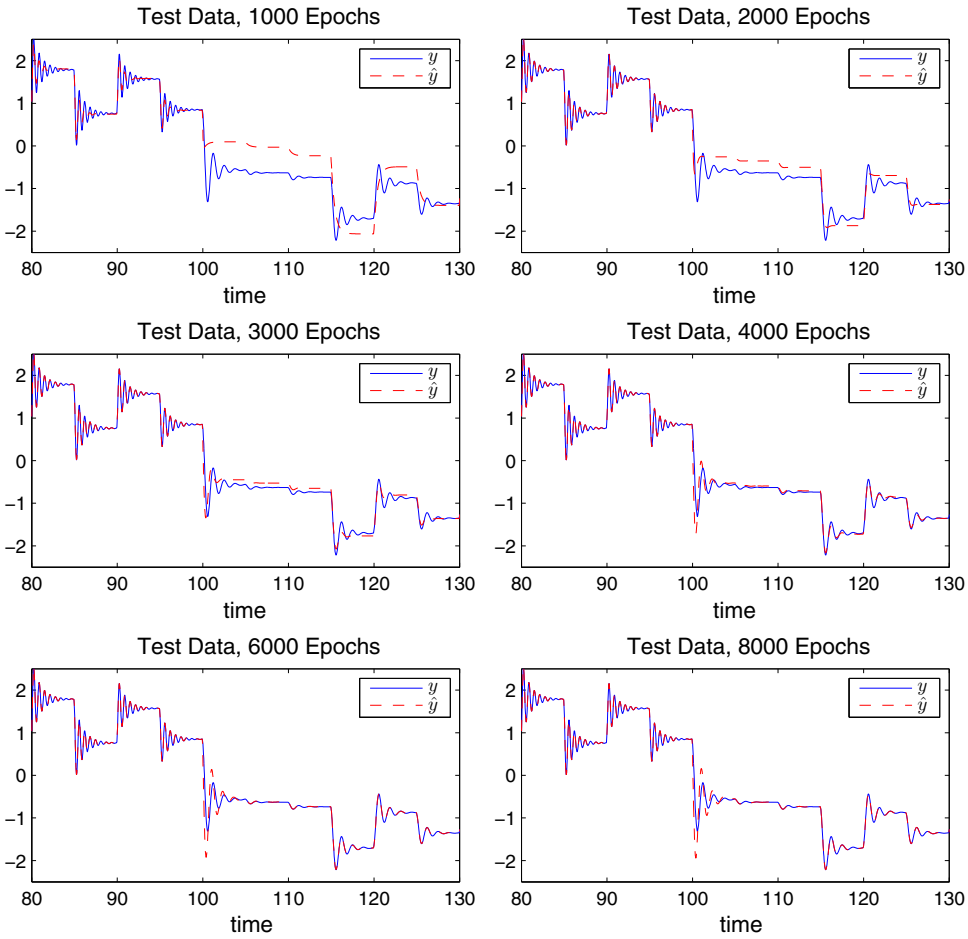


Fig. 2 Testing data versus estimated output using $T_s = 0.025\,\mathrm{s}$

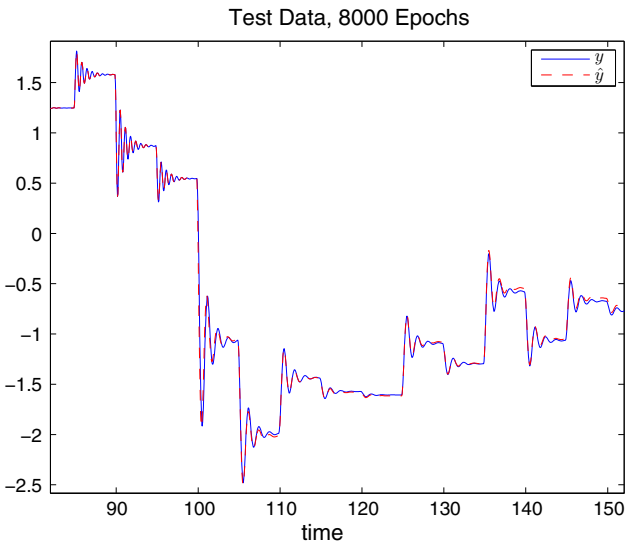


Fig. 3 Testing data versus estimated output using $T_s = 0.05\,\mathrm{s}$

system. Such identification of the 2×2 operation of the plant has been carried out using a supervised neural gas algorithm, employing its batch version.

In order to explain the controlling problem, suppose a single water deposit in Fig. 5a, the fill level of which is $y(t)$ at a certain instant t. Supposing that input flow $q(t)$ is constant, it is possible to control this level $y(t)$ by means of manipulating the valve section $s(t)$ so that $y(t)$ tracks a desired fill level $y^*(t)$ known as reference. The idea is to design a control algorithm able to provide the suitable signal (control action signal) to manipulate the valve section $s(t)$. This simple controlling example constitutes a single-input single-output (SISO) system where the input $s(t)$ influences the output $y(t)$. At this time, suppose Fig. 5b, the controlled variables are $y_1(t)$ and $y_4(t)$ and the input variables are the pump power $p(t)$ and the valve section $s(t)$. This constitutes a coupled MIMO system where both inputs $p(t)$ and $s(t)$ have simultaneous influence on both outputs $y_1(t)$ and $y_4(t)$. A classical SISO system approach is not possible in this case. There is also a nonlinear relationship between the tank levels, $y_1(t)$ and $y_4(t)$, and the output flows, $q_{s1}(t)$ and $q_{s4}(t)$, according to Torricelli's law. In this case, the control algorithm must provide two suitable control actions: one for pump power $p(t)$ and another for valve section $s(t)$.

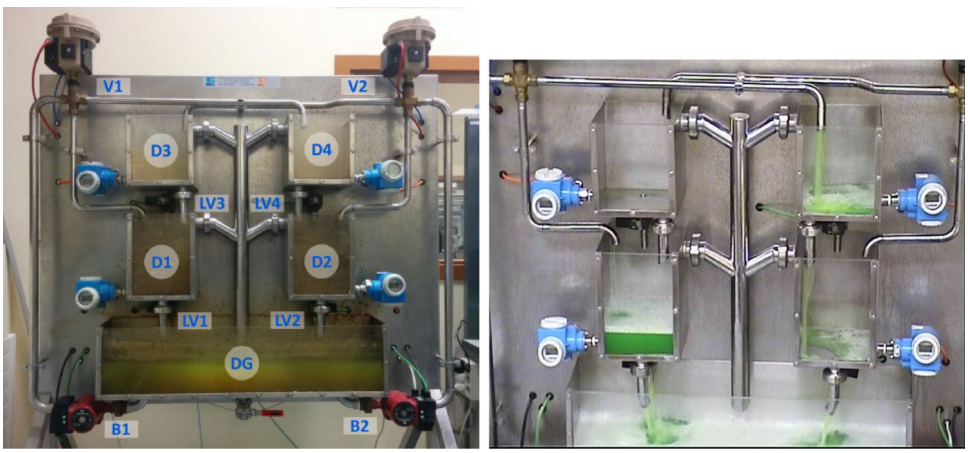
Table 7 Second order and sampling time = 0.05 s

	Plant		2000 epochs		3000 epochs		8000 epochs	
	$G_1(z)$ $i=1$	$G_2(z)$ $i=2$	Neuron $i=1$	Neuron $i=2$	Neuron $i=1$	Neuron $i=2$	Neuron $i=1$	Neuron $i=2$
$a_{i,1}$	1.6719	1.8457	1.6673	1.8380	1.6673	1.8382	1.6673	1.8382
$a_{i,2}$	− 0.9048	− 0.9048	− 0.9000	− 0.8979	− 0.9000	− 0.8982	− 0.9000	− 0.8982
$b_{i,1}$	0.1165	0.0296	0.1208	0.0310	0.1208	0.0310	0.1208	0.0310
$b_{i,2}$	0.1165	0.0296	0.1118	0.0314	0.1118	0.0314	0.1118	0.0314
y_{w_i}	–	–	1.2508	− 1.2486	1.2508	− 1.2486	1.2508	− 1.2486
$w_{i,1}$	–	–	1.2507	− 1.2479	1.2507	− 1.2479	1.2507	− 1.2479
$w_{i,2}$	–	–	1.2503	− 1.2471	1.2503	− 1.2471	1.2503	− 1.2471
$w_{i,3}$	–	–	1.2493	− 1.2496	1.2493	− 1.2496	1.2493	− 1.2496
$w_{i,4}$	–	–	1.2497	− 1.2493	1.2497	− 1.2493	1.2497	− 1.2493

Table 8 Second order and sampling time = 0.025 s

	Plant		4000 epochs		6000 epochs		8000 epochs	
	$G_1(z)$ $i=1$	$G_2(z)$ $i=2$	Neuron $i=1$	Neuron $i=2$	Neuron $i=1$	Neuron $i=2$	Neuron $i=1$	Neuron $i=2$
$a_{i,1}$	1.8906	1.9360	1.8903	1.9207	1.8903	1.9344	1.8903	1.9358
$a_{i,2}$	− 0.9512	− 0.9512	− 0.9509	− 0.9364	− 0.9509	− 0.9497	− 0.9509	− 0.9511
$b_{i,1}$	0.0303	0.0076	0.0307	0.0075	0.0307	0.0077	0.0307	0.0077
$b_{i,2}$	0.0303	0.0076	0.0299	0.0090	0.0299	0.0077	0.0299	0.0076
y_{w_i}	–	–	1.1939	− 1.3058	1.1939	− 1.3058	1.1939	− 1.3058
$w_{i,1}$	–	–	1.1938	− 1.3053	1.1938	− 1.3053	1.1938	− 1.3053
$w_{i,2}$	–	–	1.1938	− 1.3048	1.1938	− 1.3048	1.1938	− 1.3048
$w_{i,3}$	–	–	1.1938	− 1.3062	1.1938	− 1.3062	1.1938	− 1.3062
$w_{i,4}$	–	–	1.1939	− 1.3061	1.1939	− 1.3061	1.1939	− 1.3061

Fig. 4 Plant to be controlled

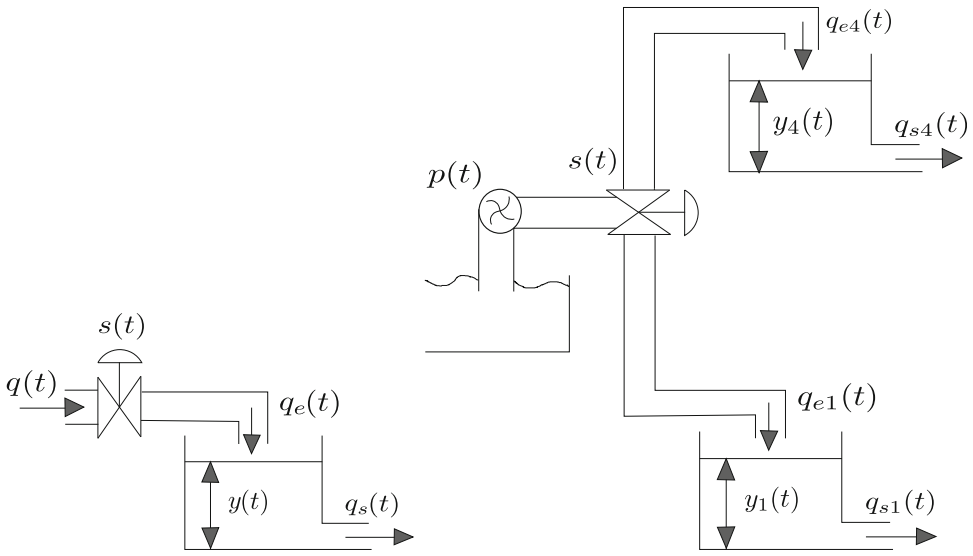


(a) Frontal view of the plant

(b) 2x2 operational mode in real working

After obtaining empirically a dataset that provides reliable information on the behavior of the system, a preliminary study was carried out to determine the structure and optimal order of the model that are used for the system identification. The results of the study were used as the starting point to carry out the identification process, beginning with obtaining the optimal number of clusters for data distribution using the unsupervised version of the

Fig. 5 Plant schemes



(a) SISO system as an early example

(b) 2x2 operation scheme

algorithm. On the other hand, a supervised version of neural gas batch algorithm was employed to associate the linear local models with each partition of the data, which characterize the operation of the plant in its surroundings and allow us to control it by means of a control strategy based on switching local linear models according to the current conditions of the plant operation.

6 Plant identification

6.1 Determining the system order

For the purpose of carrying out the data collection experiments, the sampling time was established equal to one second. The variation of the input variables was chosen considering the plant as a first-order system with a large constant time. The inputs were set by means of large and long-lasting steps, allowing to visualize complete evolutions of the outputs. The experiments were carried out with restricted ranges of variation around the operation point established for input variables, in order to characterize the different areas of linear performance and to avoid nonlinear areas due to full and empty tank states. The data samples collected in the experiment were divided into training and validation sets in order to choose the most appropriate model, thereby providing a large dataset concerning the inputs and outputs of the system. Both training and validation datasets have the same origin and characteristics, and contain reliable information of all the operating modes of the system. Thus, evaluating the parametric impulse response to the system modeled initially as a state space of order 2, i.e., with two system state variables $x_1(t)$ and $x_2(t)$, it can be pointed out, as shown in Fig. 6, that the impulse response is not established until the second sample for all the combinations of input and output, so the delay in entering the system is of one sample.

Observing the results of Table 9, which includes the fit percentage to validation data of several preliminary models established with different orders and structures by carrying out initial identifications of the training data, all of them follow the validation data relatively well, so that the simplest possible model is chosen: state space model of order 2 with one-unit delay. Moreover, observing the pole–zero map of such model (Figs. 7, 8), it is possible to cancel a pole–zero pair on each of the input–output combinations due to the proximity between them, which is smaller than the confidence interval established as three times the standard deviation of the data. It can be confirmed that the optimal system would consist of 2 state variables. In addition, each of the relationships between input and output would be composed of one pole, i.e., a first-order system.

6.2 Optimal clustering distribution

An optimal distribution of the data in clusters must be set as a previous step to carry out the identification of the plant. Each cluster is associated with a prototype w_i and its i-th local linear model. Considering that the level of both tanks is independent, each output can be treated individually as a first-order system; they depend on their own output values and both input variables at previous instant $k-1$, which will form the data vector v for each target

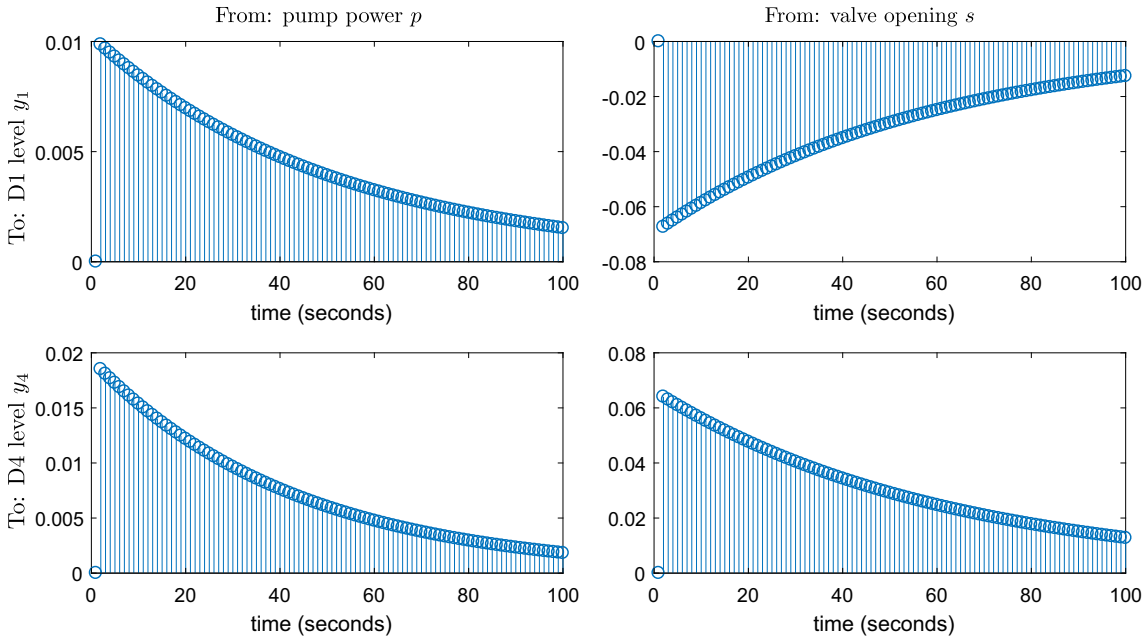


Fig. 6 Parametric impulse response to state space model of order 2

Table 9 Fit % to validation data of several preliminary model configurations

Preliminary model	Fit (%)
State space model (order 2, delay 2)	85.1
State space model (order 3, delay 2)	80.86
State space model (order 2, delay 1)	84.4
State space model (order 3, delay 1)	85.13
ARX ($n_a = 1$, $n_b = 2$)	84,42

variable $f(v)$ as output variable. The optimal number of clusters is obtained by comparing the Davies–Bouldin index for each output obtained for several distributions of the data (ranging from 2 to 20 clusters) by means of applying k-means and unsupervised neural gas batch algorithms to plant data for each output. In order to obtain the optimal prototype distribution, the unsupervised neural gas algorithm applies the learning rule formulated in (5) to training data composed of both the data vector v and the target variable $f(v)$. The Davies–Bouldin index is based on the measurement of the dispersion of data within a cluster with respect to the distance between the centroids of the clusters themselves. For that purpose, the parameter s_i in (19) measures the dispersion of all the data x belonging to the cluster C_i. This cluster is defined by its centroid which is the prototype vector w_i. The parameter d_{ij} in (20) expresses the distance between two clusters C_i and C_j represented by their centroids w_i and w_j, respectively. The parameter R_i in (21) measures such maximum similarity between each cluster and the rest, so that, when looking for distributions with the minimum similarity and maximum distance between partitions, the optimal distribution of partitions is that which minimizes this Davies–Bouldin index, formulated as an average concerning the total number of clusters in (22). Figure 9 shows such Davies–Bouldin indices for the level of the deposits D1 and D4. It can be observed that the distribution of the data split into 4, 5 or 6 clusters minimizes the value of this index regardless of the training method used.

$$s_i = \frac{1}{C_i}\sum_{x \in C_i}\{\|x - w_i\|\} \tag{19}$$

$$d_{ij} = \|w_i - w_j\| \tag{20}$$

$$R_i = \max_{i=1,\ldots,n_c, i\neq j}\left\{\frac{s_i + s_j}{d_{ij}}\right\} \tag{21}$$

$$DB_{n_c} = \frac{1}{n_c}\sum_{i=1}^{n_c} R_i \tag{22}$$

The identification of both outputs, y_1 and y_4, is carried out individually, applying the supervised batch version of the neural gas algorithm, according to learning rules (5), (8) and (9), to optimal cluster distributions (i.e., with a number of prototypes ranging from 4 to 6 neurons) and comparing the fit of the model obtained for each validation dataset. The training variables are the output and the two inputs, all of which are registered at the previous instant, while the output at the current time is the target variable. Therefore, the data vector is $v_j = [y_{k-1}, p_{k-1}, s_{k-1}]$. To choose the final model of the system, the mean squared error (MSE) of the estimation is taken into account for each

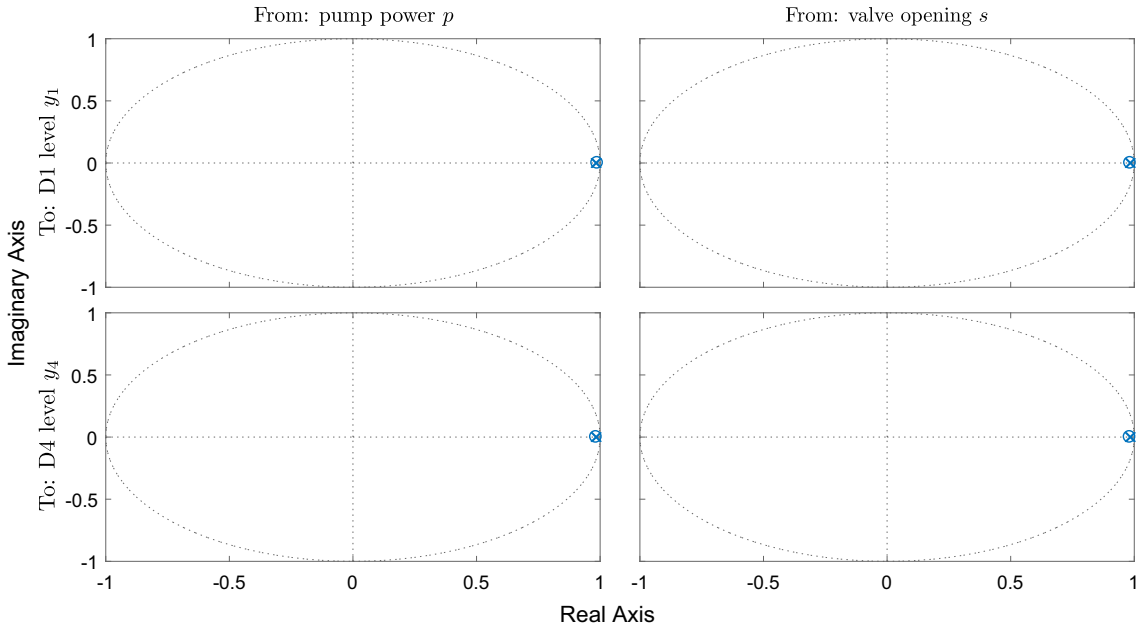
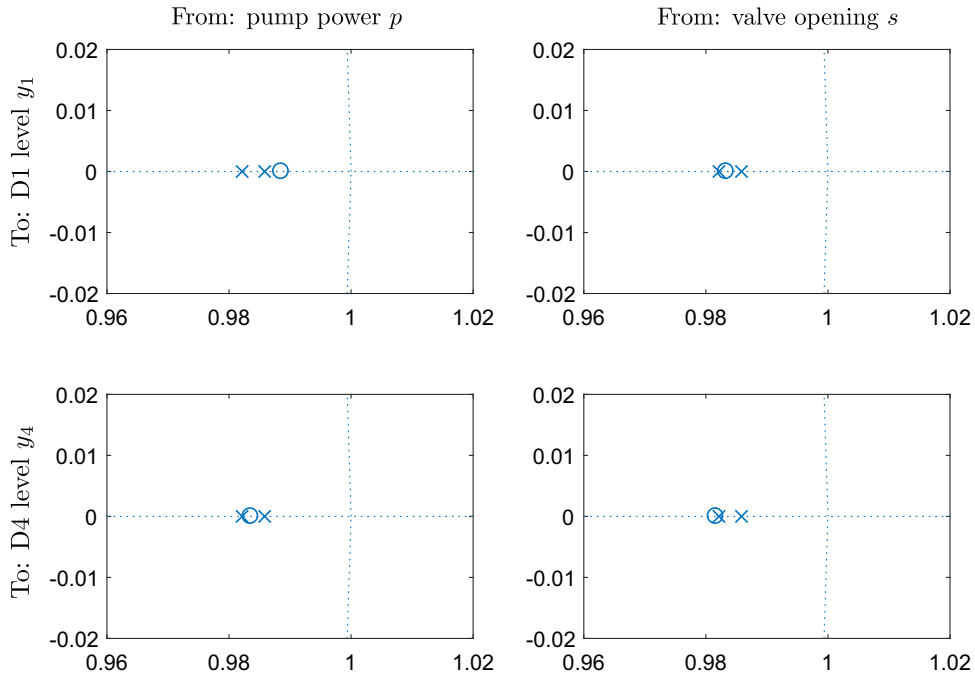


Fig. 7 Pole–zero diagram of space state (order 2, delay 1) model

Fig. 8 Pole–zero diagram of space state (order 2, delay 1) model. Detailed view of Fig. 7



output and for both the training and validation datasets, searching for the model that generates the minimum error. Table 10 shows the training and validation results.

The models obtained by means of the identification follow very accurately the training data as well as the validation data for both outputs, with virtually no difference regarding MSE estimation errors. Thereby, the choice of the final model is based on the Davies–Bouldin index, which is minimal for the distribution of 5 neurons.

7 Plant model

7.1 Plant model using supervised neural gas

Taking into account the results obtained in the previous Section, the plant model is established as a set of five MIMO linear local models in which both outputs y_1 and y_4 depend on the previous values of the output itself and the inputs, expressed in percentage values: $y_k = f(y_{k-1}, p_{k-1}, s_{k-1})$. In

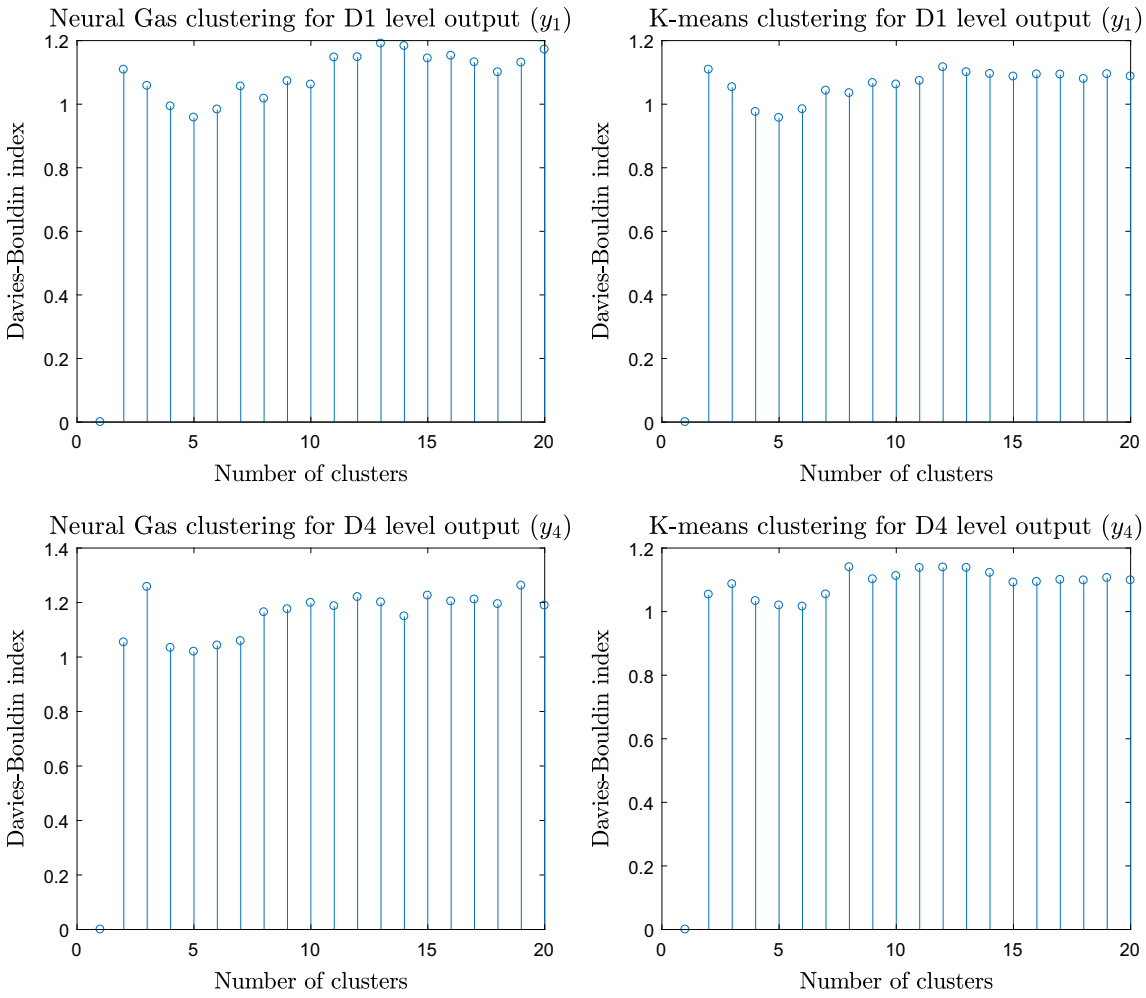


Fig. 9 Davis-Bouldin index for D1 level output (y_1) and D4 level output (y_4) using neural gas and *K*-means as clustering techniques

Table 10 MSE estimation error for D1 level output (y_1) and D4 level output (y_4) using training and validation data

Data	Number of clusters		
	4	5	6
Training data (y_1)	0.1102	0.1101	0.1097
Validation data (y_1)	0.1170	0.1192	0.1188
Training data (y_4)	0.0480	0.0472	0.0462
Validation data (y_4)	0.0549	0.0550	0.0542

this paper, the distribution of the local linear models is done according to the unsupervised version of the neural gas algorithm by means of the prototype vectors w. Therefore, the number of neurons is determined based on similarity clustering of input vectors. Some works claim that this approach may not necessarily improve the network performance in the nonlinear prediction and improve the overall network performance using the supervised learning method according to the measurement of the error estimation [22]. Every single linear local region characterizes the behavior of the system when its associated neuron acts as BMU to the data vector v constituted by the training variables y_{k-1}, p_{k-1} and s_{k-1}. The linear estimation employed for functions y_1 and y_4 arises from the identification by supervised neural gas, trained for 2000 epochs, where $\nabla f_{i^*,j}$, is the gradient of the approximated function obtained in the i-th Voronoi region defined by prototype $w_{i^*,j}$, in which i^* represents the winning neuron and j the prototype component (associated with each training variable) and y_{i^*} is the reference value learned for each neuron. Once the plant model is obtained as in (11), z-transformation can be applied and a set of transfer functions for each pair of variables can be considered, in which $\eta(z)$ represents the disturbance in (12).

$$Y(z) = \frac{\nabla f_{i^*,2} \cdot z^{-1}}{1 - \nabla f_{i^*,1} \cdot z^{-1}} P(z) + \frac{\nabla f_{i^*,3} \cdot z^{-1}}{1 - \nabla f_{i^*,1} \cdot z^{-1}} S(z) + \frac{1}{1 - \nabla f_{i^*,1} \cdot z^{-1}} \eta(z) \tag{23}$$

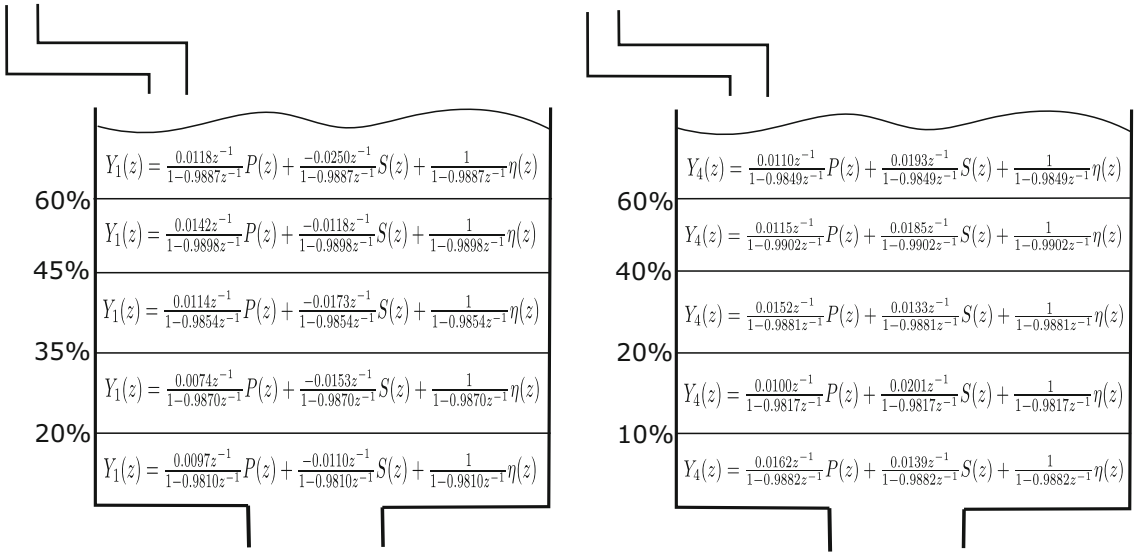


Fig. 10 Transfer function systems for D1 and D4 tanks

Figure 10 shows the set of local linear models obtained after the identification of the plant, expressed as transfer function systems. From Eq. (23) for each output, the following state space model can be obtained by matching the state variables with the outputs, so that these state variables can be measured, and by applying the superposition principle in order to take disturbance η outside the model. The general mathematical expression of the state space model is:

$$X(k+1) = F \cdot X(k) + G \cdot U(k)$$
$$Y(k) = H \cdot X(k) + J \cdot U(k)$$

In this case,

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = F \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + G \begin{bmatrix} p(k) \\ s(k) \end{bmatrix}$$

where the sytem matrix is F, the input matrix is G, the output matrix is H, and the feedforward matrix is J that in this case is a zero matrix.

$$F = \begin{bmatrix} \nabla f_{1,i^*,1} & 0 \\ 0 & \nabla f_{4,i^*,1} \end{bmatrix}$$
$$G = \begin{bmatrix} \nabla f_{1,i^*,2} & \nabla f_{1,i^*,3} \\ \nabla f_{4,i^*,2} & \nabla f_{4,i^*,3} \end{bmatrix}$$
$$\begin{bmatrix} y_1(k) \\ y_4(k) \end{bmatrix} = H \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + J \begin{bmatrix} p(k) \\ s(k) \end{bmatrix}$$
$$H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$
$$J = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

7.2 Plant model and neural controller using recurrent NNs

There is a common approach where a whole nonlinear neural controller is obtained by means of recurrent networks supervised with a reference model as the dynamic system to be followed. It is also known as model-reference neural controller. The model-reference control architecture has two subnetworks. One subnetwork is the plant model to be controlled, while the other subnetwork is the neural controller.

In this paper, the whole architecture is composed of four layers. The third and fourth layers constitute the plant model subnetwork. The first and second layers make up the controller. The transfer function of the first and third layers is hyperbolic tangent, and the number of neurons in these layers has been determined by means of trial-and-error assays trying to minimize the mean square error of the estimation. The second and fourth layers correspond to the controller outputs and to the plant model outputs, respectively. The transfer function of these layers is log-sigmoid in order to obtain a bounded signal between 0 and 1. Therefore, the bounding requirements of the control action can easily be accomplished. The number of neurons in these layers is 2 since there are two control signals $p(t)$ and $s(t)$ and two controlled variables $y_1(t)$ and $y_2(t)$. Figure 11 illustrates the general procedure.

A NARX network is firstly trained to play the role of the plant model subnetwork. The true plant outputs are available during the training of the plant model, and a series-parallel architecture is considered, also known as open-loop training. There are two main advantages. On the one hand, the inputs to the feedforward network are more

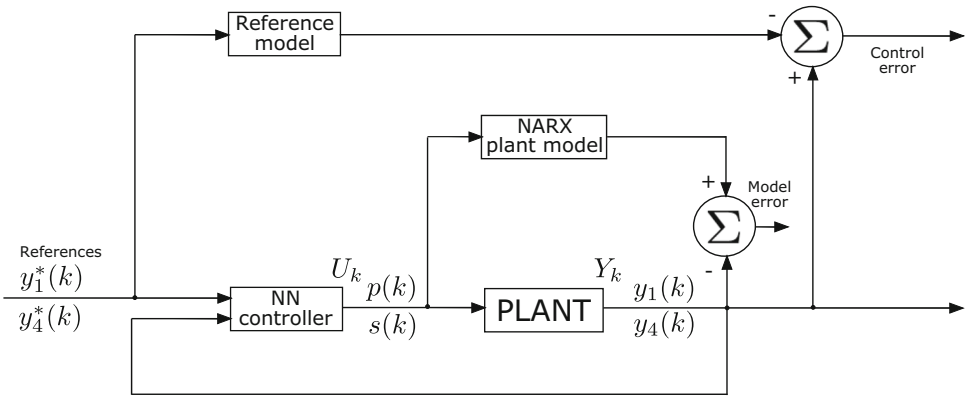
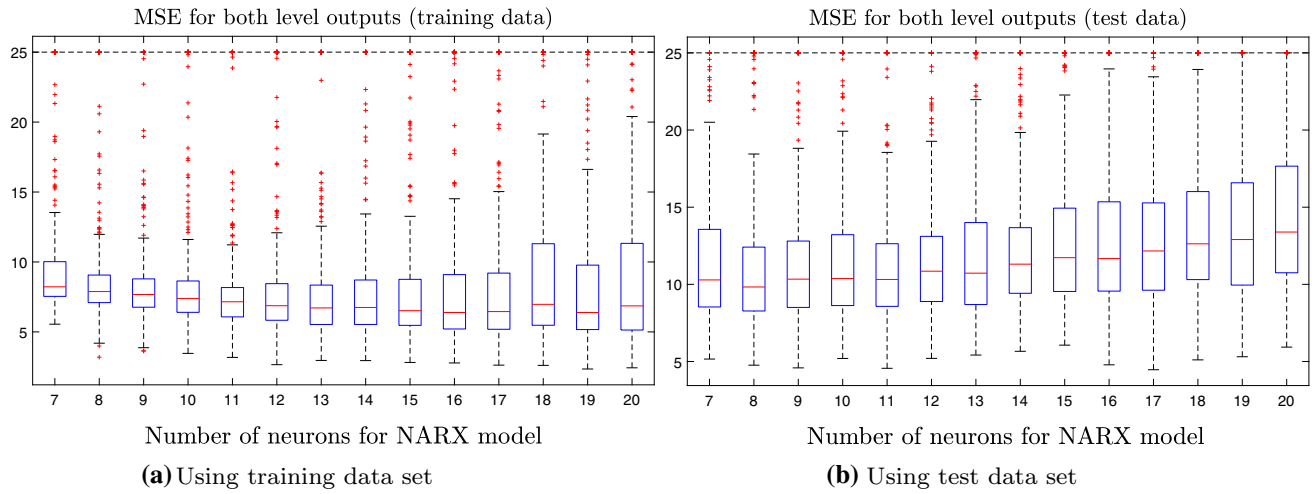


Fig. 11 Model-reference neural controller

accurate, and on the other, basic feedforward architecture with a static backpropagation algorithm can be considered for training. The accuracy obtained with this series-parallel configuration (one-step-ahead prediction) is usually excellent, but the prediction ability must be checked after training with a parallel (closed-loop) configuration. The number of neurons cannot be huge to improve network generalization. It must be just large enough to provide an adequate fit without overfitting the data. Unfortunately, it is difficult to know beforehand how large a network should be. Moreover, each training session starts with different initial weights and biases, and different divisions of data into training, validation and test sets. These diverse initial situations lead to very different results. In this application, a trial-and-error procedure is planned varying the number of neurons in the hidden layer. It is quite common to train several networks to find a good model generalization. The best number of neurons of the hidden layer in the NARX model is 12 neurons, see Fig. 12. Using the same technique, the number of neurons in the hidden layer of the NN controller was set to 7. The early stopping method is used in order to carry out a correct generalization of the recurrent neural network. The data are randomly divided into three subsets where 60% of the samples are employed to the training procedure, 20% to the validation set and 20% to the test data set. The training data set is used for computing the gradient and updating the network weights and biases. The estimation error on the validation set is checked during the training process. During the beginning of the training, the validation error decreases in the same way as the training set error. Later, when the network begins to overfit the data, the validation error usually tends to rise. If the validation error increases for a number of iterations greater than 6, the training is stopped, and the training results correspond to the weights and biases updated at the minimum of the validation error. The plant model was obtained by means of early stopping in all the cases and the training never reached the maximum number of epochs. The test set error is not used during training.



(a) Using training data set

(b) Using test data set

Fig. 12 Distribution error in function of the mean squared error

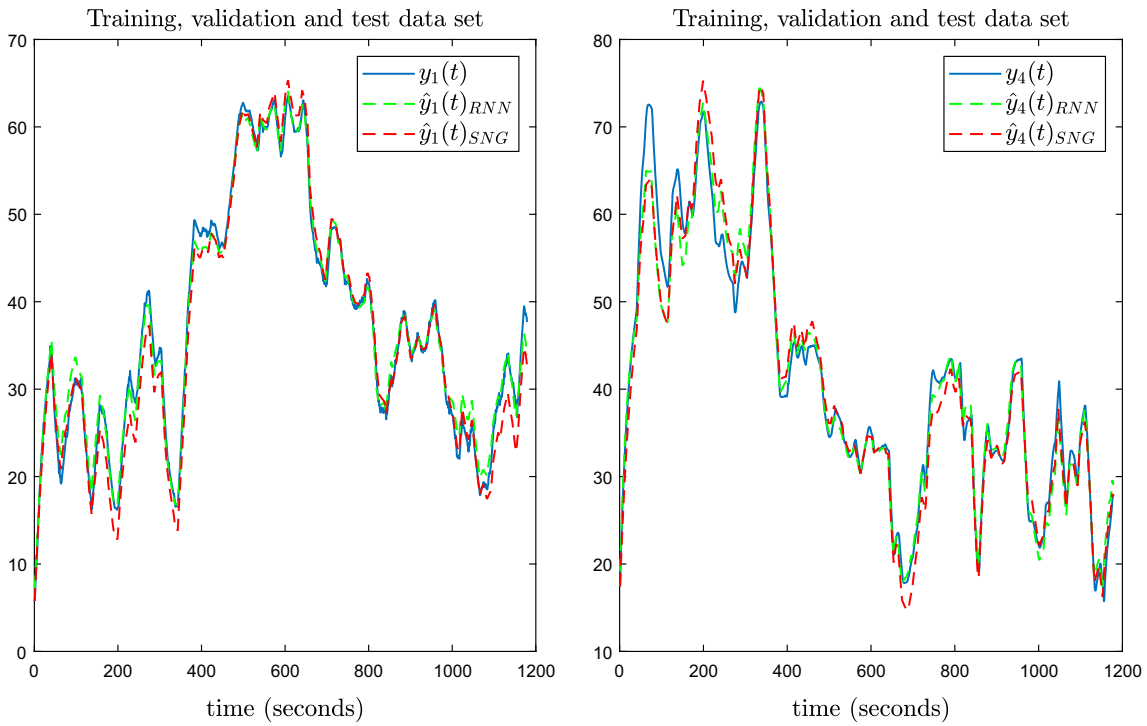


Fig. 13 Simulation of the plant models using *SNG* and *RNN*

When using early stopping, the training algorithm must not converge too quickly. As the Levenberg–Marquardt algorithm is used, the training parameters are set so that the convergence is relatively slow. The μ parameter is assigned to a comparatively large value, such as 1, trying to shift the algorithm toward gradient descent version and to avoid becoming Newton's method. Although the NARX model of the recurrent NN outperforms the estimation of the neural gas model regarding the mean squared error, it must be stated that the training of the NG is much more robust and stable with a very low standard deviation of the estimation error. In Fig. 13, both models were simulated in order to check the correct dynamics identification of the plant using all the data that was employed for training, validation and testing.

After the plan identification by means of the NARX subnet, the neural controller must be obtained using the dynamic backpropagation algorithm. The delays and the feedback connections must be set and the dynamic backpropagation algorithm must only update the weights and bias of the first and second layers because the third and fourth layers were previously defined by the NARX network and must be preserved to model the plant. The network is trained following a first-order system with constant time equal to 40 s for each plant output. The reference values are within 20–40%. Later the controller is tested in the real plant considering this reference range.

8 Plant control

8.1 Proportional control scheme

Figure 14 shows the control loop implemented to perform the proportional control by means of state feedback, using the five local linear models obtained after having carried out the plant identification. The BMU block decides which state space local linear model of the plant is used at each instant k as a function of the proximity of the data vector $v_j = [y_{k-1}, p_{k-1}, s_{k-1}]$ to the prototype vectors w. Using the neural gas model and this control scheme, the state variables and the disturbance term $\eta(k)$ can be obtained in (12) for each output at every sample k. Disturbance $\eta(k)$ can be taken outside the model by applying the superposition principle, subtracting them from the references of each output as disturbances. This result must be multiplied by the inverse of the static gain of the closed chain system $1/\bar{N}$ so that the output of the system follows the reference input in steady state, where $\bar{N} = (H - JK)(F - I - GK)^{-1}G + J$. This signal must be added to that coming from the state feedback $-K \cdot X(k)$ to obtain the control action $U(k)$, which is a column vector containing both inputs, $p(k)$ and $s(k)$, to the plant. The closed-loop poles are located according to the characteristic polynomial in (24).

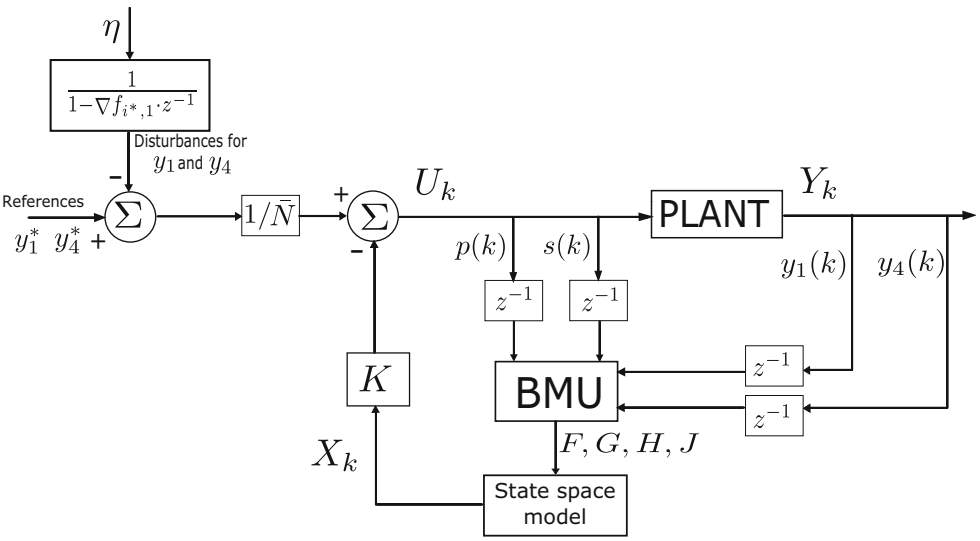


Fig. 14 Control loop implemented to carry out the proportional control

$$p(z) = (z - \lambda_1)(z - \lambda_2) \tag{24}$$

Gain matrix K is necessary to control the system, and it is obtained in (25) using the pole assignment method for MIMO systems, based on placing all the closed-loop poles of the system in desired positions to satisfy the requirements of dynamic response to obtain a stable system, looking for a compromise between the response speed and sensitivity to disturbances, relative stability in order to obtain a smooth response with low overshoot, etc. In addition, the closed-loop poles are the eigenvalues λ_1 and λ_2 in (25) and determine the absolute stability of the plant. They must be placed in a discrete control system between $0 < \lambda < 1$. The further away the eigenvalues are from $\lambda_i = 1$, the faster the response but the lower the relative stability. In this case, gain matrix K is a 2 by 2 matrix since there are two input signals and two state variables.

$$\det[(F - G \cdot K) - \lambda I]) = 0 \tag{25}$$

Figures 15 and 16 represent the controlled outputs (levels of D1 and D4) for multiple level references established and the control actions p and s (pump power of B1 and opening rate of valve V1). The local linear model (BMU neuron) employed at each time is also depicted. It has been pointed out that there is some inaccuracy between the output and the reference due to steady-state error in proportional control strategy. In Fig. 16, neither of the two levels reach either of the references established because they are so high that the pump does not have enough power to distribute such a level in both tanks.

8.2 Integral control scheme

Figure 17 shows the scheme used for integral control by state feedback. There are two additional state variables X_I indicated in (26) in the model that must be equal to the integral of the tracking error according to (27), since there are two level references, y_1^* and y_4^*, to be tracked by y_1 and y_4, respectively, and they give rise to two different error signals that must be integrated. The results of these integrations are stored in x_{I_1} and x_{I_2}. Considering this, the state space model used to obtain integral control gain matrix K by the pole assignment method is shown in (28) and control action $U(k)$ is obtained as in (29). Matrices denoted by F, G and H are the same as the original state space model. Similarly, to proportional control, the local linear model which represents the conditions of the system at every moment is calculated by BMU closeness to variables p_{k-1}, s_{k-1} and y_{k-1}. Note that disturbances η are originally located in the output of the model and they are not taken into account since integral control is able to reject them.

$$X_I(k) = \begin{bmatrix} x_{I_1}(k) \\ x_{I_2}(k) \end{bmatrix} \tag{26}$$

$$X_I(k+1) = X_I(k) + T_s(Y(k) - Y^*(k)) \tag{27}$$

where T_s is the sampling time and X_I are both state variables that compute the integral tracking error for each controlled output variable y_1 and y_4. Note that $Y(k) - Y^*(k)$ is the tracking error.

$$\begin{bmatrix} X(k+1) \\ X_I(k+1) \end{bmatrix} = \begin{bmatrix} F & \underline{0} \\ T_s \cdot H & I \end{bmatrix} \begin{bmatrix} X(k) \\ X_I(k) \end{bmatrix} + \begin{bmatrix} G \\ \underline{0} \end{bmatrix} U(k) + \begin{bmatrix} \underline{0} \\ -T_s \cdot I \end{bmatrix} \begin{bmatrix} y_1^*(k) \\ y_4^*(k) \end{bmatrix} \tag{28}$$

where I is a 2 by 2 identity matrix, $\underline{0}$ is a 2 by 2 null matrix, whereas y_1^* and y_4^* are the level references for tanks D1 and D4, respectively.

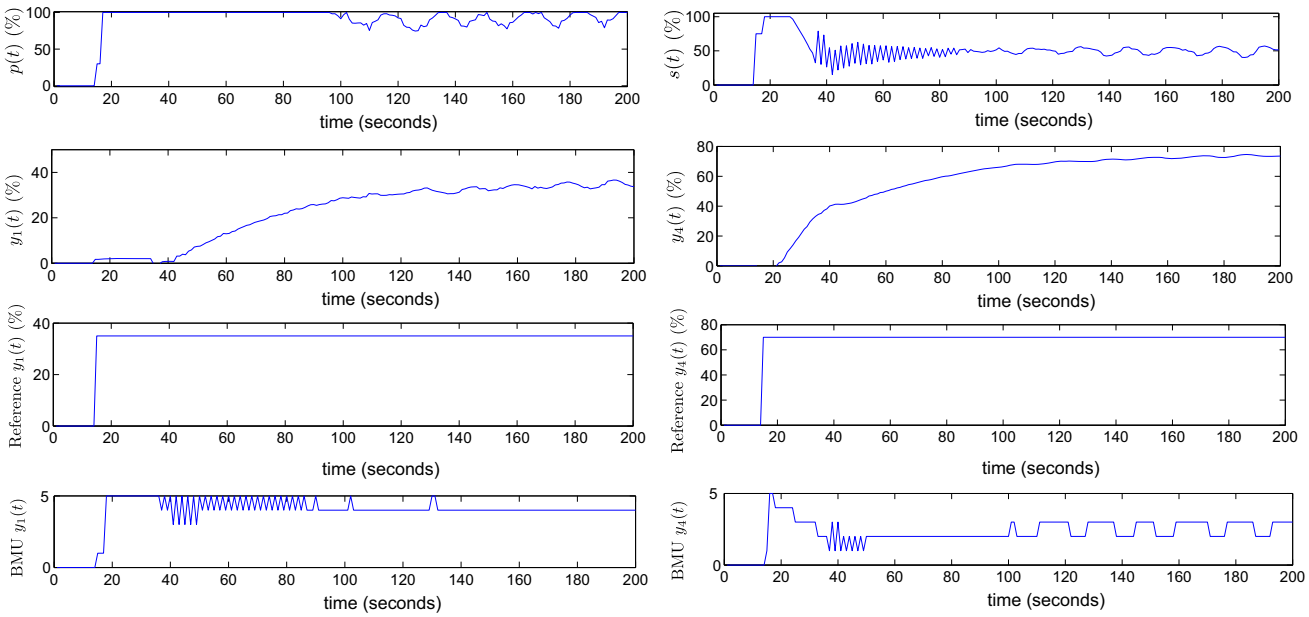


Fig. 15 Proportional control with level references of 35% for D1 tank and 70% for D4 tank

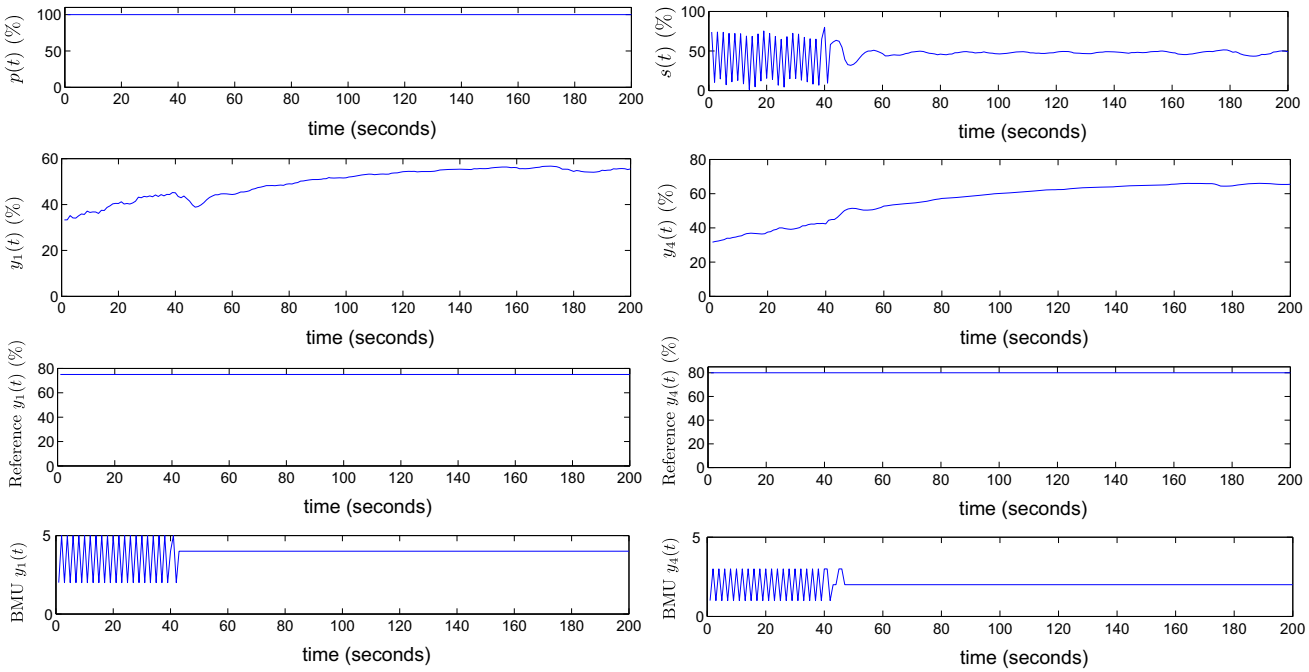


Fig. 16 Proportional control with level references of 80% for D1 and D4 tanks

$$U(k) = \begin{bmatrix} p(k) \\ s(k) \end{bmatrix} = -K \begin{bmatrix} X(k) \\ X_I(k) \end{bmatrix} = -[K_X \quad K_{X_I}] \begin{bmatrix} X(k) \\ X_I(k) \end{bmatrix} \tag{29}$$

To obtain the gain matrix K, the operation is analogous to proportional control for MIMO systems according to (30). In this case, K is a 2 by 4 block matrix which is composed of K_X and K_{X_I} matrices. The sizes of K_X and K_{X_I} are the same, 2 by 2.

$$\det[(A - B \cdot K) - \lambda I]) = 0 \tag{30}$$

where matrices A and B are indicated in (31) and (32), respectively.

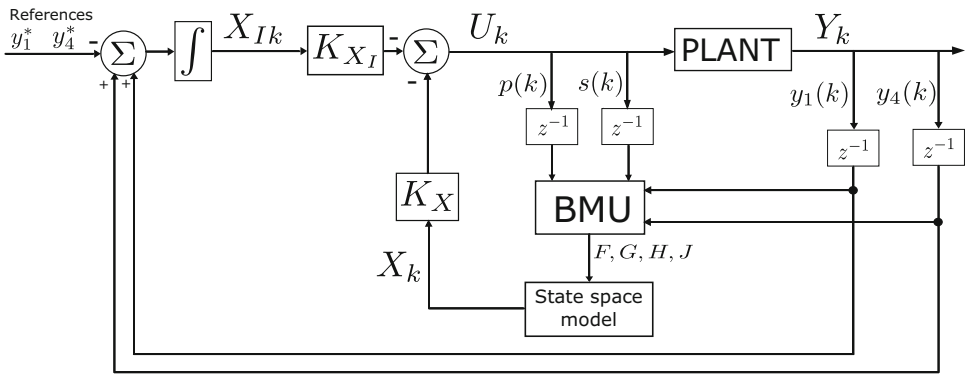


Fig. 17 Control loop implemented to carry out the integral control

$$A = \begin{bmatrix} F & \underline{0} \\ T_s \cdot H & I \end{bmatrix} \tag{31}$$

$$B = \begin{bmatrix} G \\ \underline{0} \end{bmatrix} \tag{32}$$

Figure 18 shows the controlled outputs for the indicated level references and both control actions, pump power $p(t)$ and opening percentage of the three-way valve $s(t)$. The local linear model (BMU neuron) employed each time is also represented. Selection of the eigenvalues in order to proceed with the pole placement method is critical. The eigenvalues must not have a multiplicity greater than the rank of matrix G, i.e., the number of inputs. The values of the eigenvalues were established around 0.9. Lower values can produce bad control effects such as saturation of control action U_k, increasing the overshoot of the response or oscillations of controlled variables y_1 and y_4. The tracking

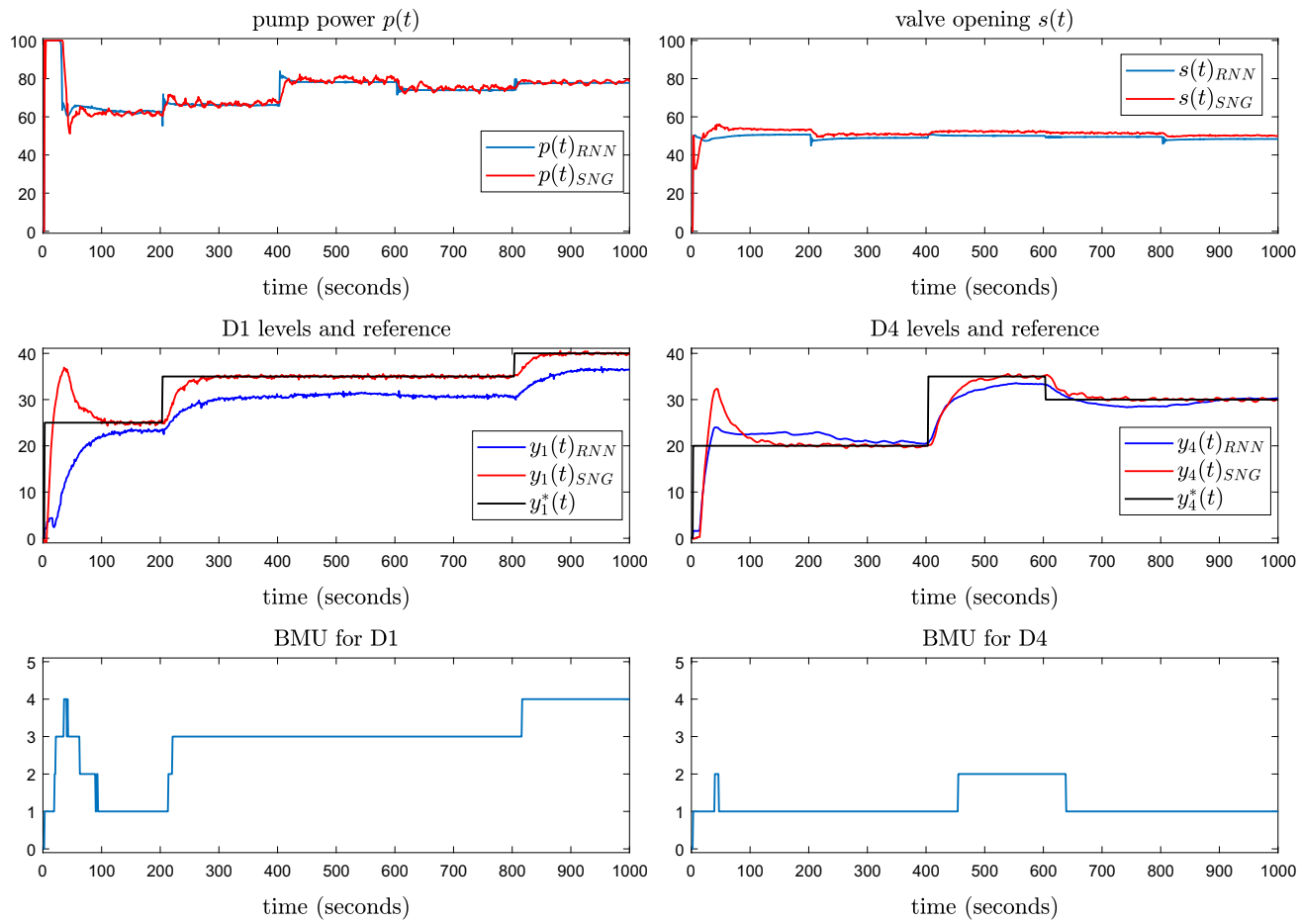


Fig. 18 Comparison between using supervised neural gas (*SNG*) versus recurrent NN (*RNN*)

of the reference is quite good, almost without steady-state error.

Finally, in order to make a comparison with a common approach by means of the recurrent NN controller, the level references have been arranged between 20% and 40% as they were considered during the controller training. When controlling a physical system or plant, there is an important issue regarding the steady-state errors in disturbance rejection, reference tracking and sensitivity to simple process changes. In this plant, the valve opening $s(t)$ is suffering from parameter changing because of its inherent engineering design. From the point of view of the control engineer, this fact is impossible to fix by means of replacing the valve and the control algorithm must resolve this situation. This is one of the main advantages of feedback control versus feedforward control. The perfect modeling of a physical system is impossible, and feedback control comes to the rescue when dealing with real situations such as this. Although the recurrent network is receiving feedback signals and its NARX model outperforms the dynamics identification of the five local linear models of the supervised neural gas, it appears to act like a feedforward controller. When using this type of NN, it is usually asserted that the steady-state error of the control will be improved with a better plant model identification, i.e., increasing the size of the training set, the number of neurons in the hidden layers, etc. This is correct, but perfect dynamics modeling is impossible and the problems of disturbance rejection and variations of the plant are always present in every physical system to a greater or lesser extent. In any case, the recurrent NN controller is quite good since it has a damped response as the reference model has determined.

9 Conclusion

In this work, it has been demonstrated that a nonlinear system composed of several linear subsystems can be approximated by means of a supervised neural gas network. There is a one-to-one assignment between the neurons and the transfer functions of the local linear models.

The equilibrium points for linearization are determined by the prototype vectors w and the reference values y_w. They are obtained in a small number of epochs. The training data must be balanced, i.e., they must contain the same number of representative samples from each local subsystem. The gradient vectors ∇f determine the pole–zero maps of the dynamic subsystems. The results confirm that the identification approach is deterministic.

The most important parameters that must be adjusted in order to train the neural gas network aiming to identify the dynamics of a nonlinear plant are the number of local linear models (or neurons), the sampling time and the number of epochs. The higher the number of neurons, the better the accuracy of the model. However, the advisable procedure is to use as few neurons as possible because the subsequent control might have switching problems between the local models. The lower the sampling time, the higher the number of epochs to achieve a correct identification of the plant dynamics by means of the gradient vectors. The values of the remaining parameters should be adjusted as outlined in Sect. 2.

The determination of model order is another important issue that must be taken into account when dealing with unknown systems. This procedure was carried out to identify the plant in its 2×2 operation mode without requiring a mathematical model to describe its nonlinear operation. The gradient vectors ∇f_{i*} obtained by its supervised version constitute the poles and zeros of the i-th local linear transfer functions of the plant. The unsupervised version of neural gas batch algorithm, in combination with Davies–Bouldin clustering index, has proved to be very robust for obtaining the optimal number of clusters.

The linear local models obtained by means of a supervised neural gas model greatly simplify the nonlinear controller design. The proportional and integral control implemented through state feedback uses a series of controllers obtained by the pole assignment method for MIMO systems. There is a different controller for each local linear model of the system. Neural network plays a supporting role to synthesize and tune feedback controllers autonomously and computationally effective. Special care must be taken when assigning the eigenvalues of the controlled system. Although the recurrent network is receiving feedback signals and its NARX model outperforms the dynamics identification of the five local linear models of the supervised neural gas, it appears to act like a feedforward controller.

Acknowledgements Our warmest thanks are expressed to the following for their financial support: The Commission of the European Communities, European Coal and Steel (ECSC), supporting MACO Pilot project "Optimization of the mixed-acid online monitoring and control in stainless steel pickling plants" with 709694 as grant agreement number.

Compliance with ethical standards

Conflicts of interest The authors declare that there is no conflict of interest.

References

1. Martinetz TM, Berkovich SG, Schulten KJ (1993) Neural-gas network for vector quantization and its application to time-series prediction. IEEE Trans Neural Netw 4(4):558–569

2. Cottrell M, Hammer B, Hasenfuss A, Villmann T (2006) Batch and median neural gas. Neural Netw 19(6–7):762–771
3. Hammer B, Strickert M, Villmann T (2005) Supervised neural gas with general similarity measure. Neural Process Lett 21:21–44
4. Hammer B, Hasenfuss A, Schleif FM, Villmann T (2006) Supervised batch neural gas. In: Artificial neural networks in pattern recognition, pp 33–45
5. Narendra KS, Parthasarathy K (1990) Identification and control of dynamical systems using neural networks. IEEE Trans Neural Netw 1(1):4–27
6. Hagan MT, Demuth HB, Beale MH (1996) Neural network design. PWS Publishing, Boston
7. Narendra KS, Parthasarathy K (1991) Learning automata approach to hierarchical multiobjective analysis. IEEE Trans Syst Man Cybern 20(1):263–272
8. Horn JM, De Jesús O, Hagan MT (2009) Spurious valleys in the error surface of recurrent networks—analysis and avoidance. IEEE Trans Neural Netw 20(4):686–700
9. Hagan MT, Demuth HB, De Jesús O (2002) An introduction to the use of neural networks in control systems. Int J Robust Nonlinear Control 12(11):959–985
10. De Jesús O, Hagan MT (2007) Backpropagation algorithms for a broad class of dynamic networks. IEEE Trans Neural Netw 18(1):14–27
11. Han X, Xie W-F, Zhijun F, Luo W (2011) Nonlinear systems identification using dynamic multi-time scale neural networks. Neurocomputing 74(17):3428–3439
12. Narendra KS, Lewis FL (2001) Editorial: introduction to the special issue on neural network feedback control. Automatica 37(8):1147–1148
13. Machón-González I, López-García H, Rodriguez-Iglesias J, Marañon-Maison E, Castrillon-Pelaez L, Fernandez-Nava Y (2013) Comparing feed-forward versus neural gas as estimators: application to coke wastewater treatment. Environ Technol 34(9):1131–1140
14. Machón-González I, López-García H (2017) Feedforward non-linear control using neural gas network. Complexity. https://doi.org/10.1155/2017/3125073
15. Hammer B, Strickert M, Villmann T (2005) Supervised neural gas with general similarity measure. Neural Process Lett 21:21. https://doi.org/10.1007/s11063-004-3255-2
16. Arnonkijpanich B, Hasenfuss A, Hammer B (2011) Local matrix adaptation in topographic neural maps. Neurocomputing 74:522–539
17. Crespo-Ramos MJ, Machón-González I, López-García H, Calvo-Rolle JL (2013) Detection of locally relevant variables using SOM-NG algorithm. Eng Appl Artif Intell 26(8):1992–2000
18. Villmann T, Hammer B, Schleif F, Geweniger T, Herrmann W (2006) Fuzzy classification by fuzzy labeled neural gas. Neural Netw 19:6–7
19. Chen L, Narendra KS (2001) Nonlinear adaptive control using neural networks and multiple models. Automatica 37(8):1245–1255
20. Narendra KS, Mukhopadhyay S (1997) Adaptive control using neural networks and approximate models. IEEE Trans Neural Netw 8:475–485
21. Franklin GF, Powell DJ, Emami-Naeini A (2001) Feedback control of dynamic systems, 4th edn. Prentice Hall PTR, Upper Saddle River
22. Kiong LC, Rajeswari M, Rao MVC (2003) Nonlinear dynamic system identification and control via constructivism inspired neural network. Appl Soft Comput 3(3):237–257