Hindawi
Complexity
Volume 2017, Article ID 3125073, 11 pages
https://doi.org/10.1155/2017/3125073

Research Article

Feedforward Nonlinear Control Using Neural Gas Network

Iván Machón-González and Hilario López-García

Departamento de Ingeniería Eléctrica, Electrónica de Computadores y Sistemas, Universidad de Oviedo, Edificio Departamental 2, Zona Oeste, Campus de Viesques s/n, 33204 Gijón/Xixón, Spain

Correspondence should be addressed to Iván Machón-González; machonivan@uniovi.es

Received 19 July 2016; Accepted 17 November 2016; Published 15 January 2017

Academic Editor: Francisco Gordillo

Nonlinear systems control is a main issue in control theory. Many developed applications suffer from a mathematical foundation not as general as the theory of linear systems. This paper proposes a control strategy of nonlinear systems with unknown dynamics by means of a set of local linear models obtained by a supervised neural gas network. The proposed approach takes advantage of the neural gas feature by which the algorithm yields a very robust clustering procedure. The direct model of the plant constitutes a piecewise linear approximation of the nonlinear system and each neuron represents a local linear model for which a linear controller is designed. The neural gas model works as an observer and a controller at the same time. A state feedback control is implemented by estimation of the state variables based on the local transfer function that was provided by the local linear model. The gradient vectors obtained by the supervised neural gas algorithm provide a robust procedure for feedforward nonlinear control, that is, supposing the inexistence of disturbances.

1. Introduction

Although some physical systems can be approximated as a linear model, almost all real plants actually have a nonlinear functioning. A wide understanding of the behavior of nonlinear processes is available but it is sometimes difficult to choose the appropriate control method. Lyapunov theory is a classic method for nonlinear system control. If and only if there is a positive definite continuous function whose derivative is negative under the suitable conditions of the control design, then the control asymptotic stability is guaranteed. However, this method is unfortunate because obtaining the Lyapunov function is difficult. This problem is even worse when dealing with unknown plants that are not defined mathematically. Therefore, it is usually not easy to guarantee the stability of a complex nonlinear system [1]. However, if the local linear system corresponding to an equilibrium point is controllable, then sufficient conditions can be stated for local stability [2].

Hartman-Grobman theorem states that the behavior of a nonlinear system in the neighborhood of an equilibrium point can be approximated by its linearized model. The systems theory is based on many mathematical procedures about stability, controllability, and observability regarding linear systems. The stability and, to a great extent, the dynamic response of a linear system can be described in terms of eigenvalues of the system matrix in state space design or poles of the transfer function. No such method exists for nonlinear systems. For this reason, industrial control processes are still usually designed using this linear control theory. After linearization, the typical approach is to design a linear controller such as PID with fixed parameters.

The classical approach to get local linear models can be achieved with RLS (Recursive Least Squares) method. However, sometimes this method throws up unfavorable results due to the intrinsic nonlinearities of the process to be controlled. The problem is to establish the different operating points for a nonlinear system. At this point, the proposed algorithm can establish each operating point as a cluster centre of the neural gas network. It is for such reason that artificial intelligence techniques improve the control performance.

Research into identification and control of nonlinear systems by means of neural networks (NN) began over two decades ago [3]. One of the major advantages of control by NN is that precise knowledge of the plant such as a mathematical model is not needed. Initially, control applications using NN were based on a trial-and-error approach. Research efforts have improved the control algorithms and several journals have published special issues with a strong mathematical foundation [4]. Many applications are based on a combination between feedforward and recurrent NN. Recurrency, also known as dynamic backpropagation, is necessary due to the dependency of the output on the previous values of the same output, which are also functions of the weights [5]. Zhang and Wang [6] proposed a pole assignment control using recurrent NN.

The typical design procedure is to carry out the system identification in order to model the plant and, secondly, to obtain the controller. Traditional methods rely heavily on models extracted from physical principles, whereas approaches based on NN theory usually create black-box models as function approximators using data obtained from the plant. Knowledge about the mathematical model of the plant or any other physical principle is not necessary.

Neural gas (NG) is an unsupervised prototype-based method [7] in which the prototype vectors are the weights and carry out a partition of the training data space. It considers the cooperation-competition computation, allowing the algorithm to be prevented from local minima problems. In addition, the batch NG allows fast training so that the convergence is achieved in a small number of epochs [8]. Supervised versions of NG have also been developed, specially for classification [9, 10]. The algorithm has a great robustness for clustering tasks but has also been proven to be robust to obtain direct models of plants [11].

After years of works in identification and control of dynamical systems by means of NN, there is agreement among researchers that linear identifiers and controllers should be used as first attempt, as stated in Chen and Narendra [2]. If a set of local linear models corresponding to several equilibrium points can approximate with certain accuracy a nonlinear system, then linear controllers can be designed for each model and the global control is related to control by switched linear models.

This divide-and-conquer approach is applied in this work. The resulting model is a set of local linear maps. Each neuron of the NG model corresponds to one local model. These local models are obtained after NG training. In this way, a direct model of the plant is obtained. After obtaining this NG model, the design of the local linear controller is simpler than that of the global nonlinear controller. Local linear mapping using another prototype-based algorithm such as SOM was successfully tested at the NASA facilities [12].

This paper aims to apply the robustness modeling capability of NG to control a nonlinear plant such as a typical robot manipulator.

The paper contains the learning rules of the considered NG algorithm in Section 2, the model of the plant and the control strategy are explained in Sections 3 and 4, respectively, and the proposed technique is tested in Section 5.

2. Neural Gas Approach

The unsupervised version of the NG algorithm is based on energy cost function (1) according to the Euclidean metric. The notation used for the squared Euclidean distance is given in (2). Moreover,

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

$$d\left(v_j, w_i\right) = \left\| v_j - w_i \right\| = \left(v_j - w_i\right)^2. \tag{2}$$

A neighborhood function (3) is needed to implement the algorithm. The rank function $k(v, w_i) \in 0, \ldots, m-1$ represents the rank distance between prototype w_i and data vector v. The minimum distance takes the value 0 and the rank for the maximum distance is equal to $m-1$, where m is the number of neurons or prototypes and $\sigma(t)$ is the neighborhood radius:

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

The neighborhood radius $\sigma(t)$ is usually chosen to decrease exponentially according to (4). The decrease goes from an initial positive value, σ_{t_0}, to a smaller final positive value, $\sigma_{t_{\max}}$:

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

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 Arnonkijpanich et al. [13]. In addition, $\sigma_{t_{\max}} = 0.0001$ in order to minimize the quantization error at the end of the training.

The learning rule of the batch version is obtained in Cottrell et al. [8]. The batch algorithm can be obtained by means of Newton's method using the Jacobian and Hessian matrices, J and H, respectively, of the cost function E_{NG}. The adaptation of the prototype w_i is formulated accordingly based on this method

$$\Delta w_i = -J\left(w_i\right) \cdot H^{-1}\left(w_i\right). \tag{5}$$

Kernel function h_{NG} can be considered locally constant [8]. In this way, the Jacobian and Hessian matrices are

$$\begin{aligned} J\left(w_i\right) &= -2 \cdot \sum_{j=1}^{N} h_\sigma\left(v_j, w_i\right) \cdot \left(v_j - w_i\right), \\ H\left(w_i\right) &= 2 \cdot \sum_{j=1}^{N} h_\sigma\left(v_j, w_i\right). \end{aligned} \tag{6}$$

Substituting (6) into (5), the increment can be obtained

$$\Delta w_i = \frac{\sum_{j=1}^{N} h_\sigma\left(v_j, w_i\right) \cdot \left(v_j - w_i\right)}{\sum_{j=1}^{N} h_\sigma\left(v_j, w_i\right)}. \tag{7}$$

Finally, the updating rule for each prototype vector appears in

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

2.1. Supervised Learning. Supervised learning with NG is possible by means of local linear mapping over each Voronoi region defined by prototype vector w_i. A constant y_i and a vector ∇f_i with the same dimension as w_i are assigned to each neuron i. The goal is to approximate the function $y = f(v)$ from $\mathfrak{R}^D$ to $\mathfrak{R}$, where D is the number of training variables, that is, the dimension of data vector v. The training thus becomes supervised and the dataset contains input-output pairs of data vector v and variable y as the objective function. The estimation is carried out by

$$\widehat{y}\left(v\right) = y_{i^*} + \nabla f_{i^*} \cdot \left(v - w_{i^*}\right), \tag{9}$$

where $\widehat{y}(v)$ is the estimated output value, y_i is the reference value learned for w_i, ∇f_i is the gradient of the approximated function obtained in the ith Voronoi region defined by w_i, and i^* is the neuron i with its closest w_i to data vector v, that is, the best matching unit (BMU). The asterisk super index denotes the winning neuron for input data vector v.

The probability distribution of the input data is represented by prototype vectors w which are previously updated according to the typical rule of the unsupervised version of the algorithm [14] using (8). Each prototype vector w_i can be considered as the centroid of the ith Voronoi region. After unsupervised training, m regions are well defined by these vectors. At this point, local models will be created in each of these regions so that m local models will represent the whole data distribution.

The energy cost function of the supervised version of the algorithm is based on the mean square error of the output variable estimation averaged over each Voronoi region [15] according to (10). Prototypes w_i are already obtained in (8), whereas the adaptation rules of y_i and ∇f_i are calculated considering Newton's method for energy cost (10). The learning rules for y_i and ∇f_i are shown in (11) and (12), respectively:

$$\begin{aligned} E_{\text{NGsup}} &= \sum_{i=1}^{m}\sum_{j=1}^{N} h_{\sigma(v_j,w_i)} \cdot \left(f\left(v_j\right) - \widehat{f}\left(v_j\right)\right)^2 \\ &= \sum_{i=1}^{m}\sum_{j=1}^{N} h_{\sigma(v_j,w_i)} \cdot \left(y_j - y_i - \nabla f_i \cdot \left(v - w_i\right)\right)^2, \end{aligned} \tag{10}$$

$$y_i = \frac{\sum_{j=1}^{N} h_{\sigma(v_j,w_i)} \cdot f\left(v_j\right)}{\sum_{j=1}^{N} h_{\sigma(v_j,w_i)}}, \tag{11}$$

$$\begin{aligned} &\Delta\nabla f_i \\ &= \frac{\sum_{j=1}^{N} h_{\sigma(v_j,w_i)} \cdot \left(v_j - w_i\right) \cdot \left(f\left(v_j\right) - y_i - \nabla f_i \cdot \left(v_j - w_i\right)\right)}{\sum_{j=1}^{N} h_{\sigma(v_j,w_i)} \cdot \left(v_j - w_i\right) \cdot \left(v_j - w_i\right)}. \end{aligned} \tag{12}$$

3. Plant Model

After NG training, the plant is modeled as a set of linear systems 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 [16, 17] and can be expressed as

$$\begin{aligned} Y_{k+d} = h\left(Y_k, Y_{k-1}, Y_{k-2}, \ldots, Y_{k-n+1}, U_k, U_{k-1}, U_{k-2}, \ldots,\right. \\ \left. U_{k-n+1}\right), \end{aligned} \tag{13}$$

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.

Considering zero delay system and substituting (13) for (9) remains

$$\begin{aligned} Y_k = {} & y_{i^*} + \nabla f_{i^*,1}\left(Y_{k-1} - w_{i^*,1}\right) + \nabla f_{i^*,2}\left(Y_{k-2} - w_{i^*,2}\right) \\ & + \cdots + \nabla f_{i^*,n}\left(Y_{k-n} - w_{i^*,n}\right) \\ & + \nabla f_{i^*,n+1}\left(U_k - w_{i^*,n+1}\right) \\ & + \nabla f_{i^*,n+2}\left(U_{k-1} - w_{i^*,n+2}\right) + \cdots \\ & + \nabla f_{i^*,2n+1}\left(U_{k-n} - w_{i^*,2n+1}\right). \end{aligned} \tag{14}$$

Hereafter, the gradients will be denoted as coefficients a_i and b_i.

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

And the following terms will be gathered to form variable η:

$$\eta_k = y_{i^*} - \sum_{j=1}^{2n+1} \nabla f_{i^*,j} \cdot w_{i^*,j}. \tag{15}$$

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

$$Y\left(z\right) = \frac{B\left(z^{-1}\right)}{A\left(z^{-1}\right)} U\left(z\right) + \frac{1}{A\left(z^{-1}\right)} \eta\left(z\right) \tag{16}$$

which reminds one of an ARMAX model, η is not only a zero mean independent identically distributed white noise process but also a known disturbance calculated according to (15), 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 where the controller must be connected.

4. Local Linear Control by State Feedback

If the system is linear (locally), then the superposition theorem can be applied remaining the linear transfer function between the system output and the control input as follows:

$$Y(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_n z^{-n}}{1 - a_1 z^{-1} - a_2 z^{-2} - \cdots - a_n z^{-n}} U(z). \tag{17}$$

Define

$$Q(z) = \frac{U(z)}{1 - a_1 z^{-1} - a_2 z^{-2} - \cdots - a_n z^{-n}} \tag{18}$$

and choose the following relationship between the state variables: $x_1(z) = z^{-n}Q(z)$, $x_2(z) = z^{-n+1}Q(z), \ldots, x_n(z) = z^{-1}Q(z)$.

Transfer function (17) can be expressed in control canonical form for linear state space design as

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ \vdots \\ x_{n-1}(k+1) \\ x_n(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ a_n & a_{n-1} & a_{n-2} & \cdots & a_1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ \vdots \\ x_{n-1}(k) \\ x_n(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} u(k), \tag{19}$$

$$Y(k) = \begin{bmatrix} b_n + a_n b_0 & b_{n-1} + a_{n-1} b_0 & \cdots & b_1 + a_1 b_0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ \vdots \\ x_n(k) \end{bmatrix} + b_0 u(k). \tag{20}$$

Assuming that the system is controllable, the purpose of the control by state feedback via pole placement is to assign a set of pole locations for the closed-loop system that will correspond to satisfactory dynamic response in terms of rise time, settling time, and overshoot of the transient response. The control law is a linear combination of the state variables x_i which are estimated in (19) by way of local transfer function (17).

The characteristic polynomial of the closed-loop system depending on system matrix F, input matrix G, and gain vector K is

$$\det[zI - (F - GK)] = 0 \tag{21}$$

whereas the characteristic polynomial of the desired pole locations is

$$p(z) = \prod_{j=1}^{n} (z - \lambda_j) = z^n + p_1 z^{n-1} + p_2 z^{n-2} + p_3 z^{n-3} + \cdots + p_n. \tag{22}$$

For an nth-order system, the gain vector $K = [K_1 \ K_2 \ \cdots \ K_n]$ for state feedback is obtained by matching coefficients in (21) and (22) forcing the closed-loop poles to be placed at the desired locations:

$$K = [p_1 + a_1 \ \ p_2 + a_2 \ \ \cdots \ \ p_n + a_n]. \tag{23}$$

It is possible that there are enough degrees of freedom to choose arbitrarily any desired root location by selecting the proper values K_i. This is an inexact procedure that may require some iteration by the designer. The solution of the local linear model lies in finding the matrix or the regulator coefficients that implement the state feedback control. The stability condition for linear discrete-time systems is that all the eigenvalues must be inside the unit circle. Obviously, this criterion is not valid for nonlinear systems but there is a region inside the stable linear area where the asymptotic stability of the switched linear systems is achieved [18]. Thus, not only a desired dynamic response can be designed, but also stability criteria will be accomplished. In this work, this stability region was found by means of trial-and-error with different eigenvalues.

The proposed control strategy scheme is shown in Figure 1. Gain vector K is calculated to fulfill the dynamics according to (22) depending on the local linear model defined by the current winning neuron i^* or BMU. State variables x_i are also obtained by the local linear model of the NG in (19). Tracking of the setpoint reference is possible using the inverse static gain of the feedback loop. In addition, since disturbance η is known (it is included by the model), it can be compensated as $-\eta/A_{i^*}(z^{-1})$. The transfer function of the prefilter has been chosen as $(1 - \lambda_{\text{prefilter}})^n/(z - \lambda_{\text{prefilter}})^n$ and determines the switching rate of the local linear models. Although the pole assignment method does not affect the zeros of the plant, the prefilter can be optionally designed in order to cancel dominant zeros located inside the unit circle.

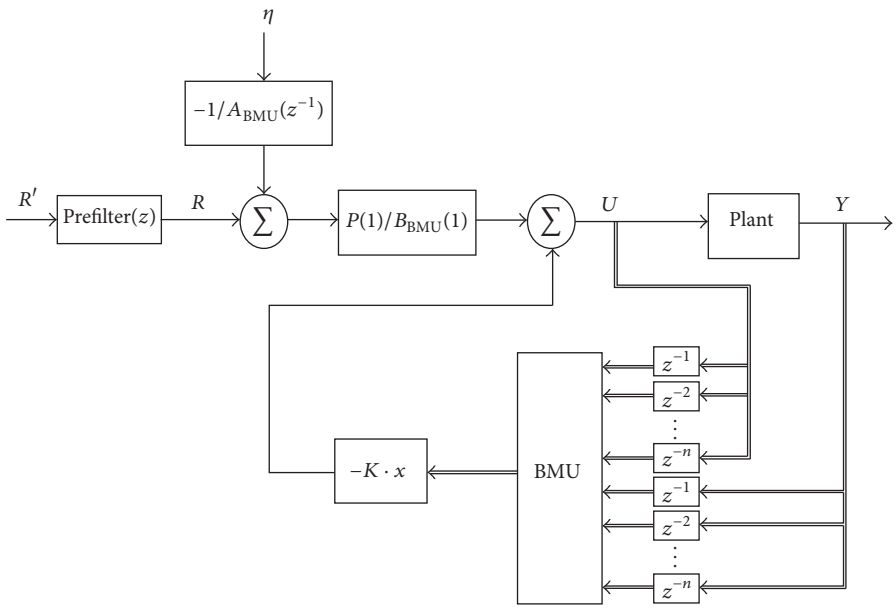


Figure 1: Control strategy by state feedback and local linear models.

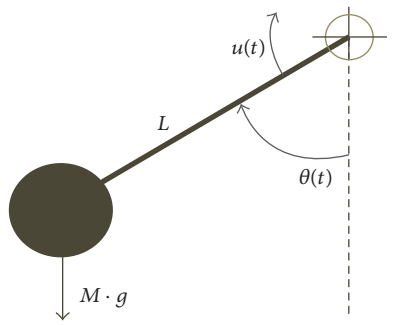


Figure 2: Plant used to test the proposed control.

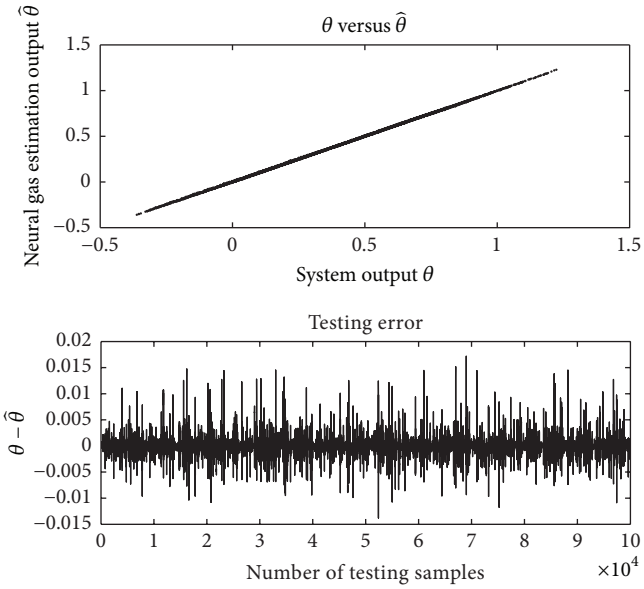


Figure 3: Results for testing data using $n = 4$.

5. Experimental Testing

The aim is to control the typical robot arm problem depicted in Figure 2. Hagan et al. [19] focused on this plant to be controlled by dynamic propagation algorithm using a Model Reference Control architecture. Obviously the proposal control is not based on the mathematical model of the plant, but this is a well-known second-order nonlinear differential equation:

$$u(t) = J\ddot{\theta}(t) + B\dot{\theta}(t) + MgL\sin\theta(t), \tag{24}$$

where $M = 1\,\mathrm{kg}$, $J = 1\,\mathrm{kgm}^2$, $L = 1\,\mathrm{m}$, $B = 1\,\mathrm{kgm}^2/\mathrm{s}$, and $g = 10\,\mathrm{m/s}^2$. The viscous friction coefficient is important regarding stability and dynamic response. If $B = 0$, then the system is unstable in open loop and the necessary training data can not be obtained by open loop simulation of the plant. In the present work, the system is simulated in open loop to acquire the training data considering $B = 1$ in order to propose a plant with more oscillatory response in comparison to Hagan et al. [19] where $B = 2$. Plant input u is a uniformly distributed random signal with −4 and 8.1 as minimum and maximum values of amplitude, respectively. The pulse width must be carefully selected in order to model the transient and steady states correctly. Thus, the pulse width is equal to 14 seconds. Since NG is a vector quantization algorithm where the neurons are updated according to the probability distribution function of the training data, it is important to obtain a uniform distribution of output value θ in the training data. After plant simulation, it was observed that the present system output θ_k did not depend on the present system input

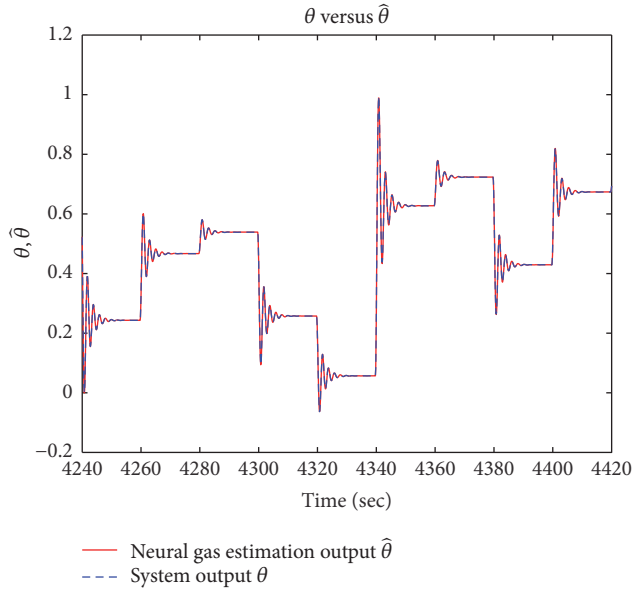


Figure 4: Results for testing data using $n = 4$.

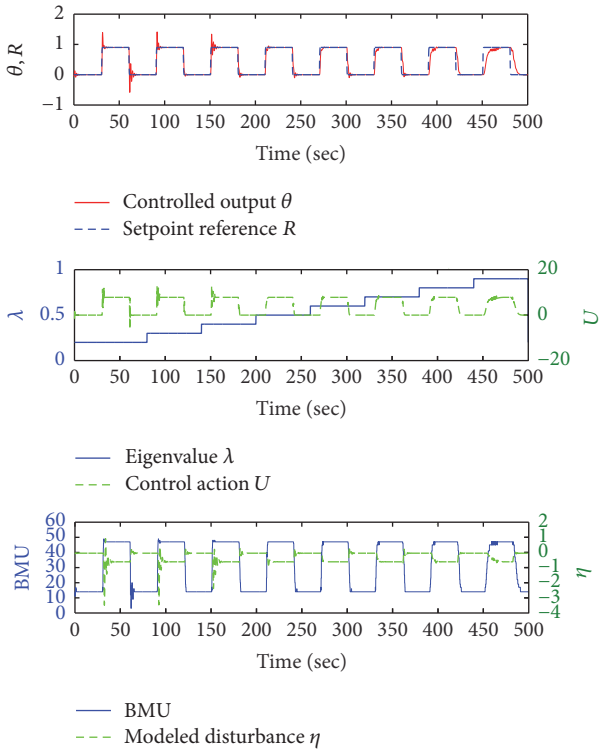


Figure 5: Results for $\lambda_{\text{prefilter}} = 0.4$.

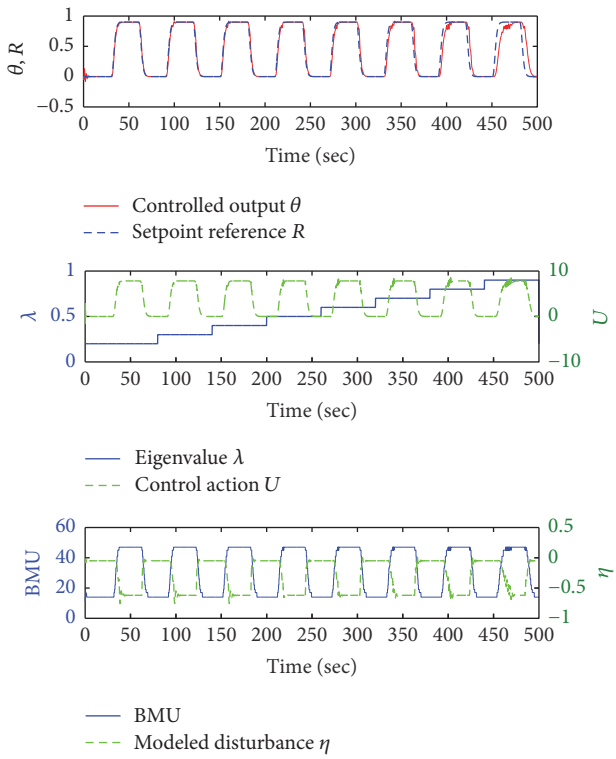


Figure 6: Results for $\lambda_{\text{prefilter}} = 0.9$.

u_k in the training dataset. If $n = 2$ is considered, then the plant is not correctly modeled, only the steady state is approximated but not the transient. If $n = 3$, then the plant model is quite accurate but the control is not well performed due to the system nonlinearities. The optimum value is $n = 4$. Batch NG training was carried out considering 50 epochs and 49 neurons to obtain the direct model of the plant. Fewer neurons cause a similar effect to that mentioned above for $n = 2$ and using more neurons does not improve the control. Figures 3 and 4 show the results for testing data after training for $n = 4$. Obviously, the sampling time must meet the criteria

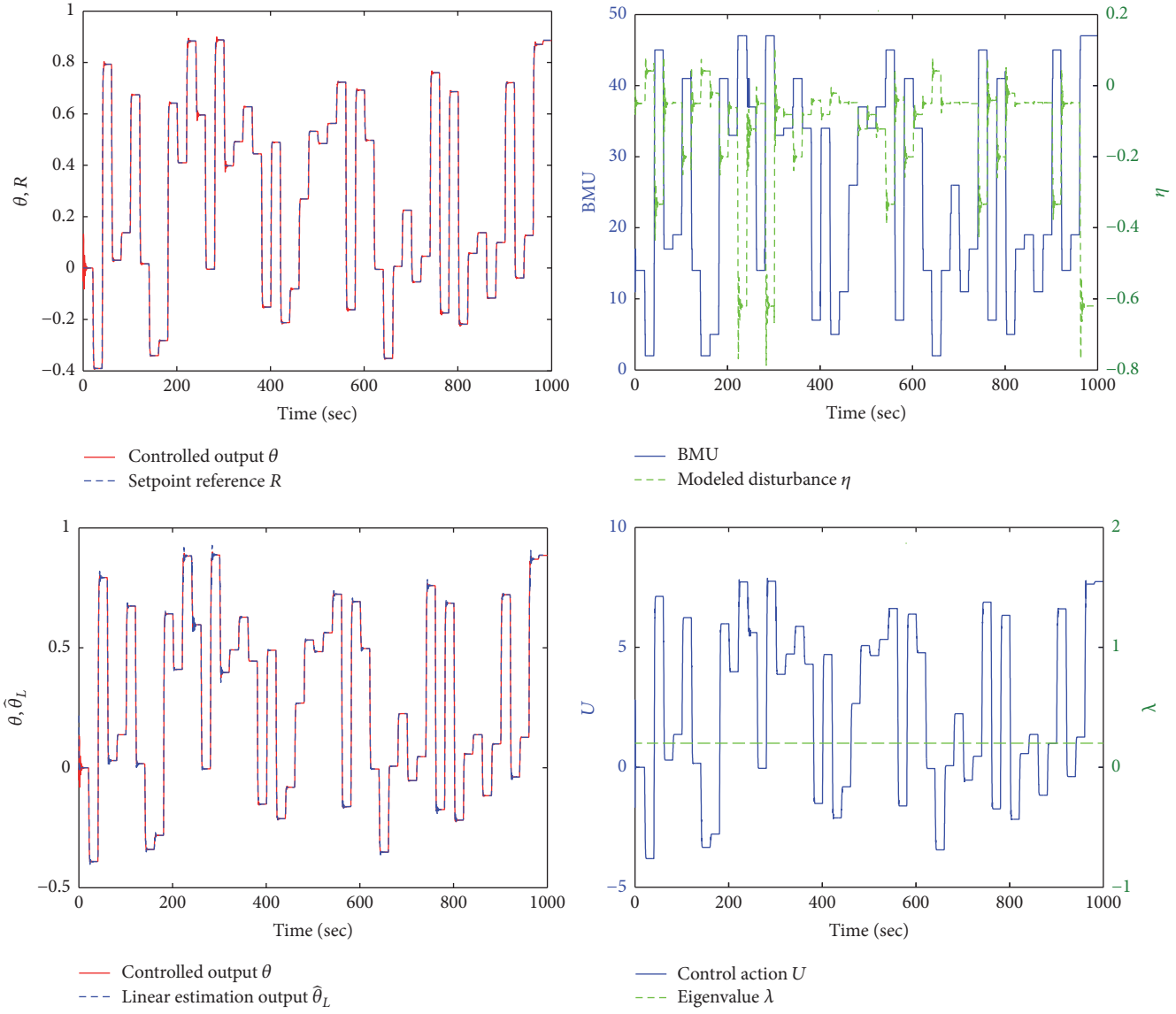


FIGURE 7: Results for $\lambda_{\text{prefilter}} = 0.7$ and $\lambda = 0.2$.

of Nyquist-Shannon theorem. The considered sampling time was 0.1 seconds.

Eigenvalues λ_j in (22) provide asymptotic stability and suitable dynamic response. We consider that all the eigenvalues are equal; that is, $p(z) = (z - \lambda)^4$. In order to tune λ, the control system was tested for different eigenvalues and prefilter poles. Steps of amplitude 0.9 at the reference setpoint were used since θ values close to 0.9 represent the worst system working zone with high nonlinearity. Obviously, θ values over 0.9 are even worse, but we refer to the training data range. Eigenvalue λ was incremented from 0.2 to 0.9 in series of 0.1 amplitude steps. Figures 5 and 6 show the results. If the prefilter has a wide bandwidth ($\lambda_{\text{prefilter}}$ is low), then lower values of λ produce instability. In Figure 5, lower λ values yield a considerable effect of the modeled disturbance η and the overshoot is high, whereas there are some rebounds between two BMU linear models for higher λ values. The lower λ, the lower the rise time (the wider the bandwidth). The optimum λ range is [0.5–0.6] for $\lambda_{\text{prefilter}} = 0.4$. Thus, when using switched linear models there is a stability region of λ within the global stability area of the linear systems theory (unit circle) depending on the switching rate of the NG local linear models [18]. Here the switching rate is determined by the prefilter transfer function. If $\lambda_{\text{prefilter}}$ is increased then good results are obtained for $\lambda = 0.2$, and see Figure 6.

Once the parameters had been determined, the NG approach was tested to track a constant reference for control of position in Figure 7 and a variable reference such as a sinusoidal signal to check control of velocity in Figure 8. The linear estimation output $\widehat{\theta}_L$ is not the NG estimation $\widehat{\theta}$ in (9) but it is calculated by means of (20) and adding the value of known disturbance η.

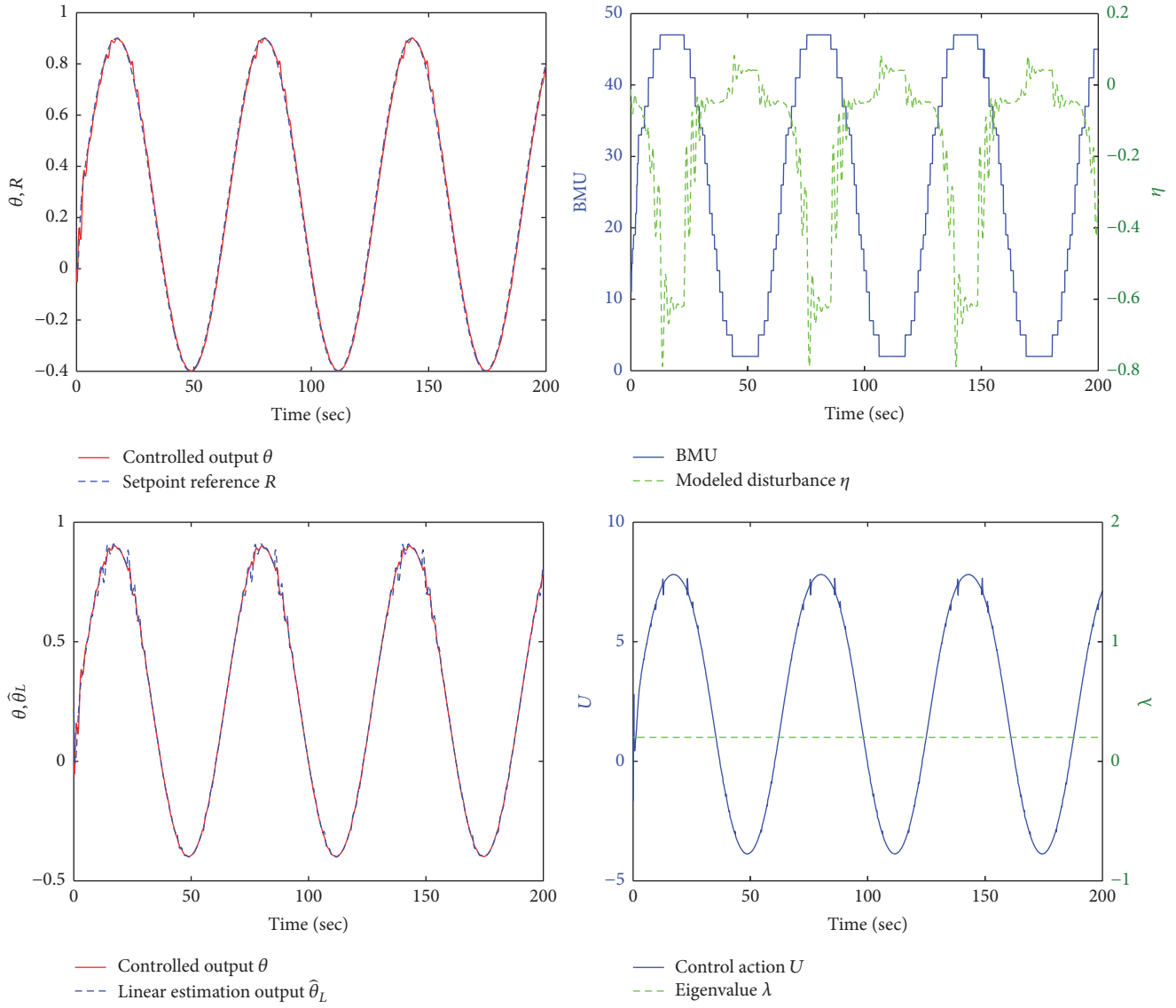


FIGURE 8: Results for $\lambda_{\text{prefilter}} = 0.7$ and $\lambda = 0.2$.

The worst control is when the reference value is close to 0.9. To illustrate this problem, PI control with fixed parameters was compared to NG approach. Obviously, PID control would achieve a fast and stable response due to derivative action but at the cost of an unrealizable control action and the system control would be affected by noisy signals. The linearized model using the s-transform in the neighborhood of θ_0, denoted as $G|_{\theta_0}$, is

$$\frac{\Theta(s)}{U(s)} = G|_{\theta_0} = \frac{1}{Js^2 + Bs + MgL\cos\theta_0}. \quad (25)$$

A suitable PI design regarding settling time and smoothness response is $\text{PI}(s) = k_p \cdot (s + (k_i/k_p))/s = 0.3 \cdot (s + 10)/s$ considering $\theta_0 = 0.5$. Figure 9 shows that the system controlled by this PI with fixed parameters becomes more oscillatory when considering $\theta_0 = 0.9$ because the two complex poles of the plant change their location involving the change of the closed-loop poles. The two complex closed-loop poles are more dominant and, therefore, the system increases oscillation. In this design the trade-off is between the settling time and the oscillatory component of the response. The tuning is to change k_p keeping constant the location of the zero in −10; that is, $k_i/k_p = 10$. Figure 10 shows the influence of the adjustment of k_p in the control design and the NG approach is displayed to be compared to that one.

The control strategy described above is valid for feedforward control, that is, supposing the inexistence of disturbances. In these conditions, the algorithm promises very good performance. However, disturbance rejection can be achieved adding an extra state variable so that $\dot{x}(t) = e(t)$, where e is the tracking error, and following the steps

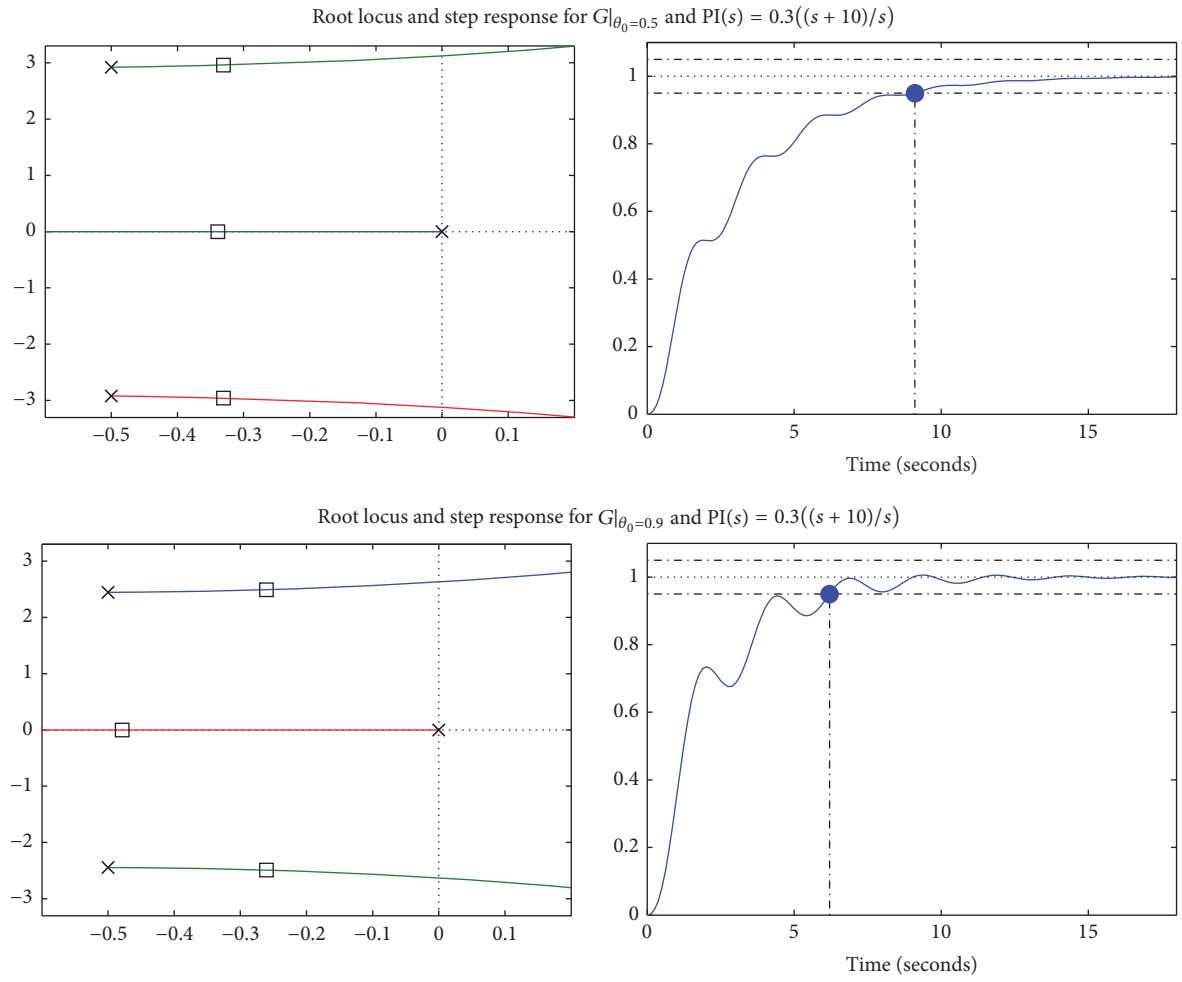


Figure 9: PI control at two different linearization points.

described from (21) to (23), Figure 11 shows the rejection of a constant output disturbance of amplitude 0.05.

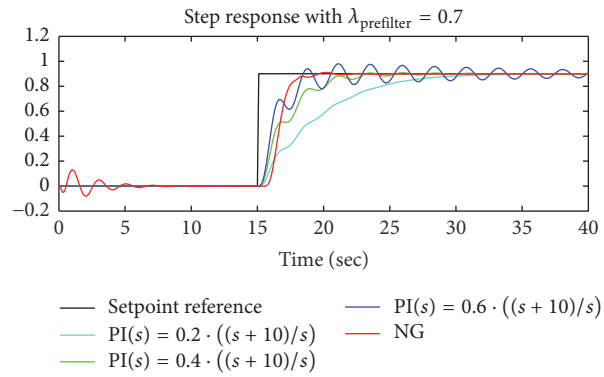
6. Conclusions

In this paper a supervised version of neural gas (NG) algorithm is proposed to control nonlinear systems whose dynamic mathematical models are unknown. The identification of the plant is achieved with the NG model. In comparison to other types of neural networks, the formation of the NG model is a robust procedure since there are neither problems of local minima nor overfitting. The training data must be carefully selected in order to model the transient and steady states correctly. The NG algorithm tends to model the steady states quite well. Obviously, the transient must be correctly modeled in order to control the plant. In this way, the number of delayed samples n and the number of neurons m are key parameters. There must be a sufficient number of neurons but the control is not improved if it is too large.

The trained NG network produces a set of piece-wise local linear models. Each of these is represented by a neuron. The global controller is a set of linear controllers which are obtained by state feedback via pole assignment. This control does not affect zeros but if these are inside the unit circle, then they can be cancelled by the poles of the prefilter.

Eigenvalues λ inside the unit circle do not guarantee the asymptotic stability because the plant to be controlled is nonlinear. Therefore, the stability corresponds to a region inside the unit circle [18]. This set of λ values was assigned by means of trial-and-error. The worst performance occurs for the highest setpoint values where the nonlinearities arise. The proposed approach provides a smoother and faster response than the typical PI with fixed parameters.

To conclude, NG algorithm provides a robust procedure not only for clustering tasks, but also for feedforward nonlinear control using the gradient vectors obtained by the supervised version. These gradient vectors constitute the poles and zeros of the local transfer function of the plant. The computational complexity is linear regarding the number of samples, neurons, and variables because of the efficient implementation in batch procedure.



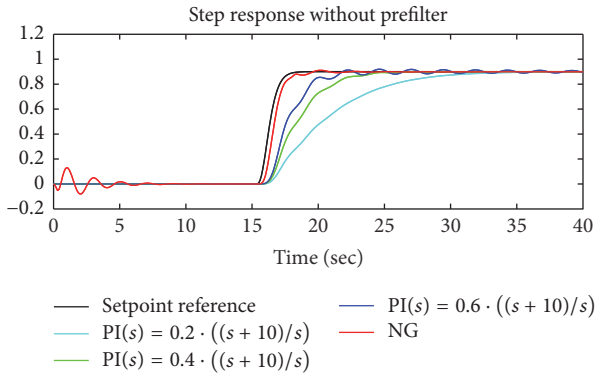


Figure 10: NG in comparison to PI.

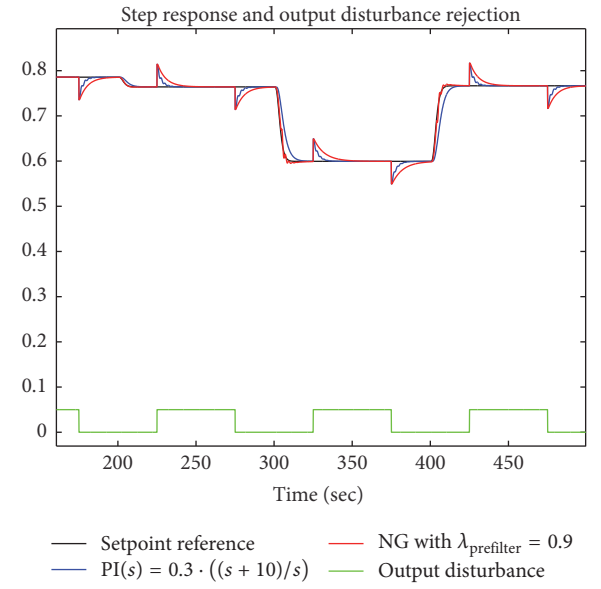


Figure 11: Disturbance rejection.

Competing Interests

The authors declare that they have no competing interests.

References

[1] B. Jakubczyk and E. D. Sontag, "Controllability of nonlinear discrete-time systems: a lie-algebraic approach," *SIAM Journal on Control and Optimization*, vol. 28, no. 1, pp. 1–33, 1990.

[2] L. Chen and K. S. Narendra, "Identification and control of a nonlinear discrete-time system based on its linearization: a unified framework," *IEEE Transactions on Neural Networks*, vol. 15, no. 3, pp. 663–673, 2004.

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

[4] K. S. Narendra and F. L. Lewis, "Editorial: introduction to the special issue on neural network feedback control," *Automatica*, vol. 37, no. 8, pp. 1147–1148, 2001.

[5] O. De Jesús and M. T. Hagan, "Backpropagation algorithms for a broad class of dynamic networks," *IEEE Transactions on Neural Networks*, vol. 18, no. 1, pp. 14–27, 2007.

[6] Y. Zhang and J. Wang, "Recurrent neural networks for nonlinear output regulation," *Automatica. A Journal of IFAC, the International Federation of Automatic Control*, vol. 37, no. 8, pp. 1161–1173, 2001.

[7] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten, "'Neural-gas' network for vector quantization and its application to time-series prediction," *IEEE Transactions on Neural Networks*, vol. 4, no. 4, pp. 558–569, 1993.

[8] M. Cottrell, B. Hammer, A. Hasenfuß, and T. Villmann, "Batch and median neural gas," *Neural Networks*, vol. 19, no. 6-7, pp. 762–771, 2006.

[9] B. Hammer, M. Strickert, and T. Villmann, "Supervised neural gas with general similarity measure," *Neural Processing Letters*, vol. 21, no. 1, pp. 21–44, 2005.

[10] B. Hammer, A. Hasenfuss, F. M. Schleif, and T. Villmann, "Supervised batch neural gas," in *Artificial Neural Networks in Pattern Recognition*, Lecture Notes in Computer Science, pp. 33–45, Springer, Berlin, Germany, 2006.

[11] I. Machon-Gonzalez, H. Lopez-Garcia, J. Rodriguez-Iglesias, E. Maranõn-Maison, L. Castrillon-Pelaez, and Y. Fernandez-Nava, "Comparing feed-forward versus neural gas as estimators: application to coke wastewater treatment," *Environmental Technology*, vol. 34, no. 9, pp. 1131–1140, 2013.

[12] D. Erdogmus, J. Cho, J. Lan, M. Motter, and J. C. Principe, "Adaptive local linear modelling and control of nonlinear dynamical systems," in *Intelligent Control Systems Using Computational Intelligence Techniques*, A. Ruano, Ed., pp. 119–152, IET, 2005.

[13] B. Arnonkijpanich, A. Hasenfuss, and B. Hammer, "Local matrix adaptation in topographic neural maps," *Neurocomputing*, vol. 74, no. 4, pp. 522–539, 2011.

[14] I. Machón-González, H. López-García, and J. Luís Calvo-Rolle, "A hybrid batch SOM-NG algorithm," in *Proceedings of the 6th IEEE World Congress on Computational Intelligence, WCCI 2010—2010 International Joint Conference on Neural Networks (IJCNN '10)*, pp. 1–5, July 2010.

[15] M. J. Crespo-Ramos, I. Machón-González, H. López-García, and J. L. Calvo-Rolle, "Detection of locally relevant variables using SOM-NG algorithm," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 8, pp. 1992–2000, 2013.

[16] L. Chen and K. S. Narendra, "Nonlinear adaptive control using neural networks and multiple models," *Automatica*, vol. 37, no. 8, pp. 1245–1255, 2001.

[17] K. S. Narendra and S. Mukhopadhyay, "Adaptive control using neural networks and approximate models," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 475–485, 1997.

[18] V. F. Montagner, V. J. Leite, R. C. Oliveira, and P. L. Peres, "State feedback control of switched linear systems: an LMI approach,"

Journal of Computational and Applied Mathematics, vol. 194, no. 2, pp. 192–206, 2006.

[19] M. T. Hagan, H. B. Demuth, and O. De Jesús, "An introduction to the use of neural networks in control systems," *International Journal of Robust and Nonlinear Control*, vol. 12, no. 11, pp. 959–985, 2002.