sensors

MDPI

Article

Deep Learning for Robust Adaptive Inverse Control of Nonlinear Dynamic Systems: Improved Settling Time with an Autoencoder

Nuha A. S. Alwan [1] and Zahir M. Hussain [2,*]

[1] College of Engineering, University of Baghdad, Baghdad 10017, Iraq; Email: n.alwan@ieee.org
2 School of Engineering, Edith Cowan University, Joondalup, WA 6027 Australia
* Correspondence: z.hussain@ecu.edu.au or zmhussain@ieee.org

Abstract: An adaptive deep neural network is used in an inverse system identification setting to approximate the inverse of a nonlinear plant with the aim of constituting the plant controller by copying to the latter the weights and architecture of the converging deep neural network. This deep learning (DL) approach to the adaptive inverse control (AIC) problem is shown to outperform adaptive filtering techniques and algorithms normally used in adaptive control, especially when the plant is nonlinear. The deeper the controller the better the inverse function approximation, provided that the nonlinear plant have an inverse and that this inverse can be approximated. Simulation results prove the feasibility of this DL-based adaptive inverse control scheme. The DL-based AIC system is robust to parameter change of the nonlinear plant in that, under such change, the plant output reassumes the value of the reference signal considerably faster than with the adaptive filter counterpart of the deep neural network. Settling times and rise times of the step response are shown to improve in the DL-based AIC system.

Keywords: deep neural network; deep learning controller; nonlinear plant; adaptive inverse control; robust control; autoencoder; computational complexity; sensor control.

Citation: Alwan, N. A. S.; Hussain, Z. M. Deep Learning for Robust Adaptive Inverse Control of Nonlinear Dynamic Systems: Improved Settling Time with an Autoencoder.

Academic Editor:

Received:
Accepted:
Published:

1. Introduction

Efficient control of nonlinear dynamic systems has been dealt with extensively in the literature. As a control problem, the essence is to make the nonlinear system output track a reference trajectory. Many approaches require accurate knowledge of the nonlinear plant in order to linearize the plant dynamics around an operating point such as in gain scheduling [1]. Others require an accurate dynamic inversion of the plant such as feedback linearization methods [2]. These methods require that a considerably accurate model or inverse model of the plant exist, in addition to their computational complexity. Adaptive control is the logical solution to the unknown plant control problem [3-5]. An adaptive filter uses its input and a desired response to modify its internal parameters such that a function of the error between its actual and desired output is minimum. Just as adaptive control of linear plants requires adaptive filtering techniques; adaptive control of nonlinear plants requires nonlinear adaptive filtering methods which are generally achieved by using neural networks [6].

Precisely, adaptive inverse control (AIC) will be considered in the present work as it is particularly simple to implement. AIC was first devised by Widrow and Walach [7] as a result of approaching the discipline of adaptive control from the point of view of adaptive signal processing. The basic scheme for linear systems is shown in Figure 1, in which an adaptive transversal filter approximates the inverse of the unknown plant when connected with the latter in an inverse modeling setting [3]. This adaptive filter is copied to form the controller in the figure. During normal operation, the plant output will track the reference signal which is input to the controller, the output of which is the driving signal to the plant. The delay $z^{-\Delta}$ makes up for possible internal delay in the plant. The control aim is to guarantee that the reference signal can be tracked accurately with minimum delay. Ideally, the cascade of controller and plant forms a unit-magnitude transfer function enabling the plant output to be an exact but delayed version of the reference in case of no noise and/or disturbance. The plant must be stable and minimum phase for its inverse to be stable. Methods exist for controlling plant noise and disturbance [6, 7] without compromising the control of plant dynamics. As these two control procedures are conveniently separable, these methods will not be considered further in the present work whose focus is on nonlinear adaptive control strategies.

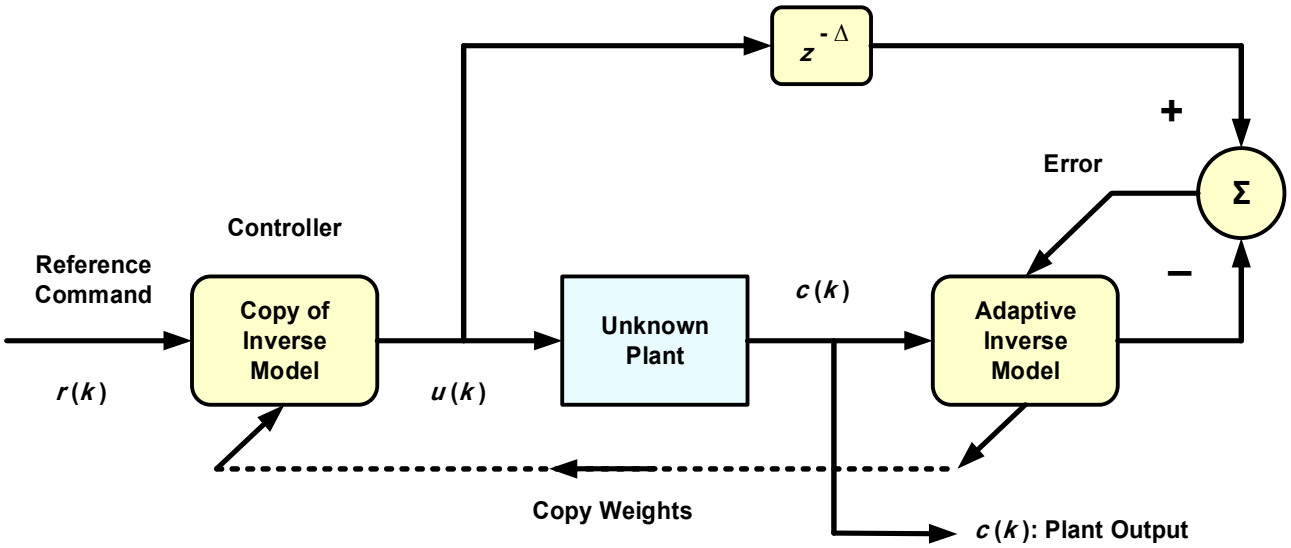

Figure 1. Adaptive inverse model control of an unknown plant [3].

It is important to note that this is feed-forward control since no feedback is employed except the feedback inherent in the adaptive algorithm. This adaptive algorithm could be, for example, the least mean square (LMS) algorithm [3] for the linear adaptive control case. When the plant is nonlinear, a nonlinear adaptive filter is needed that can be realized by a neural network. The adaptive algorithm would then be the back-propagation (BP) algorithm, for example [8]. The updating of the adaptive inverse

model in Figure 1 enables real time control (RTC) to continuously track any parameter change in the plant, resulting in robustness to parameter change. Otherwise, a non-adaptive neural inverse model control system would be very sensitive to parameter changes [9]. As hinted at in [10], RTC can also handle time-varying or predicted parameter change via triggered (possibly periodically) actions to activate updating. This procedure could markedly reduce the computational burden required in continuous updating or tracking.

In [6] and [11], AIC was successfully performed for linear and nonlinear systems using shallow neural networks. It is known that any smooth function (linear or nonlinear) can be approximated by a shallow neural network with a sufficient number of neurons in the hidden layer. Using deeper networks with more than one hidden layer, however, results in more efficient function approximation since deep neural networks with standard architectures represent compositionality of functions [12]. Deep neural networks with more than one hidden layer could not be trained successfully without the deep learning (DL) techniques arrived at in the mid-2000s. These techniques include the use of rectified linear units (ReLU) as activations functions, the use of the dropout technique during training and the advent of efficient computing hardware such as graphical processing units (GPU) [13].

In [9], AIC of nonlinear systems is implemented using a NN with two hidden layers and sigmoidal activation functions. In contrast, the present work uses up to four hidden layers with ReLU activation functions to avoid the vanishing gradient problem that would otherwise occur if sigmoidal activation functions were used in deep NNs.

A similar problem is treated in [14] but with non-adaptive inverse control. Recently, Lyapunov-based DL adaptive online control of nonlinear systems has been carried out rather than AIC [15]. Specifically, a DL controller uses restricted Boltzmann machine (RBM) to initialize the weight values and Lyapunov stability method to update a two-hidden-layer NN controller connected to the nonlinear plant in a negative feedback control loop. AIC, on the other hand, involves open loop control and is different from the negative feedback control in [15].

The aim of the present work is to achieve simple yet robust adaptive control using deep NNs as controllers with dynamics that are the inverse of those of the generally unknown nonlinear plant. Since theory is still being developed to guide the analysis of control systems with nonlinear plants, one of the main objectives of this work is to simply show how nonlinear AIC using DL differs from its linear counterpart, when both are used to control a nonlinear plant.

It will also be demonstrated that the deeper the NN controller is, the more robust the control system is to parameter change.

The organization of the paper is as follows. Section 2 introduces general mathematical models for nonlinear discrete-time plants. Section 3 explains deep neural networks and their training by the BP algorithm. The DL-based AIC nonlinear control system is illustrated and explained in Section 4, and simulation results are presented in Section 5. Finally, Section 6 concludes the paper.

2. Dynamic Nonlinear Discrete-Time Plants

Four models of discrete-time plants governed by nonlinear difference equations are described in [16] and summarized below. The input and output of the plant are denoted by $u(k)$ and $c(k)$ respectively, where k is the discrete time index. It is assumed that the present output sample is dependent on n past outputs and m present and past inputs, where $m \leq n$. Such a system is said to be dynamic, in contrast to a static or memory-less system. In the following model equations, $f(\cdot)$ and $g(\cdot)$ are nonlinear functions and α's and β's are constants.

Model I:

$$c(k) = \sum_{i=1}^{n} \alpha_i c(k-i) + g[u(k), u(k-1), \dots, u(k-m+1)]$$

Model II:

$$c(k) = f[c(k-1), c(k-2), \dots, c(k-n)] + \sum_{i=0}^{m-1} \beta_i u(k-i)$$

Model III:

$$c(k) = f[c(k-1), c(k-2), \dots, c(k-n)] + g[u(k), u(k-1), \dots, u(k-m+1)]$$

Model IV:

$$c(k) = f[c(k-1), c(k-2), \dots., c(k-n);\ u(k), u(k-1), \dots, u(k-m+1)] \quad (1)$$

The nonlinear plant functions $f(\cdot)$ and $g(\cdot)$ are assumed to be unknown when adaptive inverse control of the plant is performed. It was found in [16] that Model II is particularly suitable for control problems. If the nonlinear plant is bounded-input-bounded-output (BIBO) stable, its model must also possess this property. For Model I, this implies that the characteristic equation, comprising a polynomial in z of degree n with α's as the coefficients, must have all its roots inside the unit circle in the z-plane. For the other models, the stability conditions are not so simple, and therefore, they constitute an important research area.

3. Deep Neural Networks

Multi-layer neural networks (NN) can be regarded as nonlinear maps with their weights as parameters [16]. They can thus be used as subsystems to constitute controllers for nonlinear dynamical plants. In the present case, the NN or the controller models the inverse of the plant dynamics as explained in the introduction and Figure 1. Dynamics are introduced in the NN via tapped delay lines (TDL) at its input. A multi-layer NN with more than one hidden layer is considered deep [13]. The weights are adapted to minimize a function of the error between the NN output and the desired output according to a training algorithm such as the BP algorithm which is based on steepest descent. Training and convergence of NN's are heavily dependent on initial conditions of the weights, as well as on the nature of their input data. In this work, neural networks will be denoted by the notation $\mathcal{N}_{(I,L):J:K:\dots}$ where I is the number of input nodes, L is the number of output nodes (or neurons), J is the number of neurons in the first hidden layer, K is the number of neurons in the second hidden layer, and so on. A deep NN with two hidden layers is illustrated in Figure 2. The NN in this figure generally represents a multiple-input multiple-output (MIMO) system. The outputs of all hidden

and output nodes are subjected to nonlinear activation functions. Only the output nodes are permitted to have a linear or nonlinear activation function depending on the application and the required range of the NN outputs. The output vector Y can be expressed as:

$$Y_{L\times 1} = \psi[M_{L\times K}\,\Phi\{H_{K\times J}\,\Phi(W_{J\times I}\,X_{I\times 1} + B_{J\times 1}) + \beta_{K\times 1}\} + b_{L\times 1}] \quad (2)$$

where X is the input vector, W, H, and M are the weight matrices of the first hidden, second hidden and output layers respectively. I, J, K and L are the numbers of nodes of the input, first hidden, second hidden and output layers respectively. B, β and b are the corresponding biases. The activation functions in Equation 2 operate point-wise on the respective vectors. The BP algorithm, which is based on gradient descent, is summarized by the following set of equations with reference to Figure 2. It is assumed that nonlinear activation functions are present in the hidden layers, whereas the output nodes are linear.

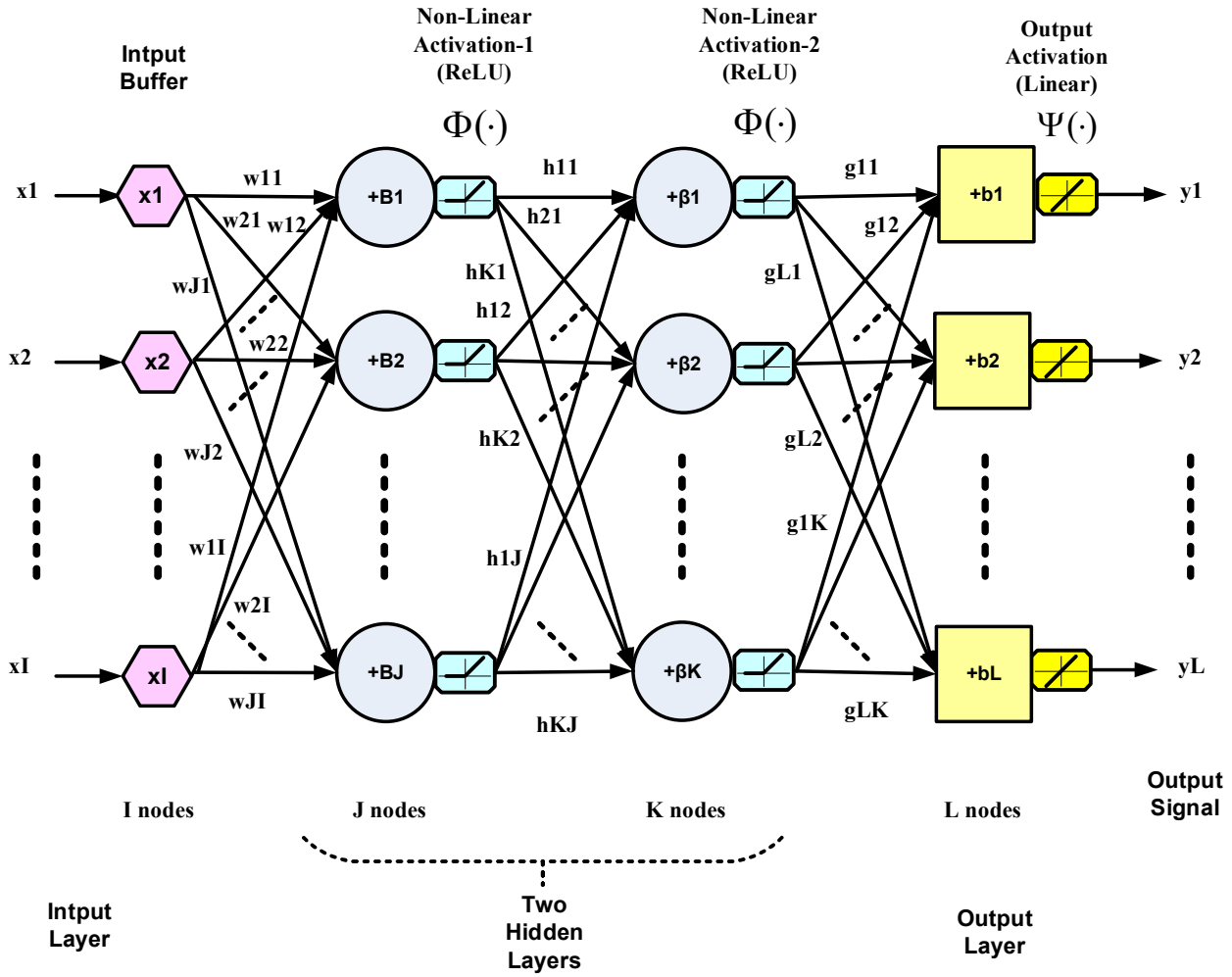

Figure 2. A generic diagram for a three-layer deep NN with two hidden layers.

$$
\begin{aligned}
&g_{lk} \leftarrow g_{lk} + \Delta g_{lk},\ \ l = 1, \cdots, L;\ \ k = 1, \cdots, K. \\
&\text{where } \Delta g_{lk} = \alpha(d_l - y_l)y_k = \alpha\delta_l y_k \\
&h_{kj} \leftarrow h_{kj} + \Delta h_{kj},\ \ k = 1, \cdots, K;\ \ j = 1, \cdots, J. \\
&\text{where } \Delta h_{kj} = \alpha\delta_k y_j, \text{with } \delta_k = [\textstyle\sum_l g_{lk}\,\delta_l] \cdot \Phi'(v_k) \\
&w_{ji} \leftarrow w_{ji} + \Delta w_{ji},\ \ j = 1, \cdots, J;\ \ i = 1, \cdots, I. \\
&\text{where } \Delta w_{ji} = \alpha\delta_j x_i, \text{with } \delta_j = \left[\textstyle\sum_k h_{kj}\,\delta_k\right] \cdot \Phi'(v_j)
\end{aligned}
\tag{3}
$$

In Equations (3) above, α is the BP learning rate, the d's are the correct outputs at the output layer needed for supervised training, the v's are the activation function inputs, the y's are the activation function outputs for output and hidden layers, the x's are the inputs to the input layer, and $\Phi'(\cdot)$ is the derivative of the nonlinear activation function $\Phi(\cdot)$. In the present application, the training data, consisting of the correct outputs (d's) and inputs (x's), would be values of the control signal $u(k)$ and the plant output signal $c(k)$ respectively.

Training deep neural networks is usually a difficult task due to different gradient magnitudes in lower and higher layers, the curvature of the objective or error function with its numerous local minima, and lack of acceptable generalization caused by the large number of parameters to be updated. The following sub-section highlights the use of NN autoencoders (AE) to pre-train deep NNs.

3.1. Deep Neural Network Initialization Using Autoencoders

Well-designed random weight initialization of a deep NN is crucial for proper convergence during training and subsequent operation. Poorly initialized networks are very difficult, if not impossible, to train. To initialize the deep NN weights, each layer can be pre-trained separately using an auxiliary error function before the whole network is trained by the usual methods such as stochastic gradient descent that employs back-propagation for instance. This method of unsupervised pretraining of one layer at a time prevents difficulties of full deep NN supervised training. The NN architecture adopted to train each layer separately is called an autoencoder (AE). NN autoencoders have their own inputs as correct outputs during training [17-19]. In this work, we will use simple NN autoencoders that have only one hidden layer. To limit the ensuing computational complexity, the present work will not consider deep AEs. The procedure is as follows [19]: the NN initial weights are determined by first greedily training a sequence of shallow AEs one layer at a time using unsupervised data. A shallow AE consists of an encoder and a decoder. Only the encoder weights of the trained AE are retained and considered to be the initial weight of that layer. The final layer is trained using supervised data. Finally, the complete network is fine-tuned using supervised data according to the back-propagation algorithm. AEs not only improve system initialization, but also lead to better generalization and prevent overfitting [20, 21].

Classical momentum is another necessary technique to improve deep NN training. Even well-initialized networks can perform poorly without momentum. This concept is explained next.

3.2. Training with Momentum

Stochastic gradient descent can be accelerated by iteratively accumulating a velocity vector in the direction of reduction of the error function. If we call the velocity vector by v, a weight update of Equations (3) will change from

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

to:

$$v \leftarrow \mu\, v + \Delta\, w_{ji}$$
$$w_{ji} \leftarrow w_{ji} + v \quad (4)$$

where μ is the momentum factor, and assuming zero initial value of v. The acceleration of the gradient descent algorithm by using momentum does not come at the expense of stability; contrary to acceleration by increasing the learning rate α. Setting μ equal to zero in Equations (4) reduces the equation to its original form. For convex objective functions, the momentum-based method will outperform the SGD-based method particularly in the transient stage of optimization, and is capable of accelerating directions of low-curvature in the objective or error function [17].

4. DL-Based Adaptive Inverse Control of a Nonlinear Plant

Several assumptions are made to achieve satisfactory control of nonlinear plants. For example, it is assumed that the nonlinear plant is BIBO stable. They are also assumed to be non-minimum phase, meaning that they have stable inverses, and first of all, that such inverses exist. Figure 3 shows a general block diagram of the AIC system for controlling nonlinear plants using deep NNs.

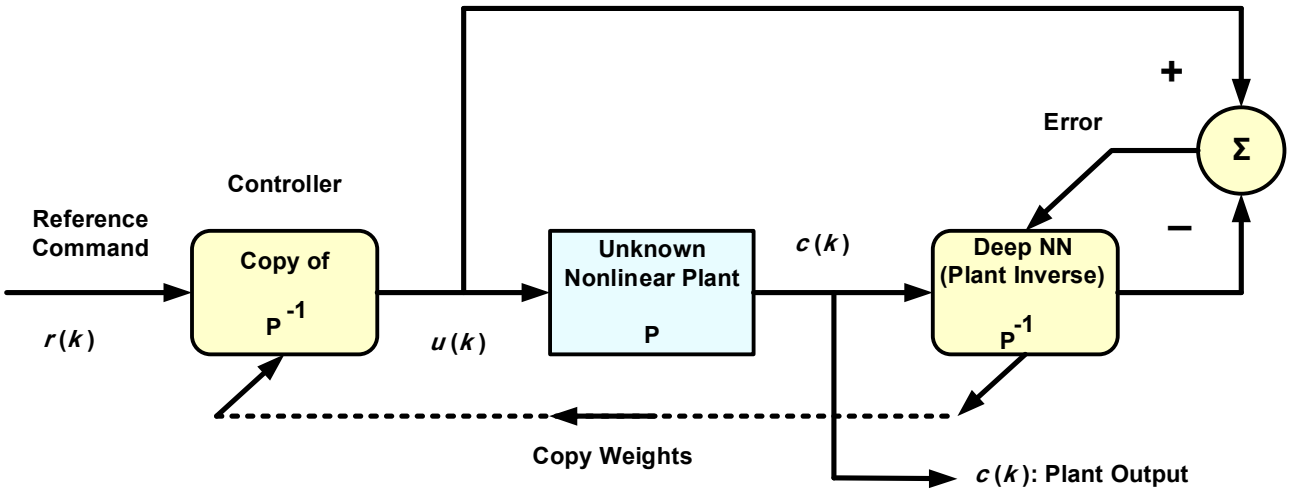

Figure 3. DL-based adaptive inverse control of a nonlinear plant.

As explained in Section 3, weight initialization of the deep NN is achieved with the aid of pre-trained AEs. To train AEs before normal system operation, we need the input-correct output training data of the deep NN. However, in the present application,

the correct output is not readily available due to the adaptive nature of this control system. Rather, the correct output builds up gradually during online operation. A solution to this problem is to use offline measurements as training data for the AEs, by inputting training data to the unknown nonlinear system and measuring its output. Since the deep NN is meant to inverse-model the nonlinear system or plant, the measured plant output is used as input training data to the NN, and the chosen input data will constitute the NN correct output. All this is done offline and the NN is thereby pre-trained using AEs.

It will be shown that the reference command signal is well tracked for this adaptive system rendering it robust to parameter variations, especially when the NN is deep enough. If a parameter of the nonlinear plant changes causing a fall (rise) in the plant output, the adaptive process compensates for this fall (rise) causing the control signal to increase (decrease) to re-attain the original plant output that tracks the reference. The plant parameter change could be abrupt due, for example, to failure of actuators or sensors [22, 23]. Soft faults such as actuator or sensor biases are also frequently encountered in industry [23]. If the above system is a temperature control system as in [9] for instance, and if the internal fan actuator abruptly changes speed to a higher value due to soft faults or failures, the temperature will drop and so will the plant output voltage. The control signal will then increase such that the system adaptively returns to its original state, and the faulty speed change of the fan has no effect on the tracking performance. Such abrupt parameter variation will be simulated and discussed in the following section on results, as they are more challenging than slowly-varying parameter changes as far as tracking in adaptive control systems is concerned.

5. Simulation Results

Simulations are carried out in MATLAB (academic licenses 30904939 and 40635944). The suggested nonlinear AIC system of Figure 3 is simulated and the results presented. The deep NN controller is then compared to a linear controller to show the difference in performance. The nonlinear plant to be controlled is governed by the nonlinear difference equation given below in accordance with Model II of Equation 1.

$$c(k) = f[c(k-1), c(k-2)] + u(k) \tag{5}$$

where the nonlinear function $f(\cdot)$ is substituted for as follows:

$$c(k) = \frac{c(k-1) \cdot c(k-2)}{1 + c^3(k-1)} + u(k) \tag{6}$$

Nonlinear plants of this form were introduced in the literature by Narendra and Parthasarathy [16]. The reference signal is taken as a square wave spanning 1000 samples. The NN used is $\mathcal{N}_{(\mathbf{4},\mathbf{1}):\mathbf{5}:\mathbf{5}:\mathbf{5}:\mathbf{5}}$ indicating four hidden layers, each having five nodes with ReLU activation functions, one linear output node and an input TDL length of 4. The BP learning rate is α=0.02. BP minimizes the quadratic cost function. As in

Figure 3, the controller is a copy of the approximate inverse plant model achieved by the adaptive neural network. Figure 4 shows the square reference and plant output signals. Figure 5 shows the corresponding control signal. It is clear that with each step change, the NN re-adapts allowing the plant output to track the reference command signal which is applied as input to the controller. The response to the step changes is oscillatory with overshoot as shown. Apart from the adaptation region near the step changes, the plant output and reference signals are indistinguishable from each other. Figures 6 and 7 show the reference and plant output signals and the control signal respectively, when an abrupt plant parameter change occurs at the 600th sample. The parameter change is assumed to render the following plant equation:

$$c(k) = \frac{c(k-1) \cdot c(k-2)}{1+4*c(k-1)^3} + u(k) \quad (7)$$

A unity parameter in the denominator has been increased to 4. This sudden change will cause an abrupt but temporary decrease in the plant output and an increase in the control signal at the 600th sample, as is clear from Figures 6 and 7. The settling time needed after parameter change is 18 time samples only. The settling time is taken as the time needed for the plant output to reach and stay within 2% of its final value. The parameter changes have little effect on the tracking performance; they are rapidly compensated for.

A further parameter change could result in the following plant equation:

$$c(k) = \frac{c(k-1) \cdot c(k-2)}{1+4 \cdot c(k-1)^3} + 2 \cdot u(k) \quad (8)$$

The above parameter change, when effected abruptly, will cause an abrupt but temporary increase in plant output and, therefore, a decrease in the control voltage at the 600th sample as in Figures 8 and 9. Again, the change is compensated and good tracking is obtained.

To further demonstrate the benefit of using DL, we operate the same nonlinear control system (Figure 3) but using an adaptive FIR filter in place of the deep NN. The adaptive FIR filter uses a four-tap TDL which is the same TDL length used in the deep NN. The results are illustrated by Figures 10 and 11. The parameter change at the 600th sample is in accordance with Equation (7). The settling time is 58 time samples when using an LMS learning rate of 0.02 as in the BP training of the deep NN. This settling time is considerably higher than that obtained with DL, a result that is to be anticipated because of the linear structure of the inverse model after convergence. The linearity implies the presence of structural errors in modeling (or inverse modeling) the nonlinear system [24].

The advantage of using DL comes at the expense of some additional computational complexity. The number of multiplications needed with DL is $\mathcal{O}\{N^2\}$ with N denoting the average number of nodes per layer, whereas that needed with the adaptive filter is $\mathcal{O}\{N\}$ only.

NN training methods based on gradient descent may cause the solution to converge to a local rather than the global minimum. As a first step, and for each comparison between systems or between plant parameter variations, initial conditions for the weight values

were adjusted to yield the best performance expected from the nonlinear system when a specific scenario is to be compared with others. This is done by trial and error to discover the general trend or behavior which shortly becomes manifest.

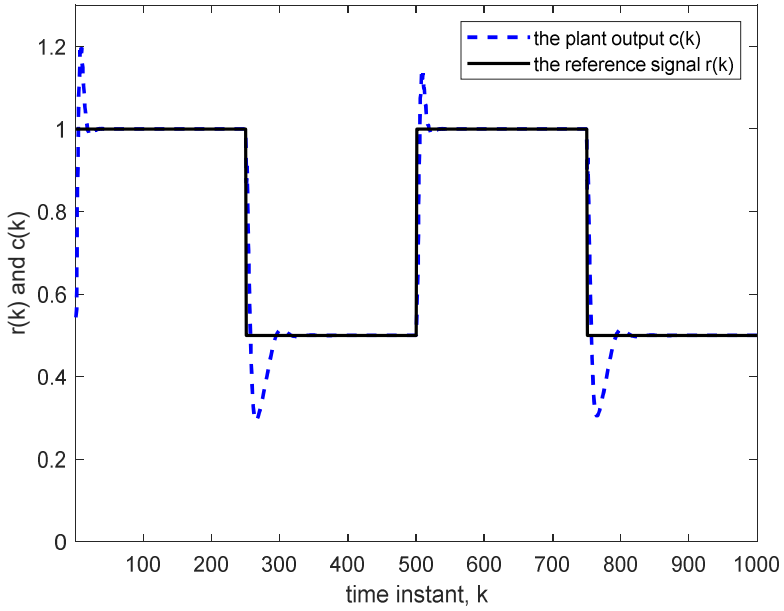

Figure 4: Reference and plant output of the AIC system with $N_{(4,1):5:5:5:5}$.

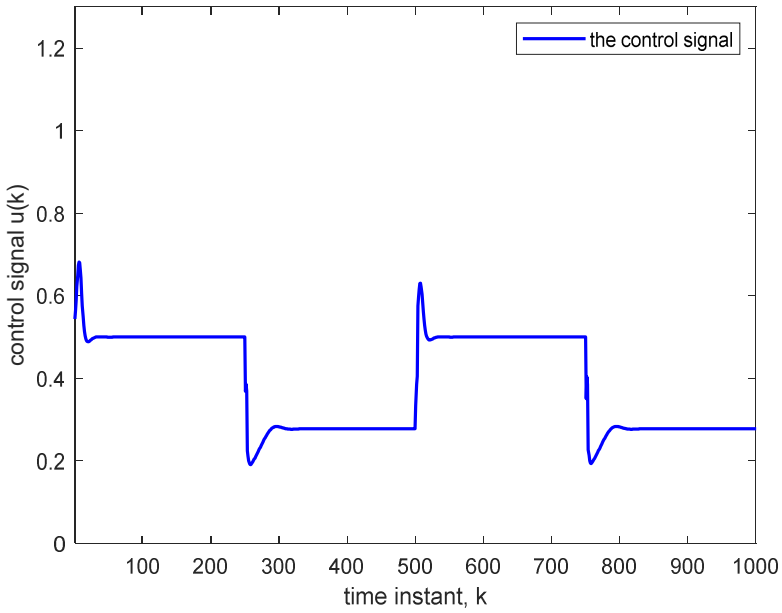

Figure 5: Control signal corresponding to Figure 4.

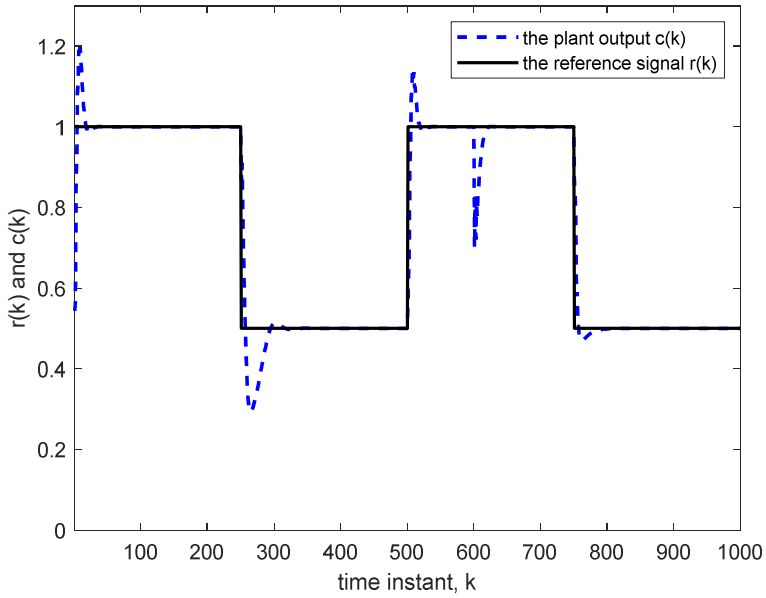

Figure 6: Reference and plant output with a parameter change at the 600th sample according to Equation (7), using $N_{(4,1):5:5:5:5}$.

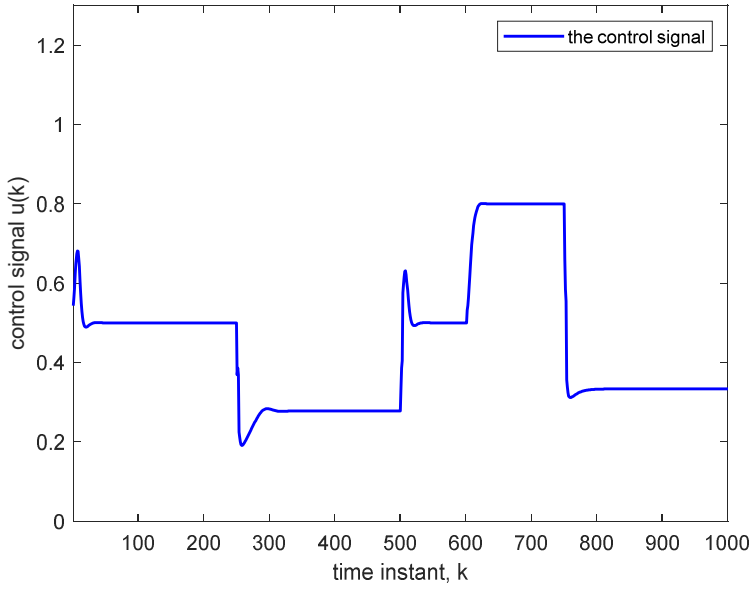

Figure 7: Control signal corresponding to Figure 6.

It is worth noting that the compensating change in control voltage upon parameter change, demonstrated by Figures 7, 9, and 11, can be used in control applications such as temperature measurement using thermoresistive sensors [25]. The integration of sensors in control and automation draws on several sensor functions that are featured in sensor networks, fault tolerant control, intelligent sensors, robot sensing, etc. [26].

In the temperature measurement application in [25], however, the thermoresistive sensor itself is the nonlinear plant whose process variable (temperature) is to be controlled and fixed to a reference value enabling the measurement of the surrounding

temperature. The resistance of such a sensor depends on temperature, thereby changing the electrical signal associated with it, which is fed back and compared with the reference. To measure the surrounding temperature, the role of feedback control is to keep the sensor temperature constant and equal to a reference temperature despite the temperature change in the surroundings. Therefore, the resulting variation or change in the control voltage is used to measure the surrounding temperature. The DL-based AIC control system of Figure 3 can be a suitable substitute for the feedback control used in this sensor control problem of temperature measurement; the justifications are ease of implementation and avoidance of techniques used with nonlinear plant control such as feedback linearization.

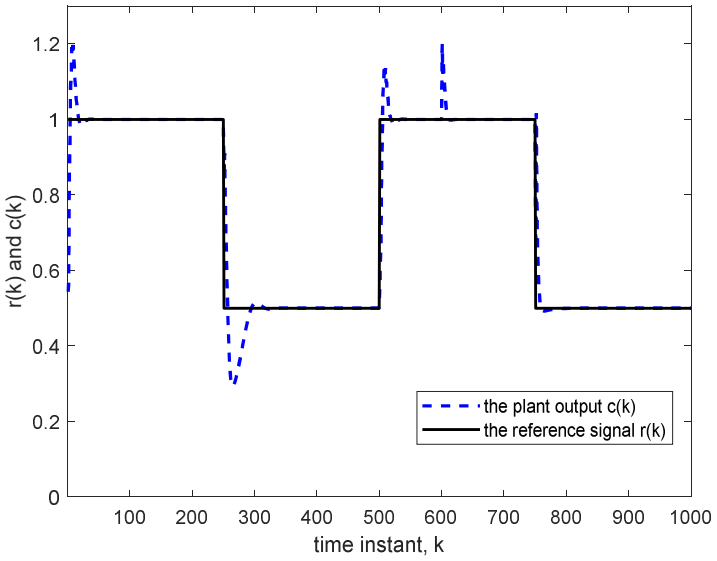

Figure 8: Reference and plant output with a parameter change at the 600th sample according to Equation (8), using $N_{(4,1):5:5:5:5}$.

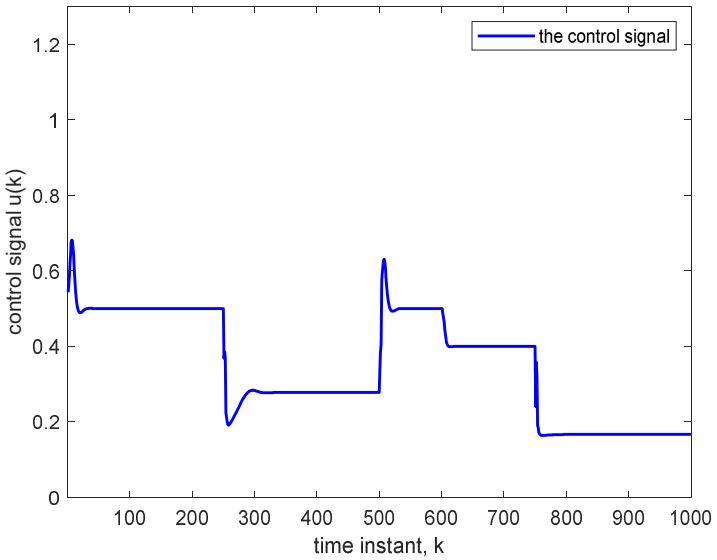

Figure 9: Control signal corresponding to Figure 8.

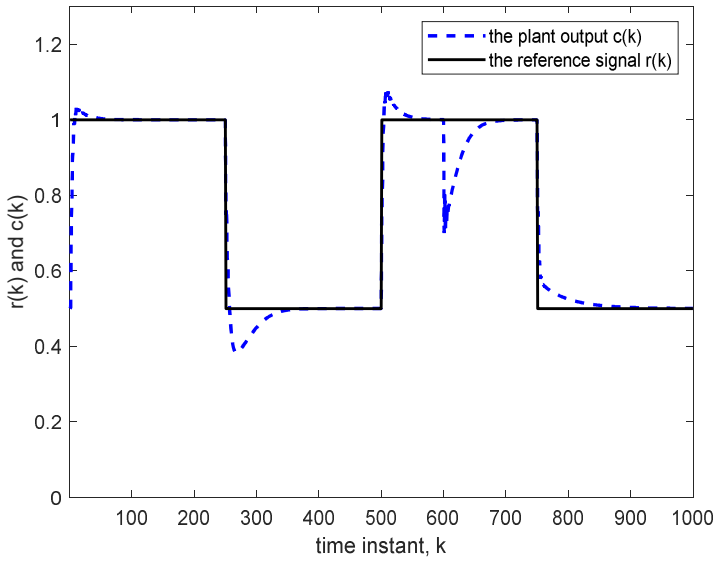

Figure 10. Reference and plant output with a parameter change at the 600th sample, using adaptive filter as the inverse model of the nonlinear plant.

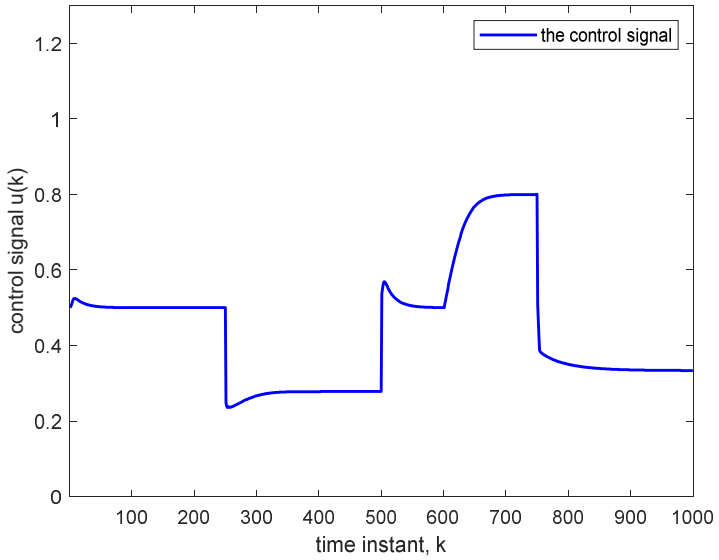

Figure 11: Control signal corresponding to Figure 10.

Table 1 shows the dependence of the settling time after parameter change, in time samples, on the number of hidden layers without and with AE initialization. In AE training, it is common practice to employ BP that minimizes the quadratic cost function, using ReLU activations in the nodes [20]. As explained in Section 4, offline measurements to train the AEs are obtained by measuring input and output of the nonlinear plant. The input to the plant will constitute the correct output, and therefore it can be chosen to resemble the control signal obtained in Figure 5. It can simply be chosen as a square wave (free of transients) with step values comparable to those of the

figure. Then, the measured plant output will constitute the input training data of the AEs.

In reference to **Table 1**, without AE, two and three hidden layers show similar performances that are intermediate between the case of four hidden layers and that of the adaptive linear filter. When AEs are used for initializing the deep NN, better results are obtained regarding settling time as can be seen from Table 1. This improvement is due to proper weight initialization which speeds up training by guaranteeing first that the stochastic gradient solution is close to a suitable local minimum. The symbol α_{ae} is used to denote the AE learning rate.

Table 2 demonstrates the benefit of using a momentum factor μ on the rise time of the step response after parameter change. The rise time of a step response is defined as the time needed for the response to rise from 10% to 90% of the final value. It is found that training with a momentum factor reduces the rise time but increases the settling time without causing instability.

For control systems with underdamped step response (which is often encountered in this work), the rise time is sometimes defined as the time required by the response to reach the final value during its first cycle of oscillation. The smaller rise time due to using momentum is clear from the Table 2. This is important in control applications; if the rise time is too long, the system may be operating with the process variable below the optimum for too long. This may have consequences depending on the particular control application. For example, the consequence could be failure to apply sufficient braking force quickly enough.

6. Conclusions

In this work, DL-based adaptive inverse control of nonlinear dynamic systems is achieved. Simulations indicate efficient online (tracking) control. It is evidently possible to generate a control signal that ensures reliable inverse modeling and control of a nonlinear plant with unknown dynamics using a deep neural network to learn the inverse model. The resulting adaptive control system is robust to parameter changes. Settling times and rise times of the step response after parameter change are shown to improve even further when using autoencoder initialization of the neural network weights, and including momentum in the cost function minimization by backpropagation. The deeper the NN that learns the inverse model, the more accentuated the robustness of the adaptive control system to parameter change.

Author Contributions: The authors contributed equally to this work.

Funding: This project was partially funded by Edith Cowan University via the ASPIRE Program.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All types of data were generated using mathematical equations.

Acknowledgments: The authors thank Edith Cowan University for supporting this project.

Conflicts of Interest: The authors declare that no conflict of interest is associated with this work.

Table 1: Settling time after nonlinear-plant parameter change [Equation (7)] versus number of deep NN hidden layers, and settling time when using a linear FIR adaptive filter. α=0.02, α_{ae}=0.1, μ=0.

Type of controller		Settling time (in number of sample intervals)	
		Without AE initialization	With AE initialization
Deep NN	$\mathcal{N}_{(4,1):5:5:5:5}$ (4 hidden layers)	18	10
	$\mathcal{N}_{(4,1):5:5:5}$ (3 hidden layers)	25	13
	$\mathcal{N}_{(4,1):5:5}$ (2 hidden layers)	25	17
Linear FIR adaptive filter		58	

Table 2: Momentum effect on settling time and rise time after nonlinear-plant parameter change [Equation (7)] versus number of deep NN hidden layers, and when using a linear FIR adaptive filter. α=0.02, α_{ae}=0.1, different momentum factor values (μ). AE is used for initialization.

Type of controller		Settling time (in number of sample intervals)		Rise time (in number of sample intervals)	
		μ=0	μ=0.4	μ=0	μ=0.4
Deep NN	$\mathcal{N}_{(4,1):5:5:5:5}$ (4 hidden layers)	10	18	9	7
	$\mathcal{N}_{(4,1):5:5:5}$ (3 hidden layers)	13	30	12	9
	$\mathcal{N}_{(4,1):5:5}$ (2 hidden layers)	17	30	15	11
Linear FIR adaptive filter		58		50	

References:

1. W. J. Rugh and J. S. Shamma, "A survey of research on gain scheduling", Automatica 36, 2000, pp. 1401-1425.
2. H. J. Marquez, Nonlinear Control Systems: Analysis and Design, Wiley Interscience, 1st Edition, 2008.
3. B. Widrow and S. D. Stearns, Adaptive Signal Processing, Prentice-Hall, 1985.
4. N. T. Nguyen, Model-Reference Adaptive Control, a Primer. 2018, Springer, Cham, Switzerland.
5. P. Ioannou and B. Fidan, Adaptive Control Tutorial. 2006, SIAM, Philadelphia, USA.
6. G. L. Plett, "Adaptive inverse control of linear and nonlinear systems using dynamic neural networks", IEEE Transactions on Neural Networks, vol. 14, no. 2, 2003, pp. 360-376.
7. B. Widrow and E. Walach, Adaptive Inverse Control, a Signal Processing Approach. 2008 IEEE, John Wiley and Sons, Inc., Hoboken, New Jersey.
8. D. Rumelhart, G. Hinton and R. Williams. Learning representations by back-propagating errors. Nature 1986, 323, 533–536.
9. R. Hedjar, "Online adaptive control of nonlinear plants using neural networks with application to temperature control system", Journal of King Saud University-Computer and Information Sciences, vol. 19, 2007, pp. 75-94.
10. N. A. S. Alwan and Z. M. Hussain, "Deep learning control for digital feedback systems: improved performance with robustness against parameter change", Electronics, 10. 11, 2021, p. 1245.
11. J. L. Calvo-Rolle, O. Fontenla-Romero, B. Perez-Sanchez and B. Guijarro-Berdinas, « Adaptive inverse control using an online learning algorithm for neural networks », Informatica, vol. 25, no. 3, 2014, pp. 401-414.
12. H. Mhaskar, Q. Liaou and T. Poggio, "When and why are deep networks better than shallow ones?", Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI-17), vol. 31, no. 1, Feb 18, 2017.
13. P. Kim, MATLAB Deep Learning, 2017, Springer.

14. R. J. Rajesh, R. Preethi, P. Mehata and B. Jaganatha-Pandian, "Artificial neural network based inverse model control of a nonlinear process". In 2015 International Conference on Computer, Communication and Control (IC4), 2015, Sep 10, IEEE, pp. 1-6.
15. A. M. Zaki, A. M. El-Nagar, M. El-Bardini and F. A. S. Soliman, « Deep learning controller for nonlinear system based on Lyapunov stability criterion", Neural Computing and Applications, vol. 33, no. 5, 2021, pp. 1515-1531.
16. K. S. Narendra and K. Parthasarathy "Identification and control of dynamical systems using neural networks", IEEE Transactions on Neural Networks, vol. 1, no. 1, March, 1990.
17. I. Sutskever, J. Martens, G. Dahl and G. Hinton, "On the importance of initialization and momentum in deep learning", International Conference on Machine Learning, 26 May 2013, pp. 1139-1147, PMLR.
18. I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, MIT, 2016.
19. Y. Bengio, P. Lamblin, D. Popovici and H. Larochelle, "Greedy layer-wise training of deep networks", In Advances in Neural Information Processing Systems, 2007.
20. M. F. Ferreira, R. Camacho and L. F. Teixeira, "Using autoencoders as a weight initialization method on deep neural networks for disease detection", BMC Medical Informatics and Decision Making, vol. 20, no. 5, August, 2020, pp. 1-8.
21. D. Erhan, A. Courville, Y. Bengio and P. Vincent, "Why does unsupervised pre-training help deep learning?" In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, 2010, pp. 201-208.
22. K. S. Narendra, J. Balakrishnan and M. K. Ciliz, "Adaptation and learning using multiple models, switching and tuning", IEEE Control System Magazine, vol. 15, no. 3, 1995, pp. 37-51.
23. A. P. Deshpande, S. C. Patwardhan and S. S. Narasimhan, "Intelligent state estimation for fault tolerant nonlinear predictive control", Journal of Process Control, vol. 19, no. 2, 2009, pp. 187-204.
24. J. Schoukens and L. Ljung, "Non-linear system identification: a user-oriented roadmap", IEEE Control System Magazine, vol. 39, no. 6, 2019, pp. 28-99.
25. M. A. Moreira, A. Oliviera, C. E. T. Dorea, P. R. Barros and J. S. de Rocha Neto, "Sensor characterization and control of measurement systems based on thermoresistive sensors via feedback linearization", Advances in Measurement Systems, M. K. Sharma (Ed.), IntechOpen, 2010.
26. R. Morales-Herrera, A. Fernandez Caballero, J. A. Somolinos and H. Sira-Ramirez, "Integration of sensors in control and automation systems", Editorial, Journal of Sensors, 2017.