Sensors **2007**, *7*, 1509-1529

sensors

ISSN 1424-8220

www.mdpi.org/sensors

Full Paper

Self-Calibration and Optimal Response in Intelligent Sensors Design Based on Artificial Neural Networks

José Rivera [1,2,*], Mariano Carrillo [1], Mario Chacón [1], Gilberto Herrera [2] and Gilberto Bojorquez [1]

[1] División de Estudios de Posgrado e Investigación del Instituto Tecnológico de Chihuahua. Ave. Tecnológico No. 2909, Chihuahua Chih. México 31310; Tel. (614) 413 7474; Fax. (614) 413 5187, E-mail:{jrivera,mcarrillo,mchacon,gbojorquez }@itchihuhua.edu.mx

[2] División de Estudios de Posgrado de la Facultad de Ingeniería de la Universidad Autónoma de Querétaro.Cerro de las Campanas S/N. Col. Las Campanas, Santiago de Querétaro Qro. México 76010; Tel. (442) 192 12 00 Fax; (442) 192 12 00 ext. 6005; E-mail: gherrera@uaq.mx

* Author to whom correspondence should be addressed. E-mail: jrivera@itchihuahua.edu.mx

Received: 1 June 2007 / Accepted: 10 August 2007 / Published: 16 August 2007

Abstract: The development of smart sensors involves the design of reconfigurable systems capable of working with different input sensors. Reconfigurable systems ideally should spend the least possible amount of time in their calibration. An autocalibration algorithm for intelligent sensors should be able to fix major problems such as offset, variation of gain and lack of linearity, as accurately as possible. This paper describes a new autocalibration methodology for nonlinear intelligent sensors based on artificial neural networks, ANN. The methodology involves analysis of several network topologies and training algorithms. The proposed method was compared against the piecewise and polynomial linearization methods. Method comparison was achieved using different number of calibration points, and several nonlinear levels of the input signal. This paper also shows that the proposed method turned out to have a better overall accuracy than the other two methods. Besides, experimentation results and analysis of the complete study, the paper describes the implementation of the ANN in a microcontroller unit, MCU. In order to illustrate the method capability to build autocalibration and reconfigurable systems, a temperature measurement system was designed and tested. The proposed method is an improvement

over the classic autocalibration methodologies, because it impacts on the design process of intelligent sensors, autocalibration methodologies and their associated factors, like time and cost.

Keywords: intelligent sensors, reconfigurable systems, autocalibration, linearization, artificial neural network.

1. Introduction

In order to design intelligent sensors for measurement systems with improved features a simple reconfiguration process for the main hardware will be required in order to measure different variables by just replacing the sensor element, building reconfigurable systems. Reconfigurable systems, ideally should spend the least possible amount of time in their calibration. An autocalibration algorithm for intelligent sensors should be able to fix major problems such as offset, variation of gain and lack of linearity, all characteristic of degradation, as accurately as possible.

The linearization of output signal sensors and the calibration process are the major items that are involved in defining the features of an intelligent sensor, for example, the capability to be used or applied to different variables, calibration time and accuracy.

The subject of linearization has been considered in different forms and stages, basically from the designs of circuits with MOS technologies [1]. Cases studied included the use of analog to digital converters to solve nonlinearities at the same time that the conversion is made [2-3]. In digital to analog type R-2R converters have also proved to be necessary to improve the linear response [4]. Other work has focused on improving the nonlinear response of specific sensors, like the thermistor [5] and the Hall effect current sensors [6]. Numerical methods have been developed using modern technologies capable of computing linearization algorthms [7, 8]. Also, ROM memories are used to save data tables and to solve the linearization problem [9,10]. Nowadays neural networks have been used for linearization [11-14]. Neural networks can be used to identify the transfer function response curve of the sensors [15]. Recently, neural networks have been used to linearize amplifiers [16] or specific sensors responses [17].

The calibration method stage is important for two important aspects: first to define the features of the measurement system and second, the maintenance costs of measurement systems. The costs associated with the maintenance of measurement systems from 70's to now have been discussed [18-21], and these documents indicate that the companies have to spend more money applied to pay for quality services due to calibration problems.

A wide application area of neural networks is in recognition of patterns in the analysis of signals from sensors [22,23]. The self-calibration concept is approached from different stages, for example in the fabrication of integrated circuits [24] or in signal transmissions [25]. In [26] the improvement of the adaptive network-based fuzzy inference system (ANFIS) technique was shown and in [27] a radial bases function RBF neural network for pyroelectric sensor array calibration was presented; in both works a complex process training and a PC are required, so they cannot easily be implemented on DSP

or small processors. A neural network for electromagnetic flow meter sensor calibration is described in [28], but only simulated results are presented, the neural network selection is not clear, large quantities of data are used for training and the capability of the method is not evaluated. Back propagation artificial neural networking was used in the data analysis shown in [29]. A neural network for linearization of the response of a pressure sensor is shown in [30,31], but only simulated results are presented and the neural network selection, the training process, the comparison with other methods and the method capability are not clear.

Most of the available literature related to autocalibration methods depends on several experiments to determine its performance. This approach impacts the time and cost of a reconfigurable system. This paper describes a new autocalibration methodology for nonlinear intelligent sensors based on artificial neural networks, ANN. The methodology involves analysis of several network topologies and training algorithms [32-34]. The artificial neural network was tested with different levels of nonlinear input signals. The proposed method was compared against the piecewise [35] and polynomial linearization methods [36] using simulation software. The comparison was achieved using different numbers of calibration points, and several nonlinear levels of the input signal. The proposed method turned out to have a better overall accuracy than the other two methods.

Besides complete experimental details, results and analysis of the complete study, the paper describes the implementation of the ANN in a microcontroller unit, MCU. In order to illustrate the method capability to build autocalibration and reconfigurable systems a temperature measurement system was designed. The system is based on a thermistor, which presents one of the worst nonlinearity behaviors, and is also found in many real world applications because of its low cost.

One important point that needs clarification before proceeding relates to the meaning of the term calibration. In this paper, calibration is used in the sense of [3, 8, 9, 15, 22-25, 28, 30, 35, 36, 40]. In these works, calibration is mainly concerned with the process of removing systematic errors and it is not in accordance with the meaning defined in the Metrology and the International Vocabulary of Basic and General Terms in Metrology (VIM), ISO VIM [42-45]. In addition, in this paper the phrase “self-calibration of intelligent sensor” or “autocalibration” has the meaning of “self-adjustment of measurement system” according to VIM [44].

The paper structure will be as follows: the basic system design considerations are presented in Section 2. The ANN design, training algorithm and simulation are described in Section 3. An ANN of temperature measurement system implementation on MCU is shown in Section 4. The tests and results are described on Section 5. The evaluation results are given in Section 6. Finally, the conclusions are in Section 7.

2. Basic System Design Considerations.

Prior to any ANN design some considerations are necessary. The output electrical signal $\boldsymbol{x'}$ of any sensor in response to the input variable $\boldsymbol{v'}$, which will be measured is defined by:

$$x' = f(v') \tag{1}$$

In most of the cases the input and output variables have different scales and they are normalized in the range of [0,1] to simplify their manipulation. This can be obtained by the following equations:

$$v = \frac{v' - v'_{\min}}{v'_{\max} - v'_{\min}} \tag{2}$$

$$x = \frac{x' - x'_{\min}}{x'_{\max} - x'_{\min}} \tag{3}$$

The desired output signal is a straight line with unit slope. This will be the target in the calibration process and will be the reference signal defined by:

$$t = v \tag{4}$$

Finally, a relative metric error can be used to determine how linear is the output signal ***y*** of the ANN obtained by:

$$\varepsilon_r = y - t \tag{5}$$

Another way to corroborate the method is by using the least mean square error MSE, expressed by

$$\varepsilon_{mse} = \frac{1}{N}\sum_{n=1}^{N}\left(y_n - t_n\right)^2 \tag{6}$$

In the next section where the ANN design for self-calibration of intelligent sensors will be described, the input signal to the ANN will be the ***x*** term from equation (3).

3. Artificial Neural Network Design to Self-Calibration of Intelligent Sensors.

This section describes the ANN design to be used in a Self-Calibration of intelligent sensors. The training algorithm and simulation are also described.

3.1. Artificial Neural Network Design.

Several topologies of ANN like a multiplayer perceptron MLP and radial basis function RBF were evaluated. Multilayer perceptron (MLP) neural networks with sufficiently abundant nonlinear units in a single hidden layer have been established as universal function approximators. Some work has been performed to show the relationship between RBF networks and MLPs. since both types of networks are capable of universation approximation capabilities [37, 38].

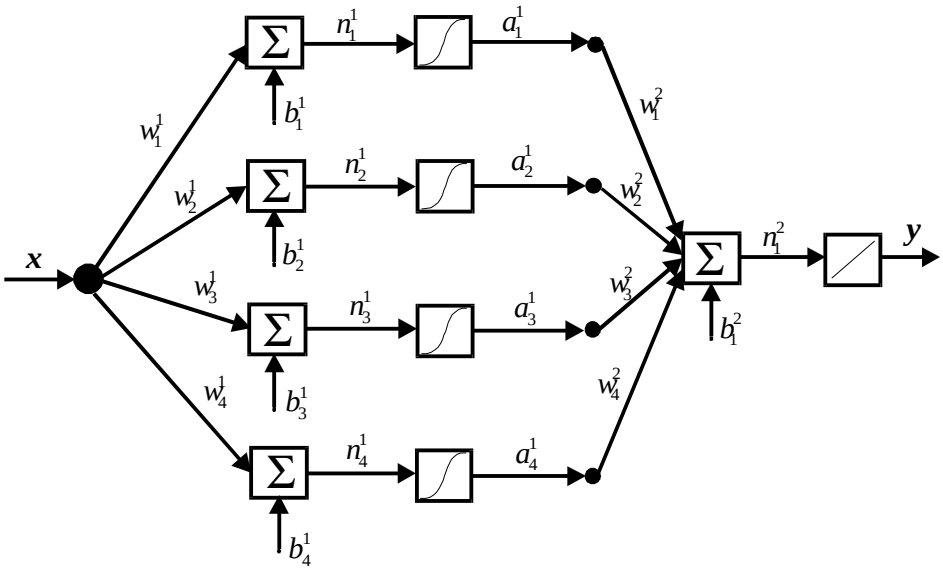
Although, it is known that RBF are good function approximation systems, the MLP was selected because is simpler than RBF and furthermore, the RBF network is computationally demanding [26, 27,

39]. Besides, in other works, the relationship between RBF networks and MLPs has been studied. Mainly, if we assume that an MLP is a universal approximator, then it may approximate an RBF network and vice versa. It has also been showed that in problems with normalized inputs, MLPs work like RBF networks with irregular basis functions [38].

Consequently for our proposal the most appropriate ANN to be implemented was a feed forward MLP with four neurons in the first layer and a logarithmic activation function. The second layer is a single neuron with a linear activation function. The architecture of the ANN is shown in Figure 1.

The number of neurons, number of layers, activation functions, training algorithm and the computation requirements are the major characteristics considered during the design. These features were determined under the restriction of archiving the least output error and simplest ANN structure to be implemented in a small MCU.

Figure 1. Architecture of the artificial neural network.



The output of the ANN is defined by:

$$y = Purelin\left[w_i^2 \left(Logsig(w_i^1 x + b_i^1) \right) + b_1^2 \right], \quad i = 1 \text{ to } 4 \tag{7}$$

where x is the normalized output sensor signal, the weights are represented by the vector w, b is the bias and y is the linearized or autocalibrated signal.

3.2. Training Algorithm.

Currently many variations of backpropagation training algorithms are available. The Adaptive Linear Neuron ADALINE and the Least Mean Square LMS were first presented in the late 50´s, they are very frequently used in adaptive filtering applications. Echo cancellers using the LMS algorithm are currently employed on many long distance telephone lines. Backpropagation is an approximate steepest

descent algorithm that minimizes mean square error, the difference between LMS and backpropagation is in the manner in which the derivatives is calculated. Backpropagation is a generalization of the LMS algorithm that can be used for multilayer networks. One of major problems with backpropagation has been the long training times needed. The backpropagation process is too slow for most practical applications and it is not feasible to use the basic backpropagation algorithm for real problems, because it can take weeks to train a network, even on a large computer. Since the backpropagation algorithm was first popularized, there has been considerable work on methods to accelerate the convergence. Another important factor in the backpropagation algorithm is the learning rate. Depending on the value of the learning rate the ANN may or may not oscillate. On this item several experiments have been made leading to selection of the appropriate learning rate [32]. Reducing the probability of oscillation of the ANN is a trade-off with respect to the training speed. Therefore, this situation needs to be faced with faster algorithms. Research on faster algorithms falls roughly into two categories: 1) the development of heuristic techniques. These heuristic techniques include such ideas as varying learning rates, using momentum and rescaling variables; 2) another category of research has focused on standard numerical optimization techniques. The conjugate gradient algorithm and the Levenberg Marquardt algorithm have been very successfully applied to the training of MLP [32].

The Levenberg Marquardt algorithm is a variation of Newton´s method and uses the backpropagation procedure. Levenberg Marquardt Backpropagation (LMBP) was designed for minimizing functions that are sums of squares of other nonlinear functions. This is very well suited to neural network training where the performance index is the mean squared error. The LMBP is the fastest algorithm that we have tested for training multiplayer networks of moderate size, even though it requires a matrix inversion at each iteration. It is remarkable that the LMBP is always able to reduce the sum of squares at each iteration [32].

Looking for a training algorithm with the best features of training time, minimum error with practical application we decided to evaluate backpropagation, backpropagation with momentum and the LMBP algorithms. Table 1 summarizes the results of this evaluation. As can be observed in the table, the Lenvenberg-Marquardt algorithm has a faster convergence time and also the lowest error. Besides, this method works extremely well in practice, and is considered the most efficient algorithm for training median sized ANN [32,34].

Table 1. Training algorithms evaluation.

Algorithm	No. Cycles	Mean Square Error MSE
Backpropagation	500	0.0576
Backpropagation Momentum	500	0.0372
Lenvenberg-Marquardt	22	1.986e-16

Next, the Lenvenberg-Marquardt algorithm for our particular application is described. Assuming ***N*** calibration points we define the input sensor signal v_n, the output sensor signal x_n and its desired t_n, with normalized values equations (2-4) as:

$$\{v_1, x_1, t_1\}, \{v_1, x_1, t_1\}, ..., \{v_N, x_N, t_N\} \tag{8}$$

The iterations of the Levemberg Marquardt algorithm to autocalibration of intelligent sensors for k cycles can be summarized as follows [32]:

Step 1: Present all the N calibration points and compute the sum of MSE with equation (6).

***Step 2*:** Determine the Jacobian matrix by:

$$J(p)=\begin{bmatrix} \frac{\partial \varepsilon_1}{\partial w_1^1} & \frac{\partial \varepsilon_1}{\partial w_2^1} & \frac{\partial \varepsilon_1}{\partial w_3^1} & \frac{\partial \varepsilon_1}{\partial w_4^1} & \frac{\partial \varepsilon_1}{\partial b_1^1} & \frac{\partial \varepsilon_1}{\partial b_2^1} & \frac{\partial \varepsilon_1}{\partial b_3^1} & \frac{\partial \varepsilon_1}{\partial b_4^1} & \frac{\partial \varepsilon_1}{\partial w_1^2} & \frac{\partial \varepsilon_1}{\partial w_2^2} & \frac{\partial \varepsilon_1}{\partial w_3^2} & \frac{\partial \varepsilon_1}{\partial w_4^2} & \frac{\partial \varepsilon_1}{\partial b_1^2} \\ \frac{\partial \varepsilon_2}{\partial w_1^1} & \frac{\partial \varepsilon_2}{\partial w_2^1} & \frac{\partial \varepsilon_2}{\partial w_3^1} & \frac{\partial \varepsilon_2}{\partial w_4^1} & \frac{\partial \varepsilon_2}{\partial b_1^1} & \frac{\partial \varepsilon_2}{\partial b_2^1} & \frac{\partial \varepsilon_2}{\partial b_3^1} & \frac{\partial \varepsilon_2}{\partial b_4^1} & \frac{\partial \varepsilon_2}{\partial w_1^2} & \frac{\partial \varepsilon_2}{\partial w_2^2} & \frac{\partial \varepsilon_2}{\partial w_3^2} & \frac{\partial \varepsilon_2}{\partial w_4^2} & \frac{\partial \varepsilon_2}{\partial b_1^2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \varepsilon_N}{\partial w_1^1} & \frac{\partial \varepsilon_N}{\partial w_2^1} & \frac{\partial \varepsilon_N}{\partial w_3^1} & \frac{\partial \varepsilon_N}{\partial w_4^1} & \frac{\partial \varepsilon_N}{\partial b_1^1} & \frac{\partial \varepsilon_N}{\partial b_2^1} & \frac{\partial \varepsilon_N}{\partial b_3^1} & \frac{\partial \varepsilon_N}{\partial b_4^1} & \frac{\partial \varepsilon_N}{\partial w_1^2} & \frac{\partial \varepsilon_N}{\partial w_2^2} & \frac{\partial \varepsilon_N}{\partial w_3^2} & \frac{\partial \varepsilon_N}{\partial w_4^2} & \frac{\partial \varepsilon_N}{\partial b_1^2} \end{bmatrix} \quad (9)$$

where ε is the error relative for each calibration point computed from the equation (6)

***Step 3*:** Now compute the variation or delta of ANN parameters, Δp_k :

$$\Delta p_k = \left[J^T(p_k) J(p_k) + \mu_k I \right]^{-1} J^T(p_k) \varepsilon(p_k) \quad (10)$$

Where the learning factor is represented by μ_k, I is the identity matrix and the $J(p_k)$ is the Jacobian matrix evaluated with the ANN parameters p_k.

***Step 4*:** Recompute the sum of squared errors with the update parameters, equation (6), step 1 for the update of parameters:

$$p_{k+1} = p_k^T + \Delta p_k^T \quad (11)$$

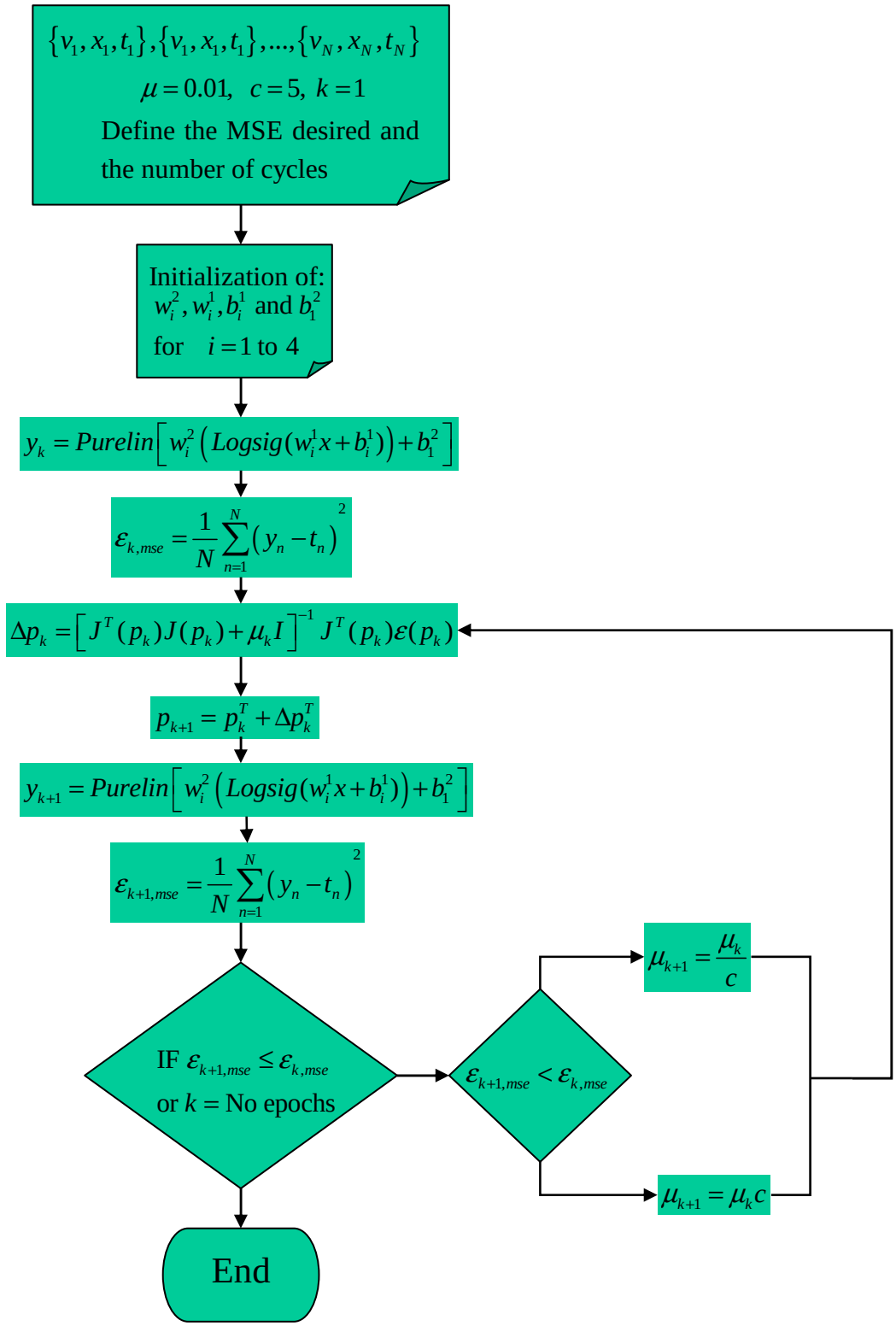
For this case: If the result of MSE is smaller than computed in the step 1, $\varepsilon_{k+1,mse} < \varepsilon_{k,mse}$, then evaluate μ_{k+1} as $\mu_{k+1} = \frac{\mu_k}{c}$, c is a constant value, and continue with the step 2. If the sum of squares is not reduced, $\varepsilon_{k+1,mse} > \varepsilon_{k,mse}$, evaluate μ_k as $\mu_{k+1} = \mu_k c$ and go to the step 2. All this will be repeated until the desired error or $k\acute{}s$ cycles are reached. Note that the initial values of μ and c are the key to the right convergence, the recommend values are μ=0.01 and c=5. Figure 2 shows the flow chart of the LBMP algorithm.

3.3 Simulation to Evaluation and Results Comparison.

The ANN training was made with five to eight calibration points using the function $x = 1 - e^{-\frac{v}{\tau}}$, normalized with the equations (3) and (4). τ represents the percentage of the nolinearity of $\boldsymbol{x}$. The

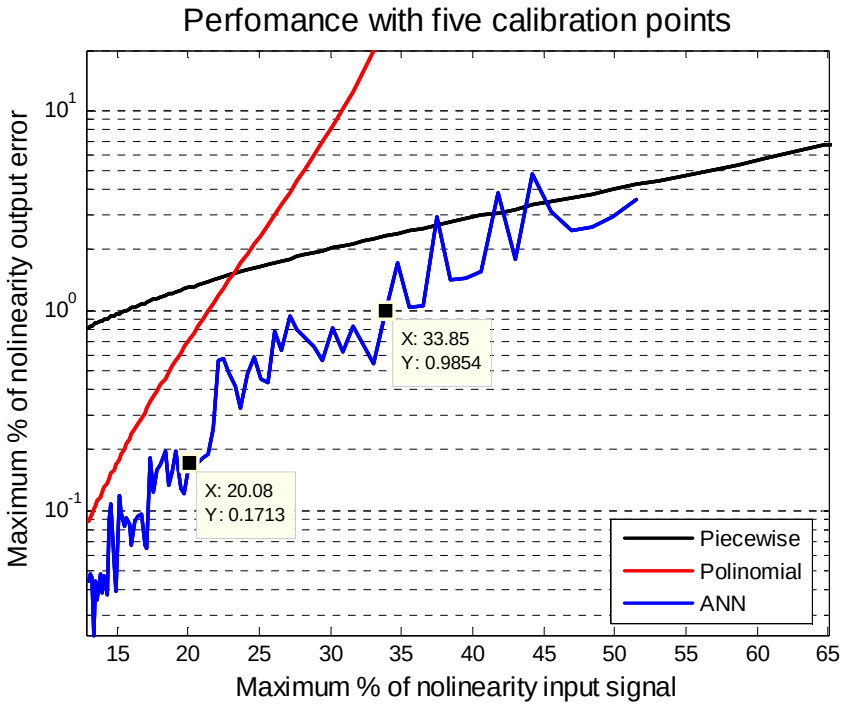
artificial neural network was tested with different level of nonlinear input signals. The proposed method was compared against the piecewise [35] and polynomial linearization methods [36] using simulation software. Figure 3 illustrates the results of the comparison of the ANN, piecewise and polynomial methods. Each figure shows the performance of these methods for different calibration points regarding different levels of input nonlinearity, from 10% to 65%.

Figure 2. Flow chart of the LMBP algorithm.



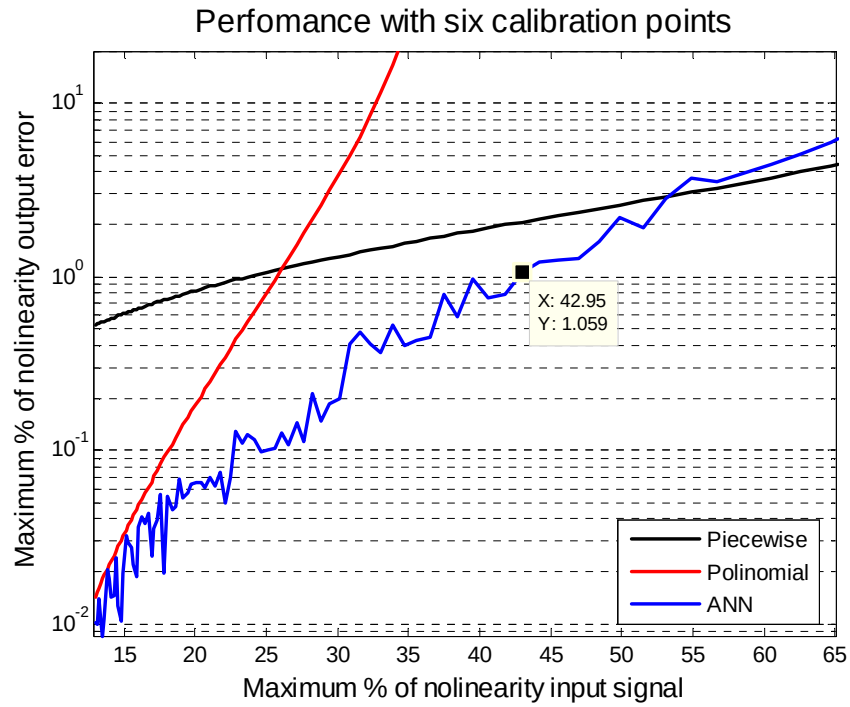
The performance is given in maximum percentage of nonlinearity output error, MPNOE. In Figure 3 it can be noted that the proposed method with ANN turned out to have a better overall accuracy than the other two methods. The ANN always presents the minimum MPNOE considering the 1% of MPNOE, maximum acceptable value of practical applications. For example, with five calibration points and supplying a signal of a 20% of relative nonlinearity, the piecewise method has an error above 1%, the polynomial has an error around 0.71% and the proposed method has an error of 0.17 %. The performance of the ANN was obtained using the same initial conditions for all cases. Another advantage of ANN method is that with five calibration points a signal below 33% of the maximum nonlinearity error can be fixed, to yield a performance in the output of less than 1% maximum error of nonlinearity. Using eight calibration points even signals with 56% of maximum error of nonlinearity can be fixed or linearized. The implementation of the ANN method on a MCU using a thermistor will be described in the next section.

Figure 3. Performance of the ANN to self-calibration. a) With five calibration points, b) with six calibration points c) with seven calibration points and d) with eight calibration points.

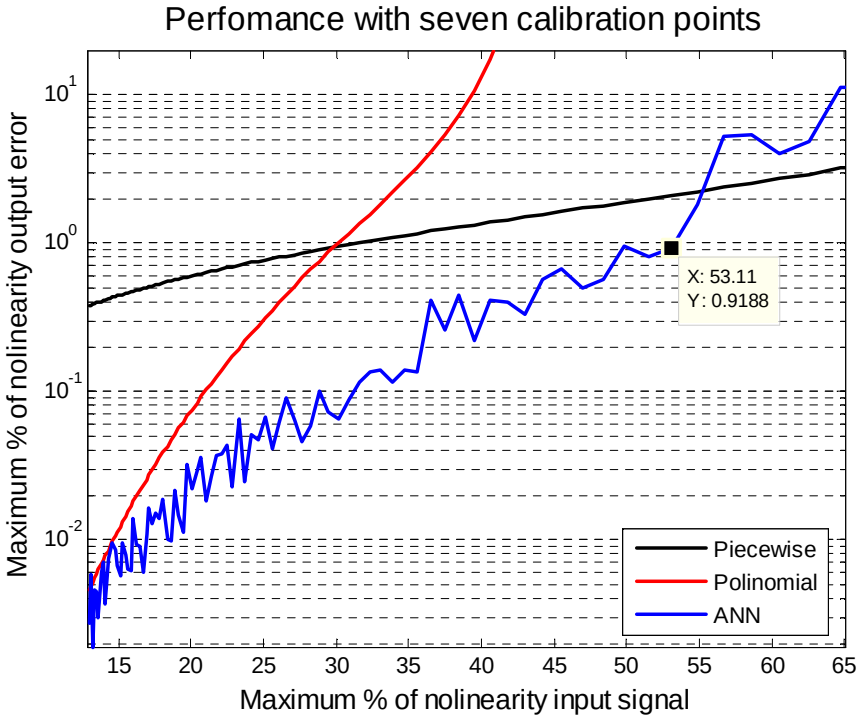


(a)

Figure 3. Cont.

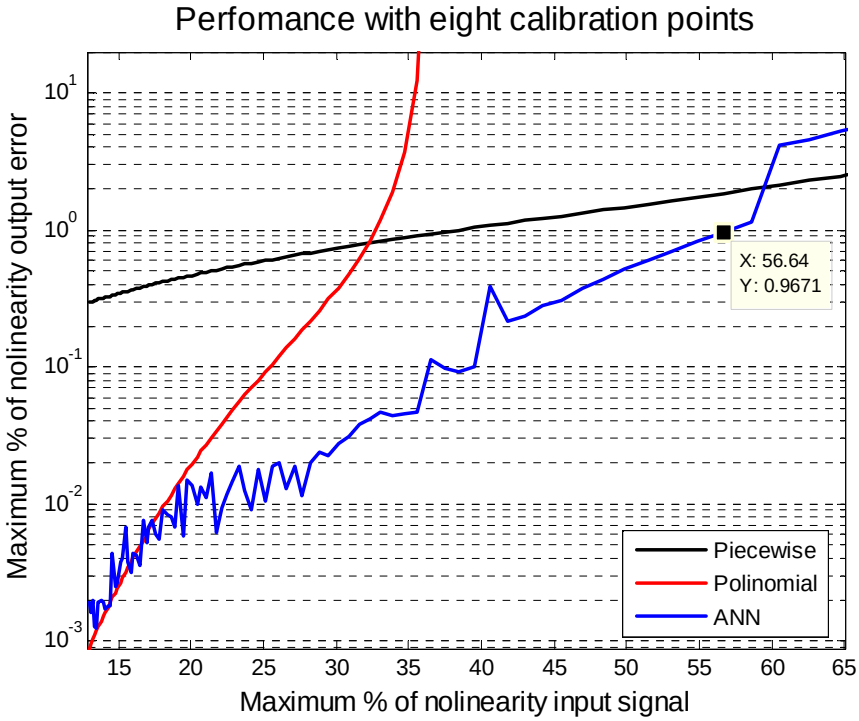


(b)



(c)

Figure 3. Cont.

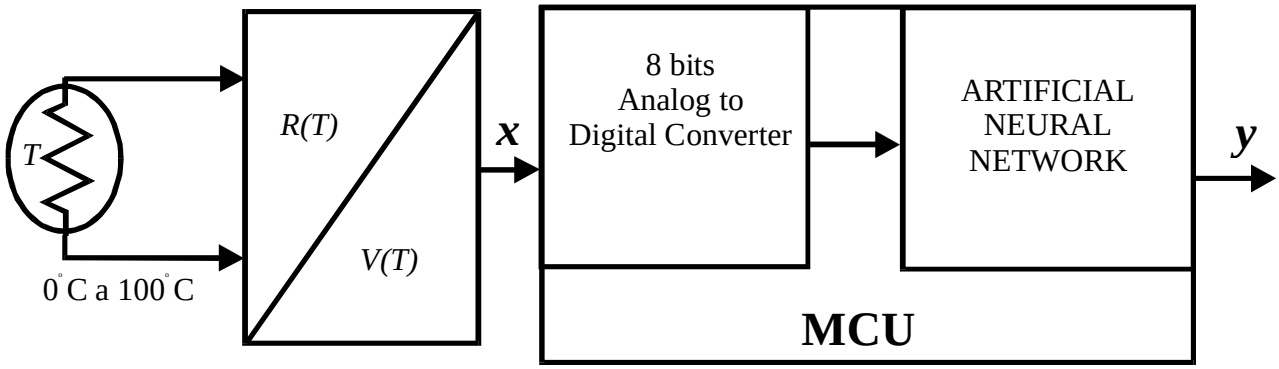


(d)

4. Temperature Intelligent Sensor Design using ANN on a Small MCU.

Temperature measurement systems are widely used in almost any process. A thermistor as temperature sensor was selected in the construction of a measurement system, Figure 4. Thermistors, besides having a diversity of applications, can be found in a great variety of types, sizes and characteristics.

Figure 4. Structure of temperature measurement system.

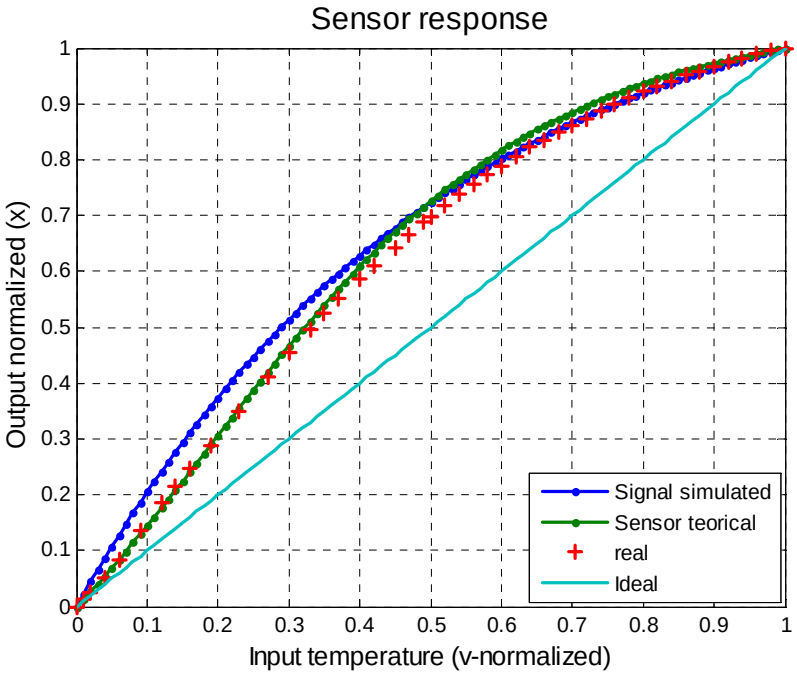


In this case, the major characteristic that will be analyzed is the β coefficient, a fundamental characteristic used to describe its nonlinearity error. For example, values of $\beta=3100$ to $\beta=4500$ generate nonlinearity relative errors ranging from 41.07% to 51.34%. It is clear that in the practical

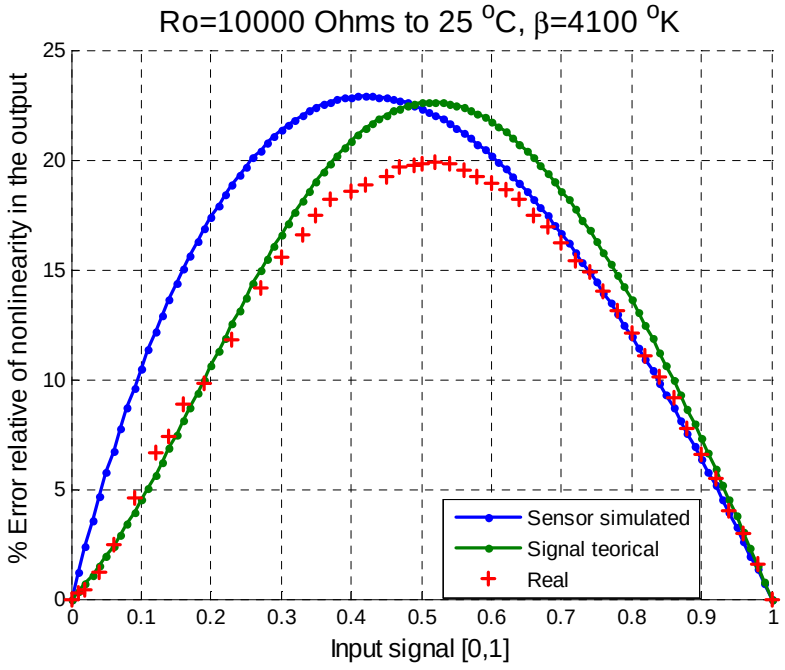
cases, the sensor is incorporated into a circuit to perform the temperature to voltage conversion. For example, it can be found in a voltage divider or a Wheatstone bridge. In our design we used a thermistor with features of $\beta = 4500 \pm 10\%$, R_o=10000 ohms to 25 ^oC, and assembled on a voltage divider to build a measurement system in the range from 0 to 100 ^oC as shown in Figure 4.

Figure 5 illustrates the characteristics of ***x***, the input signal to the ANN. This signal is the response the sensor to the temperature and the values are normalized in the ranges of [0,1].

Figure 5. Characteristics of input signal ***x*** to the ANN.



(a)



(b)

In Figure 5a the signal simulated as was described in Section 3.3, the signal evaluated with the equation characteristic of a thermistor using nominal values, the response of thermistor real obtained from a practical test and the ideal response ***t*** (a straight line) can be observed. Figure 5b shows the percentage of nolinearity of this signal ***x***, evaluated according to equation (5). In this figure all the error signals are shown: simulated signal, theorical thermistor response, and the real thermistor response. From Figure 5b, the similarity of the three signals can be easily appreciated. Taking into consideration just the parameter β and its ±10% tolerance, the real signal is under this limits and our proposal methodology is acceptable.

The major features of the MCU for the physical implementation are: eight bit words, eight bits analog to digital converter ADC, a 20 MHz clock and 3 Kbytes of RAM. Equation (7) and the training algorithm described in Section 3.2 were programmed using C language. Next the ANN implementation and its training for autocalibration and to linearize the signal ***x*** in intelligent sensor will be described.

The iterations of the Levemberg Marquardt backpropagation algorithm assuming five calibration points was programmed as follows:

The input calibration signal was $v' = [0, 23, 50, 76, 100]$, temperature in ^{o}C. Using the equation (2-3) the values normalized to the vectors **x**, **y** and **t** were calculated, to obtain the vectors of equation (8):

$$v = [0, 0.23, 0.5, 0.76, 1] \quad x = [0, 0.3483, 0.6983, 0.9001, 1] \text{ and } \quad t = [0, 0.23, 0.5, 0.76, 1]$$

The initial values for weights and bias were taken from the simulation described in Section 3.3, with the values of:

$$w^{1T} = [0.3272,\ 0.1746,\ -0.1867,\ 0.7257]$$
$$w^{2T} = [-0.5883,\ -0.1363,\ 0.1139,\ 0.0592]$$
$$b^{1} = [1,1,1,1]$$
$$b^{2} = [1]$$

Step 1: Using the equations (10) and (12) the output signal of the ANN ***y*** from equation (7) is computed. The evaluation of the error equation (5) and the MSE with equation (6) is computed.

$$y_n = Purelin\left[w^2 \left(Logsig(w^1 x_j + b^1) \right) + b^2 \right], \quad n = 1 \text{ to } 5$$

$$y = [0.5968, 0.5837, 0.5708, 0.5636, 0.5601]$$
$$\varepsilon_1 = [-0.5968 \quad -0.3537 \quad -0.0708 \quad 0.1964 \quad 0.4399]$$
$$\varepsilon_{1,mse} = 0.3791$$

***Step 2*:** The Jacobian matrix is computed. In this example the size of the Jacobian matrix according to equation (17) and five calibration points will be 5 x 13, defined as:

$$J=\begin{bmatrix}
\frac{\partial \varepsilon_1}{\partial w_1^1} & \frac{\partial \varepsilon_1}{\partial w_2^1} & \frac{\partial \varepsilon_1}{\partial w_3^1} & \frac{\partial \varepsilon_1}{\partial w_4^1} & \frac{\partial \varepsilon_1}{\partial b_1^1} & \frac{\partial \varepsilon_1}{\partial b_2^1} & \frac{\partial \varepsilon_1}{\partial b_3^1} & \frac{\partial \varepsilon_1}{\partial b_4^1} & \frac{\partial \varepsilon_1}{\partial w_1^2} & \frac{\partial \varepsilon_1}{\partial w_2^2} & \frac{\partial \varepsilon_1}{\partial w_3^2} & \frac{\partial \varepsilon_1}{\partial w_4^2} & \frac{\partial \varepsilon_1}{\partial b_1^2} \\
\frac{\partial \varepsilon_2}{\partial w_1^1} & \frac{\partial \varepsilon_2}{\partial w_2^1} & \frac{\partial \varepsilon_2}{\partial w_3^1} & \frac{\partial \varepsilon_2}{\partial w_4^1} & \frac{\partial \varepsilon_2}{\partial b_1^1} & \frac{\partial \varepsilon_2}{\partial b_2^1} & \frac{\partial \varepsilon_2}{\partial b_3^1} & \frac{\partial \varepsilon_2}{\partial b_4^1} & \frac{\partial \varepsilon_2}{\partial w_1^2} & \frac{\partial \varepsilon_2}{\partial w_2^2} & \frac{\partial \varepsilon_2}{\partial w_3^2} & \frac{\partial \varepsilon_2}{\partial w_4^2} & \frac{\partial \varepsilon_2}{\partial b_1^2} \\
\frac{\partial \varepsilon_3}{\partial w_1^1} & \frac{\partial \varepsilon_3}{\partial w_2^1} & \frac{\partial \varepsilon_3}{\partial w_3^1} & \frac{\partial \varepsilon_3}{\partial w_4^1} & \frac{\partial \varepsilon_3}{\partial b_1^1} & \frac{\partial \varepsilon_3}{\partial b_2^1} & \frac{\partial \varepsilon_3}{\partial b_3^1} & \frac{\partial \varepsilon_3}{\partial b_4^1} & \frac{\partial \varepsilon_3}{\partial w_1^2} & \frac{\partial \varepsilon_3}{\partial w_2^2} & \frac{\partial \varepsilon_3}{\partial w_3^2} & \frac{\partial \varepsilon_3}{\partial w_4^2} & \frac{\partial \varepsilon_3}{\partial b_1^2} \\
\frac{\partial \varepsilon_4}{\partial w_1^1} & \frac{\partial \varepsilon_4}{\partial w_2^1} & \frac{\partial \varepsilon_4}{\partial w_3^1} & \frac{\partial \varepsilon_4}{\partial w_4^1} & \frac{\partial \varepsilon_4}{\partial b_1^1} & \frac{\partial \varepsilon_4}{\partial b_2^1} & \frac{\partial \varepsilon_4}{\partial b_3^1} & \frac{\partial \varepsilon_4}{\partial b_4^1} & \frac{\partial \varepsilon_4}{\partial w_1^2} & \frac{\partial \varepsilon_4}{\partial w_2^2} & \frac{\partial \varepsilon_4}{\partial w_3^2} & \frac{\partial \varepsilon_4}{\partial w_4^2} & \frac{\partial \varepsilon_4}{\partial b_1^2} \\
\frac{\partial \varepsilon_5}{\partial w_1^1} & \frac{\partial \varepsilon_5}{\partial w_2^1} & \frac{\partial \varepsilon_5}{\partial w_3^1} & \frac{\partial \varepsilon_5}{\partial w_4^1} & \frac{\partial \varepsilon_5}{\partial b_1^1} & \frac{\partial \varepsilon_5}{\partial b_2^1} & \frac{\partial \varepsilon_5}{\partial b_3^1} & \frac{\partial \varepsilon_5}{\partial b_4^1} & \frac{\partial \varepsilon_5}{\partial w_1^2} & \frac{\partial \varepsilon_5}{\partial w_2^2} & \frac{\partial \varepsilon_5}{\partial w_3^2} & \frac{\partial \varepsilon_5}{\partial w_4^2} & \frac{\partial \varepsilon_5}{\partial b_1^2}
\end{bmatrix} \tag{12}$$

$$J(p)=\begin{bmatrix}
0 & 0 & 0 & 0 & 0.1157 & 0.0268 & -0.0224 & -0.0116 & -0.7311 & -0.7311 & -0.7311 & -0.7311 & -1 \\
0.0881 & 0.0091 & -0.0080 & -0.0036 & 0.1095 & 0.0260 & -0.0231 & -0.0102 & -0.7529 & -0.7428 & -0.7181 & -0.7778 & -1 \\
0.0720 & 0.0176 & -0.0166 & -0.0061 & 0.1031 & 0.0253 & -0.0237 & -0.0088 & -0.7736 & -0.7543 & -0.7047 & -0.8186 & -1 \\
0.0894 & 0.0223 & -0.0217 & -0.0072 & 0.0993 & 0.0248 & -0.0241 & -0.0080 & -0.7849 & -0.7603 & -0.6963 & -0.8393 & -1 \\
0.0975 & 0.0246 & -0.0242 & -0.0076 & 0.0975 & 0.0246 & -0.0242 & -0.0076 & -0.7904 & -0.7640 & -0.6928 & -0.8469 & -1
\end{bmatrix}$$

***Step 3*:** Solve the equation (10) to get Δp_k, for the first iteration for $k=1$, $\mu_1=0.01$ and $c=5$ the result is:

$$\Delta p_1^T = \begin{bmatrix}-2.3307 & -0.5974 & 0.6054 & 0.1711 & 0.5757 & 0.0839 \\ 0.0211 & -0.1123 & 0.6563 & 0.0195 & -1.7367 & 2.0186 & -1.0764\end{bmatrix}$$

***Step 4*:** Recompute the sum of squared errors with the update parameters, equation (6), step 1. For this case:

$$p_{k+1} = p_2 = p_1^T + \Delta p_1^T$$

If the result of MSE is smaller than computed in the step 1, $\varepsilon_{2,mse} < \varepsilon_{1,mse}$, then evaluate μ_2 as $\mu_2 = \frac{\mu_1}{c}$, and continue with step 2. If the sum of squares error is not reduced, $\varepsilon_{2,mse} > \varepsilon_{1,mse}$, evaluate μ as $\mu_2 = \mu_1 c$ and go to step 2. All this will be repeated until the desired error or $k\acute{}s$ cycles are reached. The process above described was repeated for ***k***=500 or 500 cycles and final parameters are:

$$p_{500}^T = \begin{bmatrix}-3.3221 & 0.3498 & 2.1249 & -12.3130 & 4.1837 & 1.3087 \\ 1.1658 & 15.0081 & -1.7460 & 1.3760 & 1.3712 & 2.5059\end{bmatrix}$$

The MSE was:

$$\varepsilon_{500,mse} = 2.6767\text{x}10^{-16}$$

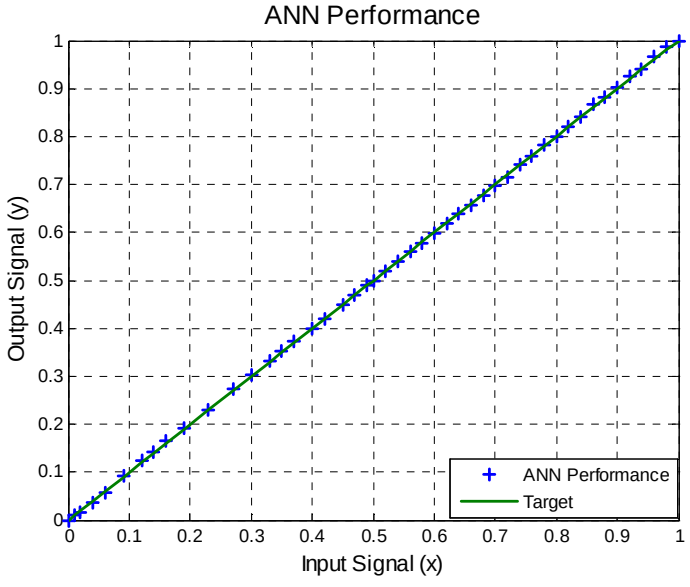
Then the ANN model from equation (7) is:

$$y = Purelin\left(\begin{bmatrix}-1.7460\\1.3760\\1.3712\\-2.5059\end{bmatrix}^{T}\left(Logsig\left(\begin{bmatrix}-3.3221\\0.3498\\2.1249\\-12.3130\end{bmatrix}x+\begin{bmatrix}4.1837\\1.3087\\1.1658\\15.0081\end{bmatrix}\right)\right)+2.0969\right) \quad (13)$$

5. Tests and results.

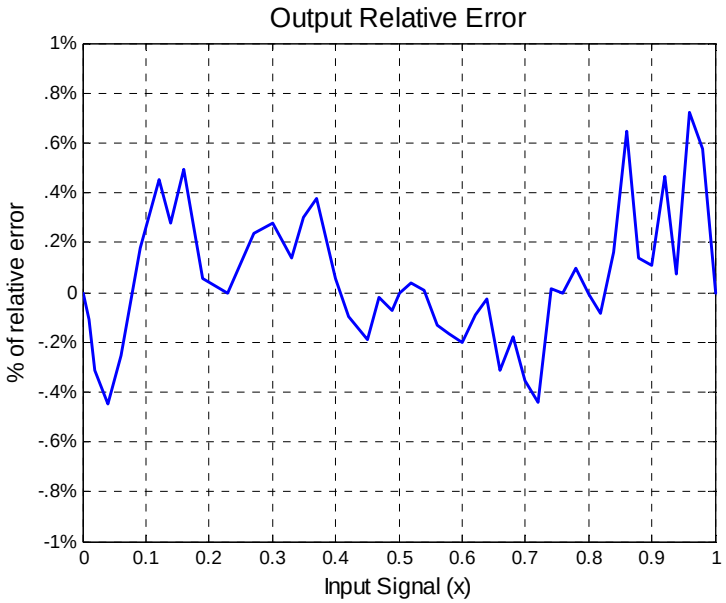
The performance of the ANN was compared against a Honeywell temperature meter number UDC3000 with a type K -29 to 538 ^{o}C (–20 to 1000 ^{o}F) range thermocouple with an accuracy of $\pm$ 0.02%, taking 50 measurements from a range of 0 to 100 ^{o}C in steps of 2 ^{o}C using an oven system to change the temperature. The results are shown in Figure 6. The ANN output signal ***y*** compared with the target straight line can be seen in Figure 6a. In order to visualize better the error between the ANN output and the target straight line the percentage of relative error of nolinearity computed with equation (5) is shown in Figure 6b. In summary, Figure 6b shows the difference between the ideal output and the output provided by the ANN. It can also be observed that the maximum percentage of relative nonlinearity error is approximately 0.7%, below 1% as was predicted from simulation and was illustrated in Figure 3a with five calibration points.

Figure 6. ANN performance with five calibration points. a) Output ANN signal. b) ANN output relative error.



(a)

Figure 6. Cont.



(b)

Important evaluations were made considering the computational work that is required in the physical implementation of each method described above. First, the number of operations required for each method using a software tool to program the MCU were counted, as well as the computation time required was predicted This process was made for *N*=5, 6, 7 and 8 calibration points, and the results are shown in the Table 2.

Table 2. Autocalibration methods. Their number of operations required and their computation time.

Method	Operation number				Time in microseconds (μs)			
	N=5	*N*=6	*N*=7	*N*=8	*N*=5	*N*=6	*N*=7	*N*=8
Piecewise	80	102	112	128	4915	5898	6881	7556
Polinomial	101	162	245	352	5609	8778	12974	18287
ANN	28				3523			

A similar process was carried out to evaluate the computational work and the training time. The computational time for one cycle of the ANN training is showed in Table 3, considering MCU clocks of 20 Mhz and 40 Mhz.

Besides, it is important to remark about the sensor resolution. The ADC determinates the sensor resolution and this is defined by: Resolution $= E_{FSR} / 2^{n-1}$, were E_{FRS} is the full range voltage scale and the number of bits of the ADC is n. In our example $E_{FRS} = 5$ volts, then the sensor resolution is 19.6 *mV*, that means, the temperature sensor is limited to detect temperature changes of 0.38 ^{o}C. In a case where this resolution might represent a problem in an specific application, the MCU can be changed

with 10 bits of ADC converter to improve the resolution to 4.9 mV (0.09 ^{o}C). Another alternative is the use of an external ADC with more than 10 bits.

Table 3. Time expense in the ANN training.

Calibration points number	Time expense in one cycle of training	
	CLOCK of 20Mhz	CLOCK of 40Mhz
N=5	0.69s	0.36s
N=6	0.74s	0.38s
N=7	0.79s	0.40s
N=8	0.84s	0.43s

6. Evaluation of Results.

The performance of the ANN method can be defined if we determine the deviation of the ANN output from a straight line, the ideal response, equation (5). The performance is obtained by using the collected data from section 5 (Figure 6) and the linear regression model [41] to evaluate and quantify the ANN output $\hat{y}$ as follows:

$$\hat{y} = mv + b \tag{14}$$

Ideally, the value of m should be one and the value of b should be zero to obtain the best performance of the ANN. The term b represents the unpredicted or unexplained variation in the response variable; it is conventionally called the "**error**" [41].

The values of m and b were computed with the calibration data and using the least squares analysis [42,43]. The results where 1.0015 and –0.0281 ^{o}C respectively. Then it is clear that the error for m is 1.5x10^{-3}, the error for b is –0.0281 ^{o}C.

We made a readjustment in the ANN equation (7), to the value of b_1^2 by adding the error value of –0.0281 ^{o}C. The experiment of Section 5 was repeated and the new values for m and b where computed again. The results where of 1.0015 and –2.15x10^{-4} ^{o}C, respectively.

The error regarding the best fitting line that was obtained with $m = 1.0015$ and $b = 0.0281$ is acceptable. From the mathematical point of view the readjustment made shows a decrease in the error of the b value, but from the practical point of view the nolinearity error was the same. Therefore, it was shown that with a single training process of the ANN was enough to obtain acceptable results.

7. Conclusions

In this paper a special methodology to design and evaluate an algorithm for autocalibration of intelligent sensors that can be used in the design of a measurement system with any sensor was described.

The major problems such as offset, variation of gain and nonlinearity in sensors can be solved by the ANN method. Most of the available literature related to autocalibration methods depends on several experiments to determine its performance. This process impacts on the time and cost of a reconfigurable system. This paper described a new autocalibration methodology for nonlinear intelligent sensors based on ANN. The artificial neural network was tested with different level of nonlinear input signals. The proposed method was compared against the piecewise and polynomial linearization methods. Method comparison was achieved using different number of calibration points, and several nonlinear levels of the input signal. The proposed method turned out to have a better overall accuracy than the other two methods. For example, for 20% relative nonlinearity the piecewise methods has an error above 1%, the polynomial one has an error around 0.71% and the proposed method has an error of 0.17 % using five calibration points in simulation results, as shown in Figure 3a.

The ANN method requires fewer operations than the other two methods and the number of operations does not increase if the number of calibration points increases. This can be observed in Table 2. A training process is required before the ANN method is used, as shown in Table 3, but this is not a problem because it will be executed each time that the sensor is under calibration or the first time when a new intelligent sensor is designed. The ANN method can be trained in the self MCU but for some cases the required computational time resources in the training process could be a disadvantage but this can be solve using a remote compute system. Thus, if the sensor is on a network, the training can be done using a computer and the ANN parameters sent to the sensor.

Besides the major advantages which are that with a low number of calibration points the ANN method achieves better results than the piecewise and polynomial methods, as a consequence it requires less time and cost for its maintenance than any measurement system.

In conclusion, the proposed method is an improvement over the classic autocalibration methodologies, because it impacts on design process of intelligent sensors, autocalibration methodologies and their associated factors, like time and cost.

References and Notes

1. Khachab N. I.; Mohammed I. Linearization Techniques for n-th Order Sensor Model in Mos VLSI Technology. *IEEE Transactions on Circuits and Systems* **1991**, *38*, 1439-1449.
2. Iglesias G. E.; Iglesias E. A., Linearization of Transducer Signal Using an Analog to Digital Converter, *IEEE Transactions on Instrumentation and Measurement* **1988**, *37*, 53-57.
3. Vargha B.; Zoltán I. Calibration Algorithm for Current-Output R-2R Ladders. *IEEE Transactions On Instrumentation And Measurement* **2001,** *50*, 1216-1220.
4. Dent A.C.; Colin F.N. Linearization of Analog to Digital Converters. *IEEE Transactions on Circuits and Systems* **1990,** *37*, 729-737.
5. Kaliyugavaradan S.; Sankaran P.; Murti V. G. K. A New Compensation Scheme for Thermistors and its Implementation for Response Linearization Over a wide Temperature Range. *IEEE Transactions on Instrumentation and Measurement* **1993,** *42*, 952-956.

6. Cristaldi L.; Ferro A.; Lazzaroni M.; Ottoboni R. A Linearization Method for Comercial Hall-Effect Current Transducer. *IEEE Transactions on Instrumentation and Measurement* **2001**, *50*, 1149-1153.
7. James H. T.; Antoniotti A. J. Lineariation Algorithm for Computer Aided Control Engineering. *IEEE Control Systems* **1993**, *58-64*.
8. Patranbis D.; Gosh D. Anovell software based transducer linearizer. *IEEE Transaction Instrumentation and Measurement* **1989,** *38*, 1122-1126.
9. Malcovati P.; Leme C.A.; O´Leary P.; Maloberti F.; Baltes H. Smart sensor interface with A/D conversion and programmable calibration. *IEEE Journal of Solid State Circuits* **1994**, *29*, 963-966.
10. Yamada M.; Watanabe K. A capacitive pressure sensor interface using oversampling Δ-Σdemodulation techniques. *IEEE Transaction Instrumentation and Measurement* **1997**, *46*, 3-7.
11. Rahman M. H. R.; Devanathan R.; Kuanyi Z.. Neural Network Approach for Linearizing Control of Nolinear Process Plant. *IEEE Transactions on Industrial Electronics* **2000**, *47*, 470-476.
12. Shing K. H.; Chao H. J. Adaptive Reinforcement Learning System for Linearization Control. *IEEE Transactions on Industrial Electronics* **2000**, *47*, 1185-1188.
13. Dai X.; He D.; Zhang T.; Zhang K. ANN generalized inversion for the linearisation and control of nonlinear systems. *IEE Proceeding in Control Theory Appl.* **2003**, *150*, 267-277.
14. Schoukens J.; Németh J. G.; Vandersteen G.; Pintelon R.; Crama P. Linearization of Nonlinear Dynamics Systems. *IEEE Transactions on Instrumentation and Measurement* **2004**, *53*, 1245-1248.
15. Ashhab M. S.; Salaymeh A. Optimization of hot-wire thermal flow sensor based on a neural net model. *Applied Thermal Engineering* **2006**, *26*, 948–955 .
16. Ciminski A. S. Neural network based adaptable control method for linearization of high power amplifiers. *International Journal of Electronics an Communications* **2005**, *59*, 239-243.
17. Capitán V. L. F.; Arroyo G. E.; Fernández R. M. D.; Cuadros R. L. "Logit Linearization of analytical response curves in optical disposable sensors based on coextraction for monovalent anions". *Analytical Chemical* **2006**, *561*, 156–163.
18. Hutchins M. A. Twenty-First Century Calibration, *The Institution of Electrical Engineers the IEE. Savoy Place, London WC2R OBL, UK* **1996**, 1-6.
19. Dack P. So What Does Industry Want From Calibration? *The Institution of Electrical Engineers. by the IEE, Savoy Place, London WCZR OBL, UK*. 1995, 1-6.
20. Williams T. A. Quality Management: Quality Spending Nears $3B Mark. *Quality Magazine: Improving your Manufacturing Process* **2004**.
21. Robins M. Quality Management: $4.4 Billon and Growin". *Quality Magazine: Improving your Manufacturing Process* **2005**.
22. King D.; Lyons W. B.; Flanagan C.; Lewis E. An Optical-Fiber Sensor for Use in Water Systems Utilizing Digital Signal Processing Techniques and Artificial Neural Network Pattern Recognition. *IEEE Sensor Journal* 2004, 4, 21-27.
23. Zhao S.; Li B.; Yuan J., Zhao D. Key Elements Extraction Based On Particle Feature and RBFNN In New Meter Calibration Method. *IEEE International Conference on Industrial Technology* **2005**, 795-798.

24. Zhang G.; Saw S.; Liu J.; Sterrantino S.; Johnson D. K.; Jung S. An Accurate Current Source With On-Chip Self-Calibration Circuits for Low-Voltage Current-Mode Differential Drivers. *IEEE Transactions on Circuits And Systems* **2006**, *53*, 40-47.
25. Kikuchi S.; Tsuji H.; Sano A. Autocalibration Algorithm for Robust Capon Beamforming. *IEEE Antennas and Wireless Propagation Letters* **2006**, *5*, 251-255.
26. Depari A., Flammini A., Marioli D.; Taroni A. Application of an ANFIS Algorithm to Sensor Data Processing. *IEEE Transactions on Instrumentation and Measurement* **2007**, *56*, 75-79.
27. Depari A.; Ferrari P.; Ferrari V.; Flammini A.; Ghisla A.; Marioli D.; Taroni A. Digital Signal Processing for Biaxial Position Measurement With a Pyroelectric Sensor Array, *IEEE Transactions on Instrumentation and Measurement* **2006**, *55*, 501-506.
28. Hooshmand R. A.; Joorabian M. Design and optimisation of electromagnetic flowmeter for conductive liquids and its calibration based on neural networks. *IEE Proceedings Science Measurements and Technologies* **2006**, *153*, 139-146.
29. Abu N. K.; Lonsmann J. J. I. Calibration of a Sensor Array (an Electronic Tongue) for Identification and Quantification of Odorants from Livestock Buildings. *Sensors* **200***7*, *7*, 103-128.
30. Patra J. C.; Luang . A.; Meher P. K. A Novel Neural Network-based Linearization and Auto-compensation Technique for Sensors. *ISCAS* **2006**, 1167-1170.
31. Ji T.; Pang Q.; Liu X. An Intelligent Pressure Sensor Using Rough Set Neural Networks. *IEEE International Conference on Information Acquisition* **2006**, 717-721.
32. Hang T. M.; Howard D. B.; Mark B. *Neural Net Work Design*. PWS Publishing Company: Beijing, **2002**, pp12.1-12.27
33. Tai C. C.; Da J. H. Acceleration of Levenberg-Marquardt Training of Neural Network with Variable Decay Rate. *IEEE Proceeding of the International Joint Conference on Neural Networks* **2003**, 1873-1878.
34. Syed M. A.; Tahseen A. J. Cemal Ardil, Lenverg-Marquardt Algorithm for Karachi Stock Exchange Share Rates Forecasting. *Transactions on Engineering Computing and Technology, World Enformatika Society* **2004**, 316-321.
35. Van der Horn G., Huijsing J. L. *Integrated Smart Sensors Design and Calibration*. Kluwer Academic Publisher: Boston, MA, **1998**.
36. Fouad L. K.; Van der Horn G.; Huijsing J.L. A Noniterative Polynomial 2-D Calibration Method Implemented in a Microcontroller. *IEEE Transactions on Instrumentation and Measurement* **1997**, *46*, 752-756.
37. Maruyama M.; Girosi F.; Poggio T. A connection between GRBF and MLP. Technical Memo. AIM-1291. *Massachusetts Institute of Technology, Artificial Intelligence Laboratory* **1992**. 1-37
38. Hu Y. H.; Hwang J. N. *Handbook of Neural Network Signal Processing.* CRC Press: Washington D.C., **2002**.
39. Haykin S. *Neural Networks a Comprensive Foundation*. Prentice Hall: Upper Saddle River NJ, **1999**.
40. Dobelin E. *Measurement Systems Application and Design*. Mc Graw Hill: New York **2004**.
41. Ronald E. W., Raymond H. M., Sharon L.M. *Probability and Statistics for Engineers and Scientist*. Prentice Hall: Upper Saddle River NJ, **1999**.

42. ISO Guide to the expression of uncertainty in measurement; ISO Publishing **1995**.
43. NIST, Guidelines for Evaluating and Expressing the Uncertainty of NIST Measurement Results **1994**.
44. *ISO International vocabulary of basic and general terms in metrology*; Second Edition, ISO Publishing **1993**.
45. Hernandez, W., Improving the Response of a Rollover Sensor Placed in a Car under Performance Tests by Using a RLS Lattice Algorithm. *Sensors* **2005**, *5*, 613-632.