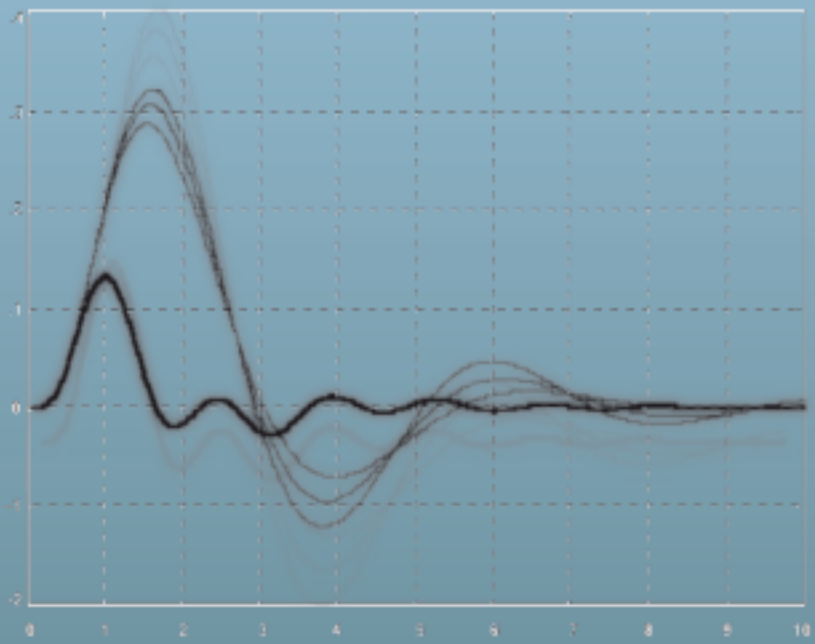
ISSN 1409-2441
www.ucr.ac.cr

Ingeniería

Revista de la Universidad de Costa Rica
ENERO/DICIEMBRE 2002 - VOLUMEN 12 - N°1 y 2



REVISTA INGENIERÍA Volumen 12 Enero / Diciembre 2002 Números 1 y 2

Ingeniería 12 (1,2): 105-114, 2 002 San José, Costa Rica

CALIBRATION OF A LOAD CELL USING A NEURAL NETWORK

Horacio Vásquez Céspedes

Abstract

A neural network is used to calibrate a load cell that was built using strain gages. The inputs to the neural network are the reference voltage applied to the Wheatstone bridge formed by the strain gages, the amplification value applied to the Wheatstone bridge's output voltage, and the 8-bit digitized voltage value acquired by a microprocessor. The output of the network is the estimated value of the weight being applied to the load cell. The network's main objective was to learn an accurate input-output relationship of the variables in the load cell system. The backpropagation Levenberg-Marquardt algorithm was used to train the network, and satisfactory results were obtained with a 5-3-1 neural network. This project could be used as an example to design similar neural networks for other applications.

1. INTRODUCTION

This project consists of calibrating a load cell using a neural network. The load cell was made with strain gages installed on a cantilever aluminum beam. The strain gages were connected to a circuit, called a Wheatstone bridge, and the output signal of the bridge was amplified and collected using an analog to digital converter, which was controlled by a Basic Stamp microprocessor. The microprocessor transmits the signal by infrared means to a computer that receives it via its serial port. This weighing system works well, but needs some type of calibration because of inherent non-linearities and variation of reference voltages. These are some of the main reasons why a neural network has been proposed to do such calibration. Also, since a computer is being used to collect and display the final result, it would be relatively simple to implement it. In essence, for this project, the neural network's most important purpose is to be able to determine an accurate weight value, no mattering where in the load cell range it is, and under different values of its inputs. This calibration problem seems to be a good application for a neural network because it requires a learning adaptable process.

2. BACKGROUND

An Artificial Neural Network (ANN) is a conglomerate of computational units, or neurons, organized in layers, and in its simplest form consists of an input layer, single or multiple middle layers, and an output layer. An important characteristic of an ANN is that all the neurons can be trained, under supervised or unsupervised learning, in an inter-dependent way, to associate, learn, and/or classify information.

After performing some research, it was determined that some previous works have been completed in similar applications of neural networks to calibrate, linearize, estimate, or fit data obtained from the behavior of a sensor or system. For example, a three-layer artificial neural network (ANN) was used to calibrate a displacement sensor [1,2]. The results obtained were compared to curve fitting results, and it was concluded that an ANN is the best candidate to do this kind of task. Also, it was determined that for this kind of application relatively few elements are required and the implementation is relatively simple. As another example, an ANN was used to linearize a thermistor behavior[3]. The purpose of this application

was to linearize the input-output relationship of a thermistor, and an ANN with two hidden layers was used. In addition, another application of an ANN, related to this project, is the calibration of the force/torque sensors used by a robot to estimate the weight of an unknown payload in order to applied the correct grasping force to pick up an object, avoiding slippage or damage of it [4]. One more case similar to this project was the use of a multi-layered ANN to calibrate a high-pressure measuring system with non-monotonic behavior and greatly influenced by temperature [5]. Better quality calibration than the one obtained with the spline-based method of calibration was obtained, and even for pressure reconstruction the measurement of temperature was not necessary [5]. Finally, another example of the use of an ANN for measurement calibration verification was done in power plants [6]. In power plants, as in many other applications, the instruments are re-calibrated on a periodic basis, and an ANN was used to predict the reading of an instrument using readings of other dissimilar instruments. I this power plant application of a neural network it was considered that a type of on-line parameter identification and model-based observer method was developed [6].

From those experiences, obtained with previous similar works, it was concluded that this project can accurately be solved using an ANN adequately trained to learn and estimate the related non-linear information provided for training. An extension of this project would be possible to many other similar calibration applications. After facing some problems with calibration of equipment because of non-linearities and variations, it is possible that while some equipment is operating "properly", with good sensitivity and repeatability, but not well calibrated, it could continue being used without trying to fix it or without getting rid of it. However, instead of that, a good signal processing would be required to compensate for the "bad" calibration, and in this particular application a neural network will do that job.

3. DESCRIPTION OF THE PROBLEM

This project consists of calibrating a load cell, made with strain gages installed on an aluminum cantilever beam with rectangular cross sectional area. One of the strain gages is on the top surface and the other at the bottom surface of the beam, which is loaded in bending. Figure 1. shows the load, P, the cantilever beam and one of the strain gages. The calibration is to be done using a neural network; therefore, the main objective is to design and train a neural network to correctly estimate the weight, or load P, applied to the load cell. A set of training cases of the known behavior of the system, some obtained experimentally and other obtained from the system's mathematical model, were used to instruct the neural network.

Notice that when one of the gages is in tension the other one is in compression; however, the strain of each gage is expected to have the same

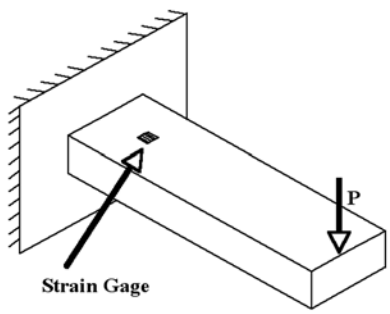


Figure 1. Cantilever beam as a load cell

magnitude. These strain gages, RT and RC, are connected to a Wheatstone bridge, as in Figure 2. The bridge is completed by adding two resistors, R, with nominal value equal to that of the strain gages. The reference, or input, voltage of the Wheatstone bridge is Vi, and it is one of the inputs to the neural network.

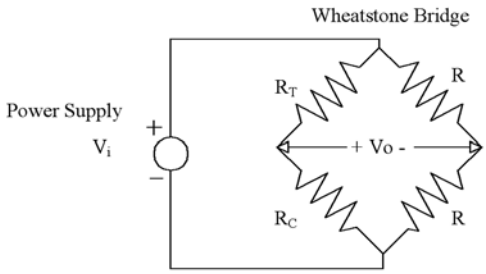


Figure 2. Wheatstone bridge connections

The bridge's output voltage, V_o, depends on the deformation of the strain gages and on the input voltage to the bridge, V_i.

An electronic circuit has been designed and implemented to condition and process the signal output from the bridge. First, there is an amplifier, with a gain, K_{amp}, in order to make the signal more manageable by other devices. The amplifier's gain, K_{amp}, is another of the inputs to the neural network because it is a variable quantity that could be adjusted and has to be measured from time to time, just to verify its value. This gain is supposed to not change too much; but, when it changes, it has an important effect on the results. Following the amplifier there is an 8-bit analogic to digital converter (8-bit A/D) which converts the amplified voltage to a 0 to 255 value (one byte). This digitized value is another of the inputs to the neural network.

Figure 3 presents a more complete idea of the weighing system. In this figure, the neural network's inputs and output can be identified. The first input to the network is the amplification value, which is called K_{amp}; the second input is the Wheatstone Bridge's reference voltage, V_i; and the third input is the digitized value, or binary value, that is proportional to the amplifier's output voltage. The amplifier's output is basically the Wheatstone bridge's output voltage, V_o, multiplied by K_{amp}. This binary value was obtained by digitizing the amplifier's output voltage using an 8-bit A/D converter driven by the Basic Stamp processor. The processor sends the information serially via infrared to a computer that is at a certain distance, 3 or 4 meters, from the measuring system. After that, the computer makes use of the trained neural network, and based on the three inputs performs the computation to estimate the weight applied to the load cell.

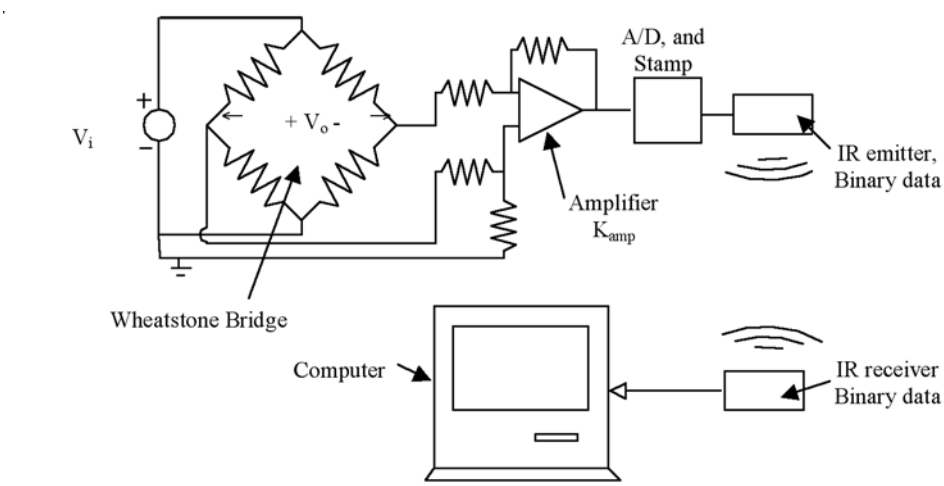


Figure 3. Load cell, signal acquisition and processing.

In summary, the Wheatstone bridge converts the weight acting on the load cell to a voltage, it is amplified and after that it is digitized. Theoretically, it is expected that the digitized value be proportional to the product of the weight applied to the load cell, the amplification value, and the Wheatstone bridge's reference voltage.

4. NEURAL NETWORK DESCRIPTION

The weighing system described above is made of several components, each introducing some uncertainty and error, and in general the final result, which is the weight being measured, is difficult to determine with the same accuracy all over the load cell's operating range. This occurs because of variations of the reference voltage applied to the Wheatstone bridge, variations in the amplifier's gain, and also to other nonlinear effects that are inherent to the system.

Inputs to the neural network:

There are three important parameters that were selected to be the inputs to the neural networks:

1- Amplification: K_{amp}
2- Input Voltage to the Wheatstone bridge: V_i
3- Binary information serially transmitted: Binary

The inputs to the ANN have the form shown in Eq. (1):

$$p\_input = \begin{bmatrix} K_{amp} \\ V_i \\ Binary \\ K_{amp} \times V_i \\ K_{amp} \times V_i \times Binary \end{bmatrix} \quad (1)$$

Notice that the input vector is a 5x1 in size, and its last two terms are a product combination of the first three terms; therefore, only the first three inputs are required and the programmed code takes care of adding the other two terms, as well as normalizing the vectors. These two dependent terms were required because the results obtained with only the first three terms were not satisfactory, and also because, theoretically, it is expected that the output from the ANN be proportional to the mutual product of the first three input terms. The input vectors are normalized by dividing each one by its magnitude.

Outputs from the neural network:

The network has a single output, which is the value of the weight applied to the load cell, and that is a function of the three inputs:

$$weight = f(K_{amp}, V_i, Binary) \quad (2)$$

The output training values of the neural network were linearly scaled to values between 0 and 1. As it was later determined the function in Eq.(2) is a combination of addition and multiplication of the three inputs. The whole purpose of the neural network could now be seen as the determination of the best function to map the weight surface as a function of the inputs.

Training examples:

Training examples were experimentally determined with several different known weights plus the offset value obtained when no weight is applied to the load cell. These weights were combined with four different reference voltages, V_i, applied to the Wheatstone bridge, and several amplification gains, K_{amp}, giving a total of 154 training input-output pairs, as shown in Table 1. All of these exemplars were used as an epoch to train the network.

Table 1. Training examples

	Kamp=200	Kamp=300	Kamp=401	Kamp=250	Kamp=200	Kamp=300	Kamp=200	Kamp=200	Kamp=800	Kamp=401	Kamp=200	Kamp=300	Kamp=401	Kamp=300
	Vin=5.01V	Vin=9.04	Vin=5.01V	Vin=12.05	Vin=7.45	Vin=7.45	Vin=12.05	Vin=9.04	Vin=5.01V	Vin=9.04	Vin=12.05	Vin=12.05	Vin=7.45	Vin=5.01
Weight (lb)	Binary	Binary	Binary	Binary	Binary	Binary	Binary	Binary	Binary	Binary	Binary	Binary	Binary	Binary
0.00	30	74	60	79	41	62	63	49	117	98	63	95	83	45
0.51	33	82	66	88	46	68	70	54	128	109	70	105	89	50
1.10	36	91	73	99	50	75	79	60	140	121	79	119	100	55
1.62	39	98	79	106	54	82	85	65	151	131	85	128	109	59
2.07	42	105	84	115	58	87	92	69	161	141	92	138	115	63
2.58	45	113	89	124	62	93	99	75	171	151	99	149	123	67
3.10	48	121	95	133	66	99	106	81	182	161	106	159	132	72
3.62	51	129	101	140	70	105	112	85	192	172	112	168	139	76
4.14	54	136	106	150	74	111	120	90	203	182	120	180	148	80
4.66	56	144	112	158	78	117	126	95	214	192	126	189	155	84
5.11	59	151	117	165	81	122	132	99	223	202	132	198	163	88

Table 2. Test Samples

Test #	1	2	3	4	5	6	7	8	9	10
Kamp	250	200	350	350	250	350	250	350	300	401
Vi	5.01	6.23	9.04	7.45	5.01	9.04	7.45	7.45	9.75	8.25
Binary	37.5	35.5	106	87.5	52.5	141	92.5	142.5	100	120
Weight (lb)	0	0	1.1	1.1	2.07	3.1	4.14	5.11	1.68	1.72

Test Samples:

After performing the training, test input cases, obtained by interpolation in Table 1, were applied to the network to determine its performance. These test samples are presented in Table 2.

In the same way as it was done for the input training examples, any input vector presented to the neural network requires two additional terms as in Eq. (1) and after that it must be normalized.

Neural Network Architecture and Training:

Since sigmoidal transfer functions were chosen for the middle layers and a linear transfer function was used in the output layer, the training examples were normalized to have proper values to work with these transfer functions.

Training was performed using backpropagation with the Levenberg-Marquardt (BPLM) algorithm as numerical optimization technique for relatively fast convergence[7]. The advantage of the Levenberg-Marquardt technique is that the results always go in the right direction, as long as that direction exists, and that is, toward the minimum of the error surface. On the other hand, the most significant disadvantage of this technique is the computation of the modified pseudo-inverse of the Jacobian, $(J^TJ + \mu_k I)^{-1}$, for every batch iteration. Where J is the Jacobian, and μ_k is a variable coefficient that is changed to adapt the algorithm to the progress accomplished after every iteration. Sometimes this matrix becomes too big and computing the inverse might be too complicated. For example, for a 3-9-1 network, this matrix is of size 46x46, and for a 3-5-3-1, this matrix is 42x42. Fortunately, nowadays there are powerful software tools and computers that perform this computation in incredible short times.

The back propagation algorithm with momentum and variable learning rate, VLBP, was used to solve the problem but it was difficult to control mainly because of the required adjustments of

parameters, and because of the convergence to stationary points that are not the desired minimum[7]. The VLBP technique could work to solve this problem, but most of the efforts to find a solution to this problem were focused on the Levenberg-Marquardt algorithm because of the reason given above.

The data in Table 1 was presented to the network input after input in a sequence row by row and the total squared error was computed per each epoch, and this error was used to start the backpropagation algorithm.

5. RESULTS

A small number of neurons, 4 and 5, were used at first in a single middle layer to train the neural network. However, progressively, to get familiar with the network, some other tests were performed using more, or fewer, neurons, as well as more middle layers as needed or just to determine what kind of results were obtained. While training was being performed, the results were displayed every hundred epochs, by plotting three graphs in a figure. The graph on the top of the figure shows convergence of the network results to the training examples, the graph in the middle presents the squared error per epoch, and the bottom graph shows the convergence of the behavior of the network to the test samples.

Most Important Result:

The most important result was obtained with a 5-3-1 neural network with sigmoidal middle layer transfer functions and linear transfer function for the output layer. Figure 4. presents the results after 5 400 epochs, but only the last 300 epochs are shown in the figure.

The minimum squared error per epoch was 0,025, as can be seen in the middle graph, and after this point the Jacobian matrix started getting close to be singular, and the program could not compute its pseudo-inverse.

It was determined that the training examples as well as the test examples were well approximated. The top graph of Figure 4. represents the convergence of the neural network results to the training examples, and columns 2 and 12 of Table 1. were used for this purpose. The horizontal axis of this graph represents the input vector according to its position in columns 2 and 12 of Table 1. The “x” marks represent the expected weight value, that is the values in column 1 of Table 1. Close attention to the top graph reveals that there are two curves, as expected, and since they are converging to the same values of weights in some parts of the graph it seems that there is only one curve. Moreover, the bottom graph presents the results obtained using the test samples from Table 2. The horizontal axis of this graph represents the input

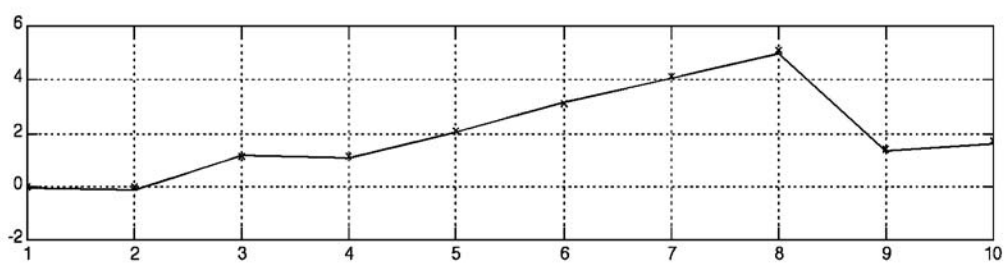


Figure 4. Neural network 5-3-1.

Table 3. Neural network results to the test samples, for a 5-3-1.

NN Result	Expected Result	% Error
-0,088	0	
-0,010	0	
1,15	1,10	5,2
1,07	1,10	-1,9
2,04	2,07	-1,1
3,17	3,10	2,3
4,08	4,14	-1,2
4,98	5,11	-2,5
1,32	1,39	-4,9
1,61	1,72	-6,4

vector according to its position in the columns of Table 2. The "x" marks represent the expected weight value associated with the values of row 5 of Table 2. Again, the neural network generates an acceptable approximation of the test samples, with errors under 6,4 % for all of the test points, as shown in Table 3. This kind of error is acceptable but there is always a desired to reduce it more.

The matrices of weights and biases obtained for this 5-3-1 network are presented next:

$$W^1 = \begin{bmatrix} 293{,}8 & 4367{,}4 & -849{,}9 & 235{,}2 & 7422{,}4 \\ 286{,}4 & 4325{,}9 & -920{,}2 & 240{,}8 & 9250{,}3 \\ 78{.}, & 1264{,}0 & 318{,}5 & 38{,}6 & 2699{,}5 \end{bmatrix}$$

$$W^2 = \begin{bmatrix} 5449{,}5 & -2807{,}1 & 285{,}5 \end{bmatrix}$$

$$b^1 = \begin{bmatrix} -7418{,}6 \\ -9247{,}3 \\ -2695{,}4 \end{bmatrix}$$

$$b^2 = -2926{,}8$$

There is some similarity between rows 1 and 2 of the weight and bias matrices for layer 1, which are W1 and b1. This might mean that there is a redundant function of a neuron in such a layer; therefore, fewer neurons in that layer could generate similar results. Then, the following step taken was to decrease the number of neuron in the middle layer to 2 instead of 3. However, the results obtained with this 5-3-1 neural network are difficult to improve due to its good results during training as well as during interpolation.

Second Most Important Result:

The second important result obtained is presented in Figure 5. A 5-2-1 neural network with sigmoidal middle layer transfer functions and linear transfer function in the output layer was trained for 3 500 epochs, and the figure presents the last 500 epochs.

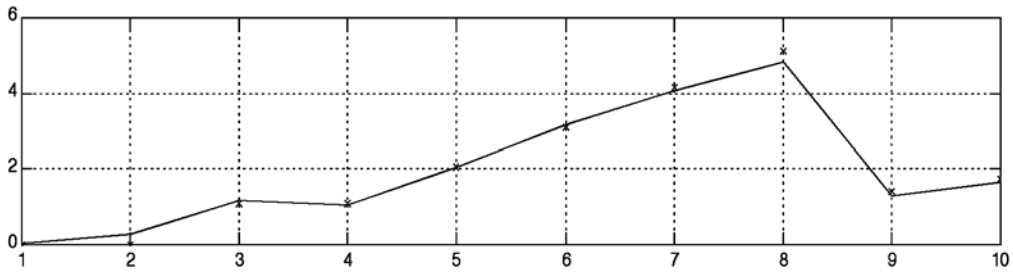


Figure 5. Neural network 5-2-1.

In this case, the minimum squared error per epoch that was obtained was 0,054, as can be seen in the middle graph. The top graph of Figure 5. shows a slightly difference in the training convergence of the neural network results for columns 2 and 12 of Table 1. The bottom graph of Figure 5. presents the results obtained using the test samples from Table 2. The network generates acceptable results, and it seems to be interpolating correctly, but there are larger errors compared to the results obtained with a 5-3-1 network. Table 4 presents a comparison of the expected results for the test samples and the results obtained with the 5-2-1 network.

Comparing results in Table 3 with the ones in Table 4, it is determined that, for almost all the test samples, the percentage of error is larger in Table 4. However, a good point in favor of the 5-2-1 network is that the error is banded within -6 % and 4,7 %, while for the 5-3-1 it is banded within -6,4 % and 5,2 %. A critical result of the 5-2-1 network is the one at test sample 2, which seem to be too far from the desired result.

Table 4. Neural network results to the test samples, for a 5-2-1.

NN Result	Expected Result	% Error
0,05	0	
0,28	0	
1,15	1,10	4,7
1,06	1,10	-3,8
2,04	2,07	-1,4
3,19	3,10	3,0
4,07	4,14	-1,6
4,83	5,11	-5,5
1,30	1,39	-6,0
1,64	1,72	-4,4

The matrices of weights and biases obtained for this 5-2-1 network are presented next:

$$W^1 = \begin{bmatrix} -18{,}8 & -364{,}9 & -52{,}4 & -128{,}6 & -1737{,}0 \\ -18{,}3 & -347{,}3 & -109{,}0 & -135{,}1 & -2904{,}9 \end{bmatrix}$$

$$W^2 = \begin{bmatrix} 3506{,}5 & -2298{,}5 \end{bmatrix}$$

$$b^1 = \begin{bmatrix} 1734{,}2 \\ 2903{,}3 \end{bmatrix}$$

$$b^2 = 4{,}6$$

Third Result:

Another important result is presented in Figure 6. A 5-1-1 neural network with sigmoidal middle layer transfer functions and linear transfer for the output layer was trained, and Figure 6. presents the results after 1 000 epochs.

In this case, the minimum squared error that was obtained was 0,98, as can be seen in the middle graph. The top graph of Figure 6 shows great difference in the training convergence of the neural network results for columns 2 and 12 of Table 1. The bottom graph of Figure 5. presents the results obtained using the test samples from Table 2. The network seems to be interpolating with a correct tendency but the errors are large. The training examples are mapped with large error, compared to the two previous 5-3-1 and 5-2-1 networks.

6. CONCLUSIONS

This project consisted of designing, training, and testing a neural network to calibrate a load cell made with strain gages installed on an aluminum beam. Two strain gages and two external resistors formed a Wheatstone bridge, and the output signal from the bridge was first amplified and later acquired using an analog to digital converter, which was controlled by a Basic Stamp microprocessor. The inputs to the neural network were the reference voltage applied to the Wheatstone bridge, the amplification gain applied to the Wheatstone bridge's output voltage, and the digitized voltage value acquired by a microprocessor. The network's output was the computed value of the weight being applied to the load cell.

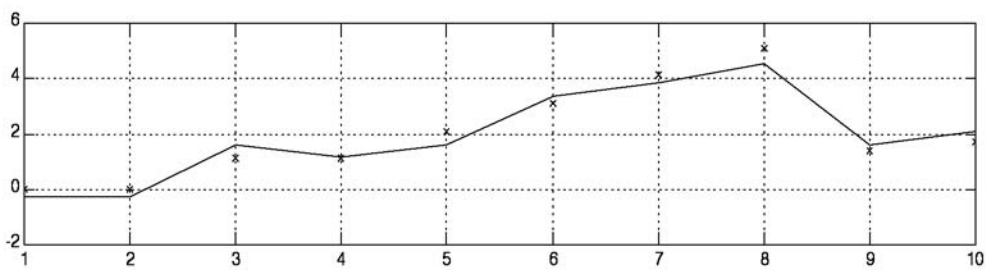


Figure 6. Neural network 5-1-1.

The network's main objective was learning an accurate input-output relationship of the variables in the load cell system and it was achieved using and a 5-3-1 configuration. The backpropagation Levenberg-Marquardt algorithm was used to train the network and a minimum squared error per epoch of 0,025 was obtained as the best result for the 154 input-output exemplars. The network was tested and errors under 6 % were obtained for all the test samples. This magnitude of error is satisfactory for this application considering that some noise was on purpose included in some of the training data, and also taking into account that the test data was obtained by interpolation from the training data. It is possible to do a more strict control and selection of the training and testing data in order to obtain less error in the testing results. More training data might be added and maybe some training data might be removed to optimize training while still getting satisfactory results. Fortunately, with the BPLM programmed code for this project, training of the neural network takes from 20 to 30 minutes using a computer with a 677 MHz processor, and a faster computer would perform the training in less time. In general, some additional adjustments could be made to try to improve results, like changing the transfer functions, or adding new training data in regions where better accuracy is desired.

Finally, based on the adequate results obtained in this project, it is concluded that the theory and methods of neural networks offer a good option to calibrate, or obtain a satisfactory input-output relationship for an instrument, equipment, or system that operate under variable input conditions. The flexibility of the neural network is such that in whichever other projects, or applications, there is always the possibility for changes, new ideas, and modifications that could be incorporated to the network to achieve the objectives. In this project, a 5-3-1 neural network accomplished the main objective of determining an accurate weight value, no mattering where in the load cell range it was, and based on different values of the neural network's inputs.

7. REFERENCES

1. Masory, O.; Aguirre, A.; N*eural network calibrates a displacement sensor*. Sensors,v 7, n 3, Mar 1 990, p 48-54

2. Masory, O.; Aguirre, A.; *Sensor Calibration methods -performance study. Proceedings of SPIE*. International Society for Optical Engineering, Applications of Neural Networks. Apr 1 990, v 1294, p 490-501

3. Attari, M.; Boudjema, F.; *Linearizing a thermistor characteristic in the range of zero to 100 degree C with two layer ANN*. Proceedings of the 1995 IEEE Instrumentation and Measurement. April 1995, p 119 - 122

4. Garg, D.; Ananthraman,S.; *Sensor integration for payload weight estimation in a robotic work cell*. ASME, Dec 1 991, v 30, p 25-29

5. Massicotte, D.; Legendre, S.; N*eural-network-based method of calibration and measurand reconstruction for a high-pressure measuring system*. IEEE Transactions on Instrumentation and Measurement. V 47 n 2 Apr 1 998 p 362-370

6. Ipakchi, A.; Khadem, M.; *Neural Network applications to measurement calibration verification in power plants*. Proceedings Instrumentation in the Power Industry. Jun 1991, v 34, p 501-511

7. Hagan, M.; Demuth, H.; Beale, Mark. *Neural Network Design*. PWS Publishing, Boston, 1995.

SOBRE EL AUTOR

Ing. Horacio Vásquez Céspedes
Profesor Asociado de la Escuela de Ingeniería Mecánica, Universidad de Costa Rica
Tel /Fax: 207-5610