Journal of the Korean
Data & Information Science Society
2000, Vol. 11, No. 2, pp. 157 ~ 172

Application of Neural Networks in Aluminum Corrosion

John Powers [1] · **M. Masoom Ali** [2]

Abstract

Metal containers represent a situation where a specific metal is exposed to a wide variety of electrolytes of varying degrees of corrosivity. For example, hundreds, if not thousands of different products are packaged in an aluminum beverage can. These products vary in pH, chloride concentration and other natural or artificial ingredients which can effect the type and severity of potential corrosion. Both localized (perforation) and uniform corrosion (metal dissolution without the onset of pitting) may occur in the can. A quick test or series of tests which could predict the propensity towards both types of corrosion would be useful to the manufacturer.

Electrochemical noise data is used to detect the onset and continuation of pitting corrosion. Specific noise parameters such as the noise resistance (the potential noise divided by the current noise) have been used to both detect pitting corrosion and also to estimate the pitting severity. The utility of noise resistance and other electrochemical parameters has been explored through the application of artificial neural networks. The versatility of artificial neural networks is further demonstrated by combing electrochemical data with electrolyte properties such as pH and chloride concentration to predict both the severity of both localized and uniform corrosion.

Key Words and Phrases: Neural networks, Electrochemical noise, Aluminum, Pitting, Corrosion.

1. Introduction

Electrochemical noise is the fluctuation of current or potential with time measured in a corroding system. It is well established that electrochemical noise will detect the onset and continuation of pitting corrosion vs uniform corrosion (Uruchurtu and Dawson (1987). A wide variety of analytical methods for electrochemical noise

[1]Ball Corporation 9343 West 108 Circle Broomfield, CO 80021 USA
[2]Department of Mathematical Sciences, Ball State University, Muncie, IN 47306-0490 USA

have been published (Gabrielli *et al* (1990), Kearns *et al* (1996), Roberge (1994), Sharland *et al* (1990)). Most of these methods focus on the differentiating between a uniformly corroding system and a perforating system. Less attention has been paid to measuring the severity of the perforating corrosion by electrochemical noise.

The derived noise resistance is the standard deviation of the potential divided by the standard deviation of the current. This parameter has been correlated with the severity of corrosive attack (Chen and Skerry (1991). It can also be defined as the root mean square of the potential divided by the root mean square of the current.

This paper looks at the utility of noise resistance and other factors in measuring the severity of localized (perforating) corrosion through the application of neural networks. Neural networks is also shown to simultaneously correlate selected parameters with both pitting severity and the metal dissolution rate.

Neural networks are versatile mathematical analysis tools (Chester (1993) and Smith (1993)). They are capable of pattern recognition and curve estimation (among other things). Very few examples of the application of neural networks to electrochemical noise are available in the literature (Henriksen *et al* (1994) and Barton *et al* (1993)). Those which have been reviewed used neural networks to detect pattern changes in the noise signals which signify the onset of localized corrosion.

Neural networks, or artificial neural networks, arose out of attempts to simulate the processes of the brain. The human brain consists of a huge network of cells (neurons) which are connected via synapses. It is known that signals are conveyed from one neuron to another via a biochemical mechanism at the synapse boundaries. The underlying reasons for a neuron to transmit ("fire") or not to transmit are not fully understood. The assumption is made that the transmission occurs when the information contained in the neuron is germane to the mental activity taking place.

Artificial neural networks employed in this paper are mathematical simulations of the brain architecture. Each node (neuron) contains numerical information which is passed on to, or received from, other nodes. Many nodes will both receive and pass on numerical information. If a node's role includes the passing on of information, a function is in place which controls how information is passed on to other nodes.

The mind can receive and process information into a conclusion. Artificial neural network receives numerical information and mathematically transforms it to a numerical conclusion. *A most important feature of an artificial neural network is that it can be "trained" to reach known conclusions when presented with the proper input.* While the detailed operation of an artificial neural network is beyond the scope of this paper, a simplified description is necessary for the reader to appreciate the process.

The operation of a single node in a neural network can be envisioned as shown

in Equation (1.1).

$$Input \rightarrow Process \rightarrow Output \tag{1.1}$$

Information is received, processed and passed on. As the word network implies, a neural network is a complex organization of many nodes, which may receive, process, or transfer numerical information. Some nodes (called hidden nodes) perform all three operations. These networks have a parallel structure and often have more than one layer. It is this complex organization which allows the network to effect highly complex correlations. The first column of nodes is called the input layer. These nodes are the source of thedata. The second column is referred to as the hidden layer as its output is not directly observed. The third column is the output layer. Some networks have two hidden layers. It should be noted that each node in the hidden layer receives data from each input node, and each output node receives input from each hidden node.

A common class of neural networks is the multi-layered perceptron. In this type of network, every received data value is multiplied by a coefficient and summed. The sum is then added to a constant. The resulting value may be processed by a transfer function before it is passed on to the next nodes. The scheme is shown below:

$$Input \rightarrow \sum w_i x_i + a_0 \rightarrow F \rightarrow Output, \tag{1.2}$$

where
x_i's are the input values received from other nodes,
a_i's are the weight coefficients,
a_0 is a constant and,
F is the transfer function.

The transfer function can simply pass on the value as is (linear transfer) or scale between 0 and 1 or -1 and 1. Theoretically any increasing function which is differentiable can be used. The most common are the linear, sigmoid and hyperbolic tangent functions.

The scheme of a three layer network is presented below. The first layer of nodes consists of the inputs, which are the data set. The second layer andthird layers are the hidden and output nodes, respectively. The scheme is:

$$x_i \rightarrow \sum a_{ij} x_i + a_0 \rightarrow F_j \rightarrow \sum b_{jk} y_j + b_0 \rightarrow F_k \rightarrow z_k, \tag{1.3}$$

where
x_i's are the input data,
a_{ij}'s are the weighting factors for the j hidden nodes, $i = 1, 2, \ldots, I$,
a_{j0}'s are the constants of the j hidden nodes,

y_j's are the outputs of the hidden nodes,
b_{jk}'s are the weighting factors of the k output nodes,
b_{k0}'s are the constants of the k output nodes,
Z_k's are the outputs for the k output nodes,
F_j's are the transform function of the hidden nodes,
F_k's are the transform function of the output nodes.

It can be seen that a neural network consists of a very complex set of equations which transform a set of numerical inputs to a set of numerical outputs. The "training" of a neural networks is the process in which the weighting coefficients and constants of the numerous nodes are adjusted so that a specified set of inputs yield the stipulated outputs. This process can be envisioned as non-linear regression without a specific model equation.

A typical non-linear regression exercise consists of adjusting the coefficients of given equation to get the "best fit" for a data set consisting of independent variables with a corresponding dependent variable. Training a neural network is similar. A target output is established for all inputs from a given example. The inputs are then processed through the network and an error function is calculated. The error function is used to calculate various derivatives, which in turn are used to establish corrections to the weights and biases of the network. The corrections start at the output node and are cascaded back through the network. This cascading backwards is the origin of the term "backpropagation." The backpropagation paradigm was developed in the 1980s. The concept was first described by Paul Werbos in his 1974 doctoral dissertation which was further elaborated in a text (Demuth and Beale (1994)).

An advantage of the network is that input data can be correlated to certain outcomes without the use of a specific relationship. In this study, electrochemical and other parameters of corroding metal are correlated with known corrosion outcomes, without the use of an exact equation. This is in contrast to most scientific research, where the objective is to quantitatively determine cause and effect relationships which is characterized by a specific equation. This "equationless" feature may be a boon to the applied scientist and engineer who must solve real problems, regardless of whether exact relationships are known.

2. Experimetal

The aluminum coupons were mounted in an E.G. & G. model K0235 flat cell holder. A flat sample cell was modified so that both the working and counter electrodes were aluminum coupons with one square centimeter exposed to the solution. A standard calomel electrode was again connected as the reference. The Gamry

120 Electrochemical Noise Test software was used. The *ZECNRAW* procedure was followed in which the potential of the aluminum vs. the reference electrode was measured virtually simultaneously with the measurement of the current between the two coupons. The output of this procedure included the potential and current time series as well as two values designated as potential noise and current noise. These latter values are the root mean square of the potential and current after the linear trend had been subtracted from the data.

The first solution, Solution A, consisted of low concentration of sodium metasilicate dissolved in distilled water. Solution A provided the examples of uniform corrosion. Solution B consisted of a mixture of citric acid, sodium chloride, and red dye #40. This solution provided the examples of localized (perforating) corrosion.

Solutions C through F were commercial beverages. A total of 68 files with 600 data points each were collected with these solutions. In all cases except Solution C, the data was collected after 24 hours exposure to the solution. With Solution C, some data was collected after only 17 hours after exposure.

In addition to the electrochemical data, the pH and chloride ion concentration of the six test solutions were measured. These values served as additional input parameters.

Two output targets were employed. The first output target was related to the severity of the pitting attack and the second target was related to the dissolution rate of aluminum. The pitting severity of a solution was ranked between 0.1 and 0.9. This scale was chosen to accommodate one of the software programs employed. The rankings were somewhat arbitrary and were based on the appearance of the coupons after 48 hours and known historical data on some of the solutions.

The second set of targets were established directly from objective experimental data. Aluminum coupons which were twelve square centimeters in area were immersed in 50 mL of the solutions for 48 hours at ambient temperatures. The dissolved aluminum concentration was then measured by atomic absorption spectroscopy. The measured concentrations were then divided by 2 so that they would fit within the 0 to 1 range of the logistic function output.

3. Data Transformation

The raw data was transformed for a number of reasons both of necessity and for convenience. In some cases it was necessary to transform the data in order to avoid the error in minimization process from getting trapped in local minima. Transformations of scale and range were completed in these cases. The specific

application and justification for these transformations are provided in the following discussions.

Other transformations included the use of aggregate properties in order to reduce the number of input parameters. A segmented parameters such as the mean moving standard deviation removed both drift and produced aggregate properties. The mean moving standard deviation was determined by calculating the standard deviation for the first sixteen data points. The process was then repeated but advancing one data point and again taking the standard deviation of sixteen continuous points (data points 2 through 17). This process was repeated to the end of the data set. The mean of all the standard deviations calculated was determined and this mean was taken as the measure of the potential or current noise. This process is analogous to a moving average for non-stationary data (Bendat and Pearsol (1986)).

The *ZECNRAW* procedure also generates values called potential and current noise. These values are generated by taking the root mean square (RMS) of the potential and current after the linear trend has been subtracted from the data. These values were used in one set of network training exercises and then dropped in favor of the mean moving standard deviation.

There were two reasons for discontinuing the use of the software generated potential and current noise. The first was that drift in the signal was not always linear over the six hundred seconds of data collection. Therefore, the subtraction of a linear trend would lead to inaccuracies. The second reason was the belief that the moving standard deviations produced essentially the same values while more accurately eliminating the effects of the signal drift. The RMS for a signal which is drifting in a non-linear fashion would be dependent on the portion of the curve which is analyzed. Six hundred data points were taken in a typical experiment in this study. It was suspected that the RMS taken for the first hundred data points, would differ from the RMS taken from the entire six hundred data points. To test this, the RMS was calculated for six different segments of the data from one file. All data segments started with the first data with the first calculation including the first one hundred points. Each subsequent determination added an additional 100 data points. The range between the highest and lowest RMS was calculated. This process was repeated with the mean moving standard deviation.

Table 1 compares the range of the root mean square of the potential with the mean moving standard deviation of the potential for different data sets taken from the same corroding system. It can be seen that the range of the root mean square can be as high as fifteen times greater than that for the mean moving standard deviation of sixteen points.

Table 1

The Range of Potential Noise Calculations Vs Method of Calculation

Hours	RMS range	Stdev (16)	Ratio
24.0	0.000550	0.000344	1.6
25.0	0.000586	0.000234	2.5
26.0	0.000432	0.000665	0.6
27.0	0.000465	0.000393	1.2
28.0	0.002203	0.000993	2.2
29.0	0.000227	0.000251	0.9
43.5	0.000832	0.000311	2.7
44.0	0.000507	4.841E-05	10.5
44.5	0.000462	8.753E-05	5.3
45.0	0.000168	2.409E-05	7.0
45.5	0.000228	0.000111	2.0
46.0	3.811E-05	2.474E-06	15.4

For experiments of shorter duration with smaller intervals between the data points, the RMS estimate of potential or current noise would serve well.

The average current and potential were also used. A fifth parameter recorded was the so-called noise resistance. This was the quotient of the potential noise divided by the current noise. This value is believed to be a strong indicator of perforating corrosion by many authors. In those cases (the majority) where the potential noise and current noise were not used, the noise resistance was calculated as the quotient of the mean moving standard deviation of the potential divided by the mean moving standard deviation of the current.

The pH and chloride ion concentration of the six solutions were also recorded. This data was also transformed by the methods described above. Sixty-eight files were available. Forty-six were used for training and twentytwo for validation exercises.

4. Network Architecture

Several network designs were used to process the data. These include:

1. A multi-layered perceptron with an adaptive learning rate and momentum which are algorithms for finding the minimum error. This design was tested with a program in Smith (1993) and the trainbpx procedure in Matlab.
2. A multi-layered perceptron with the Levenberg-Marquardt training algorithm

which is the trainlm procedure in Matlab.
3. A multi-layered perceptron with the conjugate gradient algorithm with the Neural Connection software.
4. Radial basis networks which are the solverb procedure in Matlab.

The different designs were compared with regard to accuracy, speed, and complexity of the solution. In order to compare the Smith program with Matlab and Neural Connection, the logistic function was used as the transfer function when backpropagation was used in Matlab. As a cross-check between the two programs, coefficients from Matlab were used in the Smith mapping function and identical results were obtained.

In the supervised training exercises, the network is presented with data from the input nodes. With each set of input data, target values are also given. The combination of input and target values are referred to as examples. They may also be referred to as a file. The network then proceeds with the feed forward step, calculates an error function and then backpropagates corrections on the weighting factors and biases. Each such cycle is referred to as an epoch. The training exercise terminates when either a specified error goal, expressed as the sum of squares error (SSE), or a pre-specified number of epochs is reached.

The error function is generated via the difference of the output nodes and the targets. Corrections are made to the weighting and offset. The feed forward and backpropagation of corrections proceed until a specified error goal is met or the maximal number of epochs is reached.

5. Results

The results are reported here as tabular presentations of the validation exercises. The SSE for the training exercise was set so low that all training examples would have necessarily been "mapped' within a very close agreement to the target. It is then necessary to check against overtraining by mapping the validation examples. If the validation examples were close to target then a network has been successfully trained. The output of each validation example is shown with its true target.

A training exercise was judged by two criteria. The first was the ability to reach a specified error target which is expressed as the sum of squares of the error (SSE). Generally speaking, a minimum SSE of 0.1 was expected. This meant that the worst output for any single file in the training set would be 0.317 from target. That represents the extremely unlikely case where all the error occurred in one of the 92 outputs.

The second criteria is the fit of the validation set of files. Over training must be

avoided. The training exercise can result in a solution which is so narrow in scope that only those files which were included in the exercise can be accurately mapped.

Training data sets for the six solutions consisted of seven input nodes and two output nodes. The multi-layered perceptrons with the Smith and Matlab software programs used one hidden layer with 15 nodes. The number of hidden nodes with the Neural Connection multi-layered perceptron and the radial basis method varied with the individual exercise. The various training sets differed in which files were chosen for training and which were chosen for validation. The training sets also differed with regard to how the data had been transformed. The target values did not vary.

Training with data which had not been transformed yielded poor results. In many cases, the networks could not achieve the desired error targets (SSE). The training appeared to get "trapped" in local minima. While some authors suggested that local minima should not be a problem when multiple factors were involved, this did appear to be a problem with the untransformed data. There were large differences in the order of magnitude between the different input factors and it is hypothesized that the factors with the largest values dominated the process. This could reduce the exercise to the point where there were only two or three effective factors involved and localized minima became more likely.

One data transformation which led to successful training was to multiply the data by a power of ten which transformed the largest values in each factor to a value between 0 and one. This resulted in data which could still range over three orders of magnitude in certain factors. Two more conventional transformations included normalizing each factor between 0 and 1, and converting the values in each factor to standard normal deviates.

The method used to separate the data into training and validation also affected the results. The objectivity of a purely random sampling scheme or a systematic sampling scheme in which every third file is assigned to the validation set has some appeal, but it can yield inferior results. The best training results when the training set contains the maximum and minimum value of each factor.

Tables 2 and 3 show the targets and calculated values which had been systematically separated by choosing the first two files in the sequence for training and the third for validation. This trial set could be trained with only fair success Problems arose in the validation set with Solution E and F and occasionally others. Table 2 shows typical results with this approach using the adaptive learning rate algorithm. Solution A exemplifies a good validation while Solution F shows the problems. Table 3 shows typical results with the Levenberg-Marquadt algorithm. Interestingly, a better fit was obtained with Solution F even though the SSE of the training exercise was an order of magnitude higher. This shows that overtraining was occurring with

the adaptive learning rate program with regard to Solution F.

The trial set was reorganized so that in almost all cases, the training files included the highest and lowest values for the given parameter. This trial set could be trained with great success on the Smith program. However, low SSE values could not be reached with the Levenberg-Marquardt algorithm and poor validation results were obtained with the radial basis method.

Table 2
Validation Results
1723 Epochs, SSE = 0.00984
Smith Backpropagation Program with Adaptive Learning Rate and Momentum
Solution A

Target	1	2	3	4
0.90	0.897	0.885	0.900	0.898
0.95	0.945	0.987	0.954	0.856

Solution F

Target	1	2	3
0.40	0.614	0.736	0.659
0.18	0.312	0.171	0.288

Table 3
Validation Results
33 Epochs, SSE = 0.0985
Levenberg-Marquardt Program on Matlab

Solution A

Target	1	2	3	4
0.90	0.9861	0.944	0.859	0.933
0.95	0.929	0.925	0.897	0.986

Solution F

Target	1	2	3
0.40	0.462	0.491	0.469
0.18	0.151	0.152	0.163

Training sets with the data ranged from 0 to 1 were successful with both backpropagation and radial basis networks. The Levenberg-Marquardt algorithm gave

good results, except with Solution E. Curiously, this last data set could not be successfully trained on the Smith program, with local minima the apparent problem.

The same training set with data which had been transformed into the standard normal deviates for each field was trained with the Neural Connection software. The results of this exercise is shown in Table 4. This table shows the mapping of all 22 validation examples. The results are similar to the successful exercises mentioned above with different data transformation or network architectures. Only one table is shown to avoid unnecessary redundancy.

Table 4

Neural Connections with Conjugate Gradient Algorithm

Solution A

Target	1	2	3	4
0.90	0.902	0.905	0.904	0.903
0.95	0.931	0.972	0.966	0.946

Solution B

Target	1	2	3
0.10	0.100	0.100	0.100
0.05	0.0508	0.0501	0.0489

Solution C

Target	1	2	3	4
0.70	0.696	0.683	0.703	0.698
0.20	0.204	0.214	0.209	0.207

Solution D

Target	1	2	3	4	5
0.30	0.301	0.297	0.300	0.296	0.307
0.90	0.901	0.900	0.899	0.893	0.898

Solution E

Target	1	2	3
0.90	0.906	0.904	0.904
0.80	0.796	0.802	0.802

Solution F

Target	1	2	3
0.40	0.400	0.398	0.412
0.18	0.158	0.148	0.174

A successful training exercise was accomplished with a radial basis architecture when the data was ranged from 0 to 1.

6. Discussion

The results showed that data preparation and the network architecture had a large effect on network efficiency and accuracy. The networks could differ from one another in terms of accuracy, speed (the number of epochs required to train), and the tendency of the training process to be "trapped" in local minima. Furthermore, the relative performance of the various networks depended on the specific data set and how that data had been transformed. This shows the advantages of having different network architectures available.

The speed with which networks could be trained was often dependent on architecture. Within the multi-layered perceptrons, it was found that the Levenberg-Marquardt algorithm was much faster than the adaptive learning rate. The radial basis method was found to be among the fastest methods, if not the fastest. However, as indicated above, these relationships do not hold for every data set.

The tendency towards local minima is affected by both the idiosyncrasies of the individual data set and the initial weighting factors and biases. The inability of different networks to train below certain error goals was frequently observed. These problems were sometimes removed by processing the data in different ways. While problems with localized minima can often be overcome by changing the initial weights, learning rates, or adding momentum, this can be a painstaking process. It was often more efficient to quickly try several different architectures.

The radial basis method showed several important characteristics with regard to these experiments. While the radial basis method was unable to solve certain data sets, it was very efficient with those exercises within its capability. Most training exercises were completed with less than 30 epochs. The Matlab solverb procedure added a node with every epoch. This could result in a network with a huge number of weighting factors and biases. However, this is not a problem unless one desires to print all of the coefficients. After the training exercise, a mapping exercise is completed virtually instantly so that the large number of calculations which give an accurate answer would be preferred over a less accurate, but simpler network.

The literature (Smith (1993), Demuth and Beale (1994)) reports that radial basis networks can be prone to over training and this was observed in this study. Although the results are not shown in this paper, it was not uncommon for training exercises with a radial basis network to result in relatively low SSE values, but perform miserably with the validation sets. These problems could be overcome with several procedures. It appears to be more important with the radial basis architecture to ensure (as much as possible) that the training set contained the extreme values for all input parameters. Ranging all input parameters between 0 and 1 helped as did the settings of some network parameters . When these steps were taken, very satisfactory results were obtained with the radial basis method.

The ability of neural networks to handle multiple input and output demonstrates the advantages of the neural network. This feature allows the simultaneous prediction of the propensity to perforate and the metal dissolution rate from the same input data. The complexity of these predictions is exemplified by Solution D which had a low tendency to perforate and a high metal dissolution rate. The network is able to treat the same combination of input parameters to give both high and low outputs with this solution.

Second, the "model-less" feature of a neural network allows pH and chloride levels to be considered without specifying an exact relationship. Other components of a solution which are known to affect corrosion could also be added to the network.

The ability of a neural network to correlate several parameters with the output is an advantage with corrosion data which tends to have high relative variability. A correlation table shows that several factors with a correlation coefficient with the pitting tendency above 0.85. It could be argued that factors such as noise resistance alone would allow a good prediction. Through plots of the noise resistance against the pitting severity. It can be seen that the variance is large for some solutions and this variability leads to overlap in the noise resistance for beverages with different pitting tendencies. Relying on this one factor alone could cause some erroneous classifications. While a neural network which trains with and maps several factors introduces some redundancy. However, it is this very redundancy which guards against errors due to an outlier in a specific factor.

The same could be said for the metal dissolution rate. There is a correlation of 0.81 with pH. However, solutions E and F had identical pH values but vastly different metal dissolution rates. Better predictions are made with neural networks because a network can take into account the smaller, but real contributions of other factors and complex, non-linear relationships.

The application of neural networks to corrosion problems should continue. Network utility can be improved by both adding examples from greater variety of solutions and refining the parameters which are used for each example. It is probable

that some input parameters presently used are unnecessary and are merely burdening the network with additional calculations. High correlations between two factors would be an indication of unnecessary parameters. Conversely, a factor with a low correlation with the target would be of dubious value. There should be a preference for those parameters which are truly predictive rather than those that classify. That is, the preference should be to develop a network which bases its predictions on those parameters which contribute to the corrosion process. This would be in contrast to the parameters which categorize the solutions themselves. It is better to say that a given solution will be an aggressive corrodent because of its chemical properties than to say this new solution has similar properties as Solution B and therefore will act like Solution B.

Neural networks were developed in an attempt to mimic the processes of the brain. From a certain perspective, it can be stated that neural networks evolved into sophisticated non-linear regression analysis. This study has been an example of that. However, it would be useful to remember the original analogy with the brain for the long term goals of a project such as this. Just as we humans refine our knowledge and judgment with increased experience, these neural networks should be constantly improved with additional examples and better selection of parameters to increase the network's "experience." In that sense, the networks should be considered "living" tools.

7. Conclusions

This paper demonstrated that neural networks can be used to distinguish effectively between uniform and localized corrosion processes when applied to electrochemical noise data.

The inclusion of pH and chloride concentrations in the input data also show the advantage of the neural network. It is believed that the effects of both pH and chloride concentration on metal dissolution are non-linear. It is not known if there are interactions between the two. However, these relationships need not be stipulated with precise equations when using a neural network. This feature of the neural network opens many heretofore impossible applications to quantification.

No single network architecture is able to perform all tasks or be the most efficient means of solving specific tasks. It is advantageous to have more than one type of network architecture available. It was demonstrated on several occasions in this study that one architecture would successfully complete a training exercise when another would not. Even within the same architecture, different learning algorithms can have different results with regard to individual training exercises. While some architectures or learning algorithms tended to be more efficient than others, examples

would arise where these normally more efficient methods would fail. The importance of data transformation must not be underestimated. Scaling or normalizing the data made significant differences in the ability of networks to train successfully. Normalizing the data causes all variables to have the same relative range in their values. The magnitude of the input values influences the initial value of the error function.

A final conclusion is that there are no definitive guidelines with regard to the choice of network, initial weight values, or data preparation. Neural networks present the experimenter with far more choices than most regression or ANOVA techniques. Like the networks themselves, the experimenter will require training and experience to become proficient in their application.

REFERENCES

1. Barton, T.F, Tuck, D.I., and Wells, D.B. (1993). *Proceedings 1993 the First New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems,* Los Alamitos, CA: IEEE Comput. Soc. Press, p. 325.

2. Bendat, J.S. and Piersol, A.G. (1986). *Random Data, Analysis And Measurement Procedures,* New York: Wiley-Interscience.

3. Chen, C-T, Skerry, B.S. (1991). *Corrosion,* 47, 8, p. 598.

4. Chester, M. (1993). *Neural Networks, A Tutorial,* Englewood Cliffs, NJ: Prentice Hall.

5. Demuth, H. and Beale, M. (1994). *Neural Network Toolbox User's Guide,* Natick MA: The Math Works.

6. Gabrielli, C., Huet, F., Keddam, M. and Oltra, R. (1990). Localized Corrosion as a Stochastic Process: A Review, *in Advances in Localized Corrosion,* Houston, TX: NACE, p. 93.

7. Henriksen, N., and Kristgeirson, J. (1994). *Proceedings of The 12th International Corrosion Congress,* Palo Alto, CA: Electric Power Research Institute, p. 3.1.

8. Kearns, J.R., Scully, J.R., Roberge, P.R., Reichert, D.L., and Dawson, J.L.(1996). Eds., *Electrochemical Noise Measurements,* STP Roberge, P.R. (1994). Corrosion, 50,7, p. 502.

9. Sharland, S.M, Bishop, C.M., Balkwill, P.H., and Stewart, J. (1990). The Initiation of Localized Corrosion: A Process Governed by a Strange Attractor?, *in Advances in Localized Corrosion,* Houston, TX: NACE, p. 109.

10. Smith, M. (1993). *Neural Networks for Statistical Modeling.* New York: Van Nostrand Reinhold.

11. Uruchurtu, J.C., and Dawson, J.L. (1987), *Corrosion,* 47, p.19.

12. Werbos, P.J. (1994). *The Roots of Backpropagation: From Ordered Derivatives to Neural Networks And Political Forecasting.* New York: Wiley- Interscience.