COBEM 99
XV Congresso Brasileiro de Engenharia Mecânica
15th Brazilian Congress of Mechanical Engineering
22 - 26 de Novembro de 1999 / November 22 - 26, 1999 Águas de Lindóia, São Paulo,

AN INVERSE INITIAL CONDITION PROBLEM IN HEAT CONDUCTION: A NEURAL NETWORK APPROACH

Fabio Tokio Mikki
Edison Issamoto
Jefferson I. da Luz
Pedro Paulo Balbi de Oliveira
Universidade do Vale do Paraíba – IP&D-UNIVAP
Av. Shishima Hifumi, 2911, Urbanova, São José dos Campos, SP, Brasil, 12244-000
pedrob@univap.br
Haroldo F. Campos-Velho
Jose Demisio Simoes da Silva
Instituto Nacional de Pesquisas Espaciais – LAC-INPE
Av. dos Astronautas, 1758, Jardim da Granja, São José dos Campos, SP, Brasil, 12227-970.
{demisio, haroldo}@lac.inpe.br

Abstract: *We determine the initial temperature profile on a slab with adiabatic boundary condition, from a transient temperature distribution, obtained at a given time. This is an ill-posed 1D parabolic inverse problem, where the initial condition has to be estimated. Two different artificial neural networks have been applied to address the problem: backpropagation and radial basis functions (RBF). Both approaches use the whole temperature history mapping. In our simulations, RBF presented better solutions, faster training, but higher noise sensitiveness, as compared to backpropagation.*

Key words: *Inverse Problems, Neural Networks, Backpropagation, Radial Basis Functions.*

1. INTRODUCTION

A neural network based approach is presented to the problem of determining the initial temperature profile on a slab with adiabatic boundary condition, from a transient temperature distribution, obtained at a given time. This is an ill-posed inverse problem, where the initial condition has to be estimated (Muniz et. al., 1999). Two different artificial neural network (ANN) models have been applied: a feedforward network with backpropagation, and a radial basis function (RBF) network. Both models differ in topology and use different training strategies; our simulations have shown that both are effective for solving that inverse problem, under the approach of using the whole temperature history mapping for training purposes. Comparisons of simulations with both models have shown a better performance of the RBF network with Gaussian functions, because of faster training and better solutions to the problem.

In Section 2, the inverse problem at issue is presented. Section 3 brings a briefing of artificial neural networks; Sections 4 and 5 show the implementation details and some results.

2. THE INVERSE PROBLEM

The forward problem under consideration consists of a transient heat conduction problem in a slab with adiabatic boundary condition, initially at a temperature profile denoted by *f*(*x*). The mathematical formulation of the problem is given by the following heat equation

$$\frac{\partial^2 T(x,t)}{\partial x^2} = \frac{\partial T(x,t)}{\partial t}, \text{ for } (x,t) \in \Omega \times \mathrm{R}^+, \qquad \frac{\partial T(x,t)}{\partial x} = 0,\ (x,t) \in \partial\Omega \times \mathrm{R}^+, \tag{1}$$

$$T(x,0) = f(x)\ ,\ \ (x,t) \in \Omega \times \{0\}$$

where *x* represents space (the distance between a point in the slab and one of its endpoints), *t* is the time, *f*(*x*) is the initial condition, *T*(*x*,*t*) represents the temporal evolution of the temperature at each point of the slab, and $\partial\Omega$ represents the boundaries of domain Ω. All of these terms are dimensionless quantities and $\Omega = (0,1)$.

The solution to the inverse problem for a given initial condition *f*(*x*) is explicitly obtained using separation of variables [4], for $(x,t) \in \Omega \times \mathrm{R}^+$:

$$T(x,t) = \sum_{m=0}^{+\infty} e^{-\beta_m^2 t} \frac{1}{N(\beta_m)} X(\beta_m,x) \int_{x'0}^{1} X(\beta_m,x') f(x') dx' \tag{2}$$

where $X(\beta_m,x) = \cos(\beta_m x)$ are the *eigenfunctions* associated to the problem, $\beta_m = m\pi$ are the *eigenvalues* and $N(\beta_m)=\int_\Omega X(\beta_m,x') f(x') dx'$ represents the *integral normalization* (or the *norm*).

3. BACKPROPAGATION AND RBF NETWORKS

Neural networks have emerged from an obscure field that had been discredited by perceived inadequacies into one of the fastest growing technologies in information processing (Tsoukalas e Uhrig, 1997; Haykin, 1996). Much research has being done in pursuing new neural network models and adapting the existing ones to solve real-world problems, such as those in engineering (Tsoukalas e Uhrig, 1997; Haykin, 1996).

An ANN is an arrangement of processing elements each one called a *neuron*. The first kind of neuron was proposed by McCulloch and Pitts in 1943 (Haykin, 1996), as a computational model of a biological neuron. ANN arrangements are characterized by:

- A large number of very simple neuron-like processing elements.
- A large number of weighted connections between the elements (where knowledge of a network is stored)
- Highly parallel, distributed control.
- An emphasis on automatic learning of internal representations.

The idea of an ANN is to explore the massively parallel network of simple elements that are able to yield a result very fast and, at the same time, display insensitivity to loss and failure of some of the component elements of the network (Nadler and Smith, 1993). These properties make artificial neural networks appropriate for application in many areas, such as pattern recognition, signal processing, image processing, financing, computer vision, engineering, etc. (Haykin, 1996; Tsoukalas and Uhrig, 1997, Haykin, 1994; Lin and Lee, 1996; Nadler and Smith, 1993).

The processing element in an ANN is a linear combiner with multiple weighted inputs, followed by an activation function. The simplest ANN model is the single-layer Perceptron that has a hard limiter activation function, but is only appropriate for solving linear problems. This fact made neural networks undergo obscurity in the 1970s (Haykin, 1996). In the 1980s they reemerged due to Hopfield´s paper on recurrent networks and the publication of the two volumes on parallel distributed processing (PDP) by Rumelhart and McClelland (Haykin, 1996).

There are several different architectures of ANNs, most of which directly depend on the learning strategy adopted. It is not the aim of the paper to go on a detailed background on ANNs. Instead, we will concentrate on a brief description of the two ANNs used in our simulations: the multilayer Perceptron with backpropagation learning, and radial basis functions (RBF). Good introductions on ANNs can be found in Haykin (1994), and Tsoukalas and Uhrig (1997).

The multilayer Perceptron with backpropagation learning algorithm, also referred to as the backpropagation neural network is a feedforward network composed of an input layer, an output layer, and a number of hidden layers for extracting high order statistics from the input data (Haykin, 1994).

Figure 2 shows a backpropagation neural network with one hidden layer. Functions *g* and *f* provide the activation for the hidden layer and the output layer, respectively. In order to introduce more flexibility to the network to solve nonlinear problems, the activation functions for the hidden layer are sigmoid functions (Figure 1). An alternative function for the sigmoid function in Figure 1 is the hyperbolic function. They differ in that the hyperbolic function varies between –1 to 1.

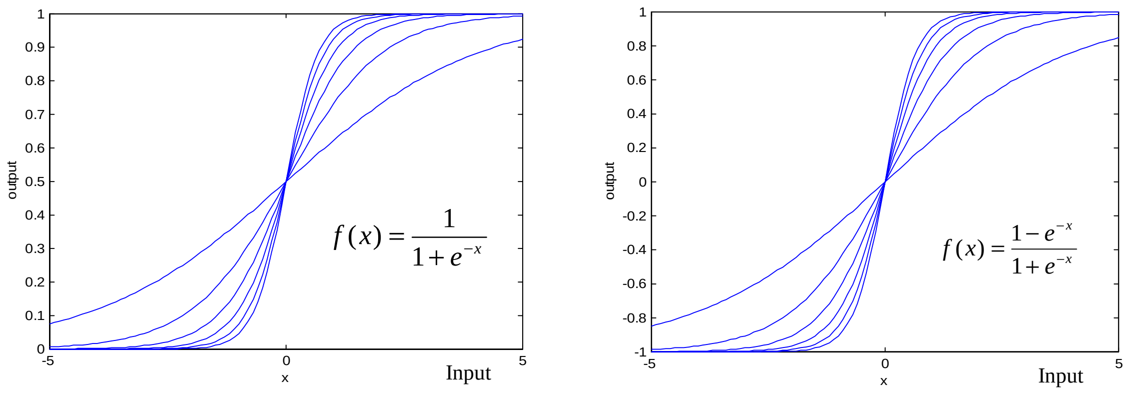


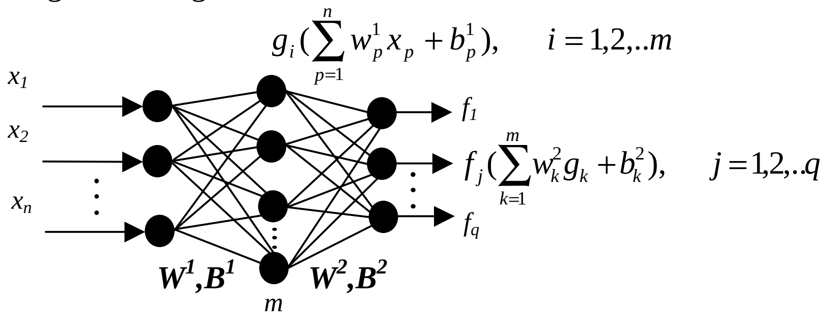
Figure 1 – Sigmoid and hyperbolic tangent activation functions.

Mathematically, a feedforward network simply maps input vectors of real values onto output vector of real values. The connections (Figure 2) have associated weights that are adjusted during the learning process, thus changing the performance of the network. There are two distinct phases in the usage of an ANN: the training phase (learning process) and the run phase (activation of the network). In the training phase, the weights are adjusted for the best performance of the network in establishing the mapping of many input/output vector pairs. Once trained, the weights are fixed and the network can be presented to new inputs for which it calculates the corresponding outputs, based on what it has learned.

The training phase of a backpropagation network is controlled by a supervised learning algorithm, which differs from unsupervised learning. The main difference is that the latter uses only information contained in the input data, whereas the former requires both input and output (desired) data, which permits the calculation of the error of the network as the difference between the calculated output and the desired vector. The network's weights adjustment is conducted by backpropagating such error to the network. The weight change rule is a development of the Perceptron learning rule. Weights are changed by an amount proportional to the error at that unit, times the output of the unit feeding into the weight. Equation 3 shows the general weight correction according to the so-called the delta rule.

$$\Delta w_{ji} = \eta \delta_j y_i \tag{3}$$

where, δ_j is the local gradient, y_i is the input signal of neuron j, and η is the learning rate parameter that controls the strength of change.



$g_i(\sum_{p=1}^{n} w_p^1 x_p + b_p^1), \quad i = 1,2,..m$
x_1
x_2
x_n
f_1
$f_j(\sum_{k=1}^{m} w_k^2 g_k + b_k^2), \quad j = 1,2,..q$
f_q
W^1,B^1
W^2,B^2
m

Figure 2 – The backpropagation neural network with one hidden layer.

Radial basis function networks are also feedforward networks with only one hidden layer. They were developed for data interpolation in multidimensional space (Mulgrew, 1996). Like the backpropagation network, RBF nets can also learn arbitrary mappings. The primary difference between a Backpropagation with one hidden layer and an RBF network is in the hidden layer. RBF hidden layer units have a receptive field, which has a center, that is, a particular input value at which they have a maximal output. Their output tails off as the input moves away from this point. Generally, the hidden unit function in an RBF network is a Gaussian (Figure 3). Figure 4 shows an RBF network.

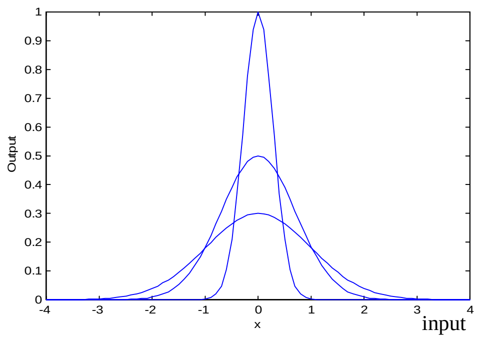


Figure 3 – Gaussians with three different standard deviations.

RBF networks are trained by deciding on how many hidden units there should be, the centers and the sharpness (standard deviation) of their Gaussians, and training up the output layer. Generally, the centers and standard deviations are decided on first by examining the vectors in the training data. The output layer weights are then trained using the Delta rule.

RBF networks can be trained: 1) on classification data (each output representing one class), and then used directly as classifiers of new data; and 2) on pair of points (x,f(x)) of an unknown function f, and then used to interpolate. They have the advantage that one can add extra units with centers near elements of the set of input data, which are difficult to classify. Like Backpropagation networks, RBF networks can be used for processing time-varying data and many other applications (Mulgrew, 1996).

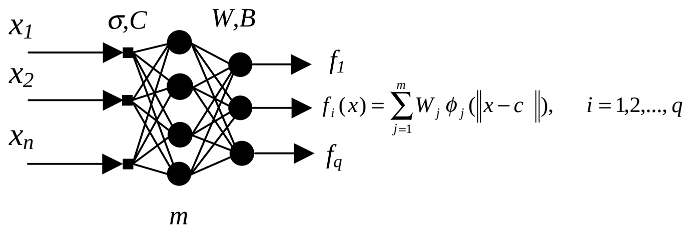

x_1
x_2
x_n
σ,C
W,B
m
f_1
f_q
$f_i(x) = \sum_{j=1}^{m} W_j \phi_j(\|x - c\|), \quad i = 1,2,\ldots,q$

Figure 4 – Radial Basis Functions network.

4. IMPLEMENTATION

In this paper, the inverse solution to the problem presented in Section 2 is obtained using a neural network approach, as shown in Figure 5, where variable t^* represents the estimated time of the measured distribution $T^{\text{measured}}(x)$ presented to ANN System 1, and $f^e(x)$ is the initial temperature distribution over the slab.

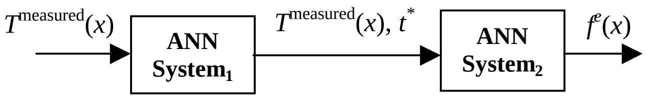


Figure 5 – Block Diagram of the ANN approach to the inverse problem.

The neural network approach shown in Figure 5 is based on the Whole History Mapping (WHM) presented in Krejsa et al. (1996). The WHM relies on the idea of an ANN mapping the whole vector of observed values of the temperature history to the corresponding vector of outputs (the heat transfer coefficients in Krejsa et al. (1996)). Thus, after training there is an exact correspondence between the observed temperature values and the output, over some time interval.

The WHM is stable and insensitive to noise in the data (Krejsa et al., 1996). But there are two critical disadvantages: 1) the number of input vectors can be quite big, leading to a large number of connections and very slow training; and 2) for every case a new network has to be constructed, thus requiring new training sets, and the whole training process.

In our simulations, three temperature distributions were obtained using the direct model of Muniz et al. (1999) over a slab with adiabatic conditions for 50 time instants. The individual temperature distributions were calculated by applying the direct model in equation (2), for 3 distinct initial distributions $f(x)$: triangular shape, logarithmic, and sinusoidal. Such distributions were all combined in a single distribution as in Figure 6. Two separated simulations were conducted using an arrangement of two neural networks (as depicted in Figure 5): two backpropagation networks and two RBF networks. The aim was to find out the differences between the two models in estimating the initial temperature distribution. Figure 8 shows the general architecture of both arrangements. Network 1 is used for estimating the time t^* of occurrence of the measured temperature distribution (only 1 output neuron is required, see Figure 8). Network 2 is used for the iterative estimation of the initial temperature distribution over the slab (n output neurons representing different positions over the slab). The network architectures used in both simulations had the following features:

Simulation with two backpropagation networks:
- 1 input layer / 1 hidden layer / 1 output layer
- Hyperbolic Tangent hidden neurons
- Linear output neurons

Simulation with two RBF networks:
- 1 input layer / 1 hidden layer / 1 output layer
- Gaussian Radial basis neurons
- Linear output neurons

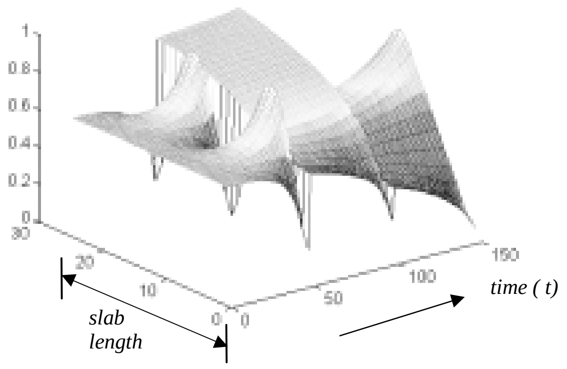


Figure 6 – Combination of three temperature distributions over 25 positions for 50 time instants each.

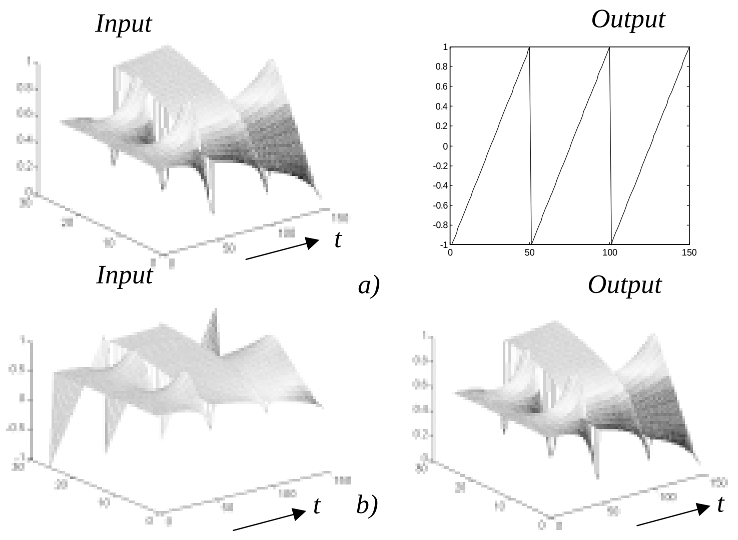


Figure 7 – Training used for neural network learning. a) Input (left) and Output (right) data sets for network 1; b) Input (left) and Output (right) data sets for network 2.

The use of two separate networks requires two training sets derived from the combined data in Figure 6. One is used for training network 1 to estimate the time instant at which a certain temperature distribution occurred, and the other one is used for training network 2 to calculate the initial temperature distribution over the slab (Figure 7). Network 1 input data was the temperature distribution history and target data was the time instants corresponding to each temperature distribution over the slab (Figure 7-a). Network 2 input data was the temperature distribution together with the time at which it occurred and target data was composed of temperature distributions one time instant ahead of each temperature distribution in the input data (Figure 7-b).

In both simulations, training network 2 consisted of presenting a certain temperature distribution together with its corresponding time *t* at the input, and the subsequent distribution at time t+1 as the output. Then, network 2 learned how to predict the distribution at a time instant ahead. Training was performed separately for each network and each arrangement. Table 1 shows the parameters used for training the networks. Both networks were trained up to the stopping criteria: network 1 achieved the target error, whereas network 2 was trained for the maximum number of epochs. Simulations were carried out under the Matlab neural network toolbox.

Table 1 – Training results for the backpropagation neural networks.

Network	**# of neurons**	**Target error**	**# of epochs**
1	**50**	**0.001**	**20000**
2	**50**	**0.0001**	**20000**

Training the RBF networks used similar parameters as shown in Table 2. The algorithm for training the RBF networks gradually builds up the final network, thus not requiring the specification of the number of neurons. In both simulations, the training parameters were chosen testing different possibilities and checking out the networks' performances.

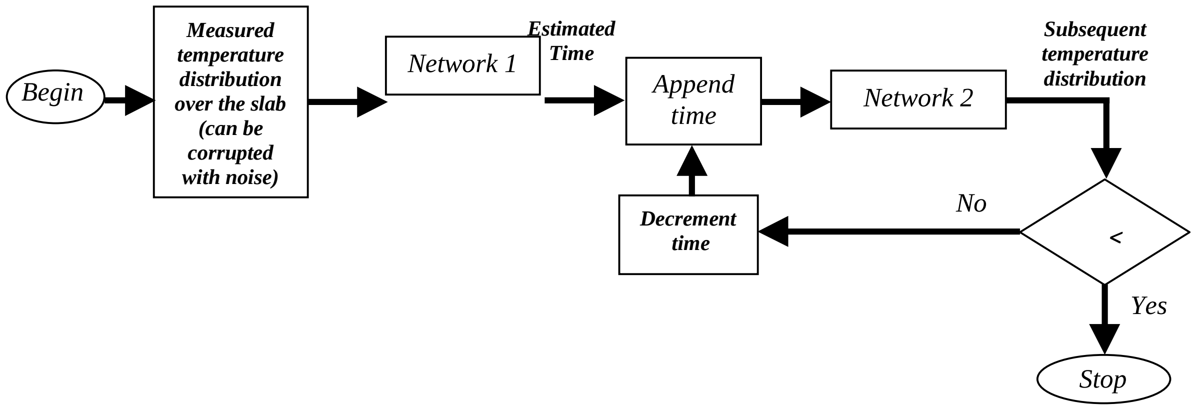

Begin
Measured temperature distribution over the slab (can be corrupted with noise)
Network 1
Estimated Time
Append time
Network 2
Subsequent temperature distribution
<
No
Decrement time
Yes
Stop

Figure 8 – Flowchart of the activation scheme for applying neural networks to solve the inverse problem described in Muniz et al. (1999).

Table 2 – Training results for the RBF networks.

Network	**RBF function**	**# of neurons**	**Spread constant**	**Target error**
1	**Gaussian**	**34**	**0.1**	**5×10^{-3}**
2	**Gaussian**	**59**	**0.3**	**10^{-5}**

The performances were tested through the activation of both arrangements by presenting a certain temperature distribution over the slab to network 1, chosen from one of the distributions in Figure 6. Network 1 estimated the time of the given distribution, which was fed into network 2, together with the given distribution, for iterative estimation of the initial distribution.

5. RESULTS

Some of the results of the activation of the networks trained with the combined distribution in Figure 6 can be seen in Figure 9. The networks were presented to a certain profile chosen from one of the individual distributions that make up the combined distribution. Four profiles were supplied to the networks, chosen at four different positions corresponding to $^1/_2$, $^1/_5$, $^1/_{10}$, and $^1/_{25}$ of the total time for steady state.

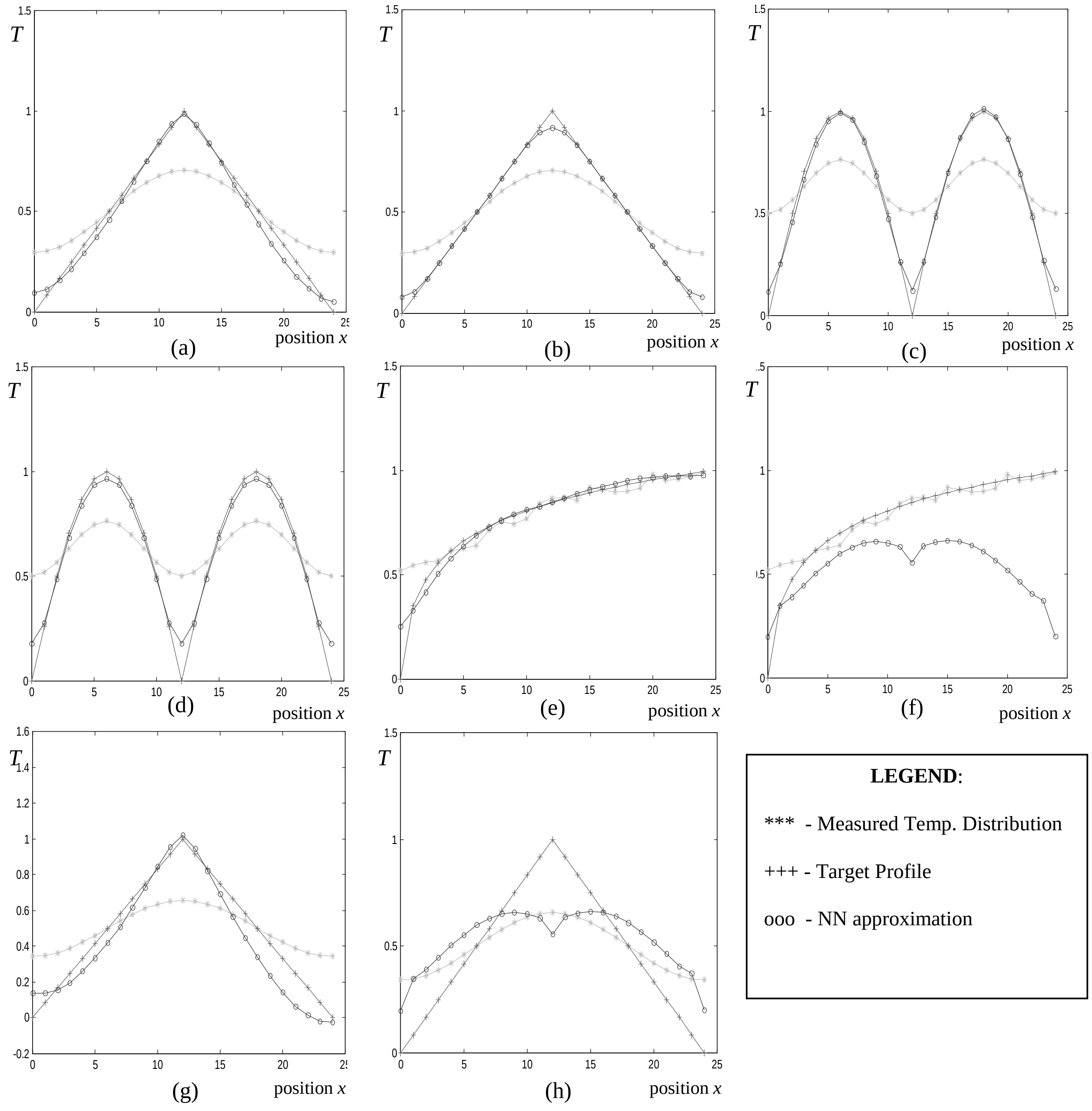


Figure 9 – Results of activation of both network arrangements. *T* is the temperature and *x* is the position on the slab.

Figure 9 displays some of the results obtained by activating the backpropagation and RBF networks with the same measured temperature distribution. The curves with (*) represents the measured temperature distribution supplied to the networks. The curves with (+) represent the target profile, that is, the desired initial temperature distribution. The curves with (o) represent the ANN approximation. Figures 9-a and 9-b show the backpropagation and RBF outputs for a triangular distribution chosen at ½ of the time of steady state. Comparison shows that the backpropagation network approximated better to the desired profile. Figures 9-c and 9-d show the results of a temperature distribution measured at $^1/_5$ of steady state time. Figures 9-e and 9-f show a logarithmic distribution chosen at $^1/_{25}$ of steady state time added with 5% noise. The perturbation was constructed with a vector of uniform random numbers weighed by the required (5%). Figures 9-g and 9-h show the results for the backpropagation and RBF networks for a distribution that was not used for training. Visual comparison shows the backpropagation networks generalized better than RBF networks. Table 3 summarizes the activation results for different initial profiles and different time instants, some of them perturbed with noise.

Table 3 – Summary of activation results

Distribution	**Time**	**Error Backpropagation**	**Error RBF**	**Noise**
Triangular	**½**	**0.001932**	**0.000912**	**0%**
Triangular	**$^1/_5$**	**0.000573**	**0.000425**	**0%**
Triangular	**$^1/_{25}$**	**0.000706**	**0.000408**	**0%**
Sinusoidal	**$^1/_5$**	**0.002148**	**0.004418**	**0%**
Logarithmic	**$^1/_{25}$**	**0.003241**	**0.001665**	**0%**
Triangular	**$^1/_5$**	**0.001311**	**0.000465**	**5%**
Logarithmic	**½**	**0.002974**	**0.102762**	**5%**
Triangular	**$^3/_{10}$**	**0.009997**	**0.039955**	**0%**

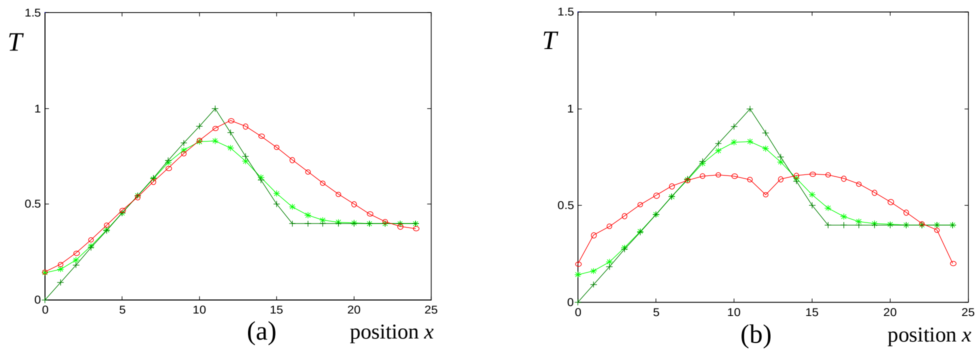


Figure 10 – Activation of both networks with a semi triangular distribution not used in the training phase. In (a) the backpropagation output; (b) the RBF output.

Figure 10 shows the results of activating both networks with a semi-triangular measured distribution at $^1/_{10}$ of the time for steady state. It should be observed in figure that the backpropagation networks approximated better to the desired distribution, whereas the RBF networks resulted in a very different curve. Some other tests were conducted in which the RBF results were similar to that in Figure 10-b. On the other hand, the backpropagation approximated even better if it were provided with a distribution at a very short time after the initial profile. The better performance of the backpropagation networks is due to its better generalization capacity.

6. CONCLUSIONS

A neural network approach for solving an inverse initial condition problem in heat conduction has been derived. Two ANN models, backpropagation and RBF networks, were tried. Training sets were constructed with the direct model in equation 2. Two neural systems (Figure 1) were used, composed by two backpropagation or two RBF networks.

Backpropagation networks interpolate better with unknown data (see Figures 9-g and 10-a). They are more robust to perturbations in the data (Figure 9-e), whereas RBF networks train faster, do not require architecture specification beforehand, but are very sensitive to noise (Figure 9-h) and do not perform well for unknown distributions (Figure 10-b).

The conducted simulations show the effectiveness of using neural networks for solving inverse problems, as it had been shown in Kresja (1996), thus deriving promising research possibilities. The results in Figures 9 and 10, and Table 3, were obtained in the experiments of an ongoing research on using neural networks for solving inverse problems. Further work will continue in trying to establish numerical comparisons with those of Muniz (1999). Also, different ANN architectures and more diverse training sets (temperature distributions with complex initial distributions) will be tried, in search for other architectures that can provide similar or better results them those obtained with the backpropagation networks so far, and enhance the generalization capacity of the networks. The general ANN approach presented will also be tried to solve inverse problems in other application areas such as geophysics, image processing and computer vision.

ACKNOWLEDGMENTS

This work has been funded by Fundação de Amparo à Pesquisa do Estado de São Paulo - FAPESP, (Thematic Project process 96/07200-8), and the British Council (Ref. SPA/881/126).

REFERENCES

Chen, S., Cowan, C.F.N., and Grant, P.M., 1991, Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks. IEEE Transactions on Neural Networks. March, 2(2), pp. 302-309.

Haykin, S., 1994, Neural Networks: A Comprehensive Foundation. Macmillan. New York.

Haykin, S., 1996. Neural Networks Expand SP´s Horizons. IEEE Signal Processing, March, pp. 24-49.

Krejsa, J., Woodbury, K.A., and Raudensky, M., 1996, Assessment of Strategies and Potential for Neural Networks in the IHCP.

Mulgrew, B., 1996, Applying Radial Basis Functions. IEEE Signal Processing Magazine, pp. 50-65.

Lin, C-T. and Lee, G., 1996, Neural Fuzzy Systems: A Neuro-Fuzzy synergism to Intelligent Systems. Prentice Hall, New Jersey.

Muniz, W.B., Campos Velho, H.F., and Ramos, F.M., 1999, A Comparison of Some Inverse Methods for Estimating the Initial Condition of the Heat Equation. Journal of Computational and Applied Mathematics, 101, pp. 153-171.

Musavi, M.T., Ahmed, W., Chan, K.H., Faris, J.B., and Hummels, D.M., 1992, On the Training of Radial Basis Function Classifiers. Neural Networks, 5, pp. 595-603.

Nadler, M. and Smith, E.P., 1993, Pattern Recognition Engineering. John Wiley, New York.

Tsoukalas, L.H. and Uhrig, R.E., 1997, Fuzzy and Neural Approaches in Engineering. John Wiley, New York.

Widrow, B. and Winter, R., 1988, Neural Nets for Adaptive Filtering and Adaptive Pattern Recognition. Computer, March, 21(3), pp. 25-39.