4th International Conference on Inverse Problems in Engineering
Rio de Janeiro, Brazil, 2002

Estimation of Initial Condition in Heat Conduction by Neural Network

Élcio H. Shiguemori **José Demísio S. da Silva** **Haroldo F. de Campos Velho**

Laboratory for Computing and Applied Mathematics – LAC
National Institute for Space Research – INPE
São José dos Campos, SP, Brazil
[elcio, haroldo, demisio]@lac.inpe.br

ABSTRACT

This paper describes a methodology for using neural networks in an inverse heat conduction problem. Three neural network (NN) models are used to determine the initial temperature profile on a slab with adiabatic boundary condition, given a transient temperature distribution at a given time. This is an ill-posed 1D parabolic inverse problem, where the initial condition has to be estimated. Three neural network models addressed the problem: a feedforward network with backpropagation, radial basis functions (RBF), and cascade correlation. The input for the NN is the temperature profile obtained from a set of probes equally spaced in the one-dimensional domain. The NNs were trainned considering a 5% of noise in the experimental data. The training was performed considering 500 similar test-functions and 500 different test-functions. Good reconstructions have been obtained with the proposed methodology.

NOMENCLATURE

ASE	Average square error
b_k	Bias employed in the NNs
CasCor	Cascate correlation NN
$f(x)$	Unknown nitial condition
$g(x)$	Activation function
w_{ji}	Conection weight of a NN
$N(b_m)$	Norm of the eigenfunction
NN	Neural network
RBF	Radial base function NN
$T(x,t)$	Temperature calculate
$\tilde{T}(x,t)$	Experimental temperature
$X(b_m x)$	eigenfunction
a	Regularization parameter
b_m	Eigenvalue in Eq. (2)
h	Learning rate
m	Random variable
s	Standard deviation
Ω	Space domain
$\Re^+$	Positive real number set

INTRODUCTION

Neural networks have emerged as a new technique to solve inverse problems. This approch was used to identify initial conditions in inverse heat conduction problem on a slab with adiabatic boundary conditions, from transient temperature distribution, obtained at a given time. Three neural networks architectures have been proposed to address the problem: the multilayer perceptron with backpropagation, radial basis functions (RBF), both trained with the whole temperature history mapping, and cascade correlation.

The results are compared with those obtained with non-linear least square approach and standard regularization schemes [1, 2].

Preliminary results using backporpagation and radial basis function neural networks were obtained using whole time history, but with only three different test functions for the learning process [3, 4]. The reconstructions obtained were worse than those identified with regularization techniques. In that strategy two NNs were coupled: the first NN was used for determining the time-period to get the observational data, and another one to find the initial condition itself. That strategy constituted in a novelty in the field, but propably the poor set of test functions for learning step did not permit a good reconstruction. In order to overcome this constrain, 500 functions were used for the learning process in this work. In addition, two groups of test functions were used. In the first group 500 completely different test functions were used, while for the second group 500 similar test-functions were used.

Numerical experiments were carried out with synthetic data with 5% of noise used to simulate experimental data.

DIRECT HEAT TRANSFER PROBLEM

The direct problem under consideration consists of a transient heat conduction problem in a slab with adiabatic boundary condition, with an initial temperature profile denoted by $f(x)$. Mathematically, the problem can be modeled by the following heat equation

$$\begin{aligned} &\frac{\partial T(x,t)}{\partial t} = \frac{\partial^2 T(x,t)}{\partial x^2} && (x,t) \in \Omega \times \Re^+ \\ &\frac{\partial T(x,t)}{\partial x} = 0 && (x,t) \in \partial\Omega \times \Re^+ \\ &T(x,0) = f(x) && (x,t) \in \Omega \times \{0\} \end{aligned} \qquad (1)$$

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 $\Omega = (0,1)$ is the 1D space domain.

The direct problem solution, for a given initial condition $f(x)$ is explicitly obtained using separation of variables, for $(x,t) \in \Omega \times R^+$:

$$T(x,t) = \sum_{m=0}^{+\infty} e^{-b_m^2 t} \frac{1}{N(b_m)} X(b_m, x) \int_0^1 X(b_m, x') f(x') dx' \qquad (2)$$

where $X(b_m,x) = \cos(b_m x)$ are the *eigenfunctions* associated to the problem, $b_m = m\pi$ are the *eigenvalues* and $N(b_m) = \int_\Omega X(b_m, x') f(x') dx'$ represents the *integral normalization* (or the *norm*) [5].

The inverse problem consists in estimating the initial temperature profile $f(x)$ for a given transient temperature distribution $T(x,t)$ at a time t [1].

NEURAL NETWORK ARCHITECTURES

Artificial neural networks (ANN) are made of arrangements of processing elements (*neurons*). The artificial neuron model basically consists of a linear combiner followed by an activation function. Arrangements of such units form the ANNs that are characterized by:

1. Very simple neuron-like processing elements;
2. Weighted connections between the processing elements (where knowledge is stored);
3. Highly parallel processing and distributed control;
4. Automatic learning of internal representations.

ANNs aim to explore the massively parallel network of simple elements in order to yield a result in a very short time slice and, at the same time, with insensitivity to loss and failure of some of the elements of the network. These properties make artificial neural networks appropriate for application in pattern recognition, signal processing, image processing, financing, computer vision, engineering, etc. [6-9].

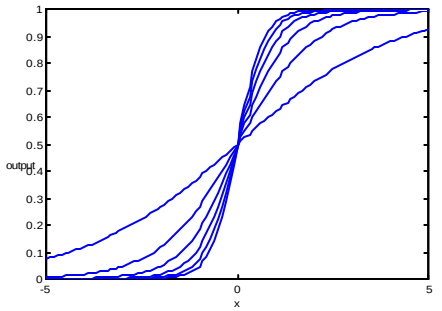
The simplest ANN model is the single-layer Perceptron with a hard limiter activation function, which is appropriate for solving linear problems. This fact prevented neural networks of being massively used in the 1970s [6]. In the 1980s they reemerged due to Hopfield´s paper on recurrent networks and the publication of the two volumes on parallel distributed processing (PDP) by Rumelhart and McClelland [6].

There exist different ANN architectures that are dependent upon the learning strategy adopted. This paper briefly describes the three ANNs used in our simulations: the multilayer Perceptron with backpropagation learning, radial basis functions (RBF), and cacade correlation. Detailed introduction on ANNs can be found in [6] and [9].

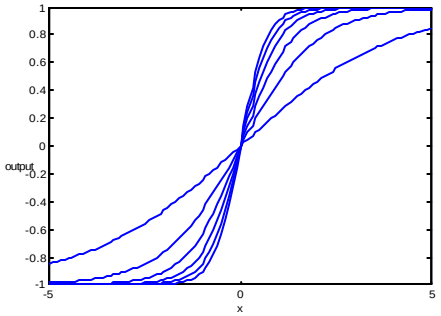
Multilayer perceptrons with backpropagation learning algorithm, commonly referred to as backpropagation neural networks are feedforward networks composed of an input layer, an output layer, and a number of hidden layers, whose aim is to extract high order statistics from the input data [4]. Figure 2 depicts a backpropagation neural network with a hidden layer. Functions g and f provide the activation for the hidden layer and the output layer neurons, respectively. Neural networks will solve nonlinear problems, if nonlinear activation functions are used for the hidden and/or the output layers. Figure 1 shows examples of such functions.

A feedforward network can input vectors of real values onto output vector of real values. The connections among the several neurons (Figure 2) have associated weights that are adjusted during the learning process, thus changing the performance of the network. Two distinct phases can be devised while using an ANN: the training phase (learning process) and the run phase (activation of the network). The training phase consists of adjusting the weights 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.



(a)



(b)

Figure 1: Two activation functions: (a) sigmoid $g(x) = (1+e^{-x})^{-1}$; (b) $g(x) = \tanh(x) = (1-e^{-x})(1+e^{-x})^{-1}$.

The backpropagation training is a supervised learning algorithm that requires both input and output (desired) data. Such pairs permit the calculation of the error of the network as the difference between the calculated output and the desired vector. The weight adjustments are conducted by backpropagating such error to the network, governed by a change rule. The 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

$$\Delta w_{ji} = \eta d_j y_i \qquad (3)$$

where, d_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.

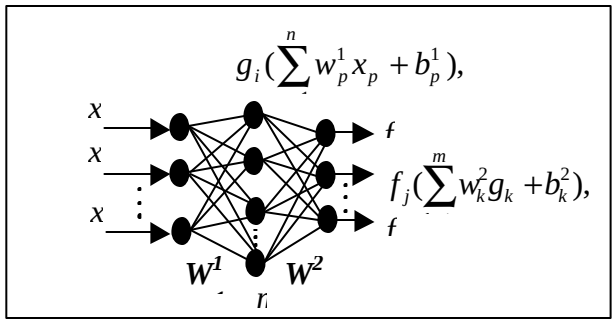


Figure2: The backpropagation neural network with one hidden layer.

Radial basis function networks are feedforward networks with only one hidden layer. They have been developed for data interpolation in multidimensional space. 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 units. 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. The most used function in an RBF network is a Gaussian (Figure 3).

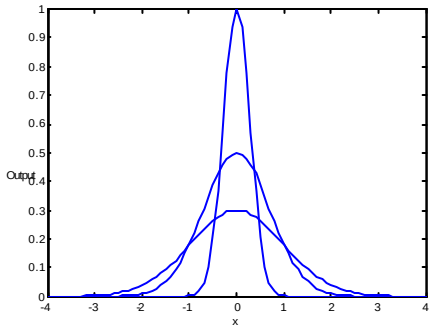


Figure 3: Gaussian for three differents variances.

RBF networks require the determination of the number of hidden units, the centers, and the sharpness (standard deviation) of their Gaussians. 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.

The training of RBF networks can be conducted: (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. The main advantage of RBF networks relies on the fact 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.

The third ANN used in the present paper is the cascade correlation. This NN permits to dynamically find out the appropriate number of neurons, begining with just the input and output layers, with all the neurons fully interconnected (there is no hidden layer). The weights on these connections are determined using a conventional learning. Next, new neurons are considered sequentially, and weights between the candidate units and the inputs are selected to maximize the correlation between the activation of the neuron(s) and the residual error of the net. Once a neuron is selected, its weights on the inputs are frozen, and are not subsequently changed when considering new neurons. Additional neurons are applied until a specified small error is reached.

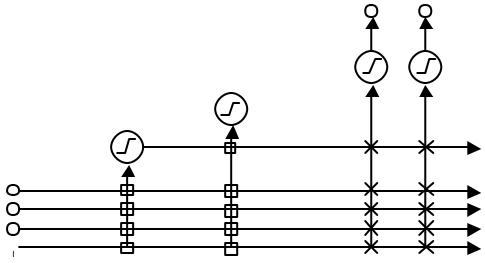


Figure 4: Casccade correlation network with 2 hidden layers. The symbol ⊘ denotes a neuron.

Figure 4 shows a cascate correlation (CasCor) network into which two candidate neurons have been implemented. These neurons use a conventional activation function, as shown in Figure 2. Each open box in the figure represents a weight that is trained only once (when the neuron is a candidate) and then is frozen. But the cross marks represent weights that are repeatedly changed as the network evolves. Note that the structure of the network is such that the inputs remain directly connected to the outputs, but also some information is filtered through the neurons. The direct input to output connection can handle the linear portion of the mapping, while the non-linearities are addressed by the neurons.

NEURAL NETWORK FOR DETERMINING THE INITIAL CONDITION

Artificial neural networks have two stages in their application, firstly the learning and activation steps. During the learning step, the corresponding weights and bias of each neuron are adjusted to some reference examples. For activation, the output is obtained based on the weights and bias computed in the learning phase. A supervised learning strategy was used for all NN architectures.

The numerical experiment for the inverse problem is based on two test functions, the triangular function

$$f(x)=\begin{cases}2x & x\in[0,0.5]\\ 2(1-x) & x\in(0.5,1]\end{cases} \tag{6}$$

and semi-triangular function

$$f(x)=\begin{cases}0.55 & 0\le x\le 0.2\\ 8/3\,x & 0.2<x<0.5\\ -28/5\,x+23/5 & 0.5<x<0.75\\ 2/9 & 0<x\le 1\end{cases}. \tag{7}$$

The experimental data (measured temperatures at a time $\tau>0$), which intrinsically contains errors in the real world, is obtained by adding a random perturbation to the exact solution of the direct problem, such that

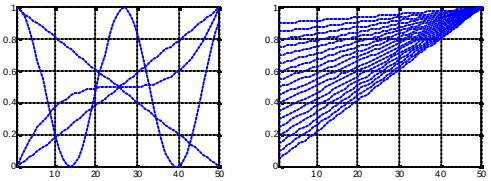
$$\tilde{T}=T_{\text{exact}}+s\,m \tag{8}$$

where s is the standard deviation of the errors and m is a random variable taken from a Gaussian distribution, with zero mean and unitary variance. Twin numerical experiments were performed. In the first one, noiseless *observational* data were employed (s=0). The second numerical experiment was carried out using 5% of noise (s=0.05).

For the NNs, the training sets are constituted by synthetic data obtained from the forward model, i.e., profile of a *measure* points from probes spread in the space domain. Two different data sets were used. The first data set is the profiles obtained from 500 similar functions (see examples in Figure 5a). The second one is that obtained with 500 non-similar functions (Figure 5b). Similar functions are those belonging to the same class (linear function class, trigonometric function class, such as sine functions with

different amplitude and/or phase, and so on). Non-similar functions are decorrelated, that is, each one belongs to a distinct class.

Figure 5 shows a set of functions used in the leaming stage, applying non-similar (Fig. 5a) and similar functions (Fig. 5b).



(a) (b)

Figure 5: Sample of test functions for training: (a) non-similar functions; (b) similar functions.

The *activaction* is a regular test used for checking out the NN performance, where a function belonging to the test function set is applied *to activate* (to run) the NN. Good activations were obtained for all three NNs for observational data with noise and noiseless data, for similar and non-similar test function sets (not shown). In the activation test the NN trained with similar data were systematicaly better than the training with non-similar functions (not shown either), with and without noise in the data. A summary of the training results for the three NNs is presented in Table 1.

Table 1: Training results for the neural networks used for initial condition reconstruction.

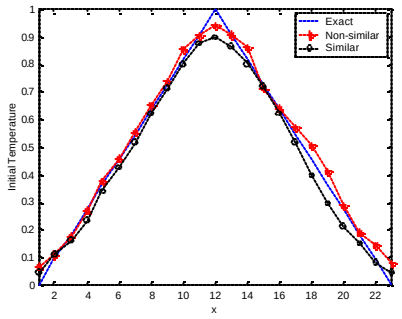
Multi-layer perceptron				
Data	Noise	Hidden Neurons	Training Epochs	ASE
Non-similar	0%	25	150000	0,0487
Similar	0%	20	50000	0,0127
Non-similar	5%	20	300000	0,0694
Similar	5%	20	50000	0,0144
Radial base function				
Data	Noise	Hidden Neurons	Training Epochs	ASE
Non-similar	0%	20	50000	0,0576
Similar	0%	20	50000	0,0095
Non-similar	5%	20	300000	0,0873
Similar	5%	20	50000	0,0123
Cascate correlation				
Data	Noise	Hidden Neurons	Training Epochs	ASE
Non-similar	0%	10	300000	0,0746
Similar	0%	05	63000	0,0230
Non-similar	5%	02	2000	0,1389
Similar	5%	05	63000	0,0318

Nevertheless, the activation test is an important procedure, indicating the permormance of a NN. The effective test is defined using a function (initial condition) that did not belong to the training function set. This action is called the *generalization* of the NN. Functions as expressed by Eqs. (6) and (7) did not belong to the function set in the traning step.

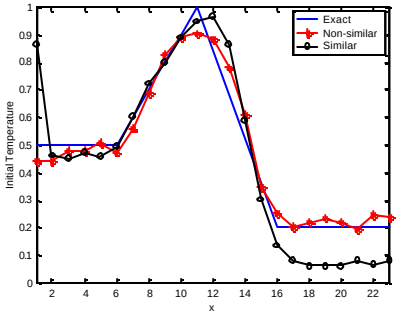
Figures 6, 7, and 8 show the initial condition reconstruction for noiseless experimental data, and Table 2 presents the Average Square Error (ASE) for three NNs used in this paper. Differently from the results for the activation test, reconstruction using non-similar functions were better than estimation with similar functions.

Table 2: Activation results for the noiseless experimental data.

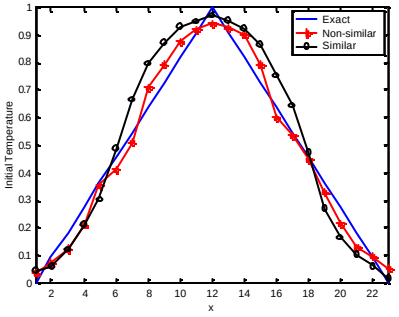
Multi-layer perceptron		
***f*(*x*)**	Data	ASE
Triangular	Non-similar	0.0136
Triangular	Similar	0.0139
Semi-triangular	Non-similar	0.0246
Semi-triangular	Similar	0.1599
Radial base function		
***f*(*x*)**	Data	ASE
Triangular	Non-similar	0.0065
Triangular	Similar	0.0079
Semi-triangular	Non-similar	0.0275
Semi-triangular	Similar	0.0498
Cascate correlation		
***f*(*x*)**	Data	ASE
Triangular	Non-similar	0.0253
Triangular	Similar	0.0845
Semi-triangular	Non-similar	0.0471
Semi-triangular	Similar	0.1462



(a)



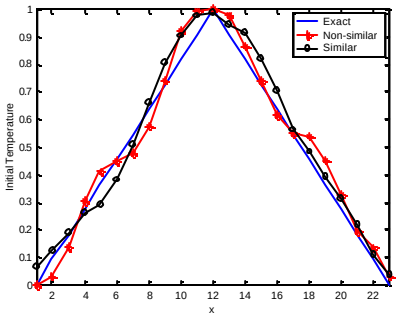
(b)



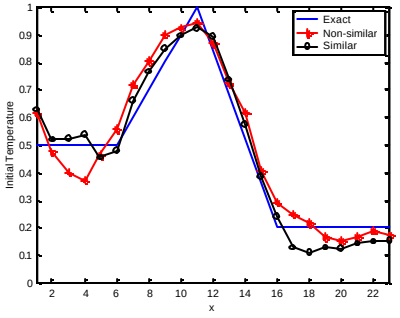
(a)

Figure 6: Reconstruction using multi-layer perceptron NN with noiseless data.

The worse reconstructions for noiseless data were obtained using CasCor-NN (see Table 2 and Figures 6, 7, and 8), and the best identifications were obtained using RBF-NN. However, good initial condition identifications were gotten with the three NN architectures.

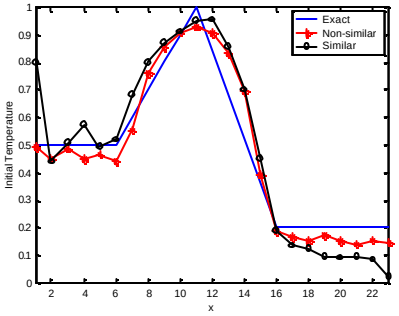


(a)



(b)

Figure 7: Reconstruction using radial base function NN with noiseless data.



(b)

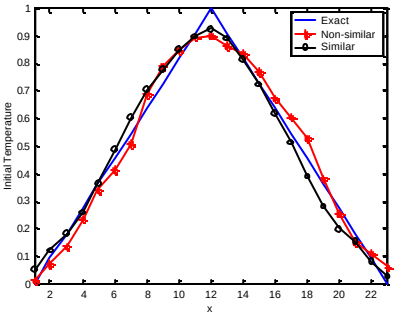
Figure 8: Reconstruction using cascate correlation NN with noiseless data.

Table 3: Activation results for the experimental data with 5% of noise.

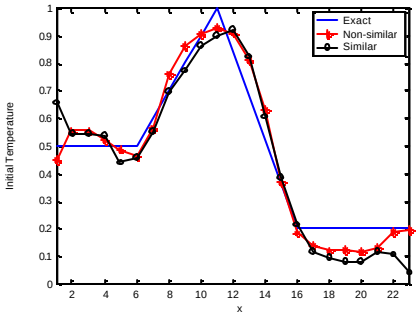
Multi-layer perceptron		
***f*(*x*)**	Data	ASE
Triangular	Non-similar	0.0227
Triangular	Similar	0.0210
Semi-triangular	Non-similar	0.0621
Semi-triangular	Similar	0.0786
Radial base function		
***f*(*x*)**	Data	ASE
Triangular	Non-similar	0.0308
Triangular	Similar	0.0331
Semi-triangular	Non-similar	0.0563
Semi-triangular	Similar	0.0396
Cascate correlation		
***f*(*x*)**	Data	ASE
Triangular	Non-similar	0.0384
Triangular	Similar	0.0947
Semi-triangular	Non-similar	0.0486
Semi-triangular	Similar	0.1294

Real tests for inverse problems must be performed using some level of noise in the synthetic experimental data. As mentioned previously, the real experimental data were simulated corrupting the output data from direct problem with Gaussian white noise, see Eq. (8).

As with our numerical experiment with noiseless data, the identification of the initial condition was effective for all NNs used here.

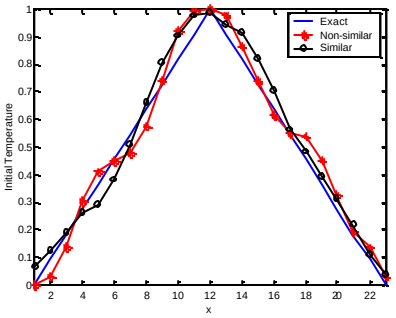


(a)

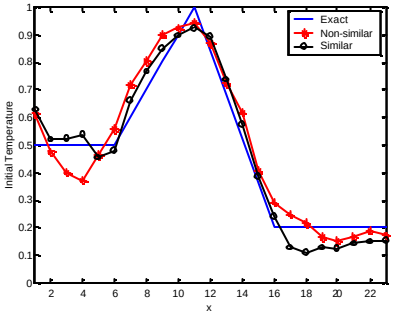


(b)

Figure 9: Reconstruction using multi-layer perceptron NN with 5% of noise.



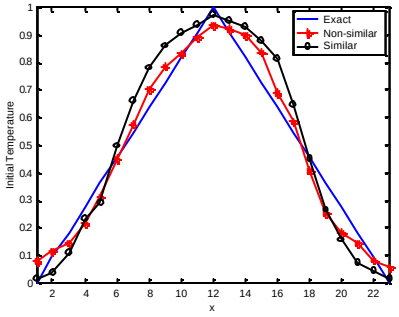
(a)



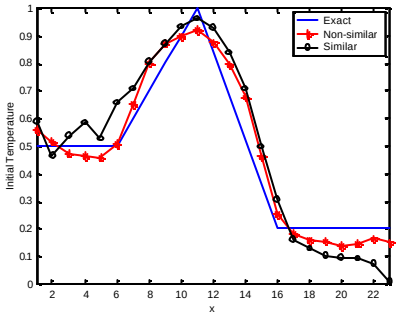
(b)

Figure 10: Reconstruction using radial base function NN with 5% of noise.

Figures 9, 10 and 11 show the reconstructions for multi-layer perceptron, RBF and CaCor NNs. Table 3 presents the ASE for two test function in the generalization. As expected, reconstructions with data contamined with noise was worse than those with noiseless data. But, the NNs were robust in the identification with noise in the experimental data.



(a)



(b)

Figure 11: Reconstruction using cascate correlation NN with 5% of noise.

FINAL REMARKS

Three architectures of neural networks were studied in the reconstruction of the initial condition of a heat conduction problem. All of NNs were effective for solving this inverse problem. Different from previous results [2, 3, 10], reconstructions are comparable with those obtained with regularization methods, even for data containing noise. However, the NNs do not remove the inherent ill-posedness of the inverse problem.

The initial condition estimation problem seems to be a harder inverse problem than the identification of boundary condition in heat transfer [11-13].

An interesting remark is the result for the activation test, where the training with similar functions produced better identification than non-similar function. However, reconstructions using non-similar functions were systematically better for the generalization, except in only one case: the estimation of semi-triangular function by RBF-NN with 5% of noise (Table 3).

The worse estimation was obtained with CasCor-NN. A future work could be done using the strategy adopted by Hidalgo and Gómez-Treviño [14]. To accommodate large amounts of noise, they added a regularization term to the least squares objective function of the neural network.

REFERENCES

1. W.B. Muniz, H.F. de Campos Velho and F.M. Ramos (1999): A comparison of some inverse methods for estimating the initial condition of the heat equation, *J. Comp. Appl. Math*., **103**, 145 (1999).

2. W.B. Muniz, F.M. Ramos and H.F. de Campos Velho, Entropy- and Tikhonov-based regularization techniques applied to the backwards heat equation, *Comp. Math. Appl*., **40**, 1071 (2000).

3. E. Issamoto, F.T. Miki, J.I. da Luz, J.D. da Silva, P.B. de Oliveira, H.F. de Campos Velho, An Inverse Initial Condition Problem in Heat Conductions: A Neural Network Approach, *Braz. Cong. Mech. Eng*. (COBEM), Proc. in CD-ROM - paper code AAAGHA, 238 (1999), Unicamp, Campinas (SP), Brasil.

4. F.T. Miki, E. Issamoto, J.I. da Luz, P.B. de Oliveira, H. F. de Campos Velho, J.D. da Silva, A Neural Network Approach in a Backward Heat Conduction Problem, *Braz. Conf. Neural Networks*, Proc. in CD-ROM - paper code 0008, 019 (1999), São José dos Campos (SP), Brasil.

5. M.N. Özisik, *Heat Conduction*, Wiley Interscience, 1980.

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

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

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

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

10. F.T. Miki, E. Issamoto, J.I. da Luz, P.B. de Oliveira, H. F. de Campos Velho, J.D. da Silva, An inverse heat conduction problem solution with a neural network approach, *Bulletim of the Braz. Soc. for Comp. Appl. Math*. (SBMAC), 2000, available in the internet: www.sbmac.org.br/publicacoes .

11. J. Krejsa, K.A. Woodbury, J.D. Ratliff, M. Raudensky (1999): Assessment of strategies and potential for neural networks in the IHCP, *Inverse Probl. Eng*., **7,** 197 (1999).

12. K.A. Woodbury, Neural networks and genetic algorithms in the solution of inverse problems, *Bulletim of the Braz. Soc. for Comp. Appl. Math*. (SBMAC), 2000, available in the internet: www.sbmac.org.br/publicacoes .

13. E.H. Shiguemori, F.P. Harter, H.F. de Campos Velho, J.D.S. da Silva, Estimation of boundary conditions in heat transfer by neural networks, *Braz. Cong. on Comp. and Appl. Math*., Belo Horizonte (MG), Brazil, 559 (2001).

14. H. Hidalgo, E. Gómez-Treviño, Application of constructive learning algorithms to the inverse problem, *IEEE T. Geosci. Remote*, **34**, 874 (1996).