Available online at www.sciencedirect.com
SCIENCE DIRECT®
International Journal of Heat and Mass Transfer 49 (2006) 2466–2478
International Journal of HEAT and MASS TRANSFER
www.elsevier.com/locate/ijhmt

Self-learning finite elements for inverse estimation of thermal constitutive models

Wilkins Aquino *, John C. Brigham

School of Civil and Environmental Engineering, Cornell University, Ithaca, NY 14853, USA

Received 10 March 2005; received in revised form 30 December 2005
Available online 29 March 2006

Abstract

In this work, a new methodology for inverse estimation of thermal constitutive models is introduced. This new methodology combines computational intelligence with finite element analysis for solving inverse heat transfer problems. A neural network (NN) representation of thermal constitutive behavior and its implementation in non-linear finite element analysis are presented. The self-learning methodology uses a novel concept for developing material models using experimental data and iterative finite element analyses. The proposed methodology searches for complete thermal constitutive models as opposed to identifying parameters in predetermined functional forms. The application of this new methodology is illustrated using a simulated steady-state heat conduction problem. It was found in simulated experiments that the self-learning finite element method can inversely recover accurate NN representations of thermal constitutive models using simple temperature measurements. Moreover, the method showed stability in the presence of imperfect or noisy data. It is shown that the use of a NN representation of the constitutive model improves the stability of solutions naturally due to the imprecision tolerance of NN.

Keywords: Finite elements; Neural networks; Self-learning finite elements; Autoprogressive neural networks; Heat transfer; Inverse problems

1. Introduction

Thermal properties of materials are commonly measured using controlled laboratory tests that employ simplistic specimen geometries and boundary conditions. However, thermal properties are difficult to determine through controlled tests when material behavior is complex such as is the case when thermal conductivity depends on temperature, position, and/or material orientation (i.e. anisotropy).

Inverse estimation of thermal properties is commonly cast as an optimization problem. The usual procedure consists in constructing a numerical representation of the heat transfer problem, defining an error functional, and minimizing this error or cost functional using gradient-based optimization algorithms. Examples of this approach can be found in the work of Scarpa et al. [1], Lin et al. [2], Huang and Chin [3], Alifanov [4], Tadi [5], Beck [6], among others. Despite significant progress made in this field, available methods for solving these inverse problems still present limitations when dealing with complex materials that involve anisotropy, heterogeneity, and non-linearity. In addition, the presence of imperfect or noisy information in the measured response, geometry, and/or initial and boundary conditions of the system may render current solution algorithms unstable [4].

More recent work in this area has involved the use of non-gradient based methods of optimization to solve the inverse heat transfer problem. Some examples of these techniques include the use of neural networks (NN) to directly map the solution of the inverse problem as shown in a comprehensive review reported by Ashforth-Frost et al. [7], as well as in [8–13]. Work has also been done in the use of stochastic global search methods such as genetic

* Corresponding author. Tel.: +1 607 255 3294; fax: +1 607 255 9004.
E-mail address: wa27@cornell.edu (W. Aquino).

0017-9310/$ - see front matter © 2006 Published by Elsevier Ltd.
doi:10.1016/j.ijheatmasstransfer.2006.01.031

algorithms to evolve a solution to the inverse problem as shown by Divo et al. [14].

This paper describes a new non-gradient based methodology for inverse identification of thermal constitutive models from laboratory or field measurements. This new methodology, called self-learning finite elements from hereon, uses a neural network (NN) representation of a material model incorporated into a non-linear finite element code, and the NN material model is trained using the autoprogressive method [15]. The autoprogressive method differs significantly from common applications of NN models in the sense that there is not a known set of data to train the NN a priori, but the material model is extracted iteratively from global measurements (e.g. temperature) using non-linear finite elements.

The use of NN to represent thermal constitutive behavior allows for very general and complex material representations that can include anisotropy, non-linearity, and heterogeneity in a natural way. This fact stems directly from the universal function approximation capabilities of NN. This new approach searches for entire constitutive thermal models defined as mappings between heat flux and variables such as temperature, temperature gradient, and spatial position.

It is well known that inverse problems are oftentimes ill-posed due to the lack of continuity of the solution on the experimentally measured data (i.e. instability), non-uniqueness, and existence of solutions. In the self-learning finite element methodology, stability of solutions arises naturally due to the imprecision tolerance of NN.

Self-learning finite elements can be cast in a general form to solve steady-state or transient heat conduction problems. For matter of clarity, this new methodology will be introduced in the context of steady-state problems. The formulation of the forward heat transfer problem is introduced first. Then, the NN representation of thermal constitutive models, and its implementation in finite element schemes are presented. After the finite element implementation, the self-learning algorithm is developed and a simple example is used to illustrate the method.

2. Formulation of the heat transfer problem

A steady-state 3D heat conduction problem can be described mathematically as follows:

$$\nabla \cdot \vec{J} = 0 \quad \text{in } \Omega, \tag{1}$$

$$\begin{aligned} q &= -\vec{J} \cdot \vec{n} \quad \text{on } \Gamma_q, \\ T &= T_0 \quad \text{on } \Gamma_T, \end{aligned} \tag{2}$$

$$\vec{J} = f(\nabla T, T, \vec{x}) \quad \forall \vec{x}, T \in \Omega, \tag{3}$$

where $\vec{J}$ is the heat flux, T is temperature, q is the external boundary heat flux, $\vec{x}$ is a position vector, Ω is a 3D domain, Γ is the boundary of the domain, Γ_q is the part of the boundary where the external heat flux is specified, Γ_T is the part of the boundary where temperature is specified,

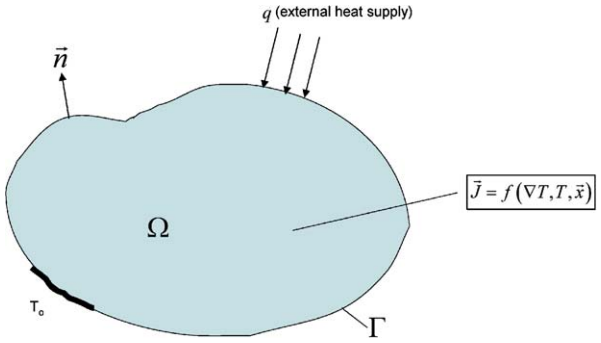

Fig. 1. Steady-state heat transfer problem.

and $f(\cdot)$ is a function that defines the constitutive thermal behavior of the material. In this work, column vectors are represented with an arrow and matrices and row vectors are represented with bold letters. The steady-state heat transfer boundary value problem is shown in Fig. 1. It is assumed that the geometry of the body and the boundary conditions represented in Eq. (2) are known. Although heat flux and temperatures are specified as boundary conditions, the methodology described in this work is general and other boundary conditions (e.g. convective, radiation, etc.) can be taken into consideration in a similar manner. The inverse problem consists in determining the unknown constitutive relationship described in Eq. (3) given boundary conditions and temperature measurements at discrete points in the body. It is important to clarify at this point that, as opposed to usual approaches, the methodology proposed in this paper is concerned with determining a thermal constitutive relationship instead of finding specific material parameters (e.g. thermal conductivity) based on postulated laws.

3. Neural network representation of thermal constitutive model

Neural networks (NN) are non-linear mapping systems that can generate complex behavior [16]. A NN consists of a number of computational units called neurons that are linked to each other through weighted connections. The basic principle behind the computations performed by a single neuron is illustrated in Fig. 2. Each neuron receives inputs, $X_1, \ldots, X_n$, from other neurons and produces a scalar output, y, through a non-linear function

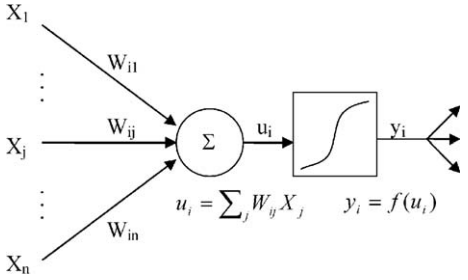

Fig. 2. A neural network computation at a single node.

of the weighted inputs. The scalar output of each neuron serves as an input to other processing neurons. The functional representation capability of the NN depends on the number of neurons, how they are interconnected, and the value of connection weights. Neural networks can be trained to learn the relationship between inputs and outputs through a process in which known cases are presented to the network and the connection weights are adjusted to minimize the error between the known values and the NN outputs. Once a NN is trained, it can generalize to other cases not used for its training. For a more complete description of NN and training approaches the reader is referred to [16].

As previously mentioned, NN have been commonly used as simple linear and non-linear regression tools in the field of heat and mass transfer. In the vast majority of NN applications in heat transfer, inputs and outputs of a system are measured and a NN model is trained directly with this information. The self-learning finite element methodology presented in this paper departs significantly from this common approach. In this approach, the universal function representation capabilities of NN are used to postulate very general thermal constitutive models, which are defined as mappings between heat flux and temperature gradient, temperature, and spatial position.

Ghaboussi et al. [17] introduced the concept of using NN to represent constitutive behavior in solid mechanics. Since then, NN have been successfully applied to model complex material behavior in the realms of plasticity and viscoplasticity [18,19]. Invoking this general concept, thermal constitutive behavior can also be modeled using NN.

The constitutive thermal model described by Eq. (3) can be represented by means of a NN as shown in Fig. 3. The NN material model shown in Fig. 3 can be represented as [15]

$$\vec{J} = JNN\{\nabla T, T, \vec{x}\}. \tag{4}$$

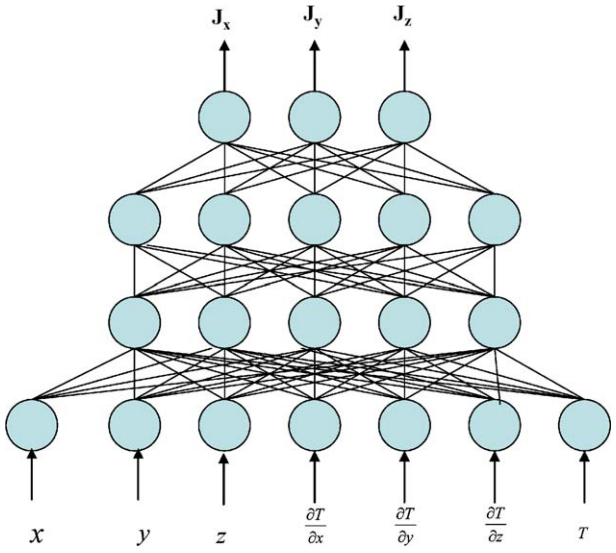

Fig. 3. Neural network representation of a thermal constitutive model.

Eq. (4) indicates that the NN takes T, ∇T, and $\vec{x}$ as inputs and produces $\vec{J}$ as the output. It should be noticed that Fourier's law is one form of thermal constitutive model falling within those represented by Eq. (3).

4. Finite element implementation of neural network thermal constitutive models

Let δT be an arbitrary continuous virtual temperature field defined on Ω and its boundary Γ. The weak form of the governing energy conservation partial differential equation can be obtained by multiplying Eq. (1) by δT, integrating over the domain, using the divergence theorem, and substituting the natural boundary conditions given in Eq. (2) to render

$$\int_{\Gamma} \delta T q \, d\Gamma + \int_{\Omega} \nabla \delta T \cdot \vec{J} \, d\Omega = 0. \tag{5}$$

To obtain an algebraic system of equations from the weak form presented in Eq. (5), the body is divided into finite elements and the temperature field is approximated within each element as

$$T(\vec{x}) = \mathbf{N}(\vec{x})\vec{T}^{e}, \tag{6}$$

where $\mathbf{N}(\vec{x})$ is a row vector containing interpolation functions and $\vec{T}^{e}$ is a column vector of element nodal temperatures. Using the Galerkin approach, the virtual temperature field is also interpolated as shown in Eq. (6). Substituting the approximated temperature and virtual temperature fields into Eq. (5) and adding individual element contributions, we obtain a system of non-linear algebraic equations as

$$\delta\vec{T}_{g}^{T}\left(\sum_{\text{elements}} \int_{\Gamma^{e}} \mathbf{N}^{T} q \, d\Gamma^{e} + \sum_{\text{elements}} \int_{\Omega^{e}} \mathbf{B}^{T}\vec{J} \, d\Omega^{e}\right) = 0, \tag{7}$$

where $\delta\vec{T}_{g}$ is a vector containing all nodal virtual temperatures in the domain, Ω^{e} is the volume of an element, Γ^{e} is the boundary of an element, and

$$\mathbf{B} = \frac{\partial \mathbf{N}}{\partial \vec{x}}. \tag{8}$$

Since Eq. (7) must hold for all $\delta\vec{T}_{g}$, the expression in parentheses is equal to zero.

Introducing the external heat flux vector supplied to the body as

$$\vec{Q} = \sum_{\text{elements}} \int_{\Gamma^{e}} \mathbf{N}^{T} q \, d\Gamma^{e}, \tag{9}$$

and the internal heat flux vector as

$$\vec{I} = \sum_{\text{elements}} \int_{\Omega^{e}} \mathbf{B}^{T}\vec{J} \, d\Omega^{e}, \tag{10}$$

then the system of non-linear algebraic equations can be simply expressed as

$$\vec{Q} + \vec{I} = 0. \tag{11}$$

The internal heat flux vector, $\vec{I}$, is usually computed using numerical integration, which requires computing the heat flux, $\vec{J}$, at Gauss points from the NN constitutive model described above.

The Newton–Raphson method is commonly used for the solution of the non-linear system described in Eq. (11). This solution procedure requires the evaluation of a Jacobian matrix defined as the partial derivative of the internal heat flux vector, $\vec{I}$, with respect to the global nodal temperature vector, $\vec{T}_{\mathrm{g}}$. Mathematically, the Jacobian matrix can be computed as

$$\mathbf{A} = \frac{\partial \vec{I}}{\partial \vec{T}_{\mathrm{g}}}. \tag{12}$$

Substituting Eqs. (6), (8) and (10) into Eq. (12), and using the chain rule of differentiation, yields

$$\mathbf{A} = \sum_{\text{elements}} \int_{\Omega^{\mathrm{e}}} \mathbf{B}^T (\mathbf{K}\mathbf{B} + \vec{r}\mathbf{N})\, \mathrm{d}\Omega^{\mathrm{e}}, \tag{13}$$

where

$$\mathbf{K} = \frac{\partial \vec{J}}{\partial \nabla T} \tag{14}$$

and

$$\vec{r} = \frac{\partial \vec{J}}{\partial T}. \tag{15}$$

When a NN constitutive model is used, the matrix $\mathbf{K}$ contains the partial derivatives of the NN outputs with respect to the temperature gradient inputs, while the vector $\vec{r}$ contains the partial derivatives of the NN outputs with respect to the temperature input. These quantities are evaluated at Gauss points in the finite elements during the numerical integration process.

A NN constitutive model is a general mapping of the relationship between temperature, temperature gradients, and fluxes. Therefore, the NN constitutive model can represent any material, including those that may not obey Fourier's law. If the material obeys Fourier's law, the partial derivatives of the NN output with respect to the temperature gradients, as shown in Eq. (14), produce the material thermal conductivity matrix.

The matrix defined in Eq. (14) can be computed from the NN constitutive model by taking derivatives of the NN flux outputs with respect to the temperature gradient inputs using an analytical approach, as shown by Hashash et al. [20] and Cardaliaguet [21], or by using a numerical approximation. In this work, the latter was used for simplicity. The jth column of the $\mathbf{K}$ matrix can be evaluated by a forward difference approximation as

$$\mathbf{K}(:,j) = \frac{\vec{J}NN\{\nabla T + \varepsilon \vec{e}_j, T, \vec{x}\} - \vec{J}NN\{\nabla T, T, \vec{x}\}}{\varepsilon}, \tag{16}$$

where $\vec{e}_j$ is the jth column of the identity matrix. The entries of the column vector $\vec{r}$ can be computed in a similar manner by differentiating the NN outputs with respect to the temperature input.

5. Self-learning algorithm for inverse solution

Neural network models are usually trained with data that is determined directly from experiments. A drawback of this approach is the need for a comprehensive set of data (e.g. heat fluxes, temperature, and temperature gradients) to train a NN material model. Such a comprehensive set involves a large amount of information that is often difficult to obtain directly from laboratory tests. Previous research [15] has shown that a comprehensive data set for training a NN material model may be obtained through iterative non-linear finite element analyses. To realize this, two essential concepts must be recognized. First, during a forward finite element analysis of the boundary value problem described in Section 2, local quantities that define material behavior such as heat flux, temperature gradients, and temperature are calculated point wise for the entire domain. Second, temperatures at any point in the domain carry information about the solution of the boundary value problem, which in turn contains information about the local material behavior. The self-learning finite element methodology described herein uses these two essential concepts to train a NN material model and inversely reconstruct thermal constitutive models from sparse experimental temperature measurements.

The main components of the self-learning method are illustrated in Fig. 4a. As shown in this figure, an experiment is carried out in which temperatures are measured in the boundary and/or inside a body of known geometry and boundary conditions. Then, a finite element model of the body is constructed. This finite element representation uses a NN as its constitutive material representation, as explained in Section 4. Each cycle or iteration of the self-learning algorithm consists of two finite element analyses, data collection from these two analyses to form a training set, and training of the NN material model with this data set. In the first finite element analysis only heat flux, $\vec{J}$, is recorded at all Gauss points, while in the second analysis measured temperatures are enforced as constraints in the analysis and only the temperature, T, and temperature gradient, ∇T, are recorded at the Gauss points. The NN model is then trained using this data, which contains information unknown to the NN. Notice that the NN model produces the data that is used for its own training, which is the reason why the methodology is called "self-learning." These steps and the rationale behind them are explained in detail below and are illustrated in the flowchart given in Fig. 4b.

5.1. Pretraining or initialization of NN model

The first step in the self-learning algorithm is to initialize the NN material model. This initialization is performed by training the NN to represent an arbitrary constitutive relationship, which is consistent with the physics of the problem. For instance, an initialization training set can be created by generating random temperatures and temperature gradients, and producing the corresponding material

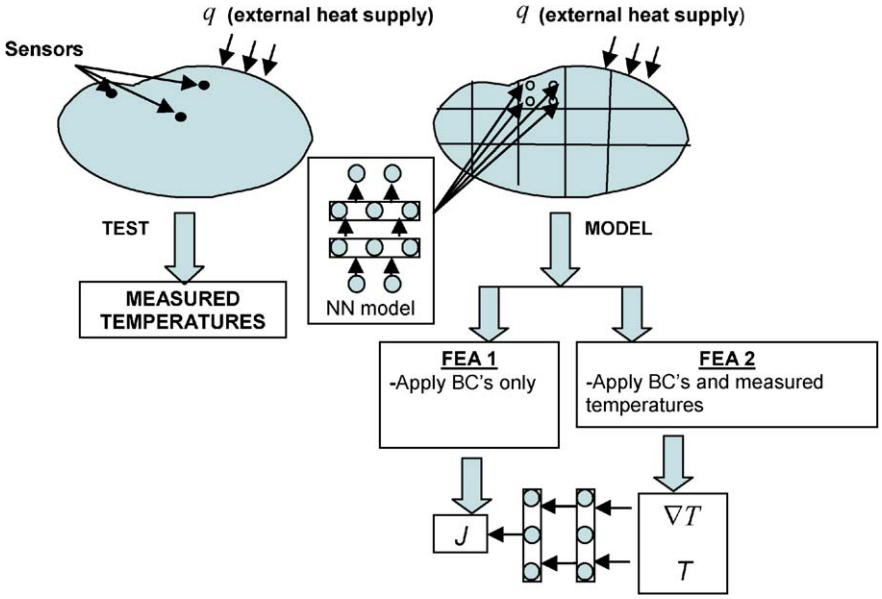

Fig. 4a. Schematic diagram of the self-adaptive finite element analysis.

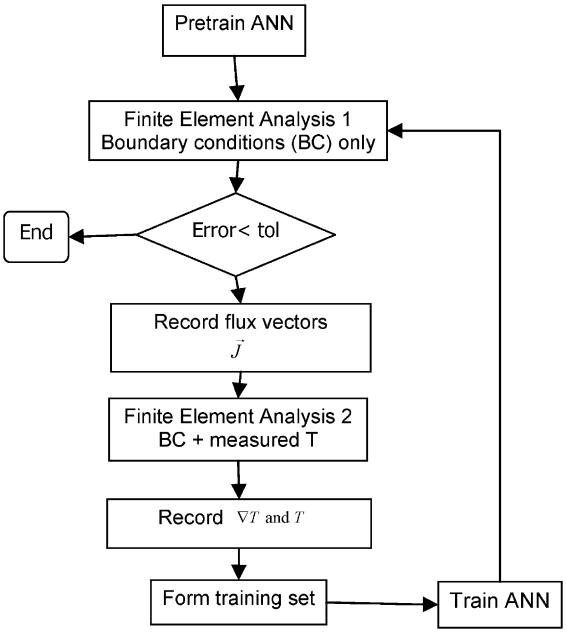

Fig. 4b. Flowchart of the self-adaptive finite element analysis.

point heat fluxes using Fourier's law with a constant thermal conductivity. The purpose of this step is to avoid instabilities in the finite element analysis during the first few runs that could arise when a NN is initialized with random weights without training.

5.2. First finite element analysis

A finite element analysis is performed and the heat flux vector $\vec{J}$ is recorded at all Gauss points in the model. The first finite element analysis computes a forward solution of the boundary value problem with the current NN material model, and is used to determine the internal heat flux field. This heat flux field is in equilibrium with the actual boundary conditions. However, the measured temperatures, in general, will disagree with the temperatures calculated in the first analysis because the NN constitutive model, to this end, does not represent the real material behavior. For this reason, only heat flux vectors are saved during the first analysis.

5.3. Second finite element analysis

A second finite element analysis is performed using the known boundary conditions, but also the measured temperatures are imposed as known constraints. Temperature and temperature gradients are recorded at all Gauss points. For better understanding of the reason behind this step, it is convenient to express the vector of measured temperatures as the sum of calculated temperatures plus a temperature correction as

$$\vec{T}^{\mathrm{M}} = \vec{T}^{\mathrm{FEA}} + \Delta\vec{T} \tag{17}$$

where $\vec{T}^{\mathrm{M}}$ is a vector containing measured temperatures, $\vec{T}^{\mathrm{FEA}}$ is a vector containing the temperatures at measurement locations computed in the first finite element analysis, and $\Delta\vec{T}$ is a vector containing the discrepancy between measured and computed temperatures. By imposing measured temperatures as additional boundary conditions, temperature corrections are propagated to all points in the domain during the second analysis, resulting in a temperature field that is closer to the temperature distribution of the actual physical system. However, the heat flux field resulting from this finite element analysis is not in equilibrium with the actual boundary conditions due to the

imposition of temperature constraints. Therefore, only temperature and temperature gradients are saved in the second analysis.

5.4. Training the NN material model

A NN training set is then formed with the temperatures and temperature gradients from the second finite element analysis as inputs to the NN, and the corresponding material point heat flux vectors from the first finite element analysis as outputs to the NN. The NN is trained with the above data set using any conventional training algorithm [16]. For this work, the resilient propagation algorithm (RPROP) was used to train the NN material model.

The heat fluxes from the first analysis and the temperatures and temperature gradients from the second analysis represent a data set that is more consistent with the measured behavior of the real system. Once trained with the new data set, the NN material model becomes a better approximation to the actual material behavior.

5.5. NN architecture

The NN architecture refers to the number of input neurons, output neurons, hidden layers, and neurons per hidden layer contained in the network. The architecture also refers to the method by which information propagates through the network. For this work, only fully connected feed-forward NN were considered, in which information passes in only one direction. That is, information flows from the input neurons to the output neurons where each neuron in a layer receives information from all neurons in the previous layer.

The number of input and output neurons of a NN is specified by the desired relationship to be mapped by the network (e.g. temperature and temperature gradients as inputs and heat fluxes as output). The number of hidden neurons and hidden neuron layers are non-unique quantities that depend on the complexity of the relationship to be mapped. In the case that too few hidden neurons are used, the NN will not be able to adequately map the training set values. Whereas, if too many hidden neurons are used, the NN can be subject to over training, in which, the interpolation ability of the NN for information outside of the initial training set will be adversely affected. In the case that sufficient knowledge is known a priori about the complexity of a data set to be mapped with a NN, the number of hidden neurons can be determined heuristically or through trial and error. Otherwise, it is advantageous to implement an adaptive method for determining the number of hidden neurons, as described by Joghataie et al.[22]. In an adaptive method, a NN is created with an arbitrarily small number of hidden neurons. Then, the number of hidden neurons is increased iteratively based on the ability of the NN to learn a portion of the training set and predict another portion of the training set.

5.6. Stopping criterion

A criterion needs to be defined for deciding when to stop the self-learning algorithm. In this research, the following procedure was used. A finite element analysis is performed with the updated NN model, and an error measure between the calculated and measured temperatures is computed. If this error is below a certain user-defined tolerance or a maximum number of iterations are exceeded, the self-learning process is stopped. Otherwise, heat fluxes at Gauss points are recorded (i.e. step 5.2) and steps 5.3 and 5.4 are repeated.

An error measure is needed for deciding when to stop the self-learning training process. Any suitable error definition can be used to measure the discrepancy between the finite element prediction and the observed quantities. For this work, we have used the following error definition:

$$E_{\text{Global}} = \sum_{k} \left(T_k^{\text{MEA}} - T_k^{\text{FEA}}\right)^2, \tag{18}$$

where T^{MEA} are the measured temperatures, T^{FEA} are the computed temperatures, and k is the index for temperature measurement points.

5.7. Global and local errors

During the self-learning process, two error definitions are used: a global error and a local error. The global error is defined by Eq. (18) and represents the discrepancy between the approximate solution to the boundary value problem and experimental measurements. The local error is the error associated with the NN learning process. During training, the NN material model learns the updated data set to within a preset tolerance. The mean squared error is commonly used in the training of NN, and it is calculated as

$$E_{\text{MSE}} = \frac{1}{NP} \sum_{i=1}^{N} \sum_{p=1}^{P} (t_{pi} - y_{pi})^2, \tag{19}$$

where N is the total number of outputs of the NN, P is the total number of training patterns, p is the index of the training patterns, and i is the index of the NN outputs.

In the event that the global error stagnates at a high value, it is likely that the NN is not sufficiently complex to map the constitutive relationship for the problem. This should also be evident in the inability of the NN training to converge to a sufficiently low local training error. This issue can be addressed by interactively adjusting the architecture of the NN, or by implementing an adaptive method for estimating the size of the network [15].

6. Example problem

A simulated example problem was created to demonstrate the feasibility of the self-learning finite element methodology described above. A simulated experiment is shown

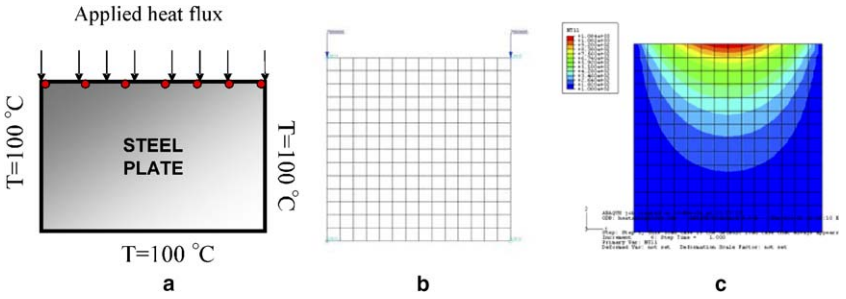

Fig. 5. Simulated experiment and finite element model: (a) simulated test, (b) finite element representation, (c) temperature distribution from finite element analysis.

in Fig. 5a. In this simulated test, a rectangular plate (20 cm × 20 cm × 1 cm) was heated with a surface heat flux equal to 7 MW/m^2 on one of its sides, while a temperature of 100 °C was held constant on the other three sides. This configuration provides a non-uniform spatial distribution of temperatures, temperature gradients, and internal fluxes suitable for training a NN material representation. A finite element model of the plate is shown in Fig. 5b, and a temperature contour plot showing the temperature distribution from an analysis of the model using a known conductivity function is shown in Fig. 5c. The temperature distribution plot shows that the largest temperature variations occur towards the top of the plate where the heat flux is applied. Therefore, it is logical that measurement points should be located in this region to exploit the largest amount of information possible during the inverse solution process.

The material used in the simulated experiment was isotropic and homogeneous, and a temperature-dependent conductivity function reported by Stelzer and Welzel [23] was used to generate the experimental data. The experimental temperature measurements at specified points were obtained through finite element simulations using the known conductivity function. The conductivity function used in the simulated experiment was defined as

$$\kappa(T) = 5 \times 10^{-8} T^3 - 9 \times 10^{-5} T^2 + 5.1 \times 10^{-3} T + 70.1. \tag{20}$$

In order to study the effect of the number of measurement points on the solution of the inverse problem at hand, three different configurations of measurement points were investigated (Fig. 6a–c). Configuration A had one row of measurement points at the top of the plate, Configuration B had three rows near the top, and Configuration C had five rows near the top. Other arrangements of measurement points were also investigated, but only selective results are shown for brevity. As will be shown later, Configurations A, B, and C clearly illustrate the effect of the number of measurements on the inverse solution. The measurement points were taken to coincide with nodes in the finite element mesh for convenience in the application of measured temperatures in the second finite element analysis of the self-learning algorithm.

When applying this method to an actual experiment, the number and layout of measurement points will be dictated by a number of factors, including geometry, boundary conditions, and experimental equipment available. For the purpose of this proof of concept, the measurement points in the example were chosen for simplicity to demonstrate the performance of the self-learning algorithm, and the effect of the amount of measurement information provided to the algorithm. It is expected that other more practical sensor configurations used in real experiments would also yield satisfactory results.

In order to study the stability of the self-learning finite element methodology, a random Gaussian noise was introduced in the simulated data for Configurations B and C, which correspond to 3 and 5 rows of measurement points, respectively. The random Gaussian noise was introduced in the simulated data as

$$T_{\text{error}} = T_{\text{exact}} + \delta_0 \omega, \tag{21}$$

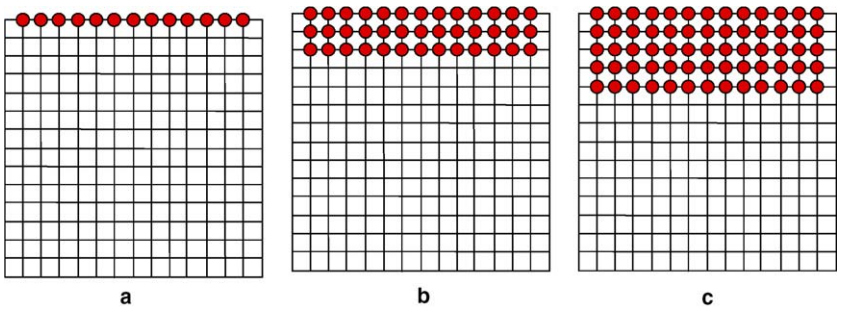

Fig. 6. Measurement points configurations used for three different simulated experiments: (a) Configuration A, (b) Configuration B, (c) Configuration C.

where T_{error} is the temperature containing the random noise, T_{exact} is the temperature of the simulated test without noise, ω is a normally distributed random variable with a unit variance and zero mean, and $\delta_0 = 1\% \max(T_{exact})$. It was found that this equation produced realistic values of temperature deviations expected during laboratory or field measurements.

6.1. Neural network architecture and pretraining

The architecture of the NN thermal constitutive model consisted of three inputs (i.e. $\frac{\partial T}{\partial x}$, $\frac{\partial T}{\partial y}$ and T), two outputs (J_x and J_y), and two hidden layers of 12 nodes each. Although a simpler architecture could have been used (i.e. two inputs and one output) due to isotropy, the architecture described above was used to illustrate a more general approach that could also be applied to anisotropic materials. An adaptive method for determining the network architecture was not used in this example due to the fact that for a simulated experiment the necessary complexity of the NN could be determined a priori. It was determined through interactive trials that two12-neuron hidden layers could capture the desired constitutive behavior for this problem.

The NN material model was pretrained using a constant conductivity of 70.5 W/m °C.

Initialization of the NN material model was carried out using a uniform distribution of randomly generated temperatures and temperature gradients from which the corresponding heat fluxes were calculated using Fourier's law as

$$\begin{Bmatrix} J_x \\ J_y \end{Bmatrix} = -\begin{bmatrix} \kappa & 0 \\ 0 & \kappa \end{bmatrix} \begin{Bmatrix} T_{,x} \\ T_{,y} \end{Bmatrix}. \tag{22}$$

The random temperatures generated for initializing the NN model were uniformly distributed in the range between 0 and 1200 °C and the temperature gradients were between −500,000 and 500,000 °C/m, resulting in heat fluxes between −32,250,000 and 32,250,000 W/m^2. The training set for initialization of the NN material model contained 600 patterns generated in the ranges stated above, and each patter consisted of two temperature gradients (X- and Y-direction), one temperature, and two heat fluxes (X- and Y-direction). The NN material model was then trained using the resilient propagation (RPROP) algorithm until the local error defined in Eq. (19) was less than or equal to 10^{-6}.

After the NN was initialized, the self-learning algorithm was used to extract a thermal constitutive NN model using the simulated experimental data. The results of this process are shown next.

6.2. Results

Five different cases of the inverse analysis were studied. The first three cases were defined as Configurations A, B, and C with no noise added to the simulated experimental data. These cases were labeled 1RF (1 Row of measurement points, Free of noise), 3RF, and 5RF, respectively. The remaining two cases were defined as Configurations B and C with random noise added to the simulated experimental data. These two cases were labeled 3RN (3 Rows of measurement points, with Noise) and 5RN, respectively.

Plots of the outputs, J_x and J_y, of the inversely recovered NN material model versus temperature and temperature gradient are shown in Figs. 7–11 for all the studied cases. Black dots in these plots represent data entries produced by the NN material model at Gauss points, while the plotted surface represents the actual material behavior described by Eq. (22) and using the temperature-dependent conductivity from Eq. (20). Figs. 7–9 correspond to cases 1RF, 3RF, and 5RF, respectively. It can be observed that in all three cases the NN was able to satisfactorily approximate the real material behavior, although Case 1RF showed a larger deviation from the solution in the Y-direction due to the lower information content available in that set of measurement locations. It can be seen in Fig. 7b that a larger discrepancy exists between the NN prediction and the true solution than that observed for the other cases (3RF and 5RF). Figs. 8 and 9 show that better approximations to the heat flux in the Y-direction were obtained as the number of measurement points (information content) was increased beyond one row.

Figs. 10 and 11, which correspond to cases 3RN and 5RN (i.e. noise added to the simulated experimental data), respectively, show that the recovered NN material model

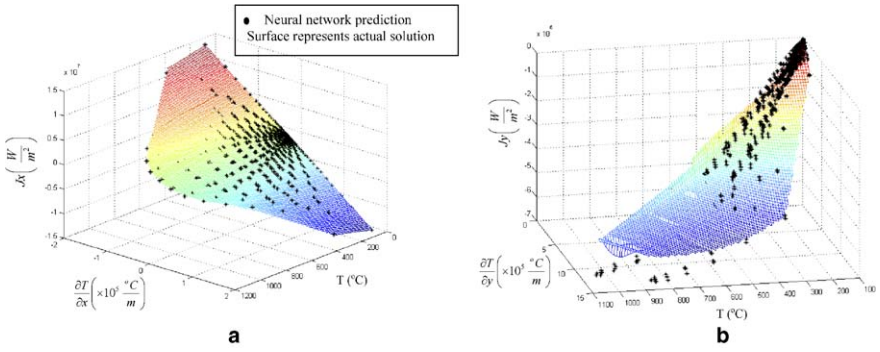

Fig. 7. Case 1RF: Configuration A—no noise. (a) Flux in X-direction, (b) flux in Y-direction.

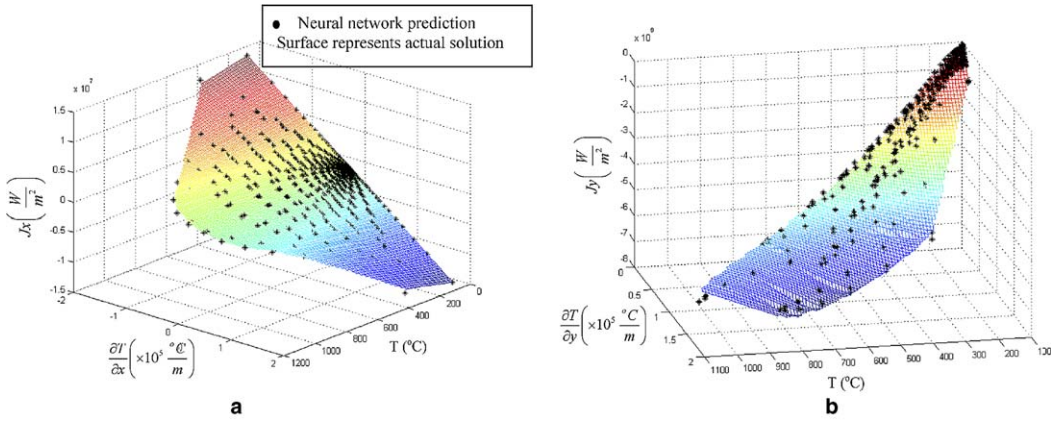

Fig. 8. Case 3RF: Configuration B—no noise. (a) Flux in X-direction, (b) flux in Y-direction.

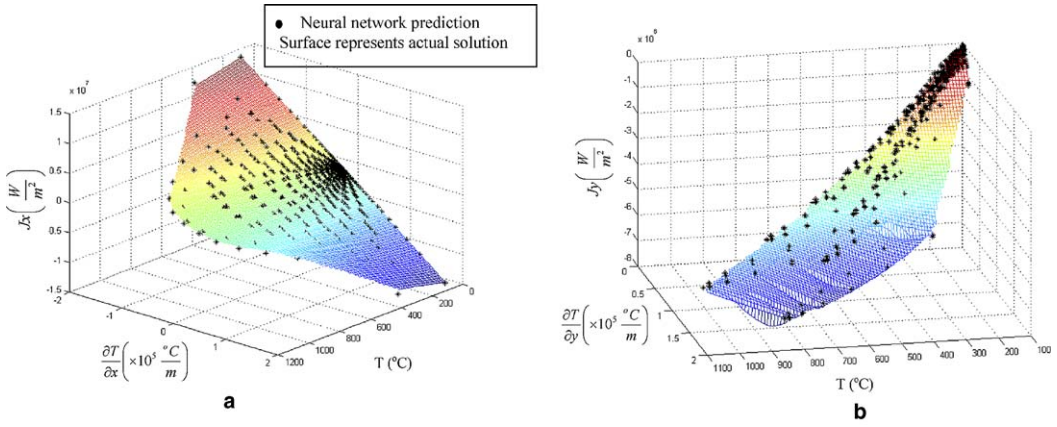

Fig. 9. Case 5RF: Configuration C—no noise. (a) Flux in X-direction, (b) flux in Y-direction.

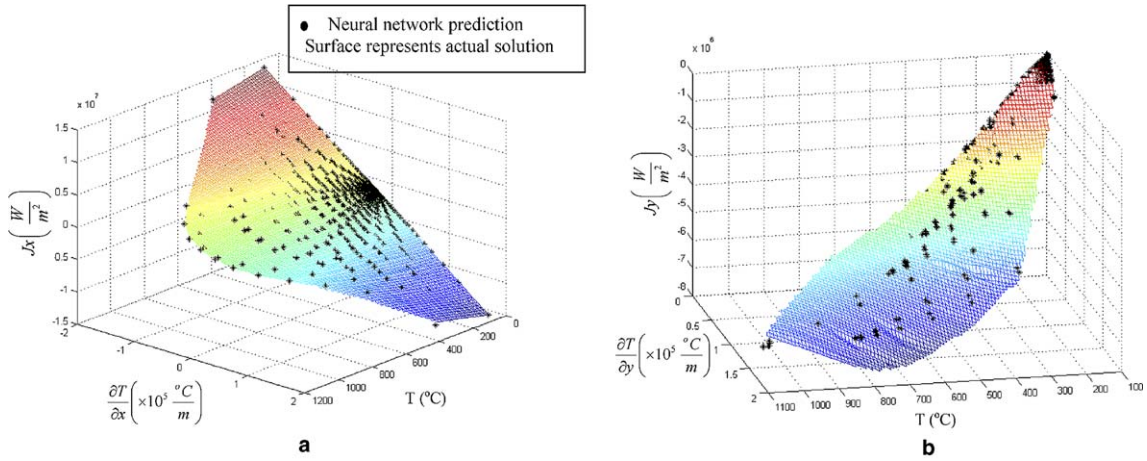

Fig. 10. Case 3RN: Configuration B—with noise. (a) Flux in X-direction, (b) flux in Y-direction.

was less accurate than in the noise-free cases, as expected, but was still able to approximate satisfactorily the true material behavior.

In order to quantify the accuracy of the recovered NN model, an average error, E_{flux}, for each output of the NN was calculated as

$$E_{\text{flux}} = \frac{1}{M}\sum_{\substack{m=1 \\ a=x,y}}^{M}\left|\frac{J_{\text{am}}^{\text{exact}} - J_{\text{am}}^{\text{NN}}}{J_{\text{am}}^{\text{exact}}}\right|, \tag{23}$$

where $J_{\text{am}}^{\text{exact}}$ is the component of heat flux in the X or Y direction corresponding to the real material model, $J_{\text{am}}^{\text{NN}}$ is

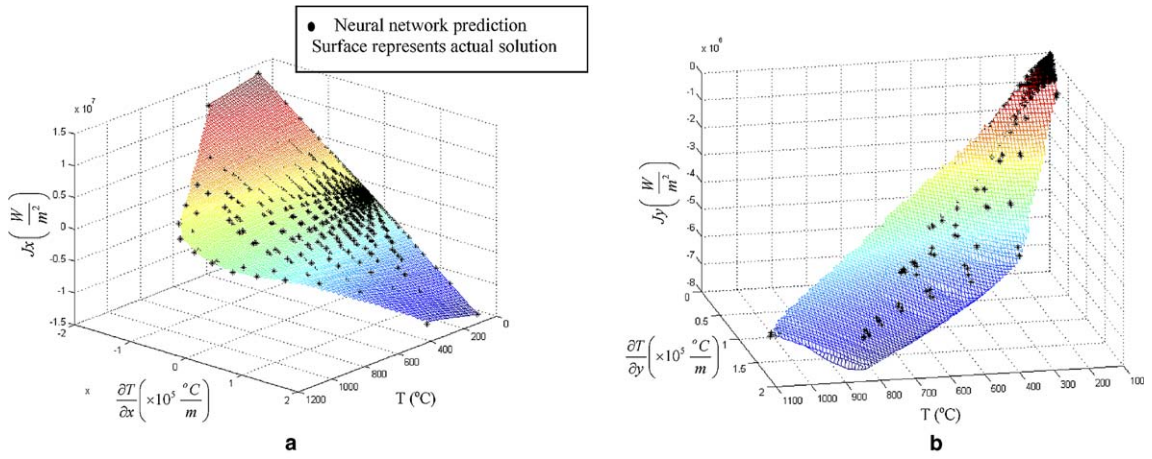

Fig. 11. Case 5RN: Configuration C—with noise. (a) Flux in X-direction, (b) flux in Y-direction.

the corresponding heat flux component obtained using the NN material model, m indexes the Gauss points, and M is the total number of Gauss points in the finite element model. The average error computed using Eq. (23) for the studied cases is illustrated in Fig. 12. It can be observed that for the noise-free cases the prediction improved as the number of measurements points increased, as expected. The average error in heat flux increased when noise was added to the data, and was of the same order of magnitude as the error introduced in the simulated measurements. Case 5RN showed a higher average error than Case 3RN, but this behavior is normal because of the random nature of the noise.

To this end, it has been shown that the self-learning methodology was able to inversely recover a NN material model that relates heat fluxes to temperatures and temperature gradients for a given material. But, has the NN also learned other fundamental information such as isotropy? In order to answer this question the entries of the matrix **K**, defined in Eq. (14), for the noise-free cases are examined. If enough information is available to the NN through the training data generated during the self-learning process, the diagonal entries of the matrix **K** should approximate the known conductivity function $\kappa(T)$, and the off-diagonal entries should be close to zero [refer to Eq. (22)]. Fig. 13a–c show plots of the diagonal entries of **K** matrix (i.e. K_{xx} and K_{yy}) computed from the NN material model at Gauss points versus temperature, while Fig. 14a–c show plots of the off-diagonal entries (i.e. K_{xy} and K_{yx}) versus temperature. It can be observed from the plots that the NN material model indeed approximates the conductivity function in all cases. However, the best approximation of the conductivity function occurs when five rows of measurements are used. This fact indicates that information about isotropy of the material was indeed present in the training data, and this information was enriched as more measurement points were added. The off-diagonal entries depart significantly from zero for cases 1RF and 3RF, indicating lack of information about material symmetry in the data. For Case 5RF, the off-diagonal entries were very close to zero, which means that this experimental setup was able to reveal very complete information about the material behavior.

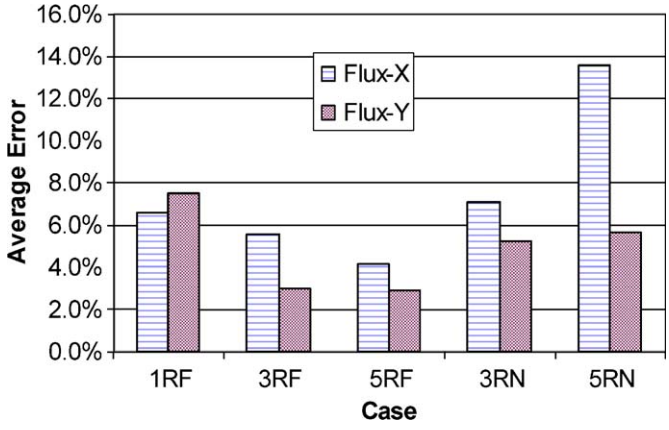

Fig. 12. Average error in neural network outputs with respect to exact solution.

The foregoing results demonstrate that the self-learning algorithm is able to inversely recover not only the mapping between temperature, temperature gradients, and heat flux, but also fundamental information about material symmetry (i.e. isotropy or anisotropy). However, it is very important to realize that the performance of the NN material model should be judged primarily from the mapping between temperatures, temperature gradients, and heat flux (Figs. 8–11) since this is the main target of the inverse problem solution. In addition, it is important to bear in mind that thermal conductivity is recovered through differentiation of the NN material model. Since differentiation is a coarsening operator (i.e. degrades function smoothness), it should be expected that the NN model would require more information to adequately capture both the mapping of inputs and outputs and their derivatives.

It is important to realize that because Fourier's law was not used in the finite element formulation described in this

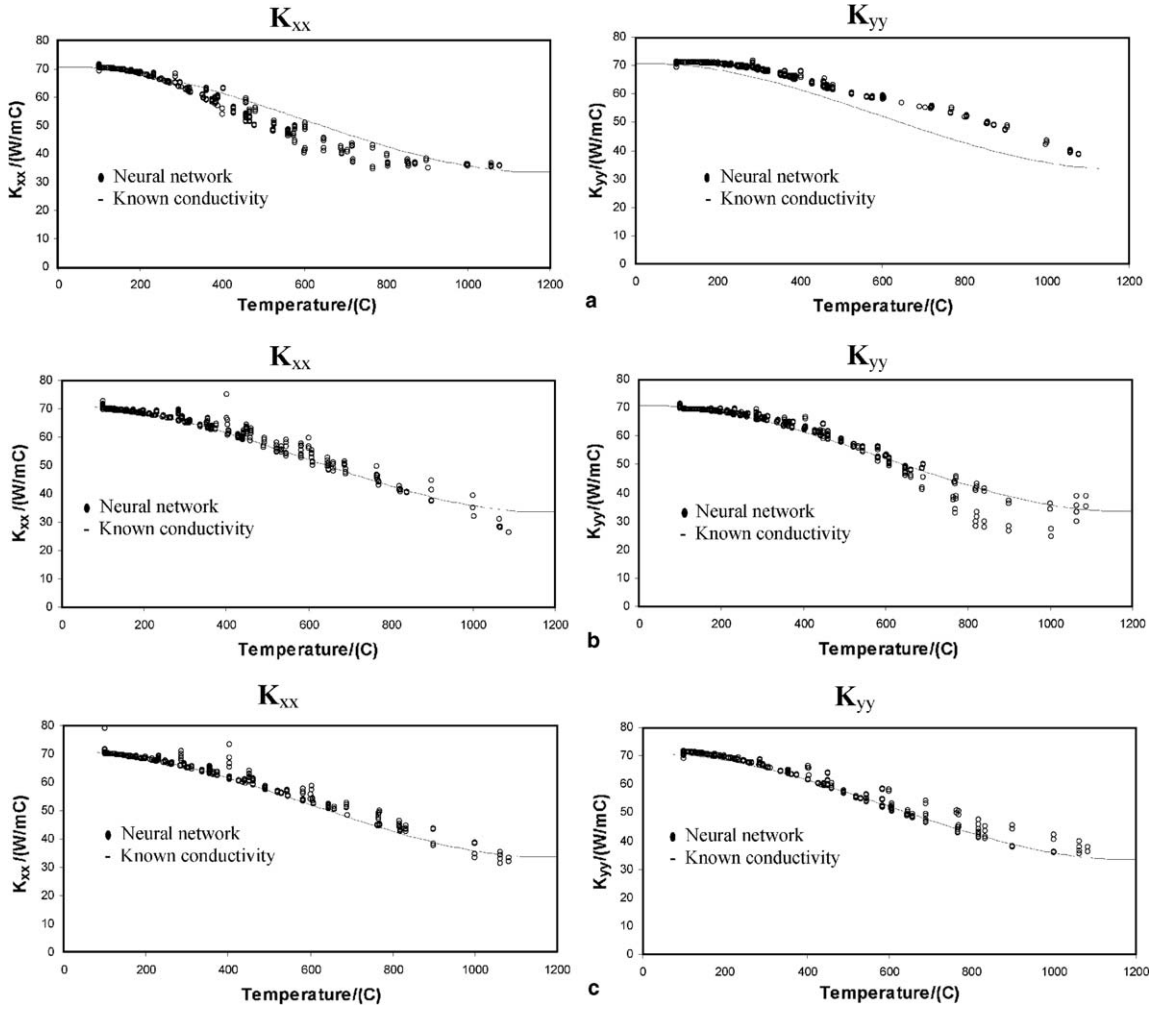

Fig. 13. Diagonal entries in matrix versus temperature. (a) Case 1RF: Configuration A—no noise, (b) Case 3RF: Configuration B—no noise, (c) Case 5RF: Configuration C—no noise.

work (see Section 4), negative quantities may occur in the off-diagonal entries and/or in the diagonal entries of the **K** matrix during the inversion process. This is acceptable in finite element analysis as long as the Jacobian matrix, **A**, remains positive-definite. The reader should bear in mind that the heat flux outputted by the NN is used to assemble the internal heat flux vector in Eq. (10), which is used to check energy balance in the body using Eq. (11). Moreover, the Jacobian matrix, **A**, influences the search direction of the Newton–Raphson algorithm, but does not play a direct role in the energy balance equations. The procedure presented in Section 4 for computing the Jacobian matrix from the NN material model preserves the quadratic convergence property of The Newton–Raphson algorithm.

Fig. 15 depicts the convergence of the self-learning algorithm in the presence and absence of noise. The square root of the global error from Eq. (18) versus iterations is shown. One iteration corresponds to two finite element analyses and training of the NN material model. It can be observed from Fig. 15 that the procedure converged monotonically for both cases. In addition, the solution was very stable in the presence of noise.

7. Discussion

The self-learning finite element method presented in this article is a hybrid of physical, mathematical, numerical, and artificial intelligence based formulations. Self-learning finite elements convey observable information (i.e. temperature measurements, external heat loads, etc.) from the global system to the local NN material model. This process constrains the NN to learn physically and mathematically feasible information.

During the error minimization in self-learning training, temperatures calculated from the first finite element analysis approach the measured temperatures, and the data sets from the two finite element analyses become close to each other (i.e. data sets converge). Eventually, the NN model will be trained with data that carries no new information and will not improve any further. The convergence of the data sets from the two finite element analyses produces a

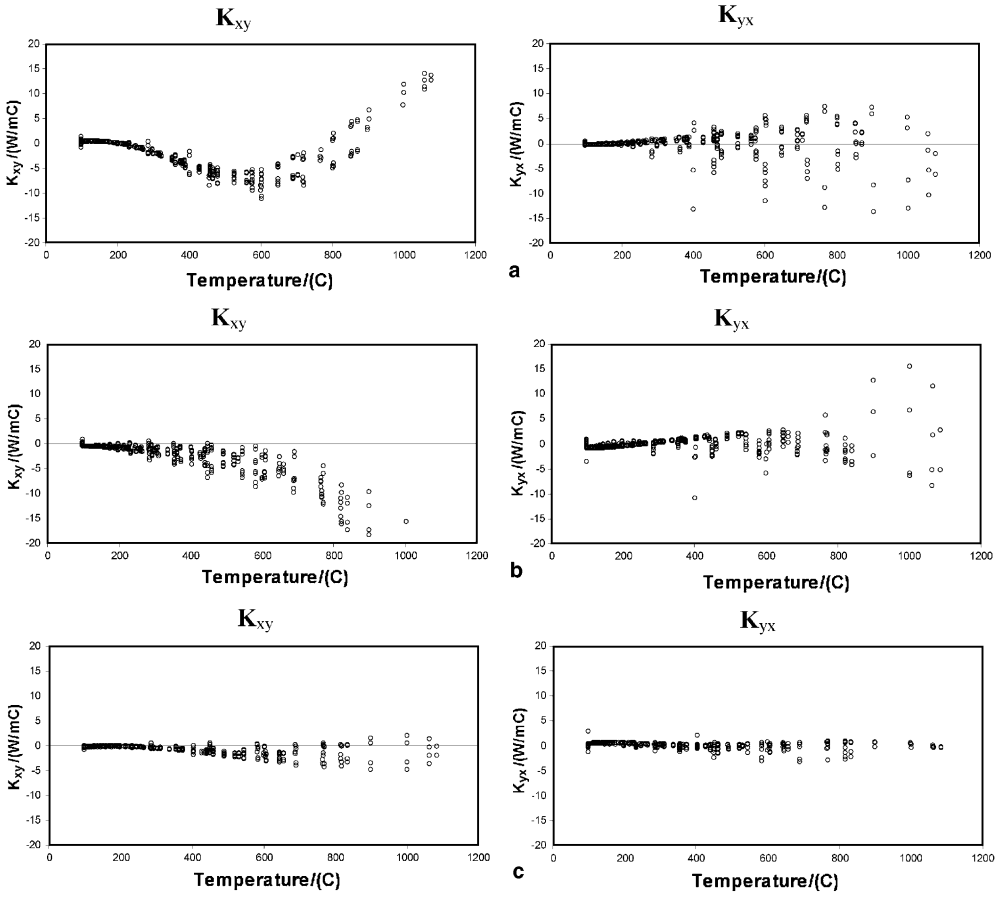

Fig. 14. Off-diagonal entries in matrix versus temperature. (a) Case 1RF: Configuration A—no noise, (b) Case 3RF: Configuration B—no noise, (c) Case 5RF: Configuration C—no noise.

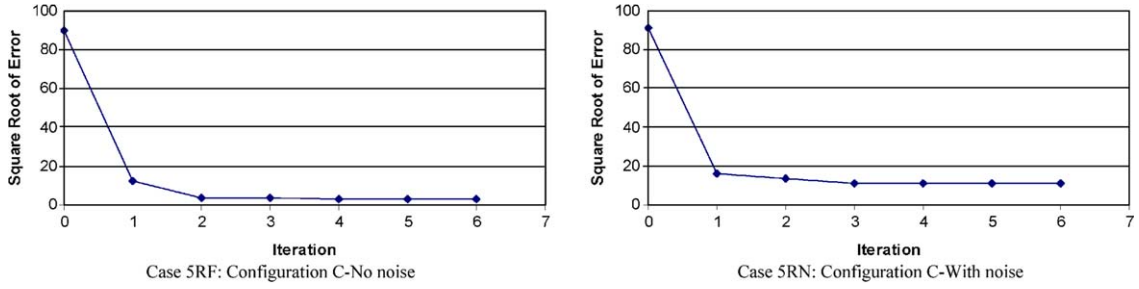

Fig. 15. Global error behavior.

very stable error behavior since the NN material model will be essentially recycling its own data. This is also the reason why, the self-learning method is expected to be very stable in the presence of noisy data. Once the training data from both analyses become sufficiently close, the NN will quickly reach its error tolerance and will not continue training. Of course, if a very tight error tolerance is set, the NN will learn the error caused by the noise, and this will be reflected in the recovered material model. As was shown above, the average error in the trained NN material model is of the same order as the error contained in the measurements, which is acceptable for any inverse problem methodology.

As with any inverse problem solution, uniqueness is not guaranteed in self-learning procedures. It is important that enough experimental measurements are available to constrain the solution. This was demonstrated in the example presented in the previous section. For simple problems, it is possible to estimate this number of measurements a priori using arguments found in inverse problem theory. This uniqueness and generalization of the NN model can only be tested so far using ad hoc approaches. That is, by employing different experimental configurations and using numerical simulations with the trained NN material model to predict the behavior of the new experimental

configurations. However, this drawback affects other classical inverse problem techniques as well.

8. Conclusions

This paper introduced a new methodology based on computational intelligence and conventional finite element analysis for inverse estimation of thermal constitutive models. This methodology, termed "self-learning finite elements" uses a neural network representation of the material behavior embedded in a non-linear finite element scheme.

Self-learning finite element schemes present the advantage that very general material behavior representations can be obtained due to the universal function approximation capabilities of NN. The methodology showed good stability in the inverse recovery of solutions. This fact stems from the noise tolerance of NN. Although the methodology was introduced in the context of steady-state heat transfer, it can be readily extended to transient heat transfer problems and coupled heat and mass transport problems.

Acknowledgements

This research was supported through funding provided by Cornell University. The authors wish to thank Mr. Shawhin Roudbari for his help in developing the initial proof of concept that led to this research.

References

[1] F. Scarpa, G. Milano, D. Pescetti, Thermophysical properties estimation form transient data: Kalman versus Gauss approach, in: Proceedings of the 1st International Conference on Inverse Problems in Engineering: Theory and Practice, ASME, 1993, pp. 109–116.

[2] J.H. Lin, C.K. Chen, Y.T. Yang, Inverse method for estimating thermal conductivity in one-dimensional heat conduction problems, J. Thermophys. Heat Transfer 15 (2001) 34–41.

[3] C.H. Huang, S.C. Chin, A two-dimensional inverse problem in imaging the thermal conductivity of a non-homogeneous medium, Int. J. Heat Mass Transfer 43 (2000) 4061–4071.

[4] O.M. Alifanov, Inverse Heat Transfer Problems, Springer-Verlag, Berlin, New York, 1994.

[5] M. Tadi, Evaluation of a two-dimensional conductivity function based on boundary measurements, J. Heat Transfer—Trans. ASME 122 (2000) 367–371.

[6] J.V. Beck, Inverse Heat Conduction: Ill-Posed Problems, Wiley, New York, 1985.

[7] S. Ashforth-Frost, V.N. Fontama, K. Jambunathan, S.L. Hartle, Role of neural networks in fluid mechanics and heat transfer, Proceedings of the 1995 IEEE Instrumentation and Measurement Technology Conference, vol. 6–9, IEEE, Piscataway, NJ, USA, 1995.

[8] G. Scalabrin, L. Piazza, Analysis of forced convection heat transfer to supercritical carbon dioxide inside tubes using neural networks, Int. J. Heat Mass Transfer 46 (2003) 1139–1154.

[9] G. Diaz, M. Sen, K.T. Yang, R.L. McClain, Dynamic prediction and control of heat exchangers using artificial neural networks, Int. J. Heat Mass Transfer 44 (2001) 1671–1679.

[10] K. Jambunathan, S.L. Hartle, S. Ashforth-Frost, V.N. Fontama, Evaluating convective heat transfer coefficients using neural networks, Int. J. Heat Mass Transfer 39 (1996) 2329–2332.

[11] I. Zbicinski, P. Strumillo, W. Kaminski, Hybrid neural model of thermal drying in a fluidized bed, Comput. Chem. Eng. 20 (1996) S695–S700.

[12] E. Alvarez, J.M. Correa, C. Riverol, J.M. Navaza, Model based in neural networks for the prediction of the mass transfer coefficients in bubble columns. Study in Newtonian and non-Newtonian fluids, Int. Commun. Heat Mass Transfer 27 (2000) 93–98.

[13] F. Larachi, L. Belfares, B.P.A. Grandjean, Prediction of liquid–solid wetting efficiency in trickle flow reactors, Int. Commun. Heat Mass Transfer 28 (2001) 595–603.

[14] E. Divo, A. Kassab, F. Rodriguez, Characterization of space dependent thermal conductivity with a BEM-based genetic algorithm, Numer. Heat Transfer Part A—Appl. 37 (2000) 845–875.

[15] J. Ghaboussi, D.A. Pecknold, M.F. Zhang, R.M. Haj-Ali, Autoprogressive training of neural network constitutive models, Int. J. Numer. Methods Eng. 42 (1998) 105–126.

[16] R.D. Reed, R.J. Marks, Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks, The MIT Press, Cambridge, MA, 1999.

[17] J. Ghaboussi, J.H. Garrett, X. Wu, Knowledge-Based Modeling of Material Behavior with Neural Networks, J. Eng. Mech.—ASCE 117 (1991) 132–153.

[18] D.E. Sidarta, J. Ghaboussi, Constitutive modeling of geomaterials from non-uniform material tests, Comput. Geotech. 22 (1998) 53–71.

[19] T. Furukawa, G. Yagawa, Implicit constitutive modelling for viscoplasticity using neural networks, Int. J. Numer. Methods Eng. 43 (1998) 195–219.

[20] Y.M.A. Hashash, S. Jung, J. Ghaboussi, Numerical implementation of a neural network based material model in finite element analysis, Int. J. Numer. Methods Eng. 59 (2004) 989–1005.

[21] P. Cardaliaguet, G. Euvrard, Approximation of a function and its derivative with a neural network, Neural Networks 5 (1992) 207–220.

[22] A. Joghataie, J. Ghaboussi, X. Wu, Learning and architecture determination through automatic node generation, Proceedings of the 1995 Artificial Neural Networks in Engineering, ANNIE'95, vol. 5, ASME, 1995, pp. 45–50.

[23] J.F. Stelzer, R. Welzel, Experiences in nonlinear-analysis of temperature-fields with finite-elements, Int. J. Numer. Methods Eng. 24 (1987) 59–73.