Expert Systems with Applications 38 (2011) 12531–12536

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Application of feedforward neural network in the study of dissociated gas flow along the porous wall

Vesna Ranković *, Slobodan Savić

Department for Applied Mechanics and Automatic Control, Faculty of Mechanical Engineering, University of Kragujevac, Sestre Janjić 6, 34000 Kragujevac, Serbia

ARTICLE INFO

Keywords:
Feedforward neural network
Gas flow
Porous wall
Velocity

ABSTRACT

This paper concerns the use of feedforward neural networks (FNN) for predicting the nondimensional velocity of the gas that flows along a porous wall. The numerical solution of partial differential equations that govern the fluid flow is applied for training and testing the FNN. The equations were solved using finite differences method by writing a FORTRAN code. The Levenberg–Marquardt algorithm is used to train the neural network. The optimal FNN architecture was determined. The FNN predicted values are in accordance with the values obtained by the finite difference method (FDM). The performance of the neural network model was assessed through the correlation coefficient (r), mean absolute error (MAE) and mean square error (MSE). The respective values of r, MAE and MSE for the testing data are 0.9999, 0.0025 and $1.9998 \cdot 10^{-5}$.

1. Introduction

Intelligent techniques such as fuzzy logic methods, neural networks and genetic programming have been used in most of research applications such as classifications, automatic control, prediction, signal processing, pattern recognition, etc. (Haykin, 1999). In recent years, these techniques have also been successfully applied in the area of fluid flow (El-Emam & Al-Rabeh, 2008; Hirschen & Schäfer, 2006; Shang, 2005).

One of successful neural networks applications is to solve partial differential equations (PDE) that govern fluid flow. The finite differences method (FDM), finite element method (FEM), boundary element method (BEM) and finite volume method (FVM) are popular and successful numerical methods for solution of partial differential equations (PDE) (Glowinski & Neittaanmaki, 2008). Reducing cost of studies and saving computational time, artificial neural networks represent efficient tools and useful alternative for solution of PDE. He, Reifb, and Unbehauenc (2000), Jianyu, Siwei, Yingjian, and Yaping (2003), Shirvany, Hayati, and Moradian (2009) investigated the application of neural networks in order to find solutions of PDE. The feedforward neural network and radial basis function networks are powerful architectures for interpolation in multidimensional space and they are universal function approximators (Hornik, Stinchcombe, & White, 1989; Park & Sandberg, 1991; Schilling, Carroll, & Al-Ajlouni, 2001). The neural network method is not a direct method to solve a differential equation because this method requires the equation system similar to the system of dynamics evolvement of ANN (Shang, 2005).

In the literature, neural networks have been used to predict fluid flow variables. Applications of neural networks are based on the training data from the experiment or the database generated by a computational fluid dynamics (CFD) analysis of the problem.

Tahavvor and Yaghoubi (2008) used artificial neural network (ANN) to determine natural convection heat transfer and fluid flow around a cooled horizontal circular cylinder for different Rayleigh numbers. They applied results obtained using finite volume technique for training and testing the ANN approach.

Mahmoud and Ben-Nakhi (2007) analyzed free laminar convection heat transfer in a partitioned enclosure. The CFD simulation software was applied to calculate the temperature, the pressure, and the horizontal and vertical components of the flow speed. They used results of the CFD for training and testing the neural networks.

Soft programming codes of ANN and Adaptive-Network-Based Fuzzy Inference System (ANFIS) were compared with the CFD code for a study of natural convection in triangular enclosures by Varol, Avci, Koca, and Oztop (2007). ANN and ANFIS were used to predict the natural convection thermal and flow variables.

Varol, Koca, Oztop, and Avci (2008) used ANFIS to estimate the flow and temperature distribution in a partially heated air-filled triangular enclosure. A database was generated using finite differences method by writing a FORTRAN code.

In this investigation the FNN with Levenberg–Marquardt algorithm was used to predict the nondimensional velocity of the dissociated gas that flows along a porous wall. Data were generated using finite differences method by writing a FORTRAN code

* Corresponding author. Tel.: +381 34 335 990; fax: +381 34 333 192.
E-mail address: vesnar@kg.ac.rs (V. Ranković).

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.04.039

(Obrović, Nikodijević, & Savić, 2009). The paper is organized as follows: The solution of the problem is presented in Section 2. The basic structure of the FNN and training algorithm is described in Section 3. The results of the simulation are shown in Section 4. Finally, in Section 5 concluding remarks are presented.

2. Description of the problem

The dissociated gas (air) flow in the boundary layer on bodies of revolution is investigated. The contour of the body within the fluid (Fig. 1) is porous.

Thermo-chemical equilibrium is assumed to be established in the whole area of the boundary layer. Therefore, a complete equation system for this case of axisymmetrical gas flow in the boundary layer, with the corresponding boundary conditions (Schlichting, 1974), is:

$$\frac{\partial}{\partial x}(\rho u r)+\frac{\partial}{\partial y}(\rho v r)=0, \tag{1}$$

$$\rho u\frac{\partial u}{\partial x}+\rho v\frac{\partial u}{\partial y}=\rho_e u_e\frac{du_e}{dx}+\frac{\partial}{\partial y}\left(\mu\frac{\partial u}{\partial y}\right), \tag{2}$$

$$\rho u\frac{\partial h}{\partial x}+\rho v\frac{\partial h}{\partial y}=-u\rho_e u_e\frac{du_e}{dx}+\mu\left(\frac{\partial u}{\partial y}\right)^2+\frac{\partial}{\partial y}\left[\frac{\mu}{Pr}(1+l)\frac{\partial h}{\partial y}\right], \tag{3}$$

$$\begin{aligned} &u=0, \quad v=v_w(x), \quad h=h_w \quad \text{for } y=0,\\ &u\to u_e(x), \qquad\quad h\to h_e(x) \quad \text{for } y\to\infty. \end{aligned} \tag{4}$$

The function $l(p,h)$ for the equilibrium two-componential mixture depends on Lewis number Le and on the enthalpy of the atomic h_A and molecular h_M components of the equilibrium dissociated gas (air). This function is determined with the expression (Obrović et al., 2009):

$$l(p,h)=(\mathrm{Le}-1)(h_A-h_M)\left(\frac{\partial C_A}{\partial h}\right)_p. \tag{5}$$

The usual notation is used. Thus, $u(x,y)$ denotes the longitudinal projection of the velocity in the boundary layer, $v(x,y)$ – transversal projection, ρ – density of the ideally dissociated gas, μ – dynamic viscosity, h – enthalpy. Here, x and y are longitudinal and transversal coordinates, respectively. Prandtl and Lewis numbers are defined with the expressions: $\mathrm{Pr}=\mu c_p/\lambda$, $\mathrm{Le}=\rho c_p D/\lambda$ in which λ – stands for the thermal conductivity coefficient, c_p – specific heat of the dissociated gas at constant pressure and D – atomic component diffusion coefficient. The radius of the cross-section of the body of revolution, which is normal to the axis of revolution, is denoted with $r(x)$. The contour of the body, which figures in the continuity equation is given by the function $r(x)$. The subscript "e" denotes the physical quantities at the outer edge of the boundary layer, and the subscript "w" stands for the quantities at the wall of the body within the fluid. Here, $v_w(x)$ denotes the given velocity of the gas that flows through the solid porous wall ($v_w>0$ or $v_w<0$).

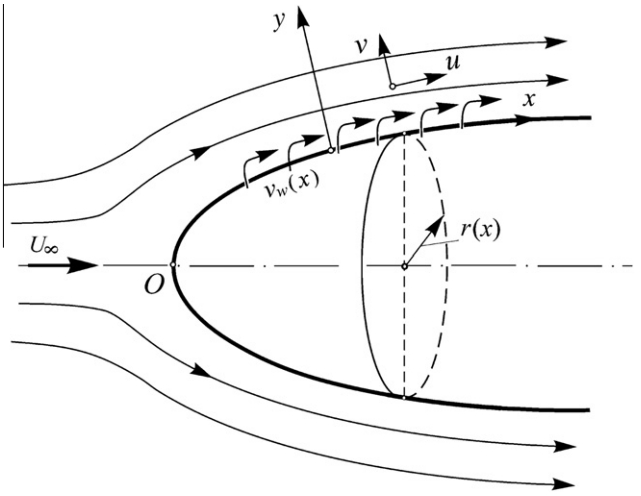

Fig. 1. Gas flow around a body of revolution.

When the governing equation system is transformed (1)–(4) and when the change (Saljnikov & Dallmann, 1989):

$$\frac{u}{u_e}=\frac{\partial\Phi}{\partial\eta}=\varphi=\varphi(\eta,\kappa,f,\Lambda) \tag{6}$$

that decreases the order of the differential equations is performed (Obrović et al., 2009), a generalized equation system with four independent variables, η, κ, f and Λ, is obtained:

$$\begin{aligned} &\frac{\partial}{\partial\eta}\left(Q\frac{\partial\varphi}{\partial\eta}\right)+\frac{aB^2+(2-b)f}{2B^2}\Phi\frac{\partial\varphi}{\partial\eta}+\frac{f}{B^2}\left(\frac{\bar h}{1-\kappa}-\varphi^2\right)+\frac{\Lambda}{B}\frac{\partial\varphi}{\partial\eta}\\ &\quad=\frac{F_{ot}f}{B^2}\left(\varphi\frac{\partial\varphi}{\partial f}-\frac{\partial\Phi}{\partial f}\frac{\partial\varphi}{\partial\eta}\right), \end{aligned} \tag{7}$$

$$\begin{aligned} &\frac{\partial}{\partial\eta}\left(\frac{Q}{Pr}\frac{\partial\bar h}{\partial\eta}\right)+\frac{aB^2+(2-b)f}{2B^2}\Phi\frac{\partial\bar h}{\partial\eta}-\frac{2\kappa f}{B^2}\varphi\left(\frac{\bar h}{1-\kappa}-\varphi^2\right)\\ &\quad+2\kappa Q\left(\frac{\partial\varphi}{\partial\eta}\right)^2+\frac{\Lambda}{B}\frac{\partial\bar h}{\partial\eta}\\ &\quad=\frac{F_{ot}f}{B^2}\left(\varphi\frac{\partial\bar h}{\partial f}-\frac{\partial\Phi}{\partial f}\frac{\partial\bar h}{\partial\eta}\right), \end{aligned} \tag{8}$$

$$\begin{aligned} &\Phi=0, \quad \varphi=0, \quad \bar h=\bar h_w=const. \quad \text{for } \eta=0,\\ &\varphi\to 1, \qquad\quad \bar h\to\bar h_e=1-\kappa \quad \text{for } \eta\to\infty. \end{aligned} \tag{9}$$

Here the following notation is used: u/u_e – nondimensional velocity, Φ – nondimensional stream function, η – nondimensional transversal coordinate, κ – compressibility parameter, f – form parameter, Λ – porosity parameter, $\bar h$ – nondimensional enthalpy, Q – nondimensional function, B – characteristics of the boundary layer, F_{ot} – characteristic boundary layer function, and a, b – constants.

3. FNN and training algorithm

Feedforward neural networks are composed of simple elements called *neurons*. The basic structure of the FNN, depicted in Fig. 2, consists of one or more hidden layers and an output layer. Suppose the total number of hidden layers is $m-1$. Inputs into the network are represented by $x_1,x_2,\ldots,x_R$. The network outputs are represented by $y_1,y_2,\ldots,y_{z_m}$. Each neuron i in the lth layer (except in the input layer) is connected with all the neurons of the $(l-1)$th layer. The number of neurons in the hidden layer l is z_l and the transfer function for the layer l is f_l. Here, $\omega_{i,j(l)}$ represents the weight of the link between the ith neuron of the lth hidden layer and jth neuron of the $l-1$th hidden layer and $b_{i(l)}$ is the biased weight for the ith hidden neuron of the lth hidden layer. Fig. 3 shows the ith neuron of the lth hidden layer.

The output of ith neuron of the lth hidden layer can be expressed as:

$$v_{i(l)}=f_l(n_{i(l)}), \tag{10}$$

where

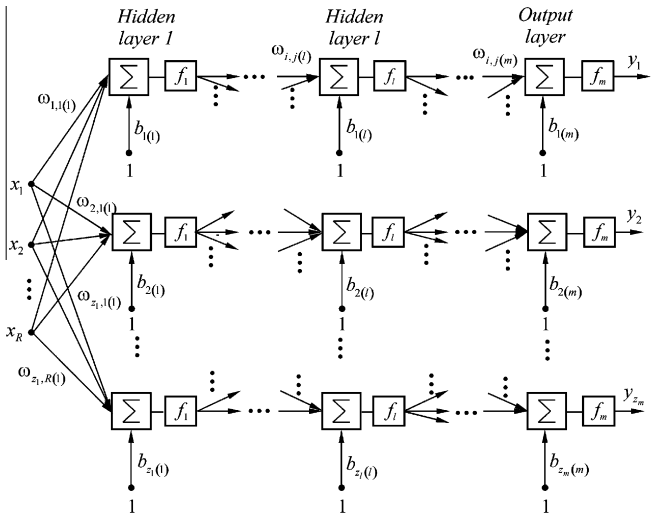

Fig. 2. Basic structure of FNN.

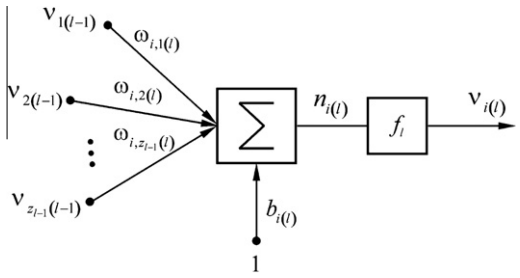

Fig. 3. The *i*th neuron of the *l*th hidden layer.

$$n_{i(l)} = \sum_{j=1}^{z_{l-1}} \omega_{i,j(l)}\, v_{j(l-1)} + b_{i(l)}. \tag{11}$$

The linear and sigmoid are the most common used transfer functions in the construction of artificial neural networks.

The linear function has form:

$$f_l(n_{i(l)}) = n_{i(l)}. \tag{12}$$

An example of the sigmoid is the logistic function, defined by:

$$f_l(n_{i(l)}) = \frac{1}{1+e^{-n_{i(l)}}}. \tag{13}$$

The hyperbolic tangent function can also be used as sigmoid function (Haykin, 1999):

$$f_l(n_{i(l)}) = \frac{1-e^{-n_{i(l)}}}{1+e^{-n_{i(l)}}}. \tag{14}$$

The outputs of the FNN can be computed as:

$$v_{i(m)} = y_i = f_m(n_{i(m)}), \quad i = 1,2,\ldots,z_m, \tag{15}$$

where

$$n_{i(m)} = \sum_{j=1}^{z_{m-1}} \omega_{i,j(m)}\, v_{j(m-1)} + b_{i(m)}. \tag{16}$$

The objective of the training is to find a set of weights and biases that minimize the error between the neural network predictions and the desired outputs. There are different learning algorithms. The back-propagation algorithm (Rumelhart, Hinton, & Williams, 1986) has been the most commonly used training algorithm. The basic algorithm is a gradient descent method in which the network weights and biases are moved along the negative performance function. It has problems of local minima and slow convergence.

Table 1
Correlation coefficient for the training, validation and test sets.

ANN-structure	3-27-1	3-29-1	3-31-1
Training	0.9952	0.9999	0.9887
Validation	0.9929	0.9999	0.9827
Test	0.9944	0.9999	0.9882

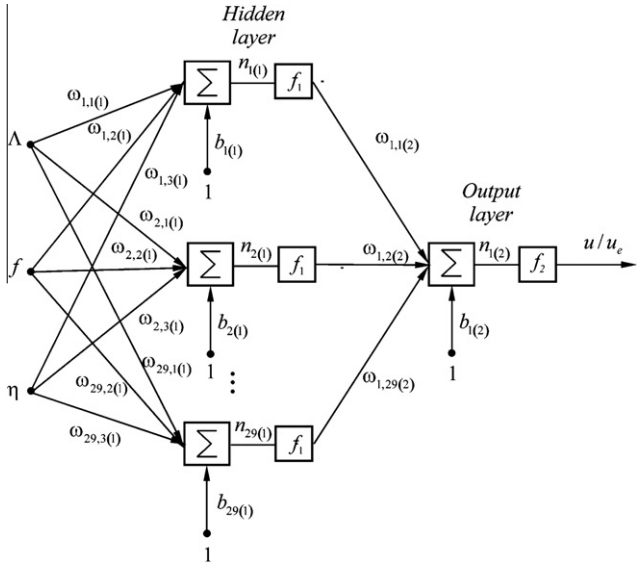

Fig. 4. FNN model for prediction of u/u_e values.

In the literature, a number of variations of the standard algorithm have been developed. The Gauss–Newton method (Zhou & Si, 1998), the quasi-Newton methods (Beigi & Li, 1993), the Marquardt method (Hagan & Menhaj, 1994), the Levenberg–Marquardt method (Lera & Pinzolas, 2002) are used for the FNN training.

Suppose the training data consists of N sample pairs. The performance function is defined as:

$$E(\omega) = \frac{1}{2}\sum_{k=1}^{N}\sum_{i=1}^{z_m}\left(y_i^{(k)} - t_i^{(k)}\right)^2 = \frac{1}{2}\sum_{k=1}^{N}\sum_{i=1}^{z_m}\left(e_i^{(k)}\right)^2, \tag{17}$$

where $y_i^{(k)}$ and $t_i^{(k)}$ denote the i*th* output and desired output of the neural network when the inputs presented to the network are $x_1^{(k)}, x_2^{(k)}, \ldots, x_R^{(k)}$ and ω is an n-element vector that contains all weights and biases of the neural network and can be written as:

$$\begin{aligned}\omega = [&\omega_{1,1(1)}, \omega_{1,2(1)}, \ldots, \omega_{z_1,R(1)}, b_{1(1)}, b_{2(1)}, \ldots, b_{z_1(1)}, \ldots,\\ &\omega_{1,1(l)}, \omega_{1,2(l)}, \ldots, \omega_{z_l,z_{l-1}(l)}, b_{1(l)}, b_{2(l)}, \ldots, b_{z_l(l)}, \ldots,\\ &\omega_{1,1(m)}, \omega_{1,2(m)}, \ldots, \omega_{z_m,z_{m-1}(m)}, b_{1(m)}, b_{2(m)}, \ldots, b_{z_m(m)}]^T\end{aligned}. \tag{18}$$

Eq. (17) can be written as:

$$E(\omega) = \frac{1}{2}e^T e, \tag{19}$$

where

$$e = \left[e_1^{(1)} e_2^{(1)} \ldots e_{z_m}^{(1)} e_1^{(2)} e_2^{(2)} \ldots e_{z_m}^{(2)} \ldots e_1^{(N)} e_2^{(N)} \ldots e_{z_m}^{(N)}\right]^T. \tag{20}$$

With the Levenberg–Marquardt's method, the increment $\Delta\omega$, by minimization of E with respect to the parameter vector ω, is:

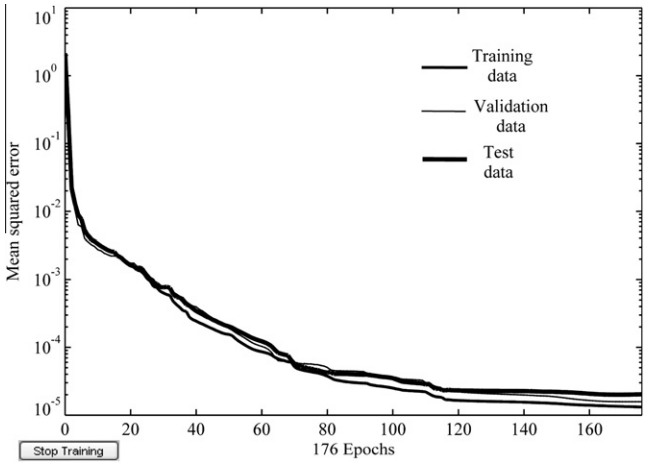

Fig. 5. MSE error for training, validation, and testing sets.

$$\Delta\omega = -(J^T J + \mu I)^{-1} \cdot J^T e, \tag{21}$$

where J is the Jacobian matrix that contains first derivatives of the network errors with respect to the weights and biases, $\boldsymbol{I}$ is the identity matrix and μ is an adaptive factor. When the scalar μ is zero, this is just Newton's method. When μ is large, this becomes gradient descent with a small step size.

The Jacobian matrix is calculated as:

$$J(\omega) = \begin{bmatrix} \frac{\partial e_1^{(1)}}{\partial \omega_{1,1(1)}} & \cdots & \frac{\partial e_1^{(1)}}{\partial b_{z_m(m)}} \\ \vdots & \ddots & \vdots \\ \frac{\partial e_{z_m}^{(N)}}{\partial \omega_{1,1(1)}} & \cdots & \frac{\partial e_{z_m}^{(N)}}{\partial b_{z_m(m)}} \end{bmatrix}. \tag{22}$$

In this paper, the FNN is used with one hidden layer and one output. The performance function can be expressed as:

Table 2
Performance parameters of the artificial neural network models for computation of the u/u_e.

FNN-structure		r	MAE	MSE
3-29-1	Training	0.9999	0.0022	$1.3399 \cdot 10^{-5}$
	Validation	0.9999	0.0024	$1.5745 \cdot 10^{-5}$
	Test	0.9999	0.0025	$1.9998 \cdot 10^{-5}$
	Training + validation + test	0.9999	0.0023	$1.505 \cdot 10^{-5}$

$$E(\omega) = \frac{1}{2}\sum_{k=1}^{N}\left(y^{(k)} - t^{(k)}\right)^2 = \frac{1}{2}\sum_{k=1}^{N}(e^{(k)})^2 = \frac{1}{2}e^T e, \tag{23}$$

where

$$\omega = [\omega_{1,1(1)}, \omega_{1,2(1)}, \ldots, \omega_{z,R(1)}, b_{1(1)}, b_{2(1)}, \ldots, b_{z(1)}, \omega_{1,1(2)}, \omega_{1,2(2)}, \ldots, \omega_{1,z(2)}, b_{1(2)}]^T \tag{24}$$

and

$$e = [e^{(1)} e^{(2)} \cdots e^{(N)}]. \tag{25}$$

The number of neurons in the hidden layer is z. The weights and biases are calculated using (21). The Jacobian matrix is defined as:

$$J(\omega) = \begin{bmatrix} \frac{\partial e^{(1)}}{\partial \omega_{1,1(1)}} & \cdots & \frac{\partial e^{(1)}}{\partial b_{1(2)}} \\ \vdots & \ddots & \vdots \\ \frac{\partial e^{(N)}}{\partial \omega_{1,1(1)}} & \cdots & \frac{\partial e^{(N)}}{\partial b_{1(2)}} \end{bmatrix}. \tag{26}$$

4. Simulation results

In this study, the FNN is used to predict the nondimensional velocity (u/u_e). The inputs of the NN model are the porosity parameter (Λ), the form parameter (f) and the nondimensional transver-

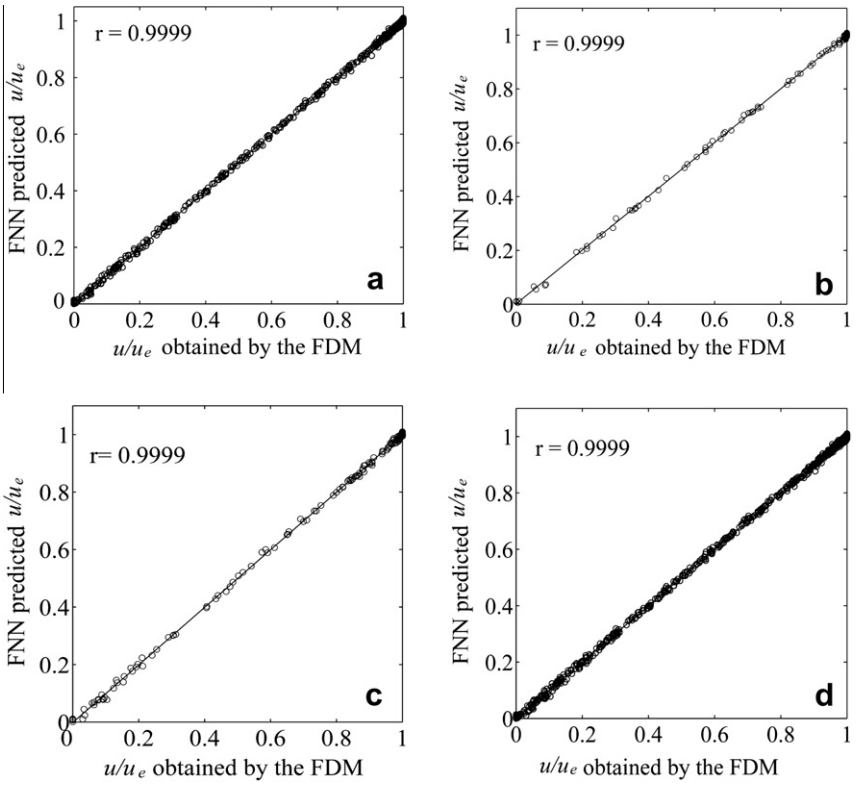

Fig. 6. Comparisons of the values obtained by the FDM and ANN-modeled values of u/u_e in (a) training set, (b) validation set, (c) test and (d) training + validation + test set.

sal coordinate (η). The MATLAB Neural Network Toolbox is applied for the implementation of the neural network. Data were generated using the finite differences method by writing a FORTRAN code for $\Lambda = [-0.1, -0.05, 0, 0.05, 0.1]$, $f = [-0.16, -0.12, -0.08, -0.04, 0, 0.04, 0.08, 0.12, 0.16]$, $\eta = [0, 0.4, 0.8, 1.2, \ldots, 19.6, 20]$ and $\kappa = 0.08$. Eqs. (7) and (8) are solved for the following values of the parameters and coefficients: Pr = 0.712; $a = 0.4408$; $b = 5.7140$ (Saljnikov & Dallmann, 1989). For the characteristic functions B and F_{ot} at a zero iteration, the following values are accepted: $B^0_{K+1} = 0.449$ and $F^0_{ot.K+1} = 0.4411$. They were also used in the investigations by Saljnikov and Dallmann (1989).

The data ($5 \times 9 \times 51 = 2295$) were divided into training, validation and test subsets. In the training process of the FNN, 1495 samples were used. 350 data were taken for the validation set. The ANN model was tested using 450 selected data.

The Pearson correlation coefficient is one of the most commonly used performance to evaluate the closeness of fit of network architecture. The correlation coefficient is defined as the degree of correlation between the values obtained from the CFD code and the predicted values:

$$r = \frac{\sum_{k=1}^{N_o}(y^{(k)} - \bar{y})(t^{(k)} - \bar{t})}{\sqrt{\sum_{k=1}^{N_o}(y^{(k)} - \bar{y})^2 \sum_{k=1}^{N_o}(t^{(k)} - \bar{t})^2}}, \tag{27}$$

where $y^{(k)}$ and $t^{(k)}$ denote the network output and the value obtained from the CFD code from the kth element; $\bar{y}$ and $\bar{t}$ denote their average respectively, and N_o represents the number of observations.

Another performance measure is the mean absolute error (MAE). It is defined as:

$$\mathrm{MAE} = \frac{1}{N_o}\sum_{k=1}^{N_o}\left|y_i^{(k)} - t_i^{(k)}\right|. \tag{28}$$

The mean square error (MSE) is calculated as:

$$\mathrm{MSE} = \frac{1}{N_o}\sum_{k=1}^{N_o}\left(y_i^{(k)} - t_i^{(k)}\right)^2. \tag{29}$$

Smaller MAE and MSE values ensure better performance.

Different FNN models were constructed and tested in order to determine the optimum number of neurons in the hidden layer and transfer functions. The two-layer network with a log-sigmoid transfer function at the hidden layer and a linear transfer function at the output layer were used.

The optimal network size was the one which resulted in a maximum correlation coefficient for the training, validation and test sets, Table 1. Based on Table 1, it was concluded that the optimal

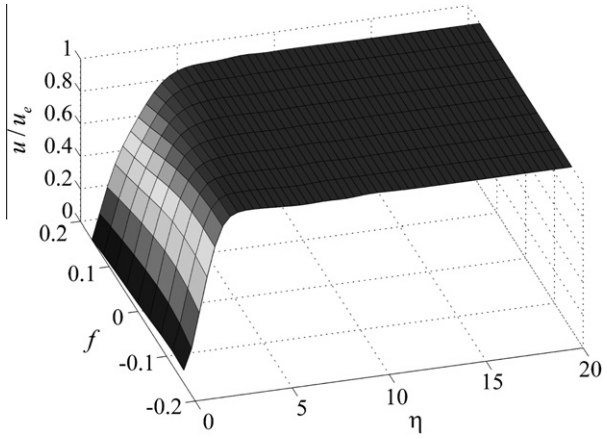

Fig. 7. u/u_e obtained by the FNN for $\Lambda = 0.7$.

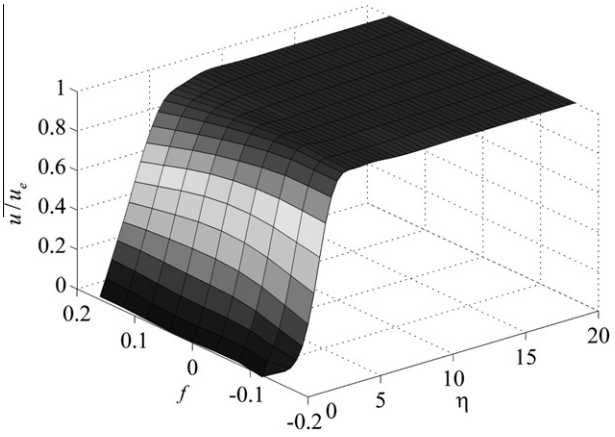

Fig. 8. u/u_e obtained by the FNN for $\Lambda = 0.15$.

number of hidden neurons is 29. The FNN model for prediction of u/u_e values is shown in Fig. 4. The training stopped after 176 epochs because the validation error started to increase (Fig. 5). The performance parameters of the artificial neural network models for computation of the u/u_e are given in Table 2.

Comparisons of the values obtained by the FDM and ANN-modeled values of u/u_e for the training, validation, test and training + validation + test data sets are shown in Fig. 6a, b, c, and d, respectively.

The values of u/u_e for $\Lambda = 0.7$ and $\Lambda = 0.15$, computed using the FNN model are plotted in Fig. 7 and Fig. 8, respectively. The values of the porosity parameter are not included in the training data set.

5. Conclusions

In the presented study, the ability of FNN model to predict the nondimensional velocity of the gas that flows along a porous wall is investigated. The two-layer FNN with Levenberg–Marquardt learning was constructed. A database was generated using finite difference method by writing a FORTRAN code. The ANN structure was designed and trained using the MATLAB Neural Network Toolbox.

The results of the simulations presented in this paper show that the application of the FNN to estimate the nondimensional velocity gives satisfactory results. The performance of the FNN network was tested using correlation coefficients, the mean absolute error and the mean square error. In the same way, the proposed approach can be applied to predict the nondimensional enthalpy and other characteristics of the boundary layer.

Soft programming methods such as FNN can be used as alternative numerical methodologies, thus saving computational time and reducing cost of studies.

Acknowledgements

This paper is the result of investigations carried out within the scientific projects ON144002 and ON144022 supported by the Ministry of Science and Environmental Protection of the Republic of Serbia.

References

Beigi, H. S. M., & Li, C. J. (1993). Learning algorithms for neural networks, based on quasi-Newton methods with self-scaling. *ASME Transactions, Journal of Dynamic Systems, Measurement, and Control, 115*(1), 38–43.

El-Emam, N. N., & Al-Rabeh, R. H. (2008). An intelligent computing technique for fluid flow problems using hybrid adaptive neural network and genetic algorithm. *Applied Soft Computing Journal*. doi:10.1016/j.asoc.2009.12.009.

Glowinski, R., & Neittaanmaki, P. (2008). *Partial differential equations: Modeling and numerical simulation*. Springer.

Hagan, M. T., & Menhaj, M. B. (1994). Training feedforward networks with the Marquardt algorithm. *IEEE Transactions on Neural Networks, 5*(6), 989–993.

Haykin, S. (1999). *Neural networks: A comprehensive foundation*. Prentice Hall.

He, S., Reifb, K., & Unbehauenc, R. (2000). Multilayer neural networks for solving a class of partial differential equations. *Neural Networks, 13*, 385–396.

Hirschen, K., & Schäfer, M. (2006). Bayesian regularization neural networks for optimizing fluid flow processes. *Computer Methods in Applied Mechanics and Engineering, 195*, 481–500.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks, 2*, 359–366.

Jianyu, L., Siwei, L., Yingjian, Q., & Yaping, H. (2003). Numerical solution of elliptic partial differential equation using radial basis function neural networks. *Neural Networks, 16*, 729–734.

Lera, G., & Pinzolas, M. (2002). Neighborhood based Levenberg–Marquardt algorithm for neural network training. *IEEE Transactions on Neural Networks, 13*(5), 1200–1203.

Mahmoud, M. A., & Ben-Nakhi, A. E. (2007). Neural networks analysis of free laminar convection heat transfer in a partitioned enclosure. *Communications in Nonlinear Science and Numerical Simulation, 12*, 1265–1276.

Obrović, B., Nikodijević, D., & Savić, S. (2009). Boundary layer of dissociated gas on bodies of revolution of a porous contour. *Strojniski vestnik – Journal of Mechanical Engineering, 55*(4), 244–253.

Park, J., & Sandberg, I. W. (1991). Universal approximation using radial basis function networks. *Neural Computing, 3*(2), 246–257.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323*, 533–536.

Saljnikov, V. N., & Dallmann, U. (1989). Verallgemeinerte Ähnlichkeitslösungen für dreidimensionale, laminare, stationäre, kompressible Grenzschichtströmungen an schiebenden profilierten Zylindern. Institut für Theoretische Strömungsmechanik DLR-FB 89-34, Göttingen.

Schlichting, H. (1974). *Grenzschicht-theorie*. Karlsruhe: Verlag G. Braun.

Schilling, R. J., Carroll, J. J., & Al-Ajlouni, A. F. (2001). Approximation of nonlinear systems using radial basis function neural networks. *IEEE Transactions on Neural Networks, 12*(1), 1–14.

Shang, Z. (2005). Application of artificial intelligence CFD based on neural network in vapor–water two-phase flow. *Engineering Applications of Artificial Intelligence, 18*, 663–671.

Shirvany, Y., Hayati, M., & Moradian, R. (2009). Multilayer perceptron neural networks with novel unsupervised training method for numerical solution of the partial differential equations. *Applied Soft Computing, 9*, 20–29.

Tahavvor, A. R., & Yaghoubi, M. (2008). Natural cooling of horizontal cylinder using Artificial Neural Network (ANN). *International Communications in Heat and Mass Transfer, 35*, 1196–1203.

Varol, Y., Avci, E., Koca, A., & Oztop, H. F. (2007). Prediction of flow fields and temperature distributions due to natural convection in a triangular enclosure using Adaptive-Network-Based Fuzzy Inference System (ANFIS) and Artificial Neural Network (ANN). *International Communications in Heat and Mass Transfer, 34*, 887–896.

Varol, Y., Koca, A., Oztop, H. F., & Avci, E. (2008). Analysis of adaptive-network-based fuzzy inference system (ANFIS) to estimate buoyancy-induced flow field in partially heated triangular enclosures. *Expert Systems with Applications, 35*, 1989–1997.

Zhou, G., & Si, J. (1998). Advanced neural-network training algorithm with reduced complexity based on Jacobian deficiency. *IEEE Transactions on Neural Networks, 9*(3), 448–453.