



PERGAMON

Available at
WWW.MATHEMATICSWEB.ORG
POWERED BY SCIENCE @ DIRECT®

Journal of the Franklin Institute 340 (2003) 307–325

Journal
of The
Franklin Institute

www.elsevier.com/locate/jfranklin

The solutions of vibration control problems using artificial neural networks

Hasan Alli^{a,*}, Ayşegül Uçar^b, Yakup Demir^b

^a *Department of Mechanical Engineering, Faculty of Engineering, Firat University, Elazig, Turkey*

^b *Department of Electrical & Electronics Engineering, Faculty of Engineering,
Firat University, Elazig, Turkey*

Received in revised form 26 March 2003; accepted 28 April 2003

Abstract

This paper introduces an alternative method artificial neural networks (ANN) used to obtain numerical solutions of mathematical models of dynamic systems, represented by ordinary differential equations (ODEs) and partial differential equations (PDEs). The proposed trial solution of differential equations (DEs) consists of two parts: The initial and boundary conditions (BCs) should be satisfied by the first part. However, the second part is not affected from initial and BCs, but it only tries to satisfy DE. This part involves a feedforward ANN containing adjustable parameters (weight and bias). The proposed solution satisfying boundary and initial condition uses a feedforward ANN with one hidden layer varying the neuron number in the hidden layer according to complexity of the considered problem. The ANN having appropriate architecture has been trained with backpropagation algorithm using an adaptive learning rate to satisfy DE. Moreover, we have, first, developed the general formula for the numerical solutions of n th-order initial-value problems by using ANN.

For numerical applications, the ODEs that are the mathematical models of linear and non-linear mass-damper-spring systems and the second- and fourth-order PDEs that are the mathematical models of the control of longitudinal vibrations of rods and lateral vibrations of beams have been considered. Finally, the responses of the controlled and non-controlled systems have been obtained. The obtained results have been graphically presented and some conclusion remarks are given.

© 2003 The Franklin Institute. Published by Elsevier Science Ltd. All rights reserved.

Keywords: Vibration control problems; Artificial neural networks; Backpropagation algorithm

*Corresponding author.

E-mail addresses: halli@firat.edu.tr (H. Alli), agulucar@firat.edu.tr (A. Uçar), ydemir@firat.edu.tr (Y. Demir).

1. Introduction

Many problems in science and engineering are represented by a set of differential equations (DEs) through the process of mathematical modeling. Analytic solutions of the established mathematical models based on physical laws may not be always possible. In addition, analytical methods are generally inadequate to obtain closed form solutions of the considered problems. The most used numerical methods developed for solving DEs include finite difference method, finite element method (FEM), finite volume method and boundary element method (BEM) [1–12]. Enormous progress in the field of numerical solutions of ordinary differential equations (ODEs) has been developed in the last three decades. However, these methods are still based on some discretization of the domain of analysis into a number of finite elements. Furthermore, remeshing is additional difficulty for problems with moving or unknown boundaries. Meshing and automatic discretization are always desired, however, it is rarely available. In addition, the mesh generation may take time and require lot of calculations.

Unlike FEM, neural networks can be considered as approximation schemes where the input data for the design of a network consist of only a set of unstructured discrete data points. Different methods were developed for solving ODEs and partial differential equations (PDEs) using feedforward artificial neural networks (ANNs) [13–18]. One of the developed methods shows the application of ANN for the first- and second-order PDEs [13]. Besides, the same study also presents how function approximation method can be used with non-linear BEM for reducing to need the discretization of domain. In the other study, He et al. [14] introduce a new method to solve a class of PDEs using multilayer neural networks for input-to-state linearizable or approximate linearizable systems. By using this method, the solution of the DE, together with the Lie derivatives, yields a change of coordinates. Then, a feedback control law has been suggested to obtain the desired behaviors of the systems.

Karr et al. [15] have proposed another approach to solve inverse initial-value and boundary value problems transformed into a non-linear optimization problem. Then genetic algorithm has been used to solve the obtained optimization problem. In the other study [16], the numerical solutions of linear ODEs and elliptic PDEs have been obtained by using multiquadric radial basis function networks (RBFNs). This method has mesh-free procedure. The mentioned methods with different networks architectures have given good results for specific class of ODEs and PDEs. However, none of them has produced general closed form solution. For the closed form solution of DEs, Lagaris et al. [17] have, first, proposed and given comparisons with solutions obtained from Galerkin FEM to present performance of their method. Then, the same authors [18] have developed their method using ANNs with RBFNs. The suggested method has given the accurate results for two- and three-dimensional PDEs.

Recent developments of linear and non-linear control theory cause the development of designing powerful non-linear control systems. However, numerical solutions of non-linear control problems and vibrations of flexible systems have problems with numerical instability. Hence, we present a method to solve both

ODEs and PDEs, which are the mathematical model of vibration control of flexible mechanical systems. The proposed closed form solutions are differentiable and easy to implement for many scientific areas. This method uses feedforward neural networks with one hidden layer in which varies the neuron number as a basic approximation element, whose weights and biases are adjusted to minimize a relevant error function. Optimization techniques requiring the calculation of the gradient of the selected error function w.r.t. the network parameter are used to train the network. The proposed solution consists of the sum of two functions. The first function should satisfy the initial conditions (ICs) and boundary conditions (BCs) and it contains no adjustable parameters. The second one contains a feedforward ANN with one hidden layer, varying the neuron number, to be trained for satisfying DE. Because of that ANNs with one hidden layer in which varies the neuron number are universal approximators, considering this type of network architecture as a candidate model for solving DEs is very useful and productive.

The use of the solution of DEs by using ANNs has many attractive characteristics as follows [17,18]:

- The main advantage of the solutions of DEs via ANNs unlike the most other techniques offering discrete solution is differentiable and the closed form easily used in any subsequent calculation.
- The solution via ANNs presents very good generalization properties.
- Less number of parameters are required than the other solution techniques. Therefore, it requires very low memory space.
- The proposed solution method is capable of applying both ODEs and PDEs.
- Realizing this method is possible by using neuron processors. Therefore, the solution of real-time DE problems existing in many engineering applications is easy to obtain. It can also be implemented on parallel structures.

The remainder of the paper is organized as follows: In Section 2, we describe the general formulation of the proposed approach. Meanwhile, we introduce the backpropagation algorithm and the extended backpropagation algorithm. In Section 3, we proposed solutions for considering different ODEs and PDEs. For numerical applications, we consider linear and non-linear mass-damper-spring system whose mathematical models are represented by ODEs and the control of longitudinal vibration of rods and lateral vibration of beams whose mathematical models are represented by the second- and fourth-order PDEs, respectively, in Section 4. The obtained results are also graphically presented and some conclusive remarks are given. All computer programs developed in this study have been performed by using the Mathworks MATLAB version 5.2. Finally, Section 5 concludes the paper.

2. Introducing the method and extended backpropagation algorithm

The suggested method can be described in terms of the following general DE form:

$$G(\vec{x}, \psi(\vec{x}), \nabla\psi(\vec{x}), \nabla^2\psi(\vec{x}), \dots, \nabla^M\psi(\vec{x})) = 0, \quad \vec{x} \in D, \quad (1)$$

where $\vec{x} = (x_1, \dots, x_n) \in R^n$, $D \subset R^n$ represents the definition domain, M represents the order of DE, and $\psi(x)$ is the solution to be obtained.

To solve Eq. (1), the method is arranged based on the collocation method assuming a discretization of domain D and its boundary S into set points \hat{D} and \hat{S} , respectively. Then, Eq. (1) can be rewritten as

$$G(\vec{x}_i, \psi(\vec{x}_i), \nabla \psi(\vec{x}_i), \nabla^2 \psi(\vec{x}_i), \dots, \nabla^M \psi(\vec{x}_i)) = 0, \quad \forall \vec{x}_i \in \hat{D} \quad (2)$$

subject to the constraints existed by the BCs and ICs [3]. If $\psi_T(\vec{x}, \vec{c})$ represents a trial solution with adjustable parameter \vec{c} , the problem is transformed into Eq. (3) having the constraints imposed by the BCs and ICs

$$\min_{\vec{c}} \sum_{x_i \in \hat{D}} (G(\vec{x}_i, \psi_T(\vec{x}_i, \vec{c}), \nabla \psi_T(\vec{x}_i, \vec{c}), \nabla^2 \psi_T(\vec{x}_i, \vec{c}), \dots, \nabla^M \psi_T(\vec{x}_i, \vec{c}))^2. \quad (3)$$

In this method, the trial solution ψ_T uses a feedforward ANN and the parameters \vec{c} correspond to the weights and biases of ANN. $\psi_T(\vec{x})$ can be chosen as a sum of two functions in the form of Eq. (4), which should satisfy the BCs, and ICs [17]

$$\psi_T(\vec{x}) = A(\vec{x}) + F(\vec{x}, \Phi(\vec{x}, \vec{c})), \quad (4)$$

where $\Phi(\vec{x}, \vec{c})$ is a single-output feedforward neural network with parameters \vec{c} and n input neurons fed with the input vector \vec{x} . The first function $A(\vec{x})$ contains no adjustable parameters and only satisfies the BCs and ICs. However, the second function F should not satisfy the BCs and ICs because $\psi_T(\vec{x})$ is already constructed to satisfy them. F uses an ANN whose weights and biases are to be adjusted to solve the constructed minimization problem. ANN constructed for this problem consists of one hidden layer whose neuron number varies. After that, the original constrained optimization problem is transformed into an unconstrained one because the proposed objective function to be minimized contains both the ICs/BCs and the DE.

In this paper, the backpropagation algorithm that is an optimization technique designed to minimize an objective function [19] is used for training ANN and extended. The most commonly used objective function is the squared error, which is defined as

$$\varepsilon^2 = [T_q - \Phi_q]^2, \quad (5)$$

where, T_q is the target output value and Φ_q is the network output.

To minimize Eq. (3), it is rearranged as Eq. (5). Fig. 1 shows the architecture of the feedforward ANN which has n input neurons, one hidden layer with m hidden neurons and one output neuron. Fig. 2 also shows the network syntax for backpropagation. In this study, we used a feedforward ANN having logistic activation function in the hidden layer and linear activation function in the output layer.

For a given input vector, the output of the network is computed as $\Phi_q = \sum_{i=1}^m v_i \Phi_p(I_{pi})$, where $I_{pi} = \sum_{j=1}^n w_{ij} x_j + u_i$, w_{ij} represents the weight from the input unit j to the hidden unit i , v_i represents the weight from the hidden unit i to the output, u_i represents the bias of hidden unit i , and $\Phi_p(I_p)$ is the logistic activation function.

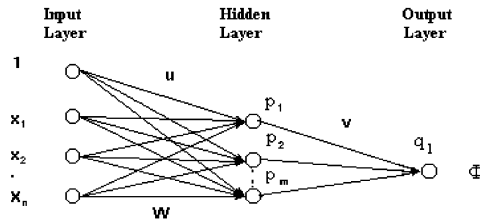


Fig. 1. Feedforward ANN architecture.

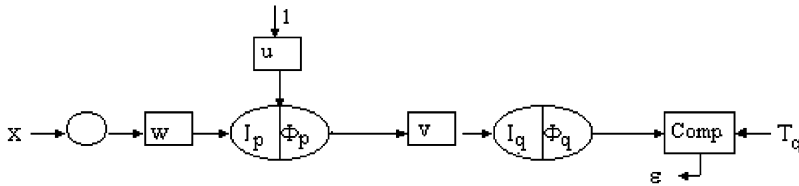


Fig. 2. The network syntax for backpropagation.

In the network syntax shown in Fig. 2, the neurons in the hidden and the output layer are indexed as p , and q , respectively. I is the internal activation and Φ_p is the hidden neuron output.

The output layer weights are changed in proportion to the negative gradient of the squared error with respect to the weights

$$\Delta v = -\eta_q \frac{\partial \varepsilon^2}{\partial v},$$

where η represents the learning rate.

These weight changes can be calculated using the chain rule

$$\Delta v = -\eta_q \frac{\partial \varepsilon^2}{\partial \Phi_q} \frac{\partial \Phi_q}{\partial I_q} \frac{\partial I_q}{\partial v}.$$

Then, the output layer weight change is given in Eq. (6):

$$\Delta v = -\eta_q (-2) [T_q - \Phi_q] \Phi_p. \quad (6)$$

Finally, the weight update formula for the output neurons is

$$v(N+1) = v(N) - \eta_p (-2) [T_q - \Phi_q] \Phi_p. \quad (7)$$

To update the output layer weights, the error term is, first calculated, then, the gradient vector of the output weight matrix is obtained, and finally, the weights are updated by a learning rate η . Until the desired error goal is obtained, the network will be trained.

The hidden layer outputs have no target values. To modify the hidden layer weights for minimizing error, a procedure is used to backpropagate the output layer

outputs to the hidden layer neurons as

$$\begin{aligned}\Delta w &= -\eta_p \frac{\partial \varepsilon^2}{\partial w} \\ &= -\eta_p \frac{\partial \varepsilon_q^2}{\partial \Phi_q} \frac{\partial \Phi_q}{\partial I_q} \frac{\partial I_q}{\partial \Phi_p} \frac{\partial \Phi_p}{\partial I_p} \frac{\partial I_p}{\partial w}.\end{aligned}\quad (8)$$

After that, the weights between the input layer and the hidden layer are updated as follows:

$$w(N+1) = w(N) - \eta_p 2 [T_q - \Phi_q] v \alpha \Phi_p [1 - \Phi_p]. \quad (9)$$

Similarly, the change of biases and updating them can be obtained by using the above-defined procedure

$$\begin{aligned}\Delta u &= -\eta_p \frac{\partial \varepsilon^2}{\partial u} \\ &= -\eta_p \frac{\partial \varepsilon_q^2}{\partial \Phi_q} \frac{\partial \Phi_q}{\partial I_q} \frac{\partial I_q}{\partial \Phi_p} \frac{\partial \Phi_p}{\partial I_p} \frac{\partial I_p}{\partial u},\end{aligned}\quad (10)$$

$$u(N+1) = u(N) - \eta_p 2 [T_q - \Phi_q] v \alpha \Phi_p [1 - \Phi_p]. \quad (11)$$

When a neural network is trained by backpropagation algorithm iteratively, it is more efficient to use an adaptive learning rate. The learning rate can be thought of as the size of a step down of the error gradient. If very small step size is taken, minimizing of the error can be guaranteed but training may take a very long time [19]. On the other hand, larger step size may result in unstable learning. To get both stability and fast procedure, the appropriate step size is selected according to the error change.

Different from conventional backpropagation training case of extended backpropagation one, computation of the error value depends not only on the network output but also the derivatives of the network output w.r.t. any of its output. Therefore, we need to calculate both the gradient of the network and the gradient of the network derivatives w.r.t. its inputs for computing the gradient of error w.r.t. the network weights. One can easily show that

$$\frac{\partial^s \Phi_q}{\partial x_i^s} = \sum_{i=1}^m v_i w_{ij}^s \Phi_{pi}^{(s)}, \quad (12)$$

where $\Phi_{pi} = \Phi_{pi}(I_{pi})$ and $\Phi_{pi}^{(s)}$ denotes the s th-order derivatives of the sigmoid. In addition, one can verify that

$$\frac{\partial^{\lambda_1}}{\partial x_1^{\lambda_1}} \frac{\partial^{\lambda_2}}{\partial x_2^{\lambda_2}} \cdots \frac{\partial^{\lambda_n}}{\partial x_n^{\lambda_n}} \Phi_q = \sum_{i=1}^n v_i O_i \Phi_{pi}^{(\Gamma)}, \quad (13)$$

where

$$O_i = \prod_{s=1}^n w_{is}^{\lambda_s} \quad (14)$$

and

$$\Gamma = \sum_{i=1}^n \lambda_i. \quad (15)$$

The derivative of network is the derivative of feedforward ANN with one hidden layer and denoted with $\Phi'_q(\vec{x})$. The derivative of network has the same w_{ij} and u_i with the original network, however, the activation function of each hidden neuron is replaced with Γ th-order derivative of logistic activation function and each v_i is replaced with $v_i O_i$.

The gradient of Φ'_q w.r.t. parameters of the original networks is obtained as Eqs. (16)–(18), which is used in chain rule as in the classical backpropagation

$$\frac{\partial \Phi'_q}{\partial v_i} = O_i \Phi_{pi}^{(\Gamma)}, \quad (16)$$

$$\frac{\partial \Phi'_q}{\partial u_i} = v_i O_i \Phi_{pi}^{(\Gamma+1)}, \quad (17)$$

$$\frac{\partial \Phi'_q}{\partial w_{ij}} = x_j v_i O_i \Phi_{pi}^{(\Gamma+1)} + v_i \lambda_j w_{ij}^{\lambda_j-1} \left(\prod_{s=1, s \neq j}^{\lambda_s} w_{is}^{\lambda_s} \right) \Phi_{pi}^{(\Gamma)}. \quad (18)$$

We use the logistic activation function represented by Eq. (19) for the hidden layer

$$\Phi_p(I_p) = \frac{1}{1 + e^{-\alpha I_p}}. \quad (19)$$

In this study, we compute n th-order derivatives of logistic activation function and find the output of derivative networks by assuming the slope constant ($\alpha = 1$)

$$\Phi'_p(I_p) = \Phi_p(I_p)(1 - \Phi_p(I_p)), \quad \Phi''_p(I_p) = \Phi'_p(I_p) - 2\Phi_p(I_p)\Phi'(I_p), \quad (20a)$$

$$\Phi'''(I) = \Phi'(I) - 6\Phi(I)\Phi'(I) + 6\Phi(I)^2\Phi'(I). \quad (20b)$$

3. Solution of differential equations

3.1. Solution of ordinary differential equations (ODEs)

To illustrate the method, we consider the first-order ODE as follows:

$$\frac{d\psi(t)}{dt} = f(t, \psi), \quad (21)$$

where $t \in [0, 1]$ and IC: $\psi(0) = A_0$.

Trial solution satisfying DE can be written as

$$\psi_T(t) = A_0 + t\Phi(t, \vec{c}), \quad (22)$$

where $\Phi(t, \vec{c})$ is the output of a feedforward ANN with one input and weight \vec{c} . The error quantity to be minimized can be defined

$$E[\vec{c}] = \sum_i \left\{ \frac{d\psi_T(t_i)}{dt} - f(t_i, \psi_T(t_i)) \right\}^2, \quad (23)$$

where t_i 's are points in $[0, 1]$. Since $d\psi_T(t)/dt = \Phi(t, \vec{c}) + t d\Phi(t, \vec{c})/dt$, the gradient of the error is computed w.r.t. weights and biases by using formulas given in the previous section. The same procedure can be applied to all ODEs

For the given second-order ODE,

$$\frac{d^2\psi(t)}{dt^2} = f\left(t, \psi, \frac{d\psi}{dt}\right) \quad (24)$$

with ICs: $\psi(0) = A_0$ and $(d/dt)\psi(0) = A_1$. The following equation can be proposed as a trial solution:

$$\psi_T(t) = A_0 + A_1 t + t^2 \Phi(t, \vec{c}). \quad (25)$$

We can generalize this equation for n th-order ODEs as

$$\psi_T(t) = \sum_{i=0}^{M-1} A_i t^i + t^M \Phi(t, \vec{c}). \quad (26)$$

For the second-order boundary value problems, if $\psi(0) = A$ and $\psi(1) = B$, then next equation can be proposed as a trial solution:

$$\psi_T(t) = A(1-t) + Bt + t(1-t)\Phi(t, \vec{c}). \quad (27)$$

In case of the second-order ODEs, the error function to be minimized can be defined as

$$E[\vec{c}] = \sum_i \left\{ \frac{d^2\psi_T(t_i)}{dt^2} - f\left(t_i, \psi_T(t_i), \frac{d\psi_T(t_i)}{dt}\right) \right\}^2. \quad (28)$$

3.2. Solution of partial differential equations (PDEs)

In this study, we considered the second and the fourth-order PDEs, where $x \in [0, 1]$ and $t \in [0, 1]$. We propose a neural trial solution generated from two parts, $\psi_T(x, t) = A(x, t) + F(\vec{x}, \Phi(x, t, \vec{c}))$, satisfying ICs and BCs as in the ODEs. If we consider the wave equation

$$\frac{\partial^2 \psi}{\partial t^2} - a^2 \frac{\partial^2 \psi}{\partial x^2} = 0, \quad (29)$$

which has the following ICs and BCs with $t \in [0, 1]$ and $x \in [0, 1]$:

$$\psi(x, 0) = g_0(x), \quad \frac{\partial \psi}{\partial t}(x, 0) = g_1(x),$$

$$\psi(0, t) = f_0(t), \quad \psi(1, t) = f_1(t).$$

A solution for the wave equation can easily be found. The trial neural solution can be written as follows:

$$\begin{aligned} A(x, t) = & (1 - t^2)\{g_0(x) - [(1 - x)g_0(0) + xg_0(1)]\} \\ & + t\{g_1(x) - [(1 - x)g_1(0) + xg_1(1)]\} \\ & + (1 - x)f_0(t) + xf_1(t) \end{aligned} \quad (30)$$

and

$$\psi(x, t) = A(x, t) + (x - x^2)t^2\Phi(x, t, \vec{c}) \quad (31)$$

the same procedure can easily be extended to the other types of PDEs. For example, the following parabolic fourth-order PDE:

$$\frac{\partial^2 \psi}{\partial t^2} + a^2 \frac{\partial^4 \psi}{\partial x^4} = 0, \quad (32)$$

which has the following ICs and BCs with $t \in [0, 1]$ and $x \in [0, 1]$:

$$\begin{aligned} \psi(x, 0) &= g_0(x), \quad \psi(x, 1) = g_1(x), \\ \psi(0, t) &= f_0(t), \quad \psi(1, t) = f_1(t), \quad \frac{\partial^2 \psi}{\partial x^2}(0, t) = f_3(t), \quad \frac{\partial^2 \psi}{\partial x^2}(1, t) = f_4(t), \\ B(x, t) &= (1 - x)f_0(t) + xf_1(t) - 1/4x(1 - x)^2(f_3(t) + 8f_4(t)) + 1/2x^2(1 - x)^2 \\ &\quad \times (f_4(t) - 4f_3(t)) + (1 - t^2)[g_0(x) - [(1 - x)g_0(0) + xg_0(1) \\ &\quad - 1/4x(1 - x)^2g_0(1) + 1/2x^2(1 - x)^2g_0(0)] + t[g_1(x) - \\ &\quad [(1 - x)g_1(0) + xg_1(1)] - 1/4x(1 - x)^2g_1(1) + 1/2x^2(1 - x)^2g_1(0)], \end{aligned} \quad (33)$$

$$\psi_T = B(x, t) + x^3(-1 + x)t^2 \left[\Phi(x, t, \vec{c}) - \Phi(1, t, \vec{c}) - \frac{1}{3} \frac{\partial \Phi(1, t, \vec{c})}{\partial x} \right]. \quad (34)$$

In the above PDE problems, the errors to be minimized are given by

$$\begin{aligned} E[\vec{c}] &= \sum_i \left\{ \frac{\partial^2 \psi_T(x_i, t_i)}{\partial t^2} - a^2 \frac{\partial^2 \psi_T(x_i, t_i)}{\partial x^2} - 0 \right\}^2, \\ E[\vec{c}] &= \sum_i \left\{ \frac{\partial^2 \psi_T(x_i, t_i)}{\partial t^2} + a^2 \frac{\partial^4 \psi_T(x_i, t_i)}{\partial x^4} - 0 \right\}^2, \end{aligned} \quad (35)$$

where (x_i, t_i) are points in $[0, 1] \times [0, 1]$.

Note that the parts satisfying the BCs and ICs of the neural solutions in Eqs. (30) and (33) should be rearranged for different BCs and ICs.

4. Numerical examples

In this section, we consider the numerical solutions of the vibration control problems of different lumped-parameter systems and distributed parameter systems

whose mathematical models are ODEs and PDEs, respectively, to illustrate the efficiency of the proposed method. Linear and non-linear mass-damper-spring systems with Lyapunov-based control, the control of longitudinal vibration of rods and the lateral vibration of beam are chosen as examples. In all cases, we use a feedforward ANN with three layers, having logistic activation function in the hidden layer but linear activation function in the output layer. The ANN is trained by backpropagation method used an adaptive learning rate. To test accuracies of this method, the above examples are also solved by either Runge–Kutta or analytical methods. Then, the obtained results are graphically presented and compared with each other. Furthermore, we test the method for the training point and outside of the training points to see approximate capability of the method for ODEs. The values of all physical parameters of the examples given in this paper are assumed to be one.

4.1. Example 1

In this example, we consider mass-damper-spring system shown in Fig. 3, whose mathematical model is represented by Eq. (36)

$$m \frac{d^2\psi}{dt^2} + c \frac{d\psi}{dt} + k\psi = 0. \quad (36)$$

where the ICs of this system are considered $\psi(0) = 1$ and $d\psi(0)/dt = 0$ with $t \in [0, 2]$. The network is trained by using a grid of 10 equidistant points in $[0, 2]$. According to Eq. (25), the trial neural form of the solution is selected to be $\psi_T(t) = 1 + t^2\Phi(t, \vec{c})$. ANN with the 1-10-1 architecture is chosen. The network output and the solution obtained from Runge–Kutta method are shown in Fig. 4a. Although the training of network is performed using only 10 points, it is clearly seen in Fig. 4b that the solution outside of interval training even has very high accuracy. The solution curves obtained from the both methods almost coincide with each other. Moreover, the extrapolation error remains small for the other points.

4.2. Example 2

Different from example 1, the damper and the spring have non-linear characteristics. The mathematical model of this system is expressed as

$$m \frac{d^2\psi}{dt^2} + b \frac{d\psi}{dt} \left| \frac{d\psi}{dt} \right| + k_0\psi + k_1\psi^3 = 0. \quad (37)$$

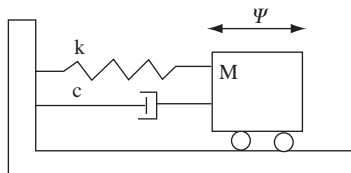


Fig. 3. Mass-damper-spring system.

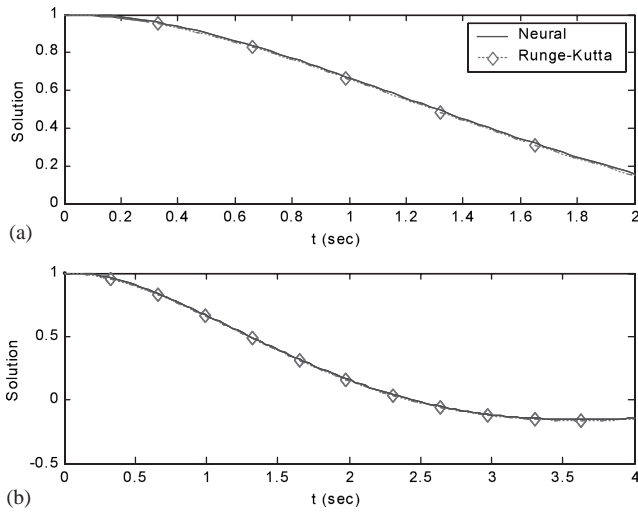


Fig. 4. Numerical solutions of the system given by Eq. (36): (a) interval training, and (b) outside of interval training.

where the ICs of this system are $\psi(0) = 1$, $d\psi(0)/dt = 0$ with $t \in [0, 2]$. The network is trained using a grid of 51 equidistant points in $[0, 2]$. According to Eq. (25), the trial neural form of solution is selected to be $\psi_T(t) = 1 + t^2\Phi(t, \vec{c})$. We obtain the appropriate results when the architecture of the ANN is chosen as 1-51-1. The network output and the solution obtained from Runge–Kutta method are shown in the Fig. 5. It is noted that the error gradually tends to zero when the neuron number in the hidden layer is increased for non-linear systems. Since the system is non-linear, the neural solution has lower performance outside of interval training if we compare with example 1. This defect can be eliminated increasing the neuron number in the hidden layer.

Then, we consider the non-linear mass-damper-spring system shown in Fig. 6 with a controller based on Lyapunov's direct method [20]. To obtain a stable controller, we define Lyapunov function as

$$V(\psi) = \frac{1}{2}m\dot{\psi}^2 + \int_0^\psi (k_0\psi + k_1\psi^3) d\psi. \quad (38)$$

The derivative of the Lyapunov function,

$$\dot{V}(\psi) = \dot{\psi}(m\ddot{\psi} + k_0\psi + k_1\psi^3), \quad (39)$$

if we rearrange Eq. (37) as follows:

$$m\ddot{\psi} = -(b\dot{\psi}|\dot{\psi}| + k_0\psi + k_1\psi^3 - u(t)). \quad (40)$$

Then, Eq. (39) becomes

$$\dot{V}(\psi) = \dot{\psi}(u(t) - b\dot{\psi}|\dot{\psi}| - k_0\psi - k_1\psi^3 + k_0\psi + k_1\psi^3). \quad (41)$$

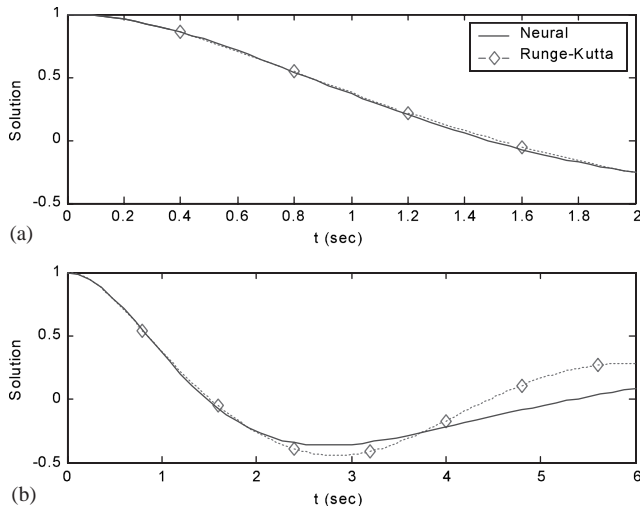


Fig. 5. Numerical solutions of system given by Eq. (37): (a) interval training, and (b) outside of interval training.

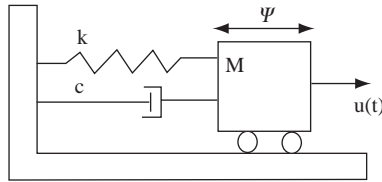


Fig. 6. Non-linear mass-damper-spring system.

If we select the control law as

$$u(t) = -k_d \dot{\psi} + b \dot{\psi} |\dot{\psi}| \quad (42)$$

and substituting this control law into Eq. (40), we get

$$\dot{V}(\psi) = -k_d \dot{\psi}^2. \quad (43)$$

The last equation implies that the system with the controller represented by Eq. (42) is asymptotically stable.

Assuming $k_d = 1$, Fig. 7a shows that the solutions of ANN having 1-51-1 architecture and Runge-Kutta method are very close to each other. The high accuracy of ANN for outside of interval training is illustrated in Fig. 7b.

4.3. Example 3

As the first example of PDEs, we consider the wave equation which is the mathematical model of the non-controlled-longitudinal vibration of rod shown

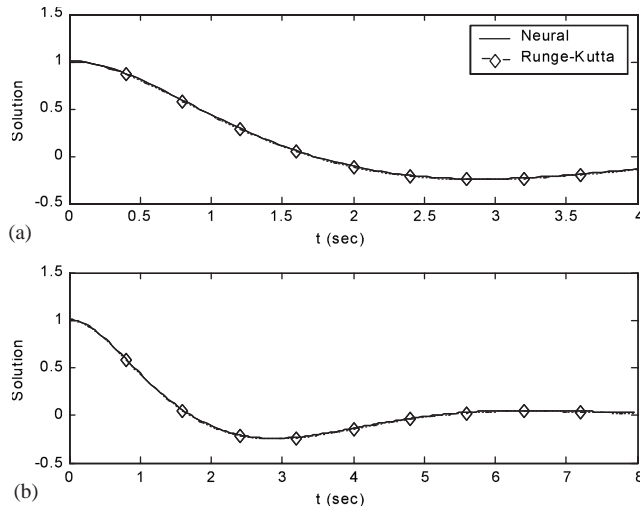


Fig. 7. The solutions of ANN and Runge-Kutta method for the controlled system represented by Eq. (37): (a) interval training, and (b) outside of interval training.

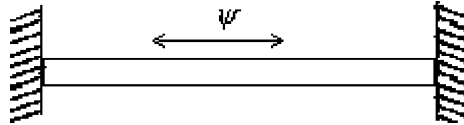


Fig. 8. Longitudinal vibrations of rods.

in Fig. 8

$$\frac{\partial^2 \psi}{\partial t^2} - a^2 \frac{\partial^2 \psi}{\partial x^2} = 0. \quad (44)$$

This equation has following the ICs and BCs with $t \in [0, 1]$ and $x \in [0, 1]$:

$$(x, 0) = \sin(\pi x), \quad \frac{\partial \psi(x, 0)}{\partial t} = 0, \quad 0 \leq x \leq 1,$$

$$\psi(0, t) = \psi(1, t) = 0.$$

For the numerical solution, we select Eq. (45) as a trial solution, assuming $a = 1$. Fig. 9 shows the solution of ANN. In Fig. 10, the error between the solution of ANN and the analytical solutions represented by Eq. (46) is very small

$$\psi_T = (1 - t^2) \sin(\pi x) + (x - x^2)t^2[\Phi(x, t, \vec{c})], \quad (45)$$

$$\psi_a = \sin(\pi x) \cos(\pi t). \quad (46)$$

4.4. Example 4

In this example, we use a boundary controller for the longitudinal vibration control of rods as shown in Fig. 11. The motion equation of the given system is the wave equation as

$$\rho \frac{\partial^2 \psi}{\partial t^2} - \frac{\partial^2 \psi}{\partial x^2} = 0. \quad (47)$$

This equation is subject to following the ICs and BCs with $t \in [0, 1]$ and $x \in [0, 1]$:

$$\psi(x, 0) = \left(1 - \cos \frac{\pi}{2}(3x)\right), \quad \frac{\partial \psi}{\partial t}(x, 0) = 0, \quad 0 \leq x \leq 1,$$

$$\psi(0, t) = 0, \quad \frac{\partial \psi}{\partial x}(1, t) + k\psi(1, t) = 0.$$

The proposed neural trial solution is

$$\psi_T = \left(1 - \cos \frac{\pi}{2}(3x)\right) \cos t^2 + x(1-x)t^2[\Phi(x, t, \vec{c}) - \Phi(1, t, \vec{c})]. \quad (48)$$

As shown in Figs. 12 and 13, the response of the controlled system applied to the boundary controller is damped, as it is desired.

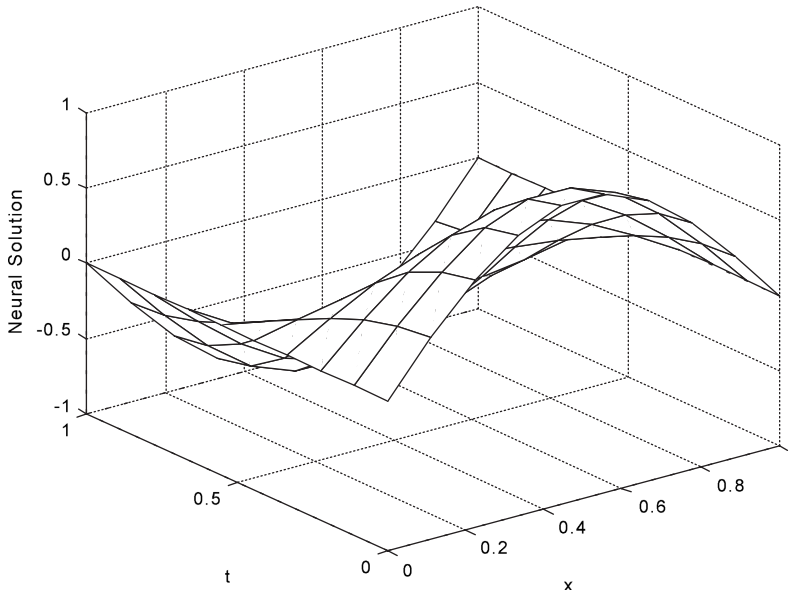


Fig. 9. The neural solution of the system represented by Eq. (44).

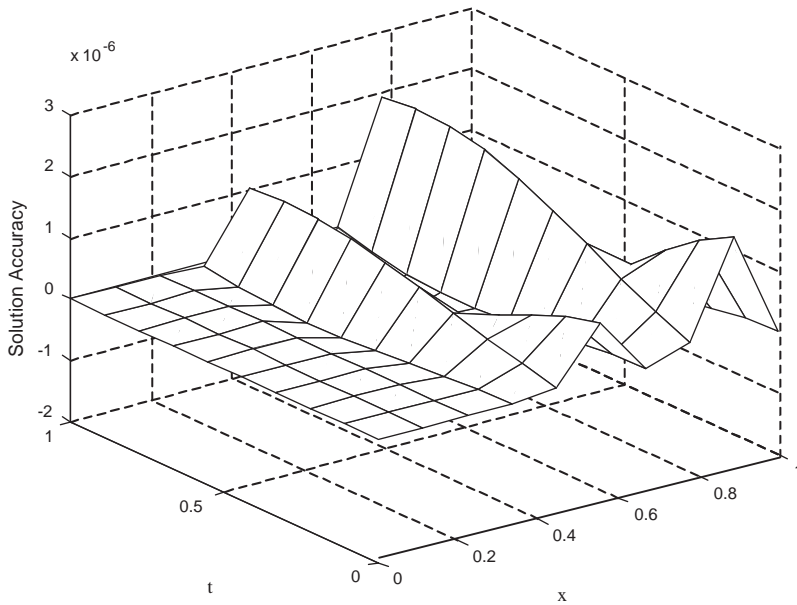


Fig. 10. Accuracy of the computed solution of systems represented by Eq. (44) at training points.

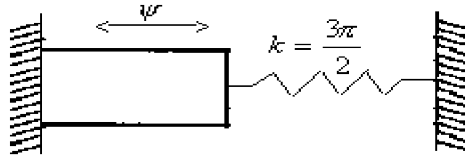


Fig. 11. Longitudinal vibrations of rods.

4.5. Example 5

The motion equation of the lateral vibration of beam shown in Fig. 14 is

$$\frac{\partial^2 \psi}{\partial t^2} + a^2 \frac{\partial^4 \psi}{\partial x^4} = 0, \quad (49)$$

where the ICs and BCs are

$$\psi(x, 0) = \sin(\pi x), \quad \psi(x, 1) = 0, \quad 0 \leq x \leq 1,$$

$$\psi(0, t) = \psi(1, t) = 0, \quad \frac{\partial^2 \psi}{\partial x^2}(0, t) = \frac{\partial^2 \psi}{\partial x^2}(1, t) = 0.$$

We propose the trial neural solution as follows:

$$\psi_T = (1 - t^2) \sin(\pi x) + x^3(-1 + x)t^2 \left[\Phi(x, t, \vec{c}) - \Phi(1, t, \vec{c}) - \frac{1}{3} \frac{\partial \Phi(1, t, \vec{c})}{\partial x} \right]. \quad (50)$$

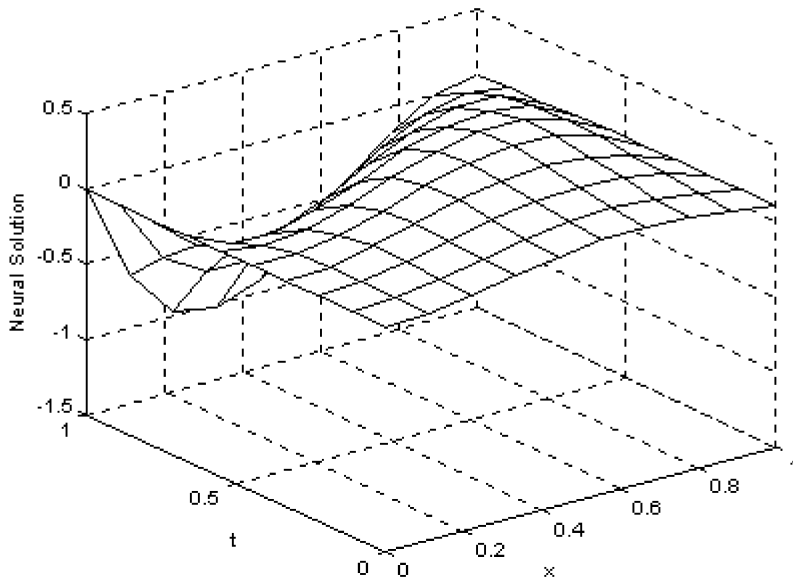


Fig. 12. The neural solution of system represented by Eq. (47).

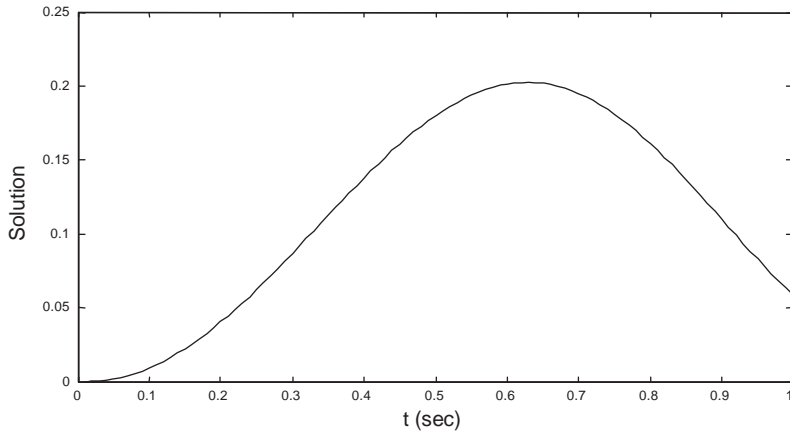


Fig. 13. The ψ versus t at $x = 0.99$.

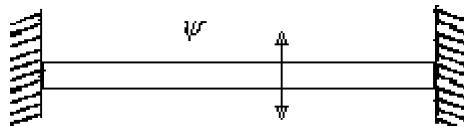


Fig. 14. Lateral vibration of beam having the fixed boundaries.

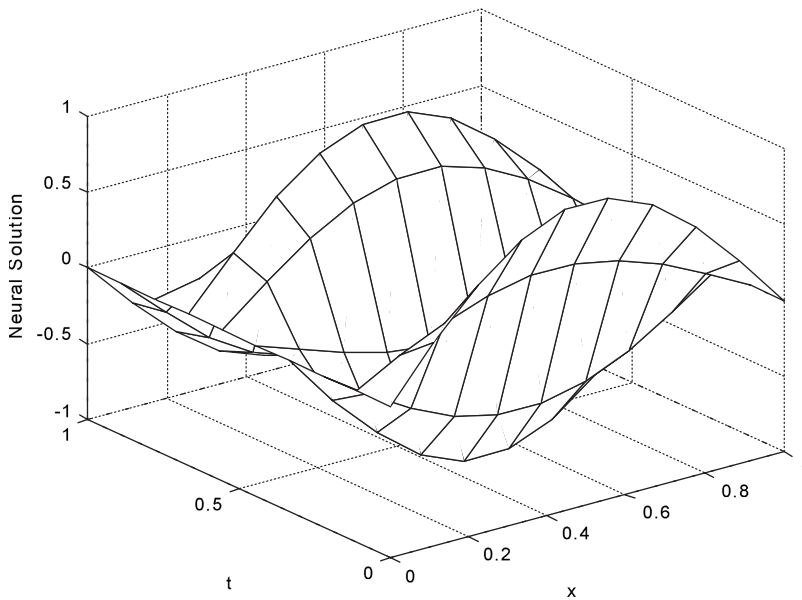


Fig. 15. The neural solution of system represented by Eq. (49).

Eq. (51) is the analytical solution of the given system:

$$\psi_a = \sin(\pi x) \cos(\pi^2 t). \quad (51)$$

Fig. 15 shows the neural solution of system represented by Eq. (49). The error between the solution obtained from ANN and the analytical solution is shown in Fig. 16. This figure illustrates that the obtained results from both the methods are in very close agreement.

5. Conclusion

The dynamics of the vibrations of flexible structure are generally represented by either ODEs or PDEs. Because of non-linearity and complex BCs, their numerical solutions always have some trouble such as numerical instability. That is why, we propose an alternative method using feedforward ANNs. The main advantages of this method are accuracy and presenting differentiable-closed form solutions. Moreover, the form of proposed trial solution satisfying BCs and transforming constrained optimization problem to unconstrained one are the main reasons of the successful applications of this method.

The architecture of the proposed ANN consists of one hidden layer varying its neuron number to deal with highly non-linear problems. We have first developed the general formula for the numerical solutions of n th-order initial-value problems by using ANN. Moreover, we successfully apply this method to many controlled and non-controlled vibration problems of flexible structures whose dynamics are

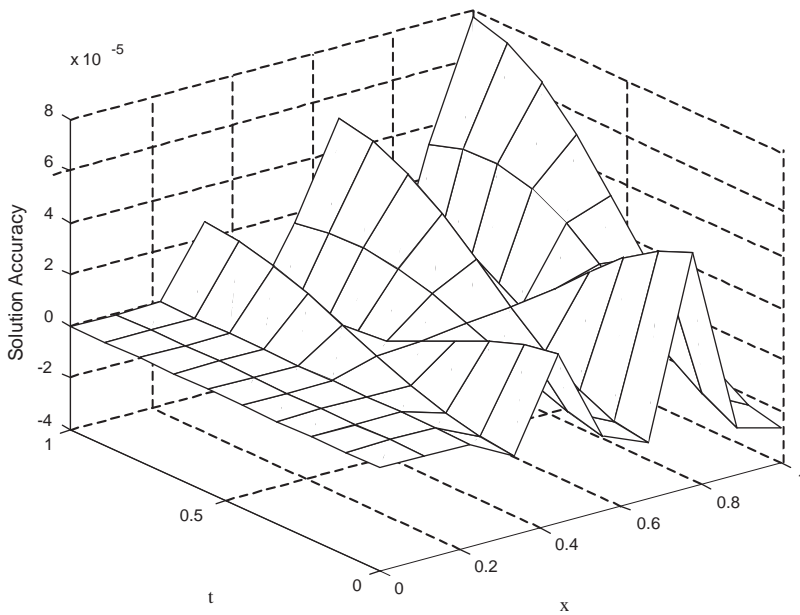


Fig. 16. Accuracy of the computed solution of the system represented by Eq. (49) at training points.

represented by ODEs and PDEs. To test this method, we also obtain the solutions of the same problems by using analytical and Runge–Kutta method. The obtained figures show that the results are in very close agreement. Furthermore, we note that this method also successes outside of the training points when the neuron numbers in the hidden layer are increased.

Consequently, this method can be used for a wide class of linear and non-linear ODEs and PDEs with complex BCs. Therefore, it is general and easy to apply for numerical solutions of dynamic problems.

References

- [1] P.M. Lima, M.P. Carpentier, Iterative methods for a singular boundary-value problem, *J. Comput. Appl. Math.* 111 (1999) 173–186.
- [2] H. Guoqiang, W. Jiong, K. Hayami, X. Yuesheng, Correction method and extrapolation method for singular two-point boundary value problems, *J. Comput. Appl. Math.* 126 (2000) 145–157.
- [3] D. Kincaid, W. Cheney, *Numerical Analysis*, Brooks/Cole, Monterey, CA, 1991.
- [4] K.S. Yee, Numerical solution of initial boundary value problems involving Maxwell's equation in isotropic media, *IEEE Trans. Antennas Propagation AP-14* (1996) 302–307.
- [5] P. Hunter, A. Pullan, *FEM/BEM Notes*, Department of Engineering Science, The University of Auckland, New Zealand, 1997.
- [6] M.A. Kolbehdari, M.S. Nakhla, M.N.O. Sadiku, Hybrid model of scattering from eccentrically nested dielectric cylinders, *J. Franklin Inst.* 225B (1999) 43–51.

- [7] G. Jacobsohn, A discrete Taylor series method for the solution of two-point boundary-value problems, *J. Franklin Inst.* 338 (2001) 61–68.
- [8] Y. Guoyou, W.J. Mansur, J.A.M. Carrer, L. Gong, Stability of Galerkin and collocation time domain boundary element methods as applied to the scalar wave equation, *Comput. Struct.* 74 (2000) 495–506.
- [9] L. Meirovitch, T.J. Stemple, A new approach to the modeling of distributed structures for control, *J. Franklin Inst.* 338 (2001) 241–254.
- [10] W-S. Lee, Y-H. Ko, C-C. Ji, A study of an inverse method for the estimation of impulsive heat flux, *J. Franklin Inst.* 337 (2000) 661–671.
- [11] J. Kouatchou, Parallel implementation of a high-order implicit collocation method for the heat equation, *Math. Comput. Simul.* 54 (2001) 509–519.
- [12] B. Bialecki, G. Fairweather, Orthogonal spline collocation methods for partial differential equations, *J. Comput. Appl. Math.* 128 (2001) 55–82.
- [13] T. Nguyen-Thien, T. Tran-Cong, Approximation of functions and their derivatives: a neural network implementation with applications, *Appl. Math. Modell.* 23 (1999) 687–704.
- [14] S. He, K. Reif, R. Unbehauen, Multilayer neural networks for solving a class of partial differential equations, *Neural Networks* 13 (2000) 385–396.
- [15] C.L. Karr, I. Yakushin, K. Nicolosi, Solving inverse initial-value, boundary-value problems via genetic algorithm, *Eng. Appl. Artif. Intell.* 13 (2000) 625–633.
- [16] N. Mai-Duy, T. Tran-Cong, Numerical solution of differential equations using multiquadric radial basis function networks, *Neural Networks* 14 (2001) 185–199.
- [17] I.E. Lagaris, A. Likas, D.I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, *IEEE Trans. Neural Networks* 9 (5) (1998) 987–1000.
- [18] I.E. Lagaris, A. Likas, D.G. Papageorgio, Neural-network methods for boundary value problems with irregular boundaries, *IEEE Trans. Neural Networks* 11 (5) (2000) 1041–1049.
- [19] J.W. Hines, *Fuzzy and Neural Approaches in Engineering MATLAB Supplement*, Prentice-Hall, Englewood Cliffs, NJ, 1997.
- [20] J-J.E. Slotine, W. Li, *Applied Nonlinear Control*, Prentice-Hall, Englewood, Cliffs NJ, 1991.