International Journal of Applied Mathematical Research, 2 (1) (2013) 134-139
©Science Publishing Corporation
www.sciencepubco.com/index.php/IJAMR

Design feed forward neural network for solving two dimension singularly perturbed integro-differential and integral equation

Eman Ali Hussain[1] , Nahdh. S. M. Al_Saif[2*]

[1]*Department of Mathematics, College of Science, University of AL-Mustansiriyah, Iraq*
[2]*Department of physics, College of science, University of AL-anbar, Iraq*
**Corresponding author E-mail: nn_ss_m68@yahoo.com*

Abstract

Recently, there has been an increasing interest in the study of singular and perturbed systems. In this paper design fast feed forward neural network to present a method to solve two dimensions singularly perturbed integro-differential and integral equations. Using a multi-layer having one hidden layer with 7 hidden units (neurons) and one linear output unit the sigmoid activation of each unit is radial basis function and Levenberg – Marquardt (trainlm) training algorithm. Finally the results of numerical experiments are compared with the exact solution in illustrative examples to confirm the accuracy and efficiency of the presented scheme..

Keywords: *singularly perturbed problems; Volterra integral equations; Volterra integro-differential equations; feed forward neural network, Levenbrg- Marquardt training.*

1 Introduction

As we know, much work has been done on developing and analyzing numerical methods for solving one-dimensional integro-differential equation of the second kind, but in two-dimensional cases a small amount of work has been done.
In the present work, we consider the two-dimensional singularly perturbed Volterra integro-differential equations (SVIDE)

$$\varepsilon \frac{d}{dx} u(x,y) = g\big(x,y,\varepsilon,u(x,y)\big) + \int_0^x \int_0^y k\big(x,y,\varepsilon,u(x,y)\big) dsdt, \quad x,y \in [0,x] * [0,y] \tag{1}$$

$$u(0,x) = \propto$$

Where
α is a constant and ε is a known perturbation parameter which $0 < \varepsilon \leq 1$.
Here, ε is small parameter that given rise to singularly perturbed nature of the problem, the kernel K and the data function g(x) are given smooth functions. Under appropriate condition on g and K, for every $\varepsilon > 0$, Eq. 1 has unique continuous solution on [0,x] x [0,y],[1].
Finding the solutions on one dimension of these problems has been widely studied by researchers in the last decade. Implicit Runge-Kutta methods were presented for singularly perturbed integro-differentialc equations in [2] and for singularly perturbed integro-differential systems in [3]. In [4], Orsi applied a Petrov-Galerkin method to singularly perturbed integro-differential-algebraic equations. El-Gendi [5] applied spectral methods to obtain solution of singularly perturbed differential, integral and integro-differential equations. Hu [6] and Horvat et al. [7] solved the SVIDEs by using the spline collocation methods. Recently, in [8] a numerical procedure based on finite difference was presented for solving a class of SVIDEs. More recently, Ramos [9] applied Piecewise-quasilinearization techniques to obtain solution of SVIDEs.
Many methods have been developed so far solving integral and integro-differential equation. Some of them produce a solution in the form of an array that contains the value of the solution at a selected group of point, other use basis function to represent the solution in analytic form and transform the original problem usually to a system of algebraic equation. owadays there is a new way of computing denominated artificial intelligence which through different methods is capable of managing the imprecisions and uncertainties that appear when trying to solve

problems related to the real world, offering strong solution and of easy implementation. One of those techniques is known as Artificial Neural Networks (ANN).
Inspired, in their origin, in the functioning of the human brain, and entitled with some intelligence. These are th e combination of a great amount of elements of process–artificial neurons interconnected that operating in a parallel way get to solve problems related to aspects of classification. the construction of any given ANN we can identify, depending on the location in the network, three kind of computational neurons: input, output and hidden.
In this paper is organized as follows: the next section definition the ANN, in section 3 describe the structure of neural network, in section 4 the Levenberg Algorithm derivation, description of method in section 5,in section 6 Illustration of the method, In section 7 report our numerical finding accuracy of method . finally conclusions the last part of the paper.

2 Artificial neural network

There are different ways of defining what the ANN are, from short and generic definitions the ones that try to explain inadetailed way what means neural network or neural computing. For this situation, the definition that was proposed by Haykin [10], appears below:

Artificial Neural Networks are massively interconnected network in parallel of simple elements (usually adapt able), with hierarchic organization, which try to interact with the objects of the real world in the same way that the biological nervous system does.

as a simple element we understand the artificial equivalent of a neuron that is known as computational neuron or node. These are organized hierarchically by layers and are interconnected between the m just as in the biological nervous systems. Upon the presence of an external stimulus the artificial neural net work generates an answer, which is confronted with the reality to determine the degree of adjustment that is required in the internal network parameters.this adjustment is known as learning network or training, after which the network is ready to answer to the external stimulus in an optimum way.

ANN is characterized by [**11**]
1-Architecture: its pattern of connections between the neurons.
2-training Algorithm: its method of determining the weight on the connections.
3- Activations function.

3 Structure of a neural network (Topology)[10]

in an artificial neural network expressions structure, architecture or topology, express the way in which computational neurons are organized in the network. Particularly, these terms are focused in the description of how the nodes are connected and in how the information is transmitted through the network. As it has been mentioned, the distribution of computational in the following:
Number of levels or layers: neurons in the neural network is done forming levels or layers of a determined num ber of nodes each one. As there are input, output and hidden neurons, we can talk about an ***input layer***, an ***output layer*** and ***single layer or multilayer hidden layers***. By the peculiarity of the behavior of the input nodes some authors consider just two kinds of layers in the ANN, the hidden and the output.
Connection patterns: Depending on the links between the elements of the different layers. the ANN can be class ified as: ***totally connected,*** when all the outputs from a level get to all and each one of the nodes in the following level, if some of the links in the network are lost, then we say that the network is ***partially connected.***
Information flow: Another classification of the ANN is obtained by considering the direction of the flow of the through the layers, When any output of the neurons is input of neurons of the same level or preceding levels, the network is described as ***feedforward***. In counter position if there is at least one connected exit as entrance of neurons of previous levels or of the same level, including themselves, the network is denominated of ***feedback.***

4 Levenberg-Marquardt algorithm (LM) [12]

For LM algorithm, the performance index to be optimized is defined

$$F(w) = \sum_{p=1}^{P}[\sum_{k=1}^{K}(d_{KP} - o_{KP})^2] \quad (2)$$

Where $w = [w_1 \ w_2 \ \dots w_N]^T$ consists of all weights of the network, d_{kp} is the desired value of the k^{th} output and the p^{th} pattern, o_{kp} is the actual value of the k^{th} output and the p^{th} pattern, N is the number of the weights, P is the number of pattern, and K is the number of the network output.
Equation (2) can be written its

$$F(w) = E^T E \quad (3)$$

Where

$$E = \left[e_{11} \dots e_{K1} e_{12} \dots e_{K2} \dots e_{1p} \ \dots e_{KP}\right]^T$$

$$e_{kp} = d_{kp} - o_{kp} \quad k = 1, \dots, K, \quad p = 1, \dots, P$$

Where E is the cumulative error vector (for all pattern)
from equation (3) the jacobian matrix is define as

$$J = \begin{bmatrix} \frac{\partial e_{11}}{\partial w_1} & \frac{\partial e_{11}}{\partial w_2} & \dots & \frac{\partial e_{11}}{\partial w_N} \\ \frac{\partial e_{21}}{\partial w_1} & \frac{\partial e_{21}}{\partial w_2} & \dots & \frac{\partial e_{21}}{\partial w_N} \\ \dots & \dots & \dots & \dots \\ \frac{\partial e_{k1}}{\partial w_1} & \frac{\partial e_{k2}}{\partial w_2} & \dots & \frac{\partial e_{k1}}{\partial w_N} \\ \dots & \dots & \dots & \dots \\ \frac{\partial e_{1p}}{\partial w_1} & \frac{\partial e_{1p}}{\partial w_2} & \dots & \frac{\partial e_{1p}}{\partial w_N} \\ \frac{\partial e_{2p}}{\partial w_1} & \frac{\partial e_{2p}}{\partial w_2} & \dots & \frac{\partial e_{2p}}{\partial w_N} \\ \dots & \dots & \dots & \dots \\ \frac{\partial e_{kp}}{\partial w_1} & \frac{\partial e_{kp}}{\partial w_2} & \dots & \frac{\partial e_{kp}}{\partial w_N} \end{bmatrix} \quad (4)$$

And the weights are calculated using the following equation

$$w_{t+1} = w_t - (J_t^T J_t + \mu_t I)^{-1} J_t^T E_t \quad (5)$$

Where I is identity unit matrix, μ is the learing parameter and J is jacobian of m output error with respect to n weights of the neural network. The μ parameter is automatically adjusted at each iteration in order to secure convergence, the LM algorithm requires computation of the jacobian J matrix at each iteration step and the inversion of $J^T J$ square matrix, the dimension of which is N*N.

5 Description of the method

In this section illustrate how our approach can be used to the approximation solution of the singularly perturbed integro-differential equation

$$\epsilon \frac{d}{dx} u(x,y) = f(x,y,\epsilon) + \int_0^{y_i} \int_0^{x_i} K(x,y,\epsilon) u(t,s) dt ds \ \ x,y \in I = [0,x]x[0,y]$$

Where a subject to certain IC . x,y ϵ I ϵ R^2, denoted the domain and y(x) is the solution to be computed.
If $y_t(x,y,p)$ denoted a trial solution with adjustable parameters p, the problem is transformed to a discretize from:

$$Min \sum_{x_i,y_i \in D} f\ (x_i, y_i \epsilon) + \int_0^{x_i} \int_0^{y_i} K(x_i, y_i, t, s, \epsilon)\ y(t,s) dtds \tag{6}$$

Subject to the constraints imposed by the IC's.
In the our proposed approach, the trial solution y_t employs a FFNN and the parameters p correspond to the weight and biases of the neural architecture, we choose a form for the trial function $y_t(x,y)$ such that it satisfies the IC's. this is achieved by writing it as a sum of two terms :

$$y_t(x_i, y_i, p) = A(y) + G\big(x, , y, N(x, y, p)\big) \tag{7}$$

Where N(x,y,p) is a single-output FFNN with parameters p and n input unit fed with the input vector x. the term A(x,y) contain no adjustable parameters and satisfies the IC's. the second term G is constructed so as not to contribute to the IC's, since $y_t(x,y)$ satisfy them. This term can be formed by using a Ann whose weight and biases are to be adjusted in order to deal with the minimization problem.

6 Illustration of the method

In this section we described solution of singularly perturbed integro-differential equation using FFNN.
To illustrate the method, we consider the integro-differential equation

$$\epsilon \frac{d}{dx} u(x,y) = f(x, y, \epsilon, u) + \int_0^{y_i} \int_0^{x_i} K(x, y, \epsilon) u(t,s) dtds \ \ x, y \in I = [0, x]x[0, y]$$

Where x,y ϵ [0, 1] and the IC: y(0,y) = A, a trial solution can be written as:

$$y_t(x) = A(x, y) + x.y.N(x, y, p) \tag{8}$$

Where N(x,y,p) is the output of a FFNN with one input unite for x,y and weights p.
Note that $y_t(x,y)$ satisfies the IC by construction the error quantity to be minimized is given by :

$$E(p) = \left\{ \in \frac{d}{dx} y_t \sum_{i=1}^{n} f(x_i, y_i \in, y_t(x_i, y_i) + \int_0^{x_i} \int_0^{y_i} K\big(x_i, t, \in, y_t(t)\big)\, dsdt \right\}^2 \tag{9}$$

where the x_i , y_i ϵ [0, 1]x[0,1].
Since:

$$\frac{d}{dx} y_t(x) = y.N(x, y, p) + x.y.\frac{d}{dx} N(x, p) \tag{10}$$

It is straight forward to compute the gradient of the error with respect to the parameters p.

7 Numerical result

In this section we report some numerical result and the solution of number of model problem. In all cases we used a multi-layer FFNN having one hidden layer with 7 hidden units (neurons) and one linear out output unit. The sigmoid activation of each hidden is radbas(radial basis function). For each test problem the exact analytic solution $y_a(x,y)$ were known in advance. Therefore we test the accuracy of obtained solutions computing the mean square error (MSE).

Example 1: In this problem, consider the following singularly perturbed Volterra integral equation

$$\varepsilon u(x,y) + \int_0^x \int_0^y x^2\, y^3 u(t,s) dsdt = x^2 + y^3 + \frac{x^5 y^4}{\varepsilon} + \frac{x^3 y^4}{\varepsilon} \tag{11}$$

which has the following exact solution:

$$u(x,y) = (x^2 + y^3)/\varepsilon$$

Applied present method and solved eq. (11) for different value of ε. Table (1) given mean square error of the designer network,

Table 1: men square error of the network for example 1

Mse						
€→	2^{-0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}
mse	$3.6445e^{-6}$	$7.6575e^{-6}$	$6.5214e^{-6}$	$3.1470e^{-5}$	$2.7238e^{-5}$	$1.9632e^{-5}$

Example 2: In this problem, consider the following nonlinear singularly perturbed Volterra integral equation

$$\varepsilon u(x,y) - \int_0^x \int_0^y (x-y)u^2(t,s)dsdt = \varepsilon xy - \frac{x^4y^3}{12} + \frac{x^3y^4}{12} \tag{12}$$

which has the following exact solution:

$$u(x,y) = xy$$

applied present method and solved eq.(12) for different value of ε.Table (2) given mean square error of the designer network,

Table 2: mean square error of the network for example 2

Mse						
€→	2^{-0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}
mse	$5.9502e^{-9}$	$1.0062e^{-8}$	$4.2834e^{-7}$	$2.5434e^{-9}$	$6.3642e^{-10}$	$1.7149e^{-8}$

Example 3: In this problem, consider the following integro-differentioal singularly perturbed Volterra integral equation

$$\varepsilon \frac{\partial}{\partial x}u(x,y) - \int_0^x \int_0^y (t+s)\,u(t,s)dsdt = \frac{(y+1)}{\varepsilon} - \frac{x^3y^2}{6} - \frac{x^3y}{3} - \frac{x^2y^3}{6} - \frac{x^2y^2}{2} \tag{13}$$

This has the following exact solution:

$u(x,y) = xy + x$, with IC : $u(0,y) = 0$.

Applied present method and solved eq.(13) for different value of ε. Table (3) given mean square error of the designer network,

Table 3: mean square error of the network for example 3

Mse						
€→	2^{-0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}
mse	$7.2729e^{-9}$	$6.9423e^{-8}$	$1.5463e^{-8}$	$3.8254e^{-8}$	$5.5524e^{-8}$	$6.8538e^{-8}$

8 Conclusion

In this paper, we design fast feed forward neural network to solve some of two dimension integro-differential and integral equation which have the singularly perturbed. A fast and efficient algorithm (LM) for FFNN with one hidden layer has been presented and tested on several examples. Through the comparison with exact solutions show that the ANN method has good accuracy and efficiency and results obtained using the ANN method is with low error.
Moreover, our method is quite general and can be used in a wide class of integral and integro-differential equation.

References

[1] M.H. Alnasr, Numerical treatment of singularly perturbed Volterra integral equations. Ph.D thesis,Egypt,1997.
[2] J.P. Kauthen. Implicit Runge-Kutta methods for some integro differential-algebraic equations. Appl. Numer. Math., 13(1993):125–134.
[3] J.P. Kauthen. Implicit Runge-Kutta methods for singularly perturbed integro-differential equations. Appl. Numer.Math., 18(1995):201–210.
[4] A. P. Orsi. Product integration for Volterra integral equations of the second kind with weakly singular kernels. Math.Comput., 65(1996):1201–1212.
[5] S.E. El-Gendi.Chebyshev solution of differential, integral and integro-differ ential equations.Comput. J., 12(1969):282–287.

[6] Q. Hu Geometric meshes and their application to Volterra integral equations with singularities. SIAM J. Numer.Anal., 18(1998):151–164.
[7] V. Horvat and M. Rogina. Tension spline collocation methods for singularly perturbed Volterra integro-differential and Volterra integral equations. J. Comput. Appl. Math., 140(2002):381–402.
[8] A. Salama and S.A. Bakr. Difference schemes of exponential type for singularly perturbed Volterra integro- differential problems. Appl. Math. Model., 31(2007):866–879.
[9] J. I. Ramos. Piecewise-quasilinearization techniques for singularly perturbed Volterra integro-differential equations. Appl.Math.Comput.,188(2007):1221–1233.
[10] S. Haykin , " Neural networks: A comprehensive foundation",1993.
[11] R. M. Hristev , " The ANN Book ", Edition 1, 1998 .
[12] B. M. wilamowski and S. Iplikei, an Algorithm for fast convergence in training neural network, IEEE.(2001):1778-1782.