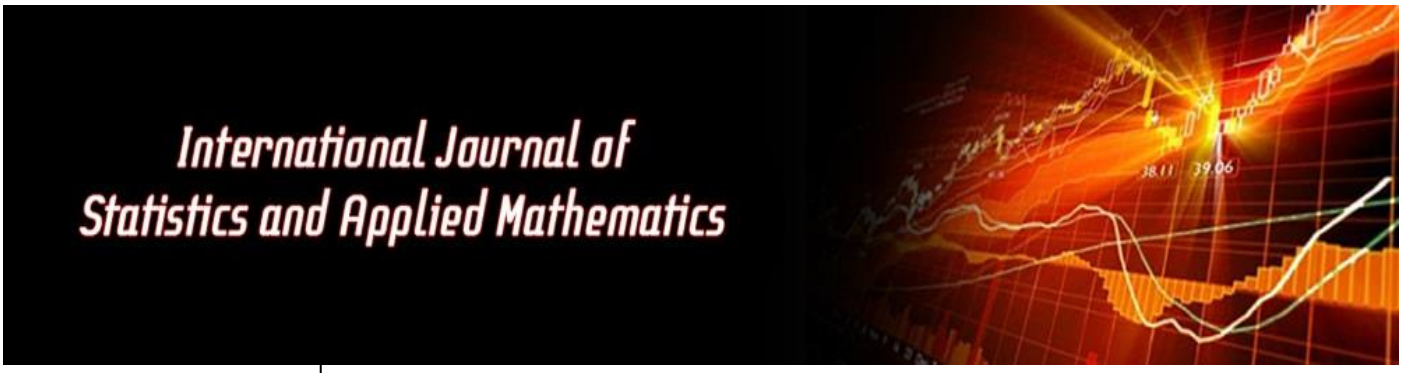
International Journal of Statistics and Applied Mathematics 2016; 2(1): 50-56



ISSN: 2456-1452
Maths 2017; 2(1): 50-56

www.mathsjournal.com
Received: 17-11-2016
Accepted: 18-12-2016

Luma. N.M Tawfiq
Department of Mathematics, College of Education Ibn Al-Haitham, Baghdad University, Iraq

Correspondence
Luma. N.M Tawfiq
Department of Mathematics, College of Education Ibn Al-Haitham, Baghdad University, Iraq

Using collocation neural network to solve nonlinear singular differential equations

Luma. N.M. Tawfiq

Abstract
The aim of this paper is to design collocation neural network to solve second order nonlinear boundary value problems in singular ordinary differential equations.
The proposed network trained by back propagation with different training algorithms quasi-Newton, Levenberg–Marquardt, and Bayesian Regulation, were the designed network trained with those algorithms using the available experimental data as the training set and the proposed network containing a single hidden layer of five nodes. The next objective of this paper was to compare the performance of aforementioned algorithms with regard to predicting ability.

Keywords: Ann, Back propagation, Training Algorithm, ordinary differential equations

1. Introduction
Many methods have been developed so far for solving differential equations, some of them produce a solution in the form of an array that contains the value of the solution at a selected group of points [1]. Others use basis functions to represent the solution in analytic form and transform the original problem usually to a system of algebraic equations [2]. Most of the previous study in solving differential equations using artificial neural network(Ann) is restricted to the case of solving the systems of algebraic equations which result from the discretization of the domain [3]. Most of the previous works in solving differential equations using neural networks is restricted to the case of solving the linear systems of algebraic equations which result from the discretization of the domain. The minimization of the networks energy function provides the solution to the system of equations [4]. Lagaris *et al.* [5] employed two networks: a multilayer perceptron and a radial basis function network to solve partial differential equations (PDE) with boundary conditions (Dirichlet or Neumann) defined on boundaries with the case of complex boundary geometry. Mc Fall *et al.* [6] compared weight reuse for two existing methods of defining the network error function, weight reuse is shown to accelerate training of ODE, the second method outperforms the fails unpredictably when weight reuse is applied to accelerate solution of the diffusion equation. Tawfiq [7] proposed a radial basis function neural network (RBFNN) and Hopfield neural network (unsupervised training network) as a designer network to solve ODE and PDE and compared between them. Malek *et al.* [8] reported a novel hybrid method based on optimization techniques and neural networks methods for the solution of high order ODE which used three layered perceptron network. Akca *et al.* [9] discussed different approaches of using wavelets in the solution of boundary value problems (BVP) for ODE, also introduced convenient wavelet representations for the derivatives for certain functions and discussed wavelet network algorithm. Mc Fall [10] presented multi-layer perceptron networks to solve BVP of PDE for arbitrary irregular domain where he used logsig transfer function in hidden layer and pureline in output layer and used gradient decent training algorithm, also, he used RBFNN for solving this problem and compared between them. Junaid *et al.* [11] used Ann with genetic training algorithm and log sigmoid function for solving first order ODE, Zahoor *et al.* [12] has been used an evolutionary technique for the solution of nonlinear Riccati differential equations of fractional order and the learning of the unknown parameters in neural network has been achieved with hybrid intelligent algorithms mainly based on genetic algorithm (GA).

Abdul Samath *et al.* [13] suggested the solution of the matrix Riccati differential equation (MRDE) for nonlinear singular system using Ann. Ibraheem *et al.* [14] proposed shooting neural networks algorithm for solving two point second order BVP in ODEs which reduced the equation to the system of two equations of first order. Hoda *et al.* [4] described a numerical solution with neural networks for solving PDE, with mixed boundary conditions. Majidzadeh [15] suggested a new approach for reducing the inverse problem for a domain to an equivalent problem in a variational setting using radial basis functions neural network, also he used "cascade feed forward to solve two - dimensional Poisson equation with back propagation and levenberg - Marqwardt train algorithm with the architecture three layers and 12 input nodes, 18 tansig transfer function in hidden layer, and 3 linear nodes in output layer. Oraibi [16] designed feed forward neural networks (FFNN) for solving IVP of ODE. Ali [17] design fast FFNN to solve two point BVP. This paper proposed FFNN to solve two point singular boundary value problem (TPSBVP) with back propagation (BP) training algorithm.

2. Singular Boundary Value Problem

The general form of second order two point boundary value problem (TPBVP) is:

$$y'' + P(x)y' + Q(x)y = 0,\ a \le x \le b \tag{1}$$

$y(a) = A$ and $y(b) = B$, where $A, B \in R$

there are two types of a point $x_0 \in [0,1]$: Ordinary Point and Singular Point.

A function $y(x)$ is analytic at x_0 if it has a power series expansion at x_0 that converges to $y(x)$ on an open interval containing x_0. A point x_0 is an ordinary point of the ODE (1), if the functions $P(x)$ and $Q(x)$ are analytic at x_0. Otherwise x_0 is a singular point of the ODE. On the other hand if $P(x)$ or $Q(x)$ are not analytic at x_0 then x_0 is said to be a singular point [18, 19].

There is at present no theoretical work justifying numerical methods for solving problems with singular points. The main practical occurrence of such problems seems to be semi - analytic technique [20].

3. Artificial Neural Network

Ann is a simplified mathematical model of the human brain, It can be implemented by both electric elements and computer software. It is a parallel distributed processor with large numbers of connections, it is an information processing system that has certain performance characters in common with biological neural networks [21]. The arriving signals, called inputs, multiplied by the connection weights (adjusted) are first summed (combined) and then passed through a transfer function to produce the output for that neuron. The activation (transfer) function acts on the weighted sum of the neuron's inputs and the most commonly used transfer function is the sigmoid function (tansig.) [17].

There are two main connection formulas (types): feedback (recurrent) and feed forward connection. Feedback is one type of connection where the output of one layer routes back to the input of a previous layer, or to same layer. Feed forward (FFNN) does not have a connection back from the output to the input neurons [22].

There are many different training algorithms but the most often used is the Delta-rule or back propagation (BP) rule. A neural network is trained to map a set of input data by iterative adjustment of the weights. Information from inputs is fed forward through the network to optimize the weights between neurons. Optimization of the weights is made by backward propagation of the error during training phase.

The Ann reads the input and output values in the training data set and changes the value of the weighted links to reduce the difference between the predicted and target (observed) values. The error in prediction is minimized across many training cycles (iteration or epoch) until network reaches specified level of accuracy. A complete round of forward–backward passes and weight adjustments using all input–output pairs in the data set is called an epoch or iteration. If a network is left to train for too long, however, it will be over trained and will lose the ability to generalize.

In this paper, we focused on the training situation known as supervised training, in which a set of input/output data patterns is available. Thus, the Ann has to be trained to produce the desired output according to the examples.

In order to perform a supervised training we need a way of evaluating the Ann output error between the actual and the expected output. A popular measure is the mean squared error (MSE) or root mean squared error (RMSE) [23].

4. Description of the Method

In the proposed approach the model function is expressed as the sum of two terms: the first term satisfies the boundary conditions (BC) and contains no adjustable parameters. The second term can be found by using Ann which is trained so as to satisfy the differential equation and such technique called collocation neural network.

In this section we will illustrate how our approach can be used to find the approximate solution of the general form a 2^{nd} order TPSBVP:

$$x^m y''(x) = F(x, y(x), y'(x)) \tag{2}$$

where a subject to certain BC's and $m \in Z$, $x \in R$, $D \subset R$ denotes the domain and $y(x)$ is the solution to be computed.

If $y_t(x, p)$ denotes a trial solution with adjustable parameters p, the problem is transformed to a discretize form :

$$\text{Min}_p \sum_{\vec{x}_i \in \hat{D}} F(x_i, y_t(x_i, p), y_t'(x_i, p)) \tag{3}$$

subject to the constraints imposed by the BC's.

In the our proposed approach, the trial solution y_t employs a Ann and the parameters p correspond to the weights and biases of the neural architecture. We choose a form for the trial function $y_t(x)$ such that it satisfies the BC's. This is achieved by writing it as a sum of two terms:

$$y_t(x_i, p) = A(x) + G(x, N(x, p)) \quad (4)$$

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

5. Computation of the Gradient

An efficient minimization of (3) can be considered as a procedure of training the network, where the error corresponding to each input x_i is the value $E(x_i)$ which has to forced near zero. Computation of this error value involves not only the Ann output but also the derivatives of the output with respect to any of its inputs.

Therefore, in computing the gradient of the error with respect to the network weights consider a multilayer Ann with n input units (where n is the dimensions of the domain) one hidden layer with H sigmoid units and a linear output unit .

For a given input x the output of the Ann is:

$$N = \sum_{i=1}^{H} \nu_i \sigma(z_i) \quad , \text{where } z_i = \sum_{j=1}^{n} w_{ij} x_j + b_i$$

w_{ij} denotes the weight connecting the input unit j to the hidden unit i

ν_i denotes the weight connecting the hidden unit i to the output unit,

b_i denotes the bias of hidden unit i, and $\sigma(z)$ is the sigmoid transfer function (tansig.).

The gradient of Ann, with respect to the parameters of the Ann can be easily obtained as :

$$\frac{\partial N}{\partial \nu_i} = \sigma(z_i) \quad (5)$$

$$\frac{\partial N}{\partial b_i} = \nu_i \sigma'(z_i) \quad (6)$$

$$\frac{\partial N}{\partial w_{ij}} = \nu_i \sigma'(z_i)\, x_j \quad (7)$$

Once the derivative of the error with respect to the network parameters has been defined, then it is a straight forward to employ any minimization technique**.** It must also be noted, the batch mode of weight updates may be employed.

6. Illustration of the Method

In this section we describe solution of TPSBVP using Ann.

To illustrate the method, we will consider the 2nd order TPSBVP:

$$x^m d^2y(x) / dx^2 = f(x, y, y') \quad (8)$$

where $x \in [a, b]$ and the BC: $y(a) = A$, $y(b) = B$, (Dirichlet case) or $y'(a) = A$, $y'(b) = B$, (Neumann case) or $y(a)' = A$, $y(b) = B$, (Mixed case)

a trial solution can be written as:

$$y_t(x, p) = (bA - aB) / (b-a) + (B-A)x / (b-a) + (x-a)(x-b)N(x, p) \quad (9)$$

where N(x, p) is the output of a FFNN with one input unit for x and weights p.

Note that

$y_t(x)$ satisfies the BC by construction. The error quantity to be minimized is given by:

$$E[p] = \sum_{i=1}^{n} \left\{ d^2y_t(x_i, p) / dx^2 - f(x_i, y_t(x_i, p), dy_t(x_i, p) / dx) \right\}^2 \quad (10)$$

where the $x_i \in [a, b]$. Since :

$$dy_t(x, p)/dx = (B-A)/(b-a) + \{(x-a) + (x-b)\}N(x, p) + (x-a)(x-b)\frac{dN(x,\vec{p})}{dx}$$

and

$$d^2y_t(x, p)/dx^2 = 2N(x, p) + 2\{(x-a) + (x-b)\}\frac{dN(x,\vec{p})}{dx} + (x-a)(x-b)\,d^2N(x, p)/dx^2$$

It is straightforward to compute the gradient of the error with respect to the parameters p using (5) – (7). The same holds for all subsequent model problems.

7. Example

In this section we report numerical result, using a multi-layer Ann having one hidden layer with 5 hidden units (neurons) and one linear output unit. The sigmoid activation of each hidden unit is tansig, the analytic solution $y_a(x)$ was known in advance. Therefore we test the accuracy of the obtained solutions by computing the deviation: $\Delta y(x) = | y_t(x) - y_a(x) |$.

In order to illustrate the characteristics of the solutions provided by the neural network method, we provide figures displaying the corresponding deviation $\Delta y(x)$ both at the few points (training points) that were used for training and at many other points (test points) of the domain of equation. The latter kind of figures are of major importance since they show the interpolation capabilities of the neural solution which to be superior compared to other solution obtained by using other methods. Moreover, we can consider points outside the training interval in order to obtain an estimate of the extrapolation performance of the obtained numerical solution.

Example 1

Consider the following 2nd order TPSBVP:

$$y'' = \frac{5x3(5x5ey-x-5)}{4+x5} - (\frac{1}{x}+1)\,y$$

with BC: $y'(0)= 0$, $y'(1) = 1$ and $x \in [0, 1]$. This problem is an application of oxygen diffusion with the analytic solution is: $y_a(x) = - \ln(x^5+ 4)$, according to (9) the trial neural form of the solution is taken to be :

$$y_t(x) = x + x(x - 1)N(x, p) .$$

The Ann trained using a grid of ten equidistant points in [0, 1]. Figure(1) display the analytic and neural solutions with different training algorithm. The neural results with different types of training algorithm such as: Levenberg – Marquardt (trainlm), quasi – Newton (trainbfg), Bayesian Regulation (trainbr) introduced in table (1) and its errors given in table (2), table (3) gives the performance of the train with epoch and time and table(4) gives the weight and bias of the designer network.

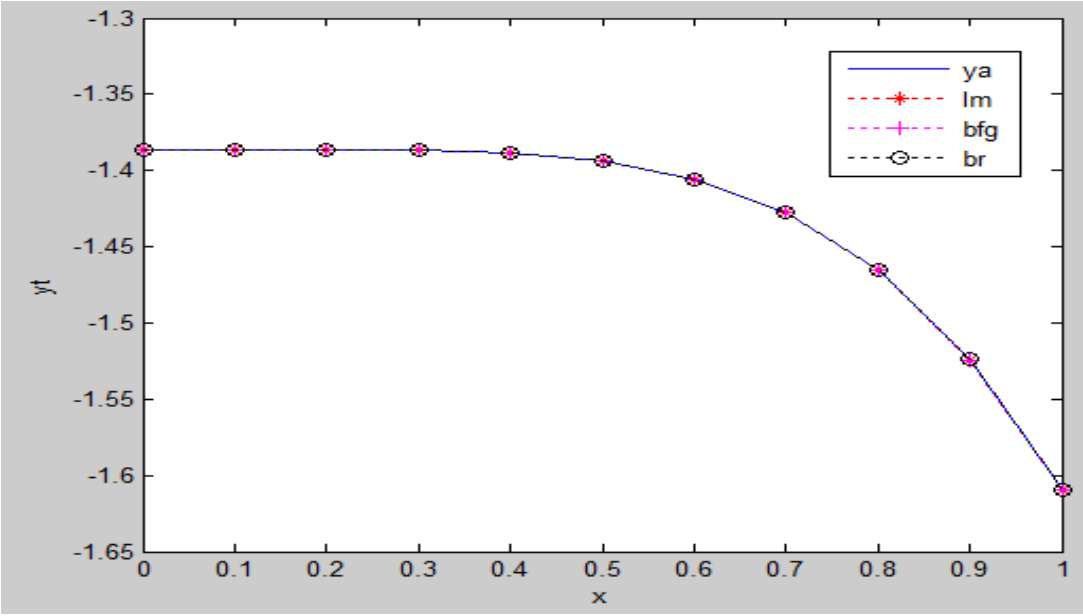


Fig 1: analytic and neural solution of example 1 using: trainbfg, trainbr, and trainlm training algorithm.

Table 1: Analytic and Neural solution of example 1

Input	Analytic solution	Out of suggested FFNN $y_t(x)$ for different training algorithm		
x	$y_a(x)$	Trainlm	Trainbfg	Trainbr
0.0	-1.38629436111989	-1.38629436111989	-1.38622827416621	-1.38629436054424
0.1	1.38629686111677-	1.38632122985118-	-1.38629686111235	1.38631996866980-
0.2	1.38637435792006-	1.38638280837857-	-1.38637435791784	1.38637437846359-
0.3	1.38690167666647-	1.38690167666647-	-1.38690167667138	1.38689369025362-
0.4	1.38885108990158-	1.38885108990158-	-1.39407650157754	1.38885075885693-
0.5	1.39407650156195-	1.39407650156195-	-1.39407650157754	1.39407759001604-
0.6	1.40554781804178-	1.40554781804178-	-1.40556126073782	1.40554604630065-
0.7	1.42745309893576-	1.42744310898688-	-1.42745309894416	1.42745483762182-
0.8	1.46503160165727-	1.46503160165727-	-1.46497360379974	1.46503054287307-
0.9	-1.52398677218731	1.52412414565470-	-1.52398677220623	1.52398714602479-
1.0	1.60943791243410-	1.60943791243410-	1.60943791244646-	1.60943785303821-

Table 2: Accuracy of solutions for example 1

The error $E(x) = \| y_t(x) - y_a(x) \|$ where $y_t(x)$ computed by the following training algorithm		
t r a i n l m	t r a i n b f g	Trainbr
0	6.60869536770470e-05	5.75652414624983e-10
2.43687344108068e-05	4.41757741498350e-12	2.31075530392477e-05
8.45045851050053e-06	2.22177831687986e-12	2.05435248812336e-08
0	4.91340301778109e-12	7.98641284549539e-06
2.22044604925031e-16	1.80729254006806e-07	3.31044651025181e-07
0	1.55979673621687e-11	1.08845409418912e-06
0	1.34426960436418e-05	1.77174112514500e-06
9.98994888146143e-06	8.40572056404199e-12	1.73868606045957e-06
2.22044604925031e-16	5.79978575339091e-05	1.05878420542105e-06
0.000137373467388624	1.89235294101309e-11	3.73837484746176e-07
0	1.23625554238060e-11	5.93958928707394e-08

Table 3: the performance of the train with epoch and time

Train Fcn	Performance of train	Epoch	Time	MSE
Trainlm	7.04e-33	811	0:00:17	1.e-009
Trainbfg	1.25e-22	401	0:00:14	7.1927e-010
Trainbr	9.69e-13	4202	0:01:31	4.e-011

Table 4: Weight and bias of the network for different training algorithm

Weights and bias for trainlm		
Net. IW{1,1}	Net. LW{2,1}	Net. B{1}
0.7094	0.1626	0.5853
0.7547	0.1190	0.2238
0.2760	0.4984	0.7513
0.6797	0.9597	0.2551
0.6551	0.3404	0.5060

Weights and bias for trainbfg		
Net. IW{1,1}	Net. LW{2,1}	Net. B{1}
0.4265	0.3864	0.9452
0.8363	0.7756	0.7842
0.7314	0.7343	0.7056
0.3600	0.4303	0.1093
0.4542	0.6938	0. 3899

Weights and bias for trainbr		
Net. IW{1,1}	Net. LW{2,1}	Net. B{1}
0.5916	0.6508	0.5158
0.2033	0.7960	0.8378
0.6359	0.2334	0.9208
0.7984	0.6008	0.4982
0.5017	0.1125	0.2776

Rasheed [20] solved this problem using semi-analytic technique and gave the following series solution:
$P_{15} = -0.1142318896x^{15} + 0.7852891723\ x^{14} - 2.23317271x^{13} + 3.413936838x^{12} - 3.087794733\ x^{11} + 1.697854523x^{10} - 0.4991720457x^{9} + 0.06414729306x^{8} - 0.25x^{5} - 1.386294361119891.$
Abukhaled *et al.* in [24] applying L'Hopital's rule to overcome the singularity at x = 0, then used the modified spline approach and get maximum error $7.79e^{-4}$ and resolution this problem using finite difference method then gives maximum error $1.46e^{-3}$**.**

Example 2

Consider the following 2nd order TPSBVP:

$y'' + (\frac{1}{x})\, y' + \exp(y) = 0$

with BC: $y'(0) = 0$, $y(1) = 0$ and $x \in [0, 1]$. The analytic solution is:

$y_a(x) = 2\ln(\frac{1.171572}{0.171572\, x2+1})$ according to (9) the trial neural form of the solution is :

$y_t(x) = x\,(x - 1)\,N(x, p)$.

The Ann trained using a grid of ten equidistant points in [0, 1]. Figure (2) display the analytic and neural solutions with different training algorithm. The neural results with different types of training algorithm such as : Levenberg – Marquardt (trainlm), quasi – Newton (trainbfg) , Bayesian Regulation (trainbr) introduced in table (5) and its errors given in table (6), table(7) gives the performance of the train with epoch and time and table(8) gives the weight and bias of the designer network.

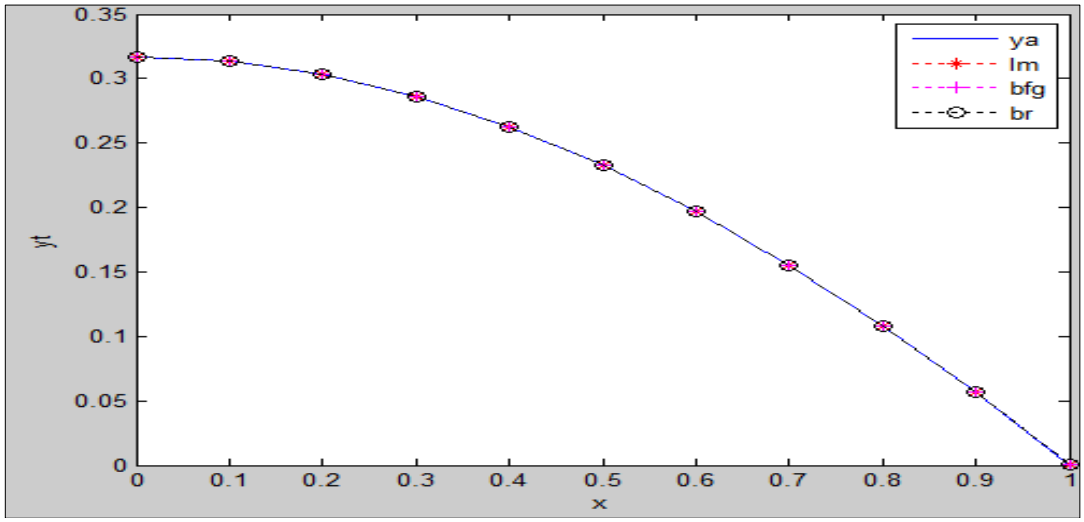


Fig 2: analytic and neural solution of example 2 using : trainbfg, trainbr, and trainlm training algorithm.

Table 5: Analytic and Neural solution of example 2

Input	Analytic solution	Out of suggested FFNN $y_t(x)$ for different training algorithm		
x	$y_a(x)$	Trainlm	Trainbfg	Trainbr
0.0	0.316692873488478	0.316692873488478	0.316692873593466	0.316692899130540
0.1	0.313264373820885	0.313262811550267		0.313264306858849
0.2	0.303013998223066	0.303013998223067	0.313264373864310	0.303028528687677
0.3	0.286045926302517	0.286045926302517	0.303014593887369	0.286046074608241
0.4	0.262529905901733	0.262530990045994	0.286045926049355	0.262522981305519
0.5	0.232695709349328	0.232695709349328	0.262529905881834	0.232695011612825
0.6	0.196825905063828	0.196821841951117	0.232695709129980	0.196827448735928
0.7	0.155247403760149	0.155242409416170	0.196825979077456	0.155245600284741
0.8	0.108322278769987	0.108322278769987	0.155247403639217	0.108323556901505
0.9	0.0564383532194616	0.0564383532194616	0.108319108185186	0.0564378293409543
1.0	0	3.51599719881610e-17	0.0564383532998103	9.64002488537051e-08

Table 6: Accuracy of solutions for example 2

The error $E(x) = \| y_t(x) - y_a(x) \|$ where $y_t(x)$ computed by the following training algorithm				
t r a i n l m		t r a i n b f g		trainbr
0		1.04988684412888e-10		2.56420623623299e-08
1.56227061820502e-06		4.34248748071298e-11		6.69620366933188e-08
5.55111512312578e-17		5.95664302882604e-07		1.45304646100297e-05
1.11022302462516e-16		2.53161991370376e-10		1.48305723868258e-07
1.08414426169823e-06		1.98982497146005e-11		6.92459621315367e-06
5.55111512312578e-17		2.19348261776275e-10		6.97736503130209e-07
4.06311271167192e-06		7.40136278376546e-08		1.54367209959094e-06
4.99434397926990e-06		1.20932069913593e-10		1.80347540823522e-06
4.16333634234434e-17		3.17058480124588e-06		1.27813151762357e-06
6.93889390390723e-18		8.03486860379010e-11		5.23878507345532e-07
3.51599719881610e-17		8.76605424371351e-05		9.64002488537051e-08

Table 7: The performance of the train with epoch and time

TrainFcn	Performance of train	Epoch	Time	MSE
Trainlm	2.48e-33	99	0:00:03	3.687e-012
Trainbfg	5.41e-27	1565	0:00:48	6.2957e-010
Trainbr	8.96e-13	2561	0:00:41	2.1858e-011

Table 8: Weight and bias of the network for different training algorithm

Weights and bias for trainlm		
Net.IW{1,1}	Net.LW{2,1}	Net.B{1}
0.2057	0.4845	0.2374
0.3883	0.1518	0.5309
0.5518	0.7819	0.0915
0.2290	0.1006	0.4053
0.6419	0.2941	0.1048

Weights and bias for trainbfg		
Net.IW{1,1}	Net.LW{2,1}	Net.B{1}
0.8397	0.8799	0.3798
0.3717	0.0441	0.9797
0.8282	0.6867	0.3990
0.1765	0.7338	0.4402
0.1295	0.4372	0.1568

Weights and bias for trainbr		
Net.IW{1,1}	Net.LW{2,1}	Net.B{1}
0.8440	0.7985	0.3755
0.1962	0.9875	0.9737
0.3039	0.1590	0.9723
0.4833	0.2369	0.6437
0.3378	0.7022	0.8601

Ramos in [25] solved this example using C^1- linearization method and gave the absolute error 3.118446e–03, also, Kumar in [26], solved this example by the three-point finite difference technique and gave the absolute error 2.0e–04.

8. Conclusion

From the above problems it is clear that the network which proposed can handle effectively TPSBVP and provide accurate approximate solution throughout the whole domain and not only at the training points. As evident from the tables, the results of proposed network are more precise as compared to method suggested in [20, 24, 25, 26].

In general, the practical results on Ann show which contain up to a few hundred weights the Levenberg-Marquardt algorithm (trainlm) will have the fastest convergence, then trainbfg and then trainbr. However, "trainbr" it does not perform well on function approximation on problems. The performance of the various algorithms can be affected by the accuracy required of the approximation.

9. References

1. Mohammed KM. On Solution of Two Point Second Order Boundary Value Problems by Using Semi-Analytic Method, MSc thesis, University of Baghdad, College of Education – Ibn – Al- Haitham, 2009.
2. LeVeque RJ. Finite Difference Methods for Differential Equations, University of Washington, AMath 585, Winter Quarter, 2006.
3. Agatonovic-Kustrin S, Beresford R. Basic concepts of artificial neural network modeling and its application in pharmaceutical, research. J. Pharm. Biomed. Anal., 2000; 22:717-727.
4. Hoda SA, Nagla HA. On Neural Network Methods for Mixed Boundary Value Problems, International Journal of Nonlinear Science. 2011; 11(3):312-316.
5. Lagaris IE, Likas AC, Papageorgiou DG. Neural-Network Methods for Boundary Value Problems with Irregular Boundaries, IEEE Transactions on Neural Networks. 2000; 11(5):1041-1049.
6. Mc Fall KS, Mahan JS. Investigation of Weight Reuse in Multi-Layer Perceptron Networks for Accelerating the Solution of Differential Equations, IEEE International Conference on Neural Networks. 2004; 14:109 -114.
7. Tawfiq LNM. Design and Training Artificial Neural Networks for Solving Differential Equations, PhD thesis, University of Baghdad , College of Education – Ibn – Al- Haitham, 2004.
8. Malek A, Beidokhti RS. Numerical solution for high order differential equations using a hybrid neural network-optimization method, Applied Mathematics and Computation. 2006; 183:260-271.
9. Akca H, Al-Lail MH, Covachev V. Survey on Wavelet Transform and Application in ODE and Wavelet Networks, Advances in Dynamical Systems and Applications. 2006; 1(2):129-162.
10. Mc Fall KS. An Artificial Neural Network Method for Solving Boundary Value Problems with Arbitrary Irregular Boundaries, PhD Thesis, Georgia Institute of Technology, 2006.
11. Junaid A, Raja MAZ, Qureshi IM. Evolutionary Computing Approach for The Solution of Initial value Problems in Ordinary Differential Equations, World Academy of Science, Engineering and Technology. 2009; 55:578-581.
12. Zahoor RMA, Khan JA, Qureshi IM. Evolutionary Computation Technique for Solving Riccati Differential Equation of Arbitrary Order, World Academy of Science, Engineering and Technology. 2009; 58:303-309.
13. Abdul Samath J, Kumar PS, Begum A. Solution of Linear Electrical Circuit Problem using Neural Networks, International Journal of Computer Applications. 2010; 2(1):6-13.
14. Ibraheem KI, Khalaf BM. Shooting Neural Networks Algorithm for Solving Boundary Value Problems in ODEs, Applications and Applied Mathematics: An International Journal. 2011; 6(11):1927-1941.
15. Majidzadeh K. Inverse Problem with Respect to Domain and Artificial Neural Network Algorithm for the Solution, Mathematical Problems in Engineering. 2011, 1-16.
16. Oraibi YA. Design Feed Forward Neural Networks For Solving Ordinary Initial Value Problem, MSc Thesis, University of Baghdad, College of Education Ibn Al-Haitham, 2011.
17. Ali MH. Design Fast Feed Forward Neural Networks to Solve Two Point Boundary Value Problems, MSc Thesis, University of Baghdad, College of Education Ibn Al-Haitham, 2012.
18. Rachůnková I, Staněk S, Tvrdý M. Solvability of Nonlinear Singular Problems for Ordinary Differential Equations, New York, USA, 2008.
19. Shampine J. Kierzenka, Reichelt MW. Solving Boundary Value Problems for Ordinary Differential Equations in Matlab with bvp4c, 2000.
20. Rasheed HW. Efficient Semi-Analytic Technique for Solving Second Order Singular Ordinary Boundary Value Problems, MSc. Thesis, University of Baghdad, College of Education - Ibn - Al - Haitham, 2011.
21. Galushkin IA. Neural Networks Theory, Berlin Heidelberg, 2007.
22. Mehrotra K, Mohan CK, Ranka S. Elements of Artificial Neural Networks, Springer, 1996.
23. Ghaffari A, Abdollahi H, Khoshayand MR, Soltani Bozchalooi I, Dadgar A, Rafiee-Tehrani M. Performance comparison of neural network training algorithms in modeling of bimodal drug delivery, International Journal of Pharmaceutics. 2006; 327:126-138.
24. Abukhaled M, Khuri SA, Sayfy A. A Numerical Approach for Solving a Class of Singular Boundary Value Problems Arising in Physiology, International Journal of Numerical Analysis and Modeling. 2011; 8(2):353-363.
25. Ramos JI. Piecewise quasilinearization techniques for singular boundary value problems, Computer Physics Communications. 2004; 158:12-25.
26. Kumar M. A three-point finite difference method for a class of singular two-point boundary value problems, J Comput. Appl. Math. 2002; 145:89-97.