Mathematical Theory and Modeling www.iiste.org
ISSN 2224-5804 (Paper) ISSN 2225-0522 (Online)
Vol.6, No.1, 2016

Design Collocation Neural Network to Solve Singular Perturbation Problems for Partial Differential Equations

Khalid. M. Mohammed. Al-Abrahemee
Department of Mathematics, College of Education, University AL-Qadisiyha

Abstract

he aim of this paper is to design neural network to present a method to solve Singular perturbation problems (SPP) for Partial Differential Equations (PDE's) with initial and boundary conditions by using network having one hidden layer with 5 hidden units (neurons) and one linear output unit, the sigmoid activation of each hidden units is tansigmoid. The neural network trained by the back propagation with different algorithms such as quasi-Newton, Levenberg-Marquardt, and Bayesian Regulation. 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; Partial Differential Equations; Neural network; QuasiNewton; Levenberg-Marquardt, Bayesian regulation.

1.Introduction:

A singular perturbation problem is a problem which depends on a parameter (or parameters) in such a way that solutions of the problem behave nonuniformly as the parameter tends toward some limiting value of interest. The nature of the nonuniformity can vary from problem to problem. Such singular perturbation problems involving differential equations arise in many areas of interest including applied mechanics, fluid dynamics, celestial mechanics, wave propagation (electromagnetic, acoustic, etc.), quantum theory, aerodynamics, electrical networks, elasticity and statistical mechanics. In practice one seeks a uniformly valid, easily interpretable approximation to the nonuniformly behaving solution.

the methods can be used to study various singular perturbation boundary value problems involving partial differential equations. which was studied by [5] using a composite expansion approach, and the equation which was studied by [3]. ([7]1968, pp. 447-459) gives a clear exposition of these results. (The results can be easily given in terms of a direct multivariable approach.) Finally, O'Malley in [6] points out that the multivariable approach can be used to study the Oseen partial differential equations for the flow of a slightly viscous incompressible fluid past a semi-infinite fiat plate at zero angle of attack.

Singularly perturbed problems (SPP) in partial differential equations (PDE) are characterized by the presence of a small parameter that multiplies the highest derivative. These problems are stiff. Many methods have been developed so far solving Singularly perturbed boundary value problems (SPBVP) , nowadays there is a new way of computing denominated artificial intelligence which through different methods is capable of managing the imprecision's 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 the 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.

2. Singularly Perturbed Problems

In this section we consider a system of partial differential equations (to gather with appropriate boundary conditions) in which the highest derivative is multiplied by a small, positive parameter (usually denoted by $\varepsilon << 1$). In what follows we give the general form of the 2nd order singularly perturbed problems (SPPs) of partial differential equations (PDE) are:

$$F(x, y, \varepsilon, \frac{\partial\Psi(x,y)}{\partial x}, \frac{\partial\Psi(x,y)}{\partial y}, \frac{\partial^2\Psi(x,y)}{\partial x^2}, \frac{\partial^2\Psi(x,y)}{\partial x\partial y}, \frac{\partial^2\Psi(x,y)}{\partial y^2}) = 0\,,\ x\in[0,1]\,,\ y\in[0,1] \text{ and } 0 < \varepsilon << 1 \qquad (1)$$

with Dirichlet BC's or mixed BC's .[4],[8]

3. Artificial Neural Network

An 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. ANN have been developed as generalizations of mathematical models of human cognition or neural biology, based on the assumptions:

1. Information processing occurs at many simple elements called neurons that is fundamental the operation of ANN's.

2. Signals are passed between neurons over connection links.

3. Each connection link has an associated weight which, in a typical neural net, multiplies the signal transmitted .

4. Each neuron applies an activation function (usually nonlinear) to its net input (sum of eighted input signals) to determine its output signal [9].

There are two main connection formulas (types):feedback(recurrent) and feed-forward connections. Feedback is one type of connection where the output of one layer routes back to the input of a previous layer , or to the same layer. Feed-forward neural network(FFNN) does not have a connection back from the output to the input neurons. There are many different training algorithms, but the most often used training algorithm is the back propagation(BP) rule. ANN is trained to map a set of input data by iterative adjustment of the weights. Information from inputs is feedforward 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. In order to perform a supervised training we need a way of evaluating the ANN output error between the actual and the expected outputs .A popular measure is the mean squared error MSE) or root mean squared error(RMSE) [10]

4.Description of The Method

In this section we will illustrate how our approach can be used to find the approximate solution of the general form a second order of singular perturbation problems (S.P.P)

$$G(\overrightarrow{x}, \varepsilon, \Psi(\overrightarrow{x}), \nabla\, \Psi(\overrightarrow{x}) \nabla^2\, \Psi(\overrightarrow{x})) = 0 \qquad (2)$$

Where a subject to certain boundary conditions (BC's) (for instance Dirichlet and / or Neumann conditions) and $0 < \varepsilon << 1$, $(\overrightarrow{x}) = (x_1, x_2, \dots \dots, x_n) \in R^n, D \subset R^n$ denotes the domain and $\Psi(\overrightarrow{\mathrm{x}})$ is the solution to be computed.

To obtain a solution to the above differential equation, the collocation method is adopted which assumes a discretization of the domain D and its boundary S into a set points $\widehat{D}$ and $\hat{S}$, respectively. The problem is then transformed into the following system of equations:

$$G(\overrightarrow{xi}, \varepsilon, \Psi_{\mathrm{t}}(\overrightarrow{xi}), \nabla\, \Psi_{\mathrm{t}}(\overrightarrow{xi}), \nabla^2\Psi_{\mathrm{t}}(\overrightarrow{xi})) = 0 \quad \forall x_i \in \widehat{D} \qquad (3)$$

Subject to the constraints imposed by the BC's .

If $\Psi_t(\overrightarrow{x_i}\, \vec{p})$ denotes a trial solution with adjustable parameters $\overrightarrow{p}$, the problem is transformed to a discretize form

$$\mathrm{Min}_{\vec{p}} \quad \textstyle\sum_{\overrightarrow{xi} \in \widehat{D}} ((G(\overrightarrow{xi}, \varepsilon, \Psi_{\mathrm{t}}(\overrightarrow{xi}, \vec{p}), \nabla\Psi_{\mathrm{t}}(\overrightarrow{xi}, \vec{p}), \nabla^2\Psi_{\mathrm{t}}(\overrightarrow{xi}, \vec{p})))^2 \qquad (4)$$

Subject to the constraints imposed by the BC's .

In the our proposed approach, the trial solution Ψ_{t} employs a feed forward neural network and the parameters $\vec{p}$ correspond to the weights and biases of the neural architecture. We choose a form for the trial function $\Psi_{\mathrm{t}}(\overrightarrow{xi})$ such that it satisfies the BC's. This is achieved by writing it as a sum of two terms.

$$\Psi_{\mathrm{t}}(\overrightarrow{xi}) = \mathrm{A}(\vec{x}) + F(\vec{x}, N(\vec{x}, \vec{p}))$$

Where $N(\vec{x}, \vec{p})$ is a single-output feed forward neural network with parameters $\vec{p}$ and n input units fed with the input vector $\vec{x}$.

The term $A(\vec{x})$ contains no adjustable parameters and satisfies the boundary conditions. The second term F is constructed so as not to contribute to the BC's , since $\Psi_t(\vec{x})$ 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. Our numerical result shows that our approach, which based on the above formulation is very effective and can be done in reasonable computing time to compute an accurate solutions.

4. Illustration of The Method

We will consider a two - dimensional singular perturbation problems of partial differential equation (S.P.P).

$$\varepsilon \; \frac{\partial^2\Psi(x,y)}{\partial x^2} = f(\frac{\partial\Psi(x,y)}{\partial x}, \frac{\partial\Psi(x,y)}{\partial y}, \frac{\partial^2\Psi(x,y)}{\partial y^2}, \frac{\partial^2\Psi(x,y)}{\partial x\partial y}, x, y\,) \qquad (5)$$

x ∈ [0 ,1] , y ∈ [0 ,1] with Dirichlet BC:

$$\Psi(0\,,y) = f_0(y), \Psi\,(1\,,y) = f_1(y), \Psi(x\,,0) = g_0(x) and\; \Psi\,(x\,,1) = g_1(x)$$

$, where\; f_0\,, f_1\,, g_0\; and\; g_1$ are continuous function.

The trial solution is written as $\Psi_t(x\,,y) = A(x\,,y) + x(1 - x)\, y(1 - y)\, N(x\,,y, \vec{p})$ (6)

where A(x ,y) is chosen so as to satisfy the BC, namely:

$$A(x\,,y) = (1 - x)f_0\,(y) + x\,f_1\,(y) + (1 - y)\{g_0\,(x) - [(1 - x)g_0\,(0) + xg_0\,(1)]\} +$$
$$y\{g_1\,(x) - [(1 - x)\,g_1\,(0) + xg_1\,(1)]\} \qquad (7)$$

For mixed boundary conditions of the form:

$\Psi(0\,,y) = f_0(y), \Psi(1\,,y) = f_1(y), \Psi(x\,,0) = g_0\,(x)\; and\; (\partial\Psi(x\,,1)/\partial y) = g1(x)$

(i.e., Dirichlet on part of the boundary and Neumann elsewhere), the trial solution can be written as

$$\Psi_t\,(x\,,y) = B(x\,,y) + x(1 - x)y[N(x\,,y,\vec{p}) - N(x\,,1,\vec{p}) - [\frac{\partial N(x\,,1,\vec{p})}{\partial y}] \qquad (8)$$

And B(x ,y) is again chosen so as to satisfy the BC's:

$$B(x\,,y) = (1 - x)f_0(y) + x\,f_1(y) + g_0(x) - [(1 - x)g_0(0) + xg_0(1)]$$
$$+ y\{g1(x) - [(1 - x)g1(0) + xg1(1)]\} \qquad (9)$$

Note that the second term of the trial solution does not affect the boundary conditions.

In all the above PDE problems the error that should be minimized is given by:

$$E[\vec{p}] = \sum_{i=1}^{n}\{\frac{\partial^2\Psi(x,y)}{\partial x^2} + \frac{\partial^2\Psi(x,y)}{\partial y^2} - f(\frac{\partial\Psi(x,y)}{\partial x}, \frac{\partial\Psi(x,y)}{\partial y}, x, y, \varepsilon\,)\}^2 \qquad (10)$$

Where(x_i, y_i) are points in [0,1] ×[0,1] [10].

7. Solution of Singular perturbation of Partial Differential Equations

We will consider a two - dimensional singular perturbation problems of partial differential equation (S.P.P). with Dirichlet conditions or Neumann conditions. All the subsequent problems were defined on the domain [0,1] × [0,1] and in order to perform training we consider a mesh points obtained by considering ten equidistant points of the domain [0,1] of each variable. In analogy with the previous cases the neural network architecture was considered to be FFNN with two inputs (accepting the coordinates x and y of each point), 5 hidden units and one linear output unit, the sigmoid activation of each hidden units is tansigmoid .For every entries x and y , the input neurons makes no changes in its inputs, so the input to the hidden neurons is:

$$Net_j = x\,w_{j1} + y\,w_{j2} + B_j\,, j = 1,2,\ldots\ldots,m \qquad (11)$$

Where $w_{j1}$and w_{j2} are a weights from the input layer to the jth unit in the hidden layer, $B_j$is an jth bias for the jth unit in the hidden layer. The output in the hidden neurons is :

$$Z_j = S(net_j)\,, j = 1,2,\ldots\ldots,m. \qquad (12)$$

The output neuron make no changes in its input, so the input to the output neuron is equal to output:

$$N = \sum_{j=1}^{m} V_j\; Z_j \qquad (13)$$

8. Numerical Examples

In the section we report some numerical results and the solution of a number of model problems of SPP of PDE. In all cases we used a three-layer FFNN having two input units, one hidden layer with 5 hidden units (neurons) and one output unit, and the sigmoid activation of each hidden units is tansigmoid (ridge basis function). For each test problem, the analytical solution $\Psi_{\mathrm{a}}(\vec{x})$ was known in advance, therefore we test the accuracy of the obtained solutions by computing the deviation:

$$\Delta\Psi(\vec{x}) = |\Psi_t(\vec{x}) - \Psi_a(\vec{x})\,| \qquad (14)$$

Example (8.1): [2]

Consider the following 2nd order nonlinear of two-dimensional wave equation for the function $\Psi\,(x,y)$.

$$\varepsilon^2\;\frac{\partial^2\Psi(x,y)}{\partial x^2} - \frac{\partial^2\Psi(x,y)}{\partial y^2} = -\Psi^3$$ with BC′s (Dirishlit case):

$\Psi(0,y) = 0$, $\Psi(1,y) = 1$, $\Psi(x,0) = \tanh\frac{x}{\varepsilon\sqrt{2}}$, $\Psi(x,1) = \tanh\frac{x}{\varepsilon\sqrt{2}}$ and $0 < \varepsilon << 1$, the equation has a static solution, called a kink. The ANN trained using a grid of ten equidistant points in $x \in[0,1]$, $y \in[0,1]$ and gave $\in=10^{-6}$. Figure (8.1) display the analytic and neural solutions with different training algorithm. The neural results with different types of training algorithm such as: trainlm, trainbfg, trainbr, introduced in Table (1) and its errors gave in Table (8.2), Table (8.3) gave the initial weight and bias of the design network, Table (8.4) gave the performance of the train with epoch and time.

Table 8.1: Analytic and Neural solution of example

input	input	Analytic solution	Out of suggested FFNN $y_t(x)$ for different training algorithm		
x	**y**	**$y_a(x)$ [9]**	**Trainlm**	**Trainbfg**	**Trainbr**
0.0	**0.0**	0	-5.24231058562918e-09	-3.43419479520435e-08	0.999940691198154
0.1	**0.1**	1	0.999999996763556	0.999999994363713	0.999940691287654
0.2	**0.2**	1	1.00000000148500	1.00000001536271	0.999940691377154
0.3	**0.3**	1	0.999999957189854	0.999417385968918	0.999940691466654
0.4	**0.4**	1	1.00000023582370	0.999999974759358	0.999940691556154
0.5	**0.5**	1	0.999999718783259	1.00000159691356	0.999940691645654
0.6	**0.6**	1	0.999999966996138	1.00000032628275	0.999940691735153
0.7	**0.7**	1	1.00000008512135	0.999999930128015	0.999940691824653
0.8	**0.8**	1	1.00000008289280	0.999999807838736	0.999940691914153
0.9	**0.9**	1	1.00000003915455	0.999999770503864	0.999940692003653
1.0	**1.0**	1	0.999999984715847	0.999999759306037	0.999940692093152

Table 8.2 : Accuracy of solutions for example

The error $E(x) = \| y_t(x) - y_a(x) \|$ where $y_t(x)$ computed by the following training algorithm		
Trainlm	**Trainbfg**	**Trainbr**
5.24231058562918e	3.43419479520435e-08	0.999940691198154
3.23644355759711e	5.63628699268293e-09	5.93087123457181e-05
1.48499612606656e	1.53627075505369e-08	5.93086228459772e-05
4.28101456506624e	0.000582614031081619	5.93085333462362e-05
2.35823701899562e	2.52406424650076e-08	5.93084438463842e-05
2.81216740560808e	1.59691355694491e-06	5.93083543465323e-05
3.30038623097551e	3.26282752816098e-07	5.93082648467913e-05
8.51213530772554e	6.98719855307672e-08	5.93081753469393e-05
8.28928001794793e	1.92161263790069e-07	5.93080858470874e-05
3.91545467159915e	2.29496135872864e-07	5.93079963473464e-05
1.52841530542958e	2.40693963426963e-07	5.93079068476055e-05

Table 8.3:The performance of the train with epoch and time

Train Function	Performance of train	Epoch	Time	MSE
Trainlm	3.38-31	700	0:00:10	1.395830396203114e-14
Trainbfg	2.44-14	745	0:00:13	3.085835639098749e-08
Trainbr	3.53-9	2053	0:00:19	0.090898311008054

Table 8.4 : Initial weight and bias of the network for different training algorithm

Weights and bias for trainlm			
Net.IW{1,1}	Net.IU{1,1}	Net.LW{2,1}	Net.B{1}
0.7298	0.8908	0.4843	0.0320
0.9823	0.7690	0.8449	0.6147
0.5814	0.9283	0.2094	0.3624
0.5801	0.0170	0.5523	0.0495
0.1209	0.8627	0.6299	0.4896

Weights and bias for trainbfg			
Net.IW{1,1}	Net.IU{1,1}	Net.LW{2,1}	Net.B{1}
0.3596	0.0567	0.1040	0.5972
0.5219	0.3358	0.7455	0.2999
0.1757	0.2089	0.7363	0.1341
0.9052	0.6754	0.5619	0.2126
0.4685	0.9121	0.1842	0.8949

Weights and bias for trainbr			
Net.IW{1,1}	Net.IU{1,1}	Net.LW{2,1}	Net.B{1}
0.0830	0.6616	0.8397	0.2393
0.5170	0.1710	0.5326	0.5789
0.9386	0.5905	0.5539	0.8669
0.4406	0.9419	0.6801	0.4068
0.6559	0.4519	0.3672	0.1126

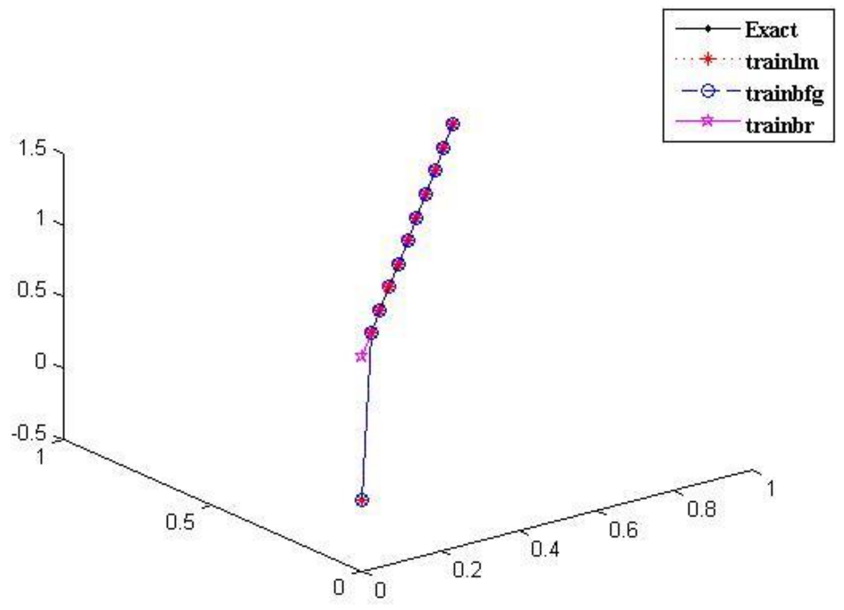

Figure 8.1: analytic and neural solution of example using : trainlm , trainbfg and trainbr training algorithm

Example (8.2) [1]

Consider the nonlinear singular perturbation problems of partial differential equation $\varepsilon x {\Psi^2}_{xx} = x^3 - \Psi_y$ where $\Psi = \Psi\,(x, y)$ with BC′s (mixed boundary conditions case): $\Psi(0, y) = 0\,, \Psi(1, y) = tanhy\,, \Psi(x, 0) = 0\,, \Psi_x\,(x, 1) = 2.2848x^2 and\ \ 0 < \varepsilon \ll 1$, whose analytical form is $\Psi(x) = x^3\ \ tanhy$. The ANN trained using a grid of ten equidistant points in $x \in [0,1]$, $y \in [0,1]$ and gave $\varepsilon = 1/36$. Figure (8.2) display the analytic and neural solutions with different training algorithm. The neural results with different types of training algorithm such as: trainlm, trainbfg, trainbr, introduced in Table (8.5) and its errors gave in Table (8.6), Table (8.7) gave the initial weight and bias of the design network, Table (8.8) gave the performance of the train with epoch and time.

Table 8.1: Analytic and Neural solution of example

input	**input**	**Analytic solution**	**Out of suggested FFNN $y_t(x)$ for different training algorithm**		
x	**y**	**$y_a(x)$**	**Trainlm**	**Trainbfg**	**Trainbr**
0.0	**0.0**	0	0	0.000224407840321146	4.56688528378790e-08
0.1	**0.1**	9.96679946249559e-05	0.000152482739437603	9.97397206456204e-05	9.96281078508055e-05
0.2	**0.2**	0.00157900256179923	0.00157900256179935	0.00157897241662771	0.00141383040288997
0.3	**0.3**	0.00786544053619296	0.00786544053619307	0.00786548252912827	0.00773922476914355
0.4	**0.4**	0.0243167335843344	0.0243167335843344	0.0242859231890898	0.0243135236274035
0.5	**0.5**	0.0577646446575012	0.0577646446575011	0.0577650918700320	0.0577777918795295
0.6	**0.6**	0.116002706471576	0.116044083480389	0.116002672980570	0.115979026806775
0.7	**0.7**	0.207298147551187	0.207298147551187	0.207089632491337	0.207322652879927
0.8	**0.8**	0.339986826377139	0.339750466600117	0.339593008636608	0.339971096178481
0.9	**0.9**	0.522181147375089	0.522214715517693	0.522181612636510	0.522187234157924
1.0	**1.0**	0.761594155955765	0.761594155955765	0.761594439747984	0.761593023636695

Table 8.2 : Accuracy of solutions for example

The error $E(x) = \| y_t(x) - y_a(x) \|$ where $y_t(x)$ computed by the following training algorithm		
Trainlm	**Trainbfg**	**Trainbr**
0	0.000224407840321146	4.56688528378790e-08
5.28147448126467e-05	7.17260206645198e-08	3.98867741503440e-08
1.17310675062932e-16	3.01451715215440e-08	0.000165172158909262
1.11022302462516e-16	4.19929353165371e-08	0.000126215767049406
2.77555756156289e-17	3.08103952445793e-05	3.20995693092047e-06
1.45716771982052e-16	4.47212530743646e-07	1.31472220283105e-05
4.13770088135701e-05	3.34910053439996e-08	2.36796648009352e-05
1.94289029309402e-16	0.000208515059850062	2.45053287400154e-05
0.000236359777022188	0.000393817740530766	1.57301986578040e-05
3.35681426043566e-05	4.65261420834473e-07	6.08678283509079e-06
3.33066907387547e-16	2.83792219168966e-07	1.13231907017397e-06

Table 8.3:The performance of the train with epoch and time

Train Function	Performance of train	Epoch	Time	MSE
Trainlm	5.20-33	274	0:00:3	5.590383501806196e-09
Trainbfg	4.03-30	266	0:00:4	2.271632801249358e-08
Trainbr	1.81-10	1005	0:00:12	4.076584251008042e-09

Table 8.4 : Initial weight and bias of the network for different training algorithm

Weights and bias for trainlm			
Net.IW{1,1}	**Net.IU{1,1}**	**Net.LW{2,1}**	**Net.B{1}**
0.0862	0.3664	0.2057	0.4845
0.3692	0.6850	0.3883	0.1518
0.5979	0.7894	0.5518	0.7819
0.3677	0.2060	0.2290	0.1006
0.0867	0.7719	0.6419	0.2941

Weights and bias for trainbfg			
Net.IW{1,1}	**Net.IU{1,1}**	**Net.LW{2,1}**	**Net.B{1}**
0.1749	0.1386	0.0684	0.4306
0.5989	0.9011	0.4363	0.9616
0.9394	0.2212	0.1739	0.7624
0.4827	0.3760	0.0261	0.0073
0.5238	0.2649	0.9547	0.6800

Weights and bias for trainbr			
Net.IW{1,1}	**Net.IU{1,1}**	**Net.LW{2,1}**	**Net.B{1}**
0.5486	0.0487	0.7283	0.3164
0.5527	0.2748	0.1758	0.6996
0.2415	0.2431	0.3604	0.6253
0.1542	0.9564	0.1888	0.5431
0.9357	0.8187	0.0012	0.4390

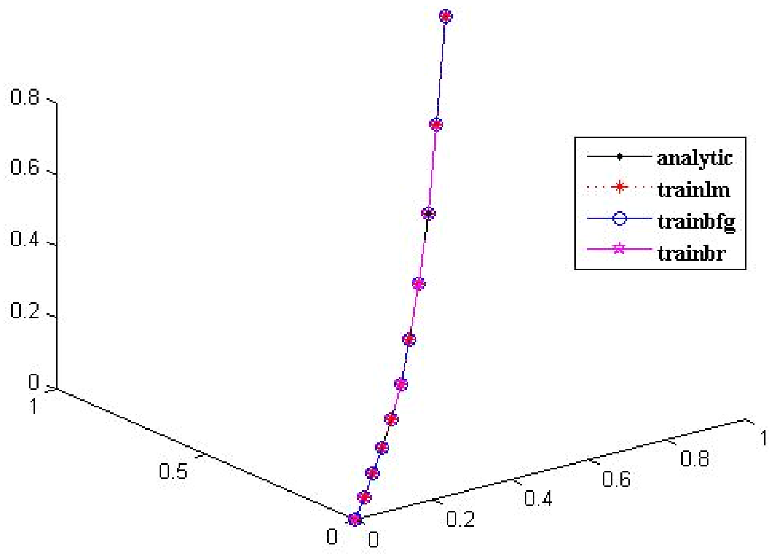

Figure 8.1: analytic and neural solution of example using : trainlm , trainbfg and trainbr training algorithm

9. Conclusion

This paper present new technique to solve 2nd order of two diminution of singular perturbed problems of partial differential equations by using artificial neural network which have the singularly perturbed ,the suggested architecture of the ANN is efficient and more accurate than other numerical method and the practical results 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 perform well on function approximation on problems, in contrast to his performance in the solution of singular perturbation problems of ordinary differential equations. The performance of the various algorithms can be affected by the accuracy required of the approximation.

References

[2] Carl M. Bender, a novel approach to Boundary Layer Problems,1989.

[1] Abdul - Majid Wazwaz, Partial Differential Equations and Solitary Waves Theory, Nonlinear Physical Science Springer Dordrecht Heidel berg London N ewYor k,2009.

[3] G. E. LaTTa , Singular perturbation problems, Ph.D. thesis, Calif. Inst. of Tech., Pasadena,Calif,1951.

[4] M. Jianzhong , " Some Singular Singularly Perturbed Problem", M.Sc. Thesis, Calgary, Alberta ,1997.

[5] N. LEVINSON, The first boundary value problem $\varepsilon\nabla$u + A(x, y)u_x + B(x, y)u_y + C(x, y)u =D(x, y) for small ε Ann. Math., 51, pp. 428-445 ,1950.

[6] R. E. O'MLLFY, JR.,A boundary value problem for certain nonlinear second order differential equations with a small parameter, Arch. Rational Mech. Anal., 29, 1968.

[7] R. E. O'MLLFY, JR., Topics of singular perturbation, Advances in Math., 2, pp. 365-470 ,1968.

[8] Roos Hans-G., Stynes Martin And Tobiska Lutz, " Robust Numerical Methods for Singularly Perturbed Differential Equations" , Computational Science & Engineering, Vol. 2012, 2012

[9] Tawfiq L . N. M. , Ali M. H., " Fast Feed Forward Neural Networks to Solve Boundary Value Problems", Lap lambert Academic Publishing , 2012.

[10] Tawfiq,L.N.M., On Design and Training of Artificial Neural Networks for Solving Differential Equations ".PH.D Thesis,52-65,2004.