Advances in the Theory of Nonlinear Analysis and its Applications 7 (2023) No. 4, 76-88.
https://doi.org/10.17762/atnaa.v7.i4.284
Available online at www.https://atnaea.org/
Research Article

Advances in the Theory of Nonlinear Analysis and its Applications

ISSN: 2587-2648 Peer-Reviewed Scientific Journal

Novel Neural Network Based on New Modification of BFGS Update Algorithm for Solving Partial Differential Equations

Muna H. Ali[a,b], Luma N. M. Tawfiq[a]

[a] *Department of Mathematics, College of Education for Pure Science Ibn Al-Haitham, University of Baghdad, Baghdad, Iraq*
[b] *Department of Mathematics, College of Education for Pure Sciences, University of Anbar, Anbar Iraq*

Abstract

In this article, an effective neural network is created using unconstrained optimization the brand-new BFGS training algorithm. The fourth order nonlinear partial differential equation is mathematically modeled with feed-forward artificial neural network with some adaptive parameters. The network is trained by new modification of BFGS method to avoid some troubles occurs when the network trained by current BFGS. The conventional updated Hessian approximations approach needed significant memory, storage, and cost computing for each iteration. One of these update's novel features is its ability to estimate the 2^{nd} order curvature of the goal function (energy functions) with high order precision while using the provided gradient and function value data. It is shown that the global convergence properties of the suggested modification, there is a parameter ρ in the update formulae which ranges from zero to one. The numerical experiments demonstrate that the improved BFGS update will be more accurate and more effective than the traditional BFGS methods. The proposed algorithm has well properties such: it has global convergence for energy function which is convex functions; also to get optimal step length we used a nonmonotone line search technique to modify the effectiveness of the proposed algorithm. Finally, used suggested training algorithm, to learned an appropriate neural network for accurately solving any non-linear PDEs.

Keywords: Partial differential equation; Neural networks; BP-training algorithm ; Unconstrained optimization; BFGS training algorithm.
2010 MSC: 78M50, 68M07, 62K05.

Email address: luma.n.m@ihcoedu.uobaghdad.edu.iq (Luma N. M. Tawfiq)

Received September 23, 2023; Accepted: November 28, 2023; Online: December 18, 2023.

1. Introduction

Partial differential equation-based mathematical models can be used to describe a wide variety of physical issues. The partial differential equations(PDEs) govern a wide range of physical, chemical, and biological events. A mathematical model is a condensed, mathematically stated depiction of physical reality [1]. Non-linear PDEs are also crucial for study in a wide range of domains, including hydrodynamics, engineering, quantum field theory, optics, plasma physics, etc [2, 3]. Non-linear high order PDEs have an important role in representing different applied science such physical or chemical phenomena arising in engineering [4].

Therefore, researchers focus their attention on capturing the behaviors of these problems. Since having an exact solution for such problems is not easy, researchers have tried to developing analytic and numerical methods [5, 6, 7] to investigate the behaviors of these problems. Herein, neural network techniques has been proposed for the following problem.

During the last several decades, there has been a lot of interest research of various machine intelligence approaches, particularly artificial neural networks (ANNs) is used to solve differential equations. Because ANN are known to have universal approximation capabilities [13], parallel processing technique, when compared to other traditional numerical approaches. So, many authores used ANN for solving ODEs, PDEs, integral equations and integro equations [14]. The authors proposed various design of ANNs depending on architectural of network: number of layers, number of nodes in each layers, partial or fully connected between layers and/or between nodes in layers, way of feeding the data forward or backword, or depending on training supervise or unsupervised learining [15]. Lee and kang [16] proposed Hopfield neural network for solving differential equations that is unsupervised learining. Lagaris et al. [17] suggested type of neural networks a multi-layer perceptron and used optimization approach to solve each of ODEs and PDEs. Also they solved two and three dimensional PDEs with uneven boundaries using multilayer ANN architecture [18]. Aarts and Van der veer proposed evolutionary ANN for solving IVP for more details see [19]. Shirvany et al. in [20] suggested a multilayer ANN type perceptron with radial basis function (RBF) transfer function for solving the nonlinear Schrodinger problem. Hoda and Nagla [21] used a multilayer ANN technique to address mixed BVPs. Mai-Duy and Tran-Cong in [22] introduced ANN with a radial basis function of type multi quadric for solving ODEs and elliptic PDEs. Jianye etal. in [23] employed ANN with RBF to solve an elliptical PDEs. Parisi et al. in [24] used a different strategy to tackle a steady-state heat transport problem.

Herein we suggest ANNof type feed forward supervise neural network with backpropagation training especially BFGS training algorithm with suggest new modification of BFGS method to avoid disadvantages of traditional BFGS.

The outline of article as follows: The next section, we define and describes the architectural and mathematical formulation of the ANNs. In section 3, BFGS training algorithm is presented. In section 4, modification for BFGS training algorithm will be given. In section 5, the application are presented for modified algorithms, then we design novel ANN for solving 4^{th} order PDEs. In section 6, the global convergence of the suggested modification will be given Finally, the conclusions are given in section 7.

2. Neural network

A parallel processing structure is said to be neural network has been used to distribute information as a series of interconnected layers made up of nodes called neurons (also known as processing elements). It is the ANN's core processor, which consists of links (also known as junctions)—line segments that point in the same direction [25]. All nodes may have an unlimited number of incoming and outgoing links, but all links' characters must be identical [26]. This is due to the fact that each node has an outgoing connection that might branch to produce numerous outgoing connections, all of which have the same sign. Each node has a transfer (activation) function must be sigmoid functions [27] that generates the node's output character using the input characters. Generally speaking, ANNs is generalizations of mathematical models of the human brain based on the idea that information processing occurs at numerous connecting nodes; characters are passed between nodes on connecting links that have an associated weight; and each node produce a transfer function to its weighted input to determine its output sign [28].

So, for a given input represent as a vector x, the weighted input represented as $W_j^T x$ is the input of hidden neurons. It is assumed that each of the hidden neurons has an identical transfer function σ, but this bias bj. Hence the output of the jth hidden neuron is $\sigma(W_j^T x + b_j)$.

We now denote the weight connecting the j^{th} hidden node to the exit by $\mho_j$. The output function $g(x)$ of ANN is therefore [29]:

$$g(x) = \sum_{j=1}^{k} \mho_j \sigma(W_j^T x + b_j) \tag{1}$$

Note that σ must be sigmoid functions, so we choose Appropriate σ defined here as [30]:

$$\sigma(n_i) = \frac{e^{n_i} - 1}{e^{n_i} + 1} \tag{2}$$

Then the input-output equation ANN has the following form: $\hat{Y} = \Phi(x^T W^T + b^T)\mho^T$ where $W \epsilon R^{n\times r}; \mho \epsilon R^{1\times n}$ and $b \epsilon R^{n\times 1}$ are input weights, output weights and adjustable bias, respectively.

There are numerous classes of ANN architectures that the ANN interconnection structure may be split into, including: Feed Forward Neural Networks (FFNN) [31]: In a strict sense, data flows from the input node to the output node as a form of feedback, or a forward loop. Organized nodes are layered, and inputs from the previous layer arrive before forwarding their output to the next layer. FBNN: Feedback Neural Network [32]. There are no restrictions on connections between layers and neurons. Data transmission via loopback in the network. Here, we go with FFNN.

3. BFGS training algorithm

In this section, we describes BFGS training algorithm for convex function f. The formula BFGS is abbreviation of (Broyden, Fletcher, Goldfar, and Shanno [32]), which is one of the best effective quasi-Newton algorithm. Convex functions can be combinedwith exact line or certain special inexact line search techniques that have global convergence [33] and superlinear convergence [34]. Consider f: $R^n \to R^n$;

$$\exists L > 0 \ni \|g(x) - g(y)\| \leq L \|x - y\|, \forall y, x \in R^n \tag{3}$$

Where $g(x)$ is gradient for f in x, i.e., $g(x) = \nabla f(x)$, so

$$G_k p_k + g_k = 0 \tag{4}$$

Where G_k is Hessian matrix, i.e., $G_k = \nabla \mathrm{g}(x)$; and p_k is search direction defined by

$$x_{k+1} = x_k + \lambda_k p_k \tag{5}$$

Satisfies

$$(x_k + \lambda_k p_k) \leq f(x_k) + \alpha \lambda_k g(x_k)^T p_k, \alpha \in (0, 1) \tag{6}$$

Where λ_k is step length along direction p_k satisfy $\lambda_k = \rho^{i_k}, \rho \in (0, 1)$ and i_k is the smallest non -ve integer satisfying Eq.(4).

There are many techniques to calculate p_k. One of the most efficient techniques is newton's method because it has a quadratic convergence that requires the least number of epochs, i.e. H. The least number of iterations to evaluate the function [32], but it has a disadvantage in computing the hessian matrix, which is the second derivative of the global error value. Thus, broyden, fletcher, goldfarb, and shanno studied an update of newton's method and reported a learning algorithm (bfgs) that overcomes this deficiency and proposed for to compute the search direction p_k as follows:

$$B_k p_k + g_k = 0 \tag{7}$$

where B_k is the update matrix for BFGS method defined as:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{\gamma_k \gamma_k^T}{\gamma_k^T s_k} \tag{8}$$

Where, $s_k = x_{k+1} - x_k$, $\gamma_k = g_{k+1} - g_k$

From Eq.(2) we can generate the p_k direction replacing G_k with the matrix

$$\bar{G}_k \triangleq G_k + r_k I; k \in (0,1)$$

There are some modifications of BFGS suggested by researchers to speed the convergence of a BFGS without the convexity condition. The present some of these were proposed by Li and Fukushima [33]:

3.1. Formula 1

The 1^{st} update formula of BFGS is defined by

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{\delta_k \delta_k^T}{s_k^T \delta_k} \tag{9}$$

where $\delta_k = (\max\left\{0, -\frac{\gamma_k^T s_k}{\|s_k\|^2}\right\} + \mathcal{H}(\|g_k\|))s_k + \gamma_k$ and function $\mathcal{H} : R \to R$ satisfies:

1. $\mathcal{H}(t) > 0; \forall t > 0$;
2. $\mathcal{H}(t) = 0$ if and only if $t = 0$
3. if t belong to a bounded set, and $\mathcal{H}(t)$ is bounded.

Depending on the definition of δ_k , it is easy to get:

$$\delta_k^T \delta_k \geq \max\{\delta_k^T \gamma_k, \mathcal{H}(\|g_k\|) \|s_k\|^2\} > 0$$

This is sufficient condition to guarantee the positive definiteness of B_{k+1} as long as B_k is positive definite. Li and Fukashima presented $\mathcal{H}(t) = \mu t$ with some constant $\mu > 0$.

3.2. Formula 2

The 2^{nd} update formula of BFGS is defined by

$$B_{k+1} = \begin{cases} B_k & -\frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{\delta_k \delta_k^T}{s_k^T \delta_k} \\ B_k & O.W \end{cases} \tag{10}$$

where the attributes are the same as those in Formula 1 for δ_k, and $\mathcal{H}$. These two techniques have global convergence and superlinear convergence for nonconvex functions. Some researchers have done further work to get a more accurate estimate of the objective function's Hessian matrix.

3.3. Formula 3

The BFGS update formula is defined by

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{\gamma_k^{m*} \gamma_k^{m*T}}{s_k^T \gamma_k^{m*}} \tag{11}$$

where $\gamma_k^{m*} = \gamma_k + \frac{\rho_k}{\|s_k\|^2} s_k$ and $\rho_k = 2\{f(x_k) - f(x_k + \lambda_k p_k)\} + (g(x_k + \lambda_k p_k) + g(x_k))^T s_k$.

It is straightforward to infer that this formula includes information on both the gradient and function values.

The resulting approaches might be thought to be superior to the standard BFGS method. In reality, the computation in practice demonstrates that the method is superior to the standard BFGS method and that it has some theoretical benefits (see [32]). Wei et al., in [34] proposed the quasi-Newton approach and demonstrated its super linear convergence for uniformly convex functions under the WWP line search. Its global convergence is shown in [25], although the approach is ineffective for all generic convex functions. The non-positive definiteness of matrix Bk for generic convex functions is one of the primary causes of failure. According to Byrd et al., in [30], matrix Bk's positive definiteness is a key factor in the quasi Ne ton algorithm's ability to converge. First, Yuan and Wei in [29] used gradient and function value data for general convex functions to examine the global convergence and super linear convergence of the modified BFGS formula in [14]. Another BFGS formula was proposed by Yuan and Wei based on Eq.(12).

3.4. Formula 4

The BFGS update formula is defined by

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{\gamma_k^m \gamma_k^{mT}}{s_k^T \gamma_k^m} \tag{12}$$

where

$$\gamma_k^m = \gamma_k + \max\left\{\frac{\rho_k}{\|s_k\|^2}, 0\right\} s_k.$$

For typically convex functions, this updated method yields global convergence and super linear convergence. Zhang et al., in [27] had already completed the same task.

3.5. Formula 5

The BFGS update formula is defined by

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{\gamma_k^{1*} \gamma_k^{1*T}}{s_k^T \gamma_k^{1*}} \tag{13}$$

Where

$$\gamma_k^{1*} = \gamma_k + \overline{A}_k s_k, \overline{A}_k = \frac{6\left[f\left(x_k\right) - f\left(x_k + \lambda_k p_k\right)\right] + 3(\nabla f\left(x_k + \lambda_k p_k\right) + \nabla f\left(x_k\right))^T s_k}{\|s_k\|^2}.$$

It has been demonstrated that the new equation has a higher order approximation to $\nabla^2 f(x)$, and it is obvious that the quasi Newton Eq.(13) also incorporates gradient and function value information. Additionally, in a limited memory BFGS method, where global convergence is only achieved for uniformly convex functions, Yuan et al., in [18] extended a similar methodology to γ_k^{1*}. There have been several other modified quasi-Newton methods reported.

4. Suggested modification for BFGS training algorithm

In this section, we proposed modified BFGS training algorithms, denoted MBFGS as follows:

Algorithm 4.1.
Step 0*: select the initial positive definite matrix B0; Starting point* $x_0 \in R^n$ *and the constants* σ_1, σ_2 *and* C *such that* $C > 0$; $0 < \sigma_1 < \sigma_2 < 1$. *Let* $k = 0$.
Step 1*: To get* p_k, *solve the following linear equation:*

$$g\left(x\right) + B_k p_k = 0;$$

Step 2*: Calculate a step size* $\lambda_k > 0$ *satisfying the Wolfe-type for line search conditions:*

$$f(x_k + \lambda_k \rho_k) \leq f\left(x_k\right) + \sigma_1 \lambda_k g_k^T \rho_k,$$

$$g(x_k + \lambda_k \rho_k)^T \geq \sigma_2 g_k^T \rho_k \tag{14}$$

Moreover, if $\lambda_k = 1$ satisfies (7), then take $k = 1$.
***Step 3**: Calculate the iteration $x_{k+1} = x_k + \lambda_k \rho_k$.*
***Step 4**: Let $s_k = x_{k+1} - x_k = \lambda_k \rho_k, \gamma_k = g_{k+1} - g_k$*

$$\mu_k = \frac{\gamma_k^T s_k}{\|s_k\|^2},$$

and

$$y_k = \gamma_k + r_k s_k$$

where

$$r_k \in [0; C]$$

***Step 5**: Using the Eq.(8) to update B_k*

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} \tag{15}$$

***Step 6**: Take $k := k + 1$ and go to Step 1.*

This problem solved by sme researchers by using different methods such as the direct integration method, homotopy perturbation method (HPM), multiple exp-function method, improved Bernoulli sub-equation function method and etc. In [6, 7, 8, 9, 10], The solutions are derived in terms of hyperbolic, trigonometric and rational functions and get the solution with free parameters. The general exact solutions of these equations are converted into different known shape waves, namely, kink, bell shape soliton, periodic soliton, singular solitons etc. Although a great deal of research work has been devoted to finding different methods to solve nonlinear high order equations

5. Application

In this section we design novel artificial neural networks based on training by new modification of BFGS to solve the following model equation

$$u_{xt} - u_{xxxy} - 2u_{xx}u_y - 4u_x u_{xy} = 0$$

The exact solution [16]:

$$u(x, y, t) = \tanh\left(\frac{1}{2}(x + y - t)\right)$$

The architecture of suggested design is 3 layer FFNN that is one hidden layer neural network depending on Hecht-Nielsen theorem [24]. This theorem, text that a one hidden layer neural network is capable of approximating a wide class of functions, with any given accuracy.

However, it may be necessary to actually increase the number of nodes, but through our work we have shown empirically that this is not really for the class of problems specific high order problems. In the end of trials we see that the best choice of architectural for suggested design in this application is one hidden layer FFNN with sigmoid transfer function especially "tanhsig" and one neurons with a "linsig" transfer function in the output layers. also we observed that the increase in layers can increase the computational costs and can lead to stock problems, as shown in Tables 1 and 2. Train the proposed ANN with the backpropagation rule and choose a new modification of BFGS (MBFGS). For any input x, y and t the process from the input layer to the hidden layer is as follows:

$$n_i = \sum_{i=1}^{5} (\ W_{x_i} x + W_{y_i} y + W_{t_i} t) + b_1, \tag{16}$$

where W_{x_i}, Wy_i and W_{t_i} are the weights link the inputs x, y and t to the nodes in the hidden layer, and b_1 represent the biase of hidden layer's. We choose the log sig. as transfer function for hidden layer.

The mechanism that links the hidden layer to the output layer has the following form:

$$h_i = \sum_{j=1}^{5} \mho_{ij} \quad \sigma\left(n_i\right) + b_2 \tag{17}$$

where $\mho_{ij}$ is the weights link the nodes of hidden layer with output layer, and b_2 is the biase.

So the output of suggested design has the form:

$$u_{net}\left(x, y, t; \theta\right) = \sum_{j=1}^{5} \mho_i \sigma\left(h_i\right)$$

Then, it is also easy to express the k-th derivatives of $u_{net}(x, y, t; \theta)$ in terms:

$$\frac{\partial^k u_{net}\left(x, y, t; \theta\right)}{\partial x^k} = \sum_{j=1}^{n} \frac{\partial^k \mho_j f(h_2)}{\partial x^k},$$

$$\frac{\partial^k u_{net}\left(x, y, t; \theta\right)}{\partial y^k} = \sum_{j=1}^{n} \frac{\partial^k \mho_j f(h_2)}{\partial y^k},$$

$$\frac{\partial^k u_{net}\left(x, y, t; \theta\right)}{\partial t^k} = \sum_{j=1}^{n} \frac{\partial^k \mho_j f(h_2)}{\partial t^k}, k = 1, \ldots, n \tag{18}$$

The performance of network solution $u_{net}(x, y, t; \theta)$ is calculated intrain, test and validation case. Figure 1 illustrates mean square error (MSE) obtained by each three cases: train, test and validation for different values of epochs. Tables 1 and 2 give a comparison of the absolute error between the neural results and exact solution in different types of BFGS update when x and y belong to $[0, 1]$ with different values of t. We choose one hidden layer having 5 nodes. The minimum error for each value of t was obtained when we take suggested modification of BFGS update as shown in Table 1. Increasing the number of nodes to 11 does not significantly reduce errors. Table 2 illustrate the same comparison but when we take formula 2, 3 and 4 for BFGS update training algorithm with the same number of neurons in each layer. We observed that the suggested modification has well properties compared to other formulas of BFGS update such increase the computational costs and can lead to stock problems. From the table, it can be seen that for all test points, the suggested design provides a very rapid and accurate approximation for the nonlinear PDEs. Table 3 consist the final weights and bias in proposed FFNN. Figure 2 illustrates the target in each of the three cases: train, test, and validation. In addition, the behavior of the gradient for 818 epoch in validation case is illustrated in Figure 3. Figure 4 illustrates the neural solution with the learning rate $(\eta) = 0.01$.

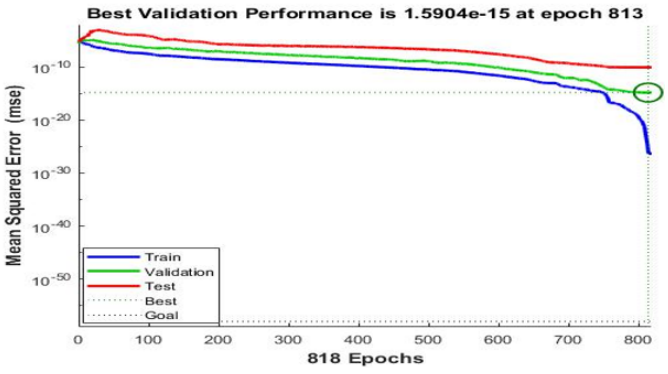


Figure 1: Mean square error for train, test and validation case for different values of epochs

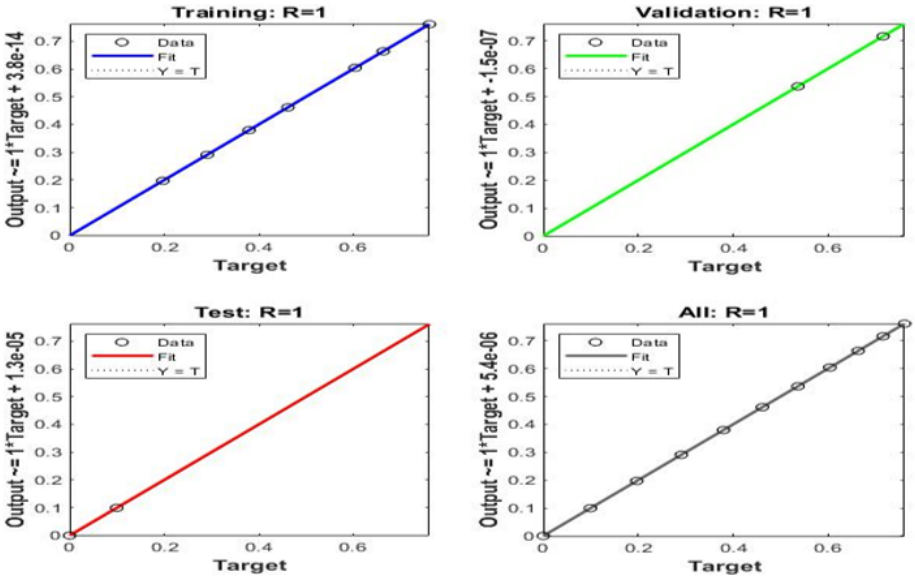


Figure 2: The target of output in train, test and validation case

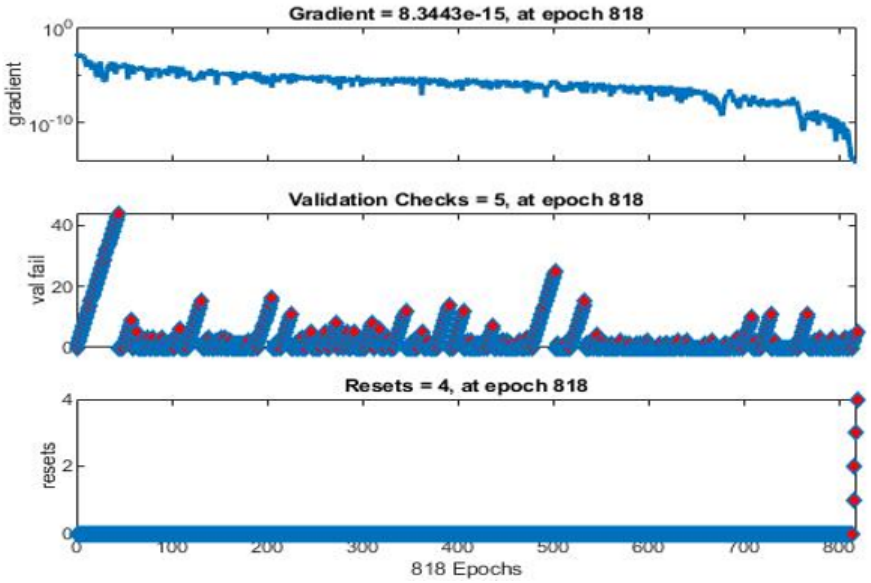


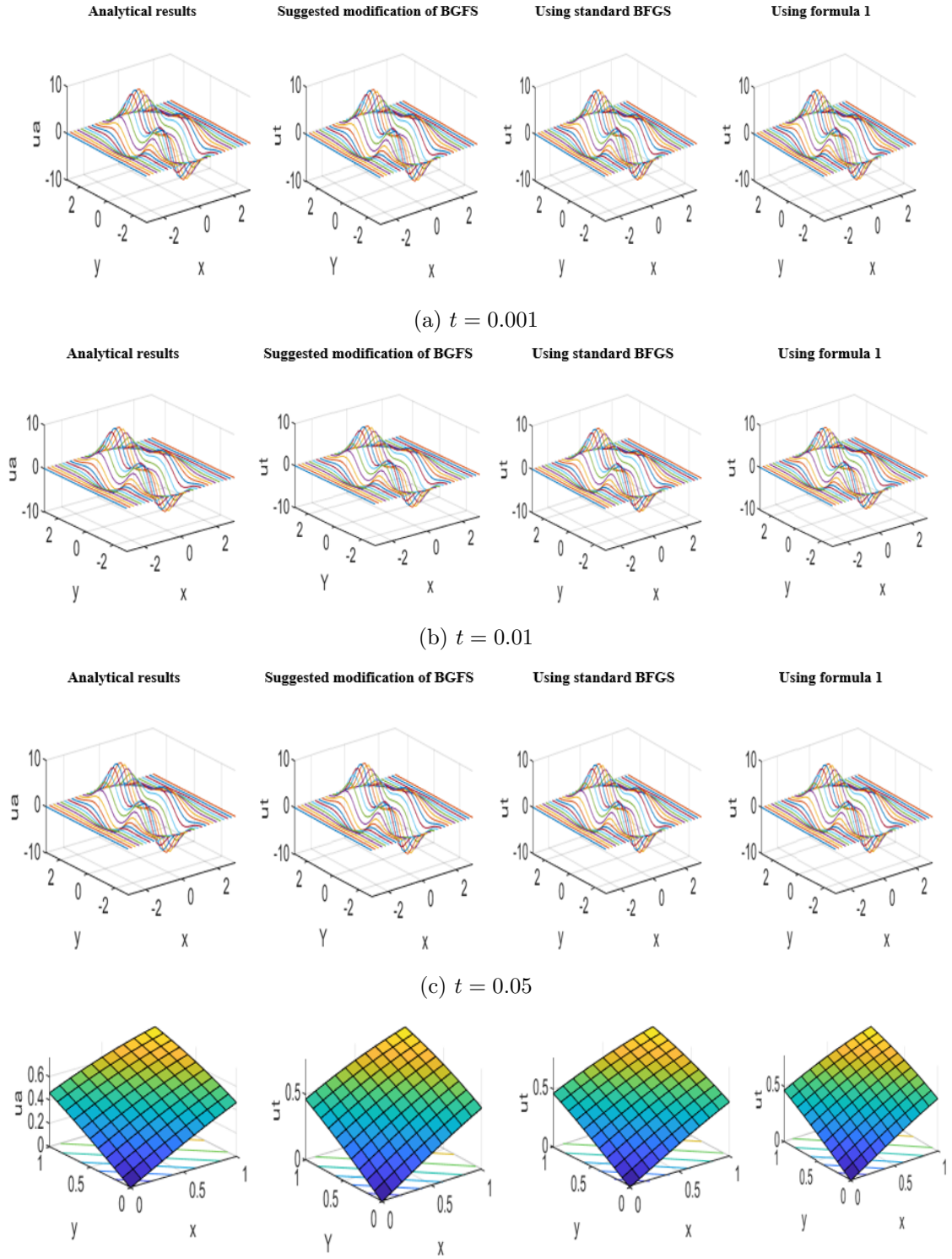
Figure 3: Behavior of gradient in validation case at epoch 818

Table 1: A comparison of the absolute error between the neural results and exact solution for different modification of BFGS

	absolute error $E(x) = \|U_a - U_N\|$		
$x = y$	t = 0.001		
	If using suggested modification layer	If using standard BFGS	If using formula 1 of modification layer
0.1	1.58564357097268e-06	4.24149712652255e-06	0.0110630092298508
0.3	1.25177646026486e-13	2.16254791851611e-10	1.78443107295978e-06
0.5	1.10855769008822e-13	5.35026578596387e-10	3.54929593493480e-06
0.7	9.69224700497762e-14	4.54914660941164e-07	7.18385033793290e-06
$x = y$	t = 0.01		
	If using suggested modification layer	If using standard BFGS	If using formula 1 of modification layer
0.1	1.15850260634653e-08	3.87217494179914e-10	0.000280065472240021
0.3	1.54058987789085e-11	1.57122199856419e-06	0.000110857112175178
0.5	7.47684747448574e-09	1.58787760717871e-09	1.06357972611271e-05
0.7	1.31811472847687e-08	4.70896968529644e-07	1.66604572513496e-05
$x = y$	t = 0.05		
	If using suggested modification layer	If using standard BFGS	If using formula 1 of modification layer
0.1	4.39033531751676e-11	2.42757033919183e-08	0.00143999420900501
0.3	1.33728028650637e-11	2.44086051759407e-07	0.00378350854874393
0.5	9.78947589103107e-08	9.98240046068410e-07	0.00314903262298566
0.7	5.35937960677302e-12	3.83466422460010e-06	9.37158550532447e-05

Table 2: A comparison of the absolute error between the neural results and exact solution for different modification of BFGS

	absolute error $E(x) = \|U_a - U_N\|$		
$x = y$	t = 0.001		
	If using formula 2 of modification	If using formula 3 of modification	If using formula 4 of modification
0.1	5.18154963380368e-13	2.50913125110497e-07	0.0504037614792988
0.3	1.41426963251590e-07	2.93177383314802e-07	1.05827643070988e-06
0.5	7.59029944452649e-08	4.55813276567518e-08	0.0348099735619090
0.7	2.11874962019465e-12	1.46132627665274e-07	1.09930213254561e-06
$x = y$	t = 0.01		
	If using formula 2 of modification	If using formula 3 of modification	If using formula 4 of modification
0.1	2.33086276696382e-07	1.43879126043855e-07	3.80212134796776e-05
0.3	8.83990234901155e-08	5.35053363815430e-05	0.000196577851321045
0.5	5.40684164107574e-12	4.64863414195715e-06	4.69595130647149e-05
0.7	5.93203264287467e-12	1.87711074195462e-06	8.30625312155942e-05
$x = y$	t= 0.05		
	If using formula 2 of modification	If using formula 3 of modification	If using formula 4 of modification
0.1	2.39905317833689e-12	1.45336520596118e-10	2.38172049304985e-08
0.3	1.37893030327518e-11	7.60035367974865e-11	6.87790768783003e-09
0.5	2.41384690014002e-12	3.74207131897952e-06	1.02185107653252e-07
0.7	8.98836560736527e-13	2.33798081705761e-05	4.50263761830882e-08



(a) $t = 0.001$

(b) $t = 0.01$

(c) $t = 0.05$

(d)

Figure 4: Results of suggested design for different formula of BFGS update

Table 3: Weights and bias for FFNN in validation case.

weights from input to hidden layer	weights from hidden to output layer	Bias
0.1119	0.9832	0.9874
0.8108	0.4678	0.2066
0.3804	0.6925	0.0535
0.1048	0.0284	0.0861
0.1991	0.2635	0.3581

This problem solved by sme researchers by using different methods such as the direct integration method, homotopy perturbation method (HPM), multiple exp-function method, improved Bernoulli sub-equation function method and etc. In [6, 7, 8, 9, 10], The solutions are derived in terms of hyperbolic, trigonometric and rational functions and get the solution with free parameters. The general exact solutions of these equations are converted into different known shape waves, namely, kink, bell shape soliton, periodic soliton, singular solitons etc. Although a great deal of research work has been devoted to finding different methods to solve nonlinear high order equations.

6. Convergence analysis

This section discussed the global convergence analysis of MBFGS. If λ_k satisfies the Wolfe condition (Eq.(14)), then B_{k+1} is symmetric positive definite that is B_k is symmetric positive definite. Also g satisfies Lipschitz condition

$$|\mu_k| = \frac{|\gamma_k^T s_k|}{\|s_k\|^2} \leq \frac{\|\gamma_k^T\|}{\|s_k\|} \leq L, k = 1, 2, \ldots \tag{19}$$

In Section 4, step 4, $r_k \in [0; C]$ with Eq.(19) implies

$$\mu_k + r_k \leq L + C; k = 1, 2, \ldots$$

That is $\mu_k + r_k$ is bounded from above. So r_k must be chosen such that for some constant $\varepsilon > 0$,

$$\mu_k + r_k \geq \varepsilon, \tag{20}$$

this means it is bounded below and hence it is bounded. Eq.(20) is hold by a suitable choice of r_k .

Now the following theorem shows a global convergence for MBFGS training algorithm

Theorem 6.1. *If $\{x_k\}$ generated by MBFGS training algorithm with Bk being updated by Eq.(15). Then*

$$\lim_{k\to\infty} \inf \|g_k\| = 0$$

Proof. We assume that $\|g_k\| \geq \gamma > 0$ for all k. Since $B_k s_k = \lambda_k B_k p_k = -\lambda_k g_k$. Since f is bounded below, by summing the 1^{st} inequalities of Wolfe conditions Eq.(14) we have:

$$\sum_{k=0}^{\infty} (-g_k^T s_k) < \infty, \tag{21}$$

But

$$\sum_{k=0}^{\infty} (-g_k^T s_k) = \sum_{k=0}^{\infty} \frac{1}{\lambda_k} s_k^T B_k s_k = \sum_{k=0}^{\infty} \frac{\|g_k\|}{\|B_k s_k\|} s_k^T B_k s_k$$

$$= \sum_{k=0}^{\infty} \|g_k\|^2 \lambda_k \frac{s_k^T B_k s_k}{\|B_k s_k\|^2} \geq \gamma^2 \sum_{k=0}^{\infty} \lambda_k \frac{s_k^T B_k s_k}{\|B_k s_k\|^2}$$

□

Therefore, for any $\delta > 0$, there exists an integer $k_0 > 0$ such that for any positive integer q

$$q\left(\prod_{k=k_0+1}^{k_0+q}\lambda_k\ \frac{s_k^T B_k s_k}{\|B_k s_k\|^2}\right)^{\frac{1}{q}} \leq \sum_{k=k_0+1}^{k_0+q}\lambda_k \frac{s_k^T B_k s_k}{\|B_k s_k\|^2} \leq \delta$$

from the geometric inequality the left-hand of above inequality follows. Thus

$$\left(\prod_{k=k_0+1}^{k_0+q}\lambda_k\right)^{\frac{1}{q}} \leq \frac{\delta}{q}\left(\prod_{k=k_0+1}^{k_0+q}\frac{\|B_k s_k\|^2}{s_k^T B_k s_k}\right)^{\frac{1}{q}} \leq \frac{\delta}{q^2}\left(\prod_{k=k_0+1}^{k_0+q}\frac{\|B_k s_k\|^2}{s_k^T B_k s_k}\right)$$

$$\leq \frac{\delta}{q^2}\left(\prod_{k=0}^{k_0+q}\frac{\|B_k s_k\|^2}{s_k^T B_k s_k}\right) \leq \frac{\delta\,(k_0+q+1)}{q^2}\,M$$

Letting $q \to \infty$ which is a contradiction, because the left-hand side of the above inequality is greater than a positive constant. Thus, we get the results.

7. Conclusions

In this article, ANNs based on unconstrained optimization based on new modification of BFGS training algorithm have been introduced and implemented to solve non-linear PDEs. The neural solutions were obtained and appeared to be efficient and accurate compared with traditional BFGS or other modification of BFGS. The results show that the proposed modification has a number of advantages based on implementation, gradient calculation, weight updates, and total processing cost. The main contribution in this article is the global convergence for convex functions (energy function). The numerical results show that the proposed modification is rival with other modifications for the test problems.

References

[1] H, Salih, Solving Modified Regularized Long Wave Equation using Collocation Method. JPCS. Vol. 1003, No. 012062, pp.1-10, 2018.

[2] Salih H. 2020. Solution of Modified Equal Width Equation Using Quartic Trigonometric-Spline Method. JPCS. 1664, 012033: 1-10.

[3] N.A. Hussein, Efficient Approach for Solving High Order (2+1)D-Differential Equation, AIP Conference Proceedings, 2398, 10, pp: 1-11, 2022.

[4] LNM, Tawfiq. The Finite Element Neural Network And Its Applications To Forward And Inverse Problem. IHJPAS. Vol. 19, No. 4, pp. 109-124, 2017.

[5] H. Altaie, Recent Modification of Homotopy Perturbation Method for Solving System of Third Order PDEs, JPCS, 1530, 012073, pp: 1-8, 2020.

[6] NA. Hussein, New Approach for Solving (1+1)-Dimensional Differential Equation. JPCS. Vol. 1530, No. 012098, pp. 1-11, 2020.

[7] Z. H. Kareem, L. N. M. Tawfiq, Recent modification of decomposition method for solving wave-like Equation, Journal of Interdisciplinary Mathematics, Vol. 26, No. 5, pp. 809–820, 2023.

[8] N A Hussein, Solitary Wave Solution of Zakharov-Kuznetsov Equation, AIP Conference Proceedings, Vol. 2398, Issue. 1, pp: 1-6, 2022.

[9] A.H. Khamas, New Approach for Calculate Exponential Integral Function. Iraqi Journal of Science, 64(8), pp. 4034 – 4042, 2023.

[10] NA Hussein, Efficient Approach for Solving (2+1) D- Differential Equations, Baghdad Sci. J. Vol. 18, pp. 166-174, 2022.

[11] LNM Tawfiq, WR Hussein. Design Suitable Neural Network for Processing Face Recognition. GJESR. Vol. 3, No. 3, pp. 58-64, 2016.

[12] Al-Noor, T.H., Estimate the Rate of Contamination in Baghdad Soils By Using Numerical Method, JPCS, 1294 (032020) 2019.

[13] LNM Tawfiq and OM Salih, Design neural network based upon decomposition approach for solving reaction diffusion equation, JPCS, 1234, 012104, pp 1-8, 2019.

[14] N.A. Hussein, Double LA-transform and their properties for solving partial differential equations, AIP Conf. Proc., AIP Conf. Proc., 2834, 080140, p.1-10, 2023.
[15] YA Oraibi, Fast Training Algorithms for Feed Forward Neural Networks. IHJPAS. Vol. 26, No. 1, pp. 275-280, 2013.
[16] AH. Khamas, New Coupled Method for Solving Burger's Equation. JPCS. 1530: 1-11, 2020.
[17] Lagaris IE, Likas A, Fotiadis DI (1998) Artificial neural networks for solving ordinary and partial differential equations. IEEE Trans Neural Netw 9:987–1000.
[18] Lagaris IE, Likas AC, Papageorgiou DG (2000) Neural network methods for boundary value problems with irregular boundaries. IEEE Trans Neural Netw 11:1041–1049.
[19] Aarts LP, Van der veer P (2001) Neural network method for solving partial differential equations. Neural Process Lett 14:261–271.
[20] ZH. Kareem, New Modification of Decomposition Method for Solving High Order Strongly Nonlinear Partial Differential Equations, AIP Conference Proceedings, Vol. 2398, Issue. 1, pp: 1-9, 2022.
[21] AQ. Ibrahim Abed, Efficient Method for Solving Fourth Order PDEs, JPCS, 2021, 1818, 012166: 1-10.
[22] N.A. Hussein, Exact Solution for Systems of Nonlinear (2+1)D-Differential Equations, Iraqi Journal of Science, 2022, 63, 10: 4388-4396.
[23] Jianyu L, Siwei L, Yingjian Q, Yaping H, Numerical solution of elliptic partial differential equation using radial basis func-tion neural networks. Neural Netw 16, p:729–734, 2003.
[24] LNM Tawfiq, AAT Hussein. Design feed forward neural network to solve singular boundary value problems. ISRN Applied Mathematics. Vol. 2013, pp. 1-7, 2013.
[25] Chen, F.; Sondak, D.; Protopapas, P.; Mattheakis, M.; Liu, S.; Agarwal, D.; Di Giovanni, M. NeuroDiffEq: A Python pack-age for solving differential equations with neural networks. Journal of Open Source Software 2020, 5, 1931.
[26] Kareem ZH, Efficient Modification of the Decomposition Method for Solving a System of PDEs. Iraqi J. Sci. 2021; 62(9): 3061-3070.
[27] M.O., Enadi, New Approach for Solving Three Dimensional Space Partial Differential Equation. Baghdad Sci. J. Vol.16, No. 3, 2019, 786-792.
[28] FF. Ghazi, New Approach for Solving Two Dimensional Spaces PDE. JPCS, 1530, 012066, pp: 1-10, 2020.
[29] L.N. M. Tawfiq, N.A. Hussein, Exact soliton solution for systems of non-linear (2+1)D-DEs, AIP Conf. Proc., AIP Conf. Proc. 2834, 080137, p.1-7, 2023.
[30] A.H. Khamas, Determine the Effect Hookah Smoking on Health with Different Types of Tobacco by using Parallel Processing Technique, JPCS, 2021, 1818, 012175: 1-10.
[31] N.A.K. Hussein and B.Al-Sarray, Deep Learning and Machine Learning via a Genetic Algorithm to Classify Breast Cancer DNA Data. Iraqi Journal of Science. 63 (7): 3153-3168, 2022.
[32] Ali, MH, Tawfiq, LNM. Design Optimal Neural Network for Solving Unsteady State Confined Aquifer Problem, Mathematical Modelling of Engineering Problems, 2023, 10, 2, 565-571.
[33] Jamil, H.J., Hookah Smoking with Health Risk Perception of Different Types of Tobacco, JPCS, 2020, 1664(1), 012127.
[34] M.H. Ali, Efficient Design of Neural Networks for Solving Third Order Partial Differential Equations, JPCS, vol. 1530, no. 1, p. 1-10, 2020.