International Journal of Mechanical, Industrial and Aerospace Sciences
ISSN: 2517-9950
Vol:5, No:1, 2011

A Hybrid Neural Network and Gravitational Search Algorithm (HNNGSA) Method to Solve well known Wessinger's Equation

M. Ghalambaz, A.R. Noghrehabadi, M.A. Behrang, E. Assareh, A. Ghanbarzadeh, N.Hedayat

***Abstract*—**This study presents a hybrid neural network and Gravitational Search Algorithm (HNGSA) method to solve well known Wessinger's equation. To aim this purpose, gravitational search algorithm (GSA) technique is applied to train a multi-layer perceptron neural network, which is used as approximation solution of the Wessinger's equation. A trial solution of the differential equation is written as sum of two parts. The first part satisfies the initial/ boundary conditions and does not contain any adjustable parameters and the second part which is constructed so as not to affect the initial/boundary conditions. The second part involves adjustable parameters (the weights and biases) for a multi-layer perceptron neural network. In order to demonstrate the presented method, the obtained results of the proposed method are compared with some known numerical methods. The given results show that presented method can introduce a closer form to the analytic solution than other numerical methods. Present method can be easily extended to solve a wide range of problems.

***Keywords*—**Neural Networks; Gravitational Search Algorithm (GSR); Wessinger's Equation.

I. Introduction

NONLINEAR phenomena, which are modeled with nonlinear differential equations, appear in different domains of engineering such as fluid dynamics, aerodynamics, nonlinear control systems, electrical engineering, and etc [1]. Many different methods have been developed for solving differential equations. However, the solution of nonlinear differential equations is still challenging [1, 2].

Lee and Kang [3] used parallel processor computers in order to solve a first order differential equation using Hopfield neural network models. Meade and Fernandez [4, 5] used feed forward neural networks architecture and B_1 splines to solve linear and nonlinear ordinary differential equations. Lagaris et al. represented a new method to solve First order linear ordinary and partitial differential equations using artificial neural networks [2]. Malek and Shekari Beidokhti used a hybrid artificial neural network- Nelder-Mead method to solve high order linear differential equations [6]. A hybrid artificial neural network- swarm intelligence method was used by Khan et al. to solve Wessinger's equation [1]. In that work (i.e. Ref [1]), the presented method could not satisfy the initial/ boundary conditions. In contrast with Khan et al. [1], present study introduces a hybrid artificial neural networks and gravitational search algorithm method to solve Wesingger's equation which satisfies the initial/ boundary conditions. Finally, in order to demonstrate the presented method, fair comparisons are made on same problem which are solved using the other numerical methods. The paper is organized as follows. A brief review of artificial neural networks and gravitational search algorithm are brought in Sections 2 and 3, respectively. Section 4 gives details of problem formulation and novel solution model. Numerical results are discussed in Section 5. Finally, conclusions and directions for future research are presented in section 6.

M. Ghalambaz., is a PhD candidate of fluid mechanics of Shahid Chamran University of Ahvaz (corresponding author to provide phone: +989166445271; e-mail: M.Ghalambaz@yahoo.com).

A.R. Noghrehabadi is with Mechanical Engineering Department of Shahid Chamran University of Ahvaz.

M.A. Behrang is with Mechanical Engineering Department of Islamic Azad University, Dezful Branch, Iran (e-mail: Mohammadali.behrang@gmail.com).

E. Assareh is with Mechanical Engineering Department of Islamic Azad University, Dezful Branch, Iran (e-mail: Ehsanolah.assareh@gmail.com).

A. Ghanbarzadeh is with Mechanical Engineering Department of Shahid Chamran University of Ahvaz.

N. Hedayat is Dean, faculty of Agriculture, Islamic Azad University, Dezful Branch, Iran (e-mail: n.hedayat@yahoo.com).

II. Artificial Neural Networks (ANNs)

Neural networks are computational models of the biological brain. Like the brain, a neural network comprises a large number of interconnected neurons. Each neuron is capable of performing only simple computation [7].Any how, the architecture of an artificial neuron is simpler than a biological neuron. ANNs are constructed in layer connects to one or more hidden layers where the factual processing is performance through weighted connections. Each neuron in the hidden layer joins to all neurons in the output layer. The results of the processing are acquired from the output layer. Learning in ANNs is achieved through particular training algorithms which are expanded in accordance with the learning laws, assumed to simulate the learning mechanisms of biological system [8]. However, as an assembly of neurons, a neural network can learn to perform complex tasks including pattern recognition, system identification, trend prediction, function approximation, and process control [7].

Multi-layer Perceptron (MLPs) are perhaps the most common type of feedforward networks [9]. Their application in function approximation is well known [6]. Fig.1 shows an

MLP which has three layers: an input layer, an output layer and a hidden layer.

Neurons in input layer only act as buffers for distributing the input signals x_i to neurons in the hidden layer. Each neurons j (Fig. 2) in the hidden layer sums up its input signals x_i after weighting them with the strengths of the respective connections w_{ji} from the input layer and adding the bias b_i to them, and computes its output n_j as a function g of the sum, viz.

$$n_j = g\left(\sum w_{ji}\, x_i + b_j\right) \quad (1)$$

where n_j is each neuron output and g can be a simple threshold function or a sigmoid, Gaussian, hyperbolic tangent or radial basis function.

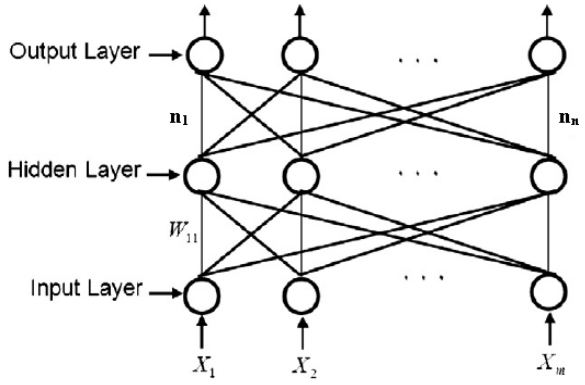

Fig. 1 A multi-layer perceptron.

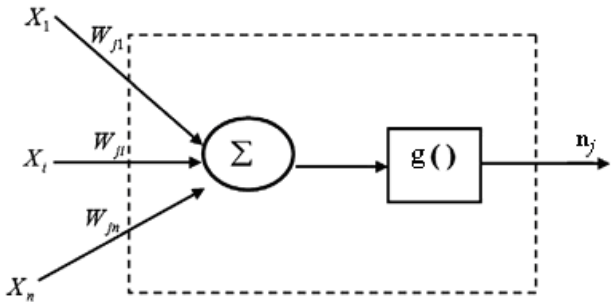

Fig. 2 Details of a neuron.

From Kolmogorov existence theorem, any continuous function of n variable can be approximated using a three-layered perceptron with $n(2n+1)$ nodes [10, 11]. So, the accuracy of the approximation dose not depend on the number of the hidden layers, but it fully depends on the number of neurons in the hidden layer [6].

Fig.3 shows a three-layered perceptron with one input x, one hidden layer consisting of H neuron, and one output $N(x,p)$. This structure is used in the present method. For each entry x the network output is computed by $N(x,p) = \sum_{i=1}^{H} \left(v_i\, g\left(w_i\, x + b_i\right)\right)$, which w_i is a weight parameter from input layer to ith neuron in hidden layer; v_i is a weight parameter from ith neuron in hidden layer to output layer, and b_i is a bias for ith neuron in hidden layer. For more details reders are refered to [12].

Some of benefits of hybrid ANNs and optimization techniques are listed in [2].

The training of an MLP network involves the minimization of an error function. As it is mentioned subsequently, this study lets the network learn from the theory of differential equations in order to approximate a function consisting adjustable parameters.

III. Gravitational Search Algorithm (GSA)

Heuristic algorithms mimic biological or physical processes. One of the newest heuristic algorithms that has been inspired by the physical laws is Gravitational Search Algorithm (GSA) [13].

In GSA, Newtonian laws of gravity and motion are applied to find the optimum solution by a set of agents called masses [14].

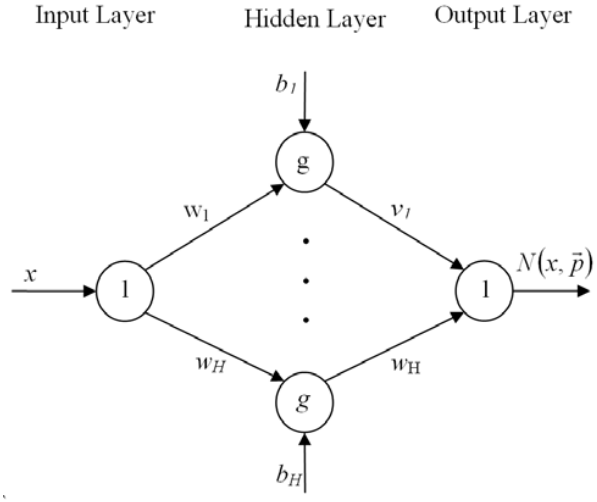

Fig. 3 Details of a three layerd perceptron with one input, one hidden layer (consist H neuron), and one output.

To describe the GSA consider a system with s masses in which position of the ith mass is defined as follow:

$$X_i = (x_i^1, ..., x_i^d, ..., x_i^n), \qquad i = 1,2,...,s \quad (2)$$

where X_i^d is position of the ith mass in the dth dimension and n is dimension of the search space.

Mass of each agent is calculated after computing current population's fitness as follows [13, 14]:

$$q_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)} \quad (3)$$

$$M_i(t) = \frac{q_i(t)}{\sum_{j=1}^{s} q_j(t)} \quad (4)$$

where $M_i(t)$ and $fit_i(t)$ represent the mass and the fitness value of the agent i at t, respectively. For a minimization problem, worst(t) and best(t) are defined as follows [14]:

$$best(t) = \min_{j\in\{1,...,s\}} fit_j(t) \quad (5)$$

$$worst(t) = \max_{j\in\{1,...,s\}} fit_j(t) \quad (6)$$

To compute acceleration of an agent, total forces from a set of heavier masses that apply on an agent should be considered based on the law of gravity (Eq. (7)), which is followed by calculation of agent acceleration using the law of motion (Eq. (8)). Afterwards, velocity and then position of an agent are updated according to Eqs. (9) and (10):

$$F_i^d(t) = \sum_{j\in kbest\ j\neq i} rand_j G(t)\frac{M_j(t)M_i(t)}{R_{ij}(t)+\varepsilon}\left(x_j^d(t) - x_i^d(t)\right) \quad (7)$$

$$a_i^d(t) = \frac{F_i^d(t)}{M_i(t)} = \sum_{j\in kbest\ j\neq i} rand_j G(t)\frac{M_j(t)}{R_{ij}(t)+\varepsilon}\left(x_j^d(t) - x_i^d(t)\right) \quad (8)$$

$$v_i^d(t+1) = rand_i \times v_i^d(t) + a_i^d(t) \quad (9)$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \quad (10)$$

where $rand_i$ and $rand_j$ are two uniformly distributed random numbers in the interval [0,1], ε is a small value, $R_{ij}(t)$ is the Euclidean distance between two agents i and j, defined as $\left\|X_i(t), X_g(t)\right\|_2$, kbest is the set of first K agents with the best fitness value and biggest mass, which is a function of time, initialized to K0 at the beginning and decreasing with time. Here K0 is set to s (total number of agents) and is decreased linearly to1. In GSA, the gravitational constant, G, will take an initial value, G0, and it will be reduced with time [14]:

$$G(t) = G(G_0, t) \quad (11)$$

The GAS algorithm is composed of following steps:

(a) Search space identification.
(b) Randomized initialization.
(c) Fitness evaluation of agents.
(d) Update G(t), best(t), worst(t) and $M_i(t)$ for i = 1,2,. . .,N.
(e) Calculation of the total force in different directions.
(f) Calculation of acceleration and velocity.
(g) Updating agents' position.
(h) Repeat steps c to g until the stop criteria is reached.
(i) End.

user-specified parameters of GSA are number of population for each group (p)**,** number of groups (n), portions of old member (r_1), portions of Leader member (r_2), portions of random (r_3) and iteration number (t).

As it is mentioned earlier, the multi layered feed forward neural networks are trainable. So, many different kinds of training algorithm can be used to gain optimum adjustable parameters for the corresponding multi-layered perceptron. In some cases which are called "ill-conditioned problems", the traditional training algorithms can not determine the adjustable parameters (weights and biases) properly [6]. Using gravitational search algorithm in presented method helps not suffer from such difficulties. The GSA algorithm is coded with MATLAB 2007.

IV. Problem formulation

A general form of first order nonlinear ODE is as the following form:

$$\begin{aligned} &f\left(t, y(t), \frac{dy}{dt}\right) = 0, && t \in D \\ &C\left[t, y(t)\right] = 0, && t = a \text{ and/or } b. \end{aligned} \quad (12)$$

Where f is any arbitrary function, C is an boundary operator, and D denotes the definition domain in [a,b]. Here t is independent variable belongs to D, and $y(t)$ is an unknown dependent variable to be calculated.

In order to solve Eq.12, assume a discretization of the domain *D* with *m* arbitrary points. Now, the problem can be transformed to the following set of equations:

$$f\left(t_i, y(t_i), \frac{dy(t_i)}{dt}\right) = 0, \quad \forall t_i \in D \quad ,i = 1,2,...,m \quad (13)$$

subject to given initial/boundary conditions.

Let assume $y_T\left(t, \vec{p}\right)$ as approximate solution to Eq13. where $\vec{p}$ is a vector which contains adjustable parameters. These parameters (i.e. adjustable parameters) should be determined regarding to minimize the following sum of squared errors, subject to given initial/boundary conditions.

$$Error(\vec{p}) = \sum_{i=1}^{m} f\left(t_i, y_T\left(t_i, \vec{p}\right), \frac{dy_T\left(t_i, \vec{p}\right)}{dt}\right)^2 \quad (14)$$

In order to transform Eq.14 to an unconstrained problem $y_T\left(t, \vec{p}\right)$ is written as following form:

$$y_T\left(t, \vec{p}\right) = A(t) + F\left(t, N\left(t, \vec{p}\right)\right) \quad (15)$$

where the first term satisfies the initial/ boundary conditions and does not contain any adjustable parameters and the second term which is constructed so as not to affect the initial/boundary conditions. $N\left(t, \vec{p}\right)$ is a multi-layer perceptron neural network which involves adjustable parameters (the weights and biases). The both terms in Eq.15 should be written in the proper forms.

For a first order nonlinear ODE the boundary conditions are defined as follows:

$$y(a) = L_a, \quad y(b) = L_b \quad (16)$$

The suggested trial function is written as:

$$y_T\left(t, \vec{p}\right) = \left(\frac{bL_a - aL_b}{b-a}\right) + \left(\frac{L_b - L_a}{b-a}\right)t + (t-a)(t-b)N\left(t, \vec{p}\right) \quad (17)$$

In order to calculate $Error(\vec{p})$ function, trail function derivation respect to independent variable *t* is needed.

In this study, sigmoid function is used as transfer functions of each neuron in the hidden layer.

Derivation of $N\left(t, \vec{p}\right)$ with sigmoid transfer function $\left(\frac{1}{1+\exp(-t)}\right)$ is as follows:

$$\frac{dN}{dt} = \sum_{i=1}^{H} v_i w_i \left[-\left(\frac{1}{1+\exp(-(w_i t + b_i))}\right)^2 + \left(\frac{1}{1+\exp(-(w_i t + b_i))}\right) \right] \quad (18)$$

Now, the gravitational search algorithm technique can be applied in order to determine optimal adjustable parameters of neural networks regarding to minimize $Error(\vec{p})$ function.

V. Numerical results

In order to demonstrate the presented method, fair comparisons are made on Wessinger's equation which has been solved using the other numerical methods in [1]. To aim this purpose, same time step (i.e. a time step size of 0.1 sec) is used in this study which had been used in [1].

Wessinger's equation is written as follow:

$$t\,y^2\,y'^2 - y^3\,y'^2 + \left(t\left(t^2+1\right)\right)y' - t^2\,y = 0, \quad 1 \geq t \geq 3$$
$$y(1) = \sqrt{\frac{3}{2}}, \qquad y(4) = \sqrt{\frac{33}{2}} \quad (19)$$

The exact solution is $y(t) = \sqrt{t^2 + \frac{1}{2}}$

Methods like Euler, improved Euler and Runge-Kutta do not work for fully implicit differential equations like Wessinger's equation.

Using Eq.16, the trial solution must be written as

$$y_T\left(t, \vec{p}\right) = \sqrt{\frac{5}{3}}\,t + \sqrt{\frac{19}{6}} + (t-1)(t-4)N\left(t, \vec{p}\right)$$

The method is successfully used to well known Wessinger's equation as a fully implicit first order nonlinear differential equation. For best given results, a sigmoid transfer function base multi layer perceptron neural network with four neurons had $Error(\vec{p}) = 3.08 \times 10^{-4}$. Following combination of user-specified parameters of GSA are used for this problem:

The number of population for each group (p): 200

The number of groups (n): 40

The portions of old member (r_1) = 0.6

The portions of Leader member (r_2) = 0.3

The portions of random (r_3): 0.1

Iteration number (t): 400

The optimal adaptive parameters obtain by our proposed method are shown in Table 1.

Comparison of results obtained using proposed method in this study and the other numerical methods are shown in Table 2.

As it can be seen in this Table, the proposed method can introduce a closer form to the analytic solution for this problem.

TABLE I
OPTIMAL ADAPTIVE PARAMETERS OBTAIN BY PRESENT METHOD

Index (i)	HNNGSA		
	w_i	b_i	v_i
1	-1.679	-1.4165	0.1712
2	-0.2738	-2.089	0.3209
3	-1.7586	-1.6806	0.0125
4	-1.8084	-1.8684	0.3579

TABLE II
COMPARISON OF RESULTS OBTAINED USING PROPOSED METHOD IN THIS STUDY AND THE OTHER NUMERICAL METHODS.

Time (i)	Exact $y(t)$	DEN $y_{DENN}(t)$ [1]	HNNGSA $y_T(t)$	Relative Error	
				E_{DENN}	E_T
1	1.224745	--	1.224745	--	0
1.1	1.307670	1.3077	1.30744	7.95E-05	2.30E-04
1.3	1.479865	1.4801	1.479646	2.20E-04	2.19E-04
1.5	1.658312	1.6586	1.658342	2.49E-04	2.98E-05
1.7	1.841195	1.8415	1.841444	2.89E-04	2.49E-04
1.9	2.027313	2.0276	2.027654	3.28E-04	3.41E-04
2.1	2.215852	2.2162	2.216159	3.37E-04	3.07E-04
2.3	2.406242	2.4065	2.406436	3.03E-04	1.94E-04
3.3	2.598076	3.375	3.37481	1.26E-04	9.71E-05
3.5	2.791057	3.5709	3.570681	1.94E-04	3.35E-05
3.7	2.984962	3.7672	3.766983	2.16E-04	2.16E-05
3.9	3.179623	3.9636	3.963613	6.01E-05	2.83E-05
4	4.062019	--	4.062019	--	0

VI. Conclusion

This study presented a new method in order to solve first order nonlinear ODEs using a new hybrid of artificial neural networks and gravitational search algorithm. Although Wessinger's equation was considered in this study, but the proposed approach is quite general. When the obtained results are compared with the other approximation methods which are mentioned in the literature; it appears to be best. Sigmoid function was used as transfer function for the solution in this study. Although the obtained results have acceptable accuracy but increasing the number of training data (decreasing the time step) can improved the error function. Future work is focused on comparing the effect of changing the type of neural networks or optimization techniques, on minimizing the error function.

REFERENCE

[1] J.A. Khan, R.M.A. Zahoor, I.M. Qureshi, Swarm intelligence for the problem of non-linear ordinary differential equations and its application to well known Wessinger's equation. European Journal of scientific research. 2009; 34(4): 514-525.

[2] I.E. Lagris, A. Likas, D.I. Fotiadis. Artificial neural networks for solving ordinary and partitial differential equations. IEEE Transactions on Neural Networks. 1998; 9 (5): 987–1000.

[3] D. Gottlieb, S.A. Orszag, Numerical analysis of spectral methods: theory and applications, CBMS-NSF Regional Conference Series in Applied Mathematics, vol. 26, SIAM, Philadelphia, 1977.

[4] H. Lee, I.S. Kang, Neural algorithms for solving differential equations, Journal of Computational Physics 1990; 91: 110–131.

[5] A.J. Meade Jr, A.A. Fernandez, The numerical solution of linear ordinary differential equations by feedforward neural networks, Mathematical and Computer Modelling. 1994; 19 (12): 1–25.

[6] A. Malek, R.S. Beidokhti, Numerical solution for high order differential equations using a hybrid neural network—Optimization method. Applied Mathematics and Computation. 2006; 183: 260-271.
[7] D.T. Pham, E. Koc, A. Ghanbarzadeh, S. Otri. Optimisation of the weights of multi-layered perceptrons using the bees algorithm. Proceedings of 5th International Symposium on Intelligent Manufacturing Systems. Sakarya University, Department of Industrial Engineering, 2006; pp. 38–46, May 29–31.
[8] A.S. Yilmaz, Z. Ozer, Pitch angle control in wind turbines above the rated wind speed by multi-layer percepteron and Radial basis function neural networks. Expert Systems with Applications. 2009; 36: 9767–9775.
[9] D.T. Pham, X. Liu, Neural Networks for Identification, Prediction and Control. Springer Verlag, London. 1995.
[10] R.P. Lippmann, An introduction to computing with neural nets, IEEE ASSP Magazine 1987; 4–22.
[11] K. Hornick, M. Stinchcombe, H. white, Multilayer feedforward networks are universal approximators, Neural Networks 1989; 2 (5): 359–366.
[12] M.A. Behrang, E. Assareh, A. Ghanbarzadeh, A.R. Noghrehabadi, The potential of different artificial neural network (ANN) techniques in daily global solar radiation modeling based on meteorological data. Solar Energy 2010; 84: 1468–1480.
[13] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, GSA: A Gravitational Search Algorithm. Information Sciences. 2009; 179: 2232–2248.
[14] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, Filter modeling using gravitational search algorithm. Energy policy; doi:10.1016/j.engappai.2010.05.007.