International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:3, No:7, 2009

Evolutionary Computing Approach for the Solution of Initial value Problems in Ordinary Differential Equations

A. Junaid, M. A. Z. Raja, and I. M. Qureshi

***Abstract*—**An evolutionary computing technique for solving initial value problems in Ordinary Differential Equations is proposed in this paper. Neural network is used as a universal approximator while the adaptive parameters of neural networks are optimized by genetic algorithm. The solution is achieved on the continuous grid of time instead of discrete as in other numerical techniques. The comparison is carried out with classical numerical techniques and the solution is found with a uniform accuracy of $MSE \approx 10^{-9}$.

***Keywords*—**Neural networks, Unsupervised learning, Evolutionary computing, Numerical methods, Fitness evaluation function.

I. Introduction

IN this paper, the problem of first-order ordinary differential equations is considered whose general form can be written as

$$\dot{y} = f(t, y) \qquad Bf = \alpha, \quad a2 \le t \le a1 \tag{1}$$

where y is the solution of the problem in the interval $(a1, a2)$, t is the time and B is the operator defining initial condition. The ordinary differential equations (ODEs) problems are encountered in many practical applications such as physics, engineering design, fluid dynamics and other scientific applications. The exact solutions of ODEs are practically difficult due to its dynamical nature, so the need to approximate the solution arises. In this regard we have numerical algorithms like Euler, Improved Euler, Runge-kutta, Adams Bashforth, Finite Difference, Differential Transform Methods etc [1], [2], [3], [4]. Some of the shortcomings in these traditional methods are reported in [5]. Artificial neural networks (ANN) have been exploited to solve the problems defined by ODEs to overcome the limitations of traditional numerical methods [6], [7]. The solution of ODEs obtained by trained ANNs offer some advantages, such as, computational complexity does not increase with the increase of the number of sampling points and provides rapid calculation for the solution at any given input points [8]. A number of differential equation (DE) problems have been solved using ANNs but only two of them are given for reference. Van Millingen et.al [9] has successfully applied unsupervised feed forward ANN for finding the general solution of magnetohydrodynamic plasma equilibrium problem represented by ODE. Radial basis functions neural network were exploited to design a moving mass attitude control system [10] to control a vehicle with three axis stabilization in intra-atmospheric space.

In this paper the initial value differential equation problems has been solved by evolutionary computing technique. First the ODEs have been modeled by ANNs and then the unknown parameters of ANN are optimized by genetic algorithms (GAs). The results are compared by proposed method with some traditional numerical methods. The proposed method can be efficiently applied as an alternate approach to various fields such as Electromagnetic, Fluid Dynamics and Control problems.

The remainder of this paper is organized as follows. Section II describes the brief introduction to genetic algorithms; section III presents the proposed method. Numerical results and discussions are presented in section IV. Finally section V provides some concluding remarks on the results.

II. Brief introduction of genetic algorithms

GA is global optimization tool based on natural selection and genetic mechanisms. GAs incorporates parallel procedure [11] as well as structured strategy for randomly searching high aptitude points. Generally the GA consists of three fundamental operators: selection, crossover and mutation. GA runs iteratively using its operators randomly based upon some fitness function. Finally it finds and decodes the solution from the last pool of mature strings obtained by ranking of strings, exchanging the portions of strings and changing some bit of the strings. GA works on the survival of fittest strategy. The following basic terms are used in GAs.

Chromosome or Individual: It is a set of genes. Chromosome or individual contains the solution represented in the form of genes.

A. Junaid is with Department of Electronics Engineering, Faculty of Engineering and Technology International Islamic University Islamabad, Pakistan (phone: +92-300-5359557;fax: +92-51-9258025; e-mail: junaid.phdee17@ iiu.edu.pk).

M. A. Z. Raja is with Department of Electronics Engineering, Faculty of Engineering and Technology International Islamic University Islamabad, Pakistan (e-mail: asif.phdee10@iiu.edu.pk).

I. M. Qureshi is associated with the Department of Electrical Engineering, Air University Islamabad, Pakistan as a professor (e-mail: imq313@au.edu.pk).

Gene: It is a part of chromosome and contains only a part of solution

Population: No of individuals present with same length of chromosome.

Fitness: Fitness is the rank value assigned to an individual. It is based on how far or close an individual is from the solution. Greater the fitness value better the solution it contains.

Fitness function: Fitness function is a function which assigns fitness value to the individual. It is problem specific.

Crossover: The process by which the chromosomes from the parents exchange systematically using probabilistic decision. It means change occurs during reproduction. Therefore, the offspring exhibit some traits of the father and some traits of the mother.

Mutation: The process of changing a random gene in an individual. How often to do mutation, how many genes to change, and how big a change to make are adjustable parameters. Suitable mutation avoids the early maturation.

Selection: Selecting individuals for creating the next generation. The aim in selection is to give the fitter individuals a better chance to survive in the next generation. It is against the nature to kill all unfit genes as they may mutate to something handy. Therefore, in selection there is always a tradeoff for better individual and diversity.

The evolutionary algorithms like GAs search the solution using the survival of the fittest strategy. More information about evolutionary algorithm can be found in [12], [13].Traditional flow diagram of GA set up for the optimization of weights in neural network is given in Fig. 1. A standard flow chart of genetic algorithm is given below:

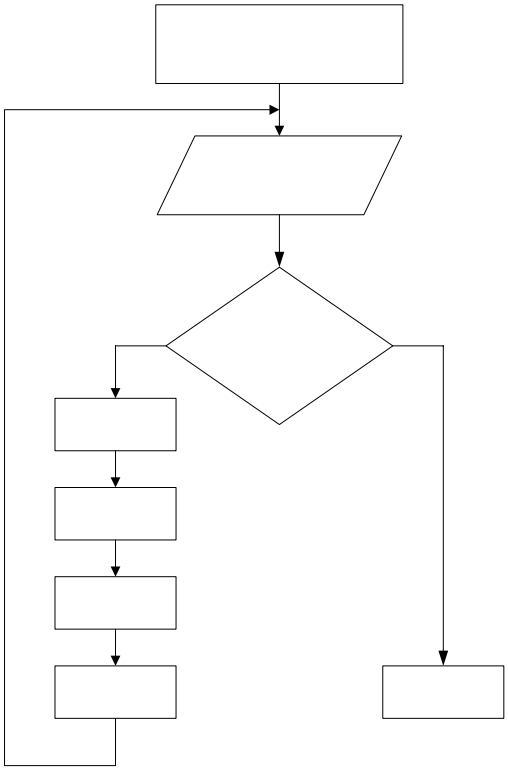

Fig. 1 Flow chart of GA

III. Proposed Method

A mathematical model for the accurate approximation of the solution of ODE is revealed here. This model consists of two parts, in the first part neural network modeling is carried out that depends upon some unknowns. These unknowns are optimized using evolutionary algorithm in the second part.

As a feed forward neural network is a universal function approximator [14], [15]. Any network suitably trained to approximate a mapping satisfying some non-linear ODE will have an output function that will also approximate the DE [16]. For this following continuous mapping is employed:

$$z(x) = \sum_{i=0}^{N} \alpha_i \phi(w_i x + b_i) \tag{2}$$

$$\dot{z}(x) = \sum_{i=0}^{N} \alpha_i w_i \dot{\phi}(w_i x + b_i) \tag{3}$$

for z and $\dot{z}$ respectively, where ϕ being activation function normally taken as log sigmoid function, given below and its characteristics curve is given in Fig. 2.

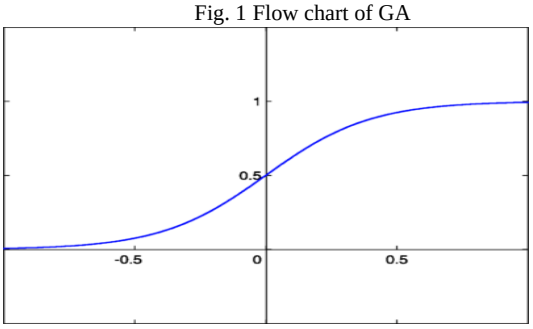

Fig. 2 Log sigmoid activation function

$$\phi(t) = \frac{1}{1+e^{-t}} \tag{4}$$

We have to find such real values of weights α_i, w_i, and b_i for N number of neurons in the hidden layer that approximate the above mapping arbitrary well. These unknown variables are highly correlated to each other due to which the training becomes difficult. These unknowns are randomly generated in a definite range to exploit the solution space. The possible candidate solutions are randomly generated as real values in the specified lower and upper bound. The population m is divided into the P sub populations, each with m/P chromosomes in order to increase the diversity in the solution space. The fitness evaluation function is defined in (5) that is the objective function i.e we have minimized this error function for the approximated solution.

$$e(x) = f^*(t,(z(t),\dot{z})^2 + Bf^*(t,z(t),\dot{z})^2) \quad (5)$$

As $e(x) \to 0$ with optimization, f^* will approach f i.e the solution of the ODE has converged. During the optimization the genetic operator selection takes half of the chromosomes from each subpopulation by taking $(m/2P - 2)$ top ranked individuals and 2 from the worst ranked individuals. While the crossover of two parents X_p and X_q generates two new off-springs, X_a and X_b , with the following elements

$$x_{ak} = \begin{cases} x_{qk} & k > i \\ x_{pk} & k \le i \end{cases}$$
$$x_{bk} = \begin{cases} x_{qk} & k \le i \\ x_{pk} & k > i \end{cases} \quad (6)$$

where i is a random integer in the range 1 to $n-1$ and n is the length of the chromosome. Few new mutated chromosomes are included by small random constant value addition in some elements of chromosomes selected at random.

```
// Initialization
For each chromosome i
        Randomly initialize x_i for chromosome i
End for
// Optimization
Do
        For each chromosome i
            Call calculate_fitness_value
      If (current_fitness_value > previous_fitness_value)
      Then
Fit_value=current_fitness_value
        End if
    End for

      For each chromosome i
            Call selection_procedure
            Call crossover_procedure using eq. (6)
            Call mutation_procedure
            Call formulation_new_population
          End for
While Max_iterations or min_value is not attained
```

Fig. 3 Pseudocode of GA

IV. Results and Discussion

For the validity of the given method a number of examples has been tested. There are a few examples presented in this paper.

A. Example 1:

Given an initial value problem in ODEs of the form defined in (1)

$$\left\{ \begin{array}{ll} \dot{y} = y - \dfrac{2t}{y} & t \in [0-1] \\ y(0) = 1 \end{array} \right\} \quad (7)$$

$\alpha = y(0) = 1,\ \dot{y}(0) = 1$, the number of neurons consist of $N = 8$ so the number of adaptive variables optimized are 24. We have restricted the values of these parameters in the interval [-5, 5] and noticed this thing that it gives the better results in this range. An initial population of 240 individuals has been taken which is divided over 10 subpopulations each with 24 individuals. The input of the training set is chosen from $t \in (0,1)$ with a step of 0.1. The fitness function is evaluated using the criteria i.e $e(x) \le 10^{-10}$ or program is run for 400 generations which ever comes earlier. The analytical solution of this problem is evaluated for the same inputs. The numerical solution for this problem for the same input times is obtained by Euler, improved Euler, Rk and our proposed method. The results are summarized in the table 1 from which it is quite clear that our approach is better than the Euler and improved Euler while it is comparable with RK method.

TABLE 1
COMPARISON OF SOME NUMERICAL METHODS AND EVOLUTIONARY COMPUTING METHOD

t	Exact $y(t)$	Euler	Improved Euler	Runge Kutta	Evolutionary Computing $\hat{y}(t)$
0.0	1.0000	1.0000	1.0000	1.0000	1.0000
0.1	1.0954	1.1000	1.0959	1.0954	1.0955
0.2	1.1832	1.1918	1.1841	1.1832	1.1831
0.3	1.2649	1.2774	1.2662	1.2649	1.2650
0.4	1.3416	1.3582	1.3434	1.3417	1.34181
0.5	1.4142	1.4325	1.4164	1.4142	1.4143
0.6	1.4832	1.5090	1.4860	1.4833	1.4833
0.7	1.5492	1.5803	1.5525	1.5492	1.54930
0.8	1.6125	1.6498	1.6165	1.6124	1.6127
0.9	1.6733	1.7178	1.6782	1.6733	1.6734
1.0	1.7321	1.7848	1.7379	1.7320	1.7321

B. Example 2:

Large Given another initial value problem in ODEs

$$\left\{ \begin{array}{ll} \dot{y} = y - t^2 + 1 & t \in [0,2] \\ y(0) = 0.5 \end{array} \right\} \quad (8)$$

$\alpha = y(0) = 0.5,\ \dot{y}(0) = 1,$ the number of neurons consist of $N = 8$ so the number of adaptive variables optimized are 24. We have restricted the values of these parameters again in the same interval of [-5, 5] and noticed that it gives better results in this range. An initial population of 200 individuals has been taken which is divided over 10

subpopulations, each with 20 individuals. The input of the training set is chosen from $t \in (0.1)$ with a step of 0.2. The fitness function is evaluated using the criteria i.e $e(x) \le 10^{-10}$ or program is run for 500 generations in this time which ever comes earlier. The analytical solution of this problem is evaluated for the same inputs. The numerical solution for this problem for the same input times is obtained by Euler, improved Euler, RK and our proposed method. The results are summarized in the table 2 from which it is quite clear that our approach is better than the Euler and improved Euler while it is comparable with RK method.

TABLE II
COMPARISION OF SOME NUMERICAL METHODS AND EVOLUTIONARY COMPUTING METHOD

t	Exact $y(t)$	Euler	Improved Euler	Runge Kutta	Evolutionary Computing $\hat{y}(t)$
0.0	0.5000	0.5000	0.5000	0.5000	0.5000
0.2	0.8292	0.8273	0.8260	0.8293	0.8294
0.4	1.2141	1.2099	1.2069	1.2141	1.2142
0.6	1.6489	1.6421	1.6372	1.6489	1.64877
0.8	2.1272	2.1176	2.1102	2.1272	2.1271
1.0	2.6408	2.6280	2.6177	2.6408	2.6406
1.2	3.1799	3.1635	3.1499	3.1799	3.1798
1.4	3.7324	3.7120	3.6973	3.7323	3.7326
1.6	4.2835	4.2588	4.2351	4.2834	4.2833
1.8	4.8152	4.7858	4.7556	4.8151	4.8150
2.0	5.3055	5.2713	5.2330	5.3053	5.3055

V. CONCLUSION

As is evident from the table, the results of proposed method are more precise as compared to Euler and Improved Euler. However, compared to Runge-Kutta method with order four the results of calculation are less precise. But the presented approach provides an alternate method to solve ODEs. Another advantage is that it gives the approximate solution on the continuous finite time domain whereas other numerical techniques provide the solution on discrete time only. Once learning and optimization is performed by given technique, then we can find the solution of ODEs readily at some given input time t within the finite domain using these unknown weights without repeating the procedure. This will reduce the time and space complexity of this problem. The method provides the alternate method for finding the solutions of ODEs associated with complex real life problems.

VI. REFERENCES

[1]. D. Kahaner, C. Moler, S. Nash, "Numerical Methods and Software," Prentice-Hall, New Jersey, 1989.
[2]. K. Kunz, R. Luebbers, "Finite difference time domain method for electromagnetic," Boca Raton, CRC Press. 1993.
[3]. J.R Dormand, P.J Prince, "A family of embedded Runge-Kutta formulae," Comp. Appl. Math. J., vol. 6, pp. 19, 1980.
[4]. MJ. Jang, CL. Chen, YC. Liu, "On solving the initial-value problems using the differential transform method," J. Comp. Appl. Math. J., vol. 115, pp. 145-160, 2000.
[5]. L.F. Shampine, M. W. Reichelt, "The MATLAB ode suite," SIAM Scientific Computing J., vol.18, pp. 1-22, 1997.
[6]. M.H Li, G.J Wang, "Solving differential equation with constant coefficient by using radial basis function," Shen Yang Institute of Chemical Technology J., vol.20, pp. 68-72, 2006.
[7]. A.J Meada, A.A Fernandez, "The numerical solution of linear ordinary differential equation by feedforward neural network," Math Compute Modeling J., vol.19, pp. 1-25, 1994.
[8]. C. Monterola, C. Saloma, "Characterizing the dynamics of constrained physical systems with unsupervised neural network," Phys Rev E 57, pp.1247R–1250R, 1998.
[9]. B.Ph van Milligen, V.Tribaldos, J.A.Jimenez, "Neural network differential equation and plasma equilibrium solver," Physical Review Letters 75, pp. 3594-3597, 1995.
[10]. L.Qin, M.Yang, moving mass attitude law based on neural network, in proc. 6th Int. Conf. machine learning and cybernetics, ICMLC, 5, art. No. 4370622, 2007, pp.2791-2795.
[11]. J.H. Holland, "Adaptation in natural and artificial systems," Ann arbor, MI, University of Michigan press, 1975.
[12]. C.R. Houck, J.A. Joines, M.G. Kay, "A genetic algorithm for function optimization," A matlab implementation, Technical Report NCSU-IE TR 95-09, North Carolina State University, Raleigh NC,1995.
[13]. Zbigniew Michalewicz, "Genetic algorithms + data structure= Evolution programs," 2nd ed, New York: Springer-verlag, Berlin, 1994.
[14]. R. Hetch-Nielsen, "Kolmogorov's mapping neural network existence theorem," 1st 1987 IEEE Int. Conf. Neural networks, San Diego CA, 3, pp. 11.
[15]. K.I. Funahashi, "On the approximate realization of continuous mappings by neural networks," Neural Networks J., vol. 2.issue 3, pp.183-192, 1989.
[16]. C. Chen, "Degree of approximation by superposition of sigmoidal function," Analysis in theory & appl J., vol. 9, no 3, pp. 17-28, 1993.