An Archived firefly Algorithm; A mathematical software to solve univariate nonlinear equations

M.K.A. Ariyaratne, T.G.I. Fernando, S. Weerakoon

Abstract— **In this article, we are presenting a software solution that proposes some modifications to the existing firefly algorithm. The modification; archived firefly algorithm [AFFA] exhibits the ability of finding almost all complex roots of a given nonlinear equation within a reasonable range. The software implementation includes two main properties; an archive to collect the better fireflies and a flag to determine poor performance in firefly generations. The new modification is tested over genetic algorithms (GA), a phenomenal success in the field of nature inspired algorithms and also with a modified GA embedded with same properties that the AFFA has. A simple graphical user interface (GUI) is developed using MATLAB GUIDE to present the findings. Computer simulations show that the AFFA performs well in solving nonlinear equations with real as well as complex roots within a specified region.**

Keywords—**Firefly Algorithm, Nonlinear Equations, Archive, Real Roots, Complex Roots.**

I. INTRODUCTION

The computer scientists are looking forward to find various approaches that can contribute towards optimization. The future of optimization is now being conquered by modern meta-heuristic algorithms. Genetic algorithms, differential evolution, harmony search, firefly algorithm and cuckoo search are such meta-heuristic algorithms which have marked their success over many optimization tasks. Simplicity of the algorithm, less memory consumption and the minimal error of the approximations can be stated as the major reasons for their popularity. Natural optimization techniques are among them and are becoming popular due to the long lasting existence of such practices in the real world. Genetic algorithms; which have come to the stage around 1970 goingalong with the theory of evolution by Charles Darwin, can be identified as one of the first such algorithms. They still play an important role among nature inspired algorithms [1], [2]. Ant colony optimization algorithms which mimic the ants' strange communication behaviour are other popular algorithms which have been adopted for many real world optimization tasks [3], [4]. Bat inspired algorithm [5], cuckoo bird algorithm [6], firefly algorithm [7] are more recent nature inspired algorithms that represent different real world optimized phenomena. These algorithms can be classified into two main categories as evolutionary algorithms and swarm intelligence algorithms.

Evolutionary algorithms are population based algorithms which use mutation, recombination, and natural selection to reproduce better generations [8]. Genetic algorithms, differential evolution [9] and genetic programming [10] are examples for evolutionary algorithms. Swarm intelligence, by its name mimics the collective behaviour of different elements in the natural world. Particle swarm optimization [11], ant colony systems, firefly algorithms are some examples. One of the most advantageous properties of these algorithms is that most of them are of meta-heuristic type. Therefore these algorithms can be adopted to solve variety of optimization problems rather than heuristic algorithms. This paper presents the research carried out using a modern nature inspired algorithm; firefly algorithm to solve univariate nonlinear equations having real as well as complex roots. We also present the mathematical software that has been developed to accomplish the task.

II. NONLINEAR EQUATIONS

Solving a nonlinear equation f(x)=0 is finding roots α, such that f(α)=0. Many numerical methods exist to solve such nonlinear equations, but they have some drawbacks such as the need for derivative information, strong dependence on the initial guess and the inability to give all the roots within an interval simultaneously. When the equation is having complex roots, the situation is even more difficult. Scientists have moved towards finding better algorithms to minimize these drawbacks. But we have to ensure the efficiency of the computational effort as well. The algorithms should maintain the speed, accuracy and low memory consumption. The research literature reveals that many improvements have taken place to tune these existing approaches as well as in finding new ways to solve nonlinear equations reducing the above mentioned drawbacks.

An improvement to the Newton's method is suggested by Weerakoon & Fernando in their joint paper: A Variant of Newton's Method with Accelerated Third-Order Convergence [12]. Their method involves changing the derivation of Newton's method. The derivation of the Newton's method involves approximating an indefinite integral of the derivative of the function by a rectangle. In their research Weerakoon & Fernando have modified it to be a trapezium so that the error of the approximation is reduced. They have proved that the order of convergence of the suggested modification is three. In fact, for some cases it is even higher than three. The main concern in this research was on accuracy of the approximation. However this method also contains the aforementioned drawbacks of numerical methods such as the need for derivative calculation.

Moving beyond numerical methods we could come across a handful of research done with the nature inspired algorithms to fulfill the task. Use of genetic algorithms to solve systems of nonlinear equations is addressed in some researches [13], [14], [15]. A hybrid algorithm implemented with genetic algorithms and particle swarm optimization also has been tested in a research [16]. Applicability of harmony

search in solving system of nonlinear equations is also addressed [17].

Heuristics were also in use of solving systems of nonlinear equations. One research has introduced the use of continuous global optimization heuristic called "continuous GRASP" to solve a nonlinear system [18]. They are addressing the problem of finding all the roots of a system of equations assuming that all roots are real.

Most of these approaches have focused on solving systems of nonlinear equations rather than a single equation. In almost all these research, they have dealt only with the real roots. The problem of finding all roots in a reasonable range within a single run is also not addressed.

Our problem of interest is to solve a univariate nonlinear equation having real and/or complex roots. We need to find all the roots within a reasonable interval/range. Since we have interpreted the problem as an optimization problem, we can define the above problem as follows.

Let f be a function s.t. f: D→R where D⊂C, where C is the set of Complex Numbers. The problem is to find all x ∈ D s.t. f(x)= 0, without requiring either the differentiability or the continuity of the function f. Thus we need to find x ∈ D s.t. |f(x)|=0. However, since the function f(x) may have multiple roots, the optimization problem |f(x)|=0, also will have multiple optimal solutions. Our objective turns out to be finding all such optimal solutions.

III. ARCHIVED FIREFLY ALGORITHM

AFFA is the software solution proposed in this paper to solve univariate nonlinear equations with real as well as complex roots. Many nature inspired algorithms are good solution providers for various optimization tasks and their algorithms are very simple. That made it easier for the researchers to develop these algorithms for different optimization tasks. Firefly algorithm is a newly implemented such algorithm with the following assumptions about fireflies' behaviour [7].

1. Attraction of the fireflies to each other is gender independent.
2. Attractiveness is proportional to the brightness, for any two fireflies, the less bright one will be attracted and move towards the brighter one, this attraction decreases when distance increases, the brightest firefly will move randomly.
3. Brightness of a particular firefly is determined by its objective function.

The following pseudo code describes the original firefly algorithm and a simple 20 line MATLAB program can implement the algorithm solving a given mathematical optimization problem. MATLAB as a mathematical software provides an easy environment to implement these types of algorithms. It is also capable of creating user interfaces and enrich with plotting different functions and data so that the solutions can be graphically represented.

Algorithm 1: Original Firefly Algorithm
$Begin$;
$Initialize\ algorithm\ parameters$:
$MaxGen$: the maximum number of generations
γ: the light absorption coefficient
r: the distance between two fireflies
D: the domain space

$Define\ the\ objective\ function\ f(X),\qquad where$
$$X = (x_1, \ldots, x_d)^T$$
$Generate\ the\ initial\ population\ of\ fireflies, X_i\quad (i = 1,2,\ldots,n)$

$Determine\ the\ light\ intensity\ I_i\ of$
$$i^{th}\ firefly\ X_i\ via\ f(X_i)$$
$\mathbf{while}\ t < MaxGen\ do$
$\quad\quad\mathbf{for}\ i = 1:n\ (all\ n\ fireflies)\ do$
$\quad\quad\mathbf{for}\ j = 1:n\ (all\ n\ fireflies)\ do$
$\mathbf{if}\ I_j > I_i$

$\quad\quad Move\ firefly\ i\ towards\ j\ by\ using\ eq\ (1);$
$\quad\quad\quad\mathbf{End}\ if$
$\quad Attractiveness\ varies\ with\ distance$
$\quad r\ via\ e^{-\gamma r^2}\ using\ eq\ (2);$
$\quad Evaluate\ new\ solutions\ and\ update$
$\quad light\ intensity;$
$\quad\quad\quad\quad\mathbf{end}\ for$
$\quad\quad\quad\quad\mathbf{end}\ for$
$\quad Rank\ the\ fireflies\ and\ find\ the\ current\ best;$
$\mathbf{end}\ while$
$Post\ process\ results\ and\ visualization;$
$\mathbf{End};$

The initial population can be defined randomly with a set of feasible solutions for the problem. Then each firefly's light intensity is calculated using the problem specific objective function. Then each firefly in the population starts moving towards brighter fireflies according to the following equation.

$$x_i = x_i + \beta(x_j - x_i) + \alpha(rand - 0.5);\ \rightarrow (1)$$

$where \beta = \beta_0 e^{-\gamma r^2} \rightarrow (2),$
$\beta_0\ is\ the\ attraction\ at\ r = 0;$

The second term of the .1 is due to the attraction between two fireflies and the third term is a randomization term where α is the randomization factor drawn from the Gaussian or the uniform distribution.

In the modified version we have added a few more qualities to the original algorithm by introducing an archive and a flag. The new algorithm is named after its archiving property as archived firefly algorithm. The pseudo code of the modified algorithm is as follows.

Algorithm 2:Archived Firefly Algorithm
$Begin$;
$Initialize\ algorithm\ parameters$:
$MaxGen$: the maximum number of generations
γ: the light absorption coefficient
r: the distance between two fireflies
D: the domain space
$Define\ the\ objective\ function\ f(X), where$
$$X = (x_1, \ldots\ldots, x_d)^T$$

$Generate\ the\ initial\ population\ of\ fireflies, X_i$
$$(i = 1,2,\ldots\ldots,n)$$
$Determine\ the\ light\ intensity\ I_i\ of$
$$i^{th}\ firefly\ X_i\ via\ f(X_i)$$
$\mathbf{while}\ t < MaxGen\ do$
$\mathbf{flag = true};$
$\mathbf{while}\ flag = true\ \mathbf{and}\ t < MaxGen\ do$

$\textbf{for } i = 1 : n \ (all \ n \ fireflies) \ do$
$\textbf{for } j = 1 : n \ (all \ n \ fireflies) \ do$
$\textbf{if } I_j > I_i$

$Move \ firefly \ i \ towards \ j \ byusing \ eq \ (1);$
$\textbf{End } if$
$Attractiveness \ varies \ with \ distance$
$r \ via \ e^{-\gamma r^2} \ using \ eq \ (2);$
$Evaluate \ new \ solutions \ and \ update$
$light \ intensity;$
$\textbf{end } for$
$\textbf{end } for$
$\textbf{Find the fireflies with the eligibility criteria}$
$|f(X)| < 0:001;$
$\textbf{Put them into the archive and replace the}$
$\textbf{positions with random fireflies};$

$\textbf{If } no \ fireflies \ matching \ with \ eligibility \ criteria$
$flag = false;$
$\textbf{End } if$

$\textbf{If } flag = false$

$count \ = \ random \ integer \ between \ 0 \ and \ n;$
$Create \ random \ fireflies \ up \ to \ count \ and$
$replace \ the \ population;$

$\textbf{End } if$
$\textbf{end } while$
$\textbf{end } while$
$Post \ process \ results \ and \ visualization;$
$\textbf{End};$

IV. MATLAB APPLICATION

A MATLAB software is developed successfully to graphically represent the results of AFFA. To create graphical user interfaces, MATLAB GUIDE (graphical user interface design environment) is used [20]. MATLAB GUIDE provides tools for designing user interfaces for custom applications. Using the GUIDE layout editor, we have designed a simple user interface. GUIDE then automatically generate the MATLAB code for the user interface, and the user can modify the program to control the application.

The first step was taken to solve nonlinear equations with real roots only. The user has to provide the equation, the lower and the upper bounds to specify the interval to find roots and the values for problem specific parameters (such that population size, number of iterations, alpha and gamma values). Thenthe simulator will display the root approximations. Fig.1 shows the simple GUI created in the MATLAB environment which is implemented to display the results graphically.

Fig. 1: A MATLAB GUI implemented to solve univariate nonlinear equations with real roots

When it comes to complex roots the environment is different. When the function is having real as well as complex roots, plotting the function and displaying the roots is sometimes not clear; instead we used real and imaginary axes to plot the roots. Apart from that we added a text box to display the root approximations. With all these modifications the new GUI to solve univariate nonlinear equations with real and/or complex roots is as follows.
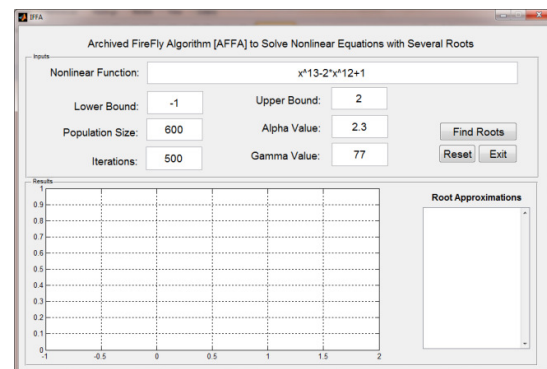

Fig. 2: A MATLAB GUI implemented to solve univariate nonlinear equations with real and/or complex roots

The new application displays the roots in an argand diagram and the root approximations are displayed in a list box. Because of the archiving property we can expect more than one approximation for an existing root. Apart from these, the application has a reset button to reset the inputs.

We have solved nonlinear functions having real as well as complex roots using the new application.

$$y = x^{13} - 2x^{12} + 1 \ \rightarrow (3)$$

The above polynomial has 13 roots; 3 real and 10 complex roots. We have used our application to find the root approximations with the accuracy of $10^{(-3)}$
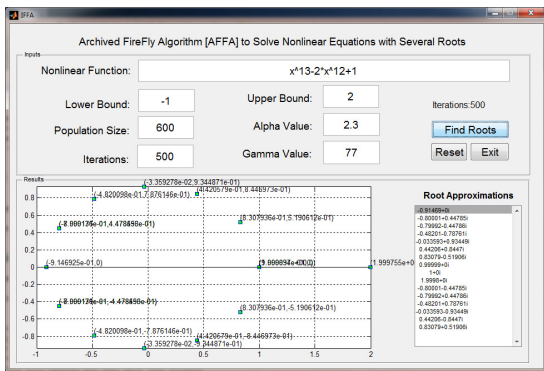

Fig. 3: Root approximations given by the application for the Eqn. 3.

V. NUMERICAL EXAMPLES

Several nonlinear functions are tested against the proposed MATLAB application of AFFA. Mainconcern is to identify different varieties of nonlinear equations where the modified algorithm can be applied. The following equations represent different types of nonlinear equations tested with the AFFA.

a. The following one dimensional trigonometric equation (adapted from Goldberg and Richardson, 1987) [2] has 51 real roots within the given interval and the AFFA approximated all 51 roots with an accuracy of 10^{-3}. This equation proves the ability of AFFA in finding all the real roots of a nonlinear equation within a given interval.

$$y = si\,n^3(5\pi x) \rightarrow (4)$$
$$where\ x \in [-5, 5]$$

b. Usual numerical methods need the equation to be differentiable to apply those methods to find roots. The Weierstrass functions possesses the property of being continuous everywhere but differentiable nowhere [19]. Therefore numerical approaches are not suitable for such functions. But the AFFA as a mathematical software application for a modified nature inspired algorithm is capable of handling such functions successfully. Here we present an example Weierstrass function having 25 real roots within [-20, 20] interval.

$$W(x) = \sum_{i=1}^{3} \left(\frac{1}{2^i}\right) \sin(2^i x) \rightarrow (5)$$

c. Numerical methods also require the property of continuity of a nonlinear function to be solved.
$y = (x)$ is a popular nonlinear function with discontinuities. Our solution does not need the continuity of the function and is capable of approximating all the roots in a large interval. (As an example we have used the interval [-40, 40].

$$y = \tan(x) \rightarrow (6)$$

d. There are some nonlinear functions where there is additional difficulty in calculating the roots. These are functions in which the desired root has a multiplicity greater than 1. Let α be a root of the function f(x), and assume that it could be factored into the form f(x)=(x-α)^m h(x), with some integer m $\geqslant$1 and some continuous function h(x) for which h(α) $\neq$ 0. Then we say that α is a root of f(x) with multiplicity m. One of the main difficulties with the numerical calculation of multiple roots is that methods such as Newton's method and the secant method do not converge in general.
We have solved the following nonlinear equation and found that our approach is successful in finding root approximations for the multiple roots.

$$y = x^4 - 2x^2 + 1 \rightarrow (7),$$
$$where\ x \in [-2, 2]$$

e. The suggested application is capable of finding complex roots too. When the function is having both real as well as complex roots within a given range, the application calculates accurate approximations. The following equations were tested using the software.

TABLE I
ROOT APPROXIMATIONS GIVEN BY THE APPLICATION FOR THE EQN. 3.

	Equation	Region	#of Roots
1	$y = x^2 + 1$	[-1.5,1.5] X [-1.5, 1.5]	2 complex
2	$y = x^3 + 2x^2 + 3x + 4$	[-2, 0] X [-2, 0]	2 Complex, 1 Real
3	$y = x^5 - 3x^4 + 3x^3 - 2x^2 - 5$	[-1, 3] X [-1, 3]	4 Complex, 1 Real
4	$y = x^7 - x^6 + 2x^5 - 3x^4 + 3x^3 - 2x^2 - 5$	[-1, 2] X [-1, 2]	6 Complex, 1 Real
5	$y = x^{10} - 3x^9 + x^8 - 7x^7 + x^6 - x^4 + 2x^2 - 5$	[-1, 4] X [-1, 4]	8 Complex, 2 Real
6	$y = x^{12} - 6x^{11} + x^{10} - 5$	[-1, 6] X [-1, 6]	10 complex, 2 Real
7	$y = x^{13} - 2x^{12} + 1$	[-1, 2] X [-1, 2]	10 Complex, 3 Real

Here also we seek roots within a specified region. Region of the function is the area we seek for the roots. According to the notation we adopted, [-1, 2] X [-1, 2] region describes the area ABCD shown in Fig. 4.
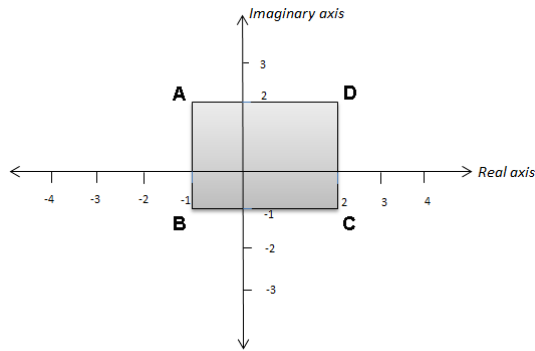
Fig. 4: [-1, 2] X [-1, 2], an example region used for the root finding

VI. RESULTS AND DISCUSSION

The quality of the developed software was compared with the quality of each of the three algorithms: the original firefly algorithm, the original genetic algorithm and also with a genetic algorithm embedded with the new features of AFFA. For the comparison purpose we have set the population size to 200 and the number of iterations per run to 200, as a reasonable scale for real roots and for both real as well as complex roots we allowed both of these values to be 600. We have run our application for 100 times (100 runs) and the average number of roots given by each algorithm, (maximum number of roots found / total roots)*100% are calculated. As an example in Table 2, a result 19, (49%) indicates the average number of roots found in 100 runs is 19 and the (maximum number of roots found in a run / total roots)*100% is 49%.

TABLE II
PERFORMANCE OF THE ALGORITHMS FOR REAL ROOTS
OVER 100 RUNS

I. FUNCTION	GA	Modified GA	FA	AFFA
$y = \sin^3(5\pi x)$	0, (0%)	51, (100%)	19, (49%)	51, (100%)
$y = \sum_{i=1}^{3} \left(\frac{1}{2^i}\right) sin(2^i\, x)$	1, (4%)	25, (100%)	13, (68%)	25, (100%)
$y = \tan(x)$	1, (4%)	25, (100%)	21, (96%)	25, (100%)
$y = x^4 - 2x^2 + 1$	2, (50%)	4, (100%)	4, (100%)	4, (100%)

The real roots situations were smoothly handled by both modified genetic algorithms and the proposed AFFA. The archiving property and diversifying the population during iterations is the reason for the good performance. When it comes to original algorithms, firefly algorithm performs better than GA.

TABLE III
PERFORMANCE OF THE ALGORITHMS FOR REAL AS
WELL AS COMPLEX ROOTS OVER 100 RUNS

EQUATION	GA	Modified GA	FA	AFFA
Table 1: Equation 1	0, (0%)	2, (100%)	2, (100%)	2, (100%)
Table 1: Equation 2	1, (33%)	1, (33%)	3, (100%)	3, (100%)
Table 1: Equation 3	1, (20%)	1, (20%)	5, (100%)	5, (100%)
Table 1: Equation 4	1, (14%)	1, (14%)	7, (100%)	7, (100%)
Table 1: Equation 5	1, (10%)	1, (10%)	9, (90%)	10, (100%)
Table 1: Equation 6	1, (8%)	3, (25%)	11, (92%)	12, (100%)
Table 1: Equation 7	1, (8%)	2, (15%)	8, (92%)	13, (100%)

The modified genetic algorithm performed well for real roots but it fails in finding complex roots. Experiments done with larger population sizes (1200, 1500) were able to give complex roots to some extent. To solve for complex roots, modified GA needs a large chromosome population under the given parameter setting to provide approximations. To check its availability over complex roots, we have to do more research on its recombination and selection parameters. With the results obtained we claim that AFFA under same conditions performed well for the complex roots as well. It is capable of handling nonlinear equations having both real as well as complex roots. The suggested method can be further extended to solve a given system of nonlinear equations for real as well as complex roots.

VII.   CONCLUSIONS

The implementation of the new software to find roots of a univariate nonlinear equation having real as well as complex roots is successful. MATLAB, mathematical package is used to implement the application and the graphical user interface design environment of MATLAB (GUIDE) is also applied successfully. The modification is successful within a reasonable interval/ region. The modification includes an archive to collect best fireflies during iterations and replacing their positions with random ones. Also a flag was used to identify poor iterations and to change the randomness of the existing population. The simulation results for finding roots of several numerical examples including an application of Weierstrass function and complex polynomials suggest that this new firefly algorithm with the archiving property is capable of finding almost all real as well as complex roots of a nonlinear equation simultaneously with the given accuracy of 10^{-3}. It works for the multiple roots functions too. Comparison with other similar nature inspired algorithms including the original firefly algorithm clearly shows that this modified firefly algorithm outperforms all of them. This evidence suggests that the proposed firefly algorithm is by far the best performer in solving nonlinear equations with several real as well as complex roots.

Although the implemented software works well we would like to suggest some modifications to improve it further. The

algorithm lacks a good parameter optimization procedure. That is anyone who uses the software has to enter algorithm specific parameters by themselves. Since the users are not experts in the field of nature inspired algorithms we can't expect them to enter good parameter values. In this initial step we have set all parameters to some reasonable values where most of the equations can be dealt with. But research is open to find methods to do a goodparameter optimization for the algorithm for a given situation.

With the results obtained, we can conclude that the proposed firefly algorithm is capable of giving reasonably good approximations for the nonlinear functions:

  a. with several roots.
  b. with multiple roots.
  c. which are continuous but not differentiable.
  d. which have discontinuities within the interval.
  e. which are having a pattern within the given range.
  f. which have roots over a large interval.
  g. having complex roots as well as real roots in a given interval.

Accuracy of the roots found by the modified FA is within 10^{-3} tolerance. For higher accuracies one can treat these solutions as initial guesses and try out a suitable numerical approach. The accuracy of the solutions is limited to 10^{-3} because our objective here is to find almost all the roots within the given region. The approximations provided here highly depend on the population size, number of iterations and also on the algorithm specific parameter values. It is essential to define the number of iterations and the population size properly according to the number of roots within the specified interval.

Differentiability and continuity of the nonlinear functions are inessential when applying nature inspired algorithms to obtain roots; thus they could be applied to the functions arising from various practical situations where it is impossible to apply formal numerical schemes. This can be considered as the biggest advantages of using nature inspired algorithms.

The basic firefly algorithm introduced by Yang is powerful, but the problem of finding all real as well as complex roots of a given nonlinear equation simultaneously has not been addressed before. Thus our approach of introducing an archive is undoubtedly advantageous. The software with a graphical user interface gives users a user-friendly environment to use the application. But still this approach needs higher number of iterations and alarge firefly population when we need higher accuracies in approximations. As an improvement, we can test parameter optimization techniques which are able to adapt with our algorithm. Apart from that, as a further improvement, one can check the ability of the given procedure in solving a given system of nonlinear equations.

REFERENCES

[1] Goldberg, D. E. (1989). Genetic algorithms in search, optimization, and machine learning (16th ed.). Reading, MA: Addison-Wesley Educational Publishers.

[2] Goldberg, D., & Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. Proceedings Of The Second International Conference On Genetic Algorithms On Genetic Algorithms And Their Application, 41-49. Retrieved from http://dl.acm.org/citation.cfm?id=42512.42519

[3] Dorigo, M., & Gambardella, L. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions On Evolutionary Computation, 1(1), 53-66. http://dx.doi.org/10.1109/4235.585892

[4] Dorigo, M. (1992). Optimization, Learning and Natural Algorithms, (Ph.D). Politecnico di Milano.

[5] Yang, X. (2010). A New Metaheuristic Bat-Inspired Algorithm. Nature Inspired Cooperative Strategies for Optimization (NICSO 2010) Studies in Computational Intelligence, 65-74. doi:10.1007/978-3-642-12538-6_6

[6] Yang, X., & Deb, S. (2009). Cuckoo Search via Levy flights. 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC). doi:10.1109/nabic.2009.5393690

[7] Yang, X. (2009). Firefly Algorithms for Multimodal Optimization. Stochastic Algorithms: Foundations and Applications Lecture Notes in Computer Science, 169-178. doi:10.1007/978-3-642-04944-6_14

[8] Back, T. (1996). Evolutionary algorithms in theory and practice. New York: Oxford University Press.

[9] Rainer, S., & P. K. (1997). Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. Journal of Global Optimization, 11, 341-359. http://dx.doi.org/10.1023/A:1008202821328

[10] Poli, R., Langdon, W. B., & McPhee, N. F. (2008). A Field Guide to Genetic Programming. Retrieved from http://www.gp-field-guide.org.uk

[11] Kennedy, J., &Eberhart, R. (1995). Particle swarm optimization. Proceedings Of ICNN'95 - International Conference On Neural Networks. http://dx.doi.org/10.1109/icnn.1995.488968

[12] Weerakoon, S., & Fernando, T. (2000). A variant of Newton's method with accelerated third-order convergence. Applied Mathematics Letters, 13(8), 87-93. http://dx.doi.org/10.1016/s0893-9659(00)00100-2

[13] El-Emary, I. M., & El-Kareem, M. A. (2008). Towards using genetic algorithm for solving nonlinear equation systems. World Applied Sciences Journal, 5(3), 282-289.

[14] Mastorakis, Nikos E. (2005). Solving non-linear equations via genetic algorithms. In Proceedings of the 6th WSEAS international conference on Evolutionary computing (EC'05), Ana Maria Madureira (Ed.). World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, 24-28.

[15] Pourrajabian, A., Ebrahimi, R., Mirzaei, M., & Shams, M. (2013). Applying genetic algorithms for solving nonlinear algebraic equations. Applied Mathematics And Computation, 219(24), 11483-11494. http://dx.doi.org/10.1016/j.amc.2013.05.057

[16] Biswas, A., &Dasgupta, S. (2008). Finding Solution Of Simultaneous Non-linear Equations Using GA-PSO Hybrid Algorithm.

[17] Solving systems of nonlinear equations by harmony search. (2013). 13Th International Conference Computational And Mathematical Methods In Science And Engineering, IV, 1176-1186. Retrieved from http://hdl.handle.net/1822/27238

[18] Hirsch, M., Pardalos, P., &Resende, M. (2009). Solving systems of nonlinear equations with continuous GRASP. Nonlinear Analysis: Real World Applications, 10(4), 2000-2006. http://dx.doi.org/10.1016/j.nonrwa.2008.03.006

[19] Duncan, L. (n.d.). The Weierstrass Function. Retrieved from https://books.google.lk/books?id=pEbDNwAACAAJ

[20] MATLAB Help Files. GUI Building Basics", MathWorks. Url: http://in.mathworks.com/help/matlab/gui-building-basics.html