2016

A self-tuning modified firefly algorithm to solve univariate nonlinear equations with complex roots

M K A Ariyaratne
Department of Computer Science
Faculty of Computing
General Sir John Kotelawala Defence University
Ratmalana, Sri Lanka
Email: anuradhaariyaratne@gmail.com

T G I Fernando
Department of Computer Science
Faculty of Applied Sciences
University of Sri Jayewardenepura
Nugegods, Sri Lanka
Email: gishantha@dscs.sjp.ac.lk

S Weerakoon
Department of Mathematics
Faculty of Applied Sciences
University of Sri Jayewardenepura
Nugegods, Sri Lanka
Email: sunethra.weerakoon@gmail.com

***Abstract*—The use of numerical methods to solve univariate nonlinear equations has many drawbacks. We propose a modified firefly algorithm [MOD FA] with a self-tuning ability to solve a given univariate nonlinear equation. Our modification is capable of finding almost all real as well as complex roots of a nonlinear equation within a reasonable interval/range. The modification includes an archive to collect best fireflies and a flag to determine poorly performed iterations. It is also capable of tuning the algorithm-specific parameters while finding the optimum solutions. The self-tuning concept allows the users of our application to use it without any prior knowledge of the algorithm. We validate our approach on examples of some special univariate nonlinear equations with real as well as complex roots. We have also conducted a statistical test: the Wilcockson sign rank test. By conducting a comparison with the genetic algorithm and differential evolution with same modifications [MOD GA] [MOD DE] and with the original firefly algorithm [FA], we confirm the efficiency and the accuracy of our approach.**

I. Introduction

Nonlinear equations play a vital role in most of the real world applications. Many fields, including engineering, mathematics, chemistry, computer science and economics often require applications of univariate as well as systems of nonlinear equations. Solving a univariate nonlinear equation $f(x) = 0$ is finding roots α, such that $f(\alpha) = 0$. Providing solutions for such nonlinear equations is challenging and the common method of solving them is the use of numerical methods; specially the techniques based on Newton's method [1], [2], [3], [4]. Numerical methods often have requirements to be fulfilled to begin with the process of finding approximations. These requirements can be considered as drawbacks of numerical methods that users encounter when using them to solve nonlinear equations. Need of the derivative information and the continuity of the function, evaluation of large matrices, inability to give more than one approximation at a time, depending over the sensitivity of the initial guess and the slow convergence are some of the major drawbacks. Thus, finding better approaches to solve nonlinear equations are still open for research.

In practice, most nonlinear equations are solved using Newton's method or its variants. Weerakoon & Fernando have suggested an improvement to the existing Newton's method in their paper: A Variant of Newton's Method with Accelerated Third-Order Convergence [1]. The 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. Weerakoon & Fernando have modified it to be a trapezium so that the error of the approximation is reduced. The researchers 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 the speed of convergence to the approximation. But this method also contains the aforementioned drawbacks of numerical methods such as the need for derivative calculation.

Apart from the numerical methods, heuristics were also being proposed to solve nonlinear equations. A continuous global optimization heuristic: Continuous Greedy Randomized Adaptive Search Procedure known as C- GRASP has been adopted by Michael J. Hirsch et al to solve a given system of nonlinear equations [5]. They address the problem of finding all the roots of a system of equations assuming that all roots are real. The heuristic does not use the derivative information of the equations of the system. The attempt of the researchers was successful, but they haven't addressed the area of complex roots.

Nature inspired algorithms; being meta-heuristic for most of the time, are now becoming the dominator of the world of optimization algorithms. Compared with other methods, these algorithms have many advantages. Some examples are genetic algorithm, differential evolution, particle swarm optimization, harmony search, firefly algorithm, cuckoo search and others [6], [7], [8], [9], [10], [11]. The meta-heuristic property makes them robust so that these algorithms are capable of touching a variety of problems [12], [13], [14]. Since the applicability of these algorithms is immense, researchers have done some experiments over adopting nature inspired algorithms to solve nonlinear equations.

Use of genetic algorithm to solve nonlinear equations has been carried out in a research done by Nikos E. Mastorakis

[15]. Here, solving a system is viewed as an optimization problem and the objective is to minimize the square root of the function value. The paper addresses the problem of finding all possible roots as well as systems of nonlinear equations, but sufficient information about the results were not provided. In the univariate case, he has tried nonlinear equations with the maximum power of two. The research is important in providing information that GA can be adopted for the desired task.

In several other research, use of GA to solve nonlinear equations are addressed. A hybrid algorithm implemented with genetic algorithms and particle swarm optimization also has been proposed in [16]. Harmony search; A new meta-heuristic algorithm has been also proposed to solve systems of nonlinear equations [17].

Most of these approaches have focused on solving systems of nonlinear equations rather than a single equation. In almost all these researches, they have dealt only with real roots and solving for complex roots is not mentioned. The problem of finding all roots in a reasonable range within a single run is also not addressed. Another important area to consider is the algorithm dependent parameters. When using nature inspired algorithms to solve any optimization problem, these parameters should be tuned properly in order to achieve accurate solutions. The values of these parameters can be different from problem to problem. The most convenient approach is to suggest a way to tune the parameters of the algorithm while optimizing the given problem.

In the present work, we address the problem of solving univariate nonlinear equations for real as well as complex roots using a modified firefly algorithm. Our modification includes an archive and a flag. The modified algorithm is capable of tuning the algorithm dependent parameters while solving a given nonlinear equation.

For a better understanding of our research problem, we have defined it as follows.
Let f be a function s.t. $f : D \to R$ where $D \subset C$,where C is the set of Complex Numbers. The problem is to find all $x \in D$ s.t. $f(x) = 0$, without requiring either the differentiability or the continuity of the function f. Thus we need to find $x \in D$ 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 while optimizing the algorithm dependent parameters.

The remainder of this paper is structured as follows. Section II provides a brief literature related to the firefly algorithm. In Section III, we introduce the modified firefly algorithm and its capabilities. Section IV points out the numerical examples and the algorithm independent parameters used and the results obtained by comparing the algorithm with the other three meta-heuristics, MOD GA, FA and MOD DE. Finally, we draw conclusions briefly in Section V.

II. Firefly Algorithm

Firefly is an eye catching creature in the night sky. The fascinating flash they emit is a way of communicating between them. Its primary purpose is thought to be to attract mates. The main idea of the firefly algorithm is to imitate this flashing behavior of the fireflies to attract them towards brighter ones. Xin-She Yang formulated the original firefly algorithm around 2008 [10]. The algorithm is based on some assumptions about fireflies behavior.

1) Fireflies' attraction to each other is gender independent.
2) Attractiveness is proportional to their brightness, for any two fireflies, the less brighter one is attracted by (and thus moves toward) the brighter one; however, the brightness can decrease as their distance increases; If there is no brighter one than a particular firefly, it moves randomly.
3) The brightness of a firefly is determined by the value of the objective function.

Algorithm 1 : Pseudo code of the basic FA

```
Begin;
Initialize algorithm parameters:
    MaxGen: the maximum number of generations
    γ: the light absorption coefficient
    β0: initial brightness of a firefly
    d: the domain space
Define the objective function f(X),  where  X = (x_1, ..., x_d)^T
Generate the initial population of fireflies, X_i  (i = 1, 2, ..., n)
Determine light intensity of I_i at i^th firefly X_i via f(X_i)
while t < MaxGen do
  for i = 1 : n (all n fireflies) do
    for j = 1 : n (n fireflies) do
      if I_j > I_i then
        Move firefly i towards j by using equation (1);
      end if
      Attractiveness varies with distance r via e^{-γr^2}
      using equation (2);
      Evaluate new solutions and update light intensity;
    end for
  end for
  Rank the fireflies and find the current best;
end while
Post process results and visualization;
```

This modern algorithm grasped the attention of the world of optimization. Because of its success many researches were carried out to check its adaptability to solve a variety of optimization problems [13], [18], [19], [20].

The most general way of formulating the initial population is to define it randomly. In the original firefly algorithm, the

user has to define the algorithm dependent parameters for the selected problem. Objective function should also be defined according to the problem available. After these initial steps, the fireflies in the population start moving towards brighter fireflies according to the following equation.

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

where

$$\beta = \beta_0 e^{(-\gamma r^2)} \quad (2)$$

β_0 is the attraction at $r = 0$

The second term of the $equation(1)$ is due the attraction between $firefly_i$ (x_i) and $firefly_j$ (x_j) and the third term is the randomization term. The value of α is drawn from a Uniform or Gaussian distribution. These two terms are known as information based movement and the random movement. The information based moment exploits the current firefly to build a new firefly. The given firefly's brightness and the distance between another brighter firefly is taken into account when modifying its solution. The random movement creates some random solutions. It is like the mutation operator in genetic algorithms. In this random process the value of the randomization parameter α plays an important role in maximizing performance. The randomness should control properly, otherwise it may lead to poor performance. The best practice is to reduce the randomness gradually. For this, Yang has introduced the following method.

$$\alpha = \alpha_0 \delta \quad where \quad \delta \in [0, 1] \quad (3)$$

Proper parameter selection should be done in order to achieve the expected performance of the algorithm. In the original implementation $\beta_0 = 1$, $\alpha \in [0, 1]$ and $\gamma \in [1, 100]$ are suggested to be applied appropriately. But in our approach, we introduce the framework suggested by Yang et al which allows the algorithm to tune its own parameters [21]. Yang has proved that the original algorithm's performance is relatively high when compared with genetic algorithm and particle swarm optimization algorithm. Popular two dimensional optimization problems were used in his original implementation to prove the idea.

Regardless numerous researches have been carried out using the firefly algorithm to solve various optimization problems, FA has been rarely applied to any kind of nonlinear root finding problem. Fascinated by its performance we are motivated to see the captivating behavior of the firefly algorithm in the process of solving univariate nonlinear equations.

III. Modified Firefly Algorithm

In this section , we will explain our adoption of the firefly algorithm to solve univariate nonlinear equations having real and complex roots.

A. Modified Firefly Algorithm for univariate nonlinear equations

In our approach each firefly in the population represents a possible approximation to a root. Apart from that, a firefly will also carry a possible approximation for the algorithm dependent parameters. The objective function is defined as the absolute value of the function, evaluated at a specific root approximation. To calculate the distance between two fireflies, we used the distance between the two complex numbers that represent those two fireflies.

Algorithm 2 : Pseudo code of the proposed firefly algorithm

```
Begin;
Initialize algorithm independent parameters:
    MaxGen: the maximum number of generations
    β0: initial brightness of a firefly
Initialize range for algorithm dependent parameters:
    γ: the light absorption coefficient
    δ: the randomness reduction factor
Define the objective function f(X)
Generate the initial population of fireflies, X_i (i = 1, 2, ..., n)
Determine light intensity of I_i at i^th firefly X_i via f(X_i)
while t < MaxGen do
  flag=true
  while flag = true & t < MaxGen do
    for i = 1 : n (all n fireflies) do
      for j = 1 : n (n fireflies) do
        if I_j > I_i then
          Move firefly i towards j using equation (1);
        end if
        Attractiveness varies with distance r via e^(-γr^2)
        Evaluate new solutions & update light intensity;
      end for
    end for
    Find the matching fireflies with the eligibility criteria
    abs(f(x)) < 0.001;
    Put them into the archive and replace their positions
    with random fireflies;
    if no fireflies matching with eligibility criteria then
      flag=false
    end if
    if flag=false then
      count= random integer between 0 and n;
      Create random fireflies up to count and replace the
      population;
    end if
  end while
end while
Post-process fireflies in the archive and get the tuned
parameter values
```

The fireflies will move towards brighter fireflies according to the $equation(1)$. After an iteration, better fireflies are noted and they are put into an archive. Their positions

are replaced by random fireflies. The flag will determine the poorly performed iterations and new fireflies will be introduced to the population.

Finally, after a fixed number of iterations, the output will be the root approximations and the suitable algorithm dependent parameter values for the problem.

B. Self tuning property

As we have discussed in section II, the firefly algorithm has several algorithm dependent parameters including α, γ, β and additionally the randomness reduction factor δ. According to the *equation*(2), the value of β depends on three factors; β_0, γ and the distance between two fireflies r. We initiate $\beta_0 = 1$ and the value of r should be calculated during the iterations, so that γ is the only factor that should be controlled.

In *equation*(3), the value of α depends on α_0 and δ. With the experience obtained through experimentations, we initiate $\alpha_0 = 2.3$, so that the consideration should be paid on δ. Here the two parameters to be tuned will become γ and δ. The purpose of using self tuning for our study is to let the users use the algorithm without algorithm specific parameter inputs.

IV. Experimentation

The experiments performed in this study are detailed here. In section A, the numerical examples used for the study are explained. Section B is dedicated to presenting the results obtained form the equations by applying MOD FA, FA, MOD GA and MOD DE. Finally, in Section C, a statistical analysis of the results are shown.

All the work of this research has been carried out on an Intel Core i3 laptop, with 2.30 GHz and a RAM of 2GB. MATLAB has been used as the programming language. 20 univariate nonlinear equations from various representative categories have been used for the study. Each instance, has been run for 100 times and the mode of the number of roots are taken for the comparative study. The accuracy of a root is set to be 10^{-2}.

A. Numerical Examples

Nonlinear functions having real as well as complex roots selected are briefly explained here.

1) The following nonlinear function has 51 real roots within the given interval (adapted from Goldberg and Richardson, 1987 [22]) .

$$y = sin^3(5\pi x) \quad where \quad x \in [-5, 5] \tag{4}$$

2) The Weierstrass function is an example of a pathological real-valued function on the real line. This type of functions possess the property of being continuous everywhere but differentiable nowhere [23]. It is named after its discoverer Karl Weierstrass. Since most of the numerical approaches need to evaluate derivatives, it is difficult to employ a numerical approach to find roots of a Weierstrass function. The following is a Weierstrass function which has 25 real roots within [-20, 20].

$$W(x) = \sum_{i=0}^{n} \left(\frac{1}{2^i}\right) \sin 2^i(x) \quad \text{where n = 3} \tag{5}$$

3) *Equation*(6) represents a popular trigonometric function with discontinuities. It has 13 roots in [-20, 20].

$$y = \tan x \tag{6}$$

4) The following function has 32 real roots scattered in a large interval [-100,100]

$$y = (\sin(x) - 1)(x - 3) \tag{7}$$

5) The following parabolic function has 6 roots. Since the derivative at 0 is equal to zero, the Newton's method cannot be applied to approximate the roots at zero.

$$y = sin(x) + 1 \quad where \quad x \in [-20, 20] \tag{8}$$

Apart from the above mentioned equations we have selected several other test functions having real roots. (see Table I).

	Equation	Interval	Number of roots
1	$y = xsin(1/x) - 0.2e^{-x}$	[-3, 1]	3
2	$y = (x-2)^3$ with multiple roots	[-4, 4]	3
3	$y = x^4 - 2x^2 + 1$ with 2 multiple roots	[-4, 4]	4
4	$y = (x-1)(x+2)(x+1)^2$ with 2 multiple roots	[-3, 3]	4
5	$y = 10x^4 - 270x^2 - 140x + 1200$	[-6, 6]	4
6	$y = (x\sin x)^2 + e^{(x^2\cos x + \sin x)} - 28$	[-10, 10]	6
7	$y = \cos x$	[-20, 20]	12
8	$y = 3e^x - 4cos(x)$	[-40, 2]	14
9	$y = xsin(x) + 0.1x$	[-30, 30]	19
10	$y = \tan(x) - x$	[-40,40]	25
11	$y = cos^3(2x)$	[-40,40]	52
12	$y = \sin(x^2 + 10)$	[-10, 10]	64

TABLE I: Nonlinear functions used to test the algorithm

6) The suggested method is capable of finding complex roots of a univariate nonlinear equation. The following polynomials were taken as test functions for the study (see Table II).

	Equation	Range	Number of roots
1	$y = x^2 + 1$	[-2, 2]X[-2, 2]	2 (0 real)
2	$y = x^2 + 2x + 10$	[-2, 4]X[-2, 4]	2 (0 real)
3	$y = x^3 + 2x^2 + 3x + 4$	[-2.5, 1]X[-2.5, 1]	3 (1 real)
4	$y = x^4 - 2x^2 - 3x - 2$	[-1, 2.5]X[-1, 2.5]	4 (2 real)
5	$y = x^5 - 3x^4 + 3x^3 - 2x^2 - 5$	[-1, 3]X[-1, 3]	5 (1 real)
6	$y = x^7 - x^6 + 2x^5 - 3x^4 + 3x^3 - 2x^2 - 5$	[-1, 2]X[-1, 2]	7 (1 real)
7	$y = x^8 - 2x^7 + 3x^6 + 4x^5 + 5x^4 + 6x^3 + 7x^2 + 8x + 9$	[-2, 2]X[-2, 2]	8 (0 real)
8	$y = x^{12} - 6x^{11} + x^{10} - 5$	[-1, 6]X[-1, 6]	12 (2 real)
9	$y = x^{13} - 2x^{12} + 1$	[-1, 2]X[-1, 2]	13 (3 real)

TABLE II: Testing the algorithm with nonlinear functions returning complex roots

Range 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. 1.

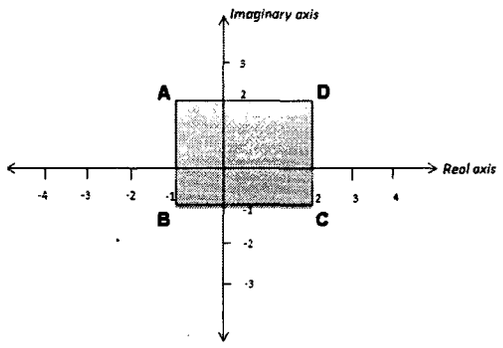

Fig. 1: [-1, 2] X [-1, 2], an example region used for the root finding

B. Results obtained by the modified firefly algorithm and the other meta-heuristics

As mentioned in the beginning of this section, an experimentation has been performed to prove the performance of the proposed modified firefly algorithm. The most important feature of this algorithm is the self tuning ability. The algorithm mainly tune γ and δ parameters. α and β values change accordingly. We implemented the 4 algorithms and for all, the initial population were generated randomly. Each algorithm finishes one execution after 100 iterations. 100 such runs were carried out to obtain the results.

$y = sin^3(5\pi x)$					
-5.000	-4.8001	-4.6001	-4.4002	-4.2033	-4.0052
-3.8001	-3.6027	-3.4029	-3.2034	-3.0001	-2.8029
-2.6006	-2.4000	-2.2021	-2.0053	-1.8007	-1.6005
-1.4001	-1.2003	-1.0022	-0.8018	-0.6016	-0.4023
-0.2017	0.0005	0.2006	0.4014	0.6031	0.8020
1.0019	1.2002	1.4010	1.6016	1.8010	2.0004
2.2010	2.4022	2.6007	2.8046	3.0003	3.2019
3.4000	3.6001	3.8002	4.000	4.2009	4.4007
4.6006	4.8012	5.0000			

TABLE III: 51 roots given by the modified firefly algorithm for $y = sin^3(5\pi x)$

y= tan(x)				
-18.8509	-15.71089	-12.5747	-9.4323	-6.2914
-3.1500	0.0015	3.1334	6.2778	9.4164
12.5577	15.7009	18.84381		

TABLE IV: 13 roots given by the modified firefly algorithm for $y = tan(x)$

The modified algorithm tunes its algorithm dependent parameters while optimizing the problem of solving a nonlinear equation. Table V illustrates the average parameter values obtained by the MOD FA for the main test functions.

To emphasize the parameter tuning process used in our study, we have obtained the average γ and δ values for each iteration for the $equation(4)$ for 5 random instances [Fig 2].

It is clear that after some number of iterations, the parameter values are stabilizing around some optimum value.

	Equation	Optimized γ value	Optimized δ value
1	$y = sin^3(5\pi x)$	90.5251	0.8978
2	$y = sin(x) + (1/2)sin(2x) + (1/4)sin(4 * x) + (1/8)sin(8 * x)$	79.7420	0.9077
3	$y = tan(x)$	91.3400	0.9636
4	$y = (sin(x) - 1)(x - 3)$	80.1758	0.9077
5	$y = sin(x) + 1$	96.2917	0.8277
6	$y = x^2 + 1$	77.1285	1
7	$y = x^2 + 2x + 10$	77.2168	0.8399
8	$y = x^3 + 2x^2 + 3x + 4$	79.8629	0.8668
9	$y = x^4 - 2x^2 - 3x - 2$	80.3693	0.8000
10	$y = x^5 - 3x^4 + 3x^3 - 2x^2 - 5$	89.5486	0.8235
11	$y = x^7 - x^6 + 2x^5 - 3x^4 + 3x^3 - 2x^2 - 5$	94.5166	0.8326
12	$y = x^8 + 2x^7 + 3x^6 + 4x^5 + 5x^4 + 6x^3 + 7x^2 + 8x + 9$	80.1703	0.8795
13	$y = x^{12} - 6x^{11} + x^{10} - 5$	86.8613	0.9067
14	$y = x^13 - 2x^12 + 1$	84.4947	0.8638

TABLE V: Algorithm dependent parameter values obtained for different equations by MOD FA

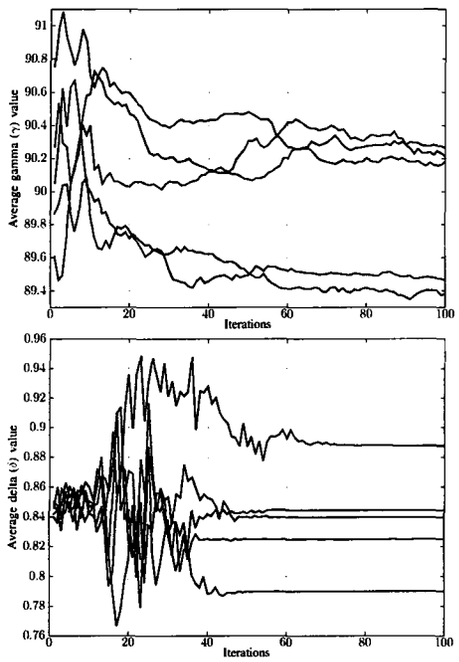

Fig. 2: Variation of the average gamma values and average delta values over iterations (Graphs were drown at randomly selected 5 runs)

To facilitate the usability of this work, the parametrization used for the three approaches are summarized in Table VI.

Table VII shows the results obtained by running the algorithm 100 times to solve 14 nonlinear equations. Results are formatted as; average number of roots found, (maximum number of roots found / total roots)*100%.

So 51, (100%) means the average number of roots found within 100 runs is 51 and the (maximum number of roots found in a run / total roots)*100 is 100%.

MOD FA		FA		MOD GA		MOD DE	
Parameter	Value	Parameter	Value	Parameter	Value	Parameter	Value
Pop Size	600	Pop Size	600	Pop Size	600	Pop Size	600
α	2.3	α	2.3	Crossover	Arithmetic crossover (0.95 probability)	Differential weight	0.2
β_0	1	β_0	1	Mutation	Subtraction operator (0.005 probability)	Crossover probability	0.95
β	Tuned by MOD FA	β	Tuned by FA				
γ	Tuned by MOD FA	γ	Tuned by FA				
δ	Tuned by MOD FA	δ	Tuned by FA				
Archiving Condition	$< 10^{-2}$	δ	Tuned by FA				

TABLE VI: Parametrization of the Mod FA, FA, MOD GA and MOD DE for the nonlinear equations

	Equation	# of roots	MOD FA	FA	MOD GA	MOD DE
1	$y = sin^3(5\pi x)$	51	51.(100%)	17.(37%)	50.(100%)	51.(100%)
2	$y = \sum_{i=0}^{3} \left(\frac{1}{2^i}\right) \sin 2^i(x)$	25	25.(100%)	12.(52%)	25.(100%)	20.(92%)
3	$y = tan(x)$	13	13.(100%)	6.(53%)	13.(100%)	12.(100%)
4	$y = (sin(x) - 1)(x - 3)$	32	30.(100%)	3.(9%)	26.(88%)	26.(84%)
5	$y = sin(x) + 1$	6	6.(100%)	6.(100%)	6.(100%)	6.(100%)
6	$y = x^2 + 1$	2	2.(100%)	2.(100%)	2.(100%)	2.(100%)
7	$y = x^2 + 2x + 10$	2	2.(100%)	2.(100%)	2.(100%)	2.(100%)
8	$y = x^3+2x^2+3x+4$	3	3.(100%)	3.(100%)	3.(100%)	1.(33%)
9	$y = x^4-2x^2-3x-2$	4	4.(100%)	4.(100%)	3.(100%)	2.(50%)
10	$y = x^5 - 3x^4 + 3x^3 - 2x^2 - 5$	5	5.(100%)	5.(100%)	3.(60%)	1.(20%)
11	$y = x^7-x^6+2x^5-3x^4+3x^3-2x^2-5$	7	7.(100%)	7.(100%)	1.(14%)	1.(14%)
12	$y = \sum_{i=1}^{9} i * x^{9-i}$	8	8.(100%)	6.(100%)	4.(50%)	2.(25%)
13	$y = x^{12} - 6x^{11} + x^{10} - 5$	12	12.(100%)	9.(83%)	3.(25%)	1.(16%)
14	$y = x^{13} - 2x^{12} + 1$	13	13.(100%)	12.(92%)	4.(31%)	2.(15%)

TABLE VII: Performance of the algorithms for real and complex roots over 100 runs

According to the results in Table VII, we can conclude that MOD FA outperforms the results obtained by FA, GA and DE, obtaining almost all real as well as complex roots of the tested 14 equations. Here the conclusions are taken from the ground level view so that the statistical analysis has been used to conclude the idea firmly. The guidelines provided by Derrac et al were followed to perform this statistical analysis [24].

C. Statistical Analysis

In addition we have conducted a statistical analysis to compare the performance of the modified firefly algorithm with other algorithms. Wilcoxon signed-rank test is a nonparametric test that can be applied to two related samples [25]. It can be used as an alternative to the paired Student's t-test or the t-test for dependent samples when the population cannot be assumed to be normally distributed [26]. We used three of two related samples; Sample results of Mod FA and FA, Mod FA and MOD GA and MOD FA and MOD DE. 20 different nonlinear equations were used for the statistical analysis. We execute each algorithm 100 times and get the mode number of roots for each equation to apply the test. It has been conducted on Minitab statistical software to test the following hypothesis.

H_0 : There is no difference between two algorithms
H_1 : There is a difference between two algorithms

	Equation	# of roots	MOD FA	FA	MOD GA	MOD DE
1	$y = sin^3(5\pi x)$	51	51	14	51	51
2	$y = tan(x)$	13	13	1	13	12
3	$y = \sum_{i=0}^{3} \left(\frac{1}{2^i}\right) \sin 2^i(x)$	25	25	7	25	20
4	$y = (sin(x) - 1)(x - 3)$	32	32	2	32	28
5	$y = sin(x) + 1$	6	6	1	6	6
6	$y = cos(x)$	12	12	5	12	12
7	$y = (x - 1)(x + 2)(x + 1)^2$	4	4	4	4	4
8	$y = xsin(x) + 0.1x$	19	19	2	16	5
9	$y = sin(x^2 + 10)$	64	64	3	57	36
10	$y = cos(2x)^3$	52	52	5	50	52
11	$y = 3 * exp(x) - 4cos(x)$	14	14	8	14	8
12	$y = x^2 + 1$	2	2	2	2	2
13	$y = x^2 + 2x + 10$	2	2	2	2	2
14	$y = x^3 + 2x^2 + 3x + 4$	3	3	3	1	1
15	$y = x^4 - 2x^2 - 3x - 2$	4	4	4	4	2
16	$y = x^5 - 3 * x^4 + 3 * x^3 - 2 * x^2 - 5$	5	5	5	3	1
17	$y = x^7 - x^6 + 2x^5 - 3x^4 + 3x^3 - 2x^5 - 5$	7	7	7	1	1
18	$y = \sum_{i=1}^{9} i * x^{9-i}$	8	8	8	2	2
19	$y = x^{12} - 6x^{11} + x^{10} - 5$	12	9	12	3	1
20	$y = x^{13} - 2x^{12} + 1$	13	13	12	4	2

TABLE VIII: Results of the proposed MOD FA with other three algorithms for the statistical analysis

The test statistic is in the following form.

$$W = \sum_{i=0}^{N} [sgn(x_{2,i} - x_{1,i}).R_i] \quad (9)$$

where:
N : sample size
R_i : Rank of the ith pair

Table VIII shows the equations and the number of roots repeated most (Mode), obtained for each algorithm in participating the Wilcoxon signed-rank test. The results demonstrate that MOD FA performs well for both real and complex root situations where MOD GA and MOD DE are successful in working on real roots. FA on the other hand lucky enough only for the complex root situation however its performance is very poor when it handles real roots.

The Wilcoxon signed-rank test has three "assumptions". The dependent variable should be measured at a continuous level or ordinal level, the independent variable should consist of two categorical, "related groups" and the distribution of the differences between the two related groups are symmetrical in shape happen to be these 3 assumptions. Here we have three of two related groups; Number of roots obtained by MOD FA and FA, MOD FA and MOD GA and MOD FA and MOD DE.

WILCOXON SIGNED RANK TEST: DIFFERENCE BETWEEN MOD FA & FA

```
Test of median = 0.000000 versus median not =
0.000000
                    N   N      Wilcoxon  P       Estimated
                        for    Statistic         Median
                        Test
Difference          20  12     76.0      0.004   8.500
between MOD FA
& FA
```

TABLE IX: Minitab output for the hypothesis tested over MOD FA and FA

Table IX shows the Minitab output for the hypothesis tested over MOD FA and FA. We have conducted the test at 0.05 significance level (α). The obtained P value (0.004) is less than α, ($0.004 < 0.05$) so we reject H_0; That is, we accept that there is a difference between MOD FA and FA. Then we have calculated the median values for both algorithms. Results are shown in the table X.

WILCOXON SIGNED RANK CI: MOD FA, FA

```
                                          Confidence
                                          Interval
          N   Estimated   Achieved        Lower  Upper
              Median      Confidence
MOD FA    20  12.8        95.0            7.5    27.5
FA        20  5.00        95.0            3.00   7.00
```

TABLE X: Median values obtained by MOD FA and FA for 20 different nonlinear equations

The estimated median is higher for MOD FA, so that we can conclude that the performance of the MOD FA is better than of FA.

Same results obtained for the algorithms MOD FA and MOD GA are presented in table XI and XII and the same can be obtained for MOD FA and MOD DE.

WILCOXON SIGNED RANK TEST: DIFFERENCE BETWEEN MOD FA & MOD GA

```
Test of median = 0.000000 versus median not =
0.000000
                    N   N      Wilcoxon  P       Estimated
                        for    Statistic         Median
                        Test
Difference          20  9      45.0      0.009   1.500
between MOD FA
& MOD GA
```

TABLE XI: Minitab output for the hypothesis tested over MOD FA and MOD GA

WILCOXON SIGNED RANK CI: MOD FA, MOD GA

```
                                          Confidence
                                          Interval
          N   Estimated   Achieved        Lower  Upper
              Median      Confidence
MOD FA    20  12.8        95.0            7.5    27.5
MOD GA    20  9.8         95.0            4.0    26.0
```

TABLE XII: Median values obtained by MOD FA and MOD GA for 20 different nonlinear equations

Here also the P value is less than the significance level ($0.009 < 0.05$), so that we reject the null hypothesis and the estimated median value of MOD FA is higher than MOD GA. The analysis shows that MOD FA is so far the best performer in obtaining almost all roots of a given nonlinear equation.

As a final supposition, we can say that using similar nonlinear functions, the proposed MOD FA outperforms the other alternative algorithms with similar and acceptable run times and showing a better robustness and convergence. The improvements are significant for most cases. For this reason, we can say that the presented MOD FA is a promising approach to solve univariate nonlinear equations with real as well as complex roots, meeting, in this respect, the main objective of this study.

V. CONCLUSIONS AND FURTHER WORK

In this work we have presented a modified firefly algorithm to solve univariate nonlinear equations. Our approach uses an archive and a flag to determine real as well as complex roots impressively. The algorithm does not need the differentiability and the continuity of the function and does not depend on the initial guess. In addition, we have used the framework proposed by Xin-she Yang and the fellows to tune the algorithm specific parameters. It can be stated as the most important feature of this algorithm since it optimize the algorithm dependent parameters while optimizing the problem itself. This is advantageous for the users who do not know the behavior of the firefly algorithm.

In order to prove the ability of our algorithm we have compared its performance on 30 nonlinear functions with the original firefly algorithm and the modified genetic algorithm. A statistical test has been conducted with the obtained results; Wilcoxon signed-rank test. Overall, the MOD FA performs well in obtaining real as well as complex roots than the other two algorithms.

It is important to highlight that the main goal of this study is not to find an optimal solution to a given nonlinear function. Hence we kept the accuracy of a solution to be 10^{-2}. We have already shown in [27], the MOD FA is capable of reaching for more than this accuracy, but the principal objective of this research is proving that the FA

can be easily adapted to solve nonlinear equations having complex roots, without concerning the differentiability or the continuity of the function and that the MOD FA is a promising approximation method for solving univariate nonlinear equations over large intervals/ regions as well.

In this work, we have paid our attention on two things, solving a univariate nonlinear equation to find all real as well as complex roots simultaneously and tune the algorithm dependent parameters using Yang's framework. To optimize parameters there could be some other methods. For this reason, as a future work, we would like to search on some additional parameter optimization techniques which can be adopted to tune algorithm dependent parameters in nature inspired algorithms. Additionally, we know there exist a large number of nature inspired algorithms; we have used only two algorithms to compare the capability of our algorithm. But we would prefer wider experimentations on other meta heuristics as well since it can be an advantage for the future of root approximations of univariate as well as systems of nonlinear equations. Inspired by the results obtained from our experiment, we are planning to expand the study on solving systems of nonlinear equations.

ACKNOWLEDGMENT

The authors would like to thank Dr. Xin-She Yang for his valuable suggestions and explanations on implementing the self tuning framework and Ms. W.J. Polegoda, lecturer at the Faculty of Animal Science & Export Agriculture, Uwa Wellassa University, Sri Lanka for the suggestions given on the statistical analysis.

REFERENCES

[1] S. Weerakoon and T. G. I. Fernando, "A variant of newton's method with accelerated third-order convergence," *Applied Mathematics Letters*, vol. 13, no. 8, pp. 87 – 93, 2000.

[2] T. G. I. Fernando, "Improved newtons method for solving nonlinear equations," Master's thesis, University of Sri Jayewardenepura, University of Sri Jayewardenepura. Sri Lanka, 1998.

[3] C. Chun, "Iterative methods improving newton's method by the decomposition method," *Computers & Mathematics with Applications*, vol. 50, no. 1012, pp. 1559 – 1568, 2005.

[4] L. Fang, L. Sun, and G. He, "An efficient Newton-type method with fifth-order convergence for solving nonlinear equations." *Computational & Applied Mathematics*, vol. 27, pp. 269 – 274, 00 2008.

[5] M. J. Hirsch, P. M. Pardalos, and M. G. Resende, "Solving systems of nonlinear equations with continuous {GRASP}," *Nonlinear Analysis: Real World Applications*, vol. 10, no. 4, pp. 2000 – 2006, 2009.

[6] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1st ed., 1989.

[7] R. Storn and K. Price, "Differential evolution a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[8] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Neural Networks, 1995. Proceedings., IEEE International Conference on In Neural Networks*, vol. 4, pp. 1942–1948, 1995.

[9] K. S. Lee and Z. W. Geem, "A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice," *Computer Methods in Applied Mechanics and Engineering*, vol. 194, no. 3638, pp. 3902 – 3933, 2005.

[10] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *Proceedings of the 5th International Conference on Stochastic Algorithms: Foundations and Applications*, SAGA'09, (Berlin, Heidelberg), pp. 169–178, Springer-Verlag, 2009.

[11] X.-S. Yang and S. Deb, "Cuckoo search via levy flights," in *Nature Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pp. 210–214, Dec 2009.

[12] C. Coello Coello, M. Rudnick, and A. Christiansen, "Using genetic algorithms for optimal design of trusses," in *Tools with Artificial Intelligence, 1994. Proceedings., Sixth International Conference on*, pp. 88–94, Nov 1994.

[13] G. Jati and Suyanto, "Evolutionary discrete firefly algorithm for travelling salesman problem," in *Adaptive and Intelligent Systems* (A. Bouchachia, ed.), vol. 6943 of *Lecture Notes in Computer Science*, pp. 393–403, Springer Berlin Heidelberg, 2011.

[14] G. W. Greenwood, "Using differential evolution for a subclass of graph theory problems," *Trans. Evol. Comp*, vol. 13, pp. 1190–1192, Oct. 2009.

[15] N. E. Mastorakis, "Solving non-linear equations via genetic algorithms," in *Proceedings of the 6th WSEAS International Conference on Evolutionary Computing*, EC'05, (Stevens Point, Wisconsin, USA), pp. 24–28, World Scientific and Engineering Academy and Society (WSEAS), 2005.

[16] A. Biswas and S. Dasgupta, "Finding solution of simultaneous non linear equations using ga-pso hybrid algorithm," in *Proceedings of the Indian Conference on Trends in Modern Engineering Systems (ICONTIMES 2008)*.

[17] G. Ramadas and E. Fernandes, "Solving systems of nonlinear equations by harmony search," in *Proceedings of the th International Conference on Computational and Mathematical Methods in Science and Engineering CMMSE 2013*.

[18] X.-S. Yang, S. S. S. Hosseini, and A. H. Gandomi, "Firefly algorithm for solving non-convex economic dispatch problems with valve loading effect," *Applied Soft Computing*, vol. 12, no. 3, pp. 1180 – 1186, 2012.

[19] M. Sayadi, R. Ramezanian, and N. Ghaffari-Nasab, "A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems," *International Journal of Industrial Engineering Computations*, vol. 1, no. 1, pp. 1 – 10, 2010.

[20] M.-H. Horng and T.-W. Jiang, "The codebook design of image vector quantization based on the firefly algorithm," in *Computational Collective Intelligence. Technologies and Applications* (J.-S. Pan, S.-M. Chen, and N. Nguyen, eds.), vol. 6423 of *Lecture Notes in Computer Science*, pp. 438–447, Springer Berlin Heidelberg, 2010.

[21] X.-S. Yang, S. Deb, M. Loomes, and M. Karamanoglu, "A framework for self-tuning optimization algorithm," *Neural Computing and Applications*, vol. 23, no. 7-8, pp. 2051–2057, 2013.

[22] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application*, (Hillsdale, NJ, USA), pp. 41–49, L. Erlbaum Associates Inc., 1987.

[23] L. Duncan, *The Weierstrass Function*. University of Maine, 1901.

[24] J. Derrac, S. Garca, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3 – 18, 2011.

[25] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[26] M. Hollander, D. A Wolfe, and E. Chicken, "Wiley series in probability and statistics," *Nonparametric Statistical Methods, Third Edition*, pp. 821–828, 1999.

[27] M. K. A. Ariyaratne, T. G. I. Fernando, and S. Weerakoon, "A modified firefly algorithm to solve univariate nonlinear equations with complex roots," in *Proceedings of the International Conference on Advances in ICT for Emerging Regions (ICTer)*, 2015.