A self-tuning algorithm to approximate roots of systems of nonlinear equations based on the firefly algorithm

M.K.A. Ariyaratne* and T.G.I. Fernando

Department of Computer Science,
Faculty of Applied Sciences,
University of Sri Jayewardenepura, Sri Lanka
Email: anuradhaariyaratne@gmail.com
Email: gishantha@dscs.sjp.ac.lk
*Corresponding author

Sunethra Weerakoon

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

Abstract: The most acquainted methods to find root approximations of nonlinear equations and systems; numerical methods possess disadvantages such as necessity of acceptable initial guesses and the differentiability of the functions. Even having such qualities, for some univariate nonlinear equations and systems, approximations of roots is not possible with numerical methods. Research are geared towards finding alternate approaches, which are successful where numerical methods fail. One of the most disadvantageous properties in such approaches is inability of finding more than one approximation at a time. On the other hand these methods are incorporated with algorithm specific parameters which should be set properly in order to achieve good results. We present a modified firefly algorithm handling the problem as an optimisation problem, which is capable of giving multiple root approximations simultaneously within a reasonable state space while tuning the parameters of the proposed algorithm by itself, using a self-tuning framework. Differentiability and the continuity of the functions and the close initial guesses are needless to solve nonlinear systems using the proposed approach. Benchmark systems found in the literature were used to test the new algorithm. The root approximations and the tuned parameters obtained along with the statistical analysis illustrate the viability of the method.

Keywords: nature-inspired algorithms; firefly algorithm; systems of nonlinear equations; self-tuning framework; parameter optimisation; archive; root approximations.

Reference to this paper should be made as follows: Ariyaratne, M.K.A., Fernando, T.G.I. and Weerakoon, S. (2020) 'A self-tuning algorithm to approximate roots of systems of nonlinear equations based on the firefly algorithm', *Int. J. Swarm Intelligence*, Vol. 5, No. 1, pp.60–96.

Biographical notes: M.K.A. Ariyaratne completed her BSc in Computer Science from the University of Sri Jayewardnepura, Sri Lanka in 2012. Currently, she is pursuing her PhD at the Department of Computer Science, Faculty of Graduate Studies, University of Sri Jayewardnepura, Sri Lanka. Her major areas of interests are nature inspired optimisation techniques and machine learning algorithms.

T.G.I. Fernando is a Senior Lecturer in Computer Science in the Department of Computer Science, University of Sri Jayewardenepura, Sri Lanka. He completed his BSc (Special) degree in Mathematics in 1993 and MSc in Industrial Mathematics in 1998 at the University of Sri Jayewardenepura, Sri Lanka. Subsequently, in 2002, he completed his second MSc in Computer Science at the Asian Institute of Technology (AIT), Thailand. Finally, he completed his PhD in Intelligent Systems at the Brunel University (UK) in 2009. He has received several awards/scholarships for his research achievements and education. His research interests are mainly in intelligent systems, evolutionary computing, swarm intelligence, neural networks (including deep neural networks), machine learning, multi-objective combinatorial optimisation and root finding of nonlinear equations.

Sunethra Weerakoon retired in 2017 as a Senior Professor of Mathematics from the University of Sri Jayawardenepura, Sri Lanka where she served since 1976. She finished her Bachelor's in Mathematics at the University of Peradeniya, Sri Lanka (1972/76). She pursued her Master's and Doctoral studies as a Fulbright Scholar at the Pennsylvania State University, University Park, USA (1979/84). She was a Visiting Teaching Fellow at the Curtin University of Technology, Western Australia (1991/93). She was also a Visiting Professor at the Departments of Mathematics, Cornell University (2004/05) and Texas A&M University, College Park (2005/06). She has written a few books: *Elementary Numerical Methods*, *Numerical Solution of Ordinary Differential Equations and Partial Differential Equations* and edited a few others. Her research interests are in numerical analysis and partial differential equations. She has received a number of awards for her academic achievements and research.

1 Introduction

There are many practical scenarios in areas such as mathematics, engineering, computer science, physics, chemistry and finance where we need to find the values of set of variables that satisfy a set of given constraints. When there are $\boldsymbol{n}$ variables and $\boldsymbol{n}$ equality constraints, the problem is one of solving a system of nonlinear equations. In general, a system of nonlinear equations will have no solution, a unique solution or many solutions. For the simple situations like finding approximations of a system of quadratics, analytical methods can be used. When the number of variables becomes high,

finding more than one solution for nonlinear systems has become critical and hence it is categorised to be an NP hard problem (Grosan and Abraham, 2007). One of the most practiced methods to solve nonlinear systems is use of numerical methods. Most of such numerical methods are based on the classical Newton's method (Remani, 2013; Rheinboldt, 1998; Gragg and Stewart, 1976; Ortega and Rheinboldt, 2000). But the inherited drawbacks in numerical methods such as necessity of proper initial guesses, necessity of differentiability and continuity of the functions, evaluation of large matrices and inability of handling multiple root situations have made them less usable and need of finding new methods to solve nonlinear systems.

Recently, researchers have been on attention of the problem of approximating roots of nonlinear systems. There can be found reasonable amount of research carried out on this area using different types of methods. Still there are efforts made on improving the existing numerical methods to solve the problem. As mentioned earlier, Newton's method has been modified in several research for the purpose of approximation of roots of nonlinear systems.

In Hueso et al. (2009), they have modified the Newton's method to solve nonlinear systems with the help of a singular Jacobian rather than using a non-singular Jacobian matrix. To overcome the difficulty of the use of non-singular Jacobian, they have presented a variant where the Jacobian can be singular. The accuracy of the approximations is very high which is 10^{-12}. The variant is better in terms of complexity when compared with the classical Newton's method, but still carry the requirement of the calculations such as the need of derivatives and continuity and proper initial guesses. The modification is capable of providing one root approximation at a time.

Sharma et al. (2016) have done a study on modified Newton-Traub composition with increasing order of convergence for solving systems of nonlinear equations. As mentioned, the aim is to improve the order of convergence of the classical Newton's method. Their analysis of convergence has shown that the method has sixth order of convergence. Still the derivative information and the evaluation of large matrices are present in the method.

A recent study has been carried out to seek an alternative way to solve systems of nonlinear equations when the classical Newton's method fails (Ramos and Monteiro, 2017). They attempt to obtain the approximations of a nonlinear system of equations by finding solutions of an associated system which can be built through the theory underlying the Newton's method. They claim that the use of an associated system along with the classical Newton's method is more efficient than applying the Newton's method alone on the original system. Examples are given to illustrate the performance of the new method. Accuracy of the obtained approximations is high but here also only a single approximation was given at a time.

Many studies on root approximations have done to improve quality of the solutions and enhance the speed of the classical Newton's method. One such variant to the Newton's method was initially proposed to solve univariate nonlinear equations with the hope to accelerate the convergence rate of the classical Newton's method (Weerakoon and Fernando, 2000). In a recent study, they have extended the method for the systems of nonlinear equations (Nishani et al., 2018). The new modification was tested over several benchmark systems up to 20 variables. The proposed method achieves the aim of improving the convergence rate but still carry all the drawbacks prevailing in the classical Newton's method.

It is clear that even for the systems of nonlinear equations, the use of numerical methods is not sufficient to completely solve the problem. Major drawbacks are the use of derivatives and large matrices. Literature carries several alternative methods including heuristics and meta-heuristics to find a successful solution to the problem.

GRASP is an iterative randomised sampling technique in which each iteration provides a solution to the problem at hand (Feo and Resende, 1995). GRASP stands for greedy randomised adaptive search procedures. Each GRASP iteration consists of two phases, a construction phase and a local search phase. The first phase builds an initial solution via an adaptive randomised greedy function and the second phase provides a local search procedure to the constructed solution to find an improvement. This GRASP heuristic was modified for continuous global optimisation problems, which is named as C-GRASP (Hirsch et al., 2007). In a study this heuristic C-GRASP has been used to solve systems of nonlinear equations (Hirsch et al., 2009). One big advantage of the heuristic is that it uses no derivative information of the functions in the system. The study has also focused on finding all the roots of a system of equations. For a particular nonlinear system, they have assumed all roots to be real and construct a corresponding optimisation problem, which then solve for multiple times, using C-GRASP. The interested nonlinear system is converted to a single objective optimisation problem where the initial objective is to minimise the square sum of each function value. To achieve finding all roots, upon finding a root, the objective function is slightly modified. It is known as the adaptive modification and it has built a penalty region around the solutions that are already been found. The accuracy of a root approximation is set to be 10^{-7}. The new method is tested with two variable nonlinear systems where the maximum number of roots in a system is 13. The proposed method has successfully worked with the given examples. However the study has focused more on expressing the method rather than validating it with more complex examples.

Neural networks have also been used in several studies in solving nonlinear systems. An artificial neural network (ANN) is introduced with a simple gradient descent rule by Hui and Zhe-zhao (2008). Difference between zero and a function value in a system is taken as the error given by that equation and the objective is to minimise the half of squared sum of such errors of the system. To minimise it, they have used simple gradient descent rule. Two variable nonlinear systems were used to test the system and the proposed study is capable of giving one root approximation at a time. The proposed approach is not tested with high dimensional systems to prove the capacity of the method. However if the system has more variables, the method is engaged with large calculations including derivations of the functions, because of the learning rule involved to minimise the error.

Another study presents a method to solve nonlinear equation systems by using Hopfield neural network is a form of recurrent artificial neural network (Luo and Han, 1995). The explanation has focused more on expressing the used method rather than the validity of the method. Because of the use of ANN, large calculations and need of derivative information are present. Although the method is proposed for systems of nonlinear equations, only one univariate nonlinear equation is solved. Therefore it can be said that the effectiveness of the method is not expressed properly.

Table 1 A summary of available research on solving systems of nonlinear equations

Title of the paper	*Method*	*Need of initial guesses*	*Provide multiple solutions simultaneously*	*Maximum dimension of a problem*	*Accuracy*
Modified Newton's method for systems of nonlinear equations with singular Jacobian (Hueso et al., 2009)	A modification of the Newton's method	Yes	No	3	10^{-3}
Solving non-linear equations via genetic algorithms (Mastorakis, 2005)	Genetic algorithms	No	No	2	10^{-4}
A new approach for solving nonlinear equations systems (Grosan and Abraham, 2008b)	An evolutionary computation technique + Pareto dominance relationship	No	Yes	20	Depends on the problem.
Solving systems of nonlinear equations by harmony search (Ramadas and Fernandes, 2013)	Harmony search + differential evolution	No	No	3	10^{-2}
Particle swarm algorithm for solving systems of nonlinear equations (Jaberipour et al., 2011)	Modified particle swarm optimisation algorithm	No	No	10	Depends on the problem
Solving nonlinear equations systems with a new approach based on invasive, weed optimization algorithm and clustering (Pourjafari and Mojallali, 2012)	Invasive weed optimisation + clustering	No	No	3	Not mentioned
Multi-population parallel imperialist competitive algorithm for Solving systems of nonlinear equations (Majd et al., 2016)	Parallel imperialist competitive algorithm	No	No	8	Depends on the problem. Highest is 10^{-15}
Locating multiple optimal solutions of nonlinear equation systems based on multiobjective optimization (Song et al., 2015)	Biobjective transformation technique + genetic algorithm	No	Yes	20	Not mentioned

Table 1 A summary of available research on solving systems of nonlinear equations (continued)

Title of the paper	*Method*	*Need of initial guesses*	*Provide multiple solutions simultaneously*	*Maximum dimension of a problem*	*Accuracy*
A weighted biobjective transformation technique for locating multiple optimal solutions of nonlinear equation systems (Gong et al., 2017)	Weighted biobjective transformation technique + adaptive multiobjective differential evolution	No	Yes	20	Not mentioned
Pattern search firefly algorithm for solving systems of nonlinear equations (Wang and Zhou, 2014)	Firefly algorithm + pattern search strategy	No	No	6	10^{-4}
An improved cuckoo optimization algorithm (Abdollahi et al., 2016)	Cuckoo optimisation algorithm	No	No	10	Depends on the problem.
Weerakoon-Fernando method with accelerated third-order convergence for systems of nonlinear equations (Nishani et al., 2018)	A variant Newton's method	Yes	No	10	10^{-13}
A new approach based on the Newton's method to solve systems of nonlinear equations (Ramos and Monteiro, 2017)	A variant Newton's method	Yes	No	3	Depends on the problem.
A simple and efficient method with high order convergence for solving systems of nonlinear equations (Xiao and Yin, 2015)	A variant Newton's method	Yes	No	101	Depends on the problem.

Same Hopfield neural network structure has been adopted in another research with modifications to address the same problem (Mishra and Kalra, 2007). With the Hopfield neural network, a new energy function is formulated. This energy function is used to calculate the weights and bias values for the network. Desired solutions are obtained when the network energy is minimum. Sigmoid function is used as the activation function. The method is capable of giving one root approximation at a time. However the calculations involve with derivatives and matrices as in earlier approaches and hence the functions in the system should be differentiable. Therefore employing the method for large systems may be complicated.

An attempt to solve nonlinear algebraic systems with polynomial equations using neural networks is presented in a study (Margaris and Adamopoulos, 2007). Hopfield network is used in this method as well but this time no energy function is to be minimised. In this approach, since no energy function is used roots are not given as network outputs but as the weights of the synapses that join the input neuron with the neurons of the first hidden layer. These weights are updated with the back propagation learning rule and after the termination of the training they keep the values of the weights which are one of the roots of the nonlinear algebraic system. The proposed method is applied to general polynomial systems with order two and three. Since hidden layer weights represents the dimension of the problem, for higher dimensional systems the hidden layer terms can be increased, which can be a big disadvantage of the approach and for an ANN, hidden layer units play an important role and the ANN will not function as desired when some number of hidden units exceeds. This method is also giving one root approximation at a time.

Linear Hopfield network with multilayer perceptron (MLP) has been used by Mathia and Saeks (1995) to form a recurrent neural network to solve systems of nonlinear equations. Data collected from a nonlinear system are approximated by an MLP. The MLP is then inverted, i.e. the approximated equation is solved, using recurrent neural networks. The ability of MLP as a universal function approximation method has been used here. For the multilayer perceptron, sigmoid activation function is used and for the Hopfield network, nonlinear activation functions are replaced with linear activation functions. The Newton's method has used to provide a pattern for designing the recurrent neural network. Therefore, this approach consists of the same drawbacks that of Newton's method, such as need of differentiability and the existence of inverse of functions.

Meta-heuristics as optimisation algorithms have also proposed to solve system of nonlinear equations. In such cases, the problem is solved as a single objective or multi objective optimisation problem.

Mastorakis (2005) has done a study on solving nonlinear systems via genetic algorithms (GA). The problem is considered as a single objective optimisation problem. Only a single example is used to test the method which will not illustrate the quality of the approach sufficiently. But here the advantage of not considering the differentiability is present.

Another approach based on evolutionary algorithms was carried out by Grosan and Abraham (2008b). Solving a system of nonlinear equations was treated as a multi-objective optimisation problem where every equation represents an objective function whose goal is to minimise the difference between the right and the left terms of the corresponding equation in the system. To compare the solutions, the concept of Pareto dominance relationship has been used. GA is used as the evolutionary algorithm

used to solve the problem. Systems up to 20 variables were tested. The approach was capable of obtaining several solutions within a single run but the accuracy of a solution is around 10^{-1}. The results have revealed that the proposed approach is able to deal with high dimensional nonlinear systems. Since the problem is handled as a multi-objective one, with the increase of variables the complexity of the problem becomes high.

Apart from these meta-heuristics, many such algorithms and their variants including harmony search (HS), particle swarm optimisation (PSO), multi population parallel competitive algorithm (PICA), artificial bees colony algorithm (ABC), cuckoo search algorithm (CS) and differential evolution (DE) have been used to solve systems of nonlinear equations. However, the following areas have still room for improvement:

- solving systems in higher dimensions
- solving systems for multiple roots
- solving systems without concerning initial guesses, differentiability and continuity
- obtaining solutions with higher accuracy
- obtaining multiple approximations simultaneously.

Table 1 summarises the related work on solving systems of nonlinear equations.

2 Preliminaries

2.1 Systems of nonlinear equations

The aim of this paper is to introduce a modified firefly algorithm capable of finding roots of a system of nonlinear equations simultaneously. The algorithm is also capable of tuning its own parameters. For a better understanding of our research problem, we present the problem of interest as follows.

Let $D \subset \mathbb{R}^n$ and suppose $f : \mathbb{R}^n \to \mathbb{R}$ is a real valued function which assigns a unique real number denoted by $f(x_1, x_2, x_3, ..., x_n)$ to each $(x_1, x_2, x_3, ..., x_n) \in D$. The set D is the domain of f and its range is the set of values that f takes on. That is,

$$\{f(x_1, x_2, x_3, ..., x_n) \in \mathbb{R} \quad | \quad (x_1, x_2, x_3, ..., x_n) \in D\}.$$

We often write $z = f(x_1, x_2, x_3, ..., x_n)$ to make explicit the value taken by f at the general point $(x_1, x_2, x_3, ..., x_n)$. The variables $(x_1, x_2, x_3, ..., x_n)$ are independent and z is dependent (Nishani, 2015).

A system of nonlinear equations has the form of

$$\begin{aligned}
&f_1(x_1, x_2, ..., x_n) = 0\\
&f_2(x_1, x_2, ..., x_n) = 0\\
&\vdots\\
&f_n(x_1, x_2, ..., x_n) = 0\\
&\underline{F}(\underline{X}) = \underline{0}
\end{aligned}$$

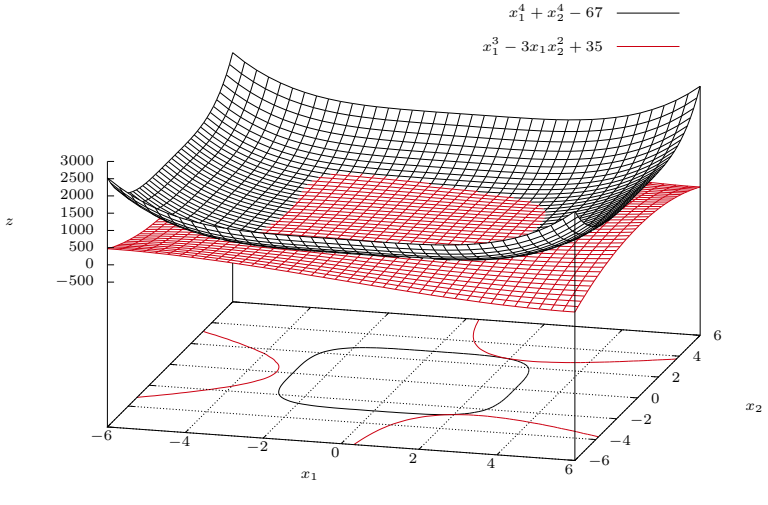
where each f_i is a nonlinear function of n variables. Some of the equations can be linear, but not all of them. Finding a solution for a nonlinear system of equations $\underline{F}(\underline{X})$ involves finding a solution such that every equation in the nonlinear system is 0, i.e., this system of n - nonlinear equations in n unknowns can alternatively be written as $\underline{F}(\underline{X}) = \underline{0}$ by defining a vector valued function $\underline{F}(\underline{X})$. The solutions we are interested in are those points (if any) that are common by producing zeros of $f_i, i = 1, ..., n$. A system of nonlinear equations can have multiple solutions. Consider the following system of two variables.

$$\begin{aligned} f_1(x_1, x_2) &: x_1^4 + x_2^4 - 67 = 0 \\ f_2(x_1, x_2) &: x_1^3 - 3x_1x_2^2 + 35 = 0 \end{aligned} \tag{1}$$

The cross sectional view (contour at $z = 0$) and the surface plot of the two functions of the `system (1)` are shown in Figure 1. For two dimensional nonlinear systems, contour plot at $z = 0$ gives a better view to get an idea about the locations of roots within the specified state space.

This system has two roots as `[1.9358, -2.6976]` and `[1.9416, 2.6954]` (approximations), within the specified region.

Figure 1 Surface plot and the contour plot at $z = 0$ for the two functions in system (1) (see online version for colours)



Here, the main focus was set on the nonlinear systems having equal number of variables and equations which are known as square systems.

2.2 State space

Solving nonlinear equations and systems of nonlinear equations are based on approximating roots rather than finding the exact solutions. Numerical methods or optimisation techniques; whatever the method adapted to solve them, it is essential to define the interval/region, where the solutions should be sought. In this research this

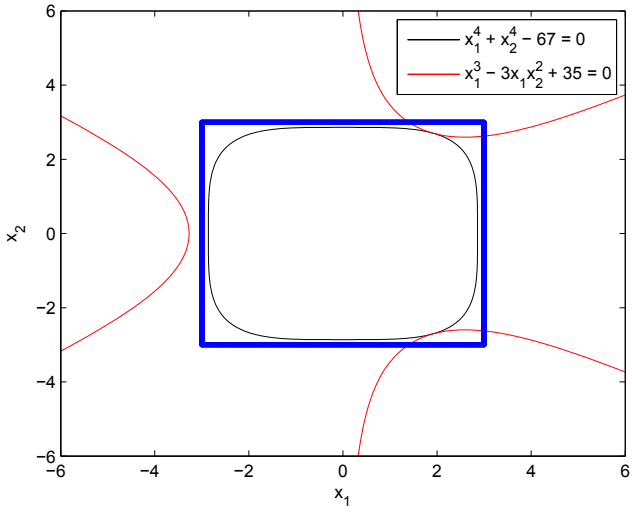
interval/region is known as state space. This section will provide an idea about the state space definitions/notations adapted in the study.

When finding root approximations for a given nonlinear system using any method, the search should be bounded with a state space. For an example, the system given in the `equation set` (1), has two roots which make both equations in the system equal to zero. When applying a method to find them, we need to define an area to find the roots, which is the state space. For this example, $0 \leq x_1 \leq 2$ and $-3 \leq x_2 \leq 3$ can be treated as the bounding values for the variables. For convenience, $-3 \leq x_1, x_2 \leq 3$ can also be used. To present the state space in a meaningful way, we have used the following notation:

$$-3 \leq x_1, x_2 \leq 3 \text{ is represented as } [x_1, x_2] \in [-3, 3]\text{x}[-3, 3] \subset \mathbb{R}^2.$$

For a two variable nonlinear system, this representation can be graphically presented as shown in Figure 2. The area bounded by the blue rectangle is the state space defined for the problem.

Figure 2 State Space representation for nonlinear system (1), $[x_1, x_2] \in [-3, 3] \times [-3, 3] \subset \mathbb{R}^2$ (see online version for colours)



When we need to generate root approximations for the given system, x_1 and x_2 values are selected from this state space. This notation is adopted throughout the paper to define the state space.

2.3 Firefly algorithm

Firefly algorithm (FA) is a meta-heuristic proposed by Yang in 2009. It is based on fireflies behaviour and how the light they emit is used to attract mates or preys. The algorithm is based on three assumptions:

1 All fireflies are gender independent, so that any individual firefly will be attracted to all other fireflies.

2 Attractiveness is proportional to their brightness, and for any two fireflies, the less bright one will be attracted by (and thus move towards) the brighter one. However, the intensity (apparent brightness) decrease as their mutual distance increases.

3 If there are no fireflies brighter than a given firefly, it will move randomly.

The brightness of a firefly should be considered and it should be associated with the objective function of the desired problem.

The pseudocode the algorithm can be stated as shown in Algorithm 1.

Algorithm 1 Pseudocode of the FA

```
Begin;
Initialise parameter values. (Set α, β0, δ and γ values according to the problem)
Define the objective function f(X)
Generate the initial population of fireflies, Xi(i = 1, 2, ..., n)
Determine the fitness Ii of i^th firefly Xi via f(Xi)
while End condition do
   for i = 1 : n (all n fireflies) do
      for j = 1 : n (all n fireflies) do
         if Ij > Ii then
            Move firefly i towards firefly j by using equation (2);
         end if
         Vary attractiveness with distance r via e^(−γr^2) using equation (3);
         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 visualisation;
End
```

The initial population for the particular problem is generated randomly. The algorithm specific parameters should be specified properly. After these initial steps, the fireflies in the population start moving towards brighter fireflies according to the following equations.

$$x_i = x_i + \beta(x_j - x_i) + \alpha(rand - 0.5) \tag{2}$$

where

$$\beta = \beta_0 . e^{-\gamma r^2} \tag{3}$$

Here x_i and x_j refer to two fireflies. β is the attraction between two fireflies and α is the parameter controlling the step size. β_0 is the attraction at $r = 0$, where r is the distance between two fireflies. The three terms in `equation` (2) represent the contribution from the current firefly, attraction between two fireflies and a randomisation term respectively. The equation supports both exploitation and exploration which are the

most essential two features that support global and local searching of a solution . α plays an important role in the randomisation process, which is from Uniform or Gaussian distribution. To control the randomness, after each iteration, Yang has used a parameter δ, the randomness reduction factor which reduces α according to `equation` (4).

$$\alpha = \alpha.\delta \quad \text{where} \quad \delta \in [0, 1] \tag{4}$$

Being a meta-heuristic stochastic natural optimiser, FA has been used in various studies in optimising problems. Applicability of FA covers many different areas including image processing (Kanimozhi and Latha, 2015; Rajinikanth and Couceiro, 2015), clustering (Senthilnath et al., 2011), forecasting (Xiao et al., 2016; Ibrahim and Khatib, 2017), engineering (Kazemzadeh Azad and Kazemzadeh Azad, 2011) and many more.

Yang et al. in 2013 introduced a framework for self-tuning algorithms and it was originally implemented with the firefly algorithm successfully.

Performance evaluation of the firefly algorithm including efficiency, robustness and convergence ability have also been discussed in many researches (Agarwal et al., 2013; Banati and Bajaj, 2013; Umbarkar et al., 2017; Senthilnath et al., 2011; Al-Saati and Alabajee, 2016).

3 Self-tuning modified firefly algorithm to solve systems of nonlinear equations

For the purpose of solving nonlinear systems, we have tried several other nature inspired algorithms as well. Since our objective is different from improving the accuracy of the roots, by experiments, FA is selected as the most suitable performer for this particular root approximating task. As many optimisation algorithms, firefly algorithm also improves solutions by both exploitation and exploration, where for the problem addressed in this research needs more exploration because to find multiple solutions simultaneously, the algorithm has to explore the search space more. The original FA has more exploration power compared to other natural algorithms so that when solving a particular nonlinear system using the original FA, it may provide several root approximations within a single run, but is limited. Within several runs it may provide several more roots but it depends on the problem and the number of roots. It is probable that it may not give almost all root approximations even after several runs as well. Therefore, as improvements, some modifications have been introduced to advance the exploration property of the algorithm. The modification includes an archive to collect root approximations and a flag to improve the algorithm enabling to explore a vast search space allowing it to find many root approximations. We have also tested the self-tuning framework on the modified firefly algorithm (Yang et al., 2013). For further clarification of the modification, the pseudo-code and the flowchart of the self-tuning modified firefly algorithm to solve systems of nonlinear equations (STMODFA) is given in Algorithm 2 and Figure 3, respectively.

The firefly algorithm has several parameters including α, γ, β and the randomness reduction parameter δ. As seen in `equation 3`, the parameter β which stands for the attraction between two fireflies, depends on three factors, β_0, γ and r. β_0 is a constant value and r which is the distance between two fireflies should be calculated during the process. Therefore the only parameter to be tuned there is γ.

Algorithm 2 Pseudocode of the self-tuning modified FA (STMODFA)

Initialise parameter values. (Set $\alpha = 0.23$, $\beta_0 = 1$ from original FA.)
Initialise intervals for algorithm dependent parameters (Set $\delta = [0.9, 1]$ and $\gamma = [0.5, 1]$ values suggested by the original implementation of FA.)
Set the objective function $f(X)$
Generate the initial population of fireflies, $X_i (i = 1, 2, ..., n)$
Determine the fitness I_i of i^{th} firefly X_i via $f(X_i)$
while End condition **do**
  for $i = 1 : n$ (all n fireflies) **do**
    for $j = 1 : n$ (all n fireflies) **do**
      if $I_j > I_i$ **then**
        Move firefly i towards firefly j by using equation (2);
      end if
      Vary attractiveness with distance r via $e^{-\gamma r^2}$ using equation (3);
      Evaluate new solutions and update light intensity;
    end for
  end for
  Find the fireflies with the eligibility criteria `fitness` $I \leq$ `0.01;`
  Put them into the `archive` and replace their positions with random fireflies;
  if No fireflies matching with eligibility criteria found at a predefined point **then**
    `count` = random integer between 1 and n/2;
    Create random fireflies up to `count` and replace the population;
  end if
end while
Post-process fireflies in the archive and get the root approximations and parameter values.

In `equation` (4), the value of the parameter α depends on the value of δ. With the experience obtained through experimentations, we initiate initial value of α to be 2.3, so that the consideration on tuning should be paid on δ.

Therefore in our modification we will be tuning two parameters, γ and δ. With this implementation, users of our STMODFA will be able to find the root approximations without the need of prior knowledge of the parameters of the firefly algorithm.

The adaption of the STMODFA to solve systems of nonlinear equations is further explained in the following sections.

3.1 Representation of a firefly

In STMODFA, each firefly in the population represents a possible approximation to the roots of the nonlinear system within the predefined state space and also a possible set of parameter values for the firefly algorithm.

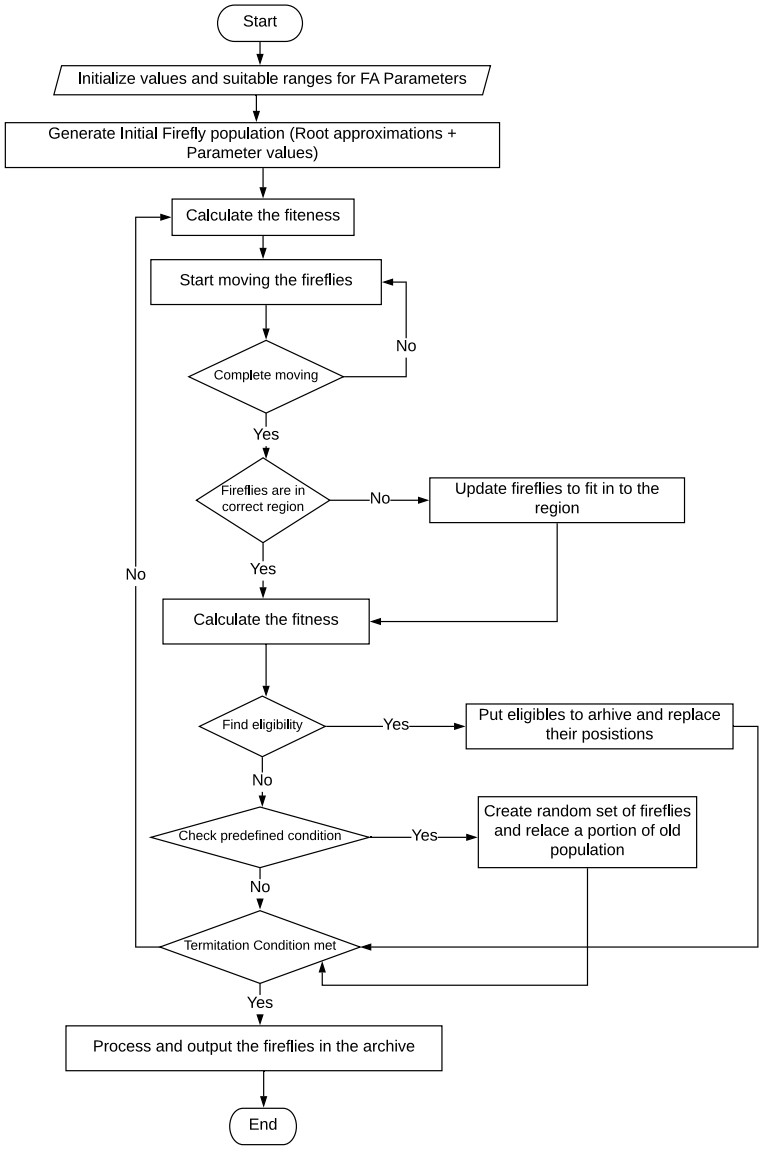
```
Firefly[i] = [(root approximations), (FA parameters)]
```

For an example, if the system is given by `equation (1)`, then a possible initial firefly would be

```
Firefly[i] = [(0.7942, -2.4148), (0.7, 0.95)]
```

within $[-3, 3] \times [-3, 3] \subset \mathbb{R}^2$ for roots and $\gamma \in [0.5, 1]$ and $\delta \in [0.9, 1]$. If the algorithm uses ten fireflies, then each firefly carries a root approximation to the system and different set of parameter approximations at the beginning.

Figure 3 Flowchart of STMODFA algorithm



3.2 Fitness/objective function

Each firefly's fitness is calculated using the objective function. Here, the fitness is calculated considering the main objective of the problem, which is finding roots. The objective function for the system is defined with the help of vector norm concept. For a particular firefly, we evaluate the functions in the system using its root approximations. The problem is treated as a minimisation problem.

For an example if we evaluate the equation set (1) at [0.7942, -2.4148] we get f_1 and f_2 values as $f_1 = -32.5986$ and $f_2 = 21.6074$. Then calculate the norm of the function values which gives the fitness of the firefly which is 39.1094.

3.3 Distance between two fireflies

To calculate the distance between two fireflies (r), we have two approaches for roots and parameters separately:

- to calculate the distance between two fireflies with regard to root approximations, we used the sum of squares of differences of root approximations of given two fireflies
- to calculate the distance between two fireflies, with regard to parameters, we used absolute value of the difference of the parameter value of given two fireflies.

3.4 Movements of fireflies

The fireflies in the population will move towards better fireflies according to the equations (2) and (3). Here a firefly who has lower fitness will move its root approximations towards a better fitted one and so its parameter values towards the parameters of the better fitted one.

3.5 Confirming region

Upon completion of one round moving towards better fireflies, the algorithm will check whether all the fireflies are within the specified range. Otherwise their values will be updated according to the defined ranges for roots as well as for parameters.

There can be various ways to confirm the region and we have adopted what Yang has done in the original implementation (Yang, 2009).

```
If X < Lb
    X = Lb; % If x less than the lower bound
elseif X > Ub
    X = Ub; % If x greater than the upper bound
```

3.6 Archiving

Once completing a moving round by all fireflies, better fireflies (root approximations + parameter values) whose objective function value is less that 10^{-2} are noted and they are put into an *archive* which can be treated as the initial modification of the new algorithm. Here the accuracy of a root is set to be 10^{-2} since our objective is to find maximum number of possible roots for the system within the specified range. Then their positions are replaced by random fireflies.

Accuracy can be improved using the concept of hybridisation where we can combine STMODFA with a numerical or other meta-heuristic algorithm. The framework of the hybrid algorithm is given in Algorithm 3.

3.7 Identifying poor iterations

Poorly performed iterations (which are not contributed to the archive) are identified at a predefined point and new fireflies will be introduced to the population in a random manner (the second main modification). In the original firefly algorithm, the movement function [`equation (2)`] will update a firefly using both exploitation and exploration properties but here enhancing the random replacements, we carry forward the exploration property further.

Finally, after a fixed number of iterations, the output will be the possible root approximations together with parameter approximations for the system within the specified state space.

Since we need to provide a single optimal approximation of one parameter for a particular nonlinear equation or a system, we have taken the average of the parameter values from the fireflies in the archive.

3.8 Improving the accuracy

The primary purpose of the STMODFA is to approximate several roots of a nonlinear system within a single run. Therefore the accuracy of a root is set to be 10^{-2}. To improve the accuracy, the idea of hybridisation can be applied. The STMODFA can be combined with FA or DE or with any better optimisation algorithm to build the desired hybrid algorithm.

A generalised pseudo-code of the hybrid algorithm is shown in Algorithm 3.

Algorithm 3 Pseudocode of the hybrid STMODFA

```
Begin;
Initialise parameter values of the selected algorithm (DE/FA/NM/WFM)
Set the objective function f(X)
Place the selected fireflies from the archive of STMODFA to create the initial populations
  of the algorithm
Determine the fitness via f(X_i)
for Each solution in the population do
    Run the algorithm to improve the accuracy
end for
Present the improved solutions as final root approximations
End
```

4 Experimentation

In this section we evaluate the performance of the proposed STMODFA in solving systems of nonlinear equations. Problems were selected from different dimensions and comparisons were carried out with different numerical and optimisation algorithms.

All the work of this research has been carried out on an Intel Core i3 laptop, with 2.30 GHz and a RAM of 8GB. MATLAB has been used as the programming language. More than 15 simple and complex nonlinear systems from various representative categories have been used to test the new algorithm. The accuracy of a root is set to be 10^{-2}.

4.1 Numerical examples

4.1.1 `Nonlinear system (1)` has been used earlier for the presentation purpose. Within $[-3, 3] \times [-3, 3]$, it has two roots.

4.1.2 The following nonlinear system has two roots within $[-6, 6]$x$[-6, 6]$

$$\begin{aligned} f_1 &: x_1^2 - x_1 + 2x_2 - 18 = 0 \\ f_2 &: (x_1 - 1)^2 + (x_2 - 6)^2 - 25 = 0 \end{aligned} \tag{5}$$

4.1.3 Apart from those two, two variable nonlinear systems shown in `Table 2` were used to test the algorithm.

Table 2 Systems of nonlinear equations with two variables

	System	*State space*	*# of roots*
1	$f_1 : x_1^2 - 10x_1 + x_2^2 - 8 = 0$ $f_2 : x_1 x_2^2 + x_1 - 10x_2 + 8 = 0$	$[-6, 6] \times [-6, 6]$	2
2	$f_1 : x_1^2 + x_2^2 - 2 = 0$ $f_2 : e^{(x_1 - 1)} + x_2^3 - 2 = 0$	$[-2, 2] \times [-2, 2]$	2
3	$f_1 : 2\cos(x_2) + 7\sin(x_1) - 10x_1 = 0$ $f_2 : 7\cos(x_1) - 2\sin(x_2) - 10x_2 = 0$	$[-1, 1] \times [-1, 1]$	1
4	$f_1 : 16x_1^2 - 80x_1 + x_2^2 + 32 = 0$ $f_2 : x_1 x_2^2 + 4x_1 - 10x_2 + 16 = 0$	$[-1, 6.2] \times [-1, 6.2]$	2

4.1.4 The famous `Merlet problem` has 13 roots within $[0, 2\pi] \times [0, 2\pi]$

$$\begin{aligned} f_1 &: -\sin(x_1)\cos(x_2) - 2\cos(x_1)\sin(x_2) = 0 \\ f_2 &: -\cos(x_1)\sin(x_2) - 2\sin(x_1)\cos(x_2) = 0 \end{aligned} \tag{6}$$

4.1.5 Mastorakis (2005) used Genetic Algorithms to solve systems of nonlinear equations. For the testing purpose he has used following systems:

$$\begin{aligned} f_1(x_1, x_2) &: x_1^2 + x_1 x_2 - 6 = 0 \\ f_2(x_1, x_2) &: x_1^2 + x_2^3 + 2x_1 x_2^2 - 3 = 0 \end{aligned} \tag{7}$$

within [–4, 4] × [–4, 4], the system has two roots

4.1.6 The following system having 13 roots within [–10, 10] × [–10, 10], is used in Effati and Nazemi (2005) and Oliveira and Petraglia (2013).

$$\begin{aligned} f_1 &: \cos(2x_1) - \cos(2x_2) - 0.4 = 0 \\ f_2 &: 2(x_2 - x_1) + \sin(2x_2) - \sin(2x_1) - 1.2 = 0 \end{aligned} \tag{8}$$

4.1.7 `Floudas problem:` this problem is used in Tsoulos and Stavrakoudis (2010) and within $[0.25, 1] \times [1.5, 2\pi]$, the system has two roots in the given domain.

$$\begin{aligned} f_1(x_1, x_2) &: 0.5\sin(x_1 x_2) - 0.25(x_2/\pi) - 0.5x_1 = 0 \\ f_2(x_1, x_2) &: (1 - (0.25/\pi))(e^{(2x_1)} - e) + (e(x_2/\pi)) - 2ex_1 = 0 \end{aligned} \tag{9}$$

4.1.8 The following two variable system has three roots within $[-2,2]\times[-2,2]$ which was tested in Ouyang et al. (2009). But there, they were able to find one root approximation at a time.

$$\begin{aligned} f_1 &: x_1^2 - x_2 + 1 = 0 \\ f_2 &: x_1 - \cos(0.5\pi x_2) = 0 \end{aligned} \tag{10}$$

4.1.9 `Himmelblau function:` This is a system having nine roots within $[-5,5]\times[-5,5]$. This system was tested in Grosan and Abraham (2008a), but they were able to approximate only seven roots within ten independent runs.

$$\begin{aligned} f_1 &: 4x_1^3 + 4x_1x_2 + 2x_2^2 - 42x_1 - 14 = 0 \\ f_2 &: 4x_2^3 + 2x_1^2 + 4x_1x_2 - 26x_2 - 22 = 0 \end{aligned} \tag{11}$$

4.1.10 This two variable system has 11 roots within $[-1,1]\times[-1,1]$.

$$\begin{aligned} f_1 &: x_1 - \sin(5\pi x_2) = 0 \\ f_2 &: x_1 - x_2 = 0 \end{aligned} \tag{12}$$

4.1.11 The following nonlinear system has six roots within $[-2,2]\times[-2,2]$, which was tested in Ramos and Monteiro (2017).

$$\begin{aligned} f_1 &: e^{x_1^2+x_2^2} - 3 = 0 \\ f_2 &: x_1 + x_2 - \sin(3(x_1 + x_2)) = 0 \end{aligned} \tag{13}$$

4.1.12 This three variable nonlinear system has two roots within the state space $[-2,2]\times[-2,2]\times[-2,2]$ which was tested in Xiao and Yin (2015).

$$\begin{aligned} f_1 &: x_1 + x_2 - e^{-x_3} = 0 \\ f_2 &: x_1 + x_3 - e^{-x_2} = 0 \\ f_2 &: x_2 + x_3 - e^{-x_1} = 0 \end{aligned} \tag{14}$$

4.1.13 Following system when n = 13 has two roots which was also tested in Xiao and Yin (2015)

$$\begin{aligned} f_i(x_1,x_2,x_3,..,x_{13}) &: x_i sin(x_{i+1}) - 1 = 0 \quad 1 \le i \le n-1 \\ f_n(x_1,x_2,x_3,..,x_{13}) &: x_n + sin(x_1) = 0 \end{aligned} \tag{15}$$

4.1.14 The `nonlinear system 16` has two roots within $[-4,4]\times[-4,4]$.

$$\begin{aligned} f_1(x_1,x_2) &: x_1^2 - x_2^2 = 0 \\ f_2(x_1,x_2) &: 1 - abs(x_1 - x_2) = 0 \end{aligned} \tag{16}$$

4.1.15 The complexity occurs when the number of variables increases. In the literature, such situations were rarely addressed and natural algorithms alone were not capable of handling them. The problems were solved treating them as multi-objective cases or problems with constraints. The new algorithm was capable of handling such situations well.

`Neurophysiology applications:` here, a complex set of nonlinear equations which concerns a neurophysiology problem is considered. The system consists of six equations with six unknown variables which was tested in Grosan and Abraham (2008b). For these types of systems it is extremely difficult to identify the number of roots exists for a particular solution space. Here we are interested about roots within $[-2,2]\times[-2,2]\ldots\times[-2,2]\subset\mathbb{R}^6$.

$$\begin{aligned}
&f_1: x_1^2+x_3^2-1=0\\
&f_2: x_2^2+x_4^2-1=0\\
&f_3: x_5x_3^3+x_6x_4^3=0\\
&f_4: x_5x_1^3+x_6x_2^3=0\\
&f_5: x_5x_1x_3^2+x_6x_4^2x_2=0\\
&f_6: x_5x_1^2x_3+x_6x_2^2x_4=0
\end{aligned}\tag{17}$$

4.1.16 `Arithmetic applications:` this is a ten variable complex nonlinear system proposed from interval arithmetic (Grosan and Abraham, 2008b; Hong and Stahl, 1994; Moore, 1980).

$$\begin{aligned}
&f_1: x_1-0.25428722-0.18324757x_4x_3x_9=0\\
&f_2: x_2-0.37842197-0.16275449x_1x_{10}x_6=0\\
&f_3: x_3-0.27162577-0.16955071x_1x_2x_{10}=0\\
&f_4: x_4-0.19807914-0.15585316x_7x_1x_6=0\\
&f_5: x_5-0.44166728-0.19950920x_7x_6x_3=0\\
&f_6: x_6-0.14654113-0.18922793x_8x_5x_{10}=0\\
&f_7: x_7-0.42937161-0.21180486x_2x_5x_8=0\\
&f_8: x_8-0.07056438-0.17081208x_1x_7x_6=0\\
&f_9: x_9-0.34504906-0.19612740x_{10}x_6x_8=0\\
&f_{10}: x_{10}-0.42651102-0.21466544x_4x_8x_1=0
\end{aligned}\tag{18}$$

We consider finding roots within $[-1,1]\times[-1,1]\times\ldots\times[-1,1]\subset\mathbb{R}^{10}$.

4.1.17 In a recent research, multi-population parallel imperialist competitive algorithm (PICA) is used to solve systems of nonlinear equations (Majd et al., 2016). They have used the following system for testing.

$$\begin{aligned}
&f_1: x_1^3-3x_1x_2^2-1=0\\
&f_2: 3x_1^2x_2-x_2^3+1=0
\end{aligned}\tag{19}$$

4.1.18 The following system has two roots, with locations far from each other.

$$\begin{aligned} f_1 &: x_2^2 + x_1 - 1 = 0 \\ f_2 &: \frac{x_2^3}{8} + x_1 - 1 = 0 \end{aligned} \tag{20}$$

4.1.19 The following system has a nonlinear function which is not differentiable at $x_1 = x_2 = 0$

$$\begin{aligned} &f_1(x_1, x_2) : (x_1^3 x_2^2 + 5)/(x_1^2 + x_2^2) = 0 \\ &f_2(x_1, x_2) : 2(x_2 - x_1) + \sin(2x_2) - \sin(2x_1) - 7 = 0 \end{aligned} \tag{21}$$

4.2 Results

4.2.1 The STMODFA has given approximations for both roots for the `nonlinear system (1)`. The algorithm rapidly converges towards the solutions with iterations (see `graph 1` in Figure 7).

# of roots	*Roots approximated by STMODFA*		*Values of nonlinear equations at approximated roots*	
	x_1	x_2	f_1	f_2
2	1.9422	2.6953	–0.3466E-03	–0.1753E-03
	1.8831	–2.7161	0.4085E-03	0.5265E-03

4.2.2 Following two approximations were given by STMODFA for the `nonlinear system` (5).

# of roots	*Roots approximated by STMODFA*		*Values of nonlinear equations at approximated roots*	
	x_1	x_2	f_1	f_2
2	–3.0000	3.0000	0	0
	4.2269	2.1807	0.0012	–0.0001

4.2.3 `Nonlinear system` (6) represents a famous `Merlet problem` having 13 roots within $[0, 2\pi] \times [0, 2\pi]$ interval. The STMODFA has given approximations for all 13 roots with the accuracy of 10^{-2}. The following shows the obtained root approximations.

	Roots approximated by STMODFA		*Values of nonlinear equations at approximated roots*	
	x_1	x_2	f_1	f_2
1	0	0	0	0
2	0	6.284850767844894	–0.3330919790753E-02	–0.1665459895376E-02
3	6.281770409549054	0	0.1414897158444E-02	0.2829794316887E-02
4	6.283388930739925	6.284034589669578	–0.1902188226076E-02	–0.1256529341282E-02
5	0	3.145322070136794	0.7458815803757E-02	0.3729407901879E-02
6	3.141888855063051	0	0.29620146892636E-03	0.59240293785272E-03
7	3.141264630453968	3.137542793922269	0.8427717198166E-02	0.4705889258982E-02
8	4.705950309337243	1.576486883159703	0.7186636924041E-02	–0.4942293085469E-02
9	1.568506201461184	4.710348179959847	0.6621030782291E-02	0.6371705876005E-02
10	1.576577228522599	1.564614184499570	0.5379518503570E-02	–0.6583240174145E-02
11	4.712236272994278	4.708826389846009	–0.3867995802519E-02	–0.7277872342834E-02
12	3.134799007359507	6.288819970485428	0.4475520795793E-02	–0.7952468790730E-02
13	6.279156782180293	3.143706109910964	0.0198370090649E-02	–0.5943572613271E-02

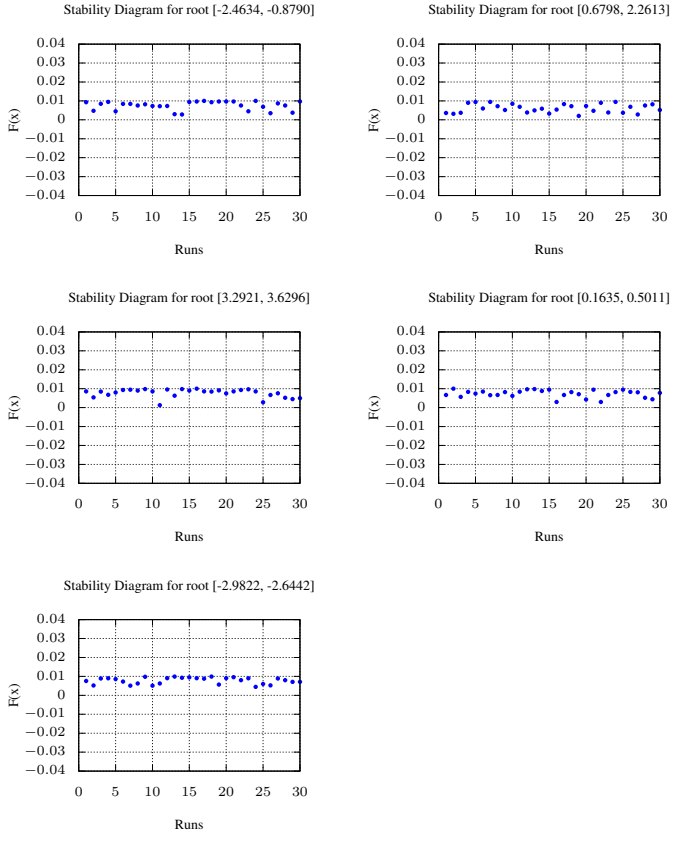
4.2.4 `Nonlinear system` (8) has 13 roots within $[-10, 10] \times [-10, 10]$, and the proposed approach was able to approximate all 13 roots simultaneously (within a single run). The results are shown in the following table. In both Effati and Nazemi (2005) and Oliveira and Petraglia (2013), they were able to find only one root approximation, but with a good accuracy. In our approach, the accuracy is set to be 10^{-2}, but is capable of finding all the approximations within a run.

	Roots given by STMODFA		*Function values approximated at given roots*	
	x_1	x_2	f_1	f_2
1	–9.2616	–8.9235	0.0091	–0.0015
2	–8.7439	–7.1683	0.0056	–0.0072
3	–6.1245	–5.7874	0.0027	–0.0010
4	–5.6052	–4.0204	–0.0011	0.0010
5	–2.9872	–2.6518	–0.0047	–0.0028
6	–2.4613	–0.8780	–0.0072	0.0057
7	0.1584	0.4928	–0.0021	–0.0091
8	0.6815	2.2615	–0.0054	–0.0006
9	3.3001	3.6352	–0.0008	–0.0070
10	3.8239	5.3993	0.0004	–0.0087
11	6.4411	6.7798	0.0046	0.0046
12	6.9637	8.5458	–0.0057	0.0036
13	9.5857	9.9248	0.0084	0.0034

The stability of the STMODFA is illustrated here by running the algorithm 30 times (runs) for a particular nonlinear system and obtaining the average norm value of the system at each run for each root approximation in the

system. The stability diagrams is presented here for the `nonlinear system` (8) within $[-5, 5] \times [-5, 5]$.

Figure 4 Stability diagrams for root approximations of `nonlinear system` (8) within $[-5, 5] \times [-5, 5]$ (see online version for colours)



4.2.5 For the `nonlinear system` (10), our approach has given all three root approximations with an accuracy of 10^{-2}.

Root number	*Roots approximated by STMODFA*		*Values of nonlinear equations at approximated roots*	
	x_1	x_2	f_1	f_2
1	0.0023	1.0036	–0.0036	0.0080
2	–1.0029	2.0000	0.0058	–0.0029
3	–0.7192	1.5186	–0.0014	0.0083

4.2.6 The STMODFA successfully worked for the `nonlinear system` (11). It has given approximations for all nine roots within $[-5, 5] \times [-5, 5]$ within a single run. The accuracy of the roots is 10^{-2}.

Root number	*Roots approximated by STMODFA*		*Values of nonlinear equations at approximated roots*	
	x_1	x_2	f_1	f_2
1	–0.2707	–0.9231	–0.0062	0.0003
2	0.0868	2.8843	–0.0032	0.0048
3	–3.0734	–0.0802	–0.0041	–0.0039
4	3.3848	0.0782	0.0026	–0.0058
5	3.0013	1.9988	0.0072	–0.0014
6	3.5850	–1.8511	0.0039	–0.0834
7	–0.1275	–1.9585	0.0170	–0.0967
8	–3.7793	–3.2833	0.0044	–0.0103
9	–2.8051	3.1314	0.0013	0.0071

4.2.7 The following shows the 11 root approximations given by STMODFA to the `nonlinear system` (12)

Roots approximated by STMODFA		*Values of nonlinear functions at approximated roots*	
x_1	x_2	f_1	f_2
–0.9313	–0.9247	–0.0056	–0.0066
–0.8602	–0.8656	–0.0027	0.0054
–0.5668	–0.5620	–0.0047	–0.0048
–0.4235	–0.4278	–0.0006	0.0043
–0.1903	–0.1880	–0.0029	–0.0023
0	0	0	0
0.1946	0.1873	–0.0036	0.0073
0.4356	0.4287	–0.0001	0.0069
0.5551	0.5627	0.0021	–0.0076
0.8624	0.8653	0.0073	–0.0029
0.9214	0.9241	–0.0078	–0.0027

4.2.8 For the `nonlinear system` (13), the Ramos method has considered the region $([-2.3, 0.1] \times [-2, -0.2]) \cup ([1.1, 2.3] \times [0.2, 2])$ with step-sizes $h_{x_1} = h_{x_2} = 0.2$ for taking the initial guesses to obtain the six root approximations. In that, one should be aware of number of roots in the specified area and carefully pick the initial guesses to obtain all approximations. Otherwise it is time consuming and it will be hard to obtain or track all the roots. Addressing such problems, STMODFA specify a single search space $[-2, 2] \times [-2, 2]$ and it has given all six root approximations with an accuracy of 10^{-2}.

Table 3 Roots approximated by STMODFA for the nonlinear system (13)

Roots approximated by STMODFA		*Function values in the nonlinear system*	
x_1	x_2	f_1	f_2
0.257033666243153	–1.015878082671395	–0.1612679089768E-02	0.2289377875509E-02
–1.015462753081635	0.257073345070842	–0.4080165934186E-02	0.3629025330187E-02
0.740942616875986	–0.740731821007316	–0.2796785058200E-02	–0.421591695190E-03
–0.742306592832714	0.740741689363404	0.3316950710431E-02	0.3129789693164E-02
1.017045229880979	–0.254074559990805	0.967935086337E-03	0.9923786355104E-02
–0.257232336685809	1.017176115398260	0.6617596364217E-02	0.953242416164E-03

4.2.9 Complex nonlinear systems (17) and (18) were solved in Grosan and Abraham (2008b) using evolutionary approach, considering the problem as a multi-objective optimisation problem with Pareto dominance relationship. We have solved the same using STMODFA with the help of the archiving property and treating the problem as a single objective problem. The results indicate that STMODFA is capable of finding better solutions (accuracy of 10^{-2}), and several solutions simultaneously, which is significantly the most advantageous property of the new algorithm. But it is clear that more computational power is needed when the dimension of the problem increases.

Table 4 Comparison of solutions of the nonlinear system (17) obtained by STMODFA and Grosan and Abraham (2008b) for neurophysiology application

Present study (STMODFA)					*Grosan and Abraham (2008b)*		
x_1	–1.0000	0.8991	–0.9699	0.9453	-0.8282	–0.6512	0.0425
x_2	–1.0000	–0.7319	0.9641	0.3767	0.5446	–0.6858	–0.1626
x_3	0.0326	0.4379	0.2445	–0.3269	–0.0094	–0.4637	–0.9215
x_4	0.0282	–0.6815	–0.2677	0.9257	0.7633	–0.6450	0.9841
x_5	–0.1737	–0.0112	0.1462	0.0044	0.0199	0.1535	–0.6789
x_6	0.1758	–0.0099	0.1473	0.0018	0.1466	–0.0036	–0.9070
f_1	0.11E-02	0.1E-03	0.5E-03	0.4E-03	0.3139	0.3607	0.1489
f_2	0.8E-03	0	0.11E-02	–0.11E-02	0.1206	0.1134	0.0049
f_3	0	0.22E-02	–0.7E-03	0.13E-02	0.0652	0.0143	0.3332
f_4	–0.22E-02	–0.42E-02	–0.14E-02	0.38E-02	0.0123	0.04123	0.0038
f_5	0	0.15E-02	0.17E-02	0.1E-02	0.0465	0.0204	0.1183
f_6	–0.7E-03	–0.3E-03	–0.3E-02	–0.1E-02	0.0330	0.02909	0.0224

4.2.10 The `nonlinear system` (20) has two roots $(1, 0)$ and $(-63, 8)$. It is clear that the two roots maintain quite long distance from each other. Ramos and Monteiro (2017) in their study obtained the two solutions by providing two different initial guesses in two different runs. The accuracy of the solutions is nearly same as the Newton's method. They have obtained approximations less than 100 iterations. When we solved the same using STMODFA, within the state space $[-70, 10] \times [-70, 10]$, both root approximations were given with an accuracy of 10^{-2}. This can be considered as a remarkable

performance of STMODFA, giving approximations within a quite large state space. However, this is a two variable system. With large nonlinear systems this might not be possible.

Figure 5 Surface plot and the contour plot at $z = 0$ for the two functions in the system (20) – two roots situated far from each other (see online version for colours)

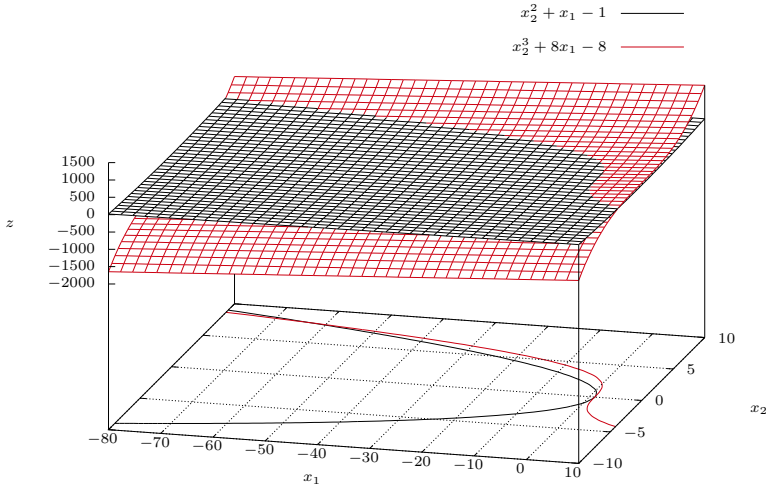


Table 5 Results obtained for the `nonlinear system` (20) by STMODFA and Ramos method

STMODFA		*Ramos and Monteiro (2017)*	
State space	[–70, 10]	Initial guess 1	[0.5, –0.5]
Root approximations	x_1 = 1.003387825129016 x_2 = 0 x_1 = –63.017321230021807 x_2 = 8.001072968678789	Root approximation 1 Function values Iterations	x_1 = 1 x_2 = –5.4867526914E-26 f_1 = 0 f_2 = 0 85
Function values	f_1 = 0.3387825129016E-02 f_2 = 0.3387825129016E-02 f_1 = –0.152579899392E-03 f_2 = 0.8433472208907E-02	Initial guess 2 Root approximation 2 Function values	[60, 6] x_1 = –63 x_2 = 8 f_1 = 0 f_2 = 0
Iterations	200	Iterations	10
Runs	1	Runs	2

4.2.11 The first function in the `nonlinear system` (21) is not differentiable at $x_1 = x_2 = 0$. The graphical representation of the systems is shown in Figure 6.

Within $[-6, 6] \times [-6, 6]$, this system possesses three roots. For the famous algorithms such as Newton's method, if the function is not continuously differentiable in a neighbourhood of the root then there is a possibility that it will always diverge and fail, unless the solution is guessed on the first try. Without considering such difficulties, STMODFA was capable of approximating all three roots in a single run with 50 fireflies at 10^{-2} accuracy.

Figure 6 Surface plot and the contour plot at $z = 0$ for the two functions in the system (21) (see online version for colours)

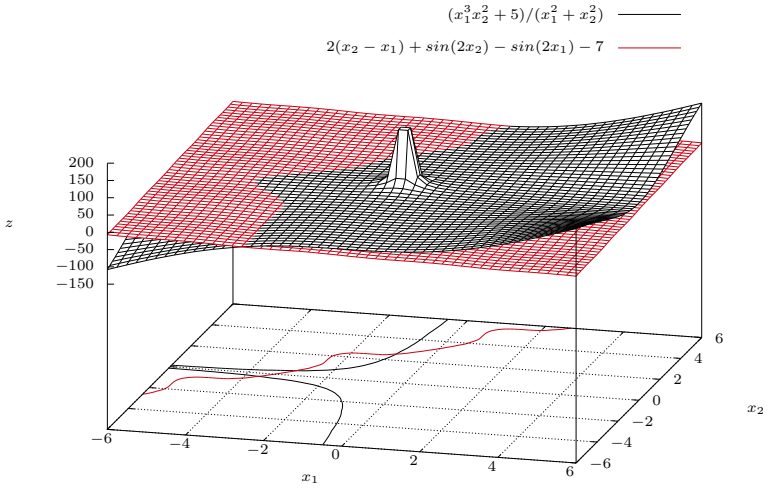


Table 6 Results obtained for the `nonlinear system` (21) by STMODFA

STMODFA	
Root approximations	*Function values*
x_1 = –3.674397454296519	f_1 = –0.1396991630696E-02
x_2 = –0.318074976794823	f_2 = –0.6373841987918E-02
x_1 = –0.911575610747701	f_1 = 0.2446517445982E-02
x_2 = 2.564527993036838	f_2 = 0.6091403594414E-02
x_1 = –2.886742507609634	f_1 = –0.8759490599100E-02
x_2 = 0.459302958961911	f_2 = –0.1068439096563E-02

The convergence ability of the STMODFA is also tested with several nonlinear systems. Since this is a problem with multi optimum solutions, there are several ways to draw convergence diagrams.

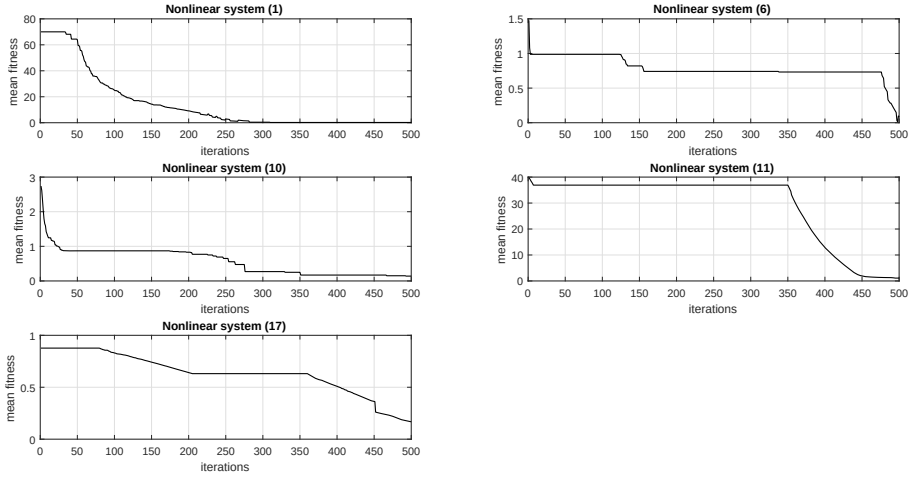
1 draw fitness of best solution vs. time/iteration

2 draw average fitness of the population vs. time/iteration

3 draw fitness of all solutions vs. time/iteration.

We have selected the second method, since it clearly expresses the idea of convergence (see Figure 7). Since this is a problem with multi optimum solutions, we can also graph the fitness of all solutions vs. time/iteration. But the diagrams can be more complex and will not show the idea of convergence properly.

Figure 7 Convergence diagrams for the nonlinear systems (1), (6), (10), (11) and (17)



Since STMODFA is a stochastic method, stability, convergence as well as the robustness of the algorithm should be discussed. The performance of the algorithm is tested over 30 runs and following measures were used for the evaluation as used in a similar study (Liao et al., 2018). The STMODFA's capability was tested against canonical FA and with two other studies (Liao et al., 2018; Song et al., 2015).

a bSuccess rate (SR): number of successful runs (runs where all known roots of a system of nonlinear equation are found) out of 30 runs.

$$SR = \frac{N_{sr}}{N_r},$$

where N_{sr} is number of successful runs and N_r is total number of runs. For all 20 test problems, an average success rate of 0.998 was obtained by STMODFA which shows the strong capability of the algorithm. Figure 8 shows the plotted SR values against the test problems.

b Root ratio (RR): it measures the average ratio of all known roots found over multiple runs.

$$RR = \frac{\sum_{i=1}^{N_r} N_{f,i}}{NoR.N_r}$$

where N_r is the number of runs; $N_{f,i}$ is the number of roots found in the i^{th} run; and NoR is the number of known roots of an nonlinear systems of equations. For the tested 20 nonlinear systems STMODFA has given an average of 0.999727 as

the RR value. There were several similar systems of nonlinear equations in STMODFA and (Liao et al., 2018). We have compared the SR and RR values of the nonlinear systems in both studies and of canonical FA. The detailed results of root ratio (RR) and success rate (SR) values are reported in the Tables 7, 8 and 9.

Figure 8 Success rate of STMODFA over 20 test problems

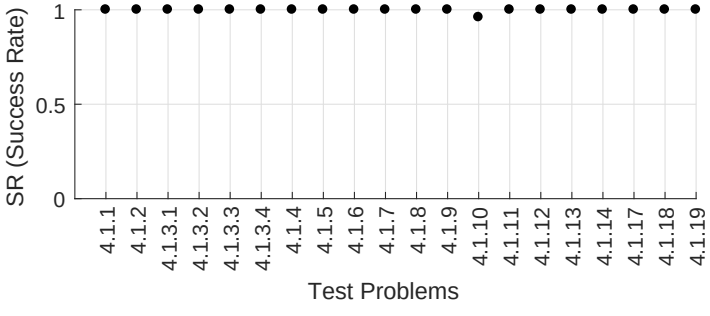


Table 7 Success rate (SR) and the root ratio (RR) values for the 30 nonlinear systems tested in STMODFA over 30 runs

Prob	*SR*	*RR*	*Prob*	*SR*	*RR*
4.1.1	1.0000	1.0000	4.1.8	1.0000	1.0000
4.1.2	1.0000	1.0000	4.1.9	1.0000	1.0000
4.1.3.1	1.0000	1.0000	4.1.10	0.9600	0.994545
4.1.3.2	1.0000	1.0000	4.1.11	1.0000	1.0000
4.1.3.3	1.0000	1.0000	4.1.12	1.0000	1.0000
4.1.3.4	1.0000	1.0000	4.1.13	1.0000	1.0000
4.1.4	1.0000	1.0000	4.1.14	1.0000	1.0000
4.1.5	1.0000	1.0000	4.1.17	1.0000	1.0000
4.1.6	1.0000	1.0000	4.1.18	1.0000	1.0000
4.1.7	1.0000	1.0000	4.1.19	1.0000	1.0000
Avg SR	0.998				
Avg RR	0.999727				

Table 8 Comparison between STMODFA and DR-JADE (Liao et al., 2018), MONES (Song et al., 2015) and FA (Yang, 2009) with respect to Success Rate (SR)

	NES as in STMODFA	*STMODFA*	*DR-JADE (Liao et al., 2018)*	*MONES (Song et al., 2015)*	*FA (Yang, 2009)*
01	4.1.4	**1.0000**	**1.0000**	**1.0000**	0.0000
02	4.1.6	**1.0000**	**1.0000**	0.0000	0.0000
03	4.1.7	**1.0000**	**1.0000**	**1.0000**	0.5000
04	4.1.9	**1.0000**	**1.0000**	0.0000	0.0000
05	4.1.10	**1.0000**	0.9667	0.7667	0.0000
06	4.1.14	**1.0000**	**1.0000**	**1.0000**	0.6666
07	4.1.17	**1.0000**	**1.0000**	**1.0000**	0.3333

Table 9 Comparison between STMODFA and DR-JADE (Liao et al., 2018), MONES (Song et al., 2015) and FA (Yang, 2009) with respect to Root Ratio (RR)

	NES as in STMODFA	*STMODFA*	*DR-JADE (Liao et al., 2018)*	*MONES (Song et al., 2015)*	*FA (Yang, 2009)*
01	4.1.4	**1.0000**	**1.0000**	**1.0000**	0.4921
02	4.1.6	**1.0000**	**1.0000**	0.5923	0.2314
03	4.1.7	**1.0000**	**1.0000**	**1.0000**	0.7215
04	4.1.9	**1.0000**	**1.0000**	0.1963	0.3421
05	4.1.10	**1.0000**	0.9970	0.9758	0.5347
06	4.1.14	**1.0000**	**1.0000**	**1.0000**	0.8845
07	4.1.17	**1.0000**	**1.0000**	**1.0000**	0.6453

4.3 Self-tuning of STMODFA

While collecting the possible root approximations of a given nonlinear system, the archive in the STMODFA also collects values approximated for parameters γ and δ for the system. The problem here is that a system might have more than one root and the archive of STMODFA will give approximations for all such roots and since root approximation is also associated with parameter approximation, ranges of the parameter values might be different with different roots. The following example will explain this further.

Within a single run, STMODFA has given following solutions in the archive to the `nonlinear system` (19) (see Table 11). The system has three roots and the archive contain 15 solutions/approximations (one for the first root, six for the second root and eight for the third root). Therefore we are getting 15 approximations for γ and δ as well.

Table 10 Optimised parameter values for different nonlinear systems given by STMODFA

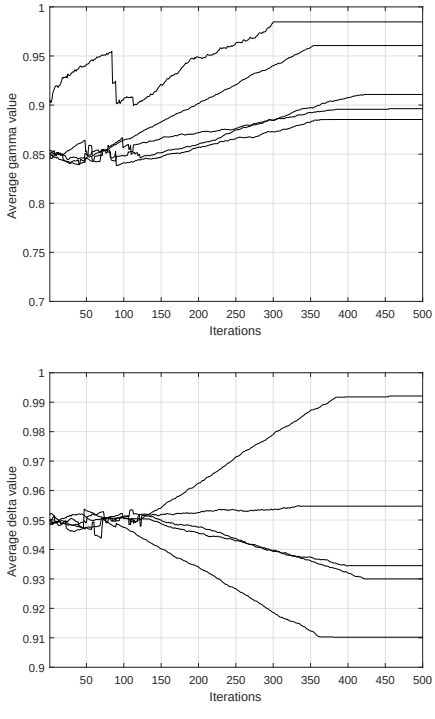
	Nonlinear system	*Average optimum gamma (γ) value*	*Average optimum delta (δ) value*
1	`Nonlinear system` (1)	0.8429	0.9871
2	`Nonlinear system` (5)	0.8250	0.9244
3	`Nonlinear system` (8)	0.8313	0.9631
4	`Nonlinear system` (10)	0.9071	0.9107
5	`Nonlinear system` (11)	0.8955	0.9021
6	`Nonlinear system` (18)	0.9153	0.9702
7	`Nonlinear system` (19)	0.9729	0.9667
8	`Nonlinear system` (20)	0.9099	0.9679
9	`Nonlinear system` (21)	0.7214	0.9633

Since we need to provide a single optimal approximation of one parameter for a particular system, we have taken the average of the parameter values from the fireflies in the archive. Such parameter approximations obtained at a particular run for several systems are shown in Table 10.

Table 11 Solutions in the archive for the `nonlinear system` (19)

	Fireflies in the archive			
	Root approximations		*Gamma (γ) value*	*Delta (δ) value*
1	1.0871	–0.2952	0.8826	0.9208
2	–0.7891	–0.7974	0.9018	0.9235
3	–0.7854	–0.7964	0.7126	0.9020
4	–0.8013	–0.7905	0.7133	0.9022
5	–0.7905	–0.7870	0.7111	0.9021
6	–0.8015	–0.7873	0.7108	0.9021
7	–0.8009	–0.7890	0.7108	0.9021
8	–0.3007	1.0851	0.9038	0.9943
9	–0.2979	1.0916	0.9037	0.9942
10	–0.3006	1.0810	0.9038	0.9941
11	–0.2985	1.0827	0.9042	0.9940
12	–0.2989	1.0871	0.9042	0.9940
13	–0.2987	1.0825	0.9033	0.9939
14	–0.2984	1.0789	0.9034	0.9939
15	–0.2994	1.0843	0.9023	0.9939

Figure 9 Variation of the average γ and δ values over iterations for the `nonlinear system` (19)



Note: Graphs were drown at randomly selected five runs.

Figure 9 shows how the average value of the parameters changes over the iterations for five random runs. It is clear that average value can be different but they are getting stable (optimise) over the iterations confirming that self-tuning works well with the new modified firefly algorithm.

Table 12 Data for the statistical analysis

Nonlinear system	*# of roots*	*STMODFA*		*DE/2-Opt/1*		*DE/2-Opt/2*	
		# of roots	*Accuracy*	*# of roots*	*Accuracy*	*# of roots*	*Accuracy*
Nonlinear system 1	2	2	10^{-3}	1	10^{-9}	1	10^{-20}
Nonlinear system 5	2	2	10^{-2}	1	10^{-2}	1	10^{-20}
Nonlinear system 2.1 (Table 2)	2	2	10^{-3}	1	10^{-6}	1	10^{-15}
Nonlinear system 2.2 (Table 2)	2	2	10^{-2}	1	10^{-3}	1	10^{-20}
Nonlinear system 2.3 (Table 2)	1	1	10^{-3}	1	10^{-4}	1	10^{-20}
Nonlinear system 2.4 (Table2)	2	2	10^{-2}	1	10^{-2}	1	10^{-20}
Nonlinear system 6	13	13	10^{-2}	1	10^{-6}	1	10^{-20}
Nonlinear system 7	2	2	10^{-2}	1	10^{-7}	1	10^{-15}
Nonlinear system 8	13	13	10^{-2}	1	10^{-3}	1	10^{-20}
Nonlinear system 9	2	2	10^{-2}	1	10^{-16}	1	10^{-16}
Nonlinear system 10	3	3	10^{-2}	1	10^{-6}	1	10^{-20}
Nonlinear system 11	9	9	10^{-2}	1	10^{-2}	1	10^{-20}
Nonlinear system 16	2	2	10^{-2}	1	10^{-11}	1	10^{-20}
Nonlinear system 17	Unknown	5	10^{-2}	1	10^{-1}	1	10^{-1}
Nonlinear system 18	Unknown	5	10^{-2}	1	10^{-1}	1	10^{-5}

4.4 Comparisons

We have compared the performances of STMODFA with a few variants of the DE. We have used three variants. Two of them are selected from a latest review article (Das et al., 2016). To achieve a faster and better convergence than classical DE, Chiang et al. (2010) proposed a variant of DE, where a mutation scheme influenced by the 2-Opt algorithm. Improving the performance of DE, the paper introduces 2-Opt based DE (2-Opt DE) which is inspired by 2-Opt algorithms to accelerate DE. The novel mutation schemes of 2-Opt DE, DE/2- Opt/1 and DE/2-Opt/2 were substituted for mutation schemes of the original DE namely DE/rand/1 and DE/rand/2. Fifteen systems are used to test the algorithms. All algorithms (STMODFA, DE/2-Opt/1 and DE/2-Opt/2) are kept under same conditions. With the results, it is clear that DE/2-Opt/1 and DE/2-Opt/2 perform well on behalf of accuracy of the roots. Accuracy is better in DE/2-Opt/2 than in DE/2-Opt/1 and STMODFA. But both algorithms has given a single root approximation at a single run where the objective of the STMODFA is different. For DE/2-Opt/1 and DE/2-Opt/2, we cannot guarantee that they will provide different root approximations at different runs. Therefore although the accuracy is good, using these methods it is

impossible to get an idea about the distribution of roots within a given state space. STMODFA with a moderate accuracy is able to provide many root approximations within a single run. The statistical analysis further proves this idea. Data used are shown in Table 12.

We have conducted a non-parametric analysis to clarify the results. The Friedman test is a non-parametric alternative to ANOVA with repeated measures. It can be used for situations where sample size is quite small (here it is 15). No normality assumption is required. It is used to test for differences between groups when the dependent variable being measured is at least in ordinal scale (can use with interval and ratio data as well). The following hypotheses were tested under a significance level (α) of 0.05:

H_0 There is no any difference between three algorithms.

H_1 At least two algorithms are different from each other.

The test has been conducted over 13 samples (two were removed) for three algorithms, STMODFA, DE/2-Opt/1 and DE/2-Opt/2. Table 12 shows number of root approximations found by each algorithm and the accuracy. Results of the statistical analysis are shown in Table 13. The data analysis is performed using Minitab statistical software version 17.0 (Minitab, 2010).

```
Friedman test: percentage versus algorithm blocked by equation.
```

Since **P value** $< \alpha$, we reject the null hypothesis and conclude that at least two algorithms are different from each other at 0.05 level of significance.

Table 13 Friedman test for the root approximations over three algorithms

S = 16.62	DF = 2	P = 0.000	
S = 24.00	DF = 2	P = 0.000 (adjusted for ties)	
Algorithm	*N*	*Est median*	*Sum of ranks*
DE2 OPT1	13	50	20.0
DE2 OPT2	13	50	20.0
STMODFA	13	100	38.0
Grand median = 66.67			

To find out which algorithm is different from which, we have calculated the critical difference value (CD) for the mean comparison, using the following equation:

$$CD = Z_\alpha / k(k-1)\sqrt{\frac{nk(k+1)}{6}} \tag{22}$$

Here $\alpha = 0.05$, k is the number of difference groups, which is 3 and n is the sample size, which is 13.

The CD value calculated according to the `equation` (22) is 12.2077. When we consider STMODFA and DE/2-Opt/1, the difference between their sum of ranks is 18 (=38-20), which is greater than the CD value (12.2077), which means the performances

of two algorithms are different when compared with number of root approximations given. Similarly, for STMODFA and DE/2-Opt/2, the difference between sum of ranks is 18 (=38 - 20), which is also greater than the CD value (12.2077), indicating that the performances of those two algorithms are also different (The complete calculations of differences of sum of ranks can be seen in Table 14, The '*' indicates algorithms which are different from each other).

Table 14 Absolute difference between sum of ranks of the 3 algorithms

	DE/2-Opt/1	*DE/2-Opt/2*
STMODFA	18*	18*
DE/2-Opt/1	-	0
DE/2-Opt/2	0	-

Therefore, we can conclude that, STMODFA performs differently from both DE/2-Opt/1 and DE/2-Opt/1 algorithms and since this is a maximisation problem and maximum sum of ranks value belongs to the STMODFA algorithm, conclusions be made as STMODFA algorithm outperforms other two algorithms.

5 Conclusions

The proposed algorithm, formed by converting a system of nonlinear equations to an optimisation problem, was successfully implemented on the root finding problems. Results presented here from various benchmark problems indicate that almost all roots of a system of nonlinear equations for a reasonable search space can be found without prior initial guesses or without considering the differentiability or even the continuity of the equations in the system.

Most of the numerical examples we have tested, carry more than one solution. For a system of nonlinear equations, especially when the number of variables increases, it is hard to determine the number of roots within a given search space. Most of the standard methods like Newton's method and new approaches found in the literature are unable to find several solutions within a single run. It is worth mentioning here that our approach obtains almost all solutions within a single run. This is undoubtedly the main advantage of using the proposed algorithm.

A self-tuning framework has also been implemented on the modified firefly algorithm. It has worked well providing parameter free environment to the algorithm. The algorithm is named as self-tuning modified firefly algorithm (STMODFA). STMODFA outperform other methods in the literature in solving systems of nonlinear equations giving several solutions simultaneously with a reasonable accuracy and providing a user friendly environment to the algorithm users. The modified firefly algorithm has become more convenient with the inclusion of the self-tuning framework.

Acknowledgements

The authors would like to thank Dr. Xin-She Yang for his valuable suggestions and explanations on implementing the Modified Firefly Algorithm and Ms. W.J. Polegoda,

lecturer at the Department of Export Agriculture, Faculty of Animal Science & Export Agriculture, Uva Wellassa University, Sri Lanka for the guidance given on the statistical analysis.

References

Abdollahi, M., Bouyer, A. and Abdollahi, D. (2016) 'Improved cuckoo optimization algorithm for solving systems of nonlinear equations', *The Journal of Supercomputing*, Vol. 72, No. 3, pp.1246–1269.

Agarwal, S., Singh, A.P. and Anand, N. (2013) 'Evaluation performance study of firefly algorithm, particle swarm optimization and artificial bee colony algorithm for non-linear mathematical optimization functions', *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, pp.1–8.

Al-Saati, D. and Alabajee, M. (2016) 'On the performance of firefly algorithm in software reliability modeling', *International Journal of Recent Research and Review*, Vol. 9, No. 4, pp.1–9.

Banati, H. and Bajaj, M. (2013) 'Performance analysis of firefly algorithm for data clustering', *Int. J. of Swarm Intelligence*, Vol. 1, No. 1, pp.19–35.

Chiang, C-W., Lee, W-P. and Heh, J-S. (2010) 'A 2-opt based differential evolution for global optimization', *Applied Soft Computing*, Vol. 10, No. 4, pp.1200–1207, Optimisation Methods & Applications in Decision-Making Processes.

Das, S., Mullick, S.S. and Suganthan, P. (2016) 'Recent advances in differential evolution – an updated survey', *Swarm and Evolutionary Computation*, Vol. 27, No. 4, pp.1–30.

Effati, S. and Nazemi, A.R. (2005) 'A new method for solving a system of the nonlinear equations', *Applied Mathematics and Computation*, Vol. 168, No. 2, pp.877–894.

Feo, T.A. and Resende, M.G.C. (1995) 'Greedy randomized adaptive search procedures', *Journal of Global Optimization*, Vol. 6, No. 2, pp.109–133.

Gong, W., Wang, Y., Cai, Z. and Yang, S. (2017) 'A weighted biobjective transformation technique for locating multiple optimal solutions of nonlinear equation systems', *IEEE Transactions on Evolutionary Computation*, Vol. PP, No. 99, p.1.

Gragg, W.B. and Stewart, G.W. (1976) 'A stable variant of the secant method for solving nonlinear equations', *SIAM Journal on Numerical Analysis*, Vol. 13, No. 6, pp.889–903.

Grosan, C. and Abraham, A. (2007) 'Exploration of multiple roots for a polynomial system', *2007 2nd International Conference on Digital Information Management, ICDIM*, pp.133–137.

Grosan, C. and Abraham, A. (2008a) 'Multiple solutions for a system of nonlinear equations', *International Journal of Innovative Computing, Information and Control*, Vol. 10, No. 10, pp.2161–2170.

Grosan, C. and Abraham, A. (2008b) 'A new approach for solving nonlinear equations systems', *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, Vol. 38, No. 3, pp.698–714.

Hirsch, M.J., Meneses, C.N., Pardalos, P.M. and Resende, M.G.C. (2007) 'Global optimization by continuous grasp', *Optimization Letters*, Vol. 1, No. 2, pp.201–212.

Hirsch, M.J., Pardalos, P.M. and Resende, M.G.C. (2009) 'Solving systems of nonlinear equations with continuous grasp', *Nonlinear Analysis: Real World Applications*, Vol. 10, No. 4, pp.2000–2006.

Hong, H. and Stahl, V. (1994) 'Safe starting regions by fixed points and tightening', *Computing*, Vol. 53, No. 3, pp.323–335.

Hueso, J.L., Martínez, E. and Torregrosa, J.R. (2009) 'Modified Newton's method for systems of nonlinear equations with singular Jacobian', *Journal of Computational and Applied Mathematics*, Vol. 224, No. 1, pp.77–83.

Hui, W. and Zhe-Zhao, Z. (2008) 'A neural-network algorithm for solving systems of nonlinear equations', *2008 International Workshop on Education Technology and Training 2008 International Workshop on Geoscience and Remote Sensing*, Vol. 1, pp.734–737.

Ibrahim, I.A. and Khatib, T. (2017) 'A novel hybrid model for hourly global solar radiation prediction using random forests technique and firefly algorithm', *Energy Conversion and Management*, Vol. 138, pp.413–425.

Jaberipour, M., Khorram, E. and Karimi, B. (2011) 'Particle swarm algorithm for solving systems of nonlinear equations', *Computers and Mathematics with Applications*, Vol. 62, No. 2, pp.566–576.

Kanimozhi, T. and Latha, K. (2015) 'An integrated approach to region based image retrieval using firefly algorithm and support vector machine', *Neurocomputing*, Vol. 151, pp.1099–1111.

Kazemzadeh Azad, S. and Kazemzadeh Azad, S.A. (2011) 'Optimum design of structures using an improved firefly algorithm', *International Journal of Optimization in Civil Engineering*, Vol. 1, No. 2, pp.327–340.

Liao, Z., Gong, W., Yan, X., Wang, L. and Hu, C. (2018) 'Solving nonlinear equations system with dynamic repulsion-based evolutionary algorithms', *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Vol. PP, No. 99, pp.1–12.

Luo, D. and Han, Z. (1995) 'Solving nonlinear equation systems by neural networks', *1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century*, Vol. 1, pp.858–862.

Majd, A., Abdollahi, M., Sahebi, G., Abdollahi, D., Daneshtalab, M., Plosila, J. and Tenhunen, H. (2016) 'Multi-population parallel imperialist competitive algorithm for solving systems of nonlinear equations', *2016 International Conference on High Performance Computing Simulation (HPCS)*, pp.767–775.

Margaris, A. and Adamopoulos, M. (2007) 'Solving nonlinear algebraic systems using artificial neural networks', *Proceedings of the 10th International Conference on Engineering Applications of Artificial Neural Networks*, August, Thessaloniki, Greece.

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

Mathia, K. and Saeks, R. (1995) 'Solving nonlinear equations using recurrent neural networks', *World Congress on Neural Networks*, July, pp.17–21.

Minitab Inc. (2010) *Minitab 17 Statistical Software* [online] http://www.minitab.com.

Mishra, D. and Kalra, P.K. (2007) 'Modified Hopfield neural network approach for solving nonlinear algebraic equations', *Engineering Letters*, Vol. 14, No. 1, pp.135–142.

Moore, R.E. (1980) 'Micro programmed interval arithmetic', *ACM SIGNUM Newsletter*, June, Vol. 15, No. 2, p.30.

Nishani, H.P.S. (2015) *Improved Newton's Method to Solve Systems of Nonlinear Equations*, Master's thesis, Faculty of Graduate Studies, University of Sri Jayawardenapura, Under the supervison of S. Weerakoon, T.G.I. Fernando and M. Liyanage.

Nishani, H.P.S., Weerakoon, S., Fernando, T.G.I. and Liyanage, M. (2018) 'Weerakoon-fernando method with accelerated third-order convergence for systems of nonlinear equations', *International Journal of Mathematical Modelling and Numerical Optimisation*, Vol. 8, No. 3, pp.287–304.

Oliveira, H. and Petraglia, A. (2013) 'Solving nonlinear systems of functional equations with fuzzy adaptive simulated annealing', *Applied Soft Computing*, Vol. 13, No. 11, pp.4349–4357.

Ortega, J. and Rheinboldt, W. (2000) *Iterative Solution of Nonlinear Equations in Several Variables*, Society for Industrial and Applied Mathematics.

Ouyang, A., Zhou, Y. and Luo, Q. (2009) 'Hybrid particle swarm optimization algorithm for solving systems of nonlinear equations', *2009 IEEE International Conference on Granular Computing*, pp.460–465.

Pourjafari, E. and Mojallali, H. (2012) 'Solving nonlinear equations systems with a new approach based on invasive weed optimization algorithm and clustering', *Swarm and Evolutionary Computation*, Vol. 4, No. 3, pp.33–43.

Rajinikanth, V. and Couceiro, M. (2015) 'RGB histogram based color image segmentation using firefly algorithm', *Proceedings of the International Conference on Information and Communication Technologies, ICICT 2014*, 3–5 December 2014, Bolgatty Palace & Island Resort, Kochi, India, *Procedia Computer Science*, Vol. 46, pp.1449–1457.

Ramadas, G.C. and Fernandes, E.M.G.P. (2013) 'Solving systems of nonlinear equations by harmony search', *13th International Conference Computational and Mathematical Methods in Science and Engineering*, Vol. 4, pp.1176–1186. CMMSE 2013.

Ramos, H. and Monteiro, M.T.T. (2017) 'A new approach based on the Newton's method to solve systems of nonlinear equations', *Computational and Mathematical Methods in Science and Engineering CMMSE-2015, Journal of Computational and Applied Mathematics*, Vol. 318, pp.3–13.

Remani, C. (2013) *Numerical Methods for Solving Systems of Nonlinear Equations*, Lakehead University Thunder Bay, Ontario, Canada.

Rheinboldt, W.C. (1998) *Methods for Solving Systems of Nonlinear Equations*, 2nd ed., CBMS-NSF Regional Conference Series in Applied Mathematics, Society for Industrial and Applied Mathematics.

Senthilnath, J., Omkar, S. and Mani, V. (2011) 'Clustering using firefly algorithm: Performance study', *Swarm and Evolutionary Computation*, Vol. 1, No. 3, pp.164–171.

Sharma, J.R., Sharma, R. and Bahl, A. (2016) 'An improved Newton–Traub composition for solving systems of nonlinear equations', *Applied Mathematics and Computation*, Vol. 290, Supplement C, pp.98–110.

Song, W., Wang, Y., Li, H.X. and Cai, Z. (2015) 'Locating multiple optimal solutions of nonlinear equation systems based on multiobjective optimization', *IEEE Transactions on Evolutionary Computation*, Vol. 19, No. 3, pp.414–431.

Tsoulos, I.G. and Stavrakoudis, A. (2010) 'On locating all roots of systems of nonlinear equations inside bounded domain using global optimization methods', *Nonlinear Analysis: Real World Applications*, Vol. 11, No. 4, pp.2465–2471.

Umbarkar, A.J., Balande, U.T. and Seth, P.D. (2017) 'Performance evaluation of firefly algorithm with variation in sorting for non-linear benchmark problems', *AIP Conference Proceedings*, Vol. 1836, No. 1, p.020032.

Wang, X. and Zhou, N. (2014) 'Pattern search firefly algorithm for solving systems of nonlinear equations', *2014 Seventh International Symposium on Computational Intelligence and Design*, Vol. 2, pp.228–231.

Weerakoon, S. and Fernando, T.G.I. (2000) 'A variant of Newton's method with accelerated third-order convergence', *Applied Mathematics Letters*, Vol. 13, No. 8, pp.87–93.

Xiao, L., Shao, W., Liang, T. and Wang, C. (2016) 'A combined model based on multiple seasonal patterns and modified firefly algorithm for electrical load forecasting', *Applied Energy*, Vol. 167, No. C, pp.135–153.

Xiao, X. and Yin, H. (2015) 'A simple and efficient method with high order convergence for solving systems of nonlinear equations', *Computers & Mathematics with Applications*, Vol. 69, No. 10, pp.1220–1231.

Yang, X-S. (2009) 'Firefly algorithms for multimodal optimization', in Watanabe, O. and Zeugmann, T. (Eds.): *Stochastic Algorithms: Foundations and Applications*, pp.169–178, Springer Berlin Heidelberg, Berlin, Heidelberg.

Yang, X-S., Deb, S., Loomes, M. and Karamanoglu, M. (2013) 'A framework for self-tuning optimization algorithm', *Neural Computing and Applications*, Vol. 23, Nos. 7–8, pp.2051–2057.