Annals of Mathematics and Computer Science ISSN: 2789-7206
Vol 20 (2024) 16-24 https://doi.org/10.56947/amcs.v20.226

OPTIMIZING ODE SOLUTIONS: APPLICATION OF NELDER-MEAD ALGORITHM FOR SOLVING MIXED BOUNDARY VALUE PROBLEMS

ASHIRIBO SENAPON WUSU[1] AND OLUSOLA AANU OLABANJO[2*]

Abstract. This work introduces an innovative application of the Nelder-Mead algorithm for optimizing solutions to Ordinary Differential Equations (ODEs) with derivative boundary conditions, emphasizing a computational mathematics perspective. The study presents an advanced methodology harnessing the Nelder-Mead algorithm's adaptability to address ODEs featuring intricate derivative constraints. In the pursuit of enhancing the computational arsenal for tackling complex ODEs, the primary goal of this research is to demonstrate the algorithm's effectiveness in optimizing solutions while satisfying derivative boundary conditions. The methodology entails customizing the Nelder-Mead algorithm to navigate the parameter space while concurrently meeting ODEs and their derivative constraints. Empirical results vividly showcase the algorithm's ability to identify solutions that conform to these stringent criteria, signifying a remarkable progress in addressing ODEs with derivative boundary conditions. The study concludes by underscoring the algorithm's potential to advance ODE-solving techniques, particularly in scenarios where gradient-based methods face challenges, thus expanding its applicability across diverse scientific and engineering domains.

1. Introduction

Ordinary Differential Equations (ODEs) serve as the mathematical backbone for understanding dynamic systems in various scientific disciplines. From physics to biology, ODEs model the evolution of variables over time, revealing intricate relationships between quantities [1, 2]. However, the complexity of many real-world systems often renders analytical solutions unattainable [3, 4, 1, 5, 6]. Solving ODEs analytically often requires simplifying assumptions that might not hold true for real-world scenarios. Numerical methods, such as the Runge-Kutta method, provide a more practical approach, yet they might lack accuracy or efficiency for intricate systems. This is where evolutionary algorithms step in, offering an innovative path to address these challenges. Applying evolutionary algorithms to ODEs introduces a novel way of finding solutions [7, 8, 9]. Instead of

Date: Received: Oct 17, 2023; Accepted: Nov 6, 2023.
* Corresponding author.
2020 *Mathematics Subject Classification.* Primary 65L06; 65L20; Secondary 90C27.
Key words and phrases. Evolutionary, optimization, ordinary differential equations, Nelder-Mead, Algorithm.

directly solving the equations, genetic representations of solutions evolve over generations. These genetic structures undergo genetic operations such as crossover (mixing of solution components) and mutation (randomly changing components), simulating the diversity and adaptation seen in natural evolution. The fitness of each solution is evaluated based on its ability to satisfy the ODEs, and over time, the population converges towards solutions that better match the desired dynamics. The application of Neural Network (NN) for solving differential equations is has gradually gained popularity [10, 11, 12, 13, 14]. Some major advantages of using NN over classical numerical methods are: the solutions obtained by NN are differentiable, and closed analytic form, the method could handle complex differential equations and helps to overcome the repetition of iteration [15]. In [16], a feedforward neural networks was implemented to approximate the solution of linear ODE. ANN was used by authors in [17] for solving ordinary and partial differential equations. Interestingly, the use of evolutionary algorithms with ODEs has emerged as a revolutionary paradigm, leveraging the principles of genetic evolution to unravel the enigmatic behavior of these equations [18, 19, 7]. The classical genetic algorithm was used to obtain approximate solutions of second–order initial value problems in [7]. In [20], approximate solutions of first-order initial value problem was computed by combining collocation method (finite elements) with genetic algorithms. In a later work, [19], the Nelder-Mead algorithm together with the classical genetic algorithm was use to solve the second–order initial value problem of the form $u'' = f(x, y)$. The use of continuous genetic algorithm for solving a two–point second–order ordinary differential equation was proposed in[21]. Differential evolution algorithm has also been used to solve different classes of problems as discussed in [22, 18, 23]. In this paper, we consider the second-order ordinary differential equation with derivative boundary conditions

$$u'' = f(t, u, u'), \quad a_0u(a) - a_1u'(a) = \gamma_1,\ b_0u(b) + b_1u'(b) = \gamma_2,\ t \in (a, b) \tag{1.1}$$

where $a_0, b_0, a_1, b_1, \gamma_1$ and γ_2 are constants such that

$$\begin{aligned} a_0a_1 \geq 0, \quad & |a_0| + |a_1| \neq 0 \\ b_0b_1 \geq 0, \quad & |b_0| + |b_1| \neq 0 \text{ and } |a_0| + |b_0| \neq 0. \end{aligned} \tag{1.2}$$

We demonstrate that the Nelder-Mead algorithm can be effectively employed to acquire an approximate solution to the given problem, (1.1), through the assumption of a trial solution, denoted as $u(t)$. The algorithm adeptly refines the coefficients of the trial solution by minimizing the corresponding mean square error, thus enabling the derivation of an optimized solution that aligns with the problem's underlying dynamics.

2. Nelder-Mead Algorithm: An overview

The Nelder-Mead algorithm, originally designed for unconstrained optimization tasks, has been creatively adapted to address the complex realm of Ordinary Differential Equations (ODEs). This adaptation involves tailoring the algorithm's principles to iteratively refine solutions within the parameter space, allowing it to navigate the intricate landscapes defined by the ODEs. This overview outlines

the key steps and considerations involved in adapting the Nelder-Mead algorithm for solving ODEs.

(1) Initialization: Let the initial simplex be defined by vertices $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n$ in an n-dimensional space. The function $f(\mathbf{x})$ is to be optimized.
(2) Ordering the Vertices: Evaluate f at each vertex and sort the vertices in ascending order of function values: $f(\mathbf{x}_0) \leq f(\mathbf{x}_1) \leq \ldots \leq f(\mathbf{x}_n)$.
(3) Reflection: Calculate the centroid $\mathbf{c}$ of the first n vertices:

$$\mathbf{c} = \frac{1}{n} \sum_{i=0}^{n-1} \mathbf{x}_i$$

Reflect the worst vertex $\mathbf{x}_n$ across the centroid to obtain the reflection $\mathbf{x}_r$:

$$\mathbf{x}_r = \mathbf{c} + \alpha(\mathbf{c} - \mathbf{x}_n)$$

where $\alpha > 0$ is a reflection coefficient.
(4) Evaluation and Replacement: Evaluate $f(\mathbf{x}_r)$. If $f(\mathbf{x}_0) \leq f(\mathbf{x}_r) < f(\mathbf{x}_{n-1})$, replace $\mathbf{x}_n$ with $\mathbf{x}_r$ and proceed to the next iteration.
(5) Expansion: If $f(\mathbf{x}_r) < f(\mathbf{x}_0)$, calculate the expansion $\mathbf{x}_e$ by extending beyond the reflection point:

$$\mathbf{x}_e = \mathbf{c} + \gamma(\mathbf{x}_r - \mathbf{c})$$

where $\gamma > 1$ is an expansion coefficient.

Evaluate $f(\mathbf{x}_e)$. If $f(\mathbf{x}_e) < f(\mathbf{x}_r)$, replace $\mathbf{x}_n$ with $\mathbf{x}_e$ and proceed to the next iteration.
(6) Contraction: If $f(\mathbf{x}_r) \geq f(\mathbf{x}_{n-1})$, perform a contraction to obtain $\mathbf{x}_c$:

$$\mathbf{x}_c = \mathbf{c} + \beta(\mathbf{x}_n - \mathbf{c})$$

where $0 < \beta < 1$ is a contraction coefficient.

Evaluate $f(\mathbf{x}_c)$. If $f(\mathbf{x}_c) < f(\mathbf{x}_n)$, replace $\mathbf{x}_n$ with $\mathbf{x}_c$ and proceed to the next iteration.
(7) Shrinkage: If none of the above steps result in replacement, perform a shrinkage operation by moving all vertices except $\mathbf{x}_0$ towards $\mathbf{x}_0$:

$$\mathbf{x}_i = \mathbf{x}_0 + \sigma(\mathbf{x}_i - \mathbf{x}_0)$$

where $0 < \sigma < 1$ is a shrinkage coefficient.
(8) Convergence Check: Check for a convergence criterion, such as the difference in function values or the size of the simplex. If the criterion is met, terminate the algorithm.
(9) Termination: If the maximum number of iterations is reached, terminate the algorithm.
(10) Output: Return the vertex $\mathbf{x}_0$ with the lowest function value as the optimal solution.

The adaptation of the Nelder-Mead algorithm for solving ODEs requires careful consideration of the fitness evaluation process, which measures the solution's ability to satisfy the ODEs' derivatives accurately. Furthermore, as ODEs may exhibit varying dynamics across the parameter space, the algorithm's performance

can benefit from techniques like adaptive parameter tuning and the incorporation of heuristics to guide exploration.

3. Proposed Technique

Consider the boundary value problem (1.1), we assume a trial solution of the form

$$u_{trial}(t) = \sum_{i=0}^{k} \psi_i t^i + \sum_{j=0}^{m} \alpha_j \exp(\omega_j t), \quad k, m \in \mathbb{Z}^+ \tag{3.1}$$

where k, m, ψ_i, α_j and ω_j are to be determined. Substituting (3.1) and its derivatives into (1.1) gives

$$\sum_{i=2}^{k} i(i-1)\psi_i t^{i-2} + \sum_{j=1}^{m} \alpha_j \omega_j^2 e^{t\omega_j} = f\left(t, u_{trial}, u'_{trial}\right) \tag{3.2}$$

Using the boundary conditions we have the constraint as

$$\left[a_0 \left(\sum_{i=0}^{k} \psi_i t^i + \sum_{j=1}^{m} \alpha_j e^{t\omega_j}\right) - a_1 \left(\sum_{i=1}^{k} i\psi_i t^{i-1} + \sum_{j=1}^{m} \alpha_j \omega_j e^{t\omega_j}\right)\right]_{t=a} = \gamma_1 \tag{3.3}$$

$$\left[b_0 \left(\sum_{i=0}^{k} \psi_i t^i + \sum_{j=1}^{m} \alpha_j e^{t\omega_j}\right) + b_1 \left(\sum_{i=1}^{k} i\psi_i t^{i-1} + \sum_{j=1}^{m} \alpha_j \omega_j e^{t\omega_j}\right)\right]_{t=b} = \gamma_2 \tag{3.4}$$

At each node point t_n, we require that

$$\mathcal{E}_n(t) = \left[\sum_{i=2}^{k} i(i-1)\psi_i t^{i-2} + \sum_{j=1}^{m} \alpha_j \omega_j^2 e^{t\omega_j} - f\left(t, u_{trial}, u'_{trial}\right)\right]_{t=t_n} = 0 \tag{3.5}$$

Now, to obtain the expression for the trial solution $u_{trial}(t)$, we need to find the set $\{k, m, \psi_i, \alpha_j,\ \omega_j | i = 0, 1, \cdots, k; j = 1, 2, \cdots, m\}$, which minimizes the sum of square of error, $\mathcal{E}_n(t)$ at each node point given by the expression

$$\sum_{n=1}^{N} \mathcal{E}_n^2(t) \tag{3.6}$$

where $N = \frac{b-a}{h}$ and h is the steplength. The problem of solving (1.1) is now formulated as an optimization problem in the following manner:

$$\text{Minimize:} \quad \sum_{n=1}^{N} \mathcal{E}_n^2(t), \tag{3.7}$$

$$\text{Subject to:} \quad \left[a_0 \left(\sum_{i=0}^{k} \psi_i t^i + \sum_{j=1}^{m} \alpha_j e^{t\omega_j}\right) - a_1 \left(\sum_{i=1}^{k} i\psi_i t^{i-1} + \sum_{j=1}^{m} \alpha_j \omega_j e^{t\omega_j}\right)\right]_{t=a} = \gamma_1 \tag{3.8}$$

$$\left[b_0 \left(\sum_{i=0}^{k} \psi_i t^i + \sum_{j=1}^{m} \alpha_j e^{t\omega_j}\right) + b_1 \left(\sum_{i=1}^{k} i\psi_i t^{i-1} + \sum_{j=1}^{m} \alpha_j \omega_j e^{t\omega_j}\right)\right]_{t=b} = \gamma_2 \tag{3.9}$$

We now implement the Nelder-Mead algorithm to the optimal values of the parameters $\{k, m, \psi_i, \alpha_j,\ \omega_j | i = 0, 1, \cdots, k; j = 1, 2, \cdots, m\}$ of the trial solution $u_{trial}(t)$ that minimizes $\sum_{n=1}^{N} \mathcal{E}_n^2(t)$.

4. Numerical Results

In order to demonstrate how the technique works, we consider two test cases. The error quantity in case is calculated using (3.6). For both cases considered, we present the comparison the results obtained via the Nelder-Mead algorithm with the exact solution. Also, the *"CPU-time"* was measured and presented. The following parameters were used in the implementation:

Table 1. Values of parameters used during implementation

Parameter	Value
Cross Ratio	0.5
Expansion Ratio	2.0
Initial Point	Automatic
Penalty Function	Automatic
Post Process	Automatic
Random Seed	0
Shrink Ratio	0.5
Tolerance	0.000001

4.1. **Problem 1.** Consider the boundary value problem

$$u''(t) = u(t) - 4te^t; \quad u(0) - u'(0) = -1,\ u(1) + u'(1) = -e, \quad t \in (0, 1) \tag{4.1}$$

Here the exact solution is $u(t) = t(1-t)e^t$. Using the proposed algorithm, optimal values of k and m were obtained as $k = 5$ and $m = 1$. The resulting optimal coefficient values: $k, m, \psi_i, \alpha_j,\ \omega_j | i = 0, 1, \cdots, k; j = 1, 2, \cdots, m$ are given in Table 2.

Table 2. Optimal coefficients of (3.1) obtained for Problem 1.

Coefficient	Value
ψ_0	$-\frac{64929476856370457089063615166 6}{22649615147540844900828894153 15}$
ψ_1	$\frac{10305298102657780235816471822 01}{98298156458212834235664116537 2}$
ψ_2	$\frac{42425542399661832233984560421}{16476588911867344459747642300179}$
ψ_3	$-\frac{64775956242986659379045119462 9}{11231496108964512537282450988 30}$
ψ_4	$-\frac{19287957998404516984552869163 1}{12877395882823468527942528524 33}$
ψ_5	$-\frac{49589947611754285020072135316 1}{17669272356660971296812487812 95}$
α_1	$\frac{46522504853802195358872259166 5}{16174632563513812600531328605 03}$
ω_1	$-\frac{30099488629329085025356660735 3}{18259048499861335424221413250 51}$

The calculated error quantity (3.6), was found to be $\frac{6277307014787950989141007178 6}{4001446217799484321361259889688 5}$ and it took the algorithm approximately 5.515625 seconds to run but this metric can be significantly influenced by the problem under consideration. The results of the Nelder-Mead approximation is given in Figure 1, where the trial solution is plotted against the exact solution. From Figure 1, we can see that the Nelder-Mead algorithm almost produced exact solution.

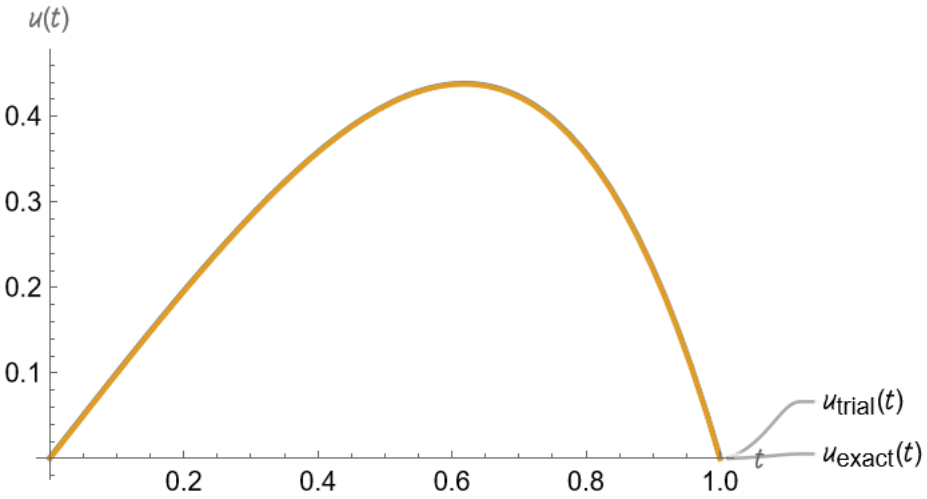

FIGURE 1. Nelder-Mead approximation for Example 1.

4.2. **Problem 2.** The second problem considered here is given as

$$u''(t) - 3u'(t) + 2u(t) = 2; \quad u(0) - u'(0) = -1,\ u(1) + u'(1) = 1, \quad t \in (0, 1) \tag{4.2}$$

The exact solution is obtained as $u(t) = 2e^{2t} - 3e^{t+1} + 1$. In this case, the algorithm gave the optimal values of k and m as $k = 3$ and $m = 2$ and the resulting optimal coefficient values: $k, m, \psi_i, \alpha_j,\ \omega_j | i = 0, 1, \cdots, k; j = 1, 2, \cdots, m$ are given in Table 3.

TABLE 3. Optimal coefficients of (3.1) obtained for Problem 2.

Coefficient	Value
ψ_0	$-\frac{154701861248903015966310614139 1}{429379339103792913481161281440}$
ψ_1	$-\frac{168944685474079975630837703830 1}{424554723091488645520729637002}$
ψ_2	$-\frac{103230019052309433444456316387 7}{662385363800573693414462090147}$
ψ_3	$-\frac{115968828401369345425792556384 9}{346656790121461922215458989377 2}$,
α_1	$-\frac{961629711072919715773445140985}{310233760093763690185937168391}$
ω_1	$\frac{209367053339517781550421041423 0}{191546410854195950637920906317 9}$
α_2	$\frac{147709909644747703589014101943 9}{953923199065104519937994039815}$
ω_1	$\frac{167644256078123174766227229541 9}{807868289645391975010730692251}$

For this problem, the calculated error quantity (3.6), was found to be $\frac{763117917355682243226739968 8}{216169182733027888206138480969479}$ and the algorithm took approximately 11.015625 seconds to run. Figure 2 shows the plots of our approximate solution and the exact solution. Again our algorithm almost produced the exact solution.

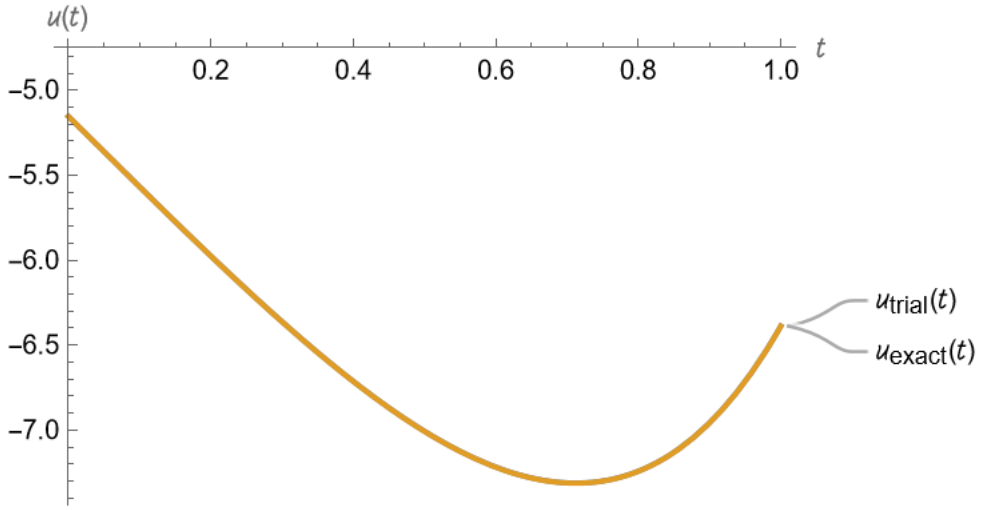

FIGURE 2. Nelder-Mead approximation for Example 2.

5. Conclusion

In this study, we have demonstrated the potent synergy between the Nelder-Mead algorithm and Ordinary Differential Equations (ODEs) with derivative boundary conditions. By harnessing the algorithm's ability to iteratively refine solutions, we ventured into a realm where gradient-based methods often falter. Our investigation unveiled a promising avenue for optimizing ODE solutions in scenarios characterized by intricate derivative constraints. Through the application of the Nelder-Mead algorithm, we showcased its efficacy in addressing ODE problems where derivative boundary conditions play a pivotal role. The algorithm, originally tailored for unconstrained optimization tasks, proved itself adaptable to the realm of ODEs, contributing to the optimization landscape. The results obtained through two illustrative examples underscored the algorithm's prowess. It efficiently navigated the parameter space while respecting the derivative boundary conditions, leading to accurate and meaningful solutions. In both cases, the Nelder-Mead algorithm showcased its versatility and ability to enhance our approach to solving ODEs with complex constraints. As we conclude, our study sheds light on a novel method for tackling ODEs with derivative boundary conditions. The Nelder-Mead algorithm, with its adaptability and effectiveness, opens avenues for optimizing solutions that align with real-world phenomena. This research marks a significant step forward in the optimization of ODE solutions, contributing to the advancement of mathematical techniques in diverse scientific and engineering domains. With a promising trajectory, the combination of the Nelder-Mead algorithm and ODEs paves the way for innovative avenues in

optimization and problem-solving.

Acknowledgement. We would like to extend our sincere appreciation to the diligent efforts of the journal editor for their invaluable contributions to the review and publication process of our research manuscript. Their insightful guidance, attention to detail, and constructive feedback have been instrumental in refining the content and overall quality of the paper. Their commitment to upholding the standards of excellence within the field is truly commendable, and we grateful for their dedication and support throughout the publication journey.

References

1. J. Butcher, *Numerical Methods for Ordinary Differential Equations*. Wiley, (2008). https://doi.org/10.1002/9780470753767
2. J.D. Lambert, *Computational methods in ordinary differential equations*. Wiley, New York, (1973).
3. J.D. Lambert, *Numerical methods for ordinary differential systems* Wiley, New York, (1991).
4. A. Huta, *Contribution a la formule de sixieme ordre dan la methode de Runge–Kutta–Nystrom*. Acta Fac. Rerum Natur. Univ. Comenian., Math. **2** (1957), 21-24.
5. A.S. Wusu and M.A. Akanbi, *A Three-Stage Multiderivative Explicit Runge-Kutta Method*. American Journal of Computational Mathematics **3** (2013), 121-126. https://doi.org/10.4236/ajcm.2013.32020
6. J. Cash, *On the integration of stiff systems of ODEs using extended backward differentiation formulae*. Numerische Mathematik, **34** (1980), 235-246. https://doi.org/10.1007/BF01396701
7. D. George, *On the appliaction of genetic algorithms to differential equations*. Romanian Journal of Economic Forecasting, **3** (2006), 5-9.
8. R.G. Nascimento, K. Fricke and F.A. Viana, *A tutorial on solving ordinary differential equations using Python and hybrid physics-informed neural network*. Engineering Applications of Artificial Intelligence. **96** (2020), 103996. https://doi.org/10.1016/j.engappai.2020.103996
9. E.A. Hussain and Y.M. Abdul-Abbass, *Solving Differential Equation by Modified Genetic Algorithms*. Journal of University of Babylon for Pure and Applied Sciences, **26** (2018), 233-241. https://doi.org/10.29196/jubpas.v26i10.1877
10. A. Junaid, M. Raja and I. Qureshi, *Evolutionary computing approach for the solution of initial value problems in ordinary differential equations*. World Academy of Science, Engineering and Technology, **55** (2009), 578-581.
11. L.S. Tan, Z. Zainuddin and P. Ong, *Solving ordinary differential equations using neural networks*. In Proceedings of the AIP Conference Proceedings. AIP Publishing, **19** (2018).
12. S.N. Zisad, M.S. Hossain, M.S. Hossain and K. Andersson, *An integrated neural network and SEIR model to predict Covid-19*. Algorithms, **14** (2021), 94. https://doi.org/10.3390/a14030094
13. M. Chiaramonte and M. Kiener, *Solving differential equations using neural networks*. Machine Learning Project, **1** (2013).
14. F. Chen, D. Sondak, P. Protopapas, M.Mattheakis, S. Liu, D. Agarwal and M. Di Giovanni, *A python package for solving differential equations with neural networks*. Journal of Open Source Software, **5** (2020), 1931. https://doi.org/10.21105/joss.01931
15. S. Mall and S.Chakraverty, *Single layer Chebyshev neural network model for solving elliptic partial differential equations*. Neural Processing Letters, **45** (2017), 825-840. https://doi.org/10.1007/s11063-016-9551-9

16. A.J. Meade Jr and A.A. Fernandez, *The numerical solution of linear ordinary differential equations by feedforward neural networks.* Mathematical and Computer Modelling, **19** (1994), 1-25. https://doi.org/10.1016/0895-7177(94)90095-7
17. I.E. Lagaris, A. Likas and D.I. Fotiadis, *Artificial neural networks for solving ordinary and partial differential equations.* IEEE transactions on neural networks, **9** (1998), 987-1000. https://doi.org/10.1109/72.712178
18. A.S. Wusu and M.A. Akanbi, *Solving oscillatory/periodic ordinary differential equations with differential evolution algorithms.* Communications in Optimization Theory, (2016), 1-8.
19. N.E. Mastorakis, *Unstable ordinary differential equations: solution via genetic algorithms and the method of Nelder-Mead.* WSEAS Transactions on Mathematics, **5** (2006), 1276-1280.
20. N.E. Mastorakis, *Numerical solution of non-linear ordinary differential equations via collocation method (finite elements) and genetic algorithms.* WSEAS Transactions on Information Science and Applications, **2** (2005), 467-473.
21. O.A. Arqub, Z. Abo-Hammour, S. Momani and N. Shawagfeh, *Solving singular two-point boundary value problems using continuous genetic algorithm.* In Proceedings of the Abstract and applied analysis. Hindawi, **2012** (2012). https://doi.org/10.1155/2012/205391
22. A.S. Wusu and M.A. Akanbi and B.O. Fatimah, *On the Derivation and Implementation of a Four Stage Harmonic Explicit Runge-Kutta Method.* Applied Mathematics, **6** (2015), 694-699. https://doi.org/10.4236/am.2015.64064
23. A.S. Wusu and O.A. Olabanjo and M. Manuel, *Approximate Analytical Solution of Unstable Ordinary Differential Equation Using Differential Evolution Algorithm.* International Journal of Scientific Advances, **3** (2022), 654-658. https://doi.org/10.51542/ijscia.v3i4.35

[1] Department of Mathematics, Lagos State University, Lagos, Nigeria.
Email address: ashiribo.wusu@lasu.edu.ng

[2] Department of Mathematics, Morgan State University, Baltimore, USA.
Email address: olola57@morgan.edu