

RESEARCH ARTICLE

# A novel ninth-order root-finding algorithm for nonlinear equations with implementations in various software tools

Srinivasarao Thota<sup>1</sup>, Amir Naseem<sup>2</sup>, Thumati Gopi<sup>3</sup>, Kashireddy Sai NandanReddy<sup>3</sup>, Padarthi Sai Kousik<sup>3</sup>, Thulasi Bikku<sup>3</sup>, and Shanmugasundaram Palanisamy<sup>4\*</sup>

<sup>1</sup>Department of Mathematics, Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Amaravati, Andhra Pradesh, India

<sup>2</sup>Department of Mathematics, University of Management and Technology, Lahore, Punjab, Pakistan

<sup>3</sup>Department of Computer Science and Engineering, Amrita School of Computing, Amrita Vishwa Vidyapeetham, Amaravati, Andhra Pradesh, India

<sup>4</sup>Department of Mathematics, College of Natural Computational Sciences, MizanTepi University, Mizan-Aman, South West Ethiopia Peoples' Region, Ethiopia

*t.srinivasarao@av.amrita.edu, amir.kasuri89@gmail.com, av.en.u4cse22043@av.students.amrita.edu, av.en.u4cse22020@av.students.amrita.edu, av.en.u4cse22037@av.students.amrita.edu, b.thulasi@av.amrita.edu, psserode@mtu.edu.et*

ARTICLE INFO

ABSTRACT

Article History:

Received: December 24, 2024

1st revised: March 19, 2025

2nd revised: April 25, 2025

3rd revised: May 9, 2025

Accepted: May 14, 2025

Published Online: June 19, 2025

Keywords:

Root-finding algorithm

Nonlinear equations

Implementations

Halley method

Exponential method

AMS Classification:

65H05, 65Hxx

Nonlinear phenomena are prevalent in numerous fields, including economics, engineering, and natural sciences. Computational science continues to advance through the development of novel numerical schemes and the refinement of existing ones. Ideally, these numerical systems should offer both high-order convergence and computational efficiency. This article introduces a new three-step algorithm for solving nonlinear scalar equations, aiming to meet these criteria. The proposed approach requires six function evaluations per iteration and achieves ninth-order convergence. To demonstrate the efficiency of the technique, various numerical examples are shown. Implementations of the method are available in both Maple and Python, and it can be readily adapted for use in other computational environments.



## 1. Introduction

Many scientific and engineering problems require nonlinear root-finding techniques for their solutions. The field of computational science is continually evolving, with ongoing development of numerical methods that offer higher-order convergence and more efficient numerical schemes. This has led to the establishment of a new ninth-order nonlinear root-finding technique aimed at improving the solution of nonlinear scalar equations. Another advantage of this method is its high converging power, which means fewer function evaluations are needed per iteration, thereby making

it computationally efficient. This novel approach has the potential to be a significant breakthrough in computational science, offering accurate solutions to nonlinear problems for various areas of knowledge.

The Newton–Raphson (NR) method is one of the most widely used approaches for solving nonlinear equations, mainly because of its simplicity. As a foundational method, it serves as a basis for the development of more sophisticated numerical techniques. However, achieving an optimal rate of convergence, as conjectured by Kung–Traub, Traub<sup>1,2</sup> theorem may not always

\*Corresponding author

be possible. Consequently, researchers have focused on developing algorithms with better convergence rates while minimizing the number of function evaluations per iteration. The efficiency index  $E = p^{1/c}$  introduced as a critical measure, estimates the convergence order  $p$  relative to the number of function evaluations per step necessary for a numerical scheme. The aim is to maximize this index as larger values indicate improved efficiency. Some notable recently developed nonlinear methods include the homotopy perturbation approach,<sup>3-5</sup> the Adomian decomposition method,<sup>6</sup> the variational iteration method,<sup>7,8</sup> and methods based on quadrature rules.<sup>9,10</sup>

This paper is based on prior work and designs a new three-step numerical method with ninth-order convergence. This method combines an exponential series-based scheme (ExpM)<sup>11-14</sup> with the classical third-order Halley method (HM)<sup>12,15-18</sup> to achieve higher performance measures than existing methods. Specifically, this method minimizes the number of function evaluations while maintaining high accuracy and computational efficiency. Noor et al.<sup>19</sup> developed a novel two-step scheme with order  $p_1 p_2$  by combining two schemes of orders  $p_1$  and  $p_2$ . At each iteration, the process adds new function evaluations, but these have a higher order of convergence. However, Noor et al.<sup>20</sup> presented two schemes with third- and fourth-order convergence that required five and eight function evaluations per iteration, respectively. Shah et al.<sup>21</sup> developed a three-step scheme with third-order convergence at the cost of five function evaluations per iteration. Abro and Shaikh<sup>21</sup> developed a three-step sixth-order convergent technique employing seven function evaluations per iteration. Even though these methods converge at high orders, they incur extra computational costs in terms of central processing unit time. A similar method was applied in a recently published work<sup>23</sup> to produce an effective three-step numerical technique. The primary goal of this study is to design a novel hybrid three-step numerical approach for solving scalar and vector nonlinear equations. Inspired by these methods, we combined the classical third-order Halley scheme<sup>24</sup> with a modified third-order Newton scheme<sup>25</sup> to create a novel three-step numerical scheme. This led to the development of a ninth-order method that requires just six function evaluations per iteration: three functions, two first-order derivatives, and one second-order derivative. Several advantages are associated with the proposed scheme over previous methods. It provides improved accuracy such that fewer iterations are necessary to reach a defined accuracy

threshold in less computational time. Furthermore, it is more efficient than other methods in terms of its efficiency index, demonstrating that it can effectively solve scalar and vector nonlinear equations. This paper is organized as follows: Section 2 presents an overview of various existing schemes, along with the formulation and convergence analysis for the proposed ninth-order algorithm. Theoretical comparisons and stability assessments are also discussed. Section 3 explores real-life applications and demonstrates the effectiveness of the proposed scheme through numerical experiments. Finally, Section 4 discusses key findings and highlights areas for future research development. Numerous hybrid root-finding algorithms, using different techniques, are available in the literature<sup>26-37</sup> and references therein.

In this article, we present a new three-step iterative method with a single parameter and memory. It is designed in a modular way, making it easier to use for solving nonlinear equations efficiently.

### 1.1. Previous materials and methods

We reviewed various numerical techniques applied to nonlinear models. Given the difficulty in obtaining exact analytical solutions for such models, we employed numerical methods that offer approximations. The classical NR method<sup>1,2</sup> is typically the first to come to mind when solving nonlinear equations numerically. NR has a second-order convergence, which is considered optimal. In simple terms, this entails one evaluation per iteration for both the function and its first derivative:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 0, 1, 2, \dots \quad (1)$$

Halley's third-order numerical scheme, Halley3, is a precise method<sup>16,24</sup> that calls for three different function evaluations per iteration as follows: the function itself, its first derivative, and its second-order derivative.

$$x_{n+1} = x_n - \frac{2f(x_n)f'(x_n)}{2f'(x_n)^2 - f(x_n)f''(x_n)}, \quad n = 0, 1, 2, \dots \quad (2)$$

In a recent study, a pioneering numerical scheme called the modified Halley method (MHM) was presented. As an interpolative technique, it combines NR and HM as a correction, while also employing a finite difference approximation for estimating the second derivative. Remarkably, MHM<sup>17</sup> achieves fifth-order convergence and requires four function evaluations per

iteration, which consist of two function evaluations and two first-order derivatives.

$$y_n = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 0, 1, 2, \dots \quad (3)$$

$$x_{n+1} = y_n - \frac{2f(x_n)f(y_n)f'(y_n)}{2f(x_n)f'(y_n)^2 - f'(x_n)^2f(y_n) + f'(x_n)f(y_n)f'(y_n)} \quad (4)$$

Another new method proposed was based on ExpM,<sup>13</sup> which provides faster root convergence compared to existing algorithms. The proposed algorithm ExpM is particularly useful for computing the real roots of transcendental equations (13).

$$x_{n+1} = x_n \exp\left(\frac{-f(x_n)}{x_n f'(x_n)}\right), \quad n = 0, 1, 2, \dots \quad (5)$$

The following section presents the proposed algorithm for solving nonlinear equations.

## 2. Proposed algorithm

Through an extensive literature review, we identified several researchers who developed the ExpM method to improve convergence order and reduce the number of function evaluations. Nevertheless, our approach advanced beyond this by adopting the HM<sup>16</sup> as a starting point:

$$x_{n+1} = x_n - \frac{2f(x_n)f'(x_n)}{2f'(x_n)^2 - f(x_n)f''(x_n)}, \quad n = 0, 1, 2, \dots \quad (6)$$

This was combined with the ExpM method Equation (5), described as follows:

$$s_n = x_n \exp\left(\frac{-f(x_n)}{x_n f'(x_n)}\right), \quad n = 0, 1, 2, \dots, \quad (7)$$

$$x_{n+1} = x_n - \frac{f(x_n) + f(s_n)}{f'(x_n)}. \quad (8)$$

The aim was to develop a nonlinear scheme that is innovative and superior in terms of convergence order, while requiring fewer function evaluations per iteration. Hence, by combining Equations (6)–(8), a new nonlinear scheme was obtained, defined as follows for  $n = 0, 1, 2, \dots$ :

$$\begin{aligned} t_n &= x_n - \frac{2f(x_n)f'(x_n)}{2f'(x_n)^2 - f(x_n)f''(x_n)}, \\ s_n &= t_n \exp\left(\frac{-f(t_n)}{t_n f'(t_n)}\right), \\ x_{n+1} &= t_n - \frac{f(t_n) + f(s_n)}{f'(t_n)}. \end{aligned} \quad (9)$$

Although the proposed method shares structural similarities with the scheme introduced by

Kalantari and Hans Lee,<sup>38</sup> the modifications incorporated into our algorithm are both intentional and significant. Specifically, the correction step of the proposed method employed an exponential-based refinement, a distinctive enhancement absent in Kalantari and Hans Lee.<sup>38</sup> This refinement modifies the intermediate estimate using an exponential transformation of the residual, which contributes to a broader region of attraction, improved numerical stability, and reduced sensitivity to initial guesses. These advantages are particularly important for nonlinear problems characterized by steep gradients or closely spaced roots. The effectiveness of this modification was demonstrated through detailed polynomiographic visualizations, semi-local convergence analysis, and its application to a real-world engineering problem involving open-channel flow. Collectively, these results highlight that the novelty of the proposed method lies not only in preserving the high order of convergence and computational efficiency but also in delivering enhanced robustness, accuracy, and stability in practical scenarios.

The pseudocode for the proposed algorithm is as shown below.

Input:

```
f(x) → Nonlinear function
f'(x) → First derivative of f(x)
f''(x) → Second derivative of f(x)
x0 → Initial approximation
maxIter → Maximum number
of iterations
```

Output:

```
x_root → Approximate root of f(x)
```

Begin:

```
Set x ← x0
Set iter ← 0
While iter < maxIter:
    t = x - (2*f(x)*f'(x)) /
        (2*(f'(x))^2 - f(x)*f''(x))
    s = t * exp(-f(t) / (t * f'(t)))
    x_new = t - (f(t) + f(s)) / f'(t)
    Update x ← x_new
    Increment iter ← iter + 1
If iter == maxIter:
    Print "Max iterations reached."
End
```

The convergence of the proposed root-finding algorithm can be analyzed using both local and semi-local techniques. This ensures that the method remains robust and reliable under various conditions.

### 2.1. Local convergence analysis

Local convergence analysis examines how the iterative method behaves when the initial approximation is sufficiently close to the root  $\alpha$ . The aim was to determine the order of convergence by deriving an error equation.

Let  $\alpha$  be a simple root of  $f(x)$ , i.e.,  $f(\alpha) = 0, f'(\alpha) \neq 0$ . Expanding  $f(x)$  in a Taylor series around  $\alpha$ , we obtain:

$$f(x) = f'(\alpha)(x-\alpha) + \frac{f''(\alpha)}{2}(x-\alpha)^2 + O((x-\alpha)^3)$$

Similarly, the derivatives are:

$$\begin{aligned} f'(x) &= f'(\alpha) + f''(\alpha)(x-\alpha) + O((x-\alpha)^2), \\ f''(x) &= f''(\alpha) + O(x-\alpha). \end{aligned}$$

The error at the  $n$ th iteration is defined as  $e_n = x_n - \alpha$ . The following theorem guarantees that the nonlinear three-step scheme presented in Equation (9) for solving  $f(x) = 0$  has ninth-order accuracy.

**Theorem 1.** *Let  $\alpha$  be the root of a sufficiently differentiable function  $f : \theta \rightarrow R$ , where  $\theta \subseteq R$  is an open interval. Then, the three-step scheme presented in Equation (9) has ninth-order convergence.*

**Proof.** The proof of this theorem is similar to that of proof of Theorem 1 in Kalantari and Hans Lee.<sup>38</sup> For simplicity, we include the main steps of the proof.

Let  $\alpha$  be the root of  $f(x) = 0$ ,  $x_j$  be the  $j$ th approximation to the root provided by Equation (IX), and

$$e_j = x_j - \alpha \tag{10}$$

be the error term after the  $j$ th iteration. Using Taylor's expansion for  $f(x_j)$  about  $\alpha$ , we obtain:

$$f(x_j) = f'(\alpha)e_j + \frac{1}{2}f''(\alpha)e_j^2 + \varphi(e_j^3). \tag{11}$$

Here,  $\varphi(e_j^3)$  describes the asymptotic behavior of a function Equation (10) as  $e_n \rightarrow 0$ , particularly focusing on the dominant order of smallness. It represents an upper bound on the rate at which the function shrinks near the root. Expanding  $f(x_j)$  using Taylor's series about  $\alpha$ , we also have:

$$f'(x_j) = f'(\alpha) + f''(\alpha)e_j + \frac{1}{2}f'''(\alpha)e_j^2 + \varphi(e_j^3) \tag{12}$$

Multiplying Equations (11) and (12), we get:

$$\begin{aligned} f(x_j) f'(x_j) &= f'^2(\alpha) e_j + \frac{3}{2} f'(\alpha) f''(\alpha) e_j^2 \\ &+ \frac{1}{2} \left( f'(\alpha) f'''(\alpha) + f''^2(\alpha) \right) e_j^3 + \varphi(e_j^4). \end{aligned}$$

Now, the Taylor expansion of the denominator in the fractional term in the first step of Equation (9), multiplied by its numerator, results in:

$$\begin{aligned} \frac{2f(x_j) f'(x_j)}{2f'(x_j)^2 - f(x_j) f''(x_j)} &= e_j - \frac{1}{4f'^2(\alpha)} f''^2(\alpha) e_j^3 \\ &- \frac{1}{4f'^3(\alpha)} f''(\alpha) \left( 5f'(\alpha) f'''(\alpha) - 6f''^2(\alpha) \right) e_j^4 \\ &+ \varphi(e_j^5), \end{aligned}$$

Following the proof of Theorem 1 in Kalantari and Hans Lee,<sup>38</sup> we obtain:

$$e_{j+1} = c \left( \frac{f''(\alpha)}{f'(\alpha)} \right)^8 e_j^9 + \varphi(e_j^{10}),$$

where  $c$  is a constant. Hence, the proposed algorithm has ninth-order convergence.  $\square$

**Theorem 2.** *Let  $f : I \rightarrow R$ . Suppose  $\alpha \in I$  is a simple root of  $f(x) = 0$  and  $\delta$  is a sufficiently small neighborhood of  $\alpha$ . Let  $f''(x)$  exists and  $f''(x) \neq 0$  in  $\delta$ . Then the iterative Equation (9) produces a sequence of iterations  $\{x_n : n = 0, 1, 2, \dots\}$  with the order of convergence nine*

**Proof.** Let

$$R(x) = t - \frac{f(t) + f(s)}{f'(t)},$$

where

$$s = t \exp\left(\frac{-f(t)}{tf'(t)}\right) \text{ and } t = x - \frac{2f(x)f'(x)}{2f'(x)^2 - f(x)f''(x)}.$$

As in Abbasbandy,<sup>5</sup> we observed that:

$$R(\alpha) = \alpha, R'(\alpha) = 0, \dots, R^{(8)}(\alpha) = 0, R^{(9)}(\alpha) \neq 0.$$

Hence, the proposed algorithm has convergence of order nine.  $\square$

### 2.2. Semi-local convergence analysis

Semi-local analysis examines the behavior of the algorithm when the initial guess is not arbitrarily close to the root. It ensures that the algorithm converges for a broader class of functions.

Kantorovich-type conditions: To ensure semi-local convergence, we verified whether the function satisfies the following:

- (i) The function  $f(x)$  is continuously differentiable in a neighborhood around the root,

and the first and second derivatives are well-behaved. That is, there exists a constant  $L$  such that  $|f''(x)| \leq L|f'(x)|$ .

- (ii) The initial guess  $x_0$  must satisfy  $|f(x_0)f'(x_0)| < \frac{2}{L}$ , ensuring that the error does not grow uncontrollably.

Sensitivity to initial conditions: although a higher order of convergence reduces sensitivity to initial conditions, the algorithm may still behave unpredictably for functions with closely spaced roots. Nevertheless, it tends to be more stable than Newton-type methods.

If the function satisfies smoothness and bounded derivative conditions, the method is guaranteed to converge for a broad range of initial guesses. The exponential correction step expands the region of attraction, improving robustness. The summary of the convergence properties is provided in Table 1.

The proposed root-finding algorithm exhibited *high stability* due to its rapid convergence and reduced function evaluations per iteration. Stability was analyzed using fixed-point theory and the basin of attraction, ensuring that small perturbations in the initial guess do not lead to divergence. The method's high-order accuracy allowed for faster convergence while minimizing numerical errors. Polynomiography results in Section 5 indicate that the method has wider and smoother basins of attraction, reducing sensitivity to initial conditions compared to NR and HM.

In the following section, we present several numerical examples to illustrate the proposed algorithm, and comparisons are made to show its accuracy and efficiency.

As mentioned in Section 1, the efficiency index  $E$  is defined as  $E = p^{1/c}$ , where  $p$  is the order of convergence of the method and  $c$  is the number of function evaluations per iteration. The proposed method has ninth-order convergence ( $p = 9$ ) and six function evaluations per iteration ( $c = 6$ ), consisting of three function evaluations ( $f(x)$ ), two first derivative evaluations ( $f'(x)$ ), and one second derivative evaluation ( $f''(x)$ ). An explicit calculation gives  $E = 9^{1/6} \approx 1.4422$ .

The computational complexity of an iterative root-finding method is determined by the following: (i) the number of function evaluations per iteration, (ii) the arithmetic operations per iteration, and (iii) the convergence order and total iterations required. As mentioned earlier, each iteration requires six function evaluations. As each step involves only a constant number of operations, the arithmetic complexity per iteration is (1), and the number of iterations required to reach a precision  $\epsilon$  is approximately  $k = O(\log_9(\frac{1}{\epsilon}))$ .

Thus, the proposed algorithm has an optimal logarithmic complexity of  $O(\log(\frac{1}{\epsilon}))$ .

As discussed previously, we have proven that the semi-local convergence of the proposed method has a ninth-order rate. We assumed that: (i)  $f \in C^3$  is on an open interval  $D \subset R$ , (ii)  $\alpha \in D$  is a simple root of  $f : f(\alpha) = 0, f'(\alpha) \neq 0$ , and there exists  $\delta > 0$  such that:  $f'(x) \neq 0$  for all  $x \in B(\alpha, \delta) \subset D$ ,  $\left| \frac{f''(x)}{f'(x)} \right|$ , and  $\left| \frac{f^{(3)}(x)}{f'(x)} \right|$  are bounded in  $B(\alpha, \delta)$ . Now, the goal was to prove that the method converges  $\alpha$  from an initial point  $x_0 \in B(\alpha, \delta)$ , under these assumptions. The key ideas in the proof consist of analyzing the error  $e_n = x_n - \alpha$  and showing that

$$|e_{n+1}| \leq K|e_n|^9$$

for some constant  $K > 0$ , when  $|e_n|$  is sufficiently small but not necessarily infinitesimal (semi-local).

In the literature, numerous root-finding algorithms exist (see, e.g., [39–43]) each with its own limitations.

### 3. Numerical experiments

In this section, the practical applicability of the proposed scheme is demonstrated using numerical examples.

**Example 1.** Consider the nonlinear equation to illustrate the proposed algorithm:

$$\sin^2(x) - x^2 + 1 = 0. \tag{13}$$

An approximate solution of Equation (13) is

$$x \approx 1.404491648215341226035086817786868077 \tag{14}$$

Using the proposed algorithm with the initial guess  $x_0 = 1$ , we obtained the following values in the first iteration:

$$t_0 = x_0 - \frac{2f(x_0)f'(x_0)}{2f'(x_0)^2 - f(x_0)f''(x_0)} = 1.352266356364,$$

$$s_0 = t_0 \exp\left(\frac{-f(t_0)}{t_0 f'(t_0)}\right) = 1.40790110417003320,$$

$$x_1 = t_0 - \frac{f(t_0) + f(s_0)}{f'(t_0)} = 1.4030669959818645244254.$$

The function value and absolute error after the first iteration are:

$$|f(x_1)| = 0.00353271303535116810231715,$$

$$\left| \frac{x_1 - x_0}{x_1} \right| = 0.2872756590641623253773259$$

respectively. Repeating this process, we have: Iteration 2

**Table 1.** Summary of convergence

Analysis type	Result
Local convergence	Ninth-order convergence ( $e_9^n$ ) confirmed via Taylor expansion and error analysis
Semi-local convergence	The method is robust under smoothness conditions and has a large basin of attraction
Practical implications	Faster convergence than the Newton–Raphson; better stability for distant initial guesses

$$\begin{aligned}
 t_1 &= 1.4044916466908547473726183272184, \\
 s_1 &= 1.4044916482153412286835185849637, \\
 x_2 &= 1.40449164821534122520772018543, \\
 |f(x_2)| &= 0.00000000000000002053915, \\
 \left| \frac{x_2 - x_1}{x_2} \right| &= 0.00101435436464643787660.
 \end{aligned}$$

Iteration 3

$$\begin{aligned}
 t_2 &= 1.404491648215341226035086817786868, \\
 s_3 &= 1.404491648215341226035086817786868, \\
 x_4 &= 1.404491648215341226035086817786868, \\
 |f(x_3)| &= 0.000000000000000000000000, \\
 \left| \frac{x_3 - x_2}{x_3} \right| &= 0.00000000000000000589086.
 \end{aligned}$$

It is evident that  $f(x_3) = 0$ , hence confirming the approximate root as:

$$\begin{aligned}
 x &= 1.404491648215341226035086817786868 \\
 &\text{as stated in Equation (14).}
 \end{aligned}$$

This example demonstrates that the proposed algorithm rapidly converges to the approximate root.

**Example 2.** Further reinforces the applicability of the proposed method by solving a transcendental equation with an initial approximation  $x_0 = 5$ . The stopping conditions are  $|f(x)| \leq \varepsilon$  or  $\left| \frac{x_{n+1} - x_n}{x_{n+1}} \right| \leq \delta$ , where  $\varepsilon = 10^{-200}$ ,  $\delta = 10^{-600}$ .

$$x^5 + x - 10000 = 0$$

Following the proposed algorithm as demonstrated in Example 1, we obtained the following values during the iteration:

$$\begin{aligned}
 t_2 &= 6.308777130, \\
 s_2 &= 6.308777130, \\
 x_3 &= 6.308777130, \\
 f(x_3) &= 0, \\
 \left| \frac{x_3 - x_2}{x_3} \right| &= 0.
 \end{aligned}$$

The approximate solution is  $x = 6.308777130$ . This example further highlights the simplicity of the proposed algorithm. It can be easily applied to real-world problems involving nonlinear equations.

Expanding its utility to engineering applications, Example 3 examines the open-channel flow problem, originally discussed as Example 3 in Kalantari and Hans Lee.<sup>38</sup> It remains a challenge to relate water flow to elements like drainage ditches, gutters, sewers, and canals, all of which affect flow dynamics within open channels. The flow rate in such cases refers to the volume of water that passes through a certain region over a given time. A particularly problematic scenario arises when the channel is poorly maintained. The water flow in an open channel with uniform flow conditions is determined by Manning’s equation:

$$Q = \frac{\sqrt{m}}{n} Wh \left[ \frac{Wh}{W + 2h} \right]^{2/3}, \quad (15)$$

where  $m$  is the slope of the channel,  $n$  is the Manning’s roughness coefficient,  $W$  is the channel width, and  $h$  is the depth of water. The equation above can be simplified to determine the water depth ( $h$ ) in the channel for a given quantity of water, as follows:

$$f(h) = \frac{\sqrt{m}}{n} Wh \left[ \frac{Wh}{W + 2h} \right]^{2/3} - Q. \quad (16)$$

In Kalantari and Hans Lee,<sup>38</sup> the authors considered the parameters:  $Q = 14.15 \text{ m}^3/\text{s}$ ,  $W = 4.572 \text{ m}$ ,  $n = 0.017$ , and  $m = 0.0015$ , with the initial guess  $h_0 = 8.5 \text{ m}$ . Substituting these values into Equation (16) yields:

$$f(h) = 28.69285373h \left[ \frac{h}{4.572 + 2h} \right]^{2/3} - 14.15. \quad (17)$$

Applying the proposed algorithm, the following results were obtained:

Iteration 1: 1.446410377,  $f(h) = 0.25324599$ ,  
 Iteration 2: 1.465091222,  $f(h) = 0.2 \times 10^7$ ,

and

Iteration 3: 1.465091220,  $f(h) = 0$ .

From this, we obtained an approximate value of  $h = 1.465091220$ . However, Equation (17) was also solved in Kalantari and Hans Lee,<sup>38</sup> where the approximate solution was obtained after four iterations using the same data. Therefore, the proposed algorithm is more efficient than the one presented by Kalantari and Hans Lee.<sup>38</sup>

Lastly, Example 4 presents a performance comparison between the proposed algorithm and existing methods, highlighting its efficiency. Consider the following equations:

$$Eq_1 : \sin^2(x) - x^2 + 1 = 0,$$

$$Eq_2 : x^5 + x - 10000 = 0,$$

$$Eq_3 : 28.69285373x \left[ \frac{x}{4.572 + 2x} \right]^{2/3} - 14.15,$$

$$Eq_4 : e^x - 3x - 2 = 0.$$

Table 2 shows the number of iterations required to reach an approximate solution of the equation using the proposed algorithm versus various numerical methods. The stopping criterion is  $\varepsilon = 10^{-200}$ ,  $\delta = 10^{-600}$ , as in Example 2

Table 2 shows that the proposed algorithm exhibits convergence that is equal to or faster than that of the existing methods.

The effectiveness and versatility of the proposed algorithm are demonstrated through a series of illustrative examples. Example 1 serves to elucidate the step-by-step implementation of the proposed algorithm, highlighting its computational framework and underlying logic. Example 2 emphasizes the simplicity and user-friendliness of the algorithm, showcasing its ease of understanding and applicability, even for users with limited experience. Its practical relevance is further reinforced in Example 3, which applies the algorithm to a real-world problem in civil and environmental engineering, namely open-channel flow. This example illustrates the algorithm's ability to address challenges in modeling water flow through elements such as drainage ditches, gutters, sewers, and canals, which significantly influence fluid dynamics in open channels. Finally, Example 4 offers a comparative analysis between the proposed method and several well-established algorithms, including the NR method, HM, MHM, numerical methods from refs.[38] and [1, 2] the enhanced method from ref. [44] the ninth-order root-finding method from ref. [38] and the ninth-order memory-based iterative technique from ref. [44] This comparative study underscores the superior efficiency and accuracy of the proposed algorithm in solving nonlinear equations. As observed in Table 2, the proposed algorithm demonstrates equal or superior convergence

speed compared to these existing methods, showcasing its efficiency and robustness. While Table 2 demonstrates that the proposed method typically requires fewer or equal iterations compared to existing methods, it is important to note that the number of iterations alone does not fully capture an algorithm's overall effectiveness. Therefore, several additional performance metrics have been evaluated to provide a more comprehensive assessment. The proposed method not only achieved ninth-order convergence with fewer iterations but also exhibited a high-efficiency index ( $\approx 1.4422$ ), low computational cost, and robust stability, confirmed through both local and semi-local convergence analyses. Its effectiveness was further supported by polynomiographic visualization (Section 5), real-world application (Example 3) in open-channel flow, and favorable comparisons with established methods across multiple performance metrics. These comprehensive results affirm the practical utility and numerical superiority of the proposed method.

## 4. Implementation

In this section, we discuss the implementation of the proposed algorithm using various software tools such as Maple and Python, along with sample computations.

### 4.1. Maple implementation

```
PropAlg := proc (a, Eq, max_iter)
local i, a1, f, r, y1, z1;
i := 0; a1 := evalf(a);
f := unapply(lhs(Eq), x);
if f(a1) = 0 then return a1 end if;
i := 0;
do
i := i+1;
y1 := a1-
2*f(a1)*(D(f))(a1)/(2*(D(f))(a1)^2
-f(a1)*((D@@2)(f))(a1));
z1 := y1*exp(
-f(y1)/(y1*(D(f))(y1)));
r := y1-(f(y1)+f(z1))/(D(f))(y1);
printf("\Iteration %a = %a.
f(x) = %a, AbsError = %a \n",
i, r, f(r), abs((r-a1)/r));
if max_iter == i then return r
else a1 := r
end if
end do
end proc
```

**Example 5.** Recall Example 1 for sample computations using the Maple implementation.

**Table 2.** Number of iterations

Eq	Root	NR	HM	MHM	NM1	NM2	EM	Sania9	MBIT9	PM
Eq <sub>1</sub>	1.404491648	11	7	5	5	4	5	3	3	3 (Example 1)
Eq <sub>2</sub>	6.308777130	12	8	6	5	6	5	3	3	3 (Example 2)
Eq <sub>3</sub>	1.465091220	10	7	5	4	4	4	3	3	3 (Example 3)
Eq <sub>4</sub>	2.125391199	10	8	6	4	4	4	3	3	3

Notes: NM1 and NM2 represent the numerical methods proposed in refs. [38] and [1,2], respectively. EM indicates the enhanced method from reference [44]. RFM refers to the ninth-order root-finding method from ref. [37], which introduced a new three-step numerical scheme for solving nonlinear scalar and vector equations. MBIT denotes the ninth-order memory-based iterative technique from ref. [37], in which the authors proposed a novel three-step, memory-based method for solving nonlinear equations. Sania is a Ninth-order method by Sania. Abbreviations: Eq, equation; HM, Halley’s method; MHM, modified Halley’s method; NR, Newton Raphson method; PM, proposed method.

```
> Prop_Alg(1, sin(x)^2-x^2+1 = 0,3);
Iteration 1 = 1.403066996.
f(x) = .35327132e-2,
AbsError = .2872756591
Iteration 2 = 1.404491649.
f(x) = -.2e-8,
AbsError = .1014354910e-2
Iteration 3 = 1.404491649.
f(x) = -.2e-8,
AbsError = 0.
1.404491649

    Example 6. Consider with x0 = 0.8.
> Prop_Alg(0.8, 10*x*exp(-x^2)-1 = 0, 4);
Iteration 1 = 1.531609079.
f(x) = .466791835,
AbsError = .4776735063
Iteration 2 = 1.679628729.
f(x) = .5200e-5,
AbsError = .8812640999e-1
Iteration 3 = 1.679630611.
f(x) = -.12e-8,
AbsError = .1120484461e-5
Iteration 4 = 1.679630611.
f(x) = -.12e-8, AbsError = 0.
1.679630611
```

```
t = x - (2 * f(x) * df(x)) / (2 *
(df(x))**2 - f(x) * ddf(x))
s = t * math.exp(-f(t) / (t * df(t)))
if t * df(t) != 0 else t
x_new = t - (f(t) + f(s)) / df(t)
print(f"Iteration {iteration}:
x = {x_new:.15f}, f(x) =
{f(x_new):.15e}")
if abs(x_new - x) < tol:
    print("\nConverged to root:",
    x_new)
    print("Total iterations:",
    iteration)
    return x_new
x = x_new
print("\nMaximum iterations reached.
Root may not be accurate.")
return x

# Example: Solve sin^2(x) - x^2 + 1 = 0
initial_guess = 1.0 # Start near an
expected root
root = ninth_order_method(initial_guess)
```

Output of the implementation:

### 4.2. Python implementation

Consider the transcendental Equation (13) from Example 1 for sample computations using the Python implementation, as follows.

```
import numpy as np
import math
def f(x):
    return np.sin(x)**2-x**2+1
def df(x):
    return 2*np.sin(x)*np.cos(x)-2*x
def ddf(x):
    return 2*(np.cos(x)**2- np.sin(x)**2)-2
def ninth_order_method(x0, tol=1e-10,
max_iter=20): x = x0
    for iteration in range(1, max_iter + 1):
```

```
Iteration 1: x = 1.403066995981864,
f(x) = 3.532713035351298e-03
Iteration 2: x = 1.404491648215341,
f(x) = -4.440892098500626e-16
Iteration 3: x = 1.404491648215341,
f(x) = -4.440892098500626e-16
```

```
Converged to root:
1.4044916482153413
Total iterations: 3
```

=== Code Execution Successful ===

## 5. Polynomiography via the proposed method

The aim of this section is to represent polynomiographs generated using our proposed method as outlined in Equation (9). A polynomiograph is an image produced through the process of polynomiography, a term coined by Kalantri in 2005. It is defined as “the art and science of visualization in approximation of the zeros of complex polynomials, via fractal and non-fractal images created using the mathematical convergence properties of iteration functions.” The concept of a “fractal,” introduced by Benoit Mandelbrot, describes a geometric shape in which each part has the same statistical properties as the whole. Although both polynomiographs and fractals can be created using various numerical algorithms, they differ significantly in their structural scale. A “polynomiographer” can systematically adjust the structure and pattern by applying different numerical algorithms to various complex polynomials. Generally, polynomiographs and fractals belong to different categories of graphical objects.

To create polynomiographs over the complex plane  $C$ , we begin by defining a rectangular region  $\mathcal{R}$  with dimensions  $[-2, 2] \times [-2, 2]$ , setting a precision threshold of  $\varepsilon = 10^{-3}$ , and an iteration limit of  $m = 15$ . Each point  $z_0$  within this region undergoes an iterative process, and the point corresponding to  $z_0$  is colored based on how closely the truncated orbit converges to a root or fails to converge. The image resolution depends on the discretization of the rectangle  $\mathcal{R}$ . For example, a  $2000 \times 2000$  grid results in a high-resolution image. The colors in polynomiographs are typically linked to the number of iterations required to approximate the zeros of the complex polynomial to a specified accuracy using a chosen numerical algorithm. The fundamental algorithm for generating polynomiographs is outlined in Algorithm 1 below.

### Algorithm 1. Polynomiograph’s generation

*Input:*

$p \in C$ —Polynomial

$A \in C$ —Area

$m$ —Maximum number of iterations

$I$ —Iteration method

$\varepsilon$ —Accuracy

Colormap[0... $C - 1$ ]—Colormap with  $C$

colors.

*Output :* Polynomiograph for the complex polynomial  $p$  in area  $A$

for  $z_0 \in A$ , do

$i = 0$

while *imdo*

$z(n + 1) = I(z_n)$

if  $|z(n + 1) - z_n| < \varepsilon$  then

break

$i = i + 1$

color  $z_0$  by means of colormap

In iterative algorithms, a stopping criterion is crucial for determining whether the process has reached convergence or divergence. This criterion, known as a convergence test, evaluates the algorithm’s progress in finding a solution. The typical form of a convergence test is

$$|z_{i+1} - z_i| < \varepsilon \quad (18)$$

where  $z_{i+1}$  and  $z_i$  are successive points in the iteration, and  $\varepsilon > 0$  is a predefined accuracy threshold. The convergence test  $(z_{i+1}, z_i, \varepsilon)$  indicates TRUE if the method converges to a root and FALSE otherwise. In this study, we used this convergence test Equation (18). The color variations in the polynomiographs represent the number of iterations needed to approximate the root within the specified accuracy  $\varepsilon$ . By modifying the parameter  $m$ , which limits the maximum number of iterations, it is possible to generate numerous visually appealing polynomiographs. The exploration of polynomiography and its artistic implications is covered in references.<sup>45–47</sup>

Here, we present some examples of the following complex polynomials using our developed algorithms and compare them with the polynomiographs obtained by using other well-known two-step iterative methods:

(i)  $p_1(z) = z^3 - z^2 - z - 1$ , Area  $[-2, 2]$

(ii)  $p_2(z) = z^4 - z^3 - z^2 - z - 1$ , Area  $[-2, 2]$

(iii)  $p_3(z) = z^5 - z^4 - z^3 - z^2 - z - 1$ , Area  $[-2, 2]$

(iv)  $p_4(z) = z^6 - z^5 - z^4 - z^3 - z^2 - z - 1$ , Area  $[-2, 2]$

The colormap used for coloring the iterations in the generation of polynomiographs is presented in the following figure.

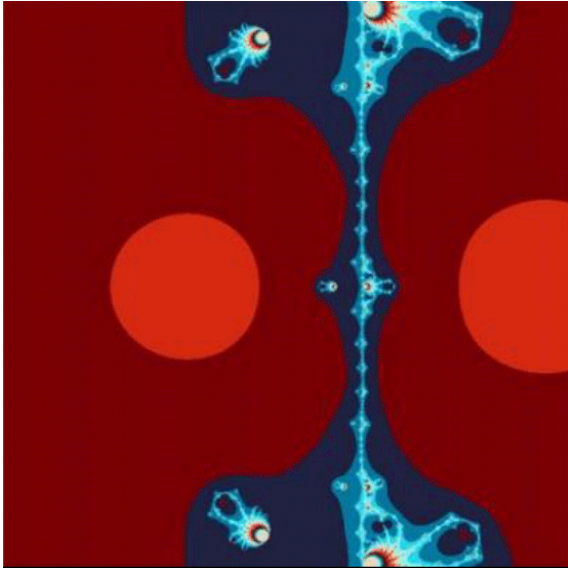


**Figure 1.** The colormap used for generating polynomiographs

The colormap shown in Figure 1 illustrates how points move toward the solutions of a polynomial. Different colors represent different roots, allowing one to see which point converges to which

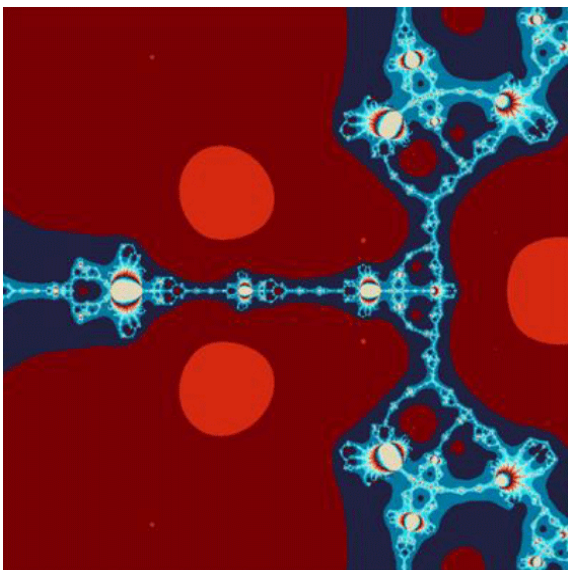
solution. Lighter colors indicate that the point converges quickly, while darker shades denote that more steps are required. The fuzzy or mixed-color edges between regions show areas where a small change can direct a point to a different solution. This creates an image that is both complex and beautiful, with patterns that resemble fractals.

**Example 7.** Polynomiographs for the polynomial  $z^2 - z - 1$  via the proposed method.



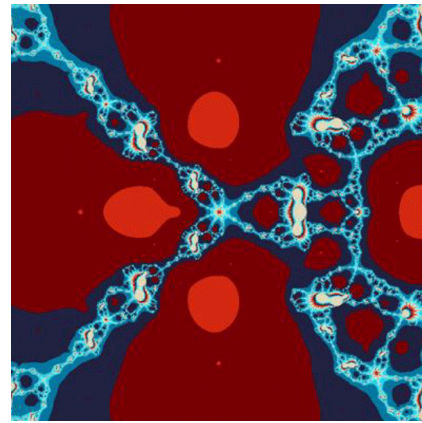
**Figure 2.** Polynomiograph for the polynomial  $p_1(z)$  using Method IX

**Example 8.** Polynomiographs for the polynomial  $z^3 - z^2 - z - 1$  via the proposed method.



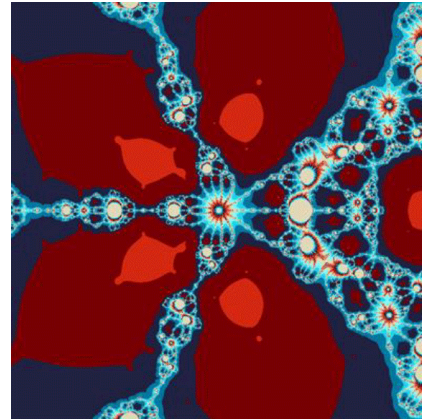
**Figure 3.** Polynomiograph for the polynomial  $p_2(z)$  using Method IX

**Example 9.** Polynomiographs for the polynomial  $z^4 - z^3 - z^2 - z - 1$  via the proposed method.



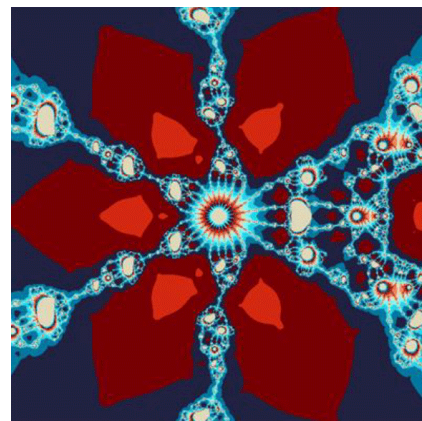
**Figure 4.** Polynomiograph for the polynomial  $p_3(z)$  using Method IX

**Example 10.** Polynomiographs for the polynomial  $z^5 - z^4 - z^3 - z^2 - z - 1$  via the proposed method.



**Figure 5.** Polynomiograph for the polynomial  $p_3(z)$  using Method IX

**Example 11.** Polynomiographs for the polynomial  $z^6 - z^5 - z^4 - z^3 - z^2 - z - 1$  via the proposed method.



**Figure 6.** Polynomiograph for the polynomial  $p_4(z)$  using Method IX

Table 3 presents the outstanding performance of the proposed iterative method applied to four different polynomials. The performance was evaluated based on three key metrics: average number of iterations (ANI), average area index (CAI),

and computation time (in seconds). The results clearly demonstrate that the proposed method efficiently handles increasing polynomial complexity while maintaining an exceptional balance between accuracy and computational efficiency, making it a highly reliable approach for solving polynomial equations.

**Table 3.** Dynamical analysis of the proposed method

Polynomial	ANI	CAI	Time
$p_1$	2.6811	0.9902	31.021
$p_2$	3.0728	0.9809	129.023
$p_3$	3.3573	0.9741	145.432
$p_4$	3.5553	0.9699	210.098

Abbreviations: ANI, average number of iterations; CAI, convergence area index.

To further highlight the advantages of the proposed method, we analyzed both numerical results and graphical representations using polynomiographs in Figures 1-6. The ANI values showed a controlled and gradual increase from 2.6811 for  $p_1$  to 3.5553 for  $p_4$ , illustrating that the iterative method remains computationally feasible even as polynomial complexity increases. Unlike other methods that may require excessive iterations for higher-degree polynomials, the proposed method ensures stability without exponential growth in iterations. This efficiency was further confirmed by the polynomiographs, where the speed of convergence, depicted by the color of each point, highlights the method’s ability to quickly approximate polynomial roots.

Moreover, the CAI values remained impressively high, with only a slight decrease from 0.9902 for  $p_1$  to 0.9699 for  $p_4$ , proving that the method consistently delivers accurate results across all tested polynomials. These accuracy trends align with the polynomiographs, where regions with minimal color variation indicate a steady and predictable convergence pattern. The ability of the method to maintain high accuracy while efficiently converging further emphasizes its superiority over traditional iterative techniques.

Although computation time increases from 31.021 s for  $p_1$  to 210.098 s for  $p_4$ , this is a natural consequence of the increasing complexity of the polynomial equations. Importantly, the polynomiographs further validate that the proposed method successfully converges in all cases, demonstrating its robustness and adaptability to varying polynomial structures. Compared to alternative approaches, this method ensures a favorable

trade-off between computational time and accuracy, making it particularly well-suited for real-world applications that demand both precision and efficiency.

In summary, the proposed iterative method stands out as a highly effective and reliable approach for solving polynomial equations. By seamlessly integrating numerical data with graphical insights from polynomiographs, we confirm that the method maintains exceptional accuracy, exhibits a stable and controlled iteration growth pattern, and efficiently handles increasing computational demands. With further optimization, such as adaptive iteration strategies or parallel processing, the method could become even more powerful, solidifying its position as an optimal choice for complex mathematical computations.

## 6. Conclusion

Nonlinear phenomena appear in various fields, including economics, engineering, and scientific research. With the rapid expansion of computational science, the development of new numerical schemes and the improvement of existing ones remain crucial. These numerical methods should ideally offer higher convergence rates while being computationally efficient. In this study, we proposed a new three-step iterative algorithm for solving nonlinear scalar equations, addressing both convergence efficiency and computational cost. The proposed method was shown to have ninth-order convergence while requiring only six function evaluations per iteration. Several numerical examples were presented to demonstrate its effectiveness and superior performance over classical methods. The algorithm was implemented in Maple and Python, showcasing its practicality in widely used computational tools. Additionally, it can be extended to other software environments, making it adaptable for various applications.

Future work will focus on reducing computational cost, extending the algorithm to systems of nonlinear equations, and exploring adaptive step-size techniques to improve robustness. The proposed method provides a reliable and efficient approach to solving nonlinear equations, contributing to the advancement of numerical analysis and computational mathematics.

## Acknowledgments

None.

## Funding

None.

## Conflict of interest

The authors declare that they have no competing interests.

## Author contributions

*Conceptualization:* Srinivasarao Thota

*Formal analysis:* Amir Naseem, Thumati Gopi, Kashireddy Sai Nandan Reddy, Padarthi Sai Kousik

*Investigation:* Thulasi Bikku, Amir Naseem

*Methodology:* Srinivasarao Thota, Amir Naseem

*Writing – original draft:* Srinivasarao Thota

*Writing – review & editing:* Thulasi Bikku, Shanmugasundaram Palanisamy

## Availability of data

The datasets generated and analyzed during the current study are presented in this manuscript.


## References

1. Traub JF. Iterative Methods for the Solution of Equations, vol. 312. Providence, RI, USA: American Mathematical Society; 1982.
2. Rafiullah M. A fifth-order iterative method for solving nonlinear equations. *Numer Anal Appl.* 2011; 4: 239-243.  
<https://doi.org/10.1134/S1995423911030062>
3. Ganji DD. The application of He's homotopy perturbation method to nonlinear equations arising in heat transfer. *Phys Lett A.* 2006;355:337-341.  
<https://doi.org/10.1016/j.physleta.2006.02.056>
4. Thota S, Shanmugasundaram P. On new sixth and seventh order iterative methods for solving non-linear equations using homotopy perturbation technique. *BMC Res Notes.* 2022;15:267.  
<https://doi.org/10.1186/s13104-022-06154-5>
5. Abbasbandy S. Improving Newton–Raphson method for nonlinear equations by modified Adomian decomposition method. *Appl Math Comput.* 2003; 145:887-893.  
[https://doi.org/10.1016/S0096-3003\(03\)00282-0](https://doi.org/10.1016/S0096-3003(03)00282-0)
6. He JH, Wu XH. Variational iteration method: new development and applications. *Comput Math Appl* 2007;54:881-894.  
<https://doi.org/10.1016/j.camwa.2006.12.083>
7. Tari H, Ganji DD, Babazadeh H. The application of He's variational iteration method to nonlinear equations arising in heat transfer. *Phys Lett A.* 2007;363:213-217.  
<https://doi.org/10.1016/j.physleta.2006.11.005>
8. Noor MA, Waseem M. Some iterative methods for solving a system of nonlinear equations. *Comput Math Appl.* 2009;57:101-106.  
<https://doi.org/10.1016/j.camwa.2008.10.067>
9. Zafar F, Mir NA. A generalized family of quadrature based iterative methods. *Gen Math.* 2010;18:43-51.
10. Thota S, Awad MM, Shanmugasundaram P. A derivative-free root-finding algorithm using exponential method and its implementation. *BMC Res Notes.* 2023;16:276.  
<https://doi.org/10.1186/s13104-023-06554-1>
11. Thota S, Gemechu T, Ayoade AA. On new hybrid root-finding algorithms for solving transcendental equations using exponential and halley's methods. *Ural Math J.* 2023; 9(1):176-186.  
<http://dx.doi.org/10.15826/umj.2023.1.016>
12. Thota S, Gemechu T, Shanmugasundaram P. New algorithms for computing non-linear equations using exponential series. *Palestine J Math.* 2021; 10(1): 128-134.
13. Thota S. A new root-finding algorithm using exponential series. *Ural Math J.* 2022;5(1):83-90.  
<https://doi.org/10.15826/umj.2019.1.008>
14. Thota S. A New Hybrid Halley-False Position type Root Finding Algorithm to Solve Transcendental Equations. Istanbul International Modern Scientific Research Congress-III, 06-08 May 2022, Istanbul Gedik University, Istanbul, Turkey.
15. Chen D, Argyros IK, Qian QS. A note on the Halley method in Banach spaces. *Appl Math Comput.* 1983;58:215-224.
16. Noor MA, Khan WA, Hussain A. A new modified Halley method without second derivatives for nonlinear equation. *Appl Math Comput.* 2007;189:1268-1273.  
<https://doi.org/10.1016/j.amc.2006.12.011>
17. Cordero A, Hueso JL, Martínez E, Torregrosa JR. A modified Newton–Jarratt's composition. *Numer Algorithms.* 2010; 55:87–99.  
<https://doi.org/10.1007/s11075-009-9359-z>
18. Thota S, Ayoade AA. A new numerical algorithm to compute a root of non-linear equations using exponential method. interplay of fractals and complexity in mathematical modelling and physical patterns. In: ISMAFDS 2023. Cham: Springer; 2025.  
[https://doi.org/10.1007/978-3-031-58641-5\\_28](https://doi.org/10.1007/978-3-031-58641-5_28)
19. Noor MA, Noor KI, Al-Said E, Waseem M. Some new iterative methods for nonlinear equations. *Math Probl Eng.* 2010; 198943.  
<https://doi.org/10.1155/2010/198943>
20. Noor MA, Noor KI. Three-step iterative methods for nonlinear equations. *Appl Math Comput.* 2006; 183:322-327.  
<https://doi.org/10.1016/j.amc.2006.05.055>
21. Shah FA, Noor MA, Waseem M. Some second-derivative-free sixth-order convergent iterative methods for non-linear equations. *Maejo Int J Sci Technol.* 2016; 10(01):79-87.  
<https://doi.org/10.14456/mijst.2016.7>
22. Abro HA, Shaikh MM. A new time-efficient and convergent nonlinear solver. *Appl Math Comput.* 2019; 355:516-536.  
<https://doi.org/10.1016/j.amc.2019.03.012>
23. Torregrosa JR, Argyros IK, Chun C, Cordero A, Soleymani F. Iterative methods for nonlinear equations or systems and their applications 2014.


- J Appl Math.* 2014; 2014: 293263.  
<https://doi.org/10.1155/2014/293263>
24. Darvishi MT, Barati A. A third-order Newton-type method to solve systems of nonlinear equations. *Appl Math Comput.* 2007; 187:630-635.  
<https://doi.org/10.1016/j.amc.2006.08.080>
  25. Qureshi S, Ramos H, Soomro AK. A new nonlinear ninth-order root-finding method with error analysis and basins of attraction. *Mathematics.* 2021;9(16):1996.  
<https://doi.org/10.3390/math9161996>
  26. Okawa H, Fujisawa K, Yamamoto Y, et al. The W4 method: a new multi-dimensional root-finding scheme for nonlinear systems of equations. *Appl Numer Math.* 2023; 183: 157-172.  
<https://doi.org/10.1016/j.apnum.2022.08.019>
  27. Thota S, Gopisairam T. On a symbolic method for second-order boundary value problems over algebras. *Int J Appl Math.* 2024;37(6):577-590.  
<http://dx.doi.org/10.12732/ijam.v37i6.1>
  28. Srivastav VK, Thota S, Kumar M. A new trigonometrical algorithm for computing real root of nonlinear transcendental equations. *Int J Appl Comput Math.* 2019; 5:44.  
<https://doi.org/10.1007/s40819-019-0600-8>
  29. Thota S, Krishna CB, Ayoade AA, Bikku T. On a new hybrid root-finding algorithm for solving nonlinear equations with implementation in excel and maple. *J Basic Sci.* 2024; 24(7): 270-277.  
<https://doi.org/10.37896/JBSV24.7/3250>
  30. Naseem A, Rehman MA, Abdeljawad T. A novel root-finding algorithm with engineering applications and its dynamics via computer technology. *IEEE Access.* 2022;10:19677-19684.  
<https://doi.org/10.1109/ACCESS.2022.3150775>
  31. Thota S, Srivastav VK. Quadratically convergent algorithm for computing real root of nonlinear transcendental equations. *BMC Res Notes.* 2018;11:909.  
<https://doi.org/10.1186/s13104-018-4008-z>
  32. Etesami R, Madadi M, Mashinchi M, Ganjoei RA. A new method for rooting nonlinear equations based on the Bisection method. *MethodsX* 2021;8:101502.  
<https://doi.org/10.1016/j.mex.2021.101502>
  33. Thota S. A numerical algorithm to find a root of non-linear equations using householder's method. *Int J Adv Appl Sci.* 2021; 10(2): 141-148.  
<http://doi.org/10.11591/ijaas.v10.i2.pp141-148>
  34. Chueca-Díez F, Gañán-Calvo AM. A fast numerical algorithm for finding all real solutions to a system of N nonlinear equations in a finite domain. *Numer Algor* 2024.  
<https://doi.org/10.1007/s11075-024-01908-7>
  35. Naseem A, Rehman MA, Abdeljawad T. Real-world applications of a newly designed root-finding algorithm and its polynomiography. *IEEE Access.* 2021;9: 160868-160877.  
<https://doi.org/10.1109/ACCESS.2021.3131498>
  36. Aggarwal A, Pant S. Beyond newton: a new root-finding fixed-point iteration for nonlinear equations. *Algorithms.* 2020;13(4):78.  
<https://doi.org/10.3390/a13040078>
  37. Mittal SK, Panday S, Jäntschi L. Enhanced ninth-order memory-based iterative technique for efficiently solving nonlinear equations. *Mathematics.* 2024;12(22): 3490.  
<https://doi.org/10.3390/math12223490>
  38. Kalantari B, Hans Lee E. Newton-ellipsoid polynomiography. *J Mathnd the Arts.* 2019;13(4): 336-352.  
<https://doi.org/10.1080/17513472.2019.1600959>
  39. Iliev A, Kyurkchiev N. Nontrivial Methods in Numerical Analysis: Selected Topics in Numerical Analysis. Saarbrücken: LAP LAMBERT Academic Publishing; 2010: 256. ISBN:978-3-8433-6793-6.
  40. Ignatova B, Kyurkchiev N, Iliev A. Multipoint algorithms arising from optimal in the sense of Kung-Traub iterative procedures for numerical solution of nonlinear equations. *Gen Math Notes.* 2012;6(2): 45-79.
  41. Kyurkchiev N, Iliev A. On some multipoint methods arising from optimal in the sense of Kung-Traub algorithms. *Int J Math Methods Models Biosci.* 2013; 2(1):12-18.
  42. Hristov V, Iliev A, Kyurkchiev N. A note on the convergence of nonstationary finite-difference analogues. *Comput Math Math Phys.* 2005; 45:194-201.
  43. S. Thota, Krishna CB, Ayoade AA, Bikku T. On A New Hybrid Root-Finding Algorithm for Solving Nonlinear Equations with Implementation in Excel and Maple. *J Basic Sci.* 2024;24(7):270-277.  
<https://doi.org/10.37896/JBSV24.7/3250>
  44. Petkovic M, Neta B, Petkovic L, Dzunic J. Multipoint Methods for Solving Nonlinear Equations. Oxford, UK: Academic press; 2013.
  45. Gdawiec K. Fractal patterns from the dynamics of combined polynomial root finding method s. *Nonlinear Dyn.* 2017;90(4):2457-2479.  
<https://doi.org/10.1007/s11071-017-3813-6>
  46. Kang SM, Naseem A, Nazeer W, Munir M, Yong C. Polynomiography via modified Abbasbanday's method. *Int J Math Anal.* 2017;11(3):133-149.  
[https://www.researchgate.net/publication/299535865\\_POLYNOMIOGRAPHY\\_VIA\\_MODIFIED\\_GOLBABAIAND\\_JAVIDI'S\\_METHOD](https://www.researchgate.net/publication/299535865_POLYNOMIOGRAPHY_VIA_MODIFIED_GOLBABAIAND_JAVIDI'S_METHOD)
  47. Thota S, Krishna CB. A root finding algorithm for transcendental equations using hyperbolic tangent function. *AIP Conf Proc.* 2024;2971:060012.  
<https://doi.org/10.1063/5.0195773>

**Srinivasarao Thota** completed his M.Sc. in Mathematics from the Indian Institute of Technology (IIT) Madras, India, and Ph.D. in Mathematics from Motilal Nehru National Institute of Technology (NIT) Allahabad, India. Dr. Thota's areas of research interest

are Computer Algebra (symbolic methods for differential equations), Numerical Analysis (root finding algorithms), and Mathematical Modelling (ecology). Dr. Thota has published more than 50 research papers in several international journals and presented his work in different national and international conferences as an oral presenter and an expert/invited speaker. He is also an editorial board member of various international journals and has received several awards for academic/research work. Presently, he is working as an Associate Professor at the Department of Mathematics, Amrita Vishwa Vidyapeetham, Amravati, Andhra Pradesh, India.

 <https://orcid.org/0000-0002-3265-5656>

**Amir Naseem** received the M.S. degree in mathematics from Lahore Leads University, Lahore, Pakistan. He is currently a Ph.D. Scholar at the Department of Mathematics, University of Management and Technology, Lahore. He has published more than 35 research articles in different well-known journals. His current research interests include numerical analysis, iterative methods, and polynomiography.


 <https://orcid.org/0000-0001-7010-6810>

**Thumati Gopi** is a bachelor of technology (B.Tech.) student in Computer Science and Engineering (CSE). He secured more than 8.6 CGPA in his first two years. Presently, he is in the final year of his B.Tech. Degree.


**Kashireddy Sai Nandan Reddy** is a bachelor of technology (B.Tech.) student in Computer Science and Engineering (CSE). He secured more than 8.8 CGPA in his first two years. Presently, he is in the final year of his B.Tech. Degree.

**Padarathi Sai Kousik** is a bachelor of technology (B.Tech.) student in Computer Science and Engineering (CSE). He secured more than 8.9 CGPA in his first two years. Presently, he is in the final year of his B.Tech. Degree.

**Thulasi Bikku** is an accomplished Associate Professor holding a Ph.D. in Computer Science and Engineering from JNTUA, Anantapur, in 2018, with a distinguished career characterized by significant contributions to academia. Her expertise spans various domains, including Unsupervised Machine Learning, Deep Learning, and Natural Language Processing, reflecting her broad knowledge base. In May 2023, she successfully concluded her Postdoctoral Fellowship (PDF) at the University of Santiago de Chile, situated in the capital city of Santiago, Chile. Her academic career features significant roles, including serving as an Associate Professor at Vignan's Institute in Guntur. Beyond her roles in academia, she has made significant contributions to the field as an editorial member for prestigious journals, a conference chair, and a respected reviewer. Additionally, she holds patents for innovative inventions (3), further highlighting her commitment to pushing the boundaries of knowledge. Her dedication to knowledge-sharing is evident through her involvement in guest lectures, active participation in workshops, and her contributions to faculty development programs. Her passion for fostering education and research is truly commendable. She has an impressive record of scholarly achievements, with a publication history that includes over 50+ research articles published in highly regarded and indexed academic journals. In summary, Dr. Bikku embodies a well-rounded academician with an unwavering commitment to education, research, and professional growth. Her remarkable contributions to the field of Computer Science and Engineering make her a valuable asset to the academic community.

 <https://orcid.org/0000-0001-6202-1438>

**P. Shanmugasundaram** was born and studied in Tamil Nadu, India. Presently, I am working as a Professor at the Department of Mathematics, Mizan-Tepi University, Ethiopia, and also looking at the K.S.R. College of Engineering, Tiruchengode, Namakal District, India. My research interest includes Fuzzy Logic, Topology, Differential Equations, and Optimization.

 <https://orcid.org/0000-0003-1176-0411>

