Hindawi
Complexity
Volume 2022, Article ID 9705690, 19 pages
https://doi.org/10.1155/2022/9705690

Research Article

Optimal Algorithms for Nonlinear Equations with Applications and Their Dynamics

Amir Naseem[1], **M. A. Rehman**[1], **and Nasr Al Din Ide**[2]

[1]*Department of Mathematics, University of Management and Technology, Lahore 54770, Pakistan*
[2]*Department of Mathematics, Faculty of Science, Aleppo University, Aleppo, Syria*

Correspondence should be addressed to Amir Naseem; amir.kasuri89@gmail.com and Nasr Al Din Ide; ide1112002@yahoo.ca

Received 16 May 2022; Accepted 18 August 2022; Published 27 September 2022

Academic Editor: Shanmugam Lakshmanan

Copyright © 2022 Amir Naseem et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In the present work, we introduce two novel root-finding algorithms for nonlinear scalar equations. Among these algorithms, the second one is optimal according to Kung-Traub's conjecture. It is established that the newly proposed algorithms bear the fourth- and sixth-order of convergence. To show the effectiveness of the suggested methods, we provide several real-life problems associated with engineering sciences. These problems have been solved through the suggested methods, and their numerical results proved the superiority of these methods over the other ones. Finally, we study the dynamics of the proposed methods using polynomiographs created with the help of a computer program using six cubic-degree polynomials and then give a detailed graphical comparison with similar existing methods which shows the supremacy of the presented iteration schemes with respect to convergence speed and other dynamical aspects.

1. Introduction

A huge number of complicated problems in mathematics and engineering disciplines are directly connected to the solution of transcendental and algebraic nonlinear equations of the form:

$$\varphi(x) = 0, \tag{1}$$

where the real-valued function φ is defined on the open connected set. Mostly, the direct solution of these problems is not possible to find via analytical methods, and ultimately, we have to move towards the iterative algorithms. By an algorithm, we mean a sequence of finite number of steps to achieve the required goal (solution) of the given problem. The repetition of these steps is called iterations. In the process of iterative algorithms, we always need a starting point to initialize the iteration process. This starting point is usually called the initial guess that is rectified after each iteration till the required accuracy is gained.

In literature, there is a plethora of root-finding iterative algorithms for the problems related to the nonlinear equations. The oldest, classical, and most widely used algorithm was suggested by Newton–Raphson in 1690 [1]. Geometrically, the derivation of Newton's algorithm was purely based on the concept of slope of the line. This method requires two evaluations per iteration and possesses the quadratic convergence order. For many years, Newton's method has been implemented successfully for root-finding of nonlinear problems. After that, Gutierrez and Hernández [2] presented a new family of Chebyshev–Halley type methods in Banach spaces which were utilized for root-finding of nonlinear problems. In 1993, Argyros et al. [3] discussed the applications of Halley method in Banach space. After few years, Chun [4] constructed Newton-like iteration methods which were purely designed for finding one-dimensional nonlinear equations' solution. After the construction of one step Newton-like iterative algorithms, a huge class of researchers tried to modify it and suggested a broad range of root-finding algorithms with the help of different mathematical techniques and established a class of multistep algorithm. For further details of the multistep algorithm, one can see Amiri et al. [5], Behl et al. [6], Naseem et al. [7], and Ozyapici [8]. The motivation behind these modifications is to attain higher-order and more efficient algorithms.

Ostrowski [9] and Traub [10] in the twentieth century introduced the concept of multistep iteration schemes and proposed two-step fourth-order iteration schemes with Newton's iteration method as a predictor. In 2006, Aslam Noor and Inayat Noor [11] introduced some new three-step iterative schemes and proved their third-order convergence. After that, Golbabi and Javidi [12] in 2007 presented a new cubically convergent method whose derivation is totally based on the homotopy-perturbation method. By utilizing the new series expansion of the nonlinear function, Noor et al. [13], in 2012, constructed and then analyzed some novel two-step iteration methods and discussed some special cases. These suggested schemes possessed the convergence of quadratic and cubic orders and were actually the modified form of Newton's algorithm. Kumar et al. [14] presented a novel class of sixth-order parameter-based methods for finding zeros of one-dimensional nonlinear equations in 2018. In 2019, Said Solaiman et al. [15] proposed derivative free optimal fourth-order and eighth-order versions of King's approach by combining the composition technique with rational interpolation, as well as the Pade's concept of approximation. In 2020, Chand et al. [16] developed some novel PotraPtak type optimal sixth- and eighth-order iteration methods by utilizing the idea of weight functions on the PotraPtak method for determining the approximate zeros of nonlinear models and applied them on some real-life engineering problems. The authors also analyzed the stability of the suggested schemes via basins of attraction for some cubic and quadratic polynomials. Naseem et al. [17] recently constructed and analyzed some novel ninth-order iteration schemes by implementing the idea of variational iteration and then investigated the dynamical behaviour via polynomiographs of different complex polynomials.

The main contributions of the present research are given as follows:

(i) We developed and examined two novel predictor corrector type iterative techniques, namely, Algorithm 1 and Algorithm 2, in which Newton's method is used in the predictor step.

(ii) We demonstrated that these newly created approaches have fourth- and sixth-order of convergence and are more efficient than the other well-known iterative methods of the same type.

(iii) The suggested approaches were used to solve several test cases in order to evaluate their validity and accuracy.

(iv) We compare the polynomiographs of newly proposed techniques to those of existing methods of the same category in terms of time and other dynamical aspects. The exhibited polynomiographs contain highly fascinating and attractive patterns that represent many polynomial characteristics.

The remaining sections of the paper are arranged as follows:

In Section 2, we have constructed two novel algorithms. The convergence criteria for the constructed algorithms have been discussed in Section 3. In Section 4, we solved different test problems for assuring the validity of the constructed methods. Section 5 includes the graphical properties of the constructed algorithms, and Section 6 contains the conclusion.

2. Construction of New Optimal Root-Finding Algorithms

By employing the technique of modified homotopy perturbation, Golbabai and Javidi [12] introduced the following iteration formula:

$$x_{p+1} = x_p - \frac{\varphi(x_p)}{\varphi'(x_p)} - \frac{\varphi^2(x_p)\varphi''(x_p)}{2\left[\varphi'^3(x_p) - \varphi(x_p)\varphi'(x_p)\varphi''(x_p)\right]}, \tag{2}$$

which is a third-order iteration method, usually known as Javidi's method. In [18], Rafiq and Rafiullah considered Newton's iteration method in the predictor step and presented a new two-step Javidi's iteration method given as

$$\begin{aligned} y_p &= x_p - \frac{\varphi(x_p)}{\varphi'(x_p)}, p = 0, 1, 2, \ldots, \\ x_{p+1} &= y_p - \frac{\varphi(y_p)}{\varphi'(y_p)} - \frac{\varphi^2(y_p)\varphi''(y_p)}{2\left[\varphi'^3(y_p) - \varphi(y_p)\varphi'(y_p)\varphi''(y_p)\right]}. \end{aligned} \tag{3}$$

To make it optimal, we consider the following approximations of first and second derivatives:

$$\begin{aligned} \varphi'(y_p) &\approx \varphi'(x_p), \\ \varphi''(y_p) &\approx \frac{4\varphi(x_p) + 10\varphi(y_p)}{y_p - x_p}, \\ &= \eta(x_p, y_p). \end{aligned} \tag{4}$$

Using the above approximations in (3), we gain new optimal root-finding algorithms having the following iterative form:

Algorithm 1.For a given x_0, compute the approximate solution x_{p+1} by the following iterative schemes:

$$\begin{aligned} y_p &= x_p - \frac{\varphi(x_p)}{\varphi'(x_p)}, \quad p = 0, 1, 2, \ldots, \\ x_{p+1} &= y_p - \frac{\varphi(y_p)}{\varphi'(y_p)} - \frac{\varphi^2(y_p)\eta(x_p, y_p)}{2\left[\varphi'^3(y_p) - \varphi(y_p)\varphi'(y_p)\eta(x_p, y_p)\right]}, \end{aligned} \tag{5}$$

which is fourth-order optimal root-finding algorithm for nonlinear scalar equations and it utilizes three functional evaluations per iteration. It should be noted that all the terms that appeared in the denominator of Algorithm 1 must not be zero; otherwise, the method will fail to find the approximate solution to the given problem. To achieve better convergence, we utilize the same idea as described above and add one more step as Newton's method which results in the following iterative method:

$$\begin{aligned} y_p &= x_p - \frac{\varphi(x_p)}{\varphi'(x_p)}, p = 0, 1, 2, \ldots, \\ z_p &= y_p - \frac{\varphi(y_p)}{\varphi'(x_p)} - \frac{\varphi^2(y_p)\eta(x_p, y_p)}{2\left[\varphi'^3(x_p) - \varphi(y_p)\varphi'(x_p)\eta(x_p, y_p)\right]}, \\ x_{p+1} &= z_p - \frac{\varphi(z_p)}{\varphi'(z_p)}. \end{aligned} \tag{6}$$

The above iteration scheme gains optimal order but does not fulfill the Kung and Traub's conjecture [19]; to fulfill this conjecture, we consider the following approximations:

$$\varphi'(z_p) \approx \frac{\varphi'(x_p)}{G(u_p, v_p, w_p)}, \tag{7}$$

where

$$\begin{aligned} G &= G(u_p, v_p, w_p), \\ &= 1 + 2u_p\left(1 + 3u_p + 3u_p^2\right) + v_p + 4w_p, \\ u_p &= \frac{\varphi(y_p)}{\varphi(x_p)}, v_p = \frac{\varphi(z_p)}{\varphi(y_p)}, w_p = \frac{\varphi(z_p)}{\varphi(x_p)}. \end{aligned} \tag{8}$$

With the help of the above approximations in (6), we are able to suggest the following algorithm:

Algorithm 2.For a given x_0, compute the approximate solution x_{p+1} by the following iterative schemes:

$$\begin{aligned} y_p &= x_p - \frac{\varphi(x_p)}{\varphi'(x_p)}, \quad p = 0, 1, 2, \ldots, \\ z_p &= y_p - \frac{\varphi(y_p)}{\varphi'(x_p)} - \frac{\varphi^2(y_p)\eta(x_p, y_p)}{2\left[\varphi'^3(x_p) - \varphi(y_p)\varphi'(x_p)\eta(x_p, y_p)\right]}, \\ x_{p+1} &= z_p - G\frac{\varphi(z_p)}{\varphi'(x_p)}, \end{aligned}$$

$$\text{where } G = G(u_p, v_p, w_p) = 1 + 2u_p\left(1 + 3u_p + 3u_p^2\right) + v_p + 4w_p, \tag{9}$$

which is a three-step optimal root-finding algorithm, having sixth convergence order and utilizing only four functional evaluations per iteration. One must keep in mind that all the terms that appeared in the denominator of Algorithm 2 must not be vanished at the initial guess in the given domain; otherwise, the method will not work properly to find the approximate solution of the given problem.

3. Convergence Analysis

This section of the paper contains the convergence analysis of the suggested iteration methods.

Theorem 1. *Suppose α be the actual root of equation $\varphi(x) = 0$. If $\varphi(x)$ is differentiable near α, Algorithm 1 is of fourth-order convergence.*

Proof. To prove the theorem, suppose α be the exact root of the equation $\varphi(x) = 0$ and e_p be p th-iteration's error, where $e_p = x_p - \alpha$, and with the help of Taylor's series expansion around $x = \alpha$, we obtain

$$\begin{aligned} \varphi(x_p) &= \varphi'(\alpha)e_p + \frac{1}{2!}\varphi''(a)e_p^2 + \frac{1}{3!}\varphi'''(a)e_p^3 + \frac{1}{4!}\varphi^{(vi)}(a)e_p^4 + \frac{1}{5!}\varphi^{(v)}(a)e_p^5 \\ &\quad + \frac{1}{6!}\varphi^{(vi)}(\alpha)e_p^6 + O(e_p^7), \end{aligned} \tag{10}$$

$$\begin{aligned} \varphi(x_p) &= \varphi'(a)\left[e_p + d_2e_p^2 + d_3e_p^3 + d_4e_p^4 + d_5e_p^5 + d_6e_p^6 + O(e_p^7)\right], \\ \varphi'(x_p) &= \varphi'(\alpha)\left[1 + 2d_2e_p + 3d_3e_p^2 + 4d_4e_p^3 + 5d_5e_p^4 + 6d_6e_p^5 + 7d_7e_p^6 + O(e_p^7)\right], \end{aligned} \tag{11}$$

where

$$\mathrm{d}_p = \frac{1}{p!}\frac{\varphi^{(p)}(\alpha)}{\varphi'(\alpha)}. \tag{12}$$

With the help of (10) and (11), we get

$$y_p = \varphi'(\alpha)\left[\alpha + d_2e_p^2 + \left(2d_3 - 2d_2^2\right)e_p^3 + \left(3d_4 - 7d_2d_3 + 4d_2^3\right)e_p^4 + \left(-6d_3^2 + 20d_3d_2^2 - 10d_2d_4 + 4d_5 - 8d_2^4\right)e_p^5 + \left(-17d_4d_3 + 28d_4d_2^2 - 13d_2d_5 + 5d_6 + 33d_2d_3^2 - 52d_3d_2^3 + 16d_2^5\right)e_p^6 + O\left(e_p^7\right)\right]. \tag{13}$$

$$\varphi\left(y_p\right) = \varphi'(\alpha)\left[d_2e_p^2 + \left(2d_3 - 2d_2^2\right)e_p^3 + \left(5d_2^3 - 7d_2d_3 + 4d_4\right)e_p^4 + \left[-24d_3^2 + 12d_3d_2^4 - 10d_2d_4 + 4d_5 - 6d_3^2\right)e_p^5 + \left(-73d_4d_2^3 + 34d_4d_2^2 - 28d_2^5 + 37d_2d_3^2 + 17d_4d_3 \; 13d_2d_5 + 5d_6\right)e_p^6 + O\left(e_p^7\right)\right]. \tag{14}$$

With the help of (10), (13), and (14), we get

$$\eta\left(x_p, y_p\right) = \varphi'(\alpha)\left[4e^{-1} + 22d_2 + \left(40d_3 + 4d_2^2\right)e_p + \left(26d_2d_3 - 4d_2^3 + 58d_4\right)e_p^2 + \left(76d_5 + 44d_2d_4 + 36d_3^2 - 40d_3d_2^2 + 4d_2^4\right)e_p^3 + (94d_6 + 62d_2d_5 + 118d_4d_3 - 58d_4d_2^2 - 102d_2d_3^2 + 64d_3d_2^3 - 4d_2^5\right)e_p^4 + (112d_7t + n80qd_2hd_{6+}x1647d_5Cd_3) - 76d_5d_2^2 + 96d_4^2 + 72d_4d_2^3 + 252d_3^2d_2^2 - 108d_3d_2^4 - 288d_4d_2d_3 - 76d_3^3 + 4d_2^6\right)e_p^5 + \left(130d_8 + 98d_2d_7 - 314d_4d_3^2 + 378d_2d_3^3 - 596d_3^2d_2^3 + 192d_3d_2^5 - 200d_2d_4^2 + 266d_4d_5 - 76d_4d_2^4 + 90d_5d_2^3 - 94d_6d_2^2 - 372d_3d_2d_5 + 596d_3d_4d_2^2 - 4d_2^7 210d_3d_6\right)e_p^6 + O\left(e_p^7\right)\right]. \tag{15}$$

Using equations (10)–(15) in Algorithm 1, we obtain

$$x_{p+1} = \alpha - \left(8d_2^3 + d_2d_3\right)e_p^4 + O\left(e^5\right), \tag{16}$$

which implies that

$$e_{p+1} = \alpha - \left(8d_2^3 + d_2d_3\right)e_p^4 + O\left(e^5\right). \tag{17}$$

The above equations confirms that Algorithm 1 is of fourth-order convergence. □

Theorem 2. *Suppose α be the actual root of $\varphi(x) = 0$. If $\varphi(x)$ is differentiable near α, Algorithm 2 is of sixth-order convergence.*

Proof. From equations (10)–(15) with the same assumptions of the previous theorem, we have

$$\eta\left(x_p, y_p\right) = \varphi'(\alpha)\left[2d_2 + (6d_2d_3 - 2d_4)e_p^2 + \left(12d_3^2 - 12d_3d_2^2 + 4d_2d_4 - 4d_5\right)e_p^3 + (2d_2d_5 - 26d_3d_4 - 42d_2d_3^2 + 24d_3d_2^3 + 2d_4d_2^2 - 6d_6\right)e_p^4 + \left(-48d_4d_2d_3 + 12d_4^2 - 24d_4d_2^3 + 28d_5d_3 + 4d_5d_2^2 + 120d_3^2d_2^2 - 48d_3d_2^4 - 8d_7 - 36d_3^3\right)e_p^5 + \left(-60d_5d_2d_3 + 28d_4d_3d_2^2 - 2d_2d_7 + 22d_5d_4 - 10d_5d_2^3 + 30d_6d_3 + 6d_6d_2^2 + 20d_2d_4^2 - 86d_4d_3^2 + 88d_4d_2^4 + 198d_2d_3^3 - 312d_3^2d_2^3 + 96d_3d_2^5 - 10d_8\right)e_p^6 + O\left(e_p^7\right)\right]. \tag{18}$$

$$\text{where } G = G\left(u_p, v_p, w_p\right) = 1 + 2u_p\left(1 + 3u_p + 3u_p^2\right) + v_p + 4w_p,$$

$$\begin{aligned}
z_p = \varphi'(\alpha)\Big[\alpha &+ \left(-8d_2^3 - c2d_3\right)e_p^4 + \left(-2d_2d_4 - 46d_3d_2^2 + 22d_2^4 - 2d_3^2\right)e_p^5 + \left(-3d_2d_5 - 7d_4d_3\right. \\
&-69d_4d_2^2 - 90d_2d_3^2 + 124d_3d_2^3 - 118d_2^5\Big)e_p^6 + \left(-272d_4d_2d_3 - 4d_2d_6 - 10d_5d_3 - 92d_5d_2^2\right. \\
&+164d_4d_2^3 + 220d_3^2d_2^2 - 1028d_3d_2^4 - 6d_4^2 - 60d_3^3 + 388d_2^6\Big)e_p^7 + \left(-364d_3d_2d_5 + 529d_3d_4d_2^2\right. \\
&- 5d_2d_7 - 274d_4d_3^2 + 93d_2d_3^3 - 3681d_3^2d_2^3 + 3586d_3d_2^5 - 206d_2d_4^2 - 17d_4d_5 - 1502d_4d_2^4 \\
&\left.+203d_5d_2^3 - 115d_6d_2^2 - 13d_3d_6 - 1743d_2^7\right)e_p^8 + O\left(e_p^9\right)\Big],
\end{aligned} \tag{19}$$

$$\begin{aligned}
\varphi\left(z_p\right) = \varphi'(\alpha)\Big[&\left(-8d_2^3 - c2d_3\right)e_p^4 + \left(-2d_2d_4 - 46d_3d_2^2 + 22d_2^4 - 2d_3^2\right)e_p^5 + \left(-3d_2d_5 - 7d_4d_3\right. \\
&-69d_4d_2^2 - 90d_2d_3^2 + 124d_3d_2^3 - 118d_2^5\Big)e_p^6 + \left(-272d_4d_2d_3 - 4d_2d_6 - 10d_5d_3 - 92d_5d_2^2\right. \\
&+164d_4d_2^3 + 220d_3^2d_2^2 - 1028d_3d_2^4 - 6d_4^2 - 60d_3^3 + 388d_2^6\Big)e_p^7 + \left(-364d_3d_2d_5 + 529d_3d_4d_2^2\right. \\
&- 5d_2d_7 - 274d_4d_3^2 + 93d_2d_3^3 - 3681d_3^2d_2^3 + 3586d_3d_2^5 - 206d_2d_4^2 - 17d_4d_5 - 1502d_4d_2^4 \\
&\left.+203d_5d_2^3 - 115d_6d_2^2 - 13d_3d_6 - 1743d_2^7\right)e_p^8 + O\left(e_p^9\right)\Big],
\end{aligned} \tag{20}$$

$$\begin{aligned}
u_p = \varphi'(\alpha)\Big[\alpha &+ \left(2d_3 - 3d_2^2\right)e_p^4 + \left(8d_2^3 - 10d_2d_3 + 3d_4\right)e_p^5 + \left(37d_3d_2^2 - 20d_2^4 - 14d_2d_4\right. \\
&\left.+4d_5 - 8d_3^2\right)e_p^6 + \left(-118d_3d_2^3 + 51d_4d_2^2 + 48d_2^5 + 55d_2d_3^2 - 18d_2d_5 + 5d_6 22d_4d_3\right)e_p^5 \\
&+\left(150d_4d_2d_3 + 6d_7 - 22d_2d_6 - 28d_5d_3 + 65d_5d_2^2 - 163d_4d_2^3 - 252d_3^2d_2^2 + 344d_3d_2^4\right. \\
&\left.-15d_4^2 + 26d_3^3 - 112d_2^6\right)e_p^6 + \left(190d_3d_2d_5 - 693d_3d_4d_2^2 + 7d_8 - 26d_2d_7 + 105d_4d_3^2\right. \\
&- 228d_2d_3^3 + 952d_3^2d_2^3 - 944d_3d_2^5 + 102d_2d_4^2 - 38d_4d_5 + 480d_4d_2^4 - 207d_5d_2^3 + 79d_6d_2^2 \\
&\left.-34d_3d_6 + 256d_2^7\right)e_p^7 + \left(258d_4d_2d_5 - 936d_4d_2d_3^2 + 2660d_4d_3d_2^3 - 876d_5d_3d_2^2 + 230d_6d_2d_3\right. \\
&+ 8d_9 - 30d_2d_8 - 46d_4d_6 + 141d_4^2d_3 - 477d_4^2d_2^2 - 1336d_4d_2^5 + 132d_5d_3^2 + 607d_5d_2^4 - 40d_7d_3 \\
&\left.+93d_7d_2^2 - 251d_6d_2^3 + 1254d_3^3d_2^2 - 3200d_3^2d_2^4 + 2480d_3d_2^6 - 24d_5^2 - 72d_3^4 - 576d_2^8\right)e_p^8 \\
&+O\left(e_p^9\right)\Big],
\end{aligned} \tag{21}$$

$$\begin{aligned}
v_p = \varphi'(\alpha)\Big[&\left(-8d_2^2 - c3\right)e_p^4 + \left(-2d_4 - 32d_2d_3 + 6d_2^3\right)e_p^5 + \left(-66d_2^4 - 3d_3d_2^2 - 49d_2d_4\right. \\
&\left.-33d_3^2 - 3d_5\right)e_p^6 + \left(-102d_4d_3 - 4d_6 - 66d_2d_5 - 22d_4d_2^2 - 88d_2d_3^2 - 520d_3d_2^3 + 130d_2^5\right)e_p^5 \\
&+\left(-138d_5d_3 - 339d_4d_2d_3 - 5d_7 - 117d_3^3 - 1645d_3^2d_2^2 771d_3d_2^4 - 79d_4^2 - 795d_4d_2^3 - 42d_5d_2^2\right. \\
&\left.-83d_2d_6 - 793d_2^6\right)e_p^6 + \left(-214d_4d_5 - 608d_4d_3^2 - 5138d_3d_4d_2^2 - 506d_3d_2d_5 - 174d_3d_6 - 6d_8\right. \\
&+ 312d_2d_4^2 + 914d_4d_2^4 - 1066d_5d_2^3 - 100d_2d_7 - 62d_6d_2^2 - 2464d_2d_3^3 + 828d_3^2d_2^3 - 8702d_3d_2^5 \\
&\left.+2686d_2^7\right)e_p^7 + \left(-915d_4d_2d_5 - 869d_5d_3^2 - 6964d_5d_3d_2^2 - 210d_7d_3 - 270d_4d_6 - 673d_6d_2d_3\right. \\
&- 1038d_4^2d_3 - 11738d_4d_2d_3^2 + 347d_4d_3d_2^3 - 7d_9 + 28970d_3d_2^6 - 145d_5^2 + 1030d_5d_2^4 - 117d_2d_8 \\
&- 82d_7d_2^2 - 1338d_6d_2^3 - 4034d_4^2d_2^2 - 12825d_4d_2^5 - 1453d_3^4 - 3392d_3^3d_2^2 - 41833d_3^2d_2^4 \\
&\left.-13161d_2^8\right)e_p^8 + O\left(e_p^9\right)\Big],
\end{aligned} \tag{22}$$

$$\begin{aligned}
w_p = \varphi'(\alpha)\Big[&\left(-8d_2^3 - c2d_3\right)e_p^4 + \left(-2d_2d_4 - 45d_3d_2^2 + 30d_2^4 - 2d_3^2\right)e_p^5 + \left(177d_3d_2^3 - 87d_2d_3^2 - 3d_2d_5\right. \\
&\left.-7d_4d_3 - 67d_4d_2^2 - 148d_2^5\right)e_p^6 + \left(239d_4d_2^3 - 262d_4d_2d_3 - 4d_2d_6 - 10d_5d_3 - 89d_5d_2^2 + 352d_3^2d_2^2e_p^5\right. \\
&\left.+1235d_3d_2^4 - 6d_4^2 - 58d_3^3 + 536d_2^6\right)e_p^6 + \left(-350d_3d_2d_5 + 903d_3d_4d_2^2 - 5d_2d_7 - 265d_4d_3^2 + 238d_2d_3^3\right. \\
&\left.- 4209d_3^2d_2^3 + 4985d_3d_2^5 - 198d_2d_4^2 - 17d_4d_5 - 1771d_4d_2^4 + 300d_5d_2^3 - 111d_6d_2^2 - 13d_3d_6 - 2215d_2^7\right)e_p^7 \\
&+\left(-530d_4d_2d_5 + 768d_4d_2d_3^2 - 12203d_4d_3d_2^3 + 1096d_5d_3d_2^2 - 438d_6d_2d_3 - 6d_2d_8 - 22d_4d_6 - 405d_4^2d_3\right. \\
&+ 563d_4^2d_2^2 + 6931d_4d_2^5 - 356d_5d_3^2 - 2300d_5d_2^4 - 16d_7d_3 - 133d_7d_2^2 + 361d_6d_2^3 - 7436d_3^3d_2^2 \\
&\left.+18656d_3^2d_2^4 - 25327d_3d_2^6 - 12d_5^2 + 4d_3^4 + 8609d_2^8\right)e_p^8 + O\left(e_p^9\right)\Big],
\end{aligned} \tag{23}$$

$$
\begin{aligned}
G\left(u_p, v_p, w_p\right) = \varphi'(\alpha)\Big[&1 + 2d_2 e_p + \left(3d_3 - 8d_2^2\right)e_p^4 + \left(-40d_2^3 - 32d_2d_3 + 4d_4\right)e_p^5 + \left(5d_5 - 49d_2 d\right. \\
&-265d_3d_2^2 + 33d_3^2 + 110d_2^4\big)e_p^6 + \big(6d_6 - 66d_2d_5 - 102d_4d_3 - 410d_4d_2^2 - 590d_2d_3^2 \\
&+552d_3d_2^3 - 588d_2^5\big)e_p^6 + \big(-1831d_4d_2d_3 + 7d_7 - 83d_2d_6 - 138d_5d_3 - 556d_5d_2^2 \\
&+663d_4d_2^3 + 615d_3^2d_2^2 - 5023d_3d_2^4 - 79d_4^2 - 441d_3^3 + 1421d_2^6\big)e_p^7 + \big(-2486d_3d_2d_5 \\
&+784d_3d_4d_2^2 + 8d_8 - 100d_2d_7 - 2058d_4d_3^2 - 672d_2d_3^3 - 18094d_3^2d_2^3 + 11042d_3d_2^5 \\
&-1422d_2d_4^2 - 214d_4d_5 - 7286d_4d_2^4 + 776d_5d_2^3 - 702d_6d_2^2 - 174d_3d_6 - 5182d_2^7\big)e_p^7 \\
&+\big(-3863d_4d_2d_5 - 5330d_4d_2d_3^2 - 53693d_4d_3d_2^3 + 348d_5d_3d_2^2 - 3141d_6d_2d_3 + 9d_9 \\
&-117d_2d_8 - 270d_4d_6 - 3204d_4^2d_3 - 228d_4^2d_2^2 + 14345d_4d_2^5 - 2797d_5d_3^2 - 9554d_5d_2^4 \\
&-210d_7d_3 - 848d_7d_2^2 + 888d_6d_2^3 - 35200d_3^3d_2^2 + 28233d_3^2d_2^4 - 58240d_3d_2^6 - 145d_5^2 \\
&-1149d_3^4 + 14579d_2^8\big)e_p^8 + O\left(e_p^9\right)\Big].
\end{aligned} \tag{24}
$$

Using equations (10)–(24) in Algorithm 2, we get

$$x_{p+1} = \alpha + \left(-8d_3d_2^3 - 64d_2^5\right)e_p^6 + O\left(e^7\right), \tag{25}$$

which implies that

$$e_{p+1} = \alpha + \left(-8d_3d_2^3 - 64d_2^5\right)e_p^6 + O\left(e^7\right). \tag{26}$$

The above equations confirm that Algorithm 2 is of sixth-order convergence. □

4. Numerical Comparisons and Applications

To demonstrate the applicability and effectiveness of our newly devised iterative approaches, we present five real-world engineering problems and one very important and well-known nonlinear problem in this section. The devised iterative approaches are compared to the following existing two-step iterative algorithms:

4.1. Noor's Method One (NM1). For the given initial guess x_0, calculate the approximate solution x_{p+1} using the iteration schemes as follows:

$$
\begin{aligned}
x_{p+1} &= x_p - \frac{\varphi\left(x_p\right)}{\varphi'\left(x_p\right)}, \quad p = 0, 1, 2, 3, \ldots, \\
x_{p+1} &= x_p - \frac{\varphi\left(x_p\right)}{\varphi'\left(x_p\right)} + \left[\frac{\varphi\left(x_p\right)}{\varphi'\left(x_p\right)}\right]\frac{\varphi'\left(y_p\right)}{\varphi'\left(x_p\right)},
\end{aligned} \tag{27}
$$

which is the second-order convergent method, known as Noor's method one [11] for solving nonlinear scalar equations.

4.2. Chun's Method (CM). For a given initial guess x_0, determine the approximate root x_{p+1} with the iteration schemes:

$$
\begin{aligned}
y_p &= x_p - \frac{\varphi\left(x_p\right)}{\varphi'\left(x_p\right)}, \quad p = 0, 1, 2, 3, \ldots, \\
x_{p+1} &= x_p - \frac{\varphi\left(x_p\right)}{\varphi'\left(x_p\right)} + \left[1 + \frac{\varphi\left(y_p\right)}{\varphi\left(x_p\right)} + 2\left(\frac{\varphi\left(y_p\right)}{\varphi\left(x_p\right)}\right)^2\right],
\end{aligned} \tag{28}
$$

which is the fourth-order convergent Chun's method [20] for solving nonlinear scalar equations.

4.3. Chand's Method (CHM). For the given initial guess x_0, calculate the approximate solution x_{p+1} using the iteration schemes:

$$
\begin{aligned}
y_p &= x_p - \frac{\varphi\left(x_p\right)}{\varphi'\left(x_p\right)}, \quad p = 0, 1, 2, 3, \ldots, \\
x_{p+1} &= x_p - \frac{\varphi\left(x_p\right) + \varphi\left(y_p\right)}{\varphi'\left(x_p\right)} + \left[1 + 2\left(\frac{\varphi\left(y_p\right)}{\varphi\left(x_p\right)}\right)^2\right],
\end{aligned} \tag{29}
$$

which is the fourth-order two-step Chand's method [16] for solving nonlinear scalar equations.

4.4. Noor's Method Two (NM2). For the given initial guess x_0, calculate the approximate solution x_{p+1} using the iteration schemes:

$$
\begin{aligned}
y_p &= x_p - \frac{\varphi\left(x_p\right)}{\varphi'\left(x_p\right)}, \quad p = 0, 1, 2, 3, \ldots, \\
z_p &= -\frac{\varphi\left(x_p\right)}{\varphi'\left(x_p\right)}, \\
x_{p+1} &= x_p - \frac{\varphi\left(x_p\right)}{\varphi'\left(x_p\right)} + \left[1 + \frac{\varphi\left(y_p + z_p\right)}{\varphi\left(x_p\right)}\right],
\end{aligned} \tag{30}
$$

which is three-step cubic-order Noor's method two [13] for solving nonlinear scalar equations.

4.5. Yun's Method (YM). For the given initial guess x_0, calculate the approximate solution x_{p+1} using the iteration schemes:

$$y_p = x_p - \frac{\varphi(x_p)}{\varphi'(x_p)}, \quad p = 0, 1, 2, 3, \ldots,$$

$$z_p = -\frac{\varphi(y_p)}{\varphi'(x_p)}, \tag{31}$$

$$x_{p+1} = x_p - \frac{\varphi(x_p)}{\varphi'(x_p)} + -\frac{\varphi(y_p)}{\varphi'(x_p)} - \frac{\varphi(y_p + z_p)}{\varphi(x_p)},$$

which is the three-step Yun's method [21] for solving nonlinear scalar equations, having the convergence of fourth order. We evaluate the following test examples to conduct a numerical comparison of the above-described methods with our proposed algorithms:

Example 1. Kinetic problem equation.

The equation of kinetic problem has the following form:

$$e^{21000/T} = 1.11 \times 10^{11} T^2, \tag{32}$$

Where T represents the temperature of the given system. (32) has been derived from the stirred reactor with the cooling coils [22]. By taking $T = x$, (32) may be rewritten in form of the following nonlinear function:

$$\varphi_1(x) = x^{-2} e^{21000/x} - 1.11 \times 10^{11}, \tag{33}$$

which can be used to find the temperature of the system. We start the iteration process with the initial guess $x_0 = 430$, and the related results from various iteration techniques are shown in Table 1.

Example 2. Planck's radiation law.

The Planck's radiation law [23] is used to compute the energy density within an isothermal black body with the standard form as

$$\varphi(\sigma) = \frac{8\pi cP}{\sigma^5 \left(e^{cP/\sigma kT} - 1\right)}. \tag{34}$$

Assume that we want to determine the wavelength sigma_1 that corresponds to maximal energy density $\varphi(\sigma_1)$. We use $x = cP/\sigma kT$ to turn the aforementioned problem into a nonlinear equation, which has the following nonlinear equation:

$$1 - \frac{x}{5} = e^{-x}, \tag{35}$$

which can be converted into the form of nonlinear function as follows:

$$\varphi_2(x) = e^{-x} + \frac{x}{5} - 1. \tag{36}$$

The maximal wavelength of the radiation is represented by the estimated root of the aforementioned function φ_2. To begin the iteration process, we take the starting guess $x_0 = 0.2$, and the relevant results from various iteration methods are shown in Table 2.

Example 3. Adiabatic flame temperature equation.

The equation of adiabatic flame temperature has the following form:

$$\varphi_3(x) = \Delta\mathrm{H} - a_1(298 - x) - \frac{a_2}{2}\left(298^2 - x^2\right) - \frac{a_3}{3}\left(298^3 - x^3\right), \tag{37}$$

where we take $\Delta\mathrm{H} = -57798, a_1 = 7.256, a_2 = 0.002298$, and $a_3 = 0.00000283$. For more information, one may see [24, 25] and the references are cited therein. The aforementioned function φ_3 is actually a third-degree polynomial, and according to algebra's fundamental theorem, it must have three unique roots (zeros) and $\alpha = 4305.30991366612556304019892945$ is a simple one among them that we estimated using the proposed algorithms with the starting guess $x_0 = 2000$, and the numerical results are presented in Table 3.

Example 4. Beam designing model.

We consider the problem of beam positioning from [26], which yields a nonlinear function as

$$\varphi_4(x) = x^4 + 4x^3 - 24x^2 + 16x + 16. \tag{38}$$

The given function φ_4 is actually a four-degree polynomial, and it must have precisely four roots (zeros) in the light of fundamental theorem of algebra. We choose the starting guess $x_0 = -0.75$ to approximate the required root using the proposed algorithms, and the numerical results are shown in Table 4.

Example 5. Open channel flow problem.

In fluid dynamics, Manning's equation (27) deals with the water flow with the following standard form:

$$\text{water flow} = F = \frac{\sqrt{\mathrm{s}}\mathrm{ar}^{2/3}}{N}. \tag{39}$$

In (39), the symbol s stands for the slope, a stands for the area, r stands for the hydraulic radius, and n stands for Manning's roughness coefficient. For a channel with the rectangular-shape of width w and depth x, we may have

$$a = wx, \& r = \frac{wx}{w + 2x}. \tag{40}$$

Using these values in (39), we obtain

$$F = \frac{\sqrt{s}wx}{N}\left(\frac{wx}{w + 2x}\right)^{2/3}. \tag{41}$$

In order to compute water's depth in a channel, we rewrite (41) in the following nonlinear form:

Table 1: Numerical comparison of various root-finding algorithms for the problem φ_1.

Method $\varphi_1(x)$, $x_0 = 430.000$	N	x_{p+1}	$\lvert\varphi(x_{p+1})\rvert$	$\sigma = \lvert x_{p+1} - x_p\rvert$	CPU time
NM1	12	551.77382493032659963621586600754 0	$7.093691e-23$	$4.808708e-16$	3.065
CM	10	551.773824930326599636215866007312	$1.837158e-18$	$9.899467e-07$	3.128
CHM	10	551.773824930326599636215866007540	$2.258490e-46$	$1.218702e-13$	3.190
NM2	15	551.773824930326599636215866007540	$2.913370e-27$	$3.081698e-18$	3.253
YM	10	551.773824930326599636215865875416	$1.064742e-15$	$4.946704e-06$	3.331
Algorithm 1	9	551.773824930326599636215866007540	$1.608420e-26$	$1.509462e-08$	2.626
Algorithm 2	8	551.773824930326599636215866007540	$8.779657e-72$	$2.409901e-13$	2.704

Table 2: Numerical comparison of various root-finding algorithms for the problem φ_2.

Method $\varphi_2(x)$, $x_0 = 0.20$	N	x_{p+1}	$\lvert\varphi(x_{p+1})\rvert$	$\sigma = \lvert x_{p+1} - x_p\rvert$	CPU time
NM1	5	0.000000000000000000000003947489	$3.157991e-24$	$2.513162e-12$	3.521
CM	3	−0.000000000000000000000000000000	$1.571613e-40$	$1.158532e-10$	3.600
CHM	3	−0.000000000000000000000000000000	$1.750456e-48$	$1.380631e-12$	3.663
NM2	5	−0.000000000000000000000000019832	$1.586544e-26$	$1.781316e-13$	3.707
YM	3	−0.000000000000000000000000000000	$2.329495e-42$	$4.155449e-11$	3.403
Algorithm 1	3	0.000000000000000000000000000000	$2.160782e-52$	$2.134128e-13$	2.934
Algorithm 2	2	0.000000000000000000000232846266	$1.862770e-22$	$1.814949e-04$	2.986

Table 3: Numerical comparison of various root-finding algorithms for the problem φ_3.

Method $\varphi_3(x)$, $x_0 = 2000.00$	N	x_{p+1}	$\lvert\varphi(x_{p+1})\rvert$	$\sigma = \lvert x_{p+1} - x_p\rvert$	CPU time
NM1	11	4305.30991366612556302092275417990 9	$4.316939e-16$	$4.270236e-07$	3.281
CM	4	4305.309913666125563040198929446342	$2.174776e-55$	$6.493737e-12$	3.329
CHM	3	4305.309913666125563040198929446342	$5.855416e-30$	$1.704358e-05$	3.391
NM2	6	4305.309913666125563040199102978796	$3.886295e-21$	$1.281244e-09$	3.322
YM	4	4305.309913666125563040198929446342	$5.772631e-71$	$8.594129e-16$	3.468
Algorithm 1	4	4305.309913666125563040198928972873	$1.060344e-23$	$1.332626e-05$	2.516
Algorithm 2	3	4305.309913666125563040116005825452	$1.857091e-18$	$6.740210e-01$	2.547

Table 4: Numerical comparison of various root-finding algorithms for the problem φ_4.

Method $\varphi_4(x)$, $x_0 = -0.75$	N	x_{p+1}	$\lvert\varphi(x_{p+1})\rvert$	$\sigma = \lvert x_{p+1} - x_p\rvert$	CPU time
NM1	4	−0.535898384862245344805625149808	$3.035870e-15$	$1.028354e-08$	3.650
CM	3	−0.535898384862245412945107316988	$4.071744e-45$	$2.889668e-12$	3.712
CHM	3	−0.535898384862245412945107316988	$1.105351e-48$	$4.228953e-13$	3.743
NM2	5	−0.535898384862245412945107316989	$2.255247e-29$	$8.863356e-16$	3.808
YM	3	−0.535898384862245412945107316988	$1.563471e-46$	$1.345709e-12$	3.854
Algorithm 1	3	−0.535898384862245412945838932746	$3.259623e-20$	$9.586839e-08$	2.670
Algorithm 2	2	−0.535898384862245404466362113905	$3.777599e-16$	$1.027830e-03$	2.732

$$\varphi_5(x) = \frac{\sqrt{swx}}{N}\left(\frac{wx}{w+2x}\right)^{2/3} - F. \tag{42}$$

In (42), the parameters haven been chosen as $w = 4.572m$, $s = 0.017$, $F = 14.15m^3/s$, and $N = 0.0015$. We start the iteration process with the initial guess $x_0 = 0.1$, and the related results from various iteration techniques are shown in Table 5.

Example 6. A well-known nonlinear problem.

To analyze the proposed methods, we consider the following very important and well-known nonlinear problem:

$$x = \tan x. \tag{43}$$

The above equation in the form of nonlinear function φ_6 can be rewritten as

$$\varphi_6(x) = x - \tan x. \tag{44}$$

Table 5: Numerical comparison among different algorithms for the problem φ_5.

Method $\varphi_5(x)$, $x_0 = 0.10$	N	x_{p+1}	$\lvert\varphi(x_{p+1})\rvert$	$\sigma = \lvert x_{p+1} - x_p\rvert$	CPU time
NM1	7	1.465091220295824642376020731453	$2.422027e-24$	$1.334423e-12$	2.943
CM	4	1.465091220295824642376021092486	$2.481534e-24$	$2.175366e-06$	3.391
CHM	4	1.465091220295824642376024509842	$4.889610e-23$	$4.918775e-06$	3.468
NM2	6	1.465091220295824642383819255359	$1.059172e-19$	$2.790535e-10$	3.431
YM	4	1.465091220295824642376020909779	$4.475999e-35$	$5.351749e-09$	3.593
Algorithm 1	4	1.465091220295824642376020909825	$6.277761e-28$	$3.483992e-07$	2.826
Algorithm 2	3	1.465091220295824642376020909779	$8.454782e-89$	$5.011500e-15$	2.920

We start the iteration process with the initial guess $x_0 = 1.0$, and the related results from various iteration techniques are shown in Table 6.

The machine used for computing numerical results has the following specifications:

(i) 64 bit operating system

(ii) x64-based processor with Core(TM) m3-7Y30 CPU@1.00 GHz 1.61 GHz

(iii) 8 GB of memory

We use the accuracy $\varepsilon = 10^{-15}$ in the stopping criteria $|x_{p+1} - x_p| < \varepsilon$ for the all aforementioned problems. We used the computer application Maple 13 to compute all of the numerical results and can be observed in Tables 1–6.

Tables 1–6 represent the detailed comparison of the suggested root-finding methods with the other above-described methods for the engineering and arbitrary problems φ_1–φ_6. In the columns of the above-presented tables, N stands for the consumption of the iterations for different methods, $|\varphi(x)|$ stands for the modulus value of $\varphi(x)$, x_{p+1} represents the final estimation, $|x_{p+1} - x_p|$ stands for the positive distance between the two consecutive estimations, and the last columns gives us the information about the CPU time consumption in seconds for different methods in comparison.

The careful examination of the obtained results in Tables 1–6 certifies that the proposed root-finding algorithms are showing better efficiency and performance which justified the supremacy of the suggested root-finding algorithms with respect to CPU-time consumption, accuracy, convergence-speed, no. of iterations, and computational order of convergence against the other comparable methods.

5. Dynamical Representation

In this section, we investigate the dynamical aspects of different algorithms with the aid of polynomiographs created through different algorithms for complex polynomials of different degrees. To generate polynomiographs of different complex polynomials by means of a computer program, we have to choose an initial rectangle $\mathscr{R}$ which contains the polynomial's roots. Then, corresponding to each starting point w_0 in the region, we execute an iterative process and then colour the point corresponding to w_0 that relies on the approximate convergence of the truncated orbit to a root. The discretization of $\mathscr{R}$ is completely responsible for the image's resolution quality. For instance, if we discretize $\mathscr{R}$ into a 2000 by 2000 grid, the output is a high-resolution picture.

We know that any complex polynomial q having degree n has exactly n-roots, and from the fundamental theorem of algebra, it may be uniquely defined as

$$q(w) = c_n w^n + c_{n-1} w^{n-1} + \cdots + c_1 w + c_0. \quad (45)$$

or by its zeros (roots) $\{w_1, w_2, \ldots, w_{n-1}, w_p\}$:

$$q(w) = (w - w_1)(w - w_2)\cdots(w - w_p). \quad (46)$$

where $\{c_n, c_{n-1}, \ldots, c_1, c_0\}$ are the complex coefficients.

The iterative algorithms can be easily applied to the both representations of the complex polynomial q. The polynomial's degree depicts the number of basins of attraction. The location of basins can be managed by changing the position of roots in the complex plane manually. The polynomiographs' colours depends upon the no. of iterations needed to attain the approximate solution of some polynomial with a given accuracy and a chosen scheme of iteration.

The main algorithm for drawing polynomiographs is given in Algorithm.

In Algorithm 3, the convergence test $(w_p + 1, w_p, \epsilon)$ would be considered true in case of convergence and vice versa. The following is the standard form of the widely used convergence test:

$$\left|w_{p+1} - w_p\right| < \epsilon. \quad (47)$$

In (47), w_p and w_{p+1} are the consecutive estimations in the iteration procedure, and the symbol $\varepsilon > 0$ stands for the accuracy. In this article, we also use the stopping criteria (47). We created visually appealing and intriguing polynomiographs using newly developed root-finding methods and compared them with the polynomiographs of the other similar methods. The colour of polynomiographs is determined by the no. of iterations required to estimate the roots of a polynomial with a certain precision ε. A huge number of similar graphics may be made by changing the value of the parameter k, where k specifies the maximum no. of iterations. The work on polynomiography was first initiated by Kalantri [28–30] who described its artistic applications in different fields of science and arts. Gdawiec et al. [31, 32] put forward the work of Kalantri and presented fractal patterns of polynomial root-finding methods. Scott et al. [33], in 2011, worked on the basis of attraction for different existing methods and compared them with respect to their basis. In

Table 6: Numerical comparison among different algorithms for the problem φ_6.

Method $\varphi_6(x)$, $x_0 = 0.10$	N	x_{p+1}	$\lvert\varphi(x_{p+1})\rvert$	$\sigma = \lvert x_{p+1} - x_p\rvert$	CPU time
NM1	15	0.0000260887664699846589886859O3	$5.918878e-15$	$2.809559e-05$	2.728
CM	16	0.0000281829363954098181324740 46	$7.461695e-15$	$2.714568e-05$	2.791
CHM	16	0.0000160152260875604917525299 52	$1.369235e-15$	$1.653393e-05$	2.822
NM2	22	0.0000218207031184945860979392 26	$3.463259e-15$	$1.420995e-05$	2.460
YM	16	0.0000265549786030208505301346 45	$6.241897e-15$	$2.583725e-05$	2.907
Algorithm 1	15	0.0000187707752872094881260859 80	$2.204578e-15$	$2.149290e-05$	2.600
Algorithm 2	8	0.0000181999851890705509229302 3	$2.009489e-15$	$6.032757e-05$	2.663

Input: $q \in C$—polynomial, $A \subset C$—area, k—max. no. of iterations, I—iteration method, ϵ—accuracy, colormap $[0 \ldots C-1]$—colormap with C colors.
Output: polynomiograph corresponding to polynomial q.
for $w_0 \in A$ do
  $i = 0$
  while $i \leq k$ do
    $w_{p+1} = I(w_p)$
    if $|w_{p+1} - w_p| < \epsilon$, then
      break
    $i = i + 1$
Colour w_0 by means of colormap.

Algorithm 3: Polynomiograph's generation.

this sense, the polynomiography gives us a new way to analyze the graphical behaviors of different existing methods in the literature.

We investigate the following complex polynomials for the aim of generating polynomiographs using the suggested methods, and we compare them with other well-known two-step iteration methods.

$$\begin{aligned} &q_1(w) = w^3 - 1, q_2(w) = w^3 - w.i - 1, q_3(w), \\ &\quad = w^3 + w.i + 1, q_4(w) = w^3 - w^2 + w - 1, \\ &q_5(w) = w^3 + w^2 + w + 1, q_6(w) = w^3.i - w^2 + i. \end{aligned} \tag{48}$$

The colormap that has been used for the coloring of iterations in the generation of polynomiographs is presented in Figure 1:

Example 7. Polynomiographs corresponding to polynomial $q_1(w)$ through various numerical algorithms.

In this example, we investigate and compare the dynamical results obtained through different iteration schemes with our presented algorithms by considering the cubic polynomial $w^3 - 1$ which possesses three distinct simple zeros: $1, -1/2 - \sqrt{3}/2i$, and $-1/2 + \sqrt{3}/2i$. We executed all the algorithms to achieve the simple zeros of the considered polynomials, and the results can be visualized in Figures 2–8.

Example 8. Polynomiographs corresponding to polynomial $q_2(w)$ through various numerical algorithms.

This example includes the dynamical comparison of the suggested iteration schemes with different similar-nature iterative algorithms by taking the cubic complex polynomial $w^3 - w.i - 1$. The degree of this complex polynomial is three, and according to the fundamental theorem of algebra, it has exactly three roots which are all simple and given as $(1/6)3[4 + (4/9)\sqrt{81 + 12i}\,]^{2/3} + 4i/4 + (4/9)\sqrt{81 + 12i}^{1/3}$, $(1/12)[3i(4 + (4/9)\sqrt{81 + 12i}\,)^{2/3} + 4] \quad \sqrt{3} - 3[4 + (4/9)$ $\sqrt{81 + 12i}\,]^{2/3} - 4i/4 + (4/9)\sqrt{81 + 12i}^{1/3}$, and $(1/12)[3i(4 +$ $(4/9)\sqrt{81 + 12i}\,)^{2/3} - 4] \quad \sqrt{3} - 3[4 + (4/9) \quad \sqrt{81 + 12i}\,]^{2/3} -$ $4i/4 + (4/9)\sqrt{81 + 12i}^{1/3}$. Using computer program, we executed all the iteration processes, and the corresponding results can be seen in Figures 9–15.

Example 9. Polynomiographs corresponding to polynomial $q_3(w)$ through various numerical algorithms.

We consider the complex polynomial $w^3 + w.i + 1$ in this experiment for analyzing the behaviors of different iteration schemes graphically. For this purpose, we generated the polynomiographs of the considered polynomials whose simple zeros are $(1/6)3[4 + (4/9) \quad \sqrt{81 + 12i}\,]^{2/3} + \quad 4i/4 + (4/9)$ $\sqrt{81 + 12i}^{1/3}$ and $(1/12)[3i(4 + (4/9)\sqrt{81 + 12i}\,)^{2/3} + 4]\sqrt{3} -$ $3[4 + (4/9)\sqrt{81 + 12i}\,]^{2/3} - 4i/4 + (4/9)\sqrt{81} \quad +12i^{1/3}(1/12)$ $[3i(4 + (4/9) \quad \sqrt{81 + 12i}\,)^{2/3} - 4] \quad \sqrt{3} - 3[4 + (4/9)$ $\sqrt{81 + 12i}\,]^{2/3} \quad -4i/4 + (4/9)\sqrt{81 + 12i}^{1/3}$, and the corresponding dynamic results are presented in Figures 16–22.

Example 10. Polynomiographs corresponding to polynomial $q_4(w)$ through various numerical algorithms.

In this example, we show the dynamics of different iteration schemes in the form of polynomiographs by taking the complex polynomial $w^3 - w^2 + w - 1$, whose simple zeros are 1, i, and $-i$ that can be visualized easily on the complex planes of the corresponding polynomiographs that are shown in Figures 23–29.

Example 11. Polynomiographs corresponding to polynomial $q_5(w)$ through various numerical algorithms.

In the eleventh example, we take the polynomial $w^3 + w^2 + w + 1$, having simple zeros -1, i, and $-i$. To draw the polynomiographs, we executed all the iteration schemes with the help of computer program, and the corresponding graphical objects can be visualized in Figures 30–36.

Example 12. Polynomiographs corresponding to polynomial $q_6(w)$ through various numerical algorithms.

In the last and final experiment, we show the graphical representations of different iteration schemes by generating

Figure 1: The colormap used for generating polynomiographs.

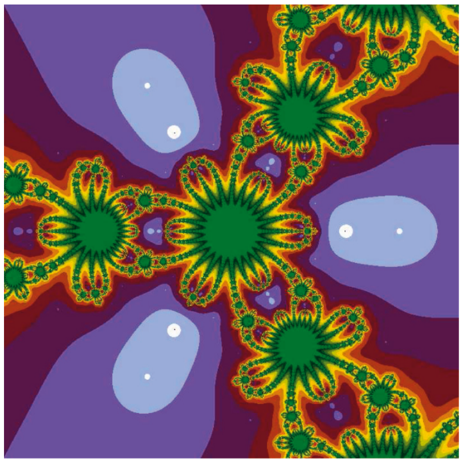

Figure 2: Polynomiograph for $q_1(w)$ using NM1.

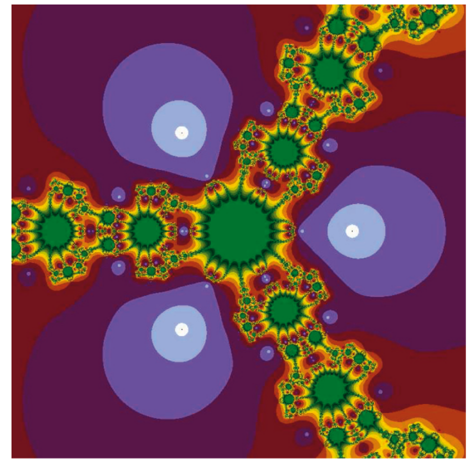

Figure 5: Polynomiograph for $q_1(w)$ using NM2.

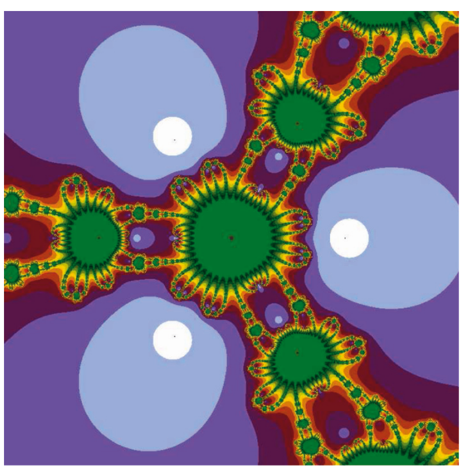

Figure 3: Polynomiograph for $q_1(w)$ using CM.

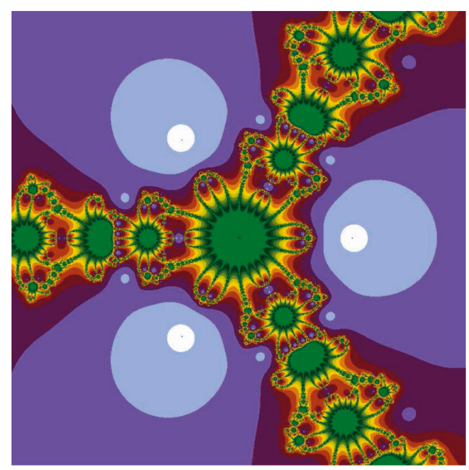

Figure 6: Polynomiograph for $q_1(w)$ using YM.

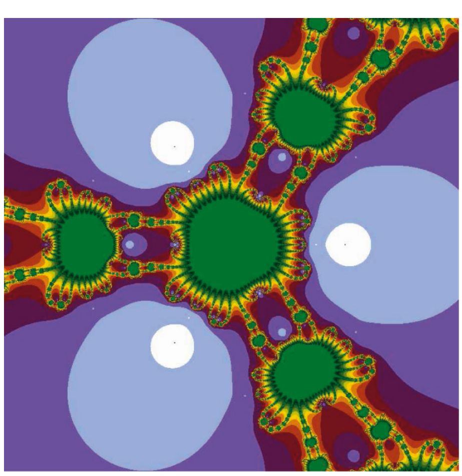

Figure 4: Polynomiograph for $q_1(w)$ using CHM.

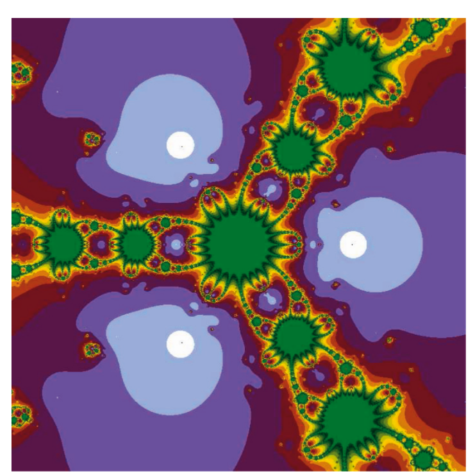

Figure 7: Polynomiograph for $q_1(w)$ using Algorithm 1.

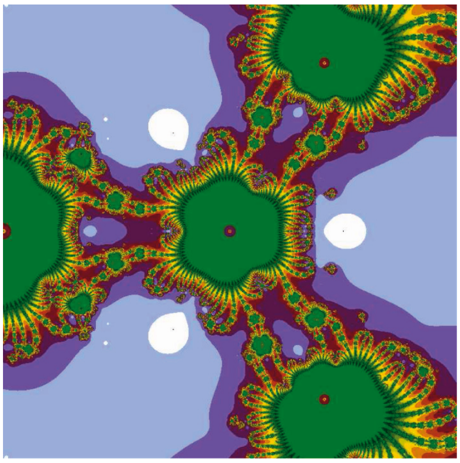

Figure 8: Polynomiograph for $q_1(w)$ using Algorithm 2.

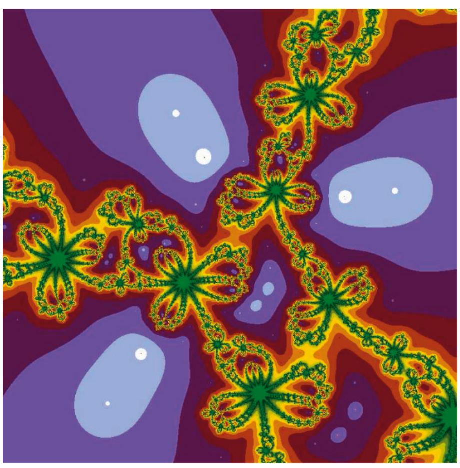

Figure 9: Polynomiograph for $q_2(w)$ using NM1.

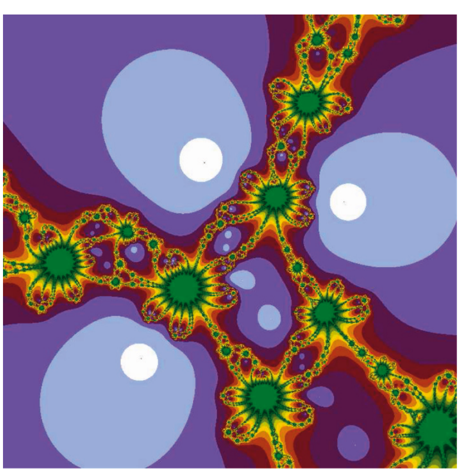

Figure 10: Polynomiograph for $q_2(w)$ using CM.

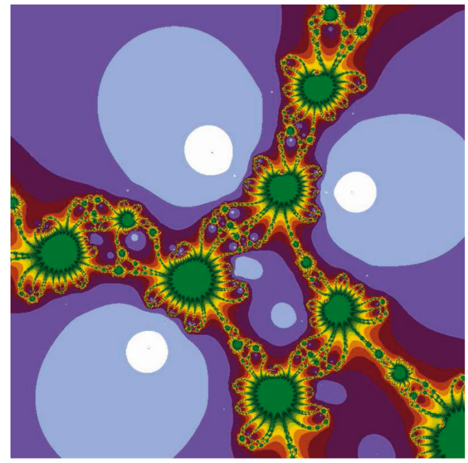

Figure 11: Polynomiograph for $q_2(w)$ using CHM.

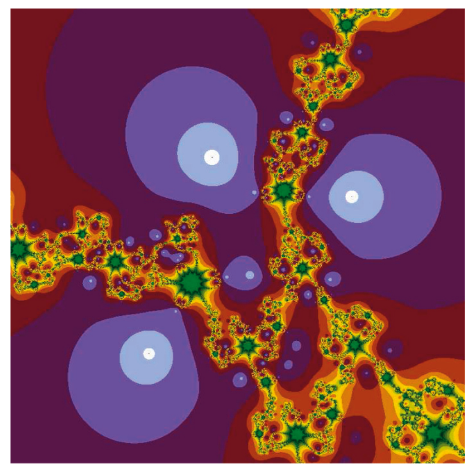

Figure 12: Polynomiograph for $q_2(w)$ using NM2.

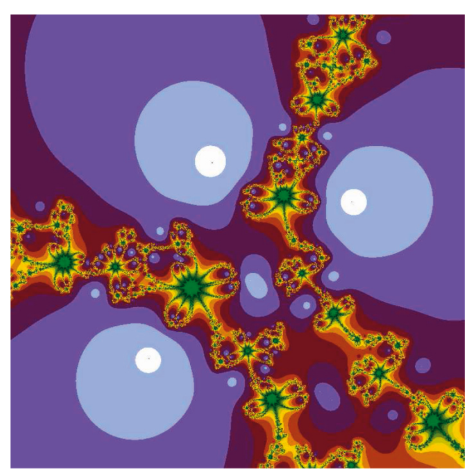

Figure 13: Polynomiograph for $q_2(w)$ using YM.

the polynomiographs of the complex polynomial $w^3.i - w^2 + i$ whose simple zeros are $[-108+8i+12\sqrt{81-12i}]^{2/3} - 4 - 2i[-108+8i+12\sqrt{81-12i}]^{1/}$ $3/6[-108+8i+12\sqrt{81-12i}]^{1/3}$, $[-108+8i+12\sqrt{81-12i}]^{2/3}+4+4i\,[-108+8i+12\sqrt{81-12i}]^{1/3}+i\sqrt{3}[-108+8i+12\sqrt{81-12i}]^{2/3}+4i\sqrt{3}/-12[-108+8i+12\sqrt{81-12i}]^{1/3}$, and $[-108+8i+$

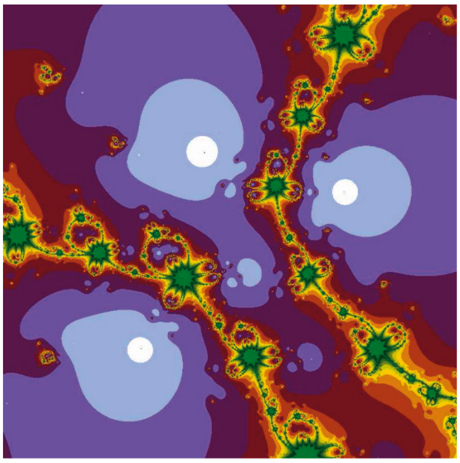

Figure 14: Polynomiograph for $q_2(w)$ using Algorithm 1.

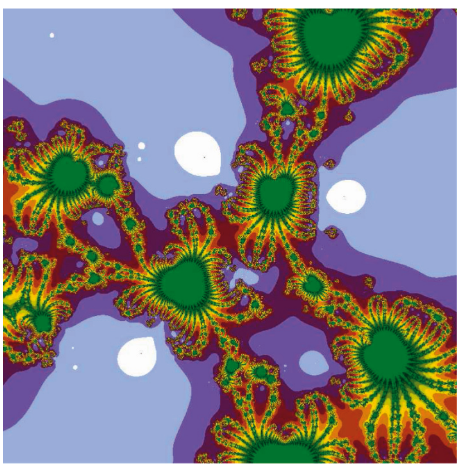

Figure 15: Polynomiograph for $q_2(w)$ using Algorithm 2.

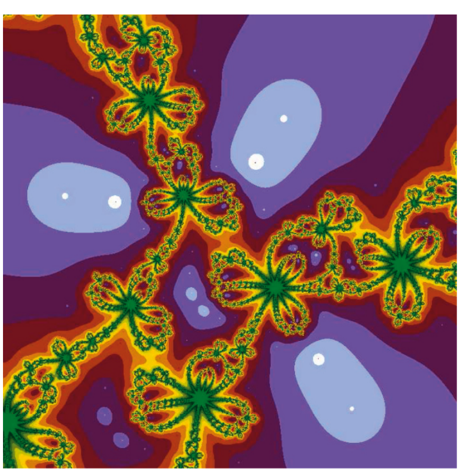

Figure 16: Polynomiograph for $q_3(w)$ using NM1.

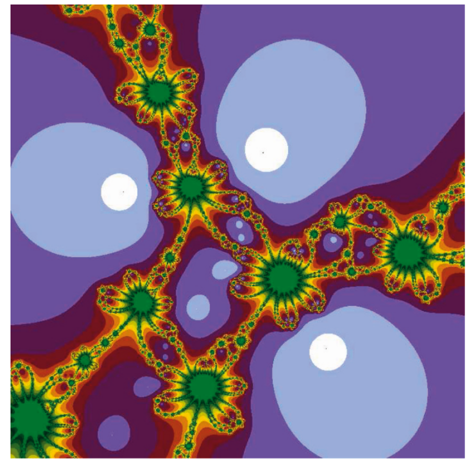

Figure 17: Polynomiograph for $q_3(w)$ using CM.

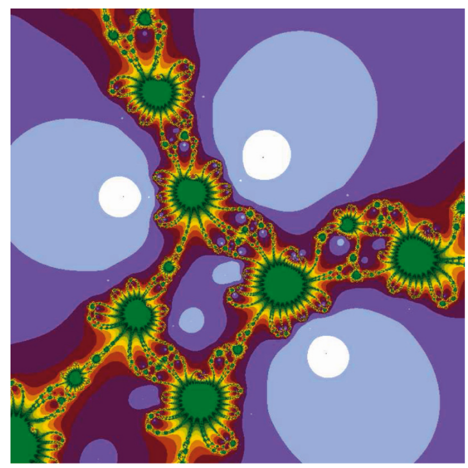

Figure 18: Polynomiograph for $q_3(w)$ using CHM.

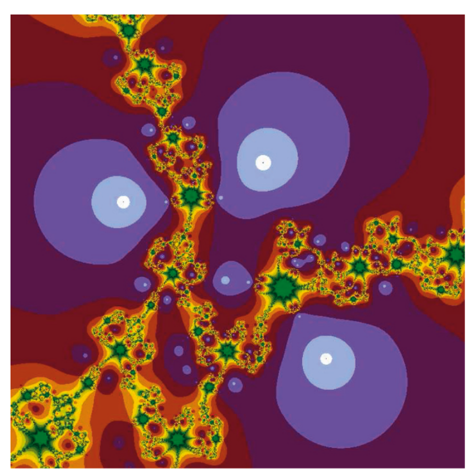

Figure 19: Polynomiograph for $q_3(w)$ using NM2.

$12\sqrt{81-12i}\,]^{2/3}-4 \;+\; 4i[-108+8i+12\sqrt{81-12i}\,]^{1/3}+i\;\sqrt{3}\;[-108+8i+12\sqrt{81-12i}\,]^{2/3}+\; 4i\sqrt{3}/-12[-108+8i+12\sqrt{81-12i}\,]^{1/3}$. For this purpose, we ran all the iteration schemes with the aid of computer program and the corresponding graphs can be seen in Figures 37–43.

In all six experiments, we considered the different cubic complex polynomials for examining the graphical aspects of the suggested iteration schemes. In all the obtained images, the regions of convergence for the suggested iteration schemes possess the larger convergence areas than the other

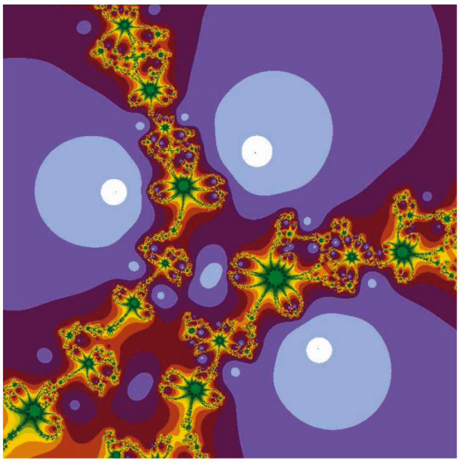

Figure 20: Polynomiograph for $q_3(w)$ using YM.

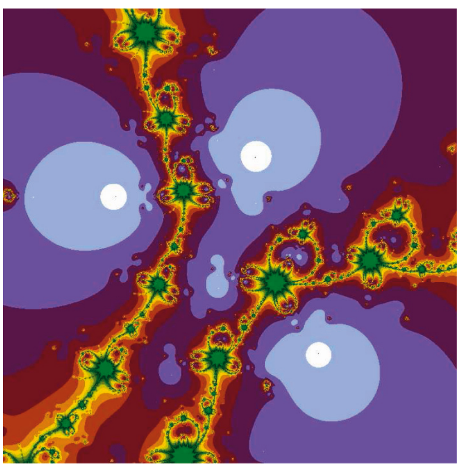

Figure 21: Polynomiograph for $q_3(w)$ using Algorithm 1.

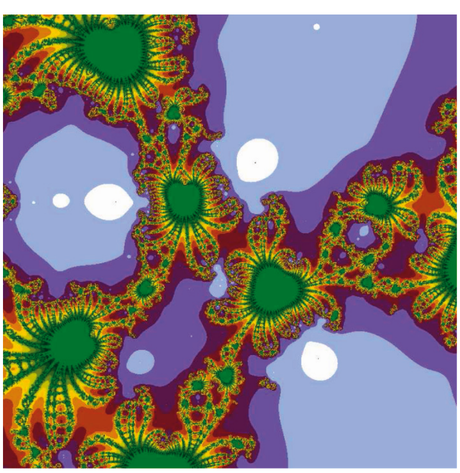

Figure 22: Polynomiograph for $q_3(w)$ using Algorithm 2.

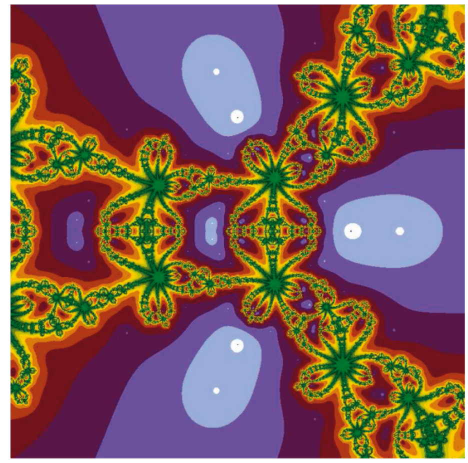

Figure 23: Polynomiograph for $q_4(w)$ using NM1.

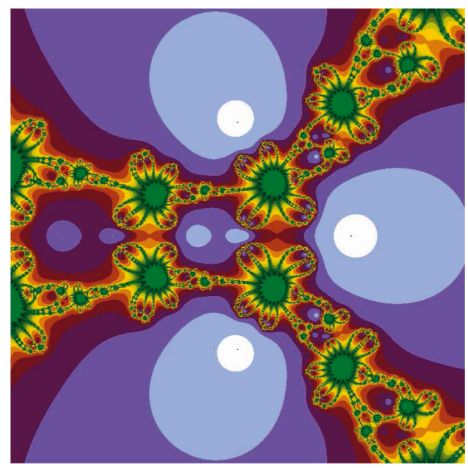

Figure 24: Polynomiograph for $q_4(w)$ using CM.

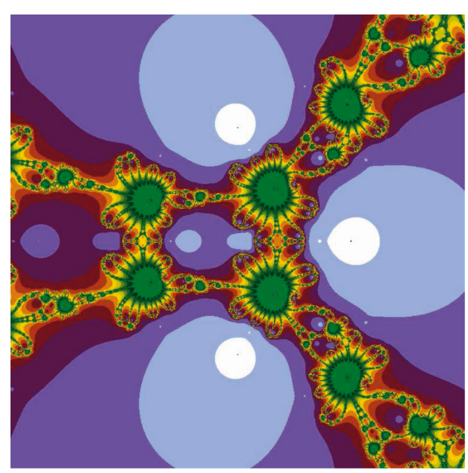

Figure 25: Polynomiograph for $q_4(w)$ using CHM.

comparable methods which certified the efficiency and better convergence of our proposed algorithms. The colour tones represent the performance and efficiency of the considered algorithm used to build the polynomiograph. The two important aspects which can be exhibited by these graphical objects are the convergence-speed and the dynamics of the

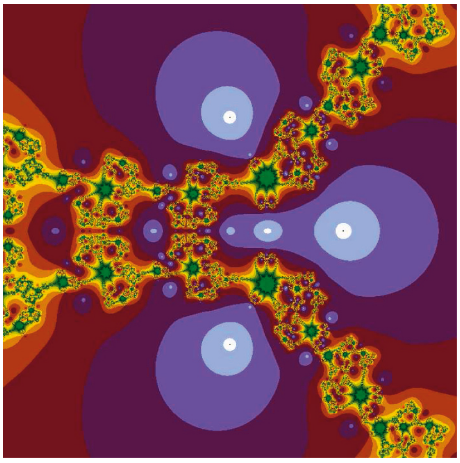

Figure 26: Polynomiograph for $q_4(w)$ using NM2.

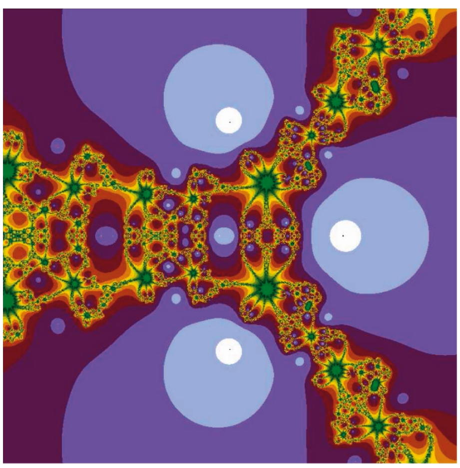

Figure 27: Polynomiograph for $q_4(w)$ using YM.

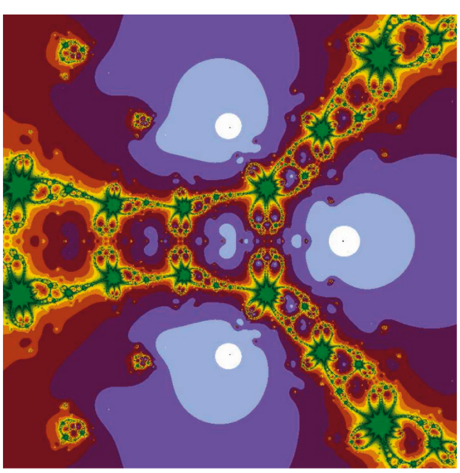

Figure 28: Polynomiograph for $q_4(w)$ using Algorithm 1.

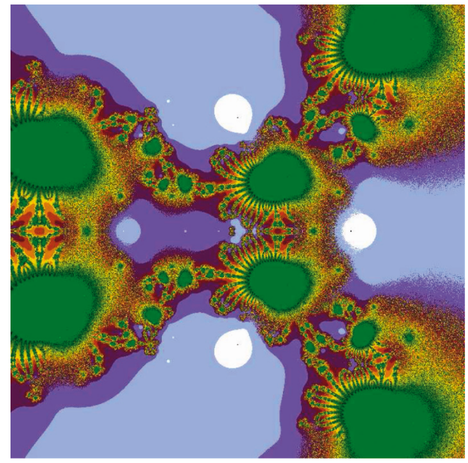

Figure 29: Polynomiograph for $q_4(w)$ using Algorithm 2.

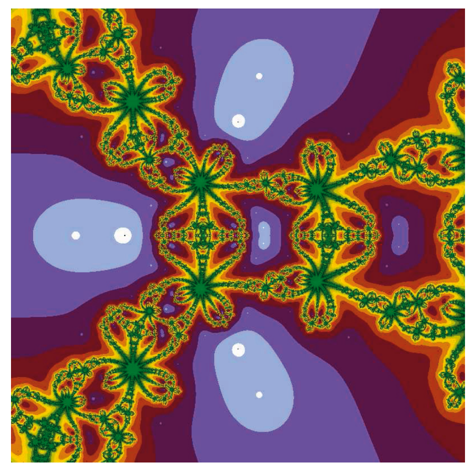

Figure 30: Polynomiograph for $q_5(w)$ using NM1.

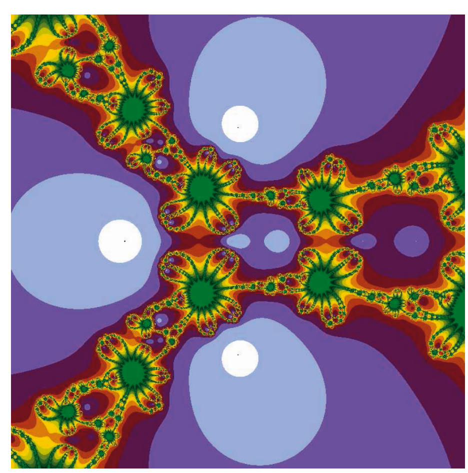

Figure 31: Polynomiograph for $q_5(w)$ using CM.

considered iteration techniques used to build these graphs. The first may be shown by examining the image's colour tones. The darkness of the colours in the presented images demonstrates fast convergence with less number of iterations, i.e., the darker the image, the more efficient the approach, and the given images demonstrate the superiority of

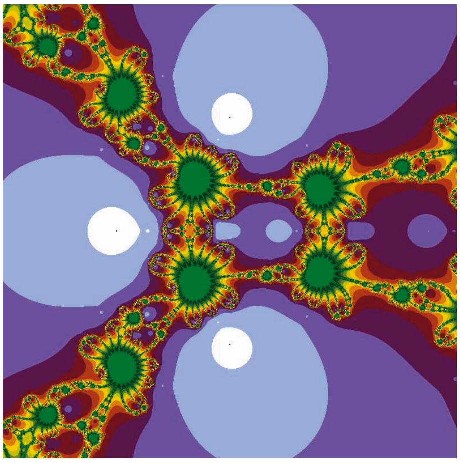

Figure 32: Polynomiograph for $q_5(w)$ using CHM.

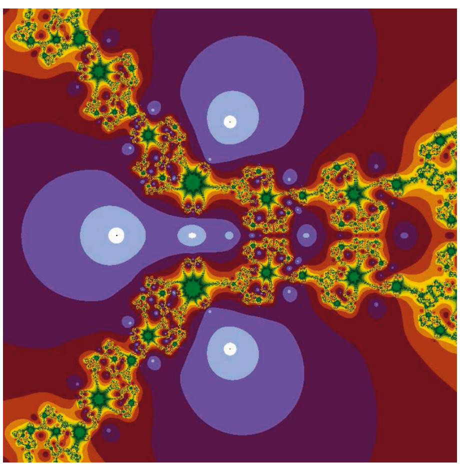

Figure 33: Polynomiograph for $q_5(w)$ using NM2.

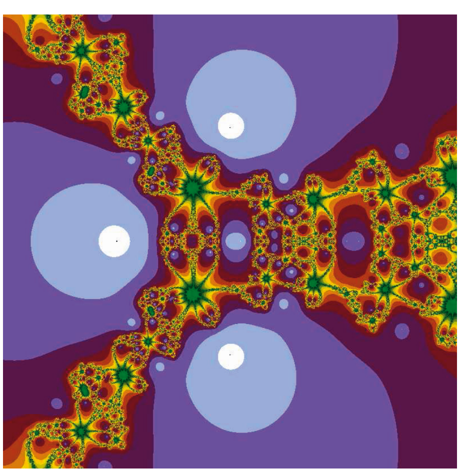

Figure 34: Polynomiograph for $q_5(w)$ using YM.

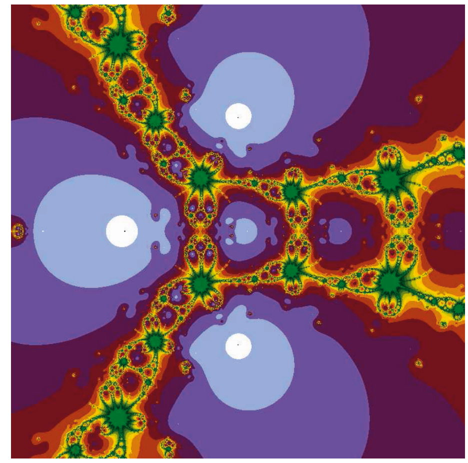

Figure 35: Polynomiograph for $q_5(w)$ using Algorithm 1.

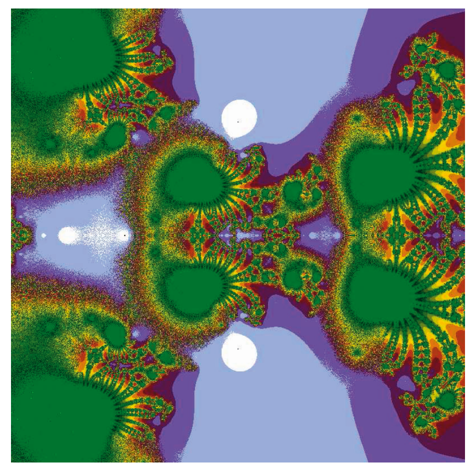

Figure 36: Polynomiograph for $q_5(w)$ using Algorithm 2.

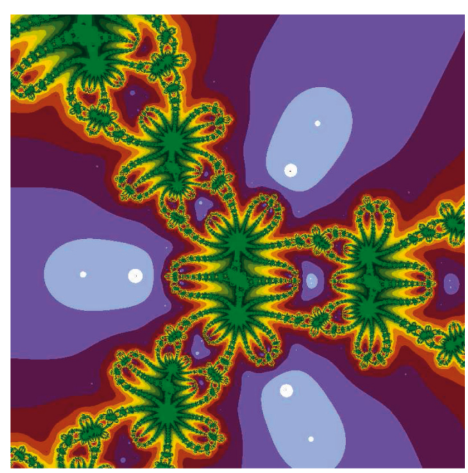

Figure 37: Polynomiograph for $q_6(w)$ using NM1.

the suggested methods. The second aspect may be examined by observing the colour fluctuation of the drawn polynomiographs. The regions with a limited variety of colours have low dynamics, whereas the regions with a great diversity of colours have high dynamics. The black colour in the displayed graphical representations demonstrates the

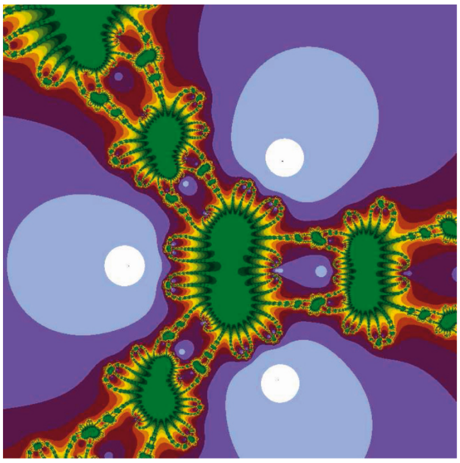

FIGURE 38: Polynomiograph for $q_6(w)$ using CM.

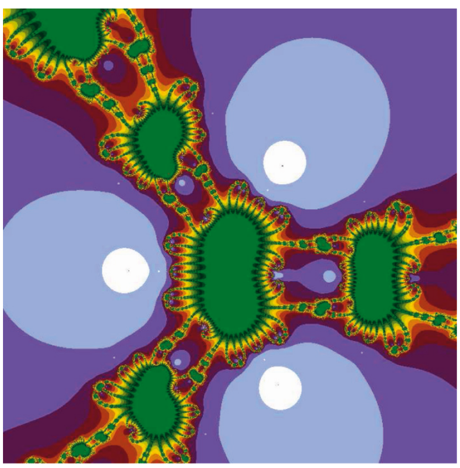

FIGURE 39: Polynomiograph for $q_6(w)$ using CHM.

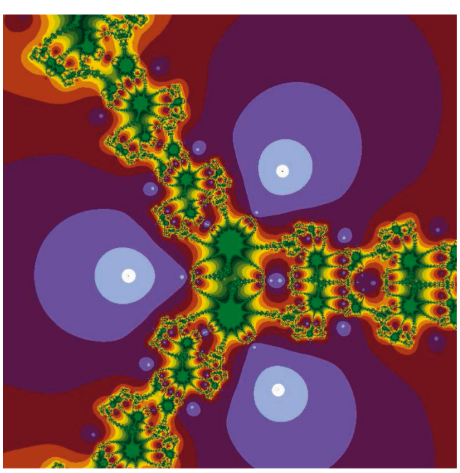

FIGURE 40: Polynomiograph for $q_6(w)$ using NM2.

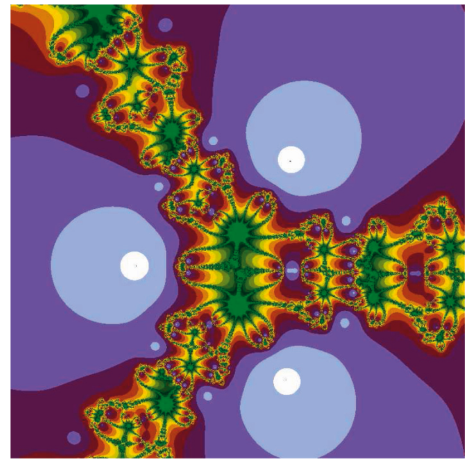

FIGURE 41: Polynomiograph for $q_6(w)$ using YM.

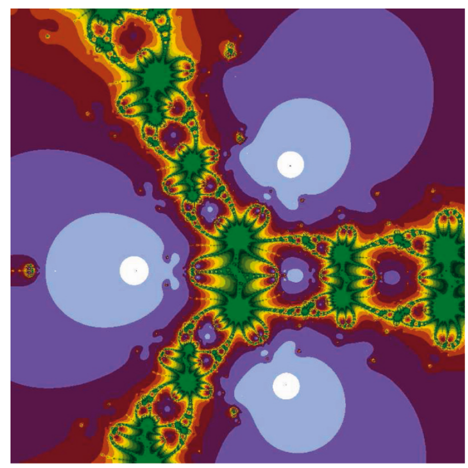

FIGURE 42: Polynomiograph for $q_6(w)$ using Algorithm 1.

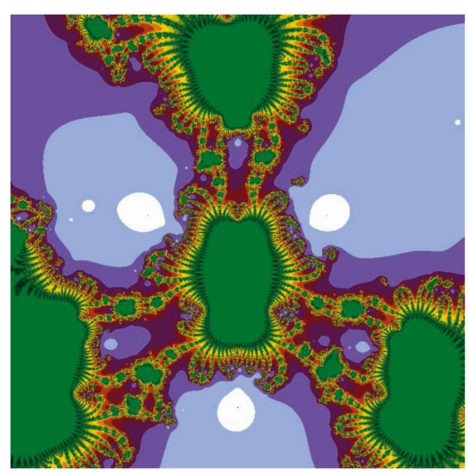

FIGURE 43: Polynomiograph for $q_6(w)$ using Algorithm 2.

method's deficiency by locating those precise locations where the solution cannot be obtained for the specified no. of iterations with the defined precision. The same-coloured regions in the figures show the same amount of iterations spent by different iteration strategies to approximate the answer and provide a comparable perspective of the contour lines on the map. All of the graphics were created using the computer program Mathematica 12.0 with the accuracy $\epsilon = 0.01$, and the upper bound of the no. of iterations $m = 15$.

6. Conclusions

In this study, we have established and analyzed two novel optimal root-finding algorithms for nonlinear functions. The convergence criterion of the presented algorithms has been discussed and verified that the suggested iterative algorithms bear fourth- and sixth-order convergence. To exhibit the applicability of the suggested algorithms, some real-life engineering applications have also been added and solved whose numerical results confirmed that the proposed algorithms are time-efficient and consumed less iterations to achieve the required solution and more accurate to the exact solution. The dynamics of the provided methods demonstrating the greater convergence area in comparison with the other comparable methods. These dynamics also highlighted the proposed algorithms' quicker convergence speed and other dynamical properties, proving their superiority over the others in comparison.

Data Availability

All data required for this paper are included within this paper.

Conflicts of Interest

The authors do not have any conflicts of interest.

Authors' Contributions

All authors contributed equally in this paper.

References

[1] R. L. Burden and J. D. Faires, *Numerical Analysis*, Brooks/Cole Publishing Company, Monterey, CA, USA, 6th edition, 1997.

[2] J. M. Gutiérrez and M. A. Hernández, "A family of Chebyshev-Halley type methods in Banach spaces," *Bulletin of the Australian Mathematical Society*, vol. 55, no. 1, pp. 113–130, 1997.

[3] I. Argyros, D. Chen, and Q. Qian, "A note on the Halley method in Banach spaces," *Applied Mathematics and Computation*, vol. 58, no. 2-3, pp. 215–224, 1993.

[4] C. Chun, "Construction of Newton-like iteration methods for solving nonlinear equations," *Numerische Mathematik*, vol. 104, no. 3, pp. 297–315, 2006.

[5] A. Amiri, A. Cordero, M. T. Darvishi, and J. R. Torregrosa, "A fast algorithm to solve systems of nonlinear equations," *Journal of Computational and Applied Mathematics*, vol. 354, pp. 242–258, 2019.

[6] R. Behl, M. Salimi, M. Ferrara, S. Sharifi, and S. K. Alharbi, "Some real-life applications of a newly constructed derivative free iterative scheme," *Symmetry*, vol. 11, no. 2, p. 239, 2019.

[7] A. Naseem, M. A. Rehman, and T. Abdeljawad, "Higher-order root-finding algorithms and their basins of attraction," *Journal of Mathematics*, vol. 2020, Article ID 5070363, 11 pages, 2020.

[8] A. Özyapıcı, "Effective numerical methods for non-linear equations," *International Journal of Applied and Computational Mathematics*, vol. 6, no. 2, p. 35, 2020.

[9] A. M. Ostrowski, *Solution of Equations and Systems of Equations*, Academic Press, Cambridge, MA, USA, 1966.

[10] J. F. Traub, *Iterative Methods for the Solution of Equations*, Chelsea Publishing Company, New York, NY, USA, 1982.

[11] M. Aslam Noor and K. Inayat Noor, "Three-step iterative methods for nonlinear equations," *Applied Mathematics and Computation*, vol. 183, no. 1, pp. 322–327, 2006.

[12] A. Golbabai and M. Javidi, "A third-order Newton type method for nonlinear equations based on modified homotopy perturbation method," *Applied Mathematics and Computation*, vol. 191, no. 1, pp. 199–205, 2007.

[13] M. A. Noor, K. I. Noor, and K. Aftab, "Some new iterative methods for solving nonlinear equations," *World Applied Sciences Journal*, vol. 20, no. 6, pp. 870–874, 2012.

[14] A. Kumar, P. Maroju, R. Behl, D. K. Gupta, and S. S. Motsa, "A family of higher order iterations free from second derivative for nonlinear equations in R," *Journal of Computational and Applied Mathematics*, vol. 330, pp. 215–224, 2018.

[15] O. Said Solaiman, S. A. Abdul Karim, and I. Hashim, "Optimal fourth- and eighth-order of convergence derivative-free modifications of King's method," *Journal of King Saud University - Science*, vol. 31, no. 4, pp. 1499–1504, 2019.

[16] P. B. Chand, F. I. Chicharro, N. Garrido, and P. Jain, "Design and complex dynamics of potra-pták-type optimal methods for solving nonlinear equations and its applications," *Mathematics*, vol. 7, no. 10, p. 942, 2019.

[17] A. Naseem, M. A. Rehman, and T. Abdeljawad, "Some new iterative algorithms for solving one-dimensional non-linear equations and their graphical representation," *IEEE Access*, vol. 9, pp. 8615–8624, 2021.

[18] A. Rafiq and M. Rafiullah, "Some multi-step iterative methods for solving nonlinear equations," *Computers & Mathematics with Applications*, vol. 58, no. 8, pp. 1589–1597, 2009.

[19] H. T. Kung and J. F. Traub, "Optimal order of one-point and multipoint iteration," *Journal of the ACM*, vol. 21, no. 4, pp. 643–651, 1974.

[20] C. Chun, "Some variants of King's fourth-order family of methods for nonlinear equations," *Applied Mathematics and Computation*, vol. 190, no. 1, pp. 57–62, 2007.

[21] J. H. Yun, "A note on three-step iterative method for nonlinear equations," *Applied Mathematics and Computation*, vol. 202, no. 1, pp. 401–405, 2008.

[22] M. ShachamShacham and E. Kehat, "An iteration method with memory for the solution of a non-linear equation," *Chemical Engineering Science*, vol. 27, no. 11, pp. 2099–2101, 1972.

[23] M. Planck, *The Theory of Heat Radiation. Translated by Masius, M*, Blakiston's Son & Co, Philadelphia, PA, USA, 2nd edition, 1914.

[24] M. Shacham, "An improved memory method for the solution of a nonlinear equation," *Chemical Engineering Science*, vol. 44, no. 7, pp. 1495–1501, 1989.

[25] M. Shacham and E. Kehat, "Converging interval methods for the iterative solution of a non-linear equation," *Chemical Engineering Science*, vol. 28, no. 12, pp. 2187–2193, 1973.

[26] C. S. Chapra, *Applied Numerical Methods with MATLAB for Engineers and Scientists*, McGraw-Hill, New York, NY, USA, 2010.

[27] R. Manning, “On the flow of water in open channels and pipes,” *Transactions of the Institution of Civil Engineers of Ireland*, vol. 20, pp. 161–207, 1891.
[28] B. Kalantari, “Polynomiography: from the fundamental theorem of algebra to art,” *Leonardo*, vol. 38, no. 3, pp. 233–238, 2005.
[29] B. Kalantari and E. H. Lee, “Newton-Ellipsoid polynomiography,” *Journal of Mathematics and the Arts*, vol. 13, no. 4, pp. 336–352, 2019.
[30] B. Kalantari, “An invitation to polynomiography via exponential series,” 2017, https://arxiv.org/abs/1707.09417.
[31] K. Gdawiec, W. Kotarski, and A. Lisowska, “Visual analysis of the Newton’s method with fractional order derivatives,” *Symmetry*, vol. 11, no. 9, p. 1143, 2019.
[32] K. Gdawiec, “Fractal patterns from the dynamics of combined polynomial root finding methods,” *Nonlinear Dynamics*, vol. 90, no. 4, pp. 2457–2479, 2017.
[33] M. Scott, B. Neta, and C. Chun, “Basin attractors for various methods,” *Applied Mathematics and Computation*, vol. 218, no. 6, pp. 2584–2599, 2011.