IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 2, 2021, accepted June 18, 2021, date of publication June 22, 2021, date of current version July 5, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3091473

Novel Iteration Schemes for Computing Zeros of Non-Linear Equations With Engineering Applications and Their Dynamics

AMIR NASEEM[1], M. A. REHMAN[1], THABET ABDELJAWAD[2,3,4], AND YU-MING CHU[5,6]

[1]Department of Mathematics, University of Management and Technology, Lahore 54770, Pakistan
[2]Department of Mathematics and General Sciences, Prince Sultan University, Riyadh 11586, Saudi Arabia
[3]Department of Medical Research, China Medical University Hospital, Taichung 40402, Taiwan
[4]Department of Computer Science and Information Engineering, Asia University, Taichung 41354, Taiwan
[5]Department of Mathematics, Huzhou University, Huzhou 313000, China
[6]Hunan Provincial Key Laboratory of Mathematical Modeling and Analysis in Engineering, Changsha University of Science and Technology, Changsha 410114, China

Corresponding author: Yu-Ming Chu (chuyuming@zjhu.edu.cn)

ABSTRACT The task of root-finding of the non-linear equations is perhaps, one of the most complicated problems in applied mathematics especially in a diverse range of engineering applications. The characteristics of the root-finding methods such as convergence rate, performance, efficiency, etc., are directly relied upon the initial guess of the solution to execute the process in most of the systems of non-linear equations. Keeping these facts into mind, based on Taylor's series expansion, we present some new modifications of Halley, Househölder and Golbabai and Javidi's methods and then making them second derivative free by applying Taylor's series. The convergence analysis of the suggested methods is discussed. It is established that the proposed methods possess convergence of orders five and six. Several numerical problems have been tested to demonstrate the validity and applicability of the proposed methods. These test examples also include some real-life problems associated with chemical and civil engineering such as open channel flow problem, the adiabatic flame temperature equation, conversion of nitrogen-hydrogen feed to ammonia and the van der Wall's equation whose numerical results prove the better performance of the suggested methods as compared to other well-known existing methods of the same kind in the literature. Finally, the dynamics of the presented algorithms in the form of polynomiographs have been shown with the aid of computer program by considering some complex polynomials and compared them with the other well-known iterative algorithms that revealed the convergence speed and other dynamical aspects of the presented methods.

INDEX TERMS Order of convergence, non-linear equations, Newton's method, Halley's method, polynomiography.

I. INTRODUCTION

In applied mathematics and engineering sciences, the root-finding algorithms for the solution of non-linear algebraic equations of the general form:

$$\xi(s) = 0, \tag{1}$$

have played a key role especially in a diverse range of fuzzy systems, image processing and engineering applications, where $\xi : \mathcal{D} \subset \mathbb{R} \to \mathbb{R}$ is a scalar function defined on the domain $\mathcal{D}$ which is an open connected set.

The associate editor coordinating the review of this manuscript and approving it for publication was Santi C. Pavone.

We assume that α is a simple zero of (1) and s_0 is an initial guess sufficiently close to α. Using the Taylor's series around s_0 for (1), we have

$$\xi(s_0) + (s - s_0)\xi(s_0) + \frac{1}{2!}(s - s_0)^2\xi''(s_0) + \ldots = 0. \tag{2}$$

If $\xi'(s_0) \neq 0$, we can evaluate the above expression as follows:

$$\xi(s_0) + (s - s_0)\xi'(s_0) = 0.$$

If we choose s_{j+1} as the root of equation, then we have

$$s_{j+1} = s_j - \frac{\xi(s_j)}{\xi'(s_j)}. \tag{3}$$

This is so-called the Newton's method [8], [11], [21] for root-finding of nonlinear functions, which converges quadratically. From (2), one can evaluate

$$s = s_0 - \frac{2\xi(s_0)\xi'(s_0)}{2\xi'^2(s_0) - \xi(s_0)\xi''(s_0)}. \tag{4}$$

In iterative form:

$$s_{j+1} = s_j - \frac{2\xi(s_j)\xi'(s_j)}{2\xi'^2(s_j) - \xi(s_j)\xi''(s_j)}, \tag{5}$$

which is cubically convergent and well-known Halley's method [5], [15] for root-finding of non-linear scalar equations. After simplifying (2), the following equality can be obtained:

$$s = s_0 - \frac{\xi(s_0)}{\xi'(s_0)} - \frac{\xi^2(s_0)\xi''(s_0)}{2\xi'^3(s_0)}. \tag{6}$$

In iterative form:

$$s_{j+1} = s_j - \frac{\xi(s_j)}{\xi'(s_j)} - \frac{\xi^2(s_j)\xi''(s_j)}{2\xi'^3(s_j)}, \tag{7}$$

which is known as Househölder's method [16] for solving nonlinear equations in one variable and converges cubically.

For finding zeros of algebraic equation (1), Golbabai and Javidi [14], by applying the basic idea of homotopy perturbation, arrived at the following method:

$$s = s_0 - \frac{\xi(s_0)}{\xi'(s_0)} - \frac{\xi^2(s_0)\xi''(s_0)}{2[\xi'^3(s_0) - \xi(s_0)\xi'(s_0)\xi''(s_0)]}. \tag{8}$$

In iterative form:

$$s_{j+1} = s_j - \frac{\xi(s_j)}{\xi'(s_j)} - \frac{\xi^2(s_j)\xi''(s_j)}{2[\xi'^3(s_j) - \xi(s_j)\xi'(s_j)\xi''(s_j)]}, \tag{9}$$

which is cubically convergent Golbabai and Javidi's method for finding zeros of non-linear equations in one dimension

In recent years, many mathematicians tried to modify existing methods using different mathematical techniques and proposed some new multi-step iterative methods, having higher orders of convergence [1]–[4], [7], [10], [28], [29], [32], [35], [39].

In 20th century, Ostrowski [34] proposed a fourth-order two-step iterative algorithm by considering Newton's iteration method as a predictor. Later, Traub [40] considered Newton's algorithm as a predictor and corrector step and presented a new root-finding algorithm with the same convergence of order four. Noor *et al.* in 2007 [31], established a new second-derivative free two-step Halley's method with the help of a finite difference scheme and proved that the proposed algorithm possessed the fifth order of convergence. Nazeer *et al.* [30], in (2016), proposed a new second derivative free generalized Newton-Raphson's method with convergence of order five using finite difference scheme. After that Kumar *et al.* [23], in 2018, established a new sixth-order parameter based family of algorithm for solving non-linear equations. In 2019, Solaiman *et al.* [38] suggested derivative free optimal fourth-order and eighth-order modifications of King's method by implementing the composition technique combined with rational interpolation, and the idea of Padé approximation. Very recently, Naseem *et al.* [27] presented some new ninth order iterative algorithms for determining the zeros of non-linear scalar equations and then presented their graphical representation by means of polynomiographs using different complex polynomials.

In this paper, we suggested and then analyzed six new iteration schemes. The derivation of these iteration schemes is purely based on Taylor's series expansion. Out of the six suggested schemes, three are second-derivative free. We approximate the second derivative by means of Taylor's series that results some new efficient techniques. The suggested iterations schemes bearing quintic and sextic convergence and showing fast and better performance in comparison with the similar existing well-known iterative algorithms. We compared the numeric behaviors of the suggested methods with the other similar existing methods in literature by solving some test functions which also include some real-life problems associated with the chemical and civil engineering and their results proved the supremacy of the suggested methods to the other comparable methods. The dynamics of the suggested methods in the form of polynomiographs with the aid of computer technology have been presented and compared with the other similar existing methods that revealed the convergence speed and other dynamical aspects of the suggested methods and proved the superiority of the proposed methods to the other ones in comparison.

The rest of the paper is divided as follows. Six novel iteration schemes for computing zeros of non-linear equations are given in Section 2. In Section 3, we discussed the convergence criterion of the suggested schemes. In Section 4, several test examples along with the engineering problems have been solved to show the performance, applicability and validity of the suggested schemes. In Section 5, the polynomiographs of some complex polynomials through the suggested iteration schemes have been presented and compared with other methods. Finally, the conclusion of the paper is given in Section 6.

II. MAIN RESULTS

Let $\xi : \mathcal{D} \to \mathbb{R}$, $\mathcal{D} \subset R$ is a scalar function defined on the domain $\mathcal{D}$ where $\mathcal{D}$ is an open connected set, then from (2), one can write:

$$s = s_0 - \frac{\xi(s_0)}{\xi'(s_0)} - \frac{(s - s_0)^2\xi''(s_0)}{2\xi'(s_0)}. \tag{10}$$

Now from (4), (6) and (8), we can write the following expressions:

$$s - s_0 = -\frac{2\xi(s_0)\xi'(s_0)}{2\xi'^2(s_0) - \xi(s_0)\xi''(s_0)}, \tag{11}$$

$$s - s_0 = -\frac{\xi(s_0)}{\xi'(s_0)} - \frac{\xi^2(s_0)\xi''(s_0)}{2\xi'^3(s_0)}, \tag{12}$$

$$s - s_0 = -\frac{\xi(s_0)}{\xi'(s_0)} - \frac{\xi^2(s_0)\xi''(s_0)}{2[\xi'^3(s_0) - \xi(s_0)\xi'(s_0)\xi''(s_0)]}. \tag{13}$$

Using (11), (12) and (13) in (10), we obtain the following expressions:

$$s = s_0 - \frac{\xi(s_0)}{\xi'(s_0)} - \frac{2\xi^2(s_0)\xi'(s_0)\xi''(s_0)}{4[\xi'^4(s_0)-\xi(s_0)\xi'^2(s_0)\xi''(s_0)]+\xi^2(s_0)\xi''^2(s_0)}, \quad (14)$$

$$s = s_0 - \frac{\xi(s_0)}{\xi'(s_0)} - \frac{\xi^2(s_0)\xi''(s_0)[2\xi'^2(s_0)+\xi(s_0)\xi''(s_0)]^2}{8\xi'^7(s_0)}, \quad (15)$$

$$s = s_0 - \frac{\xi(s_0)}{\xi'(s_0)} - \frac{\xi^2(s_0)\xi''(s_0)[2\xi'^2(s_0)-\xi(s_0)\xi''(s_0)]^2}{8\xi'^3(s_0)[\xi'^2(s_0)-\xi(s_0)\xi''(s_0)]^2}. \quad (16)$$

After rewriting the above expressions in the general form with the insertion of Newton's iteration method as a predictor, we arrive at a new algorithms of the form:

Algorithm 1: For a given s_0, compute the approximate solution s_{j+1} by the following iterative scheme:

$$t_j = s_j - \frac{\xi(s_j)}{\xi'(s_j)}, j = 0, 1, 2, \ldots,$$
$$s_{j+1} = t_j - \frac{\xi(t_j)}{\xi'(t_j)} - \frac{2\xi^2(t_j)\xi'(t_j)\xi''(t_j)}{4[\xi'^4(t_j)-\xi(t_j)\xi'^2(t_j)\xi''(t_j)]+\xi^2(t_j)\xi''^2(t_j)}.$$

Algorithm 2: For a given s_0, compute the approximate solution s_{j+1} by the following iterative schemes:

$$t_j = s_j - \frac{\xi(s_j)}{\xi'(s_j)}, j = 0, 1, 2, \ldots,$$
$$s_{j+1} = t_j - \frac{\xi(t_j)}{\xi'(t_j)} - \frac{\xi^2(t_j)\xi''(t_j)[2\xi'^2(t_j)+\xi(t_j)\xi''(t_j)]^2}{8\xi'^7(t_j)}.$$

Algorithm 3: For a given s_0, compute the approximate solution s_{j+1} by the following iterative schemes:

$$t_j = s_j - \frac{\xi(s_j)}{\xi'(s_j)}, j = 0, 1, 2, \ldots,$$
$$s_{j+1} = t_j - \frac{\xi(t_j)}{\xi'(t_j)} - \frac{\xi^2(t_j)\xi''(t_j)[2\xi'^2(t_j)-\xi(t_j)\xi''(t_j)]^2}{8\xi'^3(t_j)[\xi'^2(t_j)-\xi(t_j)\xi''(t_j)]^2}.$$

Which are the modifications of Halley, Househölder and Golbabai and Javidi's methods respectively. For applying the above defined algorithms, one has to find the first as well as second derivatives of the given function $\xi(s)$. But in several cases, we face such a situation where the second derivative of the function does not exist and our method fails to find the solution. To resolve this issue, we use Taylor's series around s for the approximation of the second derivative as follows:

$$\xi(s_j) = \xi(t_j) + (s_j - t_j)\xi'(t_j) + \frac{(s_j - t_j)^2}{2}\xi''(t_j), \quad (17)$$

which implies:

$$\xi''(t_j) = \frac{2[\xi'(s_j)(\xi(s_j)-\xi(t_j))-\xi(s_j)\xi'(t_j)]\xi'(s_j)}{\xi^2(s_j)} = Q(s_j, t_j). \quad (18)$$

Using (18) in Algorithms 1-3, we obtain some new algorithms as follows:

Algorithm 4: For a given s_0, compute the approximate solution s_{j+1} by the following iterative scheme:

$$t_j = s_j - \frac{\xi(s_j)}{\xi'(s_j)}, j = 0, 1, 2, \ldots,$$
$$s_{j+1} = t_j - \frac{\xi(t_j)}{\xi'(t_j)} - \frac{2\xi^2(t_j)\xi'(t_j)Q(s_j, t_j)}{4[\xi'^4(t_j)-\xi(t_j)\xi'^2(t_j)Q(s_j, t_j)]+\xi^2(t_j)Q^2(s_j, t_j)}.$$

Algorithm 5: For a given s_0, compute the approximate solution s_{j+1} by the following iterative schemes:

$$t_j = s_j - \frac{\xi(s_j)}{\xi'(s_j)}, j = 0, 1, 2, \ldots,$$
$$s_{j+1} = t_j - \frac{\xi(t_j)}{\xi'(t_j)} - \frac{\xi^2(t_j)Q(s_j, t_j)[2\xi'^2(t_j)+\xi(t_j)Q(s_j, t_j)]^2}{8\xi'^7(t_j)}.$$

Algorithm 6: For a given s_0, compute the approximate solution s_{j+1} by the following iterative schemes:

$$t_j = s_j - \frac{\xi(s_j)}{\xi'(s_j)}, j = 0, 1, 2, \ldots,$$
$$s_{j+1} = t_j - \frac{\xi(t_j)}{\xi'(t_j)} - \frac{\xi^2(t_j)Q(s_j, t_j)[2\xi'^2(t_j)-\xi(t_j)Q(s_j, t_j)]^2}{8\xi'^3(t_j)[\xi'^2(t_j)-\xi(t_j)Q(s_j, t_j)]^2}.$$

Which are novel second-derivative free iterative algorithms for computing zeros of non-linear algebraic equations. The most important characteristic of these suggested methods is their applicability to all those non-linear scalar functions in which the second derivative does not exist. The approximation of the second derivative causes less number of computations per iteration which results in better efficiency indices of the presented methods as compared to those methods which require second derivative. The numerical results of the solved test examples proved their best performance in comparison with the other similar existing methods in literature.

III. CONVERGENCE ANALYSIS

In this section, we shall discuss the convergence criterion of the suggested iteration schemes.

Theorem 1: Suppose that α is a simple root of the equation $\xi(s) = 0$. If $\xi(s)$ is sufficiently smooth in the neighborhood of α, then the orders of convergence of Algorithms 1–3 are at least six.

Proof: To analyze the convergence of the proposed algorithms, suppose that α is a root of the equation $\xi(s) = 0$ and e_j be the error at nth iteration, then $e_j = s_j - \alpha$ and by using Taylor's series expansion, we have

$$\begin{aligned}\xi(s_j) &= \xi'(\alpha)e_j + \frac{1}{2!}\xi''(\alpha)e_j^2 + \frac{1}{3!}\xi'''(\alpha)e_j^3 \\ &\quad + \frac{1}{4!}\xi^{(iv)}(\alpha)e_j^4 + \frac{1}{5!}\xi^{(v)}(\alpha)e_j^5 + \frac{1}{6!}\xi^{(vi)}(\alpha)e_j^6 + O(e_j^7), \\ \xi(s_j) &= \xi'(\alpha)[e_j + c_2e_j^2 + c_3e_j^3 + c_4e_j^4 + c_5e_j^5 + c_6e_j^6 + O(e_j^7)] \end{aligned} \tag{19}$$

$$\begin{aligned}\xi'(s_j) &= \xi'(\alpha)[1 + 2c_2e_j + 3c_3e_j^2 + 4c_4e_j^3 + 5c_5e_j^4 \\ &\quad + 6c_6e_j^5 + 7c_7e_j^6 + O(e_j^7)], \end{aligned} \tag{20}$$

where

$$c_j = \frac{1}{j!}\frac{\xi^{(j)}(\alpha)}{\xi'(\alpha)}.$$

With the help of equations(19)–(20), we get

$$\begin{aligned} t_j &= \xi'(\alpha)[\alpha + c_2e_j^2 + (2c_3 - 2c_2^2)e_j^3 + (3c_4 - 7c_2c_3 \\ &\quad + 4c_2^3)e_j^4 + (-6c_3^2 + 20c_3c_2^2 - 10c_2c_4 + 4c_5 - 8c_2^4)e_j^5 \\ &\quad + (-17c_4c_3 + 28c_4c_2^2 - 13c_2c_5 + 5c_6 + 33c_2c_3^2 \\ &\quad - 52c_3c_2^3 + 16c_2^5)e_j^6 + O(e_j^7)], \end{aligned} \tag{21}$$

$$\begin{aligned} \xi(t_j) &= \xi'(\alpha)[c_2e_j^2 + (2c_3 - 2c_2^2)e_j^3 + (5c_2^3 - 7c_2c_3 \\ &\quad + 3c_4)e_j^4 + (24c_3c_2^2 - 12c_2^4 - 10c_2c_4 + 4c_5 \\ &\quad - 6c_3^2)e_j^5 + (-73c_3c_2^3 + 34c_4c_2^2 + 28c_2^5 \\ &\quad + 37c_2c_3^2 - 17c_4c_3 - 13c_2c_5 + 5c_6)e_j^6 \\ &\quad + O(e_j^7)], \end{aligned} \tag{22}$$

$$\begin{aligned} \xi'(t_j) &= \xi'(\alpha)[1 + 2c_2^2e_j^2 + (4c_2c_3 - 4c_2^3)e_j^3 \\ &\quad + (6c_2c_4 - 11c_3c_2^2 + 8c_2^4)e_j^4 + 28c_3c_2^3 \\ &\quad - 20c_4c_2^2 + 8c_2c_5 - 16c_2^5)e_j^5 + (-16c_4c_2c_3 \\ &\quad - 68c_3c_2^4 + 12c_3^3 + 60c_4c_2^3 - 26c_5c_2^2 + 10c_2c_6 \\ &\quad + 32c_2^6)e_j^6 + O(e_j^7)], \end{aligned} \tag{23}$$

$$\begin{aligned} \xi''(t_j) &= \xi'(\alpha)[2c_2 + 6c_2c_3e_j^2 + (12c_3^2 - 12c_3c_2^2)e_j^3 \\ &\quad + (-42c_2c_3^2 + 18c_4c_3 + 24c_3c_2^3 + 12c_4c_2^2)e_j^4 \\ &\quad + (-12c_2c_4c_3 + 24c_5c_3 - 36c_3^3 + 120c_3^2c_2^2 \\ &\quad - 48c_3c_2^4 - 48c_4c_2^3)e_j^5 + (-78c_3c_2c_5 + 30c_3c_6 \\ &\quad - 54c_4c_3^2 - 96c_3c_4c_2^2 + 198c_2c_3^3 - 312c_3^2c_2^3 \\ &\quad + 96c_3c_2^5 + 72c_2c_4^2 + 144c_4c_2^4 + 20c_5c_2^3)e_j^6 \\ &\quad + O(e_j^8)]. \end{aligned} \tag{24}$$

Using equations (21)–(24) in Algorithms 1–3, we arrive at the same result as follows:

$$s_{j+1} = \alpha - c_3c_2^3e_j^6 + O(e_j^7),$$

which implies that

$$e_{i+1} = -c_3c_2^3e_j^6 + O(e_j^7).$$

The above equality shows that the orders of convergence of the Algorithms 1–3 are at least six. □

Theorem 2: Suppose that α is a simple root of the equation $\xi(s) = 0$. If $\xi(s)$ is sufficiently smooth in the neighborhood of α, then the convergence orders of Algorithms 4–6 are at least five.

Proof: With the help of equations (19)–(23), we have

$$\begin{aligned} Q(s_j, t_j) &= \xi'(\alpha)[2c_2 + 2c_3e_j + (4c_2c_3 + 2c_4)e_j^2 \\ &\quad + (4c_2c_4 - 8c_3c_2^2 + 8c_3^2 + 2c_5)e_j^3 + (4c_2c_5 \\ &\quad + 20c_3c_4 - 28c_2c_3^2 + 16c_3c_2^3 - 2c_2^2c_4 + 2c_6)e_j^4 \\ &\quad + (4c_2c_6 + 12c_4^2 - 8c_4c_2^3 + 24c_5c_3 - 2c_5c_2^2 \\ &\quad + 80c_3^2c_2^2 - 32c_3c_2^4 + 2c_7 - 44c_2c_3c_4 - 24c_3^3)e_j^5 \\ &\quad + (4c_2c_7 + 28c_4c_5 + 28c_3c_6 - 2c_6c_2^2 - 4c_2c_4^2 \\ &\quad - 68c_4c_3^2 + 40c_4c_2^4 + 132c_2c_3^3 - 208c_3^2c_2^3 \\ &\quad + 64c_3c_2^5 + 2c_8 - 56c_2c_3c_5 + 60c_4c_3c_2^2)e_j^6 \\ &\quad + O(e_j^7)]. \end{aligned} \tag{25}$$

Using equations (21), (22), (23) and (25) in Algorithms 4–6, we achieve the same equality as given below:

$$s_{j+1} = \alpha - c_3c_2^2e_j^5 + O(e_j^6),$$

which implies that

$$e_{i+1} = -c_3c_2^2e_j^5 + O(e_j^6).$$

The above equality shows that the orders of convergence of Algorithms 4–6 are at least five. □

IV. NUMERICAL RESULTS

To prove the supremacy of the suggested iteration schemes on the other comparable methods, we included four real-life engineering problems and seven arbitrary problems in the form of transcendental and algebraic equations and compared the numerical results of the suggested iteration schemes (Algorithms 1–6) with Noor's method one (NM1) [33], Noor's method two (NM2) [33], Ostrowski's method (OM) [34], Traub's method (TM) [40] and modified Halley's method (MHM) [31].

To show the numerical comparison of the above defined methods with our presented algorithms, we consider the following nine examples. In all examples, we take $\varepsilon = 10^{-15}$ in the following stopping criterion of the computer programs $|s_{j+1} - s_j| < \varepsilon$. We solved all the test examples with the aid of the computer program Maple 15.

Example 1: Open Channel Flow Problem The water flow in an open channel with a uniform flow condition is given by Manning's equation [26], having the following standard form:

$$Water\ Flow = F = \frac{\sqrt{m}ar^{\frac{2}{3}}}{n}, \tag{26}$$

where m, a and r represent the slope, area and hydraulic radius of the corresponding channel respectively and n

TABLE 1. Numerical comparison among different algorithms for the engineering problems $\xi_1 - \xi_4$.

Method	N	$\lvert\xi(s_{j+1})\rvert$	s_{j+1}	$\sigma = \lvert s_{j+1} - s_j\rvert$	$ACOC$	CPU Time
$\xi_1(s)$, $s_0 = 0.40$						
NM1	6	$2.230770e-24$	1.46509122029582464237602074553413096	$1.280653e-12$	2	3.174
NM2	3	$5.751429e-27$	1.46509122029582464237602091020202537	$4.220179e-09$	3	3.237
OM	3	$9.825352e-31$	1.46509122029582464237602090977863844	$6.464833e-08$	4	3.268
TM	3	$7.765624e-44$	1.46509122029582464237602090977856610	$4.884637e-11$	4	3.330
MHM	3	$4.020841e-64$	1.46509122029582464237602090977856610	$5.749371e-13$	5	3.409
Algorithm 1	2	$6.125642e-20$	1.46509122029582464238053102366591282	$2.289868e-03$	6	2.249
Algorithm 2	2	$2.716057e-20$	1.46509122029582464237802065566113214	$1.999354e-03$	6	2.311
Algorithm 3	2	$2.399230e-20$	1.46509122029582464237778738571302562	$1.958414e-03$	6	2.374
Algorithm 4	4	$1.524713e-74$	1.46509122029582464237602090977856610	$1.272498e-12$	5	2.355
Algorithm 5	4	$1.175533e-72$	1.46509122029582464237602090977856610	$1.224679e-14$	5	2.418
Algorithm 6	4	$2.299801e-72$	1.46509122029582464237602090977856610	$1.400597e-14$	5	2.496
$\xi_2(s)$, $s_0 = 2050.00$						
NM1	9	$3.688522e-28$	4305.30991366612556304019892944632561606	$3.947209e-13$	2	2.651
NM2	4	$2.919985e-37$	4305.30991366612556304019892944634208621	$9.938805e-11$	3	2.714
OM	3	$2.865067e-36$	4305.30991366612556304019892944634208621	$6.456917e-07$	4	2.760
TM	3	$6.063382e-31$	4305.30991366612556304019892944634209814	$1.002459e-05$	4	2.791
MHM	3	$1.311971e-69$	4305.30991366612556304019892944634208621	$1.526816e-11$	5	2.837
Algorithm 1	2	$5.380154e-19$	4305.30991366612556304017490575908846475	$1.299948e+00$	6	2.472
Algorithm 2	2	$5.757982e-21$	4305.30991366612556304019867233849842696	$6.102740e-01$	6	2.519
Algorithm 3	2	$3.119649e-18$	4305.30991366612556304005962958656859603	$1.742335e+00$	6	2.565
Algorithm 4	3	$3.154009e-73$	4305.30991366612556304019892944634208621	$8.601532e-10$	5	2.098
Algorithm 5	3	$1.031794e-77$	4305.30991366612556304019892944634208621	$3.964046e-13$	5	2.145
Algorithm 6	3	$3.404338e-79$	4305.30991366612556304019892944634208621	$2.003672e-13$	5	2.192
$\xi_3(s)$, $s_0 = 0.10$						
NM1	7	$9.675391e-26$	0.27775954284172065909591015386800788	$1.053628e-13$	2	2.623
NM2	3	$1.203488e-18$	0.27775954284172065896195690011981922	$6.173552e-07$	3	2.685
OM	3	$3.158175e-34$	0.27775954284172065909591016463712051	$2.153605e-09$	4	2.732
TM	3	$3.260304e-39$	0.27775954284172065909591016463712048	$1.412011e-10$	4	2.763
MHM	2	$2.057683e-15$	0.27775954284172043006718791209749563	$1.970771e-04$	5	2.810
Algorithm 1	2	$1.559630e-21$	0.27775954284172065909608375788470140	$2.520565e-04$	6	2.414
Algorithm 2	2	$1.552395e-21$	0.27775954284172065909608295267204378	$2.518612e-04$	6	2.461
Algorithm 3	2	$7.111881e-22$	0.27775954284172065909598932282992437	$2.211236e-04$	6	2.539
Algorithm 4	2	$2.581334e-19$	0.27775954284172065906717884086887426	$4.027591e-04$	5	2.282
Algorithm 5	2	$3.375345e-16$	0.27775954284172062152692154711491517	$5.567325e-04$	5	2.345
Algorithm 6	2	$5.537620e-16$	0.27775954284172059745990330048257529	$6.146273e-04$	5	2.407
$\xi_4(s)$, $s_0 = 2.00$						
NM1	4	$1.319023e-19$	1.92984624284786221696144418547353349	$5.000588e-10$	2	2.917
NM2	3	$3.958485e-15$	1.92984624284790801736694582212524590	$1.021691e-05$	3	2.965
OM	3	$2.230713e-36$	1.92984624284786221848752742786545651	$6.360563e-10$	4	2.996
TM	3	$8.395139e-34$	1.92984624284786221848752742786546620	$2.556739e-09$	4	3.073
MHM	2	$8.089146e-19$	1.92984624284786222784650752141958847	$1.079121e-04$	5	3.136
Algorithm 1	2	$1.745514e-26$	1.92984624284786221848752722591298239	$2.066340e-05$	6	2.211
Algorithm 2	2	$1.241265e-36$	1.92984624284786221848752742786545647	$4.206257e-07$	6	2.273
Algorithm 3	2	$2.834446e-23$	1.92984624284786221848719948816763864	$7.082722e-05$	6	2.336
Algorithm 4	2	$6.661957e-20$	1.92984624284786221771675230225238157	$1.780982e-04$	5	2.131
Algorithm 5	2	$9.238351e-18$	1.92984624284786232537339892619197953	$1.904797e-04$	5	2.163
Algorithm 6	2	$6.764358e-17$	1.92984624284786300111019378938122558	$2.835554e-04$	5	2.209

denotes Manning's roughness coefficient. For a rectangular shaped channel, having width b and depth of water in the channel s, then we may write:

$$a = bs, \quad \& \quad r = \frac{bs}{b+2s}.$$

Using these values in (26), we obtain:

$$F = \frac{\sqrt{m}bs}{n}\left(\frac{bs}{b+2s}\right)^{\frac{2}{3}}.$$

To find the depth of water in the channel for a given quantity of water, the above equation may be written in the form of non-linear function as:

$$\xi_1(s) = \frac{\sqrt{m}bs}{n}\left(\frac{bs}{b+2s}\right)^{\frac{2}{3}} - F.$$

We take the values of different parameters as $F = 14.15$ m^3/s, $b = 4.572m$, $m = 0.017$ *and* $n = 0.0015$. *We choose the initial guess* $s_0 = 0.4$ *to start the iteration process and the corresponding results through different iteration schemes are given in Tab. 1.*

Example 2: Adiabatic Flame Temperature Equation The adiabatic flame temperature equation is represented by the following relation:

$$\xi_2(s) = \Delta H + a_1(s-298) + \frac{a_2}{2}\left(s^2 - 298^2\right) + \frac{a_3}{3}\left(s^3 - 298^3\right),$$

where $\Delta H = -57798, a_1 = 7.256, a_2 = 0.002298, a_3 = 0.00000283$. *For further details, see [36], [37] and the references therein. The above function is actually a polynomial*

TABLE 2. Numerical comparison among different algorithms for transcendental and Algebraic problems $\xi_5 - \xi_9$.

Method	N	$\lvert\xi(s_{j+1})\rvert$	s_{j+1}	$\sigma = \lvert s_{j+1} - s_j\rvert$	$ACOC$	CPU Time
$\xi_5(s)$, $s_0 = 0.20$						
NM1	58	$9.603649e-15$	1.36523001341409626419371470793292919	$3.444222e-08$	2	2.854
NM2	6	$1.075324e-20$	1.36523001341409684576015564644490256	$1.340024e-07$	3	2.916
OM	4	$6.197363e-20$	1.36523001341409684576455975890305487	$1.436482e-05$	4	2.962
TM	5	$1.936699e-57$	1.36523001341409684576080682898166608	$5.616857e-15$	4	2.993
MHM	4	$8.480209e-65$	1.36523001341409684576080682898166608	$1.880599e-13$	5	3.041
Algorithm 1	3	$7.355896e-30$	1.36523001341409684576080682898122063	$1.991716e-05$	6	2.537
Algorithm 2	3	$1.052108e-19$	1.36523001341409684575443558870086825	$9.816529e-04$	6	2.600
Algorithm 3	3	$1.559045e-54$	1.36523001341409684576080682898166608	$1.537922e-09$	6	2.646
Algorithm 4	4	$1.899683e-38$	1.36523001341409684576080682898166608	$4.867159e-07$	5	2.115
Algorithm 5	4	$1.108846e-64$	1.36523001341409684576080682898166608	$2.151827e-13$	5	2.193
Algorithm 6	4	$1.058588e-57$	1.36523001341409684576080682898166608	$5.355234e-12$	5	2.255
$\xi_6(s)$, $s_0 = 0.60$						
NM1	46	$1.022435e-24$	2.15443469003188372175929349309369117	$3.977321e-13$	2	3.083
NM2	8	$9.161675e-17$	2.15443469003188371517988362885526052	$2.969171e-06$	3	3.314
OM	5	$1.145348e-51$	2.15443469003188372175929356651935050	$1.874175e-13$	4	3.160
TM	5	$1.033331e-29$	2.15443469003188372175929356652009258	$5.219308e-08$	4	3.223
MHM	4	$3.229596e-33$	2.15443469003188372175929356651935073	$3.980564e-07$	5	3.285
Algorithm 1	3	$4.472343e-29$	2.15443469003188372175929356651613871	$2.765363e-05$	6	2.274
Algorithm 2	4	$4.492834e-62$	2.15443469003188372175929356651935050	$8.751585e-11$	6	2.336
Algorithm 3	3	$6.218046e-37$	2.15443469003188372175929356651935050	$1.356049e-06$	6	2.415
Algorithm 4	4	$1.507877e-19$	2.15443469003188372174846482450084130	$7.518652e-04$	5	2.184
Algorithm 5	4	$7.516603e-47$	2.15443469003188372175929356651935050	$8.100982e-10$	5	2.246
Algorithm 6	4	$2.788275e-22$	2.15443469003188372175927354266273303	$6.643663e-05$	5	2.309
$\xi_7(s)$, $s_0 = 2.00$						
NM1	5	$3.163807e-17$	0.40999201798913710058260361746084531	$5.629386e-09$	2	3.331
NM2	4	$3.382231e-18$	0.40999201798913713493940881351470300	$1.512669e-06$	3	3.347
OM	4	$6.783183e-56$	0.40999201798913713162125837649907539	$1.630792e-14$	4	3.425
TM	5	$6.063382e-56$	0.40999201798913713162125837649907539	$1.586231e-14$	4	3.487
MHM	3	$4.499472e-42$	0.40999201798913713162125837649907539	$1.880726e-08$	5	3.550
Algorithm 1	3	$2.055736e-75$	0.40999201798913713162125837649907539	$1.086654e-12$	6	2.145
Algorithm 2	3	$1.738462e-68$	0.40999201798913713162125837649907539	$1.551045e-11$	6	2.192
Algorithm 3	2	$1.658501e-15$	0.40999201798913875870050922484589056	$1.059288e-02$	6	2.254
Algorithm 4	3	$6.196919e-52$	0.40999201798913713162125837649907539	$2.349916e-09$	5	2.361
Algorithm 5	3	$1.744151e-53$	0.40999201798913713162125837649907539	$1.064704e-10$	5	2.423
Algorithm 6	3	$3.008168e-67$	0.40999201798913713162125837649907539	$1.881797e-13$	5	2.471
$\xi_8(s)$, $s_0 = 0.50$						
NM1	5	$1.722167e-17$	−0.99267257254612160050741528837974267	$1.198166e-09$	2	3.295
NM2	5	$2.119699e-29$	−0.99267257254612160321469740373981011	$8.821302e-11$	3	3.343
OM	114	$4.228166e-22$	−0.99267257254612160321476387137685218	$2.461460e-06$	4	3.624
TM	4	$1.406203e-15$	−0.99267257254612182427273413023422613	$7.577371e-05$	4	3.670
MHM	5	$6.634668e-57$	−0.99267257254612160321469740374314232	$2.373401e-12$	5	3.701
Algorithm 1	3	$4.429119e-19$	−0.99267257254612160314507075458005786	$3.983237e-04$	6	2.464
Algorithm 2	3	$2.634426e-16$	−0.99267257254612156180097697588726648	$1.154689e-03$	6	2.527
Algorithm 3	3	$2.296726e-47$	−0.99267257254612160321469740374314232	$7.693481e-09$	6	2.573
Algorithm 4	4	$8.655181e-65$	−0.99267257254612160321469740374314232	$8.409556e-12$	5	2.170
Algorithm 5	4	$8.569042e-41$	−0.99267257254612160321469740374314232	$4.293505e-09$	5	2.232
Algorithm 6	4	$1.338673e-33$	−0.99267257254612160321469740374314211	$1.179127e-07$	5	2.310
$\xi_9(s)$, $s_0 = 2.50$						
NM1	7	$4.723335e-18$	1.30296400121601255525948434085137479	$5.396458e-09$	2	3.455
NM2	3	$4.160283e-22$	1.30296400121601255253235453196212347	$3.445369e-07$	3	3.503
OM	3	$6.549890e-35$	1.30296400121601255253211430697335799	$1.290378e-08$	4	3.580
TM	3	$2.330912e-38$	1.30296400121601255253211430697335803	$2.011919e-09$	4	3.644
MHM	2	$6.275008e-17$	1.30296400121601251629867379454820012	$3.463864e-03$	5	3.690
Algorithm 1	2	$6.218574e-15$	1.30296400121600896177442065242678728	$3.001902e-02$	6	2.235
Algorithm 2	2	$4.357422e-15$	1.30296400121601003644900804341887115	$2.829998e-02$	6	2.297
Algorithm 3	2	$9.927122e-15$	1.30296400121600682036716846236476526	$3.243867e-02$	6	2.328
Algorithm 4	2	$4.346672e-16$	1.30296400121601280351967011313397846	$1.354926e-02$	5	1.765
Algorithm 5	2	$1.128699e-15$	1.30296400121601190079351332048873270	$6.727718e-03$	5	1.812
Algorithm 6	2	$7.564752e-16$	1.30296400121601211572470440302029921	$6.203060e-03$	5	1.859

of degree three and by the fundamental theorem of Algebra, it must have exactly three roots. Among these roots, $\alpha = 4305.3099136661$ *is a simple one which we approximated through the proposed methods by choosing the initial guess* $s_0 = 2050$ *and the numerical results have been shown in Tab. 1.*

Example 3: Fraction Conversion of Nitrogen-Hydrogen to Ammonia We take this example from [6], which describe the fractional conversion of nitrogen-hydrogen feed to ammonia, usually known as fractional conversion. In this problem, the values of temperature and pressure have been taken as 500^0C *and 250 atm respectively. This problem has the*

TABLE 3. Comparison of number of iterations required for different iterative methods for $\varepsilon = 10^{-100}$.

Method	$\xi_1(s)$ $s_0 = 0.40$	$\xi_2(s)$ $s_0 = 2050.00$	$\xi_3(s)$ $s_0 = 0.10$	$\xi_4(s)$ $s_0 = 2.00$	$\xi_5(s)$ $s_0 = 0.20$	$\xi_6(s)$ $s_0 = 0.60$	$\xi_7(s)$ $s_0 = 2.00$	$\xi_8(s)$ $s_0 = 0.50$	$\xi_9(s)$ $s_0 = 2.50$
NM1	08	09	09	07	61	48	08	08	10
NM2	05	05	06	05	08	10	06	07	05
OM	04	04	06	04	06	06	05	116	04
TM	05	04	06	04	06	06	05	06	04
MHM	03	03	06	03	05	05	04	06	04
Algorithm 1	03	03	03	03	04	04	04	04	04
Algorithm 2	03	03	03	03	04	05	04	05	04
Algorithm 3	03	03	03	03	04	04	04	04	04
Algorithm 4	04	04	03	03	05	05	04	05	04
Algorithm 5	04	04	04	04	05	05	04	05	04
Algorithm 6	04	04	04	04	05	05	04	05	04

FIGURE 1. The colormap used for generating polynomiographs.

following non-linear form:

$$-0.186 - \frac{8s^2\,(s-4)^2}{9\,(s-2)^3} = 0,$$

which can be easily reduced to the following polynomial:

$$\xi_3(s) = s^4 - 7.79075s^3 + 14.7445s^2 + 2.511s - 1.674.$$

Since the degree of the above polynomial is four, so, it must have exactly four roots. By definition, the fraction conversion lies in $(0, 1)$ *interval, so only one real root exists in this interval which is 0.2777595428. The other three roots have no physical meanings. We started the iteration process by the initial guess* $s_0 = 0.1$. *The numerical results through different methods have been shown in Tab. 1.*

Example 4: Finding Volume from Van Der Wall's Equation In Chemical Engineering, the van der Waal's equation has been used for interpreting real and ideal gas behavior [41], having the following form:

$$(P + \frac{A_1 n^2}{V^2})\,(V - nA_2) = nRT.$$

By taking the specific values of the parameters of the above equation, we can easily convert it to the following non-linear function:

$$\xi_4(s) = 0.986s^3 - 5.181s^2 + 9.067s - 5.289,$$

where s represents the volume that can easily be found by solving the function ξ_4. *Since the degree of the polynomial is three, so it must possess three roots. Among these roots, there is only one positive real root* 1.9298462428 *which is feasible because the volume of the gas can never be negative. We start the iteration process with the initial guess* $s_0 = 2.0$ *and their results can be seen in Tab. 1.*

Example 5: Transcendental and Algebraic Problems To numerically analyze the suggested algorithms, we consider the following five transcendental and algebraic equations and their numerical results can be seen in Tab. 2.

$$\xi_5(s) = s^3 + 4s^2 - 10,$$

$$\xi_6(s) = s^3 - 10,$$

$$\xi_7(s) = s^2 + \sin(\frac{s}{5}) - \frac{1}{4},$$

$$\xi_8(s) = se^{s^2} - \sin^2(s) - 3\cos(s) + 5,$$

$$\xi_9(s) = \ln(s) - \cos(s).$$

Algorithm 7: Polynomiograph's Generation

Input: $p \in \mathbb{C}$ — polynomial, $A \subset \mathbb{C}$ — area, m — maximum number of iterations, I — iteration method, ϵ — accuracy, colormap $[0 \ldots C-1]$ — colormap with C colors.

Output: Polynomiograph for the complex polynomial p in area A.

```
for z_0 ∈ A do
    i = 0
    while i ≤ m do
        z_{j+1} = I(z_j)
        if |z_{j+1} − z_j| < ε then
            break
        i = i + 1
    color z_0 by means of colormap.
```

Tables 1–2 exhibit the numerical comparison of the suggested iteration schemes with the other similar-nature existing algorithms. In the columns of the presented tables, N represents the iterations consumed by different algorithms, $|\xi(s)|$ denotes the absolute value of $\xi(s)$ at final approximation, s_{j+1} shows the final approximated root, $|s_{j+1} - s_j|$ represents the absolute distance between the two consecutive approximations, (ACOC) denotes the approximated computational order of convergence having the following approximated formula:

$$ACOC \approx \frac{ln\frac{|(s_{j+1}-s_j)|}{|(s_j-s_{j-1})|}}{ln\frac{|(s_j-s_{j-1})|}{|(s_{j-1}-s_{j-2})|}}.$$

The above approximation was suggested in (2007) by Cordero and Torregrosa [12] and the last column represents CPU time consumption in seconds, taken by different iteration schemes to find the required approximate solution of the given problem.

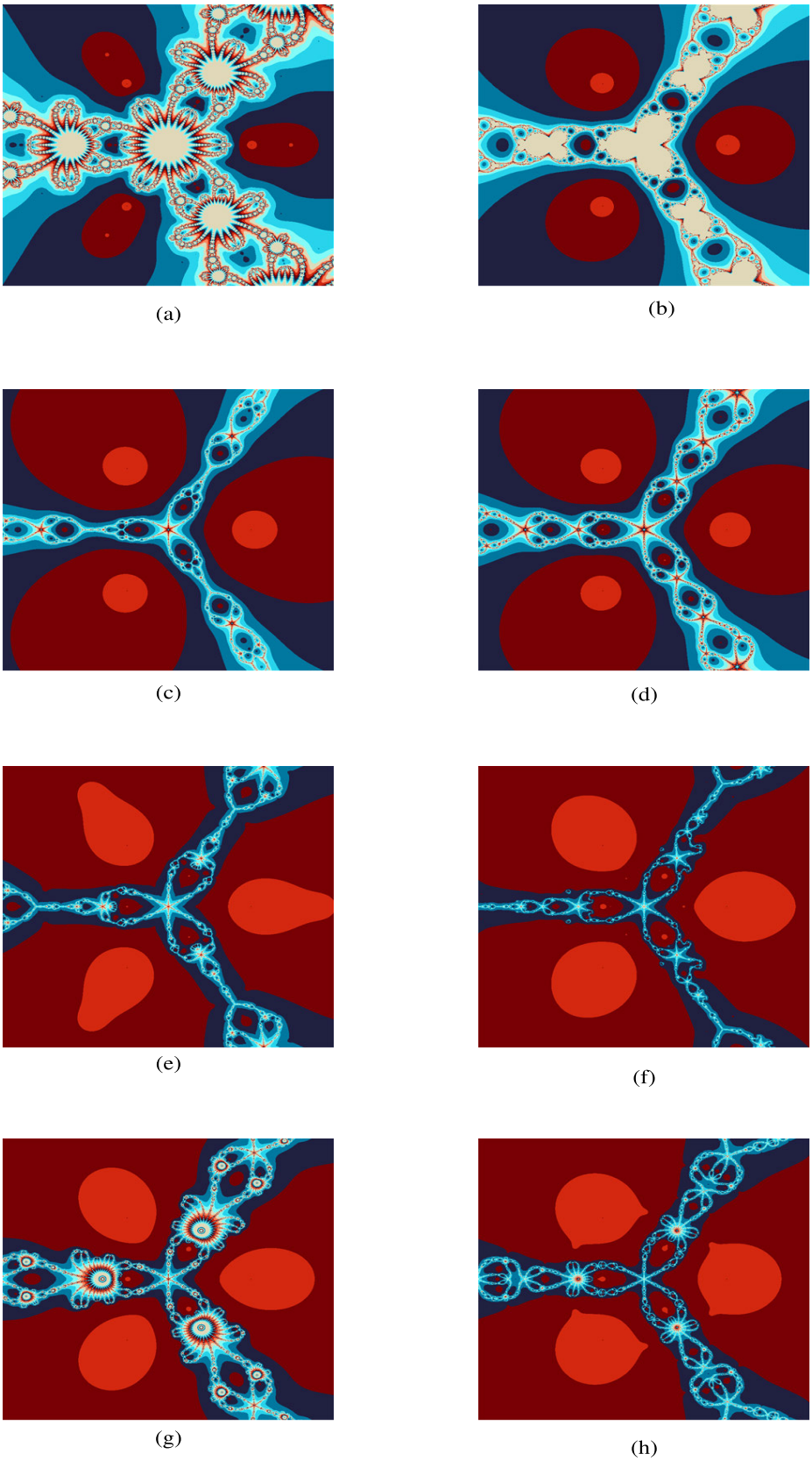


FIGURE 2. Polynomiographs associated with the polynomial $p_1(z)$. (a) stands for NM1, (b) for NM2, (c) for OM, (d) for TM, (e) for MHM, (f) for Algorithm II, (g) for Algorithm II, (h) for Algorithm II, (i) for Algorithm II, (j) for Algorithm II, and (k) for Algorithm II.

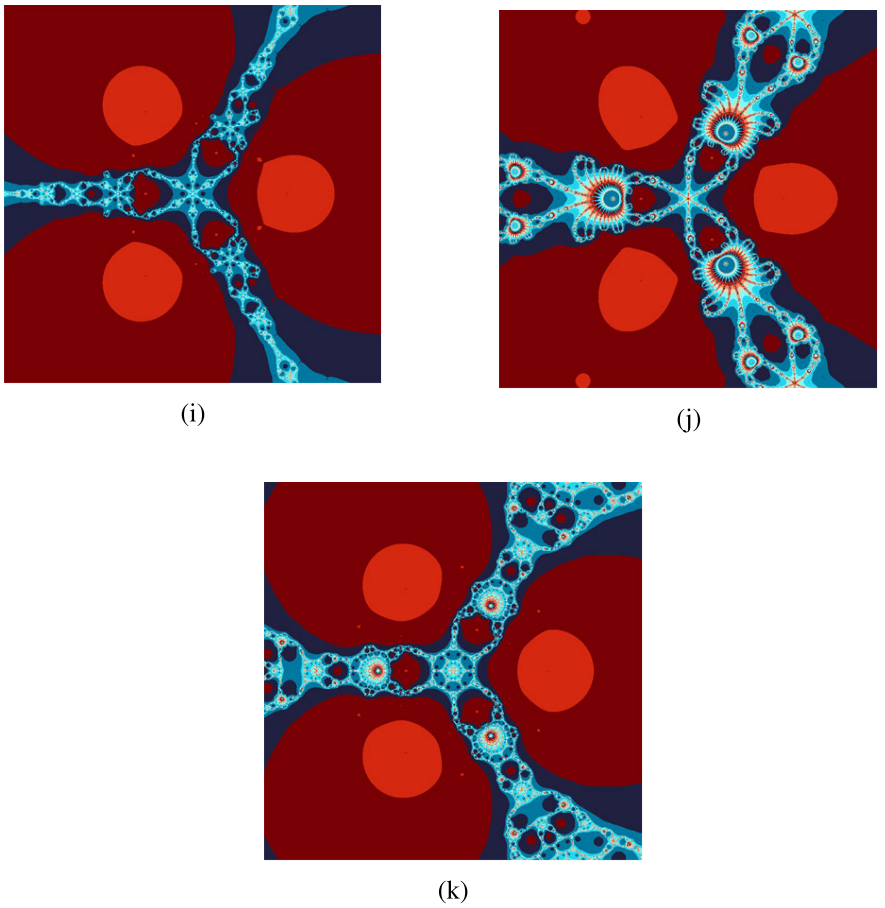


FIGURE 2. *(Continued.)* **Polynomiographs associated with the polynomial $p_1(z)$. (a) stands for NM1, (b) for NM2, (c) for OM, (d) for TM, (e) for MHM, (f) for Algorithm II, (g) for Algorithm II, (h) for Algorithm II, (i) for Algorithm II, (j) for Algorithm II, and (k) for Algorithm II.**

By carefully examining the numerical results contained in Tables 1–2, one can easily predict that the newly constructed iteration schemes took less number of iterations and determined the approximate roots of different test problems with better precision than the approximations obtained through the other five comparable methods.

Also, the proposed iterations schemes performing fast and achieved the defined stopping criterion for determining the approximate solutions of different test problems by taking less CPU time in seconds that can be seen in the last column of Tables 1–2. Shorty, we can claim that the performance and efficiency of the proposed iterations schemes are better as compared to the other comparable existing iteration schemes. We did all the numerical calculations and the CPU time consumption in seconds by using the computer software Maple 15.

Table 3 exhibits the comparison of the iterations consumed by different algorithms with the newly presented methods for the root-finding of non-linear functions with the accuracy $\varepsilon = 10^{-100}$. Here the columns of the table denote the iterations' number for various test functions together with the initial guess s_0.

The numerical results as shown in Table 3, again certified the fast and better performance of the presented algorithms in terms of number of iterations for the above defined stopping criterion with the given accuracy. In all test examples, the proposed algorithms consumed less number of iterations in comparison with the other iterative algorithms.

V. POLYNOMIOGRAPHY

The problems related to finding the roots of polynomials have played a vital role in engineering and mathematical sciences. It is one of the oldest and most deeply studied mathematical problems as one can study the history of Mathematics that the ancient Greeks and Sumerians considered such practical problems in 3000 B.C, that can now be stated as a root-finding problems using modern mathematical language. In the seventeenth century, Newton suggested an algorithm for approximating the roots of polynomials. After that Cayley [9] studied the chaotic and strange behavior while applying Newton's method on cubic polynomial $x^3 - 1$ in the complex plane. The problem arose from Cayley was explained by Julia in 1919. The Julia sets bought many new discoveries i.e., Mandelbrot set and Fractal in 1970 [25]. The term "Polynomiography"

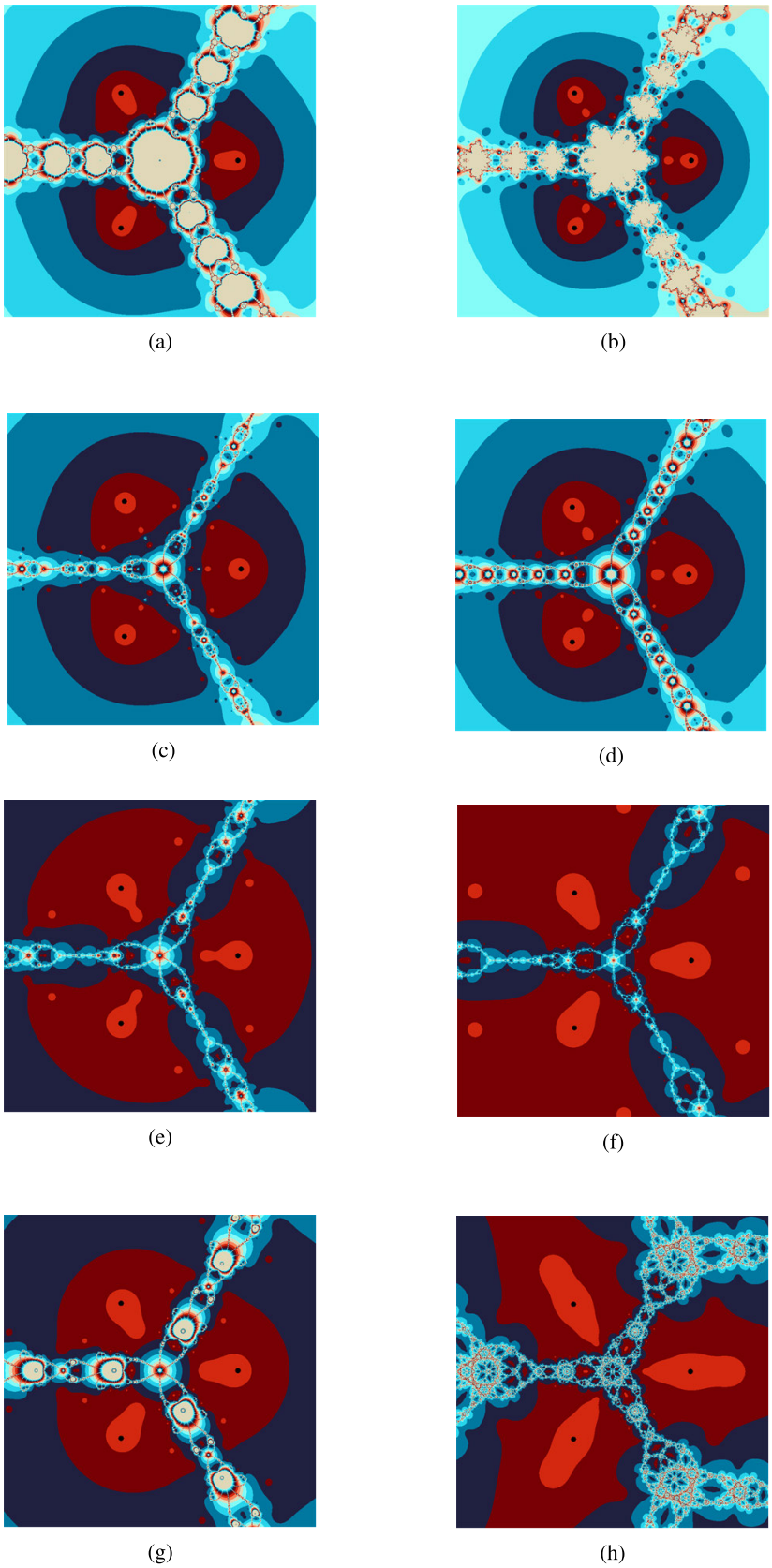
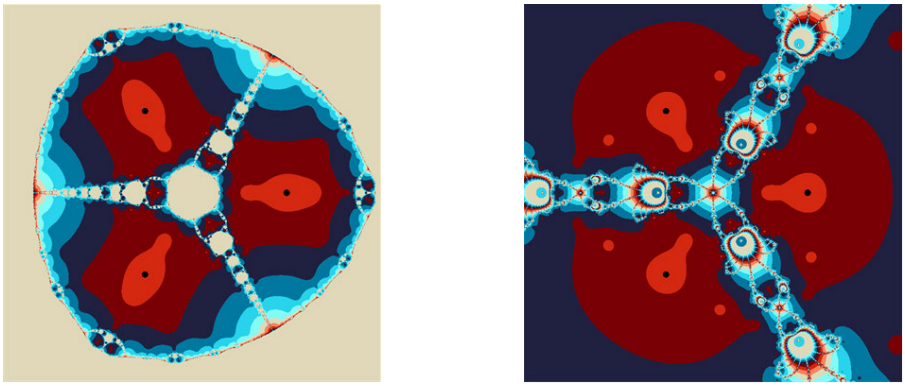
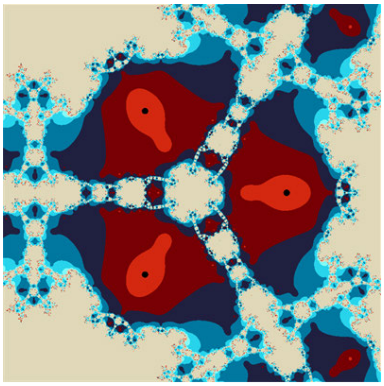


FIGURE 3. Polynomiographs associated with the polynomial $p_2(z)$. (a) stands for NM1, (b) for NM2, (c) for OM, (d) for TM, (e) for MHM, (f) for Algorithm II, (g) for Algorithm II, (h) for Algorithm II, (i) for Algorithm II, (j) for Algorithm II, and (k) for Algorithm II.



(i) (j)



(k)

FIGURE 3. *(Continued.)* **Polynomiographs associated with the polynomial $p_2(z)$. (a) stands for NM1, (b) for NM2, (c) for OM, (d) for TM, (e) for MHM, (f) for Algorithm II, (g) for Algorithm II, (h) for Algorithm II, (i) for Algorithm II, (j) for Algorithm II, and (k) for Algorithm II.**

was first introduced in 2005 by Kalantari [17], [18]. He defined polynomiography as the combination of both science and art related to visualization of the root-finding process for a polynomial in the complex plane. The individual image produced as a result of polynomiography is thus called a ''Polynomiograph''. The polynomiographs can be generated by any iterative scheme i.e., Newton's method, Halley's method etc. Polynomiography has vast applications in many fields of science and art. In the last few years, many researchers worked on polynomiography and generated new and nice-looking images through different algorithms. Soleimani *et al.* [39], suggested some new iterative methods free from derivatives and then presented their fractal behaviors by means of their basins of attraction. In [22], the authors generated beautiful polynomiographs using Ishikawa and Mann iterations that were quite new and looked aesthetically pleasing comparing to the ones from standard Picard iteration. In 2016, the authors modified Abbasbandy's method [1] and then presented polynomiographs through the modified method [20]. Gdawiec [13] in 2017, used three different approaches, i.e., affine and s-convex combination, the use of iteration processes from fixed point theory and multi-step polynomiography and obtained new and diverse fractal patterns that have many applications in textile or ceramics patterns. In [24], the authors presented some new graphical objects obtained by the use of the escape time algorithm and the derived criteria. They presented graphical examples by means of Jungck-CR iteration process with s-convexity. In 2019, Kalantari and Lee [19] introduced new ways of creating mathematical art through a novel Newton-Ellipsoid method for solving polynomial equations. The nature of polynomiographs under Newton-Ellipsoid seems to be very different from the other images, which opens the possibility of generating novel artistic images.

A. APPLICATIONS

To generate a polynomiograph via a computer program, we have to select an initial rectangle R containing the roots of the polynomial. Then for each point z_0 in the region, we run an iterative method, and then color the point corresponding to z_0 is depended upon the approximate convergence of the truncated orbit to a root, or lack thereof. The resolution of the image depends on our discretization of the rectangle R.

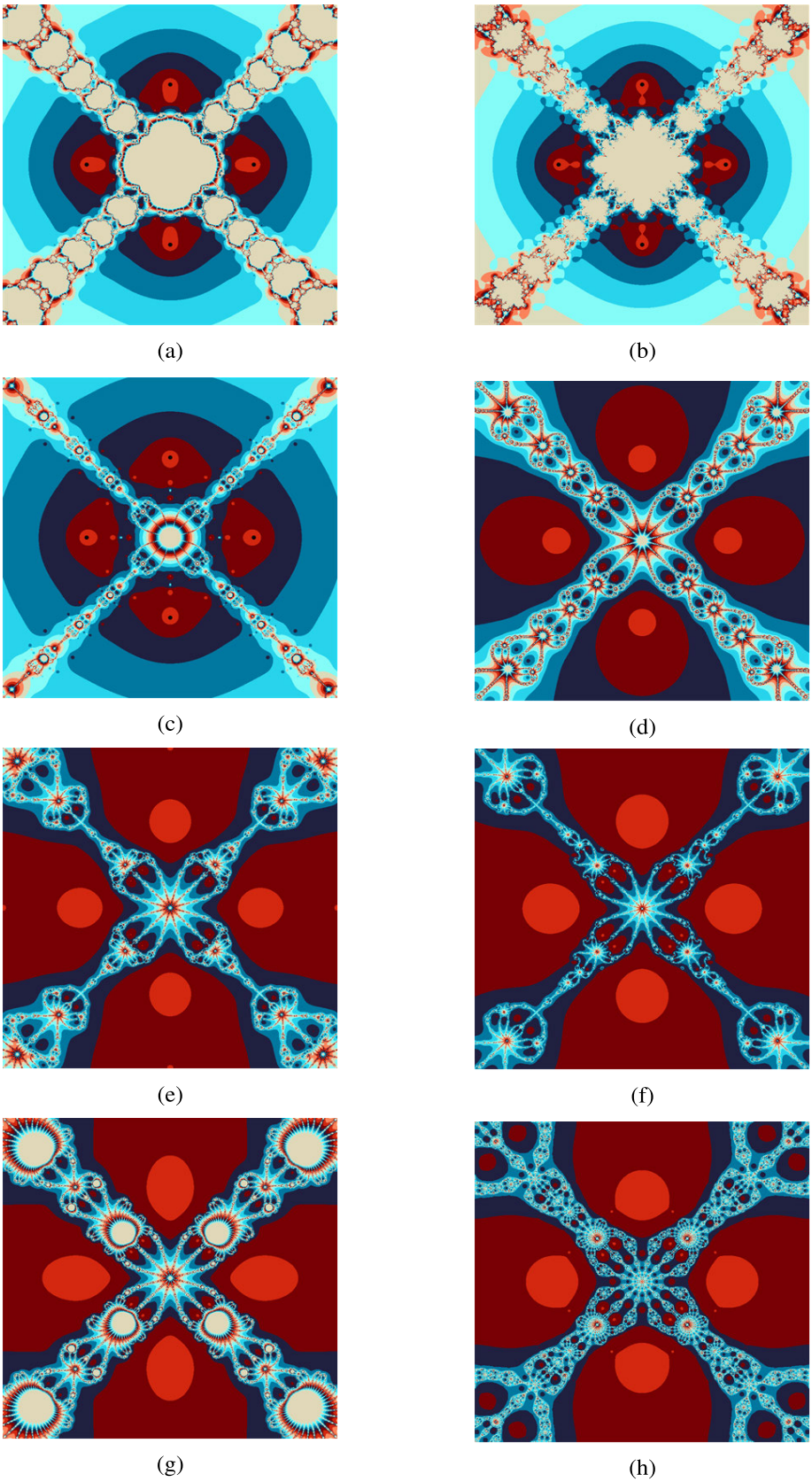


FIGURE 4. Polynomiographs associated with the polynomial $p_3(z)$. (a) stands for NM1, (b) for NM2, (c) for OM, (d) for TM, (e) for MHM, (f) for Algorithm II, (g) for Algorithm II, (h) for Algorithm II, (i) for Algorithm II, (j) for Algorithm II, and (k) for Algorithm II.

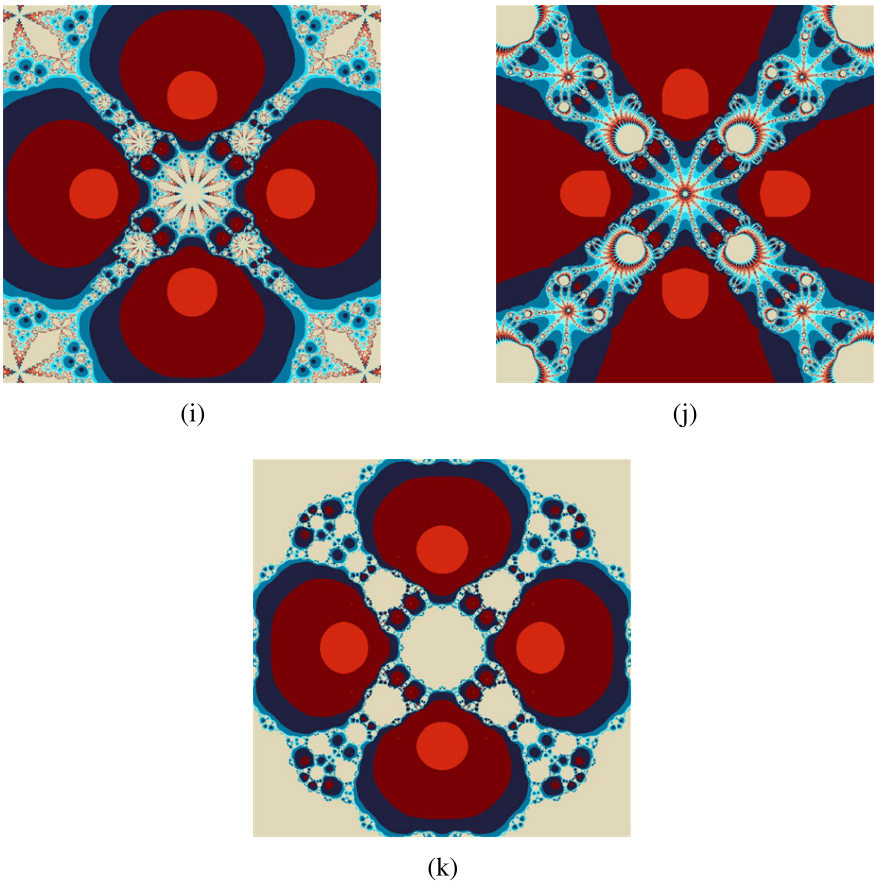


FIGURE 4. *(Continued.)* **Polynomiographs associated with the polynomial $p_3(z)$. (a) stands for NM1, (b) for NM2, (c) for OM, (d) for TM, (e) for MHM, (f) for Algorithm II, (g) for Algorithm II, (h) for Algorithm II, (i) for Algorithm II, (j) for Algorithm II, and (k) for Algorithm II.**

For example, discretizing R into 2000 by 2000 grid yields a high-resolution image.

According to the Fundamental Theorem of Algebra, any complex polynomial with complex coefficients $\{c_n, c_{n-1}, \ldots, c_1, c_0\}$:

$$p(z) = c_n z^n + c_{n-1} z^{n-1} + \ldots + c_1 z + c_0 \tag{27}$$

or by its zeros (roots) $\{r_1, r_2, \ldots, r_{n-1}, r_n\}$:

$$p(z) = (z - r_1)(z - r_2) \ldots (z - r_n) \tag{28}$$

of degree n has n roots (zeros) which may or may not be distinct. The degree of polynomial describes the number of basins of attraction and localization of basins can be controlled by placing roots on the complex plane manually.

Usually, the polynomiographs are colored and their coloring depends upon the number of iterations needed to approximate the roots of some polynomial with given accuracy and a chosen iterative scheme. The general and base algorithm for the generation of polynomiograph is presented in the following Algorithm 7.

In Algorithm 7, the convergence test $(z_j + 1, z_j, \epsilon)$ returns TRUE if the applied method has converged to the root, and FALSE otherwise. The most common and widely used convergence test has the following standard form:

$$|z_{j+1} - z_j| < \varepsilon, \tag{29}$$

where z_{j+1} and z_j are two consecutive points in an iteration process and $\varepsilon > 0$ is a given accuracy. In this paper, we also use the stopping criterion (29).

Using newly developed algorithms and other similar existing methods, we obtained novel, colorful, attractive and interesting polynomiographs. The different coloring of polynomiographs relies on the number of iterations requires to approximate the roots of the polynomial with given accuracy ε. A large number of such images can be generated by giving different values to the parameter m, where m represents the upper bound of the number of iterations.

We consider the following complex polynomials for the purpose of polynomiographs' generation through the proposed algorithms and compare them with the other well-known two-step algorithms.

$$p_1(z) = z^3 - 1, \quad p_2(z) = (z^3 - 1)^2, \quad p_3(z) = z^4 - 1,$$
$$p_4(z) = (z^4 - 1)^2.$$

We used the computer program Mathematica 12.0 for creating all the presented images by taking the accuracy

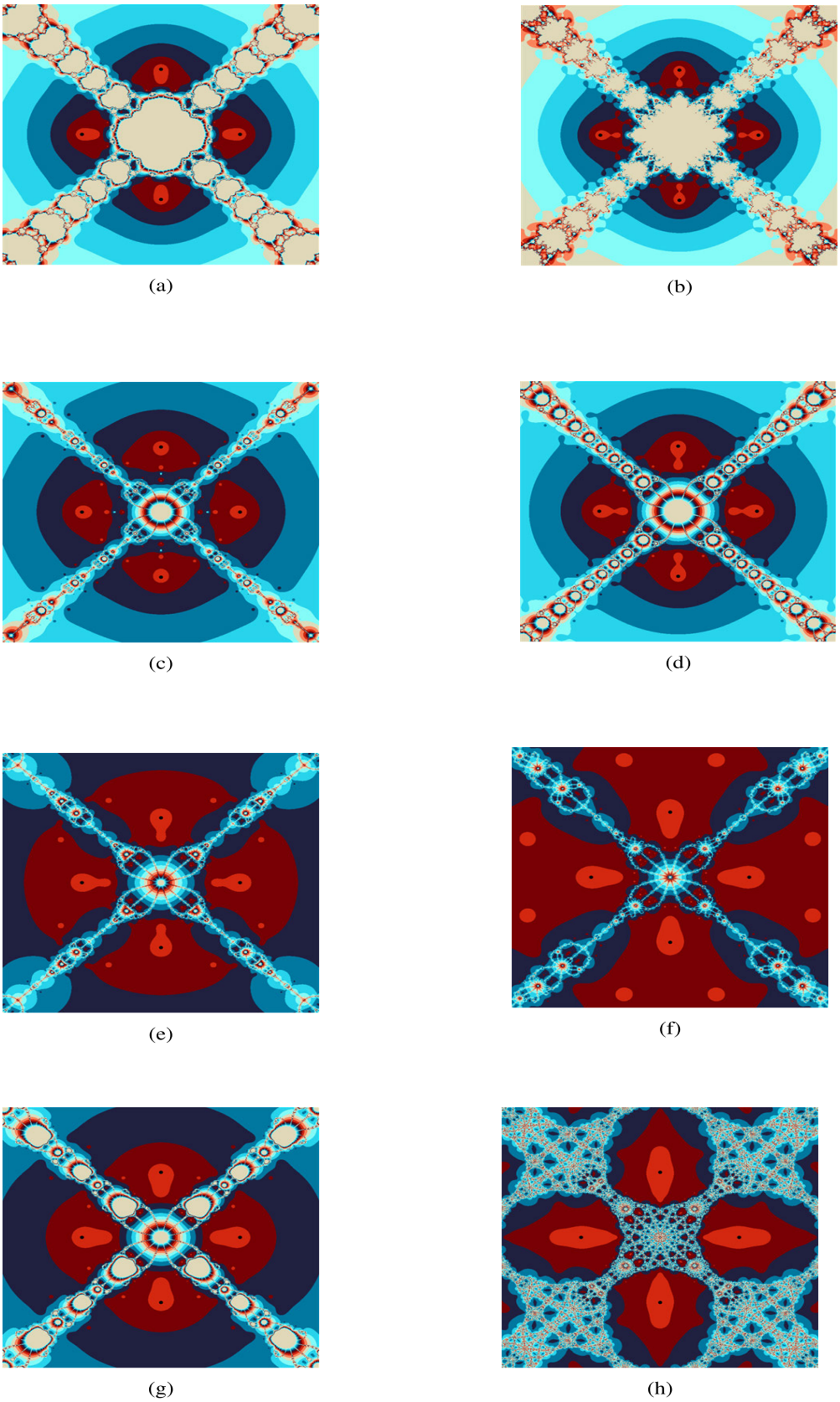
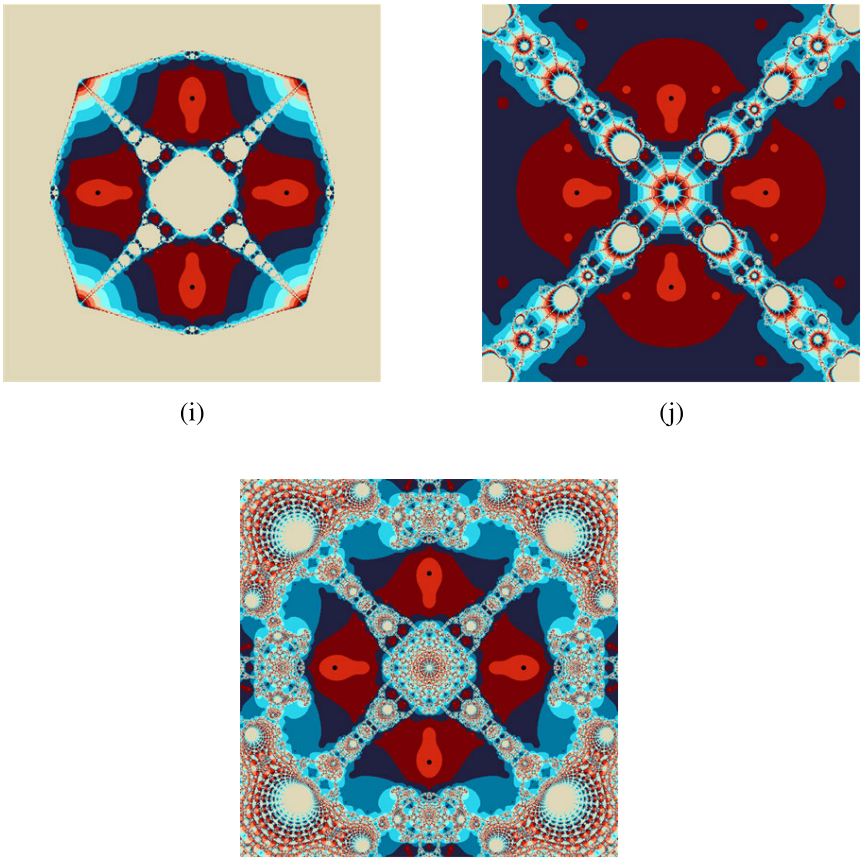


FIGURE 5. Polynomiographs associated with the polynomial $p_4(z)$. (a) stands for NM1, (b) for NM2, (c) for OM, (d) for TM, (e) for MHM, (f) for Algorithm II, (g) for Algorithm II, (h) for Algorithm II, (i) for Algorithm II, (j) for Algorithm II, and (k) for Algorithm II.



(i) (j)

(k)

FIGURE 5. *(Continued.)* **Polynomiographs associated with the polynomial $p_4(z)$. (a) stands for NM1, (b) for NM2, (c) for OM, (d) for TM, (e) for MHM, (f) for Algorithm II, (g) for Algorithm II, (h) for Algorithm II, (i) for Algorithm II, (j) for Algorithm II, and (k) for Algorithm II.**

$\varepsilon = 0.01$, and the upper bound of the number of iterations $m = 15$. The colormap used for the coloring of iterations in the generation of polynomiographs is presented in the following Fig. 1.

Example 6 (Polynomiographs for the Polynomial $p_1(z)$ Via Different Iteration Schemes): In this example, we compare the dynamical results of different iterative methods with our presented algorithms by considering the cubic polynomial $z^3 - 1$. This complex polynomial has three distinct zeros: 1, $-\frac{1}{2} - \frac{\sqrt{3}}{2}i$, $-\frac{1}{2} + \frac{\sqrt{3}}{2}i$. We ran all the algorithms to obtain the simple zeros of the considered polynomials and the results in the form of polynomiographs are presented in Fig. 2.

Example 7 (Polynomiographs for the Polynomial $p_2(z)$ Via Different Iteration Schemes): This example includes the dynamical comparison of the proposed algorithms with different iterative methods of the same nature by considering the complex polynomial $(z^3 - 1)^2$. This complex polynomial has six degree and according to the fundamental theorem of Algebra, it has exactly six roots. These roots are not simple and have a multiplicity 2. Out of these six roots, only three roots are distinct which are: 1, $-\frac{1}{2} - \frac{\sqrt{3}}{2}i$, $-\frac{1}{2} + \frac{\sqrt{3}}{2}i$. With the aid of computer program, We ran all the algorithms to obtain the simple zeros of the considered polynomials and the results in the form of polynomiographs are presented in Fig. 3.

Example 8 (Polynomiographs for the Polynomial $p_3(z)$ Via Different Iteration Schemes): In this experiment, we consider the quartic polynomial $z^4 - 1$ to show the dynamical aspects of different algorithms. This polynomial has four simple zeros namely: -1, $-i$, i, 1 which can be easily seen on the complex plane of the presented images. We obtained the polynomiographs through the proposed algorithms and then compared them with the other methods by means of computer technology. The presented figures showing the regions of convergence of the considered algorithms and certified that the proposed algorithms possess larger convergence areas than the other methods in comparison. The dynamic results in the form of polynomiographs are presented in Fig. 4.

Example 9 (Polynomiographs for the Polynomial $p_4(z)$ Via Different Iteration Schemes): In the last and final experiment, we show the dynamics of different algorithms by taking the complex polynomial $(z^4 - 1)^2$, having all zeros with multiplicity 2. This polynomial has eight degree with repeated roots and has only four distinct zeros namely: -1, $-i$, i, 1 which

can be easily seen on the complex plane of the presented images. The repeated roots appear with the same colors on the corresponding complex plane of the polynomiographs. The shades of the colors showing the performance and efficiency of the considered algorithm through which the polynomiograph have been generated. The darker the color, the more efficient will be the method and the presented images showing the superiority of the proposed algorithms that can be seen from Fig. 5.

After observing the generated images carefully, one can predict two important properties of the algorithms. One of them is the convergence speed of the considered algorithm, which can be depicted by the shade of the color. The darkness of the colors showing less number of iterations of the considered algorithm and vice versa. The second property of the algorithms is their dynamical aspects. The low dynamics are in those specific areas which contain a small variation of colors, whereas in areas having large variation of colors the dynamics are high. The appearance of black color in the presented images locate those particular places in which the solution cannot be achieved for the given number of iterations. The same color areas in the presented figures represent the same number of iterations consumed by different algorithms to approximate the solution and give a similar view to the contour lines on the map.

VI. CONCLUDING REMARKS

Based on Taylor's series expansion, six new iteration schemes for the solution of non-linear functions have been presented, and three of them are second derivative free. We approximated the second derivative by means of Taylor's series that results in some new efficient techniques. The convergence criterion of the proposed methods has been discussed. It is established that the suggested iteration schemes bearing the convergence of orders five and six. The numeric and dynamic comparisons of the proposed methods have been presented by considering some engineering and arbitrary test problems. The overall numerical results contained in Tables 1–2, certified the faster and more accurate approximations to the exact solutions than the other compared methods of the same kind. The dynamics of the suggested methods in the form of polynomiographs represented the convergence with a larger area in comparison with the other compared methods. These dynamics also revealed the faster convergence speed and other dynamical features of the proposed algorithms and proved the superiority of the suggested methods among the other ones in comparison. Using the idea of this paper, one can derive a class of new iterative algorithms and create some new mathematical art through these algorithms in future work. The proven results of this paper may give rise to further research in this field.

REFERENCES

[1] S. Abbasbandy, "Improving Newton–Raphson method for nonlinear equations by modified Adomian decomposition method," *Appl. Math. Comput.*, vol. 145, nos. 2–3, pp. 887–893, Dec. 2003.

[2] M. Abbas, A. A. Majid, A. I. M. Ismail, and A. Rashid, "The application of cubic trigonometric B-spline to the numerical solution of the hyperbolic problems," *Appl. Math. Comput.*, vol. 239, pp. 74–88, Jul. 2014.

[3] M. Abbas, A. A. Majid, A. I. M. Ismail, and A. Rashid, "Numerical method using cubic trigonometric B-spline technique for nonclassical diffusion problems," *Abstract Appl. Anal.*, vol. 2014, May 2014, Art. no. 849682.

[4] A. R. Alharbi, M. I. Faisal, F. A. Shah, M. Waseem, R. Ullah, and S. Sherbaz, "Higher order numerical approaches for nonlinear equations by decomposition technique," *IEEE Access*, vol. 7, pp. 44329–44337, 2019, doi: 10.1109/ACCESS.2019.2906470.

[5] D. Chen, I. K. Argyros, and Q. S. Qian, "A note on the Halley method in Banach spaces," *Appl. Math. Comput.*, vol. 58, nos. 2–3, pp. 215–224, Oct. 1993.

[6] G. V. Balaji and J. D. Seader, "Application of interval Newton's method to chemical engineering problems," *Reliable Comput.*, vol. 1, no. 3, pp. 215–223, Sep. 1995.

[7] R. Behl and E. Martínez, "A new high-order and efficient family of iterative techniques for nonlinear models," *Complexity*, vol. 2020, Jan. 2020, Art. no. 1706841, doi: 10.1155/2020/1706841.

[8] R. L. Burden and J. D. Faires, *Numerical Analysis*, 6th ed. Pacific Grove, CA, USA: Brooks/Cole, 1997.

[9] A. Cayley, "The Newton–Fourier imaginary problem," *Amer. J. Math.*, vol. 2, no. 1, p. 97, Mar. 1879

[10] D. Cebic, M. Paunovic, and N. M. Ralevic, "A variant of Sharma-Arora's optimal eighth-order family of methods for finding a simple root of nonlinear equation," in *Proc. IEEE 16th Int. Symp. Intell. Syst. Inf. (SISY)*, Subotica, Serbia, Sep. 2018, pp. 81–86, doi: 10.1109/SISY.2018.8524857.

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

[12] A. Cordero and J. R. Torregrosa, "Variants of Newton's method using fifth-order quadrature formulas," *Appl. Math. Comput.*, vol. 190, pp. 686–698, Jun. 2007.

[13] K. Gdawiec, "Fractal patterns from the dynamics of combined polynomial root finding methods," *Nonlinear Dyn.*, vol. 90, no. 4, pp. 2457–2479, Dec. 2017.

[14] A. Golbabai and M. Javidi, "A third-order Newton type method for nonlinear equations based on modified homotopy perturbation method," *Appl. Math. Comput.*, vol. 191, no. 1, pp. 199–205, Aug. 2007.

[15] J. M. Gutiérrez and M. A. Hernández, "An acceleration of Newton's method: Super-Halley method," *Appl. Math. Comput.*, vol. 117, nos. 2–3, pp. 223–239, Jan. 2001.

[16] A. S. Householder, *The Numerical Treatment of a Single Nonlinear Equation*. New York, NY, USA: McGraw-Hill, 1970.

[17] B. Kalantari, "Method of creating graphical works based on polynomials," U.S. Patent 6 894 705, May 17, 2005.

[18] B. Kalantari, "Polynomiography: From the fundamental theorem of algebra to art," *Leonardo*, vol. 38, no. 3, pp. 233–238, Jun. 2005.

[19] B. Kalantari and E. H. Lee, "Newton–Ellipsoid polynomiography," *J. Math. Arts*, vol. 13, no. 4, pp. 336–352, Nov. 2019, doi: 10.1080/17513472.2019.1600959.

[20] S. M. Kang, A. Naseem, W. Nazeer, M. Munir, and C. Yong, "Polynomiography via modified Abbasbanday's method," *Int. J. Math. Anal.*, vol. 11, no. 3, 133–149, 2017.

[21] D. E. Kincaid and E. W. Cheney, *Numerical Analysis*. Pacific Grove, CA, USA: Brooks/Cole, 1990.

[22] W. Kotarski, K. Gdawiec, and A. Lisowska, "Polynomiography via Ishikawa and Mann iterations," in *Advances in Visual Computing* (Lecture Notes in Computer Science), vol. 7431, G. Bebis, R. Boyle, B. Parvin, Eds. Berlin, Germany: Springer, 2012, pp. 305–313.

[23] 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 $\mathbb{R}$," *J. Comput. Appl. Math.*, vol. 330, pp. 676–694, Mar. 2018.

[24] Y. C. Kwun, M. Tanveer, W. Nazeer, K. Gdawiec, and S. M. Kang, "Mandelbrot and Julia sets via Jungck-CR iteration with S-convexity," *IEEE Access*, vol. 7, pp. 12167–12176, 2019, doi: 10.1109/ACCESS.2019.2892013.

[25] B. Mandelbrot, *The Fractal Geometry of Nature*. New York, NY, USA: W.H. Freeman, 1983.

[26] R. Manning, "On the flow of water in open channels and pipes," *Trans. Inst.*, vol. 20, pp. 161–207, 1891.

[27] 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, doi: 10.1109/ACCESS.2021.3049428.
[28] A. Naseem, M. A. Rehman, T. Abdeljawad, and F. Balibrea, "Numerical algorithms for finding zeros of nonlinear equations and their dynamical aspects," *J. Math.*, vol. 2020, Sep. 2020, Art. no. 2816843.
[29] A. Naseem, M. A. Rehman, and T. Abdeljawad, "Higher-order root-finding algorithms and their basins of attraction," *J. Math.*, vol. 2020, Nov. 2020, Art. no. 5070363.
[30] W. Nazeer, A. Naseem, S. M. Kang, and Y. C. Kwun, "Generalized Newton Raphson's method free from second derivative," *J. Non-linear Sci. Appl.*, vol. 9, pp. 2823–2831, Oct. 2016.
[31] M. A. Noor, W. A. Khan, and A. Hussain, "A new modified halley method without second derivatives for nonlinear equation," *Appl. Math. Comput.*, vol. 189, no. 2, pp. 1268–1273, Jun. 2007.
[32] M. A. Noor and F. A. Shah, "Variational iteration technique for solving nonlinear equations," *J. Appl. Math. Comput.*, vol. 31, nos. 1–2, pp. 247–254, Sep. 2009.
[33] M. A. Noor, K. I. Noor, and K. Aftab, "Some new iterative methods for solving nonlinear equations," *World Appl. Sci. J.*, vol. 20, no. 6, pp. 870–874, 2012.
[34] A. M. Ostrowski, *Solution of Equations and Systems of Equations*. New York, NY, USA: Academic, 1966.
[35] M. S. Rhee, Y. I. Kim, and B. Neta, "An optimal eighth-order class of three-step weighted Newton's methods and their dynamics behind the purely imaginary extraneous fixed points," *Int. J. Comput. Math.*, vol. 95, no. 11, pp. 2174–2211, Nov. 2018, doi: 10.1080/00207160.2017.1367387.
[36] M. Shacham, "An improved memory method for the solution of a nonlinear equation," *Chem. Eng. Sci.*, vol. 44, no. 7, pp. 1495–1501, 1989.
[37] M. Shacham and E. Kehat, "Converging interval methods for the iterative solution of nonlinear equations," *Chem. Eng. Sci.*, vol. 28, pp. 2187–2193, Oct. 1973.
[38] O. Said Solaiman, S. A. Abdul Karim, and I. Hashim, "Optimal fourth- and eighth-order of convergence derivative-free modifications of King's method," *J. King Saud Univ.-Sci.*, vol. 31, no. 4, pp. 1499–1504, Oct. 2019, doi: 10.1016/j.jksus.2018.12.001.
[39] F. Soleimani, F. Soleymani, and S. Shateyi, "Some iterative methods free from derivatives and their basins of attraction for nonlinear equations," *Discrete Dyn. Nature Soc.*, vol. 2013, May 2013, Art. no. 301718, doi: 10.1155/2013/301718.
[40] J. F. Traub, *Iterative Methods for the Solution of Equations*. New York, NY, USA: Chelsea, 1982.
[41] V. D. Waals and J. Diderik, "Over de Continuiteit van den Gas- en Vloeistoftoestand (on the continuity of the gas and liquid state," Ph.D. dissertation, Leiden Univ., Leiden, The Netherlands, 1873.

AMIR NASEEM received the M.S. degree in mathematics from Lahore Leads University, Lahore, Pakistan. He is currently pursuing the Ph.D. degree with the Department of Mathematics, University of Management and Technology, Lahore. He has published more than 25 research articles in different well known journals. His current research interests include numerical analysis, iterative methods, and polynomiography.

M. A. REHMAN received the Ph.D. degree in mathematics from Government College University Lahore, Pakistan. He is currently a Professor with the University of Management and Technology, Lahore. He has published over 50 research articles in different international journals. His research interests include modeling, graph theory, and mathematical analysis.

THABET ABDELJAWAD received the Ph.D. degree in mathematics from Middle East Technical University, in 2000. He is currently a Full Professor with Prince Sultan University. He has published more than 350 research articles in different well known journals. His research interests include fractional calculus, discrete fractional operators, metric spaces, and fixed point theory.

YU-MING CHU was born in Huzhou, Zhejiang, China, in June 1966. He received the B.S. degree from Hangzhou Normal University, Hangzhou, China, in 1988, and the M.S. and Ph.D. degrees from Hunan University, Changsha, China, in 1991 and 1994, respectively. He worked as an Assistant Professor, from 1994 to 1996, and an Associate Professor, from 1997 to 2002, with the Department of Mathematics, Hunan Normal University, Changsha. Since 2002, he has been a Professor and the Dean of the Department of Mathematics, Huzhou University, Huzhou. His current research interests include special functions, functional analysis, numerical analysis, operator theory, ordinary differential equations, partial differential equations, inequalities theory and applications, and robust filtering and control.