algorithms

MDPI

Article

Adaptive Parallel Methods for Polynomial Equations with Unknown Multiplicity

Mudassir Shams [1,2] **and Bruno Carpentieri** [2,*]

[1] Department of Mathematics, Faculty of Arts and Science, Balikesir University, 10145 Balıkesir, Turkey; mudassir.shams@balikesir.edu.tr
[2] Faculty of Engineering, Free University of Bozen-Bolzano, 39100 Bolzano, Italy
* Correspondence: bruno.carpentieri@unibz.it

Abstract

New two-step simultaneous iterative techniques are proposed for solving polynomial equations with multiple roots of unknown multiplicity. The developed schemes achieve a local convergence order of ten and address key limitations of existing solvers, namely their dependence on prior multiplicity information and their reduced efficiency when dealing with clustered or repeated roots. Root multiplicities are adaptively estimated within the iterative process, avoiding additional function evaluations beyond those required for parallel updates. The robustness and stability of the proposed methods are assessed using both random and distant initial guesses and validated on benchmark polynomials as well as nonlinear models from biomedical engineering. The numerical results show notable improvements in residual error, iteration count, CPU time, memory usage, and overall convergence rate compared with established classical techniques. These findings demonstrate that the proposed schemes provide reliable, high-order, and computationally efficient tools for solving challenging nonlinear problems in science and engineering.

Keywords: polynomial equations; multiple roots; unknown multiplicity; simultaneous iterative methods; adaptive algorithms; theoretical local convergence

1. Introduction

Polynomial equations arise in many areas of science and engineering [1–3]. In both natural and artificial systems, their solutions often correspond to steady states or equilibrium conditions. Multiple roots frequently occur in such equations, particularly in applications across physics, chemistry, and engineering, where they are commonly associated with system degeneracy or repeated equilibrium states [4]. In biomedical engineering, representative examples include equilibrium points in glucose–insulin regulation [5,6], tumor growth models [7], cardiac electrophysiology [8], and drug delivery systems [9], which can all be formulated as nonlinear systems exhibiting multiple roots. In these settings, simplifying assumptions made during root identification may affect the accuracy of model predictions or the reliability of numerical simulations. Consequently, the development and analysis of iterative methods for solving polynomial equations with multiple roots remain important in both theoretical investigations and practical applications.

The problem of finding solutions to nonlinear polynomial equations has attracted sustained attention for thousands of years. Ancient Babylonian mathematicians [10] developed geometric and arithmetic procedures to approximate the roots of quadratic and cubic equations, laying the foundation for some of the earliest systematic root-finding methods.

Academic Editors: Alicia Cordero and Juan Ramón Torregrosa Sánchez

Received: 17 November 2025
Revised: 21 December 2025
Accepted: 22 December 2025
Published: 24 December 2025

Greek scholars, including Euclid [11] and Diophantus [12], extended these ideas through geometric constructions to address quadratic and higher-order problems. During the Renaissance, Gerolamo Cardano [13], Rafael Bombelli [14], and François Viète [15] advanced algebraic techniques, deriving closed-form solutions for cubic and quartic polynomials. Later, Thomas Simpson [16], Joseph–Louis Lagrange [17], and William Horner [18] introduced methods for polynomial evaluation and approximation, further refining analytic approaches. In modern times, closed–form solutions remain available only for polynomials of degree up to four, whereas transcendental equations often require the use of special functions [19,20], such as the Lambert W function, elliptic functions, or the Mittag–Leffler function. Despite these advances, such methods have important limitations:

- They apply only to limited classes of equations, such as quadratic, cubic, or quartic polynomials.
- Symbolic manipulations can be cumbersome and computationally expensive for complex nonlinear systems.
- Many equations arising in biomedical engineering are nonlinear, transcendental, or fractional, making exact closed-form solutions impractical in most cases.

Numerical methods are widely used for root finding, particularly in nonlinear problems where analytical solutions are not available. Classical iterative schemes such as Newton's method [21], Halley's method [22], Chebyshev's method [23], and Schröder's method [24] have long been employed to approximate single roots with quadratic or higher convergence rates. Numerous extensions and modifications, including Ostrowski's method [25], Jarratt's method [26,27], and members of the King family [28], were later introduced to improve stability and convergence properties. Super–Halley iterations, Homeier's methods, and Traub's fourth-order schemes are examples of higher-order techniques that maintain computational efficiency while accelerating convergence. When multiplicity corrections are applied, these techniques can handle both simple and multiple roots. Modern numerical root-finding therefore relies heavily on iterative methods, which are essential in applications requiring high accuracy and efficiency in biology, physics, and engineering. In particular, when the root multiplicity is known, multiplicity-aware schemes and modified Newton methods achieve improved convergence.

Despite their widespread use, single-root numerical methods present several important limitations when applied to real-world nonlinear polynomial equations:

i. *Slow convergence for multiple roots.* Newton's method converges quadratically for simple roots but only linearly for multiple roots unless the multiplicity is explicitly taken into account.
ii. *Sensitivity to multiplicity.* Most classical methods require the root multiplicity to be known in advance; otherwise, convergence may deteriorate substantially.
iii. *Challenges in high–dimensional settings.* For multivariable systems or high-degree polynomials, sequential methods often need to be restarted from several initial guesses, resulting in reduced efficiency.
iv. *Redundancy.* In the presence of multiple roots, single-root methods usually compute them one at a time, often repeating similar computational steps.

These limitations motivate the development of simultaneous schemes that determine all roots at once, balance the computational effort, and improve convergence speed. Consider the general nonlinear equation

$$f(x) = 0. \tag{1}$$

If f is a polynomial with n distinct roots $\alpha_1, \alpha_2, \ldots, \alpha_n$ and corresponding multiplicities $m_1, m_2, \ldots, m_n \geq 1$, then f can be written in the form

$$f(x) = a\prod_{i=1}^{n}(x-\alpha_i)^{m_i}, \qquad a \neq 0, \tag{2}$$

with degree

$$\deg(f) = \sum_{i=1}^{n} m_i. \tag{3}$$

In the special case where all roots have the same multiplicity m, we obtain

$$f(x) = a\prod_{i=1}^{n}(x-\alpha_i)^{m}, \qquad \deg(f) = mn. \tag{4}$$

More generally, if f is analytic on a domain $\Omega \subset \mathbb{C}$ with zeros $\alpha_1, \ldots, \alpha_n$ of multiplicities $m_1, \ldots, m_n$, then there exists an analytic, nowhere-vanishing function $h(x)$ such that

$$f(x) = h(x)\prod_{i=1}^{n}(x-\alpha_i)^{m_i}, \qquad x \in \Omega.$$

For finding all roots of (1), the Weierstrass method [29] (WDK$^{[*]}$) with quadratic convergence is given by

$$x_i^{[k+1]} = x_i^{[k]} - \frac{f\left(x_i^{[k]}\right)}{\prod_{\substack{j=1 \\ j\neq i}}^{n}\left(x_i^{[k]} - x_j^{[k]}\right)}, \qquad i = 1, 2, \ldots, n. \tag{5}$$

For multiple roots, the method can be generalized as

$$x_i^{[k+1]} = x_i^{[k]} - \frac{f\left(x_i^{[k]}\right)}{\prod_{\substack{j=1 \\ j\neq i}}^{n}\left(x_i^{[k]} - x_j^{[k]}\right)^{m_j}}, \tag{6}$$

where m_j denotes the multiplicity of the root α_j. This formulation preserves the same order of convergence. For polynomials, the Weierstrass–Durand–Kerner methods [29–31] and related simultaneous approaches—such as the Ehrlich–Aberth method [32], the Börsch–Supan method [33], schemes by Cordero et al. [34], and Nourein's method [35]—update all approximations in each iteration and exploit vectorized computations, features that are particularly advantageous in high-performance and distributed environments, see e.g., [36–38].

Despite these advances, important challenges remain. A major limitation is multiplicity awareness: most existing techniques assume that all roots are simple, and their performance deteriorates markedly for roots with higher multiplicities. Another difficulty concerns the treatment of unknown multiplicities. Only a few approaches [39–42] attempt to detect or adaptively manage them, yet in many real-world problems—particularly those arising in biomedical and engineering models—the multiplicity of roots is rarely known in advance. Efficiency also remains a concern, since many two-step or higher-order simultaneous schemes involve significant computational overhead, which restricts their applicability to large-scale or high-dimensional systems. Thus, although iterative approaches have been extensively investigated in numerical mathematics, their systematic use in biomedical engineering problems involving multiple roots is still limited, leaving a gap between theoretical development and practical implementation.

In this context, there is a clear need for simultaneous algorithms that can efficiently compute all roots of polynomials or factorized analytic equations, independently of their multiplicity, while remaining applicable to real-world problems. Simultaneous schemes, which update all approximations simultaneously and are inherently suitable for parallel implementation, represent one of the most effective strategies for root finding. However, their classical forms generally assume simple roots and tend to perform poorly when multiple or unknown multiplicities are present.

This study develops novel two-step simultaneous methods that extend existing approaches in two directions:

- **Known multiplicity setting:** Compared with classical methods, the proposed iterative strategy improves the rate of convergence by explicitly using information on root multiplicity.
- **Unknown multiplicity setting:** The framework adaptively estimates the multiplicities of all roots in polynomial equations while maintaining high-order convergence through a dynamic modification of the iterative process.

The main contributions of this work are as follows:

- A new two-step simultaneous iterative method is proposed to compute all roots of polynomials or factorized analytic equations with both known and unknown multiplicities.
- A detailed theoretical convergence analysis is carried out for the single-root and simultaneous versions of the method, including local convergence theorems, which confirm the proposed algorithms' efficiency and consistency when compared with existing approaches.
- Numerical experiments on benchmark problems from biological, mechanical, and electrical engineering are used to evaluate the accuracy, stability, and performance of the iterative methods.
- Compared with earlier techniques, the proposed approach achieves higher convergence rates, improved robustness, and better memory efficiency in solving nonlinear models.

The proposed iterative schemes for determining all roots of (1) are scalable, reliable, and computationally efficient. They address the challenge of unknown multiplicities and broaden the range of nonlinear applications that can be solved, including physiological modeling and pharmacokinetics, where accurate root computation is essential for understanding system behavior.

- Section 2 describes the construction of the proposed two-step simultaneous schemes and presents the associated convergence analysis.
- Section 3 discusses the percentage computational efficiency of the simultaneous methods.
- Section 4 outlines the implementation details, numerical experiments on benchmark problems, and comparative evaluations against existing methods.
- Section 5 concludes the paper with a summary of the main findings and future research directions.

2. Construction and Analysis of the Computational Schemes

Single-root-finding methods play a fundamental role in numerical analysis, as they provide accurate approximations to polynomial equations when exact solutions are not available. Owing to their efficiency and relative simplicity, these methods are widely employed in engineering, physics, and biomedical applications. Over the years, a variety of strategies have been proposed to enhance convergence properties and improve accuracy in the solution of nonlinear problems. The development of iterative methods for solving nonlinear equations can be traced back to the seminal work of Isaac Newton in the seventeenth

century. The resulting Newton–Raphson method established the foundation of root-finding techniques exhibiting quadratic convergence. Since then, numerous higher-order methods have been introduced with the aim of increasing convergence efficiency. Chun et al. [43] proposed a third-order method that improves upon classical schemes, while Traub and Ostrowski [44] developed a well-known fourth-order method for single-root problems. Further fourth-order methods were later introduced by Jarratt [45], Naila et al. [46], as well as Khattri and Abbasbandy [47], each employing different correction strategies to accelerate convergence. In addition, Zein [48] presented a family of fifth-order iterative methods. Other higher-order schemes have also been reported in the literature; see, for example, [49,50] and the references therein.

In this work, we restrict our presentation to those iterative schemes that are directly required for the subsequent theoretical developments and numerical comparisons, in order to maintain clarity and conciseness.

Despite their broad applicability, single-root iterative methods exhibit well-known limitations when applied to problems involving multiple roots. In particular, the convergence rate typically deteriorates in the presence of root multiplicity, often decreasing from quadratic to linear. Moreover, many classical schemes rely on the evaluation of derivatives, which may be computationally expensive or analytically inconvenient for complex nonlinear problems.

2.1. Methods for Simple Roots with Known Multiplicity

To address the challenges posed by multiple roots, several modified iterative schemes have been proposed. Classical approaches, such as the multiplicity-corrected Newton and Schröder methods, explicitly incorporate the known multiplicity m of the root into the iteration formula. A modified Newton-type method introduced in [51] restores the optimal order of convergence and improves computational efficiency for problems in which the root multiplicity is known. Building on this idea, a number of higher-order iterative schemes for multiple roots have subsequently been developed, including multi-step fourth-order methods proposed by Chicharro et al. [52], Shengguo et al. [53], Zafar et al. [54], and Lee et al. [55]. Under suitable assumptions, these methods achieve local fourth-order convergence. Despite their improved convergence properties, a common limitation of these approaches is the requirement that the root multiplicity m be known a priori, which substantially restricts their applicability in practical settings.

Methods for Simple Roots of Unknown Multiplicity

Multiplicity-aware approaches rely on prior knowledge of the root multiplicity, which can represent a major limitation. In practical applications, such as biological or engineering models, this information is often unavailable, causing Newton-type schemes to lose quadratic convergence or even diverge. When the multiplicity is unknown, it can be approximated as follows [56]:

$$m \approx \frac{x^{[k+1]} - x^{[k]}}{f\left(x^{[k+1]}\right) - f\left(x^{[k]}\right)}, \tag{7}$$

where m represents the reciprocal of the divided difference of f between consecutive iterates. Several other studies have addressed the estimation of m; see, for example, [57–59] and the references therein. However, such techniques may increase computational cost and lead to numerical instability, particularly for closely spaced or high-multiplicity roots.

To overcome these limitations, several schemes have been developed that do not require explicit knowledge of m. For example, the modified Newton-type iteration [60],

$$x^{[k+1]} = x^{[k]} - \frac{f(x^{[k]})\, f'(x^{[k]})}{\left(f'(x^{[k]})\right)^2 - f(x^{[k]})\, f''(x^{[k]})}, \tag{8}$$

achieves quadratic convergence without requiring prior information on m. Alternatively, using the transformation

$$\phi(x) = \frac{f(x)}{f'(x)} = 0, \tag{9}$$

where $f(x)$ has a root of multiplicity m, Shah et al. [61] proposed the iterative scheme

$$x^{[k+1]} = x^{[k]} - \frac{\phi(x^{[k]})}{\phi'(x^{[k]})}, \qquad \phi'(x^{[k]}) \neq 0, \tag{10}$$

which preserves quadratic convergence without requiring prior multiplicity information. Such transformation-based methods maintain high efficiency and convergence order while avoiding the drawbacks associated with explicit multiplicity estimation. Extensions of this idea have also led to higher-order multi-step methods for unknown multiplicities [62–64].

Despite these advances, such algorithms often involve high computational costs, show sensitivity to numerical derivatives, and achieve only moderate convergence rates. To overcome these limitations, a family of generalized two-step schemes was proposed in [46] for cases with known multiplicity. As an example, the fourth-order extension for multiple roots is given as

$$\begin{cases} y^{[k]} = x^{[k]} - m\dfrac{f(x^{[k]})}{f'(x^{[k]})}, \\ x^{[k+1]} = y^{[k]} - m\left(\dfrac{\sqrt[m]{\frac{f(y^{[k]})}{f(x^{[k]})}}}{1-\beta\left(\sqrt[m]{\frac{f(y^{[k]})}{f(x^{[k]})}}\right)^2} + 2\left(\sqrt[m]{\frac{f(y^{[k]})}{f(x^{[k]})}}\right)^2\right)\dfrac{f(x^{[k]})}{f'(x^{[k]})}, \end{cases} \tag{11}$$

with $\beta \in \mathbb{R}$, $u^{[k]} = \sqrt[m]{\frac{f(y^{[k]})}{f(x^{[k]})}}$, and the method achieves fourth-order convergence. Its error relation is

$$e^{[k+1]} = \left(\frac{(9-2\beta+m)c_1^3 - 2mc_1c_2}{2m^3}\right)\left(e^{[k]}\right)^4 + O\Big(\left(e^{[k]}\right)^5\Big), \tag{12}$$

where

$$c_\gamma = \frac{m!}{(m+1)!}\frac{f^{(m+\gamma)}(\zeta)}{f^{(m)}(\zeta)}, \qquad \gamma \geq 1,$$

and $e^{[k]} = x^{[k]} - \zeta$.

While effective when m is known, this approach is restrictive in practice, since the multiplicity is rarely available in advance. To overcome this limitation, we introduce a multiplicity-reducing transformation. Writing

$$f(x) = (x-\zeta)^m g(x), \qquad g(\zeta) \neq 0,$$

we define

$$\wp(x) = \frac{f(x)}{f'(x)}, \tag{13}$$

for which ζ is a *simple* root. Applying the two-step framework to $\wp$ instead of f yields the new scheme

$$\begin{cases} y^{[k]} = x^{[k]} - \dfrac{\wp(x^{[k]})}{\wp'(x^{[k]})}, \\ x^{[k+1]} = y^{[k]} - \left(\dfrac{\frac{\wp(y^{[k]})}{\wp(x^{[k]})}}{1 - \beta\left(\frac{\wp(y^{[k]})}{\wp(x^{[k]})}\right)^2} + 2\left(\frac{\wp(y^{[k]})}{\wp(x^{[k]})}\right)^2 \right) \dfrac{\wp(x^{[k]})}{\wp'(x^{[k]})}, \end{cases} \tag{14}$$

with $\beta \in \mathbb{R}$.

Theorem 1. *Let $\zeta \in \mathbb{C}$ be a root of f of multiplicity m. If $x^{[0]}$ is sufficiently close to ζ, then the iteration scheme (14) converges with order four and satisfies*

$$e^{[k+1]} = \left(\frac{((-5+\beta+3m^2)\, c_1^3) - 9m^2 c_1 c_2 + 9m^2 c_3}{m^2} \right) (e^{[k]})^4 + \mathcal{O}\Big((e^{[k]})^5\Big), \tag{15}$$

where

$$c_\gamma = \frac{m!}{(m+j)!} \frac{f^{(m+\gamma)}(\zeta)}{f^{(m)}(\zeta)}, \qquad \gamma \geq 1, \quad e^{[k]} = x^{[k]} - \zeta.$$

Proof. Assume

$$f(x) = (x-\zeta)^m g(x), \tag{16}$$

where ζ is a multiple root of multiplicity m and $g(x)$ is smooth with $g(\zeta) \neq 0$. Thus,

$$\wp(x) = \frac{f(x)}{f'(x)} = \frac{(x-\zeta)^m g(x)}{m(x-\zeta)^{m-1} g(x) + (x-\zeta)^m g'(x)}. \tag{17}$$

Equation (17) shows that the multiple root of (2) can be transformed into a simple root of the nonlinaer equation $f(x) = 0$.

Expanding $g(x)$ around ζ,

$$g(x^{[k]}) = g(\zeta)\Big[1 + c_1 e^{[k]} + c_2 (e^{[k]})^2 + c_3 (e^{[k]})^3 + c_4 (e^{[k]})^4 + \mathcal{O}\big((e^{[k]})^5\big)\Big], \tag{18}$$

$$g'^{[k]}) = g(\zeta)\Big[c_1 + 2c_2 e^{[k]} + 3c_3 (e^{[k]})^2 + 4c_4 (e^{[k]})^3 + \mathcal{O}\big((e^{[k]})^4\big)\Big]. \tag{19}$$

Using (18) and (19) in (17), we obtain

$$\begin{aligned} \wp(x^{[k]}) &= \frac{e^{[k]} g(x^{[k]})}{m g(x^{[k]}) + e^{[k]} g'^{[k]})} \\ &= \frac{1}{m} e^{[k]} - \frac{c_1}{m^2} (e^{[k]})^2 + \frac{c_1 + m c_1^2 - 2m c_2}{m^3} (e^{[k]})^3 \\ &\quad + \frac{-(1+m^2) c_1^3 + m(4+3m) c_1 c_2 - 3m^2 c_3}{m^4} (e^{[k]})^4 + \mathcal{O}\big((e^{[k]})^5\big). \end{aligned} \tag{20}$$

Differentiating (20),

$$\begin{aligned} \wp'^{[k]}) &= \frac{1}{m} - \frac{2c_1}{m^2} e^{[k]} + \frac{3(c_1 + m c_1^2 - 2m c_2)}{m^3} (e^{[k]})^2 \\ &\quad + \frac{-4(1+m^2) c_1^3 + 4m(4+3m) c_1 c_2 - 12 m^2 c_3}{m^4} (e^{[k]})^3 + \mathcal{O}\big((e^{[k]})^4\big). \end{aligned} \tag{21}$$

Hence,

$$\frac{\wp(x^{[k]})}{\wp'^{[k]})} = e^{[k]} + \frac{c_1}{m}(e^{[k]})^2 - \frac{2(c_1^2 - 2mc_2)}{m^2}(e^{[k]})^3 + \frac{(-1+3m)c_1^3 + (-2+9m)c_1c_2 - 9mc_3}{m^3}(e^{[k]})^4 + \cdots \tag{22}$$

so that the first substep error becomes

$$y^{[k]} - \zeta = -\frac{c_1}{m}(e^{[k]})^2 + \frac{2(c_1^2 - 2mc_2)}{m^2}(e^{[k]})^3 - \frac{(-1+3m)c_1^3 + (-2+9m)c_1c_2 - 9mc_3}{m^3}(e^{[k]})^4 + \cdots. \tag{23}$$

Expanding $g(y^{[k]})$ around ζ gives

$$\wp(y^{[k]}) = 1 + (y^{[k]} - \zeta)c_1 + (y^{[k]} - \zeta)^2c_2 + (y^{[k]} - \zeta)^3c_3 + \cdots. \tag{24}$$

Simplification yields

$$g(y^{[k]}) = 1 - \frac{c_1^2}{m}(e^{[k]})^2 + \frac{2c_1(c_1^2 - 2c_2)}{m^2}(e^{[k]})^3 + \left(\frac{c_1^2c_2}{m^2} + \frac{(-1+3m)c_1^3 + (-2+9m)c_1c_2 - 9mc_3}{m^3}\right)(e^{[k]})^4 + \cdots. \tag{25}$$

Thus,

$$\begin{aligned}\wp(y^{[k]}) &= \frac{e^{[k]}g(y^{[k]})}{mg(y^{[k]}) + e^{[k]}g'^{[k]})} \\ &= -\frac{c_1}{m}(e^{[k]})^2 + \frac{2(c_1^2 - 2c_2)}{m^2}(e^{[k]})^3 \\ &\quad + \frac{(-1+3m)c_1^3 + (-2+9m)c_1c_2 - 9mc_3}{m^3}(e^{[k]})^4 + \cdots.\end{aligned} \tag{26}$$

Dividing $\wp(y^{[k]})$ by $\wp(x^{[k]})$ gives

$$\frac{\wp(y^{[k]})}{\wp(x^{[k]})} = -\frac{c_1}{m}e^{[k]} + \frac{((-1+2m)c_1^2 - 4mc_2)}{m^2}(e^{[k]})^2 + \left(\frac{c_1^2c_2}{m^2} + \frac{(4-3m)c_1^3 + (-8+9m)c_1c_2 - 9mc_3}{m^2}\right)(e^{[k]})^3 + \mathcal{O}((e^{[k]})^3). \tag{27}$$

Furthermore,

$$\begin{aligned}&\frac{\frac{\wp(y^{[k]})}{\wp(x^{[k]})}}{1 - \beta\left(\frac{\wp(y^{[k]})}{\wp(x^{[k]})}\right)^2} + 2\left(\frac{\wp(y^{[k]})}{\wp(x^{[k]})}\right)^2 \\ &= -\frac{c_1}{m}e^{[k]} + \frac{((-1+2m)c_1^2 - 4mc_2)}{m^2}(e^{[k]})^2 \\ &\quad + \frac{-((-4-\beta-4m-3m^2)c_1^3) + m(-8+9m)c_1c_2 - 9m^2c_3}{m^3}(e^{[k]})^3 + \cdots.\end{aligned} \tag{28}$$

Therefore,

$$Q_1^{[*]}\frac{\wp(x^{[k]})}{\wp'^{[k]})} = -\frac{c_1}{m}(e^{[k]})^2 + \frac{2(c_1^2-2c_2)}{m}(e^{[k]})^3 + \left(\frac{(-(-5+\beta+3m^2)c_1^3)+9m^2c_1c_2-9m^2c_3}{m^2}\right)(e^{[k]})^4 + \mathcal{O}\Big((e^{[k]})^5\Big), \tag{29}$$

where

$$Q_1^{[*]} = \left(\frac{\frac{\wp(y^{[k]})}{\wp(x^{[k]})}+2\left(\frac{\wp(y^{[k]})}{\wp(x^{[k]})}\right)^2}{1-\beta\left(\frac{\wp(y^{[k]})}{\wp(x^{[k]})}\right)^2}\right).$$

Thus, the error of the final step becomes

$$z^{[k]}-\zeta = \left(\frac{((-5+\beta+3m^2)c_1^3)-9m^2c_1c_2+9m^2c_3}{m^2}\right)(e^{[k]})^4 + \mathcal{O}\Big((e^{[k]})^5\Big), \tag{30}$$

or equivalently,

$$e^{[k+1]} = \left(\frac{((-5+\beta+3m^2)c_1^3)-9m^2c_1c_2+9m^2c_3}{m^2}\right)(e^{[k]})^4 + \mathcal{O}\Big((e^{[k]})^5\Big). \tag{31}$$

Hence, the theorem is proven. □

These generalizations provide a systematic framework for handling unknown multiplicities while improving the order of convergence, thus laying the foundation for the development of efficient simultaneous schemes. A further challenge, however, lies in computing *all* roots simultaneously, rather than one at a time, in a manner that remains effective for both known and unknown multiplicities.

2.2. Iterative Schemes for Finding All Roots with and Without Multiplicity

To address this more general problem, we propose two types of two-step iterative techniques. In the first scheme, the root multiplicities are assumed to be known in advance. This approach incorporates multiplicity information directly into the iteration, enabling the simultaneous computation of both simple and multiple roots, thus improving upon the classical Weierstrass method. The second approach removes this restriction by approximating the multiplicities through a dynamically updated iterative procedure. Even while achieving high-order convergence, the adaptive strategy maintains computational efficiency when multiplicities are unknown. Applying the iteration formula to the denominator ensures rapid and reliable convergence to all roots of the polynomial equation.

1. **Simultaneous scheme with known multiplicities.** This variant of the classical Weierstrass approach computes all roots simultaneously by incorporating the known multiplicities m_j of the roots α_j.
2. **Simultaneous scheme with unknown multiplicities.** In this case, the scheme adaptively estimates the multiplicities during iteration. By dynamically updating the denominator at each step according to the current approximations, the method computes all roots simultaneously without requiring prior knowledge of m_j.

2.2.1. Two-Step Simultaneous Scheme with Known Multiplicity

This scheme extends the Weierstrass [29] iteration to account for multiplicities and simultaneously approximate all roots:

$$x_i^{[k+1]} = x_i^{[k]} - \frac{f(x_i^{[k]})}{\prod\limits_{\substack{j=1 \\ j\neq i}}^{n}\left(x_i^{[k]} - x_j^{[k]}\right)}, \qquad i = 1, 2, \ldots, n, \tag{32}$$

where the Weierstrass correctionis defined as

$$\Xi\left(x_i^{[k]}\right) = \frac{f(x_i^{[k]})}{\prod\limits_{\substack{j=1 \\ j\neq i}}^{n}\left(x_i^{[k]} - x_j^{[k]}\right)}. \tag{33}$$

For multiple roots, the correction is modified to

$$\Xi\left(x_i^{[k]}\right) = \frac{f(x_i^{[k]})}{\prod\limits_{\substack{j=1 \\ j\neq i}}^{n}\left(x_i^{[k]} - x_j^{[k]}\right)^{m_j}}, \tag{34}$$

where m_j is the multiplicity of the root $x_j^{[k]}$. Replacing $x_j^{[k]}$ with $z_j^{[k]}$, we construct the proposed two-step simultaneous scheme (WKM$^{[*]}$) for simultaneously finding all distinct and multiple roots:

$$\begin{cases} y_i^{[k]} = x_i^{[k]} - \dfrac{f\left(x_i^{[k]}\right)}{\prod_{\substack{j=1 \\ j\neq i}}^{n}\left(x_i^{[k]} - z_j^{[k]}\right)}, \\ x_i^{[k+1]} = y_i^{[k]} - \prod\limits_{\substack{j=1 \\ j\neq i}}^{n}\left(\dfrac{y_i^{[k]} - y_j^{[k]}}{x_i^{[k]} - x_j^{[k]}}\right)\left[2 - \prod\limits_{\substack{j=1 \\ j\neq i}}^{n}\left(\dfrac{y_i^{[k]} - y_j^{[k]}}{x_i^{[k]} - x_j^{[k]}}\right) + \dfrac{5}{4}\left(1 - \prod\limits_{\substack{j=1 \\ j\neq i}}^{n}\left(\dfrac{y_i^{[k]} - y_j^{[k]}}{x_i^{[k]} - x_j^{[k]}}\right)\right)^2\right. \\ \qquad \left. - \dfrac{1}{6}\left(1 - \prod_{\substack{j=1 \\ j\neq i}}^{n}\left(\dfrac{y_i^{[k]} - y_j^{[k]}}{x_i^{[k]} - x_j^{[k]}}\right)\right)^3\right] \dfrac{f\left(y_i^{[k]}\right)}{\prod_{\substack{j=1 \\ j\neq i}}^{n}\left(y_i^{[k]} - y_j^{[k]}\right)}, \end{cases} \tag{35}$$

where the auxiliary points are given by

$$z_j^{[k]} = y_j^{[k]} - m_j \left(\frac{\sqrt[m_j]{\dfrac{f\left(y_j^{[k]}\right)}{f\left(x_j^{[k]}\right)}}}{1 - \beta\left(\sqrt[m_j]{\dfrac{f\left(y_j^{[k]}\right)}{f\left(x_j^{[k]}\right)}}\right)^2} + 2\left(\sqrt[m_j]{\frac{f\left(y_j^{[k]}\right)}{f\left(x_j^{[k]}\right)}}\right)^2 \right) \frac{f\left(x_j^{[k]}\right)}{f'\left(x_j^{[k]}\right)},$$

and

$$y_j^{[k]} = x_j^{[k]} - m_j \frac{f(x_j^{[k]})}{f'(x_j^{[k]})}.$$

2.2.2. Two-Step Simultaneous Scheme with Unknown Multiplicity

For the case of unknown root multiplicity, we replace $x_j^{[k]}$ by $Z_j^{[k]}$ and proceed analogously; this variant is hereafter referred to as WKM$^{[*]}$:

$$\begin{cases} y_i^{[k]} = x_i^{[k]} - \dfrac{f\left(x_i^{[k]}\right)}{\prod_{\substack{j=1 \\ j\neq i}}^{n}\left(x_i^{[k]} - Z_j^{[k]}\right)}, \\ x_i^{[k+1]} = y_i^{[k]} - \prod\limits_{\substack{j=1 \\ j\neq i}}^{n}\left(\dfrac{y_i^{[k]} - y_j^{[k]}}{x_i^{[k]} - x_j^{[k]}}\right)\left[2 - \prod\limits_{\substack{j=1 \\ j\neq i}}^{n}\left(\dfrac{y_i^{[k]} - y_j^{[k]}}{x_i^{[k]} - x_j^{[k]}}\right) + \dfrac{5}{4}\left(1 - \prod\limits_{\substack{j=1 \\ j\neq i}}^{n}\left(\dfrac{y_i^{[k]} - y_j^{[k]}}{x_i^{[k]} - x_j^{[k]}}\right)\right)^2\right. \\ \qquad \left. - \frac{1}{6}\left(1 - \prod_{\substack{j=1 \\ j\neq i}}^{n}\left(\dfrac{y_i^{[k]} - y_j^{[k]}}{x_i^{[k]} - x_j^{[k]}}\right)\right)^3\right] \dfrac{f\left(y_i^{[k]}\right)}{\prod_{\substack{j=1 \\ j\neq i}}^{n}\left(y_i^{[k]} - y_j^{[k]}\right)}, \end{cases} \tag{36}$$

where

$$Z_j^{[k]} = y_j^{[k]} - \left(\frac{\frac{\wp(y_j^{[k]})}{\wp(x_j^{[k]})}}{1 - \beta\left(\frac{\wp(y_j^{[k]})}{\wp(x_j^{[k]})}\right)^2} + 2\left(\frac{\wp(y_j^{[k]})}{\wp(x_j^{[k]})}\right)^2\right)\frac{\wp(x_j^{[k]})}{\wp'(x_j^{[k]})},$$

and

$$y_j^{[k]} = x_j^{[k]} - \frac{\wp(x_j^{[k]})}{\wp'(x_j^{[k]})}.$$

2.3. Analysis of Convergence

We present the local convergence analysis of the newly developed schemes, WKM$^{[*]}$ and WUM$^{[*]}$, for finding all multiple roots of polynomial equations with and without known multiplicities, as stated in the following theorems.

Theorem 2. *Let $\boldsymbol{\zeta} = (\zeta_1, \zeta_2, \ldots, \zeta_n)$ denote the distinct roots of (2). Assume that the initial approximations $x_1^{[0]}, \ldots, x_n^{[0]}$ are sufficiently close to the corresponding roots and are pairwise distinct. Define the iteration scheme (35) for the case of multiple roots. Then, the iteration (35) converges to $\boldsymbol{\zeta}$ with order* 10*, that is, the local error satisfies*

$$e_i^{[k+1]} = O\big((e^{[k]})^{10}\big),$$

provided that the initial vector is sufficiently close to $\boldsymbol{\zeta}$.

Proof. Let the local errors be defined as

$$e_{i,x}^{[k]} = x_i^{[k]} - \zeta_i, \qquad e_{i,y}^{[k]} = y_i^{[k]} - \zeta_i, \qquad e_i^{[k+1]} = x_i^{[k+1]} - \zeta_i. \tag{37}$$

First substep:

$$y_i^{[k]} - \zeta_i = x_i^{[k]} - \zeta_i - \frac{f(x_i^{[k]})}{\prod\limits_{\substack{j=1 \\ j\neq i}}^{n}(x_i^{[k]} - z_j^{[k]})}, \tag{38}$$

and from (37), the corresponding error satisfies

$$e_{i,y}^{[k]} = e_{i,x}^{[k]} - \frac{f(x_i^{[k]})}{\prod_{j\neq i}(x_i^{[k]} - z_j^{[k]})}, \tag{39}$$

where

$$\begin{cases} y_j^{[k]} = x_j^{[k]} - m_j \dfrac{f(x_j^{[k]})}{f'(x_j^{[k]})}, \\ z_j^{[k]} = y_j^{[k]} - m_j \left(\dfrac{\sqrt[m_j]{\frac{f(y_j^{[k]})}{f(x_j^{[k]})}}}{1 - \beta \left(\sqrt[m_j]{\frac{f(y_j^{[k]})}{f(x_j^{[k]})}} \right)^2} + 2 \left(\sqrt[m_j]{\frac{f(y_j^{[k]})}{f(x_j^{[k]})}} \right)^2 \right) \dfrac{f(x_j^{[k]})}{f'(x_j^{[k]})}, \end{cases} \tag{40}$$

and $\beta \in \mathbb{R}$. Using the factorization (2), we can write

$$f(x_i^{[k]}) = \prod_{j=1}^{n}(x_i^{[k]} - \zeta_j) = (e_{i,x}^{[k]}) \prod_{\substack{j=1 \\ j\neq i}}^{n}(x_i^{[k]} - \zeta_j).$$

Substituting this into the first line of (35) yields

$$e_{i,y}^{[k]} = e_{i,x}^{[k]} - e_{i,x}^{[k]} \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{(x_i^{[k]} - \zeta_j)}{(x_i^{[k]} - z_j^{[k]})}. \tag{41}$$

For $j \neq i$, we have $z_j^{[k]} = \zeta_j + O(|e_j^{[k]}|^4)$ under the stated local hypotheses (this follows from the local expansion of the auxiliary corrections defining the error in $z_j^{[k]}$, as given in (12)). Hence,

$$\frac{x_i^{[k]} - \zeta_j}{x_i^{[k]} - z_j^{[k]}} = 1 + \frac{z_j^{[k]} - \zeta_j}{x_i^{[k]} - z_j^{[k]}} = 1 + O(|e_j^{[k]}|^4),$$

and therefore,

$$\prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{(x_i^{[k]} - \zeta_j)}{(x_i^{[k]} - z_j^{[k]})} = (1 + O(|e_j^{[k]}|^4)^{n-1} = (1 + (n-1)O(|e_j^{[k]}|^4) = 1 + O(|e^{[k]}|^4), \tag{42}$$

where $|e^{[k]}|$ denotes a representative magnitude of the component errors (we assume all component errors are of comparable size). Combining (41) and (42) gives

$$e_{i,y}^{[k]} = e_{i,x}^{[k]} - (e_{i,x}^{[k]})\,(1 + O(|e^{[k]}|^4)). \tag{43}$$

and thus

$$e_{i,y}^{[k]} = O(|e^{[k]}|^5). \tag{44}$$

Second substep:

Using the factorization at $y_i^{[k]}$, we have

$$f(y_i^{[k]}) = e_{i,y}^{[k]} \prod_{\substack{j=1 \\ j\neq i}}^{n} (y_i^{[k]} - \zeta_j),$$

and substituting this into the second line of (35) leads to

$$\begin{aligned} e_i^{[k+1]} = e_{i,y}^{[k]} - (e_{i,y}^{[k]}) \left[\prod_{\substack{j=1 \\ j\neq i}}^{n} \left(\frac{y_i^{[k]} - y_j^{[k]}}{x_i^{[k]} - x_j^{[k]}} \right) \right] \left(2 - \prod_{\substack{j=1 \\ j\neq i}}^{n} \left(\frac{y_i^{[k]} - y_j^{[k]}}{x_i^{[k]} - x_j^{[k]}} \right) \right. \\ \left. + \frac{5}{4}(1 - \Pi^{[k]})^2 - \frac{1}{6}(1 - \Pi^{[k]})^3 \right) \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{(y_i^{[k]} - \zeta_j)}{(y_i^{[k]} - y_j^{[k]})}, \end{aligned} \tag{45}$$

where, for brevity, we set

$$\Pi^{[k]} := \prod_{\substack{j=1 \\ j\neq i}}^{n} \left(\frac{y_i^{[k]} - y_j^{[k]}}{x_i^{[k]} - x_j^{[k]}} \right).$$

Under the local smallness assumptions, one has $\Pi^{[k]} = 1 + O(|e^{[k]}|)$. Moreover, using the same argument as in (42),

$$\prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{(y_i^{[k]} - \zeta_j)}{(y_i^{[k]} - y_j^{[k]})} = 1 + O(|e_{i,y}^{[k]}|).$$

Combining these bounds with (45) and using (44), we obtain

$$e_i^{[k+1]} = e_{i,y}^{[k]} - (e_{i,y}^{[k]})\,(1 + O(|e_{i,y}^{[k]}|)) = O(|e_{i,y}^{[k]}|^2). \tag{46}$$

Because $e_{i,y}^{[k]} = O(|e^{[k]}|^5)$ by (44), the dominant term in (46) is $e_{i,y}^{[k]}$, while the corrective term is of order $(e_{i,y}^{[k]}) = O(|e^{[k]}|^5)$. Combining these estimates yields

$$e_i^{[k+1]} = O(|e^{[k]}|^{10}).$$

Remark 1. *The previous estimate holds component-wise for $i = 1, \ldots, n$; hence, the full vector error satisfies*

$$\|e^{[k+1]}\| = O(\|e^{[k]}\|^{10}).$$

Therefore, the scheme (35) converges to $\boldsymbol{\zeta}$ with local order 10, provided that it is initialized sufficiently close to the exact roots. □

This result demonstrates that the proposed two-step simultaneous method for the case of known multiplicities achieves a substantial acceleration over classical Weierstrass-type iterations, which are typically only quadratic. The next theorem shows that the same tenth-order convergence can also be obtained in the more challenging setting where the multiplicities are not assumed to be known in advance.

Theorem 3. *Let $\boldsymbol{\zeta} = (\zeta_1, \ldots, \zeta_n)$ be the solutions of the polynomial equations (2). For sufficiently close and distinct initial approximations $x_1^{[0]}, \ldots, x_n^{[0]}$, the method (36) converges with order* 10.

Proof. Define the errors

$$e_{i,x}^{[k]} = x_i^{[k]} - \zeta_i, \quad e_{i,y}^{[k]} = y_i^{[k]} - \zeta_i, \quad e_i^{[k+1]} = x_i^{[k+1]} - \zeta_i. \tag{47}$$

For the second substep, we have

$$y_i^{[k]} - \zeta_i = x_i^{[k]} - \zeta_i - \frac{f(x_i^{[k]})}{\prod\limits_{\substack{j=1 \\ j\neq i}}^{n}(x_i^{[k]} - Z_j^{[k]})}. \tag{48}$$

From (48), the error iteration becomes

$$e_{i,y}^{[k]} = e_{i,x}^{[k]} - \frac{f(x_i^{[k]})}{\prod\limits_{\substack{j=1 \\ j\neq i}}^{n}(x_i^{[k]} - Z_j^{[k]})}. \tag{49}$$

where

$$\begin{cases} y_j^{[k]} = x_j^{[k]} - \dfrac{\wp(x_j^{[k]})}{\wp'(x_j^{[k]})}, \\ Z_j^{[k]} = y_j^{[k]} - \left(\dfrac{\frac{\wp(y_j^{[k]})}{\wp(x_j^{[k]})}}{1 - \beta\left(\frac{\wp(y_j^{[k]})}{\wp(x_j^{[k]})}\right)^2} + 2\left(\frac{\wp(y_j^{[k]})}{\wp(x_j^{[k]})}\right)^2 \right) \dfrac{\wp(x_j^{[k]})}{\wp'(x_j^{[k]})}, \end{cases} \tag{50}$$

with $\beta \in \mathbb{R}$.

Using the factorization of $f(x_i^{[k]})$,

$$f(x_i^{[k]}) = \prod_{j=1}^{n}(x_i^{[k]} - \zeta_j) = e_{i,x}^{[k]} \prod_{\substack{j=1 \\ j\neq i}}^{n}\left(x_i^{[k]} - \zeta_j\right). \tag{51}$$

Substituting into (49) gives

$$e_{i,y}^{[k]} = e_{i,x}^{[k]} - \left(e_{i,x}^{[k]}\right) \prod_{\substack{j=1 \\ j\neq i}}^{n}\left(\frac{x_i^{[k]} - \zeta_j}{x_i^{[k]} - Z_j^{[k]}}\right). \tag{52}$$

Now, for $j \neq i$, expand the fraction:

$$\frac{x_i^{[k]} - \zeta_j}{x_i^{[k]} - Z_j^{[k]}} = 1 + \frac{Z_j^{[k]} - \zeta_j}{x_i^{[k]} - Z_j^{[k]}} = 1 + O(|e_j^{[k]}|^4), \tag{53}$$

since $Z_j^{[k]} - \zeta_j = O(|e_j^{[k]}|^4)$ using (15).

Therefore,

$$\prod_{\substack{j=1\\ j\neq i}}^{n}\frac{x_i^{[k]}-\zeta_j}{x_i^{[k]}-Z_j^{[k]}}=\prod_{\substack{j=1\\ j\neq i}}^{n}\Big(1+O(|e_j^{[k]}|^4)\Big)=1+O(|e_{j,x}^{[k]}|^4). \tag{54}$$

Thus,

$$e_{i,y}^{[k]}=e_{i,x}^{[k]}-\big(e_{i,x}^{[k]}\big)\big(1+O(|e^{[k]}|)\big), \tag{55}$$

which reduces to

$$e_{i,y}^{[k]}=O\Big(|e^{[k]}|^5\Big). \tag{56}$$

For the second substep, we have

$$\begin{aligned}x_i^{[k+1]}-\zeta_i=y_i^{[k]}-\zeta_i-\prod_{\substack{j=1\\ j\neq i}}^{n}\left(\frac{y_i^{[k]}-y_j^{[k]}}{x_i^{[k]}-x_j^{[k]}}\right)\left(2-\prod_{\substack{j=1\\ j\neq i}}^{n}\left(\frac{y_i^{[k]}-y_j^{[k]}}{x_i^{[k]}-x_j^{[k]}}\right)\right.\\ \left.+\tfrac{5}{4}\Big(1-\prod_{\substack{j=1\\ j\neq i}}^{n}\left(\frac{y_i^{[k]}-y_j^{[k]}}{x_i^{[k]}-x_j^{[k]}}\right)\Big)^2-\tfrac{1}{6}\Big(1-\prod_{\substack{j=1\\ j\neq i}}^{n}\left(\frac{y_i^{[k]}-y_j^{[k]}}{x_i^{[k]}-x_j^{[k]}}\right)\Big)^3\right)\frac{f(y_i^{[k]})}{\prod_{\substack{j=1\\ j\neq i}}^{n}(y_i^{[k]}-y_j^{[k]})}.\end{aligned} \tag{57}$$

From (57), the error iteration becomes

$$\begin{aligned}e_i^{[k+1]}=e_{i,y}^{[k]}-\prod_{\substack{j=1\\ j\neq i}}^{n}\left(\frac{y_i^{[k]}-y_j^{[k]}}{x_i^{[k]}-x_j^{[k]}}\right)\left(2-\prod_{\substack{j=1\\ j\neq i}}^{n}\left(\frac{y_i^{[k]}-y_j^{[k]}}{x_i^{[k]}-x_j^{[k]}}\right)\right.\\ \left.+\tfrac{5}{4}\Big(1-\prod_{\substack{j=1\\ j\neq i}}^{n}\left(\frac{y_i^{[k]}-y_j^{[k]}}{x_i^{[k]}-x_j^{[k]}}\right)\Big)^2-\tfrac{1}{6}\Big(1-\prod_{\substack{j=1\\ j\neq i}}^{n}\left(\frac{y_i^{[k]}-y_j^{[k]}}{x_i^{[k]}-x_j^{[k]}}\right)\Big)^3\right)\frac{f(y_i^{[k]})}{\prod_{\substack{j=1\\ j\neq i}}^{n}(y_i^{[k]}-y_j^{[k]})}.\end{aligned} \tag{58}$$

Using the factorization

$$f(y_i^{[k]})=\prod_{j=1}^{n}(y_i^{[k]}-\zeta_j)=e_{i,y}^{[k]}\prod_{\substack{j=1\\ j\neq i}}^{n}(y_i^{[k]}-\zeta_j), \tag{59}$$

we obtain

$$\begin{aligned}e_i^{[k+1]}=e_{i,y}^{[k]}-\prod_{\substack{j=1\\ j\neq i}}^{n}\left(\frac{y_i^{[k]}-y_j^{[k]}}{x_i^{[k]}-x_j^{[k]}}\right)\left(2-\prod_{\substack{j=1\\ j\neq i}}^{n}\left(\frac{y_i^{[k]}-y_j^{[k]}}{x_i^{[k]}-x_j^{[k]}}\right)\right.\\ \left.+\tfrac{5}{4}\Big(1-\prod_{\substack{j=1\\ j\neq i}}^{n}\left(\frac{y_i^{[k]}-y_j^{[k]}}{x_i^{[k]}-x_j^{[k]}}\right)\Big)^2-\tfrac{1}{6}\Big(1-\prod_{\substack{j=1\\ j\neq i}}^{n}\left(\frac{y_i^{[k]}-y_j^{[k]}}{x_i^{[k]}-x_j^{[k]}}\right)\Big)^3\right)\times\\ e_{i,y}^{[k]}\prod_{\substack{j=1\\ j\neq i}}^{n}\left(\frac{y_i^{[k]}-\zeta_j}{y_i^{[k]}-y_j^{[k]}}\right),\end{aligned} \tag{60}$$

Expanding

$$\frac{y_i^{[k]} - \zeta_j}{y_i^{[k]} - y_j^{[k]}} = 1 + \frac{y_j^{[k]} - \zeta_j}{y_i^{[k]} - y_j^{[k]}} = 1 + O(|e_{j,y}^{[k]}|), \tag{61}$$

gives

$$\prod_{\substack{j=1 \\ j \neq i}}^{n} \left(\frac{y_i^{[k]} - \zeta_j}{y_i^{[k]} - y_j^{[k]}} \right) = 1 + O(|e_{j,y}^{[k]}|). \tag{62}$$

Thus,

$$e_i^{[k+1]} = e_{i,y}^{[k]} \Big(1 - \big(1 + O(|e_{j,y}^{[k]}|) \big) \Big). \tag{63}$$

Hence,

$$e_{i,y}^{[k]} = O(|e^{[k]}|^5), \tag{64}$$

and finally

$$e_i^{[k+1]} = O\Big(|e^{[k]}|^{10} \Big), \tag{65}$$

which shows that the convergence order of (36) is 10. □

3. Percentage Computational Efficiency Analysis

The percentage computational efficiency of an iterative root-finding method quantifies the trade-off between its convergence rate and computational cost, thereby providing an integrated measure of algorithmic performance. For a simultaneous root-finding scheme (M), the efficiency index is defined as

$$\Phi(M) = \frac{\log r}{D_*}, \tag{66}$$

where r denotes the order of convergence of the method and D_* represents the total computational cost per iteration. The latter is given by

$$D_* = \vartheta_{as} AS_n + \vartheta_m M_n + \vartheta_d D_n, \tag{67}$$

where AS_n, M_n, and D_n denote the numbers of additions/subtractions, multiplications, and divisions, respectively, and the coefficients ϑ_{as}, ϑ_m, and ϑ_d represent their associated computational weights. Following standard operation–cost models commonly adopted in the literature [65], the weights are normalized with respect to addition and chosen as $\vartheta_{as} = 1$, $\vartheta_m = 1.5$, and $\vartheta_d = 5$, reflecting typical relative costs on a general-purpose processor. This normalization provides a consistent basis for the efficiency comparisons reported in Table 1. Because this metric accounts for the combined effects of convergence behavior and computational workload, it is particularly suitable for comparing simultaneous root-finding techniques.

To illustrate the comparative performance, we consider three well-known parallel root-finding algorithms from the literature:

- The method of Perković et al. [66];
- The method of Wang and Wu [67];
- The method of Farmer and Loizou [68].

These iterative methods were chosen because they can simultaneously locate all roots of polynomial equations, offer explicit iterative formulations, and exhibit good convergence properties in a variety of computational contexts. The corresponding iterative update formulas are presented below for comparison.

- The Perković–Miodrag method (MPM$^{[*]}$):

$$y_i^{[k]} = x^{[k]} - \frac{1}{\frac{f'\left(x_i^{[k]}\right)}{f\left(x_i^{[k]}\right)} - \sum\limits_{\substack{j=1 \\ j\neq i}}^{n}\left(\frac{1}{\left(x_i^{[k]} - z_j^{[k]}\right)}\right)}, \tag{68}$$

where

$$s_j^{[k]} = x_j^{[k]} - \frac{f\left(x_j^{[k]}\right)}{f'\left(x_j^{[k]}\right)},$$

$$v_j^{[k]} = s_j^{[k]} - \frac{f\left(x_j^{[k]}\right) f\left(s_j^{[k]}\right)\left(\frac{f\left(x_j^{[k]}\right)}{f'\left(x_j^{[k]}\right)}\right)}{\left(f\left(x_j^{[k]}\right) - f\left(s_j^{[k]}\right)\right)^2},$$

$$z_j^{[k]} = v_j^{[k]} - \left(\frac{\left(s_j^{[k]} - f_j^{[k]}\right) f\left(v_j^{[k]}\right)\left(\frac{f\left(x_j^{[k]}\right)}{f'\left(x_j^{[k]}\right)}\right)}{\left(f\left(x_j^{[k]}\right) - f\left(v_j^{[k]}\right)\right)^2}\right)\left[f\left(s_j^{[k]}\right) + \frac{\left(f\left(x_j^{[k]}\right)\right)^2}{\left(f\left(s_j^{[k]}\right) - f\left(v_j^{[k]}\right)\right)}\right].$$

- The Wang–Wu method (WWM$^{[*]}$):

$$x_i^{[k+1]} = x_i^{[k]} - \left[\frac{1}{h_i^{[k]}} - \frac{\left(\frac{f\left(x_j^{[k]}\right)}{f'\left(x_j^{[k]}\right)}\right)}{2}\left(\left(\psi_{1,i}^{[k]}\right)^2 + \left(\psi_{2,i}^{[k]}\right)\right)\right]^{-1}. \tag{69}$$

- The Farmer–Loizou method (FLM$^{[*]}$) combined with the Newton method at the first step:

$$x_i^{[k+1]} = x_i^{[k]} - \left[\frac{\left(\frac{f\left(x_j^{[k]}\right)}{f'\left(x_j^{[k]}\right)}\right)\phi_{2,i}^{[*]}}{1 - 2\left(\frac{f\left(x_j^{[k]}\right)}{f'\left(x_j^{[k]}\right)}\right)\phi_{2,i}^{[*]} + \left(\frac{\left(u_i^{[k]}\right)^2}{2}\right)}\left(\left(\phi_{2,i}^{[*]}\right)^2 - \left(\psi_{2,i}^{[k]}\right)\right)\right]^{-1}, \tag{70}$$

where

$$h_i^{[k]} = \left(\frac{f'\left(x_j^{[k]}\right)}{f\left(x_j^{[k]}\right)} - \frac{f''\left(x_j^{[k]}\right)}{2f'\left(x_j^{[k]}\right)}\right), \quad \psi_{\sigma,i}^{[k]} = \sum_{\substack{j=1 \\ j\neq i}}^{n}\left(\frac{1}{\left(x_i^{[k]} - x_j^{[k]} + \left(\frac{f\left(x_j^{[k]}\right)}{f'\left(x_j^{[k]}\right)}\right)\right)^{\sigma}}\right),$$

$$\phi_{2,i}^{[*]} = \left(\frac{f''\left(x_j^{[k]}\right)}{2f'\left(x_j^{[k]}\right)}\right), \quad u_i^{[k]} = \frac{f\left(x_i^{[k]}\right)}{f'\left(x_i^{[k]}\right)}, \quad \sigma = 1, 2.$$

Compared with existing approaches, all proposed iterative techniques demonstrate better convergence characteristics and lower computational cost. The enhanced computational efficiency of the proposed methods results from their streamlined parallel structure and the reduced number of arithmetic operations per iteration. The percentage computational efficiency between simultaneous schemes is computed as

$$E(\Phi(M_1), \Phi(M_2)) = \left[\frac{\Phi(M_1)}{\Phi(M_2)} - 1\right] \times 100,$$

where a positive value of E indicates that the proposed method M_1 i.e., WKM$^{[*]}$ and WUM$^{[*]}$ are more computationally efficient than the reference methods M_2, namely MPM$^{[*]}$, WWM$^{[*]}$, and FLM$^{[*]}$. This metric quantifies the improvement achieved through algorithmic optimization and parallelization by comparing performance against a sequential baseline.

Table 1. Arithmetic operation counts for different methods.

Operations	MPM$^{[*]}$	WWM$^{[*]}$	FLM$^{[*]}$	WKM$^{[*]}$	WUM$^{[*]}$
Addition and Subtraction	$22n^2 + O(n)$	$29n^2 + O(n)$	$27n^2 + O(n)$	$15n^2 + O(n)$	$15n^2 + O(n)$
Multiplication	$18n^2 + O(n)$	$26n^2 + O(n)$	$26n^2 + O(n)$	$13n^2 + O(n)$	$14n^2 + O(n)$
Division	$2n^2 + O(n)$	$2n^2 + O(n)$	$2n^2 + O(n)$	$2n^2 + O(n)$	$2n^2 + O(n)$

Based on the data presented above, Table 2 reports the percentage computational efficiencies. The proposed scheme (WUM$^{[*]}$) achieves the highest efficiency owing to its superior order of convergence and reduced arithmetic complexity, followed by WKM$^{[*]}$. In contrast, conventional schemes (MPM$^{[*]}$, WWM$^{[*]}$, and FLM$^{[*]}$) exhibit lower efficiency because of their higher operation counts.

Table 2. Percentage computational efficiency (%) of simultaneous root-finding methods.

Method	MPM$^{[*]}$	WWM$^{[*]}$	FLM$^{[*]}$	WKM$^{[*]}$	WUM$^{[*]}$
Order of Convergence (r)	10	10	10	10	10
Percentage Efficiency(η)	87.3	79.4	69.6	89.0	91.3

Compared with the classical iterative schemes (MPM$^{[*]}$, WWM$^{[*]}$, and FLM$^{[*]}$), it is evident from Table 2 and Figure 1 that the proposed approaches (WKM$^{[*]}$ and WUM$^{[*]}$) achieve substantially higher percentage computational efficiency. This improvement primarily stems from their reduced arithmetic complexity and higher order of convergence, which lead to fewer iterations and more effective utilization of computational resources. The iterative schemes WKM$^{[*]}$ and WUM$^{[*]}$ thus provide an optimal balance among accuracy, speed, and numerical stability.

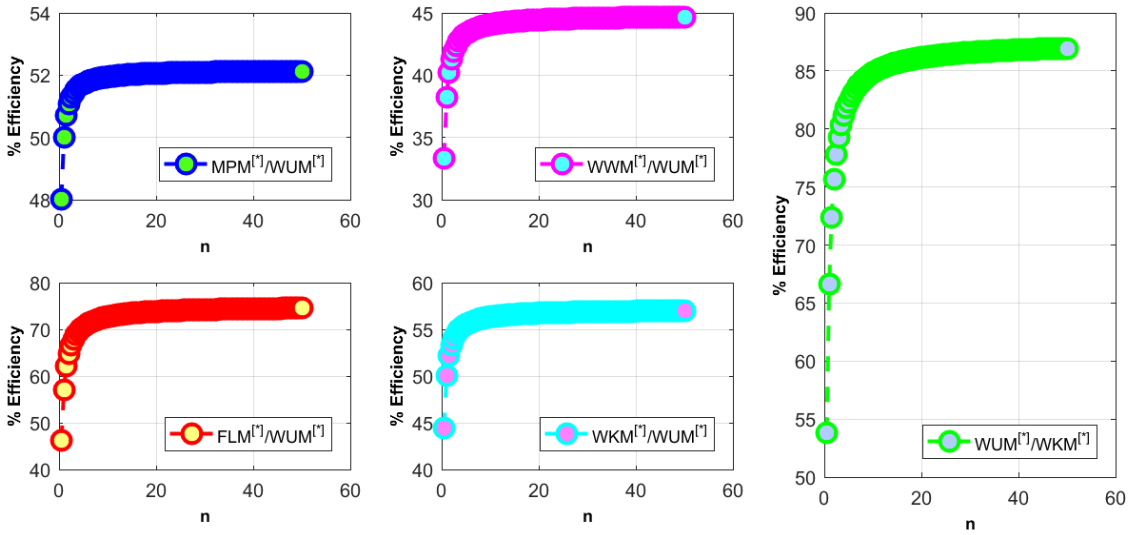


Figure 1. Percentage computational efficiency of the simultaneous schemes with respect to each other.

4. Implementation of the Methodology and Numerical Outcomes

This section presents a numerical framework for computing all distinct and multiple roots of (1) arising in engineering, natural, and medical sciences. For the comparative analysis, we consider several benchmark methods from the literature, including lower-order schemes such as the third-order WDK$^{[*]}$ method and the fifth-order Zhang method [69] (ZHM$^{[*]}$), as well as three well-known tenth-order simultaneous techniques for multiple roots, namely MPM$^{[*]}$, WWM$^{[*]}$, and FLM$^{[*]}$. The evaluation is based on CPU time, memory usage, residual error, and the number of iterations required for convergence, using randomly chosen and mutually distinct initial guesses in the vicinity of the exact roots. In all numerical experiments, the free parameter β appearing in the proposed schemes WKM$^{[*]}$ and WUM$^{[*]}$ is fixed at $\beta = 0.5$.

4.1. Computer Architecture, Implementation of the Schemes, and Outcomes

All simulations were performed on a PC equipped with an Intel Core i7 processor and 16 GB of RAM, using variable-precision arithmetic (VPA) in MATLAB R2016b. Convergence was declared when the residual norm fell below the prescribed tolerance (`tol`). The following performance metrics were evaluated:

- Iteration count (k);
- Maximum observed order of convergence (Max-σ_i^{n-1});
- Computational time in seconds (CPU time);
- Memory usage in megabytes (Mem-Usage (MB));
- Total number of arithmetic operations ($[\pm, \times, \div]$);
- Maximum error across all cores (Max-Error (all cores)).

4.1.1. Implementations

The numerical implementation of the scheme consists of the following components:

1. **Initial guess selection criteria.** Algorithm 1 updates each solution component in parallel using randomly generated vectors. Initial approximations are constructed for standard nonlinear functions with symmetric roots of varying multiplicities, arising in applications from biomedical, mechanical, and electrical engineering. The initial vector is defined as
$$x^{[0]} = \left[x_1^{[0]}, \cdots, x_n^{[0]}\right]^t. \tag{71}$$
To assess the convergence behavior of the scheme, its performance is evaluated using both nearby and distant initial guesses.
2. **Stopping criteria.** For each sampled initial vector, the iterative process is terminated when the following condition is satisfied:
$$\left\| x_i^{[k+1]} - x_i^{[k]} \right\| \leq tol, \tag{72}$$
where $tol = 10^{-64}$. All computations are carried out using 128-digit variable-precision arithmetic via the `vpa` function in MATLAB, ensuring high numerical accuracy. Among all sampled vectors, the solution achieving the highest accuracy in terms of the Euclidean norm is retained.
3. **Parallel execution using `parfor` in MATLAB.** The proposed framework exploits parallelism by computing all roots simultaneously, where each root is independently updated from a distinct initial guess. The `parfor` construct enables parallel execution of independent loop iterations across multiple CPU cores, thereby reducing computation time for computationally intensive tasks such as iterative root-finding and fractional-order system simulations. To evaluate the efficiency of the parallel imple-

mentation, we measure the parallel CPU time (T_{para}), the serial CPU time (T_{seri}), and the corresponding speedup ratio defined by

$$\phi_{\text{sp}} = \frac{T_{\text{seri}}}{T_{\text{para}}}, \tag{73}$$

where T_{para} denotes execution on multiple cores using `parfor`, and T_{seri} corresponds to execution on a single core without parallelization. This `parfor`-based approach preserves the numerical accuracy and stability of the serial implementation while achieving significant reductions in computational time.
An illustrative MATLAB implementation is given below:

```
parfor i = 1:N
    results(i) = myFunction(inputs(i));
end
```

Each iteration is executed independently, allowing the concurrent computation of all roots in polynomial problems. This strategy efficiently exploits available memory, reduces data transfer overhead, and enhances the scalability of simultaneous root–finding methods for large problem sizes.

Algorithm 1 Generation of Initial Guesses for Polynomial Equation Roots

Require: Number of roots n, polynomial degree n, or $f(x) = 0$ with multiplicity m. Convert f, f' to numerical function handles.

Ensure: Initial guesses $x^{[0]}$ for root-finding methods

1: Generate equally spaced angles on the unit circle:

$$\theta_0 = \text{linspace}(0, 2\pi, n+1), \quad \theta_0(n+1) \text{ removed}$$

2: Map angles to points on the unit circle:

$$x^{[0]} = e^{i\theta_0}$$

3: Add a small random perturbation for better convergence:

$$x^{[0]} = x^{[0]} \cdot \left(1 + 10^{-6}\text{randn}(1, n) + 10^{-6} i\, \text{randn}(1, n)\right)$$

4: Convert to high-precision arithmetic:

$$x^{[0]} = \text{vpa}((x^{[0]})^T)$$

5: **return** $x^{[0]}$

4.1.2. Visualization and Validation

Iterative techniques are validated by comparing the computed solutions with benchmark or exact results. Such comparisons allow for the evaluation of the method's accuracy and convergence properties. To illustrate these characteristics effectively, both the reference and computed results are presented together. Algorithm 2 summarizes the complete procedure of the proposed method, while Figure 2 depicts the data relationships and overall computational workflow.

Algorithm 2 Root Trajectories for Three Simultaneous Methods (WKM$^{[*]}$, WUM$^{[*]}$, MPM$^{[*]}$)

Input: Polynomial $f(x)$ of degree n, tolerance tol, maximum iterations $N_{\max}$
Output: Approximated roots $\{x_i\}$, function evaluations, derivative evaluations, arithmetic operations, maximum error

Initialization:
Expand polynomial $f(x)$ and extract coefficients
Compute symbolic derivative $f'(x)$
Obtain exact roots $\{r_i\}$ symbolically
Generate n initial guesses $x_i^{[0]}$ on perturbed unit circle
Initialize counters: function evaluations, derivative evaluations, arithmetic operations
for each method $M \in \{$WKM$^{[*]}$, WUM$^{[*]}$, MPM$^{[*]}$, WWM$^{[*]}$ and FLM$^{[*]}\}$ **do**
    Set $x_i^{[0]} = x0$, iteration counter $k = 0$
    repeat
        for $i = 1$ to n **do**
            Evaluate $f(x_i^{[k]})$
            Compute predictor

$$y_i^{[k]} = x_i^{[k]} - \frac{f(x_i^{[k]})}{\prod_{\substack{j=1 \\ j\neq i}}^{n}\left(x_i^{[k]} - z_j^{[k]}\right)}$$

            Compute coupling product

$$P_i^{[k]} = \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{y_i^{[k]} - y_j^{[k]}}{x_i^{[k]} - x_j^{[k]}}$$

            Update approximation

$$x_i^{[k+1]} = y_i^{[k]} - \left[2 - P_i^{[k]} + \frac{5}{4}\left(1 - P_i^{[k]}\right)^2 - \frac{1}{6}\left(1 - P_i^{[k]}\right)^3\right] \frac{f(y_i^{[k]})}{\prod_{\substack{j=1 \\ j\neq i}}^{n}\left(y_i^{[k]} - y_j^{[k]}\right)}$$

        end for
        $k \leftarrow k + 1$
    until maximum update $<$ tol or $k = N_{\max}$
    Store trajectories $x_i^{[0]}, x_i^{[1]}, \ldots, x_i^{[k]}$
    Compute maximum error:

$$E = \max_i |x_i^{[k+1]} - x_i^{[k]}|$$

    Record counters (function/derivative evaluations, arithmetic operations)
end for

Output Results:
Plot root trajectories for each method with unit circle
Print summary table with metrics for each method

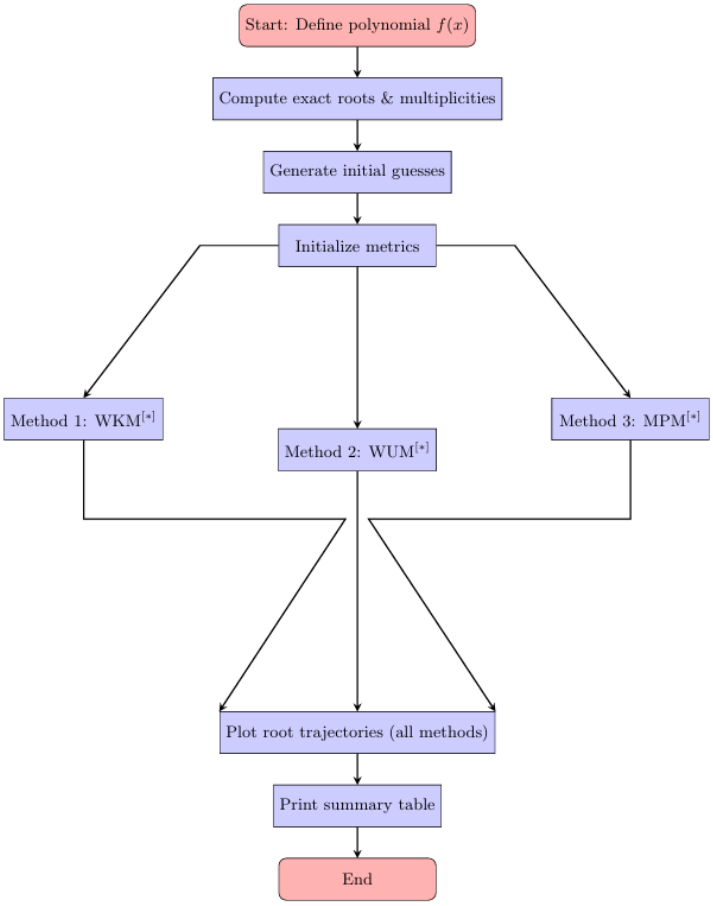


Figure 2. Convergence trajectories of the simultaneous method applied to $f(x) = 0$.

4.2. Standard Polynomials with Multiple Roots [70]

In biomedical applications, benchmark models are employed to evaluate the accuracy and reliability of numerical techniques. They provide a controlled environment that enables a systematic assessment of computational performance under challenging conditions, such as complex interactions and highly polynomial equation systems commonly found in biological processes. In particular, benchmark problems based on fractional-order parallel models are useful for assessing the accuracy, stability, and convergence of numerical methods applied to biological and medical system simulations.

4.2.1. Example 1: Standard Test Functions with Multiple Roots

We consider a benchmark nonlinear function with a symmetric root structure and known multiplicity to assess the performance of the proposed simultaneous methods. Specifically, the test function is

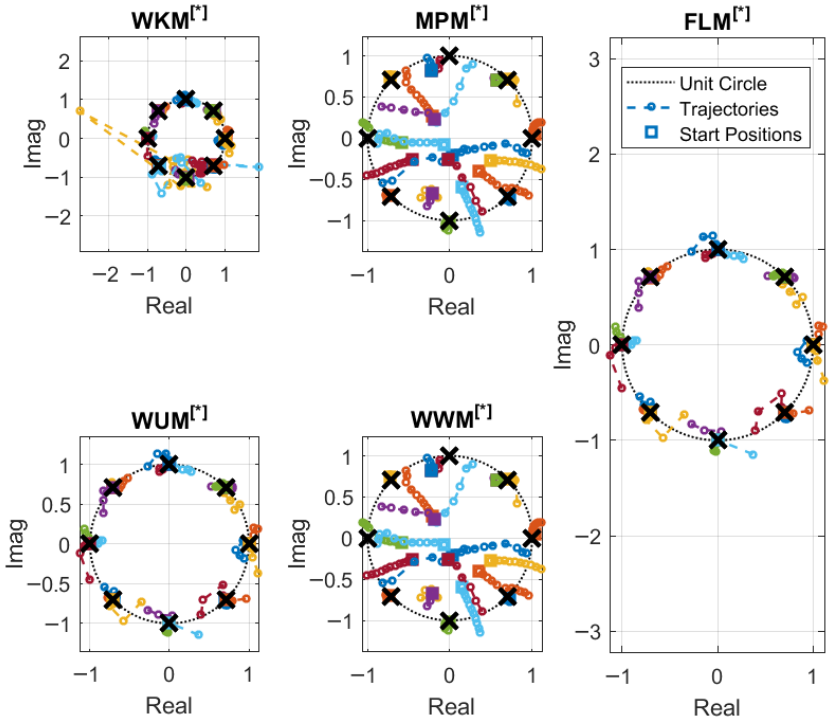
$$f(x) = \prod_{k=0}^{7} \left(x - \exp\left(\frac{4\pi k i}{4}\right)\right)^3, \tag{74}$$

which has eight distinct complex roots,

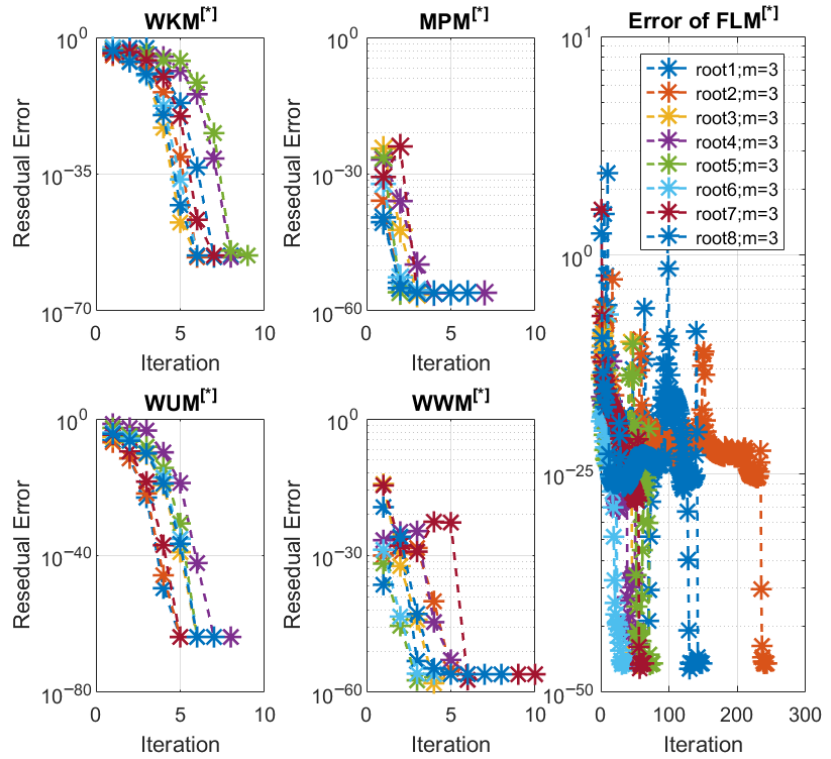
$$\left\{1, e^{\frac{\pi i}{4}}, e^{\frac{\pi i}{2}}, e^{\frac{3\pi i}{4}}, -1, e^{\frac{5\pi i}{4}}, e^{\frac{3\pi i}{2}}, e^{\frac{7\pi i}{4}}\right\},$$

each with multiplicity of three. This example is employed in the subsequent analysis to illustrate the stability and numerical efficiency of the proposed schemes.

The numerical results for $f(x)$ are summarized in Figure 3a,b and Table 3. This example exhibits a higher degree of symmetry and root multiplicity, thereby constituting a stringent test under randomly generated initial guesses (Appendix A Table A1) and further confirming the robustness of the proposed method in handling multiple roots.



(**a**) Root trajectory of (74).



(**b**) Error graph of (74).

Figure 3. (**a**,**b**) Trajectory of the roots and error graph for the function $f(x)$ using the simultaneous schemes.

Table 3. Consistency analysis of the simultaneous schemes for solving (74).

Metric	Max-Error	CPU-Time	Mem-Usage (Mbs)	$[\pm, \times, \div]$	Iterations (k)	Max-σ_i^{n-1}
			Comparison with lower-order methods			
WDK$^{[*]}$	1.4×10^{-9}	2.5673	67.012	1743	17	3.8654
ZHM$^{[*]}$	6.3×10^{-17} †	2.245	55.679	2145	13	4.6475
			Comparison with higher-order methods			
WKM$^{[*]}$	9.7×10^{-45} †	1.1332	45.796	2554	8	9.006
WUM$^{[*]}$	4.8×10^{-53} †	1.3758	44.109	2342	6	10.007
MPM$^{[*]}$	8.0×10^{-45} †	1.9697	53.005	2564	10	9.370
WWM$^{[*]}$	4.7×10^{-44} †	1.5643	45.367	2613	11	9.087
FLM$^{[*]}$	9.7×10^{-35} †	2.0872	49.115	2744	10	9.657

† All computations were performed using MATLAB variable precision arithmetic (VPA) with `digits` = 128, employing a numerical tolerance of 10^{-64}.

Table 3 presents a consistency analysis of the considered simultaneous schemes for solving (74), evaluated in terms of accuracy, computational cost, and numerical stability. Among the lower-order approaches, WDK$^{[*]}$ and ZHM$^{[*]}$ achieve acceptable levels of accuracy; however, they require a larger number of iterations, higher CPU time, and increased memory usage. In contrast, the higher-order schemes exhibit significantly improved performance across all considered metrics. In particular, WUM$^{[*]}$ attains the smallest maximum error (10^{-53}), converges in only six iterations, and shows the lowest CPU time and memory footprint among the methods examined. Although WKM$^{[*]}$ and MPM$^{[*]}$ also provide high accuracy, they require a greater number of iterations and arithmetic operations when compared to WUM$^{[*]}$. Overall, the results indicate that WUM$^{[*]}$ and WKM$^{[*]}$ offer a favorable balance between accuracy, convergence speed, and computational efficiency, making them particularly effective and reliable for the problem under consideration.

The simulation results reported in Table 4 were obtained using MATLAB's `parfor` construct with 1, 2, and 4 computing cores for the considered simultaneous schemes. Across all core configurations, the methods exhibit high accuracy and consistent numerical behavior. Among them, the proposed WKM$^{[*]}$ and WUM$^{[*]}$ schemes achieve the lowest computational times for all tested core counts. As a result, these two methods show improved performance compared with the existing WDK$^{[*]}$, ZHM$^{[*]}$, MPM$^{[*]}$, WWM$^{[*]}$, and FLM$^{[*]}$ schemes in terms of computational efficiency while maintaining comparable or higher numerical accuracy.

Table 4. Comparison of speedup ratio (ϕ_{sp}) and maximum error for existing and proposed schemes for solving (74).

Cores	WDK$^{[*]}$	ZHM$^{[*]}$	WKM$^{[*]}$	WUM$^{[*]}$	MPM$^{[*]}$	WWM$^{[*]}$	FLM$^{[*]}$
1	0.0154	0.0235	0.0050	0.0063	0.0161	0.0135	0.0122
2	0.0436	0.0342	0.0101	0.0116	0.0232	0.0194	0.0185
4	0.0453	0.0234	0.0223	0.0239	0.0320	0.0305	0.0280
Max-Error (all cores)	7.0×10^{-10}	8.5×10^{-27} †	1.9×10^{-75} †	2.8×10^{-74} †	5.1×10^{-48} †	1.7×10^{-64} †	7.1×10^{-52} †

† All computations were performed using MATLAB variable precision arithmetic (VPA) with `digits` = 128, employing a numerical tolerance of 10^{-64}.

4.2.2. Example 2

Mignotte's polynomial [71] is a specially constructed polynomial commonly used in numerical analysis and computational algebra to test the stability and robustness of root-finding algorithms. It is defined as

$$f(x) = x^n - (ax - 1)^n, \tag{75}$$

where n and a are positive integers that control the polynomial's degree and the spacing of its roots. For example, setting $n = 18$ and $a = 9$ gives

$$f(x) = x^{18} - 81x^2 + 18x - 1. \tag{76}$$

The numerical results used to assess the efficiency of the simultaneous scheme are reported in Table 5. The proposed schemes are evaluated using randomly generated initial guesses located far from the exact solutions, as listed in Appendix A Table A2. Starting from these initial points, the schemes converge to the exact roots within a small number of iterations, as documented in Table 5.

Table 5. Consistency analysis of the simultaneous schemes for solving (76).

Metric	Max-Error	CPU-Time	Mem-Usage (Mbs)	$[\pm, \times, \div]$	Iterations (k)	Max-σ_i^{n-1}
			Comparison with lower-order methods			
WDK$^{[*]}$	1.07×10^{-9}	4.5475	88.0435	5087	18	3.543
ZHM$^{[*]}$	0.91×10^{-15}	4.7546	94.4536	5784	17	4.764
			Comparison with higher-order methods			
WKM$^{[*]}$	3.45×10^{-36} †	4.7803	76.556	5463	10	8.878
WUM$^{[*]}$	8.35×10^{-42} †	3.9016	67.879	4998	6	9.007
MPM$^{[*]}$	1.09×10^{-19} †	5.3753	86.445	5987	10	7.719
WWM$^{[*]}$	0.11×10^{-16}	5.7864	88.675	5908	12	6.331
FLM$^{[*]}$	4.54×10^{-15}	4.9865	91.987	6001	13	7.063

† All computations were performed using MATLAB variable precision arithmetic (VPA) with `digits` = 128, employing a numerical tolerance of 10^{-64}.

The consistency results summarized in Table 5 provide a further assessment of the performance of the simultaneous schemes for solving (76) under identical computational settings. The lower-order methods, namely WDK$^{[*]}$ and ZHM$^{[*]}$, exhibit comparatively larger errors and require a greater number of iterations, resulting in increased CPU time and memory consumption. In contrast, the higher-order schemes offer substantial improvements in both accuracy and computational efficiency. In particular, WUM$^{[*]}$ attains the smallest maximum error (10^{-42}), converges in the fewest iterations, and requires fewer computational resources than the remaining methods. Although WKM$^{[*]}$ and MPM$^{[*]}$ also achieve competitive accuracy, their overall computational cost remains higher. These results indicate that WUM$^{[*]}$ provides the most favorable balance between accuracy, convergence speed, and computational efficiency for solving (76).

The simulation results reported in Table 6 for the simultaneous schemes were obtained using MATLAB's `parfor` with 1, 2, and 4 computing cores. All schemes produce highly accurate results with negligible errors. The proposed WKM$^{[*]}$ method demonstrates the best time performance, closely followed by WUM$^{[*]}$. Both approaches exhibit faster convergence and lower computational cost compared with the traditional WDK$^{[*]}$, ZHM$^{[*]}$, MPM$^{[*]}$, and FLM$^{[*]}$ schemes.

Table 6. Comparison of speedup ratio (ϕ_{sp}) and maximum error for existing and proposed schemes for solving (76).

Cores	WDK$^{[*]}$	ZHM$^{[*]}$	WKM$^{[*]}$	WUM$^{[*]}$	MPM$^{[*]}$	WWM$^{[*]}$	FLM$^{[*]}$
1	0.0435	0.0325	0.0042	0.0057	0.0149	0.0131	0.0120
2	0.0546	0.0367	0.0088	0.0109	0.0215	0.0192	0.0178
4	0.0643	0.0507	0.0205	0.0228	0.0304	0.0288	0.0271
Max-Error (all cores)	6.4×10^{-10}	4.1×10^{-19} †	2.1×10^{-75} †	1.9×10^{-74} †	4.6×10^{-47} †	1.5×10^{-63} †	5.9×10^{-52} †

† All computations were performed using MATLAB variable precision arithmetic (VPA) with `digits` = 128, employing a numerical tolerance of 10^{-64}.

The results reported in Tables 5 and 6 indicate that, in terms of CPU time, number of arithmetic operations per iteration, and local order of convergence, the proposed method for unknown multiplicities exhibits improved performance compared with the WKM$^{[*]}$ and MPM$^{[*]}$ schemes. Moreover, the consistency analysis shows that the newly developed methods achieve substantially better performance than the classical approaches reported in the literature. The corresponding error plots shown in Figure 4 further illustrate that the proposed method requires fewer iterations and converges more rapidly, yielding smaller residual errors when computing all roots of (76) simultaneously.

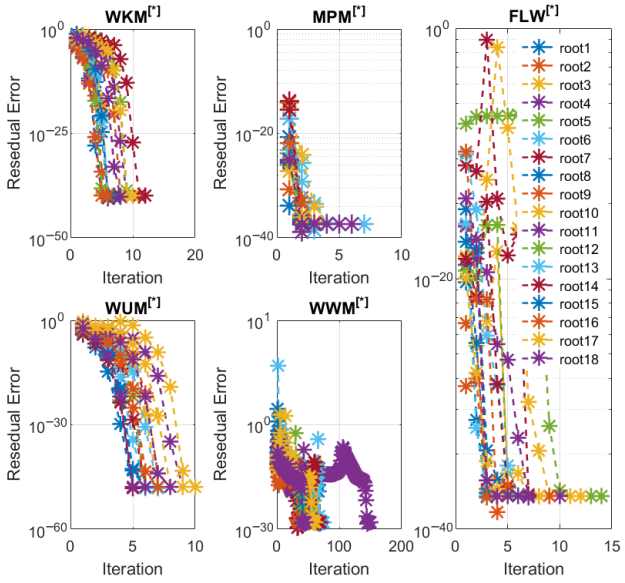


Figure 4. Error graph for the function (76) using the simultaneous schemes.

4.3. Applications in Biomedical Engineering

Benchmark models are employed to evaluate the reliability and effectiveness of numerical methods in real biomedical contexts. They provide a structured framework for assessing computational performance in challenging scenarios, such as highly nonlinear systems. In particular, benchmarks based on fractional-order simultaneous systems enable the systematic evaluation of convergence, accuracy, and stability in simulations of biological and medical processes.

4.3.1. Pharmacokinetics Delayed Absorption Model [72]

Pharmacokinetics describes the processes of drug absorption, distribution, metabolism, and elimination. In the delayed absorption model with nonlinear elimination, the plasma drug concentration $C(t)$ evolves according to

$$\frac{dC}{dt} = K_{es}C(t) + \frac{V_{\max}\, C(t)^n}{K_m^n + C(t)^n}, \tag{77}$$

where K_{es} denotes the linear elimination/absorption rate constant, $V_{\max}$ is the maximum rate of nonlinear elimination, K_m is the Michaelis–Menten constant, and n is the Hill coefficient.

Equation Breakdown.

- $C(t)$: plasma drug concentration at time t;
- K_{es}: linear clearance rate (time^{-1});
- $V_{\max}$: maximum nonlinear clearance rate (concentration/time);
- K_m: half-saturation constant (concentration);
- n: Hill coefficient, quantifying cooperativity.

The first term in (77) represents linear clearance, while the second term captures saturable clearance with cooperative binding.

Equilibrium (Steady State). At steady state, $\frac{dC}{dt} = 0$, leading to

$$K_{es}C + \frac{V_{\max}C^n}{K_m^n + C^n} = 0. \tag{78}$$

Multiplying by $(K_m^n + C^n)$ yields

$$K_{es}K_m^n C + K_{es}C^{n+1} + V_{\max}C^n = 0. \tag{79}$$

Factoring C gives

$$C\left(K_{es}C^n + V_{\max}C^{n-1} + K_{es}K_m^n\right) = 0, \tag{80}$$

showing that $C = 0$ (drug-free state) is always a root, while non-trivial equilibria satisfy

$$K_{es}C^n + V_{\max}C^{n-1} + K_{es}K_m^n = 0. \tag{81}$$

Multiple Roots Behavior. Equation (79) is polynomial-like in C, and the number of real positive solutions depends on n:

- $n = 1$: a unique positive equilibrium exists.
- $n > 1$: higher-degree equations may yield multiple real equilibria, leading to multistability.

Different steady states may correspond to physiologically safe concentrations (stable equilibria) or to toxic accumulation (unstable equilibria). For biologically plausible parameter values,

$$K_{es} = 0.2\ \text{hr}^{-1}, \quad V_{\max} = 3\ \text{mg/L/hr}, \quad K_m = 1.5\ \text{mg/L}, \quad n = 3,$$

the steady-state equation becomes

$$0.2C^4 + 3C^2 + 0.675C = 0, \tag{82}$$

which factors as

$$f(C) = \begin{bmatrix} C(C-(0.1121241057+3.877849332i))^3(C+0.2242482115)^4 \\ (C-(0.1121241057-3.877849332i))^5 \end{bmatrix}. \quad (83)$$

Thus, $C = 0$ represents a trivial solution, whereas the cubic term may yield up to three additional real roots, illustrating the possibility of multiple equilibria and multistability in drug concentration dynamics.

The efficiency of the proposed numerical schemes is evaluated using randomly generated initial guesses located far from the exact solutions, as listed in Appendix A Table A3. Starting from these initial points, the WKM$^{[*]}$ and WUM$^{[*]}$ schemes converge to the exact roots within a small number of iterations, as reported in Table 7.

Table 7. Consistency analysis of the simultaneous schemes for solving (83).

Metric	Max-Error	CPU-Time	Mem-Usage (Mbs)	$[\pm, \times, \div]$	Iterations (k)	Max-σ_i^{n-1}
Comparison with lower-order methods						
WDK$^{[*]}$	3.47×10^{-21} †	2.435	47.43	1765	13	4.0214
ZHM$^{[*]}$	8.09×10^{-35} †	3.546	65.67	1943	14	5.0437
Comparison with higher-order methods						
WKM$^{[*]}$	7.56×10^{-37} †	1.897	34.78	1665	9	9.7651
WUM$^{[*]}$	1.65×10^{-66} †	1.253	25.54	1324	5	10.007
MPM$^{[*]}$	1.09×10^{-34} †	2.009	31.76	2730	10	9.1432
WWM$^{[*]}$	7.14×10^{-31} †	3.789	43.43	2175	11	9.5655
FLM$^{[*]}$	1.09×10^{-27} †	3.165	39.87	2843	11	9.8791

† All computations were performed using MATLAB variable precision arithmetic (VPA) with `digits` = 128, employing a numerical tolerance of 10^{-64}.

Table 7 presents a consistency and performance comparison of the proposed and existing simultaneous schemes for solving (83). Among the lower-order methods, WDK$^{[*]}$ exhibits moderate accuracy and computational cost, whereas ZHM$^{[*]}$ achieves improved accuracy at the expense of increased CPU time, memory usage, and iteration count. In contrast, the higher-order schemes show substantially improved performance across the considered metrics. In particular, WUM$^{[*]}$ attains the smallest maximum error (1.65×10^{-66}), the lowest CPU time and memory consumption, as well as the minimum number of iterations ($k = 5$). Although WKM$^{[*]}$ and MPM$^{[*]}$ also demonstrate competitive accuracy and reduced iteration counts relative to the classical schemes, they require a higher number of arithmetic operations than WUM$^{[*]}$. Overall, these results indicate that WUM$^{[*]}$ offers the most favorable balance between accuracy, computational efficiency, and consistency among the methods considered.

The simulation results reported in Table 8 for the simultaneous schemes were obtained using MATLAB's `parfor` with 1, 2, and 4 computing cores. The WUM$^{[*]}$ and WKM$^{[*]}$ methods demonstrate high efficiency, achieving consistently minimal computation times as compared to WDK$^{[*]}$, ZHM$^{[*]}$, MPM$^{[*]}$ and FLM$^{[*]}$. Their accuracy remains stable across all core configurations, confirming strong scalability and robustness against precision loss during simultaneous execution.

Table 8. Comparison of speedup ratio (ϕ_{sp}) and maximum error for existing and proposed schemes for solving (83).

Cores	WDK$^{[*]}$	ZHM$^{[*]}$	WKM$^{[*]}$	WUM$^{[*]}$	MPM$^{[*]}$	WWM$^{[*]}$	FLM$^{[*]}$
1	0.0235	0.0435	0.0053	0.0066	0.0172	0.0151	0.0137
2	0.0546	0.0687	0.0107	0.0122	0.0240	0.0207	0.0193
4	0.0654	0.0785	0.0241	0.0259	0.0341	0.0326	0.0299
Max-Error (all cores)	9.5×10^{-26} †	8.7×10^{-39} †	1.3×10^{-74} †	3.0×10^{-75} †	6.2×10^{-48} †	2.5×10^{-64} †	8.2×10^{-53} †

† All computations were performed using MATLAB variable precision arithmetic (VPA) with `digits` = 128, employing a numerical tolerance of 10^{-64}.

Figure 5 illustrates the convergence behavior, including the error evolution of all roots for Equation (83). The error plots show that the proposed method converges in fewer iterations than classical approaches of higher-order (MPM$^{[*]}$, WWM$^{[*]}$, and FLM$^{[*]}$) and achieves faster convergence with smaller residual errors when computing all roots of (83) simultaneously.

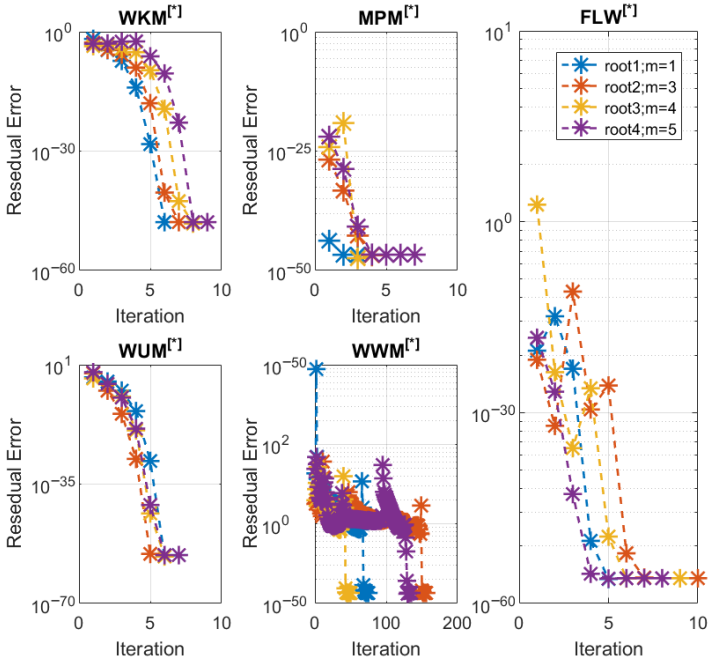


Figure 5. Error graph for the function (83) using the simultaneous schemes.

Physical Interpretation.

- At low concentrations ($C \ll K_m$), nonlinear clearance is weak and elimination is dominated by linear clearance.
- At high concentrations ($C \gg K_m$), elimination saturates near V_{max}.
- Multiple equilibria imply that plasma drug levels may stabilize at different values depending on initial dosage.

4.3.2. Enzyme Kinetics with Cooperative Binding [73]

Enzyme kinetics with cooperative binding extends the classical Michaelis–Menten model by introducing the Hill coefficient n, which captures the cooperative effects in substrate binding. Here, the substrate concentration is denoted by $x(t)$, and the reaction rate follows a nonlinear saturation curve.

The governing rate equation is given by

$$f(x) = \frac{V_{\max} x^n}{K_m^n + x^n} - K_m x, \tag{84}$$

where $V_{\max}$ is the maximum reaction velocity, K_m is the half-saturation constant, and n is the Hill coefficient.

Equation Breakdown

- $x(t)$: Substrate concentration at time t.
- $V_{\max}$: Maximum rate of enzymatic reaction.
- K_m: Michaelis constant, concentration at which rate is half-maximal.
- n: Hill coefficient, measuring the degree of cooperativity.

Multiple Roots and Nonlinearity

The possibility of several steady-state solutions for $n > 1$, which corresponds to cooperative binding, is introduced by the nonlinear dependency on x^n. The model reduces to the single steady-state classical Michaelis–Menten equation for $n = 1$. For $n > 2$, the polynomial-like structure might have several real roots, causing bistability or multistability.

Consider the parameter set:

$$V_{\max} = 5 \text{ units}, \quad K_m = 2 \text{ units}, \quad n = 3.$$

Then, the rate equation becomes

$$f(x) = \left[(x + 0.7i)^4 (x - 0.9i)^3 (x + 1.7i)^2 (x - 1.4i)^2 (x - 0.765i)^3 (x - 1i)^4\right], \tag{85}$$

which exhibits a sigmoidal response curve and the potential for multiple steady states depending on system feedback. Random initial guesses listed in Appendix A Table A4 are used to assess the efficiency of the proposed scheme in comparison with existing methods, with the corresponding results reported in Table 9.

Table 9. Consistency analysis of the simultaneous schemes for solving (85).

Metric	Max-Error	CPU Time	Mem-Usage	$[\pm, \times, \div]$	Iterations (k)	Max-σ_i^{n-1}
			Comparison with lower-order methods			
WDK$^{[*]}$	0.16×10^{-11}	4.546	61.876	2187	15	3.657
ZHM$^{[*]}$	5.04×10^{-13}	4.756	72.657	2543	17	4.765
			Comparison with higher-order methods			
WKM$^{[*]}$	6.43×10^{-33} †	3.765	65.12	1875	9	9.768
WUM$^{[*]}$	0.67×10^{-49} †	2.887	58.786	1765	5	10.124
MPM$^{[*]}$	3.87×10^{-27} †	3.957	69.112	2231	10	9.765
WWM$^{[*]}$	1.13×10^{-27} †	4.012	76.562	2387	10	9.098
FLM$^{[*]}$	2.10×10^{-29} †	4.136	78.687	2354	10	9.125

† All computations were performed using MATLAB variable precision arithmetic (VPA) with `digits` = 128, employing a numerical tolerance of 10^{-64}.

Detailed error results and consistency metrics for (85) are presented in Table 9. The method converges rapidly to the exact roots, typically within ten iterations, and consistently attains smaller residuals with fewer iterations, indicating improved computational efficiency. In particular, the proposed WUM$^{[*]}$ approach achieves lower error magnitudes, reduced CPU time, and smaller memory usage, providing a favorable balance across all

considered performance metrics. These results suggest that the method effectively handles root multiplicity while maintaining numerical stability.

The simulation results reported in Table 10 were obtained using MATLAB's `parfor` construct with 1, 2, and 4 computing cores for the considered simultaneous schemes. The observed accuracy remains consistent across all core configurations, indicating stable numerical behavior. Among the tested methods, WKM$^{[*]}$ exhibits the shortest runtime, highlighting its suitability for large-scale nonlinear systems, whereas WUM$^{[*]}$ provides a favorable balance between accuracy and computational efficiency.

Table 10. Comparison of speedup ratio (ϕ_{sp}) and maximum error for existing and proposed schemes for solving (85).

Cores	WDK$^{[*]}$	ZHM$^{[*]}$	WKM$^{[*]}$	WUM$^{[*]}$	MPM$^{[*]}$	WWM$^{[*]}$	FLM$^{[*]}$
1	0.0123	0.0234	0.0049	0.0060	0.0151	0.0138	0.0115
2	0.0407	0.0378	0.0098	0.0114	0.0220	0.0195	0.0180
4	0.0564	0.0543	0.0224	0.0236	0.0314	0.0301	0.0275
Max-Error (all cores)	3.5×10^{-13}	7.1×10^{-17} †	1.0×10^{-79} †	2.9×10^{-75} †	3.0×10^{-53} †	5.3×10^{-65} †	4.8×10^{-51} †

† All computations were performed using MATLAB variable precision arithmetic (VPA) with `digits` = 128, employing a numerical tolerance of 10^{-64}.

Figure 6 illustrate the convergence behavior of the considered methods for Equation (85), including the error evolution of all computed roots. The error plots indicate that the proposed method converges in fewer iterations and achieves smaller residuals than the classical higher-order approaches (MPM$^{[*]}$, WWM$^{[*]}$, and FLM$^{[*]}$) when computing all roots simultaneously.

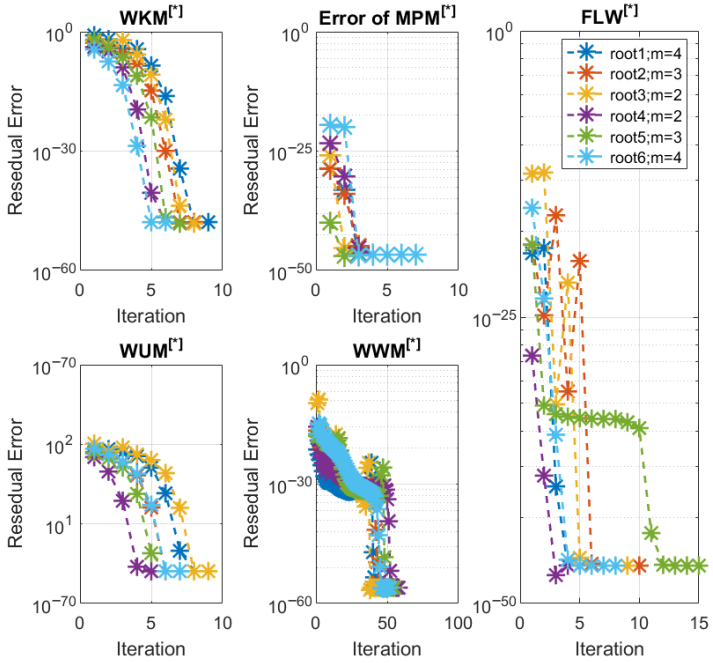


Figure 6. Error graph for the function (83) using the simultaneous schemes.

Physical Behavior

- At low $S \ll K_m$: $v(S) \approx \dfrac{V_{\max}}{K_m^n} S^n$, i.e., weak reaction rate.
- At high $S \gg K_m$: $v(S) \to V_{\max}$, i.e., reaction saturates.

- Cooperative binding ($n > 1$) yields sigmoidal kinetics, allowing sharp transitions in reaction rate and multiple equilibrium states.

4.3.3. Micro–Electro–Mechanical Systems [74]

Micro–electro–mechanical systems (MEMS) are miniature devices that integrate electrical and mechanical components—such as micro–cantilevers, resonators, and sensors—on a single silicon chip. Many MEMS resonators (either beam-type or plate-type) exhibit nonlinear vibration behavior when subjected to large electrostatic or piezoelectric excitation.

The amplitude–frequency relationship can be expressed as a nonlinear transcendental equation in the amplitude x:

$$f(x) = e^{-(x-A_1)^2} \sin^2(x - A_1) \tanh^2(x - A_2) = 0, \tag{86}$$

where A_1 and A_2 denote two distinct resonant amplitudes corresponding to different vibration modes or electrostatic equilibrium positions.

Mathematical Model

A concrete representative function illustrating two distinct multiple roots is by considering $A_1 = 1.1$ and $A_2 = 2.5$

$$f(x) = e^{-(x-1.1)^2} \sin^2(x - 1.1) \tanh^2(x - 2.5) = 0. \tag{87}$$

This function combines

- A Gaussian term $e^{-(x-1.1)^2}$ to localize the first mode around $x = 1.1$;
- A squared sine term $\sin^2(x - 1.1)$ representing a periodic restoring-force zero (vibration node);
- A squared hyperbolic tangent $\tanh^2(x - 2.5)$ modeling a nonlinear damping or electrostatic softening effect around $x = 2.5$.

The efficiency of the proposed numerical schemes is evaluated using randomly generated initial guesses located far from the exact solutions, as listed in Appendix A Table A5. Starting from these initial points, the schemes converge to the exact roots within a small number of iterations, as reported in Table 11. The corresponding error analysis indicates that the proposed method consistently yields smaller residuals and requires fewer iterations than the existing schemes, reflecting improved computational efficiency. Detailed consistency metrics and error analyses for (87) are presented in Table 11. In particular, the proposed WUM$^{[*]}$ approach attains lower error magnitudes, reduced CPU time, and smaller memory usage, offering a favorable balance across all considered performance metrics. These results suggest that the method effectively handles root multiplicity while maintaining numerical stability.

Table 11 summarizes the consistency analysis for solving (87) and highlights a clear separation between lower- and higher-order schemes. The lower-order methods, WDK$^{[*]}$ and ZHM$^{[*]}$, achieve acceptable accuracy but require more iterations and comparatively larger CPU time and memory usage. In contrast, the higher-order schemes significantly enhance performance. Notably, WUM$^{[*]}$ attains the highest accuracy (0.15×10^{-64}) while requiring the least CPU time and a minimal number of iterations, demonstrating remarkable computational efficiency. Although WKM$^{[*]}$, MPM$^{[*]}$, and WWM$^{[*]}$ also exhibit strong convergence properties, they remain inferior to WUM$^{[*]}$ in terms of overall cost-effectiveness. These findings confirm the superior consistency and efficiency of WUM$^{[*]}$ for this problem.

Table 11. Consistency analysis of the simultaneous schemes for solving (87).

Metric	Max-Error	CPU-Time	Mem-Usage	$[\pm, \times, \div]$	Iterations (k)	Max-σ_i^{n-1}
			Comparison with lower-order methods			
WDK$^{[*]}$	6.94×10^{-19} †	1.324	46.765	687	11	4.005
ZHM$^{[*]}$	8.32×10^{-21} †	1.432	57.786	846	10	4.986
			Comparison with higher-order methods			
WKM$^{[*]}$	0.67×10^{-53} †	0.0598	25.121	775	5	9.768
WUM$^{[*]}$	0.15×10^{-64} †	0.0073	23.099	705	4	10.124
MPM$^{[*]}$	3.35×10^{-41} †	1.0034	31.325	887	5	9.765
WWM$^{[*]}$	1.41×10^{-43} †	1.0563	43.875	954	6	9.154
FLM$^{[*]}$	1.72×10^{-39} †	1.0987	39.243	973	4	9.786

† All computations were performed using MATLAB variable precision arithmetic (VPA) with `digits` = 128, employing a numerical tolerance of 10^{-64}.

The simulation results reported in Table 12 for the simultaneous schemes were obtained using MATLAB's `parfor` with 1, 2, and 4 computing cores. Simultaneous execution enhances the performance of all methods; however, the proposed WKM$^{[*]}$ and WUM$^{[*]}$ schemes maintain the best overall balance between runtime and error accuracy. These approaches also demonstrate superior adaptability to multi-core architectures.

Table 12. Comparison of speedup ratio (ϕ_{sp}) and maximum error for existing and proposed schemes for solving (87).

Cores	WDK$^{[*]}$	ZHM$^{[*]}$	WKM$^{[*]}$	WUM$^{[*]}$	MPM$^{[*]}$	WWM$^{[*]}$	FLM$^{[*]}$
1	0.0342	0.0165	0.0056	0.0070	0.0165	0.0147	0.0131
2	0.0456	0.0375	0.0112	0.0128	0.0235	0.0209	0.0194
4	0.0698	0.053	0.0253	0.0269	0.0330	0.0312	0.0286
Max-Error (all cores)	7.1×10^{-23} †	9.3×10^{-31} †	1.6×10^{-75} †	2.2×10^{-74} †	4.2×10^{-48} †	1.8×10^{-64} †	6.4×10^{-52} †

† All computations were performed using MATLAB variable precision arithmetic (VPA) with `digits` = 128, employing a numerical tolerance of 10^{-64}.

Figure 7 illustrate the convergence behavior of the considered methods for Equation (87), including the error evolution of all computed roots. The error plots indicate that the proposed method converges in fewer iterations and attains smaller residuals than the classical higher-order approaches (MPM$^{[*]}$, WWM$^{[*]}$, and FLM$^{[*]}$) when computing all roots simultaneously.

Physical Interpretation of Multiple Roots.

Equation (87) exhibits two distinct roots corresponding to two stable vibration amplitudes:

- $x_1 \approx 1.1$: *Low-amplitude resonance*, dominated by linear stiffness and representing the fundamental vibration mode. The Gaussian and sine terms produce a localized and nearly flat zero crossing, indicating the presence of a multiple root.
- $x_2 \approx 2.5$: *High-amplitude resonance*, influenced by nonlinear cubic stiffness and electrostatic softening (modeled by $\tanh^2(x - 2.5)$). This root represents a secondary stable equilibrium at higher amplitude.

These multiple roots are physically significant because MEMS resonators often display bistability or jump phenomena in their amplitude–frequency response curves. From a numerical perspective, they constitute challenging test cases for iterative solvers, since classical Newton-type methods converge slowly near flat tangencies unless explicitly corrected for multiplicity.

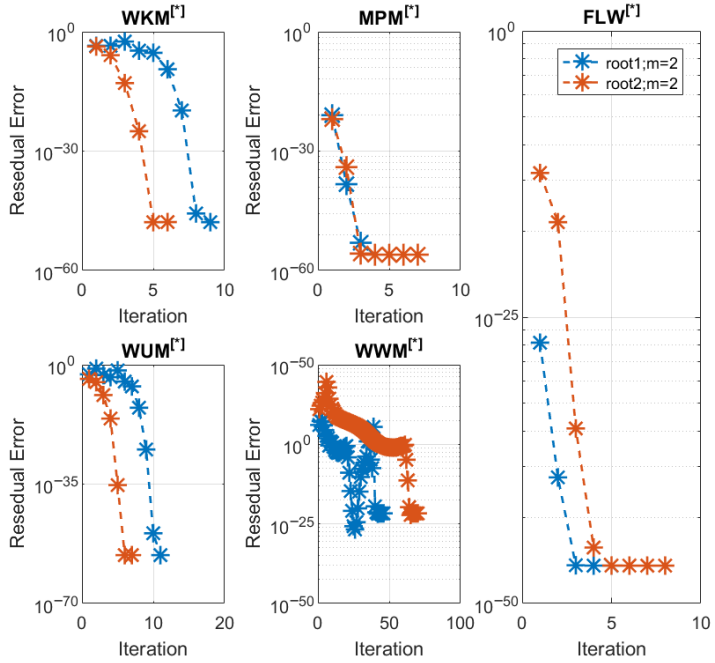


Figure 7. Error graph for the function (87) using the simultaneous schemes.

4.3.4. Nonlinear Bias Network [75]

We consider a nonlinear DC biasing node in an electronic circuit where multiple physical effects give rise to distinct equilibrium states. A representative smooth, non-polynomial model is given by

$$f(x) = \left(e^{\frac{x-0.6}{x_T}} - 1\right)^2 \tanh^2(x-1.5)\left(\ln(1+x) - \ln 9\right)^3, \tag{88}$$

where x denotes the node voltage and $x_T \approx 0.02585$ V is the thermal voltage.

Equation (88) exhibits *three distinct multiple roots*:

$$x_1 = 0.6, \quad x_2 = 1.5, \quad x_3 = 8.0.$$

The numerical results reported in Tables 13 and 14 are obtained using randomly generated initial guesses listed in Appendix A Table A6 and implemented via Algorithm 1. Table 13 presents a consistency analysis for problem (88), showing that the proposed schemes achieve significantly smaller residual errors with fewer iterations compared to existing methods. In particular, the WUM$^{[*]}$ scheme exhibits a favorable balance among accuracy, CPU time, and memory usage, indicating efficient handling of root multiplicity without compromising numerical stability.

The consistency results reported in Table 13 for solving (88) further highlight the advantages of higher-order simultaneous schemes. While the lower-order methods WDK$^{[*]}$ and ZHM$^{[*]}$ converge reliably, they require a substantially larger number of iterations and greater computational resources to achieve moderate accuracy. Among the higher-order approaches, WUM$^{[*]}$ attains the lowest maximum error (0.23×10^{-53}) while requiring the fewest iterations and reduced CPU time. Although WKM$^{[*]}$ and MPM$^{[*]}$ also provide competitive accuracy, their higher computational cost and memory usage result in lower overall efficiency. Overall, the numerical results indicate that WUM$^{[*]}$ offers a favorable balance between accuracy and computational efficiency for simultaneously computing all roots of (88).

Table 13. Consistency analysis of the simultaneous schemes for solving (88).

Metric	Max-Error	CPU-Time	Mem-Usage	$[\pm, \times, \div]$	Iterations (k)	Max-σ_i^{n-1}
			Comparison with lower-order methods			
WDK$^{[*]}$	4.59×10^{-9}	1.087	45.876	975	6	3.645
ZHM$^{[*]}$	7.01×10^{-11}	2.097	55.853	1076	6	4.897
			Comparison with higher-order methods			
WKM$^{[*]}$	6.19×10^{-45} †	0.041	35.762	886	3	9.801
WUM$^{[*]}$	0.23×10^{-53} †	0.007	29.301	704	2	10.093
MPM$^{[*]}$	0.97×10^{-34} †	0.986	34.762	965	3	9.897
WWM$^{[*]}$	1.12×10^{-35} †	1.063	31.276	964	4	9.107
FLM$^{[*]}$	4.65×10^{-31} †	1.181	31.009	976	3	9.711

† All computations were performed using MATLAB variable precision arithmetic (VPA) with `digits` = 128, employing a numerical tolerance of 10^{-64}.

Table 14. Comparison of speedup ratio (ϕ_{sp}) and maximum error for existing and proposed schemes for solving (88).

Cores	WDK$^{[*]}$	ZHM$^{[*]}$	WKM$^{[*]}$	WUM$^{[*]}$	MPM$^{[*]}$	WWM$^{[*]}$	FLM$^{[*]}$
1	0.0453	0.0365	0.0046	0.0059	0.0143	0.0127	0.0113
2	0.0657	0.0509	0.0093	0.0107	0.0209	0.0183	0.0172
4	0.1650	0.0932	0.0208	0.0226	0.0298	0.0282	0.0264
Max-Error (all cores)	3.7×10^{-13}	0.3×10^{-15}	1.8×10^{-76} †	2.4×10^{-75} †	5.3×10^{-48} †	1.9×10^{-65} †	7.0×10^{-53} †

† All computations were performed using MATLAB variable precision arithmetic (VPA) with `digits` = 128, employing a numerical tolerance of 10^{-64}.

Table 14 summarizes the parallel performance of the schemes using MATLAB's `parfor` with 1, 2, and 4 computing cores. The results demonstrate that WUM$^{[*]}$ preserves high accuracy across all core configurations, while WKM$^{[*]}$ provides the best runtime efficiency. The consistently low error levels confirm the robustness and scalability of the proposed methods. Figure 8 illustrates the convergence behavior of the higher-order simultaneous schemes for Equation (88).

Figure 8 illustrate the convergence behavior of the considered methods, including the error evolution of all computed roots. The error plots indicate that the proposed method converges in fewer iterations and attains smaller residual errors than the classical approaches when computing all roots of (88) simultaneously.

Physical interpretation:

- x_1: Threshold for transistor or diode turn-on, representing the onset of conduction.
- x_2: Feedback or saturation midpoint, corresponding to a balanced operating state caused by mild clipping or internal feedback effects.
- x_3: Load or gain saturation limit, associated with the maximum voltage threshold determined by device constraints or circuit topology.

Nonlinear bias networks, Schmitt triggers, and unstable circuits are typical examples of systems that exhibit multiple equilibria. The *flat tangency* at each root presents a significant challenge for conventional iterative techniques, making this example well suited for evaluating solvers designed to efficiently handle multiple roots. Although still analytically tractable for convergence analysis, its non-polynomial structure realistically represents the behavior of nonlinear electronic devices.

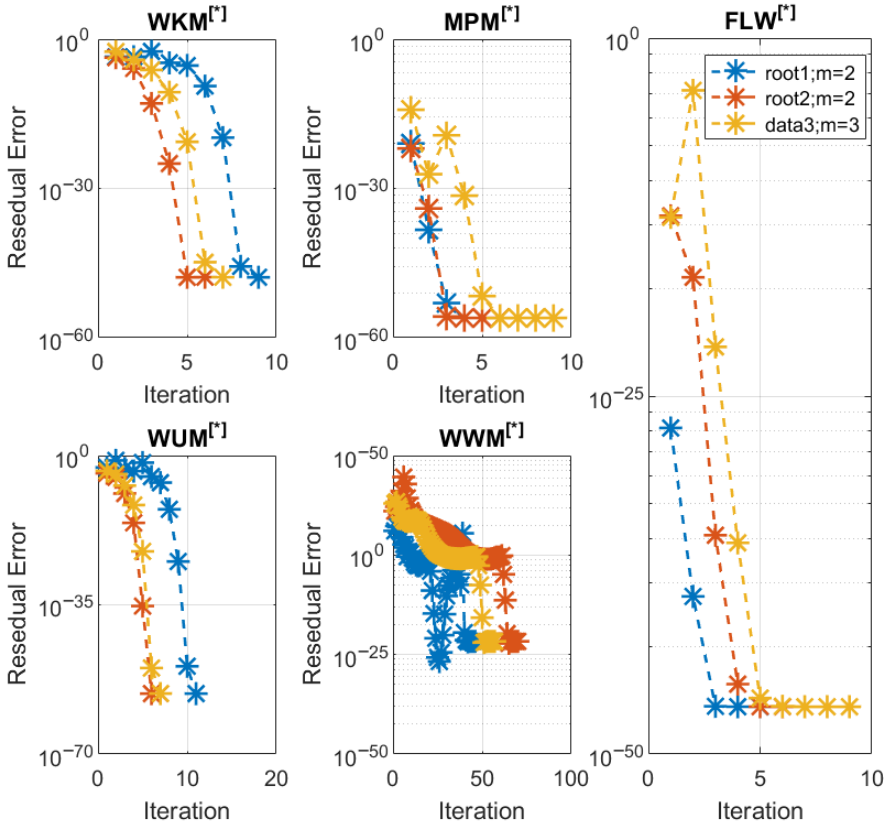


Figure 8. Error graph for the function (88) using the simultaneous schemes.

5. Conclusions

To efficiently compute all roots of polynomial equations with multiple or unknown multiplicities, this study introduces two families of two-step simultaneous algorithms, denoted as WKM$^{[*]}$ and WUM$^{[*]}$. Starting from a single-root framework, the proposed approach is extended to the multiple-root setting through a multiplicity-reducing transformation that removes the need for prior knowledge of the root multiplicity. By incorporating these corrections into a simultaneous iteration scheme, the resulting methods significantly improve upon the classical quadratic Weierstrass technique and achieve tenth-order convergence.

The theoretical analysis establishes fourth-order convergence for the modified single-root scheme and tenth-order convergence for the simultaneous formulations with both known and unknown multiplicities. Numerical experiments indicate that the proposed WKM$^{[*]}$ and WUM$^{[*]}$ methods exhibit improved performance compared with the established WDK$^{[*]}$, ZHM$^{[*]}$, MPM$^{[*]}$, WWM$^{[*]}$, and FLM$^{[*]}$ techniques in terms of residual error, CPU time, memory usage, and robustness under both random and distant initial guesses. Comprehensive numerical experiments, summarized in Tables 3–14, support these findings across a wide range of test cases, including biomedical engineering models, general polynomial equations, and benchmark polynomials.

For clarity, the main advantages of the proposed WKM$^{[*]}$ and WUM$^{[*]}$ schemes over the existing MPM$^{[*]}$, WWM$^{[*]}$, and FLM$^{[*]}$ methods are summarized as follows:

- The proposed schemes require reduced computational time across all tested core configurations, indicating improved efficiency in simultaneous computing environments (Tables 4–14).

- Both methods achieve high numerical accuracy with consistently small residual errors (Figures 3–8), including for problems involving multiple roots.
- In contrast to classical multiplicity-dependent approaches, the proposed schemes exhibit stable convergence without requiring prior knowledge of root multiplicities.
- The use of random complex initial guesses together with 128-digit variable-precision arithmetic results in robust and consistent convergence across all benchmark problems.
- When implemented using MATLAB's `parfor` construct, the schemes demonstrate favorable speedup ratios, reflecting good scalability on multicore architectures.
- Error analyses and efficiency indicators (Tables 1 and 2 and Figure 2) show that WKM$^{[*]}$ and WUM$^{[*]}$ achieve a more favorable balance between accuracy and computational cost than the compared methods.
- The proposed frameworks are flexible and may be extended to fractional-order systems and nonlinear biomedical models, broadening their potential applicability in scientific and engineering applications.

Future research will focus on extending the proposed WKM$^{[*]}$ and WUM$^{[*]}$ techniques to fractional-order and stochastic systems, which frequently arise in engineering and biomedical applications characterized by memory and uncertainty effects. Additional efforts will be directed toward the development of high-performance computational frameworks that incorporate GPU-based parallelism, distributed computing, and adaptive-precision arithmetic, with the aim of further improving numerical stability, scalability, and practical applicability. These developments are expected to facilitate efficient large-scale simulations, real-time parameter estimation, and data-driven model calibration for complex nonlinear dynamical systems.

Author Contributions: Conceptualization, M.S. and B.C.; methodology, M.S.; software, M.S.; validation, M.S.; formal analysis, B.C.; investigation, M.S.; resources, B.C.; writing—original draft preparation, M.S. and B.C.; writing—review and editing, B.C.; visualization, M.S. and B.C.; supervision, B.C.; project administration, B.C.; funding acquisition, B.C. All authors have read and agreed to the published version of the manuscript.

Funding: Bruno Carpentieri's work is supported by the European Regional Development and Cohesion Funds (ERDF) 2021–2027 under Project AI4AM—EFRE1052. He is a member of the *Gruppo Nazionale per il Calcolo Scientifico* (GNCS) of the *Istituto Nazionale di Alta Matematica* (INdAM), and this work was partially supported by INdAM-GNCS under the *Progetti di Ricerca 2024 program.*

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare that there are no conflicts of interest regarding the publication of this article.

Appendix A

In this appendix, we report the randomly generated initial guesses located far from the exact roots that are used in the numerical experiments. These initial values are generated using Algorithm 1, which constructs perturbed points on the complex unit circle and maps them into high-precision arithmetic in MATLAB. This initialization strategy ensures adequate separation of the starting points and helps prevent premature clustering, thereby improving the robustness and convergence of the simultaneous root-finding schemes.

Table A1 reports the far random initial guesses employed by the proposed schemes for solving (74).

Table A1. Random initial guess values for the test polynomial (74).

$x_1^{[0]}$	$x_2^{[0]}$	$x_3^{[0]}$	$x_4^{[0]}$	$x_5^{[0]}$	$x_6^{[0]}$	$x_7^{[0]}$	$x_8^{[0]}$
$-1.5+1.5i$	$0.7+2.0i$	$-1.8+0.1i$	$-1.0+1.6i$	$1.4-0.5i$	$2.2-1.2i$	$1.6+0.5i$	$-0.1-1.6i$
$-0.9-1.5i$	$-1.9-0.5i$	$-0.7-1.9i$	$-1.2-2.1i$	$0.8+1.7i$	$1.7+1.8i$	$1.8-0.0i$	$1.0+1.9i$
$0.9+1.3i$	$1.3-1.6i$	$-1.5-1.9i$	$-1.5+1.5i$	$0.6+1.6i$	$-1.7+0.8i$	$-1.9+0.2i$	$1.8+1.7i$
$0.9-1.9i$	$1.7-0.2i$	$-0.2-1.9i$	$-1.2+1.7i$	$-1.5-0.9i$	$1.4+1.1i$	$1.8-1.2i$	$1.3-1.2i$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table A2 lists the far random initial guesses used by the proposed schemes for solving (76).

Table A2. Random initial guess values for Polynomial (76).

$x_1^{[0]}$	$x_2^{[0]}$	$x_3^{[0]}$	$x_4^{[0]}$	$x_5^{[0]}$	$x_6^{[0]}$	$\cdots$
$1.1+1.9i$	$-1.4-1.4i$	$-1.8-0.9i$	$1.8+0.6i$	$1.5-0.7i$	$-0.3-2.0i$	$\cdots$
$2.4-0.2i$	$-1.7-0.4i$	$-0.5-1.7i$	$1.8-0.0i$	$-0.5+1.9i$	$-1.8+1.1i$	$\cdots$
$1.7-1.3i$	$0.6+1.9i$	$2.0+1.3i$	$-0.6+1.5i$	$-2.2+0.6i$	$1.8+1.3i$	$\cdots$
$-1.9+1.5i$	$1.9+0.8i$	$0.3-1.6i$	$-1.2+2.0i$	$-1.4-1.2i$	$-0.1-1.8i$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Table A3 lists the far random initial guesses used by the proposed schemes for solving (83).

Table A3. Random initial guess values for Polynomial (83).

$x_1^{[0]}$	$x_2^{[0]}$	$x_3^{[0]}$	$x_4^{[0]}$
$-1.1+1.5i$	$1.3+1.8i$	$-1.9-1.2i$	$-0.2+2.0i$
$-0.6-1.4i$	$-1.4+1.7i$	$-0.1-1.7i$	$-1.5+1.2i$
$-1.1+1.9i$	$-2.0+1.0i$	$-0.2+1.9i$	$0.5+1.4i$
$0.3-2.3i$	$-1.5+1.2i$	$0.9-2.2i$	$0.1-1.9i$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table A4 lists the far random initial guesses used by the proposed schemes for solving (85).

Table A4. Random initial guess values for Polynomial (85).

$x_1^{[0]}$	$x_2^{[0]}$	$x_3^{[0]}$	$x_4^{[0]}$	$x_5^{[0]}$	$x_6^{[0]}$
$-1.7+0.7i$	$0.9-1.5i$	$0.3-2.3i$	$1.2+2.1i$	$1.2-1.4i$	$2.2-0.1i$
$-0.0-2.0i$	$1.1-2.1i$	$0.5-2.0i$	$-0.7+1.5i$	$-1.7-0.4i$	$1.6+1.0i$
$1.0+1.3i$	$1.5+0.6i$	$1.3-1.8i$	$-1.9-0.7i$	$1.5-0.7i$	$-0.7-1.9i$
$-1.6-1.4i$	$2.3-0.1i$	$-2.3-0.4i$	$-2.5+0.3i$	$0.5-1.4i$	$0.3+2.3i$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table A5 lists the far random initial guesses used by the proposed schemes for solving (87).

Table A5. Random initial guess values for nonlinear function (87).

$x_1^{[0]}$	$x_2^{[0]}$
$-1.8-0.9i$	$1.4-2.0i$
$0.0+2.2i$	$-1.0-1.8i$
$-0.8-1.3i$	$-0.3-2.1i$
$0.3-2.4i$	$-2.2-1.2i$
$\vdots$	$\vdots$

Table A6 lists the far random initial guesses used by the proposed schemes for solving (88).

Table A6. Random initial guess values for nonlinear function (88).

$x_1^{[0]}$	$x_2^{[0]}$	$x_3^{[0]}$
$1.8+1.7i$	$1.4-1.6i$	$-1.5+0.1i$
$-1.6-1.7i$	$1.7+0.3i$	$-1.5-1.4i$
$0.7+1.7i$	$1.1+1.1i$	$0.5+1.9i$
$-1.1-1.2i$	$-0.3+1.7i$	$-1.5-0.4i$
$\vdots$	$\vdots$	$\vdots$

References

1. Rhoads, J.F.; Shaw, S.W.; Turner, K.L. Nonlinear dynamics and its applications in micro-and nanoresonators. *J. Dyn. Sys. Meas. Control* **2010**, *132*, 034001. [CrossRef]
2. Cristadoro, G.; Knight, G.; Degli Esposti, M. Follow the fugitive: An application of the method of images to open systems. *J. Phys. A Math. Theor.* **2013**, *46*, 272001. [CrossRef]
3. Cruz, D.A.; Kemp, M.L. Hybrid computational modeling methods for systems biology. *Prog. Biomed. Eng.* **2021**, *4*, 012002. [CrossRef] [PubMed]
4. Li, Q.; Deng, Z.; Ahadi, A.; Chu, K.; Yan, J.; Huang, K.; Sun, Q. Giant-Thermal-Expansion Martensitic Alloys for Ultrahigh-Efficiency Solid-State Heat-Pumping. 2024. Available online: https://www.researchsquare.com/article/rs-4856090/v1 (accessed on 17 December 2025).
5. Li, J.; Kuang, Y. Analysis of a model of the glucose-insulin regulatory system with two delays. *SIAM J. Appl. Math.* **2007**, *67*, 757–776. [CrossRef]
6. Alhazmi, M.; Saber, S. Application of a fractal fractional derivative with a power-law kernel to the glucose–insulin interaction system based on Newton's interpolation polynomials. *Fractals* **2025**, 2540201. . [CrossRef]
7. Slepkov, V.A.; Sukhovolsky, V.G.; Khlebopros, R.G. Population dynamics in modeling tumor growth. *Biophysics* **2007**, *52*, 426–431. [CrossRef]
8. Piccini, J.P.; Russo, A.M.; Sharma, P.S.; Kron, J.; Tzou, W.; Sauer, W.; Park, D.S. Advances in cardiac electrophysiology. *Circ. Arrhythmia Electrophysiol.* **2022**, *15*, e009911. [CrossRef] [PubMed]
9. Siepmann, J.; Siepmann, F. Mathematical modeling of drug delivery. *Int. J. Pharm.* **2008**, *364*, 328–343. [CrossRef]
10. Aaboe, A.; Boardman, J. Babylonian mathematics, astrology, and astronomy. In *The Cambridge Ancient History*;Vol. 3, Pt. 2: The Assyrian and Babylonian Empires and Other States of the Near East; Cambridge University Press: Cambridge, UK, 1991; pp. 276–292. [CrossRef]
11. Allen, J. *An Invitation to Mathematical Physics and Its History*; Springer Nature: Cham, Switzerland, 2020. [CrossRef]
12. Cai, T. *A Brief History of Mathematics*; Springer Nature: Cham, Switzerland, 2023. [CrossRef]
13. Chahal, J.S. Solution of the cubic. *Resonance* **2006**, *11*, 53–61. [CrossRef]
14. Gomez Samper, P. Rafael Bombelli's Posthumous Books IV and V of L'Algebra: Algebraic-Geometrical Relations in 16th-Century Italy. Master's Thesis, Utrecht University, Utrecht, Netherlands, 2025. Available online: https://studenttheses.uu.nl/handle/20.500.12932/49074 (accessed on 11 December 2025).

15. Oaks, J.A. François Viète's revolution in algebra. *Arch. Hist. Exact Sci.* **2018**, *72*, 245–302. [CrossRef]
16. Kollerstrom, N. Thomas Simpson and 'Newton's method of approximation': An enduring myth. *Br. J. Hist. Sci.* **1992**, *25*, 347–354. [CrossRef]
17. Hamburg, R.R. The theory of equations in the 18th century: The work of Joseph Lagrange. *Arch. Hist. Exact Sci.* **1976**, *16*, 17–36. [CrossRef]
18. Collyer, A. A.; Pathan, A. The wonder of Horner's method. *Math. Gaz.* **2003**, *87*, 230–242. [CrossRef]
19. Mezo, I. *The Lambert W Function: Its Generalizations and Applications*; Chapman and Hall/CRC: Boca Raton, FL, USA, 2022. [CrossRef]
20. Aydi, H.; Felhi, A.; Ayoob, I.; Mlaiki, N. On multivalued contractions via θ-hyperbolic sine distance functions. *Eur. J. Pure Appl. Math.* **2025**, *18*, 6015–6015. [CrossRef]
21. Cajori, F. Historical note on the Newton-Raphson method of approximation. *Am. Math. Mon.* **1911**, *18*, 29–32. [CrossRef]
22. Halley, E. Methodus nova accurata & facilis inveniendi radices aequationum quarumcumque generaliter, sine praevia reductione. *Philos. Trans.* **1694**, *18*, 136–148. [CrossRef]
23. Peyret, R. Chebyshev method. In *Spectral Methods for Incompressible Viscous Flow*; Springer: New York, NY, USA, 2002; pp. 39–100. [CrossRef]
24. Kanwar, V.; Sharma, K.K.; Behl, R. A new family of Schröder's method and its variants based on power means for multiple roots of nonlinear equations. *Int. J. Math. Educ. Sci. Technol.* **2010**, *41*, 558–565. [CrossRef]
25. Ostrowski, A.M. *Solution of Equations and Systems of Equations*; Series in Pure and Applied Mathematics; Academic Press: New York, NY, USA, 1966.
26. Jarratt, P. Some Fourth Order Multipoint Iterative Processes for Solving Equations. *Math. Comput.* **1966**, *20*, 434–437. [CrossRef]
27. Abad, M.F.; Cordero, A.; Torregrosa, J.R. Fourth-and Fifth-Order Methods for Solving Nonlinear Systems of Equations: An Application to the Global Positioning System. *Abstr. Appl. Anal.* **2013**, *2013*, 586708. [CrossRef]
28. King, R.F. A family of fourth-order methods for nonlinear equations. *SIAM J. Numer. Anal.* **1973**, *10*, 876–879. [CrossRef]
29. Weierstrass, K. Neuer Beweis des Satzes, dass jede ganze rationale Function einer Veränderlichen dargestellt werden kann als ein Product aus linearen Functionen derselben Veränderlichen. In *Sitzungsberichte Königlich Preussischen Akademie der Wissenschaften zu Berlin*; Prussian Academy of Sciences: Berlin, Germany, 1854 ; pp. 1085–1101.
30. Durand, É. *Solutions Numériques des Équations Algébriques. Tome I: Équation du Type $F(\chi) = 0$; Racines d'un Polynôme*; Masson: Paris, France, 1960. Available online: https://cir.nii.ac.jp/crid/1570009749930308096 (accessed on 17 December 2025).
31. Kerner, I.O. Ein Gesamtschrittverfahren zur Berechnung der Nullstellen von Polynomen. *Numer. Math.* **1966**, *8*, 290–294. [CrossRef]
32. Aberth, O. Iteration methods for finding all zeros of a polynomial simultaneously. *Math. Comp.* **1973**, *27*, 339–344. . [CrossRef]
33. Börsch-Supan, W. Residuenabschätzung für Polynom-Nullstellen mittels Lagrange-Interpolation. *Numer. Math.* **1970**, *14*, 287–296. [CrossRef]
34. Cordero, A.; Torregrosa, J.R.; Triguero-Navarro, P. Jacobian-Free Vectorial Iterative Scheme to Find Simple Several Solutions Simultaneously. *Math. Methods Appl. Sci.* **2025**, *48*, 5718–5730. [CrossRef]
35. Nourein, A.W.M. An improvement on two iteration methods for simultaneous determination of the zeros of a polynomial. *Int. J. Comput. Math.* **1977**, *6*, 241–252. [CrossRef]
36. Dongarra, J.; Grigori, L.; Higham, N.J. Numerical algorithms for high-performance computational science. *Philos. Trans. R. Soc. A* **2020**, *378*, 20190066. [CrossRef]
37. Pernet, C. High Performance and Reliable Algebraic Computing. Doctoral Dissertation, Université Joseph Fourier, Grenoble, France, 2014. Available online: https://theses.hal.science/tel-01094212 (accessed on 3 December 2025).
38. Emeliyanenko, P. High-performance polynomial GCD computations on graphics processors. In Proceedings of the 2011 International Conference on High Performance Computing & Simulation, Istanbul, Turkey, 4–8 July 2011; IEEE: Piscataway, NJ, USA; pp. 215–224. [CrossRef]
39. Cordero, A.; Neta, B.; Torregrosa, J.R. Memorizing Schröder's method as an efficient strategy for estimating roots of unknown multiplicity. *Mathematics* **2021**, *9*, 2570. [CrossRef]
40. Behl, R.; Gonzalez, D.; Maroju, P.; Motsa, S.S. An optimal and efficient general eighth-order derivative-free scheme for simple roots. *J. Comput. Appl. Math.* **2018**, *330*, 666–675.. [CrossRef]
41. Neta, B. On a fifth-order method for multiple roots of nonlinear equations. *Symmetry* **2023**, *15*, 1694. [CrossRef]
42. Chicharro, F.I.; Contreras, R.A.; Garrido, N. A family of multiple-root finding iterative methods based on weight functions. *Mathematics* **2020**, *8*, 2194. [CrossRef]
43. Chun, C.; Kim, Y.-I. Several new third-order iterative methods for solving nonlinear equations. *Acta Appl. Math.* **2010**, *109*, 1053–1063. [CrossRef]
44. Traub, J.F. *Iterative Methods for the Solution of Equations*; American Mathematical Society: Providence, RI, USA, 1982; Volume 312. Available online: https://bookstore.ams.org/chel-312 (accessed on 9 December 2025).

45. Jarratt, P. Some efficient fourth-order multipoint methods for solving equations. *BIT Numer. Math.* **1969**, *9*, 119–124. [CrossRef]
46. Rafiq, N.; Akram, S.; Mir, N.A.; Shams, M. Study of dynamical behavior and stability of iterative methods for nonlinear equation with applications in engineering. *Math. Probl. Eng.* **2020**, *2020*, 3524324. [CrossRef]
47. Khattri, S.K.; Abbasbandy, S. Optimal fourth-order family of iterative methods. *Mat. Vesn.* **2011**, *63*, 67–72. Available online: http://eudml.org/doc/252586 (accessed on 14 December 2025).
48. Zein, A. A general family of fifth-order iterative methods for solving nonlinear equations. *Eur. J. Pure Appl. Math.* **2023**, *16*, 2323–2347. [CrossRef]
49. Kelley, C.T. *Iterative Methods for Linear and Nonlinear Equations*; Frontiers in Applied Mathematics; SIAM: Philadelphia, PA, USA, 1995; Volume 16. [CrossRef]
50. Singh, S.; Gupta, D.K. Iterative methods of higher order for nonlinear equations. *Vietnam J. Math.* **2016**, *44*, 387–398. [CrossRef]
51. Rall, L.B. Convergence of the Newton process to multiple solutions. *Numer. Math.* **1966**, *9*, 23–37. [CrossRef]
52. Chicharro, F.I.; Garrido, N.; Jerezano, J.H.; Pérez-Palau, D. Family of fourth-order optimal classes for solving multiple-root nonlinear equations. *J. Math. Chem.* **2023**, *61*, 736–760. [CrossRef]
53. Shengguo, L.; Xiangke, L.; Lizhi, C. A new fourth-order iterative method for finding multiple roots of nonlinear equations. *Appl. Math. Comput.* **2009**, *215*, 1288–1292. [CrossRef]
54. Zafar, F.; Cordero, A.; Torregrosa, J.R. Stability analysis of a family of optimal fourth-order methods for multiple roots. *Numer. Algorithms* **2019**, *81*, 947–981. [CrossRef]
55. Lee, M.Y.; Kim, Y.I.; Magreñán, Á.A. On the dynamics of a triparametric family of optimal fourth-order multiple-zero finders with a weight function of the principal mth root of a function-to function ratio. *Appl. Math. Comput.* **2017**, *315*, 564–590... [CrossRef]
56. King, R.F. A secant method for multiple roots. *BIT Numer. Math.* **1977**, *17*, 321–328. [CrossRef]
57. Parida, P.K.; Gupta, D.K. An improved method for finding multiple roots and its multiplicity of nonlinear equations in $\mathbb{R}$. *Appl. Math. Comput.* **2008**, *202*, 498–503. [CrossRef]
58. Kioustelidis, J.B. A derivative-free transformation preserving the order of convergence of iteration methods in case of multiple zeros. *Numer. Math.* **1979**, *33*, 385–389. [CrossRef]
59. Petković, I.; Neta, B. On an application of symbolic computation and computer graphics to root-finders: The case of multiple roots of unknown multiplicity. *J. Comput. Appl. Math.* **2016**, *308*, 215–230. [CrossRef]
60. Ahmad, F.; Serra-Capizzano, S.; Ullah, M.Z.; Al-Fhaid, A.S. A family of iterative methods for solving systems of nonlinear equations having unknown multiplicity. *Algorithms* **2015**, *9*, 5. [CrossRef]
61. Shah, F.A.; Waseem, M. Quadrature based innovative techniques concerning nonlinear equations having unknown multiplicity. Examples and Counterexamples. *Math. Probl. Eng.* **2024**, *6*, 100150. [CrossRef]
62. Akram, S.; Akram, F.; Khalid, L. *Iterative Techniques for Nonlinear Equations: Addressing Multiple Roots with Unknown Multiplicity*; IntechOpen: London, UK, 2025. [CrossRef]
63. Ahmad, F.; Bhutta, T.A.; Shoaib, U.; Zaka Ullah, M.; Alshomrani, A.S.; Ahmad, S.; Ahmad, S. A preconditioned iterative method for solving systems of nonlinear equations having unknown multiplicity. *Algorithms* **2017**, *10*, 17. [CrossRef]
64. Kalitkin, N.N.; Poshivailo, I.P. Determining the multiplicity of a root of a nonlinear algebraic equation. *Comput. Math. Math. Phys.* **2008**, *48*, 1113–1118. [CrossRef]
65. Milovanovic, G.V.; Petkovic, M.S. On computational efficiency of the iterative methods for the simultaneous approximation of polynomial zeros. *ACM Trans. Math. Softw. (TOMS)* **1986**, *12*, 295–306. [CrossRef]
66. Petković, M.S.; Petković, L.D.; Džunić, J. On an efficient simultaneous method for finding polynomial zeros. *Appl. Math. Lett.* **2014**, *28*, 60–65. [CrossRef]
67. Wang, D.R.; Wu, Y.J. Some modifications of the parallel Halley iteration method and their convergence. *Computing* **1987**, *38*, 75–87. [CrossRef]
68. Petković, M.S.; Rančić, L.; Milošević, M. The improved Farmer–Loizou method for finding polynomial zeros. *Int. J. Comput. Math.* **2012**, *89*, 499–509. [CrossRef]
69. Zhang, X.; Peng, H.; Hu, G. A high order iteration formula for the simultaneous inclusion of polynomial zeros. *Appl. Math. Comput.* **2006**, *179*, 545–552. [CrossRef]
70. Gautschi, W. *Numerical Analysis*, 2nd ed.; Birkhäuser: Basel, Switzerland, 2012. [CrossRef]
71. Mignotte, M. Polynômes et lemme de Siegel. In *Cinquante Ans de Polynômes Fifty Years of Polynomials: Proceedings of the Conference Held in Honour of Alain Durand at the Institut Henri Poincaré, Paris, France, 26–27 May 1988*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 160–166. [CrossRef]
72. Hu, S.; Dunlavey, M.; Guzy, S.; Teuscher, N. A distributed delay approach for modeling delayed outcomes in pharmacokinetics and pharmacodynamics studies. *J. Pharmacokinet. Pharmacodyn.* **2018**, *45*, 285–308. [CrossRef]
73. Kuby, S.A. *A Study of Enzymes: Enzyme Catalysts, Kinetics, and Substrate Binding*; CRC Press: Boca Raton, FL, USA, 2019. [CrossRef]

74. Ben, F.; Fernando, J.; Ou, J.Y.; Dupré, C.; Ollier, E.; Hassani, F.A.; Tsuchiya, Y. Characterisation and Modelling of Nonlinear Resonance Behaviour on Very-High-Frequency Silicon Nanoelectromechanical Resonators. *Micro Nano Eng.* **2023**, *19*, 100212. [CrossRef]
75. Guo, M.; Zhang, Q.; Zeng, D.D. An Interpretable Deep Learning-Based Model for Decision-Making through Piecewise Linear Approximation. *ACM Trans. Knowl. Discov. Data* **2025**, *19*, 56 . [CrossRef]