A Highly Efficient Computer Method for Solving Polynomial Equations Appearing in Engineering Problems

Naila Rafiq 1, Mudassir Shams 2, Nazir Ahmad Mir 1, and Yaé Ulrich Gaba 3,4,5

1 Department of Mathematics, National University of Modern Languages, Islamabad, Pakistan
2 Department of Mathematics and Statistics, Riphah International University I-14, Islamabad 44000, Pakistan
3 Quantum Leap Africa (QLA), AIMS Rwanda Centre, Remera Sector KN 3, Kigali, Rwanda
4 Institut de Mathématiques et de Sciences Physiques (IMSP/UAC), Laboratoire de Topologie Fondamentale, Computationnelle et Leurs Applications (Lab-ToFoCApp), BP 613 Porto-Novo, Benin
5 African Center for Advanced Studies, P.O. Box 4477, Yaounde, Cameroon

Correspondence should be addressed to Yaé Ulrich Gaba; yaeulrich.gaba@gmail.com

Received 10 August 2021; Revised 12 November 2021; Accepted 13 November 2021; Published 22 December 2021

Copyright © 2021 Naila Rafiq et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A highly efficient two-step simultaneous iterative computer method is established here for solving polynomial equations. A suitable special type of correction helps us to achieve a very high computational efficiency as compared to the existing methods so far in the literature. Analysis of simultaneous scheme proves that its convergence order is 14. Residual graphs are also provided to demonstrate the efficiency and performance of the newly constructed simultaneous computer method in comparison with the methods given in the literature. In the end, some engineering problems and some higher degree complex polynomials are solved numerically to validate its numerical performance.

1. Introduction

Determining the roots of polynomial equations is among the oldest problems in mathematics, whereas the polynomial equations have a wide range of applications in science and engineering. For example, aerospace engineers may use polynomials to determine acceleration of a rocket or jet or even stability of an aeroplane and mechanical engineers use polynomials to design engines and machines. Simultaneous methods are very popular as compared to the methods for individual finding of the roots. These methods have a wider region of convergence, are more stable, and can be implemented for parallel computing. More details on simultaneous methods, their convergence properties, computational efficiency, and parallel implementation may be found in the works of Cosnard et al. [1], Kanno et al. [2], Proinov et al. [3], Sendov et al. [4] Ikhile [5], Mir at al. [6], Wahab et al. [7], Cholakov [8], Proinov and Ivanov [9], Iliev [10], and Kyncheva [11]. Nowadays, mathematicians are working on iterative methods for finding all the zeros of polynomial simultaneously (see [12–18] and references therein).

The main objective of this paper is to develop simultaneous method which not only has a higher convergence order but also is more efficient as compared to existing methods. A very high computational efficiency for the newly constructed scheme for finding distinct as well as multiple roots is achieved by using a suitable corrections [19] which enable us to achieve fourteenth-order convergence with minimal number of functional evaluations in each step. So far among the higher order simultaneous methods, only the Midrog Petkovic method [20] of order ten and the Gargantini–Farmer–Loizou method of $2N + 1$ convergence order (where $N$ is positive integer) [21–24] exist in the literature. Consider nonlinear polynomial equation of degree $m$: 
where \( \sigma_i \) is the multiplicity of the root \( \zeta_i \) of equation (1). We would like to convert (2) into the simultaneous method for estimating all roots of (1). We use fifth-order Thukral et al. method [19] as a correction to increase the efficiency and convergence order requiring no additional evaluations of the function:

\[
\begin{align*}
&\begin{cases}
  y^{(k)} = t^{(k)} - \sigma \frac{f(t^{(k)})}{f'(t^{(k)})}, \\
  u^{(k)} = y^{(k)} - \sigma \frac{f(y^{(k)})}{f'(y^{(k)})},
\end{cases} \\
&z^{(k)} = y^{(k)} - \sigma \left( 1 + \left( \frac{f(y^{(k)})}{f(t^{(k)})} \right)^{2/\sigma} \right) \frac{f(y^{(k)})}{f'(y^{(k)})}.
\end{align*}
\]

Suppose equation (1) has \( m \) distinct roots; then,

\[
\begin{align*}
  f(t) &= \prod_{i=1}^{m} (t - \zeta_i), \\
  f'(t) &= \sum_{k=1}^{m} \prod_{\substack{i=1 \atop i \neq k}}^{m} (t - \zeta_i).
\end{align*}
\]

This implies

\[
\frac{f(t_i)}{f'(t_i)} = \sum_{j=1}^{m} \frac{1}{(t - t_j)},
\]

or

\[
\frac{f(t_i)}{f'(t_i)} = \frac{1}{(1/(t - t_i)) - \sum_{j \neq i}^{m} (1/(t - t_j))},
\]

or

\[
\frac{1}{t - t_i} = \frac{f'(t_i)}{f(t_i)} - \frac{m}{\sum_{j \neq i}^{m} (1/(t - t_j))},
\]

or

\[
t - t_i = \frac{1}{(f'(t_i)/f(t_i)) - \sum_{j \neq i}^{m} (1/(t - t_j))},
\]

This gives

\[
t - t_i = \frac{1}{(1/(N_i(t_i))) - \sum_{j \neq i}^{m} (1/(t - t_j))},
\]

where

\[
N_i(t_i) = \frac{f'(t_i)}{f(t_i)}.
\]

or

\[
f(t_i) = \frac{1}{(1/(N_i(t_i))) - \sum_{j \neq i}^{m} (1/(t - t_j))},
\]

For multiple roots, equation (7) can be written as

\[
\frac{f(t_i)}{f'(t_i)} = \frac{\sigma_i}{(\sigma_i/(N_i(t_i))) - \sum_{j \neq i}^{m} (\sigma_j/(t - t_j))},
\]

where \( \zeta_1, \ldots, \zeta_n \) are now multiple roots of respective unknown multiplicities \( \sigma_1, \ldots, \sigma_n, (\sigma_1 + \cdots + \sigma_n = m) \).

Replacing \( t_j \) by \( z_j \) in (8), we have

\[
\frac{f(t_j)}{f'(t_j)} = \frac{\sigma_i}{(\sigma_i/(N_i(t_j))) - \sum_{j \neq i}^{m} (\sigma_j/(t - z_j))},
\]

where
\[ z_j = y_j - \sigma_j \left( 1 + \frac{f(y_j)}{f(t_j)} (2\sigma_i) \right) \frac{f(y_j)}{f'(y_j)}, \]  
\[ y_j = t_j - \sigma_j \frac{f(t_j)}{f'(t_j)}. \]  
(10)

Using (9) in (2), we have
\[
\begin{align*}
\sigma_j & = \frac{\sigma_i}{(\sigma_i)/N(t_j)} - N(t_j) \sum_{j \neq i} \sigma_i \left( \frac{t_j - z_j}{t_j - z_j} \right) \\
\sigma_i & = \frac{\sigma_i}{(\sigma_i)/N(t_j)} - N(t_j) \sum_{j \neq i} \sigma_i \left( \frac{y_j - y_j}{y_j - y_j} \right) \\
\end{align*}
\]  
(11)

where \( z_j^{(k)} = y_j^{(k)} - \sigma_j \left( 1 + \frac{f(y_j^{(k)})}{f(t_j^{(k)})} \right) \frac{f(y_j^{(k)})}{f'(y_j^{(k)})} \) and \( y_j^{(k)} = t_j^{(k)} - \sigma_j \left( \frac{f(t_j^{(k)})}{f'(t_j^{(k)})} \right) \).

Thus, we have constructed a new simultaneous method (11) abbreviated as MMN14M for calculating all multiple roots of polynomial equation (1). The simultaneous method (11) requires two evaluations of the function and two evaluations of the first derivative. For multiplicity unity, i.e., \( \sigma_j = 1, i = 1, \ldots, n \), we use method (11) for determining all the distinct roots of equation (1) and abbreviate it as MMN14D.

2.1. Convergence Analysis. In this section, we discuss the convergence analysis of the two-step simultaneous method (11) which is given in the form of the following theorem.

**Theorem 1.** Let \( \zeta_1, \ldots, \zeta_n \) be the roots of equation (1) with multiplicity \( \sigma_1, \ldots, \sigma_n \) \( \sigma_1 + \cdots + \sigma_n = m \). If \( t_1^{(0)}, \ldots, t_n^{(0)} \) are the initial approximations of the roots, respectively, and sufficiently close to actual roots, the order of convergence of method (11) equals fourteen.

**Proof.** Let
\[
\begin{align*}
\epsilon_i &= t_i^{(k)} - \zeta_i, \\
\epsilon_i' &= y_i^{(k)} - \zeta_i, \\
\epsilon_i'' &= u_i^{(k)} - \zeta_i, \\
\end{align*}
\]  
(12)
be the errors in \( t_i^{(k)} \), \( y_i^{(k)} \), and \( u_i^{(k)} \) approximations, respectively. Consider the first step of (11):
\[ y_i^{(k)} = t_i^{(k)} - \frac{\sigma_i}{(\sigma_i)/N(t_i^{(k)})} - N(t_i^{(k)}) \sum_{j \neq i} \sigma_i \left( \frac{t_i^{(k)} - z_j^{(k)}}{t_i^{(k)} - z_j^{(k)}} \right) \]  
(13)
where
\[ N(t_i^{(k)}) = \frac{f(t_i^{(k)})}{f'(t_i^{(k)})} \]  
(14)

Then, obviously, for distinct roots,
\[
\frac{1}{N(t_i^{(k)})} = \frac{f'(t_i^{(k)})}{f(t_i^{(k)})} = \sum_{j \neq i} \frac{1}{(t_i^{(k)} - \zeta_j)} = \frac{1}{(t_i^{(k)} - \zeta_i)} + \sum_{j \neq i} \frac{1}{(t_i^{(k)} - \zeta_j)}. 
\]  
(15)

Thus, for multiple roots, we have, from (11),
\[
\begin{align*}
y_i^{(k)} &= t_i^{(k)} - \frac{\sigma_i}{N(t_i^{(k)})} \sum_{j \neq i} \sigma_i \left( \frac{t_i^{(k)} - z_j^{(k)}}{t_i^{(k)} - z_j^{(k)}} \right) \\
y_i^{(k)} - \zeta_i &= t_i^{(k)} - \zeta_i - \frac{\sigma_i}{N(t_i^{(k)})} \sum_{j \neq i} \sigma_i \left( \frac{t_i^{(k)} - z_j^{(k)}}{t_i^{(k)} - z_j^{(k)}} \right) \\
\epsilon_i' &= \epsilon_i - \frac{\sigma_i}{\sigma_i + \sum_{j \neq i} \sigma_j \left( \frac{t_i^{(k)} - z_j^{(k)}}{t_i^{(k)} - z_j^{(k)}} \right)} \\
\epsilon_i'' &= \epsilon_i - \frac{\sigma_i}{\sigma_i + \sum_{j \neq i} E_j \epsilon_j^*} 
\end{align*}
\]  
(16)
where \((Z^{(k)}_j - \zeta_j) = \epsilon_j^5\) [19] and \(E_i = (-\sigma_j) / (t_i^{(k)})\).

Thus,

\[
\epsilon_i' = \frac{\epsilon_i^2 \sum_{j=1}^n E_i \epsilon_j^5}{\sigma_i + \epsilon_i \sum_{j=1}^n E_i \epsilon_j^5}
\]  

(17)

From the second equation of (11),

\[
\frac{u_i^{(k)}}{u_i^{(k)} - \zeta_i} = \frac{\sigma_i}{\sigma_i / N(y_i^{(k)}) - \sum_{j=1}^n (\sigma_i / (y_i^{(k)} - y_j^{(k)}))}
\]

(19)

This implies

\[
\epsilon_i'' = \epsilon_i' - \frac{\sigma_i}{\sigma_i / (\sum_{j=1}^n (\sigma_j / (y_i^{(k)} - y_j^{(k)))) - \sum_{j=1}^n (\sigma_j / (y_i^{(k)} - y_j^{(k))))} \\
= \epsilon_i' - \frac{\sigma_i \epsilon_i'}{\sigma_i + \epsilon_i \sum_{j=1}^n (\sigma_j / (\sum_{j=1}^n (\sigma_j / (y_i^{(k)} - y_j^{(k)))) - \sum_{j=1}^n (\sigma_j / (y_i^{(k)} - y_j^{(k)))))}
\]

(20)

where \(F_i = \frac{-\sigma_j}{(y_i^{(k)} - \zeta_i)(y_i^{(k)} - y_j^{(k)})}\)

This implies

\[
\epsilon_i'' = \epsilon_i' - \frac{\sigma_i \epsilon_i'}{\sigma_i + \epsilon_i \sum_{j=1}^n \epsilon_j F_i - \epsilon_i \alpha}
\]

(21)

Since, from (18), \(\epsilon_i' = O(\epsilon)^7\), thus,
\[ \varepsilon''' = O(\varepsilon^2), \]
\[ \varepsilon'''' = O(\varepsilon^{14}), \] (22)

which shows convergence order of simultaneous iterative scheme (11) is fourteen. Hence, the theorem is proved. The above results are equally valid for complex polynomial by performing real arithmetic. Numerical Examples 4 and 5 for complex polynomials are provided to verify its validity.

### 3. Computational Aspect

Here, we compare the computational efficiency and convergence behaviour of our new fourteenth-order method MMN14M (11) with the Midrog Petkovic method [20] of order 10 and the Gargantini–Farmer–Loizou method [21–24] of order 15 (abbreviated as GFLM15M for multiple and GFLM15D for distinct roots). As presented in [20], the efficiency index given by

\[ EF(m) = \frac{\log r}{d} \] (23)

where \( d \) is the computational cost and \( r \) is the order of convergence of the iterative method. We use arithmetic operation per iteration with certain weight depending on the execution time of operation to evaluate the computational cost \( d \). The weights used for division, multiplication, and addition plus subtraction are \( w_d, w_m, \) and \( w_a \), respectively. For a given polynomial of degree \( m \), the number of division, multiplication, addition, and subtraction per iteration for all roots is denoted by \( A_{SM}, M_m, \) and \( D_m \). The cost of computation can be calculated as

\[ d = d(m) = w_d A_{SM} + w_m M_m + w_a D_m. \] (24)

Thus, (23) becomes

\[ EF(m) = \frac{\log r}{w_d A_{SM} + w_m M_m + w_a D_m}. \] (25)

Apply (25) and data given in Table 1, we find the percentage ratio \( \rho ((11), (X)) \) [20] given by

\[ (a) \rho ((11), (X)) = \left( \frac{EF(11)}{EF(X)} - 1 \right) \times 100 \text{ (in percent),} \]
\[ (b) \rho ((X), (11)) = \left( \frac{EF(X)}{EF(11)} - 1 \right) \times 100 \text{ (in percent),} \] (26)

where \( X \) and (11) are the Petkovic method (abbreviated as PJM10), GFLM15M, and our new method MMN14M, respectively. These ratios are graphically displayed in Figure 1(a)–1(d). It is evident from Figure 1(a)–1(d) that the new method (11) is more efficient as compared to the PJM10 and GFLM15M methods.

### 4. Numerical Results

Here, some numerical examples are considered in order to demonstrate the performance of our family of two-step fourteenth-order simultaneous methods, namely, MMN14D (for multiplicity unity) and MMN14M (for multiple roots) (11). We compare our family of methods with J. Džunic, M. S. Petkovic, and L. D. Petkovic [20] method of order ten for distinct roots (abbreviated as the PJM10 method) and with the Gargantini–Farmer–Loizou method (GFLM15D and GFLM15M) of order 15, respectively. All the computations are performed using Maple-18 with 64 digits’ floating point arithmetic. We take \( \varepsilon = 10^{-30} \) as a tolerance and use the following stopping criteria for estimating the roots:

\[ e_i^{(k)} = \left| f(i_i^{(k+1)}) \right| < \varepsilon, \] (27)

where \( e_i^{(k)} \) represents the absolute error of function values.

Numerical tests’ examples from [6, 17, 20, 33] are taken and compared on the same number of iterations and provided in Tables 2–15. In all the tables, \( n \) represents the number of iterations and CPU represents execution time in seconds. All the numerical calculations are performed using maple-18 on the computer (Processor Intel(R) Core(TM)i3-3110m CPU@2.4GHz) with 64-bit operating system. Figures 2–11 show the residue falls of the methods MMN14D, MMN14M, PJM10, GFLM15D, and GFLM15M for Examples 1–9. The residual falls show that the methods MMN14D, MMN14M are more efficient as compared to PJM10, GFLM15D, and GFLM15M methods. We observe that numerical results of the methods MMN14M and MMN14D are better than PJM10, GFLM15D, and GFLM15M methods in terms of absolute errors and CPU time (Algorithm 1).

**Example 1** (car stability). Application in mechanical engineering.

The design of a car suspension system requires to be balanced for getting good comfort and stability for all driving conditions and speeds. The following equations must be satisfied for stability of a design of a car which has good comfort on rough roads:

\[ \left( \frac{\omega}{\rho} \right)^4 - 1.9404 \times \left( \frac{\omega}{\rho} \right)^2 + 0.75 = 0. \] (28)

Let

\[ \frac{\omega}{\rho} = t. \] (29)

Then, we get the following polynomial equation:
Step 1: for given initial estimates \( t_i^{(0)} (i = 1, 2, \ldots, n) \), tolerance \( \epsilon > 0 \), and iterations \( p \), set \( k = 0 \).

Step 2: calculate
\[
y_j^{(k)} = t_j^{(k)} - \sigma_j (f(t_j^{(k)})/f'(t_j^{(k)})) \quad \text{and} \quad \sigma_j \text{ is the multiplicity of actual multiple roots } \zeta_j.
\]

\[
z_j^{(k)} = y_j^{(k)} - \sigma_j (1 + (f(y_j^{(k)})/f'(y_j^{(k)})) (f(y_j^{(k)})/f'(y_j^{(k)})))
\]

update \( y_i^{(k+1)} = (\sum_{j=1}^{n} (\sigma_j/((\sigma_j/N_i(t_i^{(k)}))) - (\sum_{j=1}^{n} (\sigma_j/((\sigma_j/N_i(t_i^{(k)}))) - y_j^{(k)})))\), \( i, j = 1, 2, \ldots, n \), \( n \).

Step 3: \( t_i^{(k+1)} = u_i^{(k)} (i = 1, 2, \ldots, n) \).

Step 4: if \( |f(t_i^{(k+1)})| < \epsilon \) or \( k > p \), then stop.

Step 5: set \( k = k + 1 \) and go to step 2.

**Algorithm 1:** Algorithm of the simultaneous iterative method (MMN14M).

| Method | \( n \) | CPU | \( e_1 \) | \( e_2 \) | \( e_3 \) | \( e_4 \) |
|--------|--------|-----|--------|--------|--------|--------|
| PJM10  | 3      | 0.076 | 1.2e − 26 | 5.8e − 27 | 1.1e − 26 | 1.9e − 26 |
| GFLM15D| 3      | 0.087 | 1.0e − 40 | 3.6e − 8  | 3.4e − 8  | 3.0e − 14 |
| MMN14D | 3      | 0.045 | 1.1e − 63 | 1.0e − 63 | 2.0e − 64 | 2.0e − 64 |

**Table 2:** Simultaneous finding of all distinct roots.
| Method   | n   | CPU   | $e_1$   | $e_2$   | $e_3$   | $e_4$   |
|---------|-----|-------|---------|---------|---------|---------|
| PJM10   | 2   | 0.032 | 4.2e-44 | 8.8e-41 | 8.2e-42 | 4.1e-44 |
| GFLM15D | 2   | 0.057 | 4.9e-48 | 1.5e-35 | 6.3e-53 | 7.1e-60 |
| MMN14D  | 2   | 0.023 | 1.8e-65 | 0.0     | 0.0     | 1.0e-65 |

| Method   | n   | CPU   | $e_1$   | $e_2$   | $e_3$   | $e_4$   |
|---------|-----|-------|---------|---------|---------|---------|
| PJM10   | 2   | 0.147 | 1.4e-24 | 1.4e-19 | 2.0e-19 | 2.0e-19 |
| GFLM15D | 2   | 0.153 | 1.0e-63 | 1.2e-57 | 3.7e-55 | 1.2e-50 |
| MMN14D  | 2   | 0.116 | 2.0e-64 | 1.1e-63 | 2.1e-64 | 1.1e-63 |

| Method   | n   | CPU   | $e_1$   | $e_2$   | $e_3$   | $e_4$   |
|---------|-----|-------|---------|---------|---------|---------|
| PJM10   | 3   | 0.067 | 3.0e-85 | 1.5e-86 | 0.0     | 0.0     |
| GFLM15D | 3   | 0.071 | 3.3e-19 | 1.1e-13 | 8.1e-65 | 7.1e-75 |
| MMN14D  | 3   | 0.031 | 0.0     | 0.0     | 0.0     | 0.0     |

| Method   | n   | CPU   | $e_1$   | $e_2$   | $e_3$   | $e_4$   |
|---------|-----|-------|---------|---------|---------|---------|
| PJM10   | 3   | 1.076 | 7.0e-24 | 1.2e-15 | 1.8e-15 | 1.6e-17 |
| GFLM15D | 3   | 1.340 | 1.6e-84 | 1.2e-69 | 1.8e-56 | 7.1e-80 |
| MMN14D  | 3   | 0.815 | 0.0     | 0.0     | 0.0     | 0.0     |

| Method   | n   | CPU   | $e_1$   | $e_2$   | $e_3$   | $e_4$   |
|---------|-----|-------|---------|---------|---------|---------|
| PJM10   | 5   | 1.643 | 2.8     | 0.4     | 13.7    | 1.3e-4  |
| GFLM15D | 5   | 1.987 | 4.7e-28 | 4.9e-29 | 1.1e-60 | 3.7e-65 |
| MMN14D  | 5   | 0.915 | 8.1e-80 | 1.9e-76 | 3.0e-69 | 62e-71  |

| Method   | n   | CPU   | $e_1$   | $e_2$   | $e_3$   | $e_4$   |
|---------|-----|-------|---------|---------|---------|---------|
| PJM10   | 4   | 1.132 | 2.8e-43 | 1.8e-73 | 4.0e-42 | 1.1e-62 |
| GFLM15D | 4   | 1.031 | 1.2e-54 | 2.8e-68 | 1.8e-84 | 2.7e-133|
| MMN14D  | 4   | 1.132 | 2.8e-43 | 1.8e-73 | 4.0e-42 | 1.1e-62 |
### Table 11: Simultaneous finding of all distinct roots.

| Method     | n  | CPU | $e_1$   | $e_2$   | $e_3$   | $e_4$   | $e_5$   | $e_6$   |
|------------|----|-----|---------|---------|---------|---------|---------|---------|
| PJM10      | 2  | 0.063 | 9.4e-5 | 9.4e-5 | 1.5e-25 |
| GFLM15D    | 2  | 0.57  | 1.8e-7 | 1.8e-7 | 2.0e-7  |
| MMN14D     | 2  | 0.035 | 1.1e-13| 1.1e-14| 7.5e-14 |

### Table 12: Simultaneous finding of all distinct roots.

| Method     | n  | CPU | $e_1$   | $e_2$   | $e_3$   | $e_4$   | $e_5$   | $e_6$   |
|------------|----|-----|---------|---------|---------|---------|---------|---------|
| PJM10      | 5  | 1.043 | 1.3e-7 | 4.6e-10| 3.5e-11 | 1.2e-12 | 5.9e-5 | 3.1e-7  |
| GFLM15D    | 5  | 1.154 | 0.0     | 7.0e-144| 0.0    | 3.6e-84 | 0.0     | 0.0     |
| MMN14D     | 5  | 1.015 | 0.0     | 0.0     | 0.0     | 0.0     | 0.0     | 0.0     |

### Table 13: Simultaneous finding of all distinct roots.

| Method     | n  | CPU | $e_1$   | $e_2$   | $e_3$   | $e_4$   | $e_5$   | $e_6$   |
|------------|----|-----|---------|---------|---------|---------|---------|---------|
| GFLM15D    | 5  | 2.154 | 1.3e-85| 0.0     | 1.7e-84 | 3.0e-70 | 0.0     | 0.0     |
| MMN14D     | 5  | 1.875 | 0.0     | 0.0     | 1.1e-145| 0.0     | 0.0     | 0.0     |

### Table 14: Simultaneous finding of all distinct roots.

| Method     | n  | CPU | $e_1$   | $e_2$   | $e_3$   | $e_4$   | $e_5$   | $e_6$   | $e_7$   | $e_8$   | $e_9$   | $e_{10}$ |
|------------|----|-----|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| PJM10      | 3  | 2.043 | 1.1e-10 | 1.3e-16 | 2.7e-15 | 4.3e-17 | 3.4e-15 | 1.7e-16 | 1.7e-15 | 3.1e-17 | 6.1e-18 | 2.1e-25 |
| GFLM15D    | 3  | 2.141 | 1.3e-27 | 1.4e-25 | 3.0e-19 | 1.6e-35 | 2.7e-37 | 6.1e-47 | 4.9e-33 | 1.2e-37 | 1.6e-50 | 1.3e-32 |
| MMN14D     | 3  | 1.105 | 3.1e-64 | 9.4e-20 | 8.1e-29 | 9.1e-36 | 9.3e-45 | 3.5e-65 | 1.3e-70 | 5.4e-60 | 3.2e-45 | 1.7e-53 |

### Table 15: Simultaneous finding of all distinct roots of linear combination of Legendre polynomial.

| Method     | n  | CPU | $e_1$   | $e_2$   | $e_3$   | $e_4$   | $e_5$   | $e_6$   | $e_7$   | $e_8$   | $e_9$   | $e_{10}$ |
|------------|----|-----|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| PJM10      | 3  | 1.143 | 1.0e-3  | 1.1e-2  | 7.5e-6  | 0.1e-3  | 1.2e-5  | 3.5e-4  | 4.1e-3  | 2.1e-3  | 6.1e-5  | 1.1e-3  |
| GFLM15D    | 3  | 1.748 | 2.1e-110| 1.3e-125| 6.7e-100| 2.0e-27 | 3.2e-25 | 3.6e-88 | 1.2e-15 | 1.7e-75 | 3.5e-65 | 1.7e-105|
| MMN14D     | 3  | 1.015 | 1.1e-125| 3.8e-125| 4.8e-126| 3.7e-126| 1.8e-126| 1.0e-127| 1.0e-127| 2.1e-126| 9.7e-126|

Figure 2: Residual error graph of $f_1(t)$ using PJM10, MMN14D, and GFLM15D.
Figure 3: Residual error graph of $f_2(t)$ using PJM10, MMN14D, and GFLM15D.

Figure 4: Residual error graph of $f_3(t)$ using PJM10, MMN14D, and GFLM15D.
Figure 5: (a, b) Residual error graph of \( f_4(t) \) and \( f_{4-1}(t) \) using PJM10, MMN14D, and GFLM15D.
Figure 6: (a, b) Residual error graph of $f_5(t)$ and $f_{5-1}(t)$ using PJM10, MMN14D, and GFLM15D.
Figure 7: (a, b) Residual error graph of $f_6(t)$ and $f_{6-1}(t)$ using PJM10, MMN14D, and GFLM15D.
\[ f_1(t) = (t)^4 - 1.9404 \times (t)^2 + 0.75, \]  

having exact roots

\[ \zeta_1 = 1.1864, \zeta_2 = -1.1864, \zeta_3 = 0.73, \zeta_4 = -0.73. \]  

The initial estimations of (28) are taken as

\[ t_1^{(0)} = 4, t_2^{(0)} = -5, t_3^{(0)} = 6, t_4^{(0)} = 8. \]  

Figure 12 shows that (28) has two positive roots which are determined in 3 iterations by PJM10, GFLM15D, and MMN14D methods, and the comparison is shown in Table 2. We observe that MMN14D has better performance in terms of CPU time and absolute errors as compared to PJM10 and GFLM15D, respectively. Residual errors of MMN14D are also very less as compared to PJM10 and GFLM15D as shown by residual graph for this polynomial in Figure 2.

Figure 2 shows residual graph for approximating roots of nonlinear function \( f_1(t) \) using simultaneous methods PJM10, MMN14D, and GFLM15D, respectively.

Figure 12 shows that \( f_1(t) \) has two positive roots and one negative root. However, negative root is redundant.

Thus, for car stability, the required positive roots are \( t_1^{(2)} = 1.18640839595484 \) and \( t_2^{(2)} = 0.7299555589266 \). Using these values in \( (\omega/p) = t \) and

\[ p = \sqrt{\frac{1.397 \times 10^7}{1.2 \times 10^9}} = 34.12s^{-1}, \ (p = \text{natural frequency}), \]  

we have

\[ w_{i1} = p^{*}t_1^{(2)} = 34.12 \times 1.18640839595484 = 40.48025447s^{-1}, \]

\[ w_{i3} = p^{*}t_3^{(2)} = 34.12 \times 0.7299555589266 = 24.90608367s^{-1}, \]  

which yields the following velocity for car stability:

\[ v_{i1} = \frac{w_{i1}D}{2\pi} = 463.869546976 \text{ km/hr}, \]

\[ v_{i3} = \frac{w_{i3}D}{2\pi} = 285.4026956 \text{ km/hr}, \]  

where \( D = 20 \text{ meter} \) is the distance between peeks and \( v_{i1} \) and \( v_{i3} \) are the horizontal speeds of the car at times \( t1 \) and \( t3 \).

**Example 2. Application in civil engineering.**

Figure 13(a) shows a uniform beam subject to a linearly increasing distributed load.

The equation for the elastic curve (Figure 13(b)) is

\[ f(t) = \frac{\omega_0}{120EIL} (-t^5 + 2L^2t^3 - L^4t), \]  

We have to find the point of maximum deflection, i.e., the value of \( t \), where \( f'(t) = 0: \)

\[ \frac{\omega_0}{120EIL} (-5t^4 + 6L^2t^2 - L^4) = 0. \]  

Let

\[ f_2(t) = \frac{\omega_0}{120EIL} (-5t^4 + 6L^2t^2 - L^4). \]  

Then, substituting this value in (38), we get the value of maximum deflection. Use the following values in
Figure 9: (a, b) Residual error graph of $f_8(t)$ and $f_{8-1}(t)$ using PJM10, MMN14D, and GFLM15D.
computation:
$L = 600\text{ cm}, E = 50,000\text{ kN/cm}^3, I = 30,000\text{ cm}^4,$ and $\omega_0 = 2.5\text{ kN/cm}.$

The exact roots of (38) are

$\zeta_1 = -599.999, \zeta_2 = -268.328, \zeta_3 = 268.328, \zeta_4 = 599.999.$

(39)

The initial estimations of (38) have been taken as

$t^{(0)}_1 = -900, t^{(0)}_2 = -400, t^{(0)}_3 = 400, t^{(0)}_4 = 1000.$

(40)

We observe that the method MMN14D is superior in terms of numerical results, CPU time, and error as compared to PJM10 and GFLM15D as shown in Table 3 and residual graph by Figure 3.

Thus, for maximum deflection, we put $t^{(2)}_1 = -599.9999999800000006144$ in (38) and get the deflection equal to $1 \times 10^{-9}.$

Figure 3 shows the residual graph for approximating roots of nonlinear function $f_3(t)$ using simultaneous methods PJM10, MMN14D, and GFLM15D, respectively.

Example 3 (thermodynamics). Mechanical engineering Application.

In general, mechanical engineering as well as most other scientists use thermodynamics extensively in their research work. The following polynomial is used to relate the zero-pressure specific heat of dry air, $C_\rho$, to temperature:

$$C_\rho = 1.9520 \times 10^{-14} t^4 - 9.5838 \times 10^{-11} t^3 + 9.7215 \times 10^{-8} t^2 + 1.671 \times 10^{-4} t + 0.99403.$$  

(41)

We have to determine the temperature that corresponds to specific heat of 1.2 (kJ/kgK).

Putting $C_\rho = 1.2$ in the above equation, we have the following polynomial:

$$f_3(t) = 1.9520 \times 10^{-14} t^4 - 9.5839 \times 10^{-11} t^3 + 9.7215 \times 10^{-8} t^2 + 1.671 \times 10^{-4} t - 0.20597,$$  

(42)

with exact roots

$$\zeta_1 = 1126.009751, \zeta_2 = 2536.837119 + 910.5010371i, \zeta_3 = -1289.950382,$$

$$\zeta_4 = 2536.837119 - 910.5010371i.$$  

(43)

The initial estimations of (42) have been taken as...
We observe that our method, namely, MMN14D, has better performance in terms of numerical results, CPU time, and residual errors as compared to PJM10 and GFLM15D as shown in Table 4 and residual graph in Figure 4. shows

\[ t_1^{(0)} = -300 - 800i, \ t_2^{(0)} = -300 + 800i, \ t_3^{(0)} = 8000 - 1000i, \ t_4^{(0)} = 8000 + 1000i. \]  

Figure 11: Residual error graph of \( f_{11}(t) \) using PJM10, MMN14D, and GFLM15D.

Figure 12: Two positive root of \( f_1(t) \).
residual graph for approximating roots of nonlinear function \( f_3(t) \) using simultaneous methods PJM10, MMN14D, and GFLM15D, respectively.

**Example 4.** Multiple complex roots [33].
Consider
\[
f_4(t) = (t + 1)^{14} (t - 2)^{12} (t - 1 - i)^{14} (t - 1 + i)^{10},
\]
with multiple exact roots,\[
\zeta_1 = -1, \quad \zeta_2 = 2, \quad \zeta_3 = 1 + i, \quad \zeta_4 = 1 - i,
\]
of the multiplicity \( \sigma_1 = 14, \sigma_2 = 12, \sigma_3 = 14, \) and \( \sigma_4 = 10 \), respectively. The initial estimations have been taken as\[
t_1^{(0)} = -4, \quad t_2^{(0)} = 4.1, \quad t_3^{(0)} = 4.9 + 4.0i, \quad t_4^{(0)} = 6.0 - 6.0i.
\]
For distinct roots, we take
\[
f_{4-1}(t) = (t + 1)(t - 2)(t - 1 - i)(t - 1 + i).
\]
We observe that our methods, namely, MMN14D and MMN14M, have better performance in terms of numerical results, CPU time, and residual errors as compared to PJM10, GFLM15D, and GFLM15M as shown in Tables 5 and 6 and residual graph in Figures 5(a) and 5(b).

**Example 5.** Multiple complex roots.
Consider
\[
f_5(t) = (t - 0.3 - 0.6i)^{100} (t - 0.1 - 0.7i)^{200} \cdot (t - 0.7 - 0.5i)^{300} (t - 0.3 - 0.4i)^{400},
\]
with multiple exact roots,\[
\zeta_1 = 0.3 + 0.6i, \quad \zeta_2 = 0.1 + 0.7i, \quad \zeta_3 = 0.7 + 0.5i, \quad \zeta_4 = 0.3 + 0.4i,
\]
of the multiplicity \( \sigma_1 = 100, \sigma_2 = 200, \sigma_3 = 300, \) and \( \sigma_4 = 400 \), respectively. The initial estimations have been taken as \( t_1^{(0)} = 6, \quad t_2^{(0)} = -5, \quad t_3^{(0)} = -3, \quad t_4^{(0)} = 7 \).

For distinct roots,
\[
f_{z-1}(t) = (t - 0.3 - 0.6i)(t - 0.1 - 0.7i) \cdot (t - 0.7 - 0.5i)(t - 0.3 - 0.4i).
\]

We observe that our methods, namely, MMN14D and MMN14M, have better performance in terms of numerical results, CPU time, and residual errors as compared to PJM10, GFLM15D, and GFLM15M as shown in Tables 7 and 8 and residual graph in Figures 6(a) and 6(b).

**Example 6.** Real roots with high multiplicity.
Consider
\[
f_6(t) = (t - 1)^{40} (t - 2)^{30} (t - 3)^{20} (t - 4)^{10},
\]
with multiple exact roots,\[
\zeta_1 = 1, \quad \zeta_2 = 2, \quad \zeta_3 = 3, \quad \zeta_4 = 4,
\]
of the multiplicity \( \sigma_1 = 40, \sigma_2 = 30, \sigma_3 = 20, \) and \( \sigma_4 = 10 \), respectively. The initial estimations have been taken as \( t_1^{(0)} = 10.1, \quad t_2^{(0)} = 7.1, \quad t_3^{(0)} = 9.1, \quad t_4^{(0)} = 12.1 \).

For distinct roots,
\[
f_{g-1}(t) = (t - 1)(t - 2)(t - 3)(t - 4).
\]

We observe that our method, namely, MMN14D and MMN14M, have better performance in terms of numerical results, CPU time, and residual errors as compared to PJM10, GFLM15D, and GFLM15M as shown in Tables 9 and 10 and residual graph in Figures 7(a) and 7(b).

**Example 7.** Fluid permeability in biogels [34].
Specific hydraulic permeability relates the pressure gradient to fluid velocity in porous medium (agarose gel or...
\[ f_1(t) = \text{Red}, \quad f_2(t) = \text{Yellow}, \quad f_3(t) = \text{Green}, \quad f_4(t) = \text{Blue}, \]
\[ f_5(t) = \text{Gray}, \quad f_6(t) = \text{Magenta}, \quad f_7(t) = \text{Cyan}, \quad f_8(t) = \text{Purple}, \quad f_9(t) = \text{Pink}, \quad f_{10}(t) = \text{Brown} \]

Figure 14: (a) Legendre polynomial from \( f_0 - f_{10} \) and (b) Legendre polynomials of degree 10 only.
extracellular fiber matrix) results the following nonlinear polynomial equations:

\[ k = \frac{R_e x^3}{20(1 - t)^3}, \]  

or

\[ R_e t^3 - 20k(1 - t)^2 = 0, \]  

where \( k \) is specific hydraulic permeability, \( R_e \) radius of the fiber, and \( t \) is the porosity [35]. Using \( k = 0.4655 \) and \( R_e = 100 \times 10^{-9} \), we have

\[ f_7(t) = -100 \times 10^{-9} t^3 + 9.3100 \times t^2 - 18.6200 \times t + 9.3100. \]  

(59)

The exact roots of (59) are

\[ \zeta_1 = 0.9999999997, \zeta_2 = 1.000000000, \zeta_3 = 9.31 \times 10^{18}. \]  

We choose the following initial estimates for simultaneous determination of all roots of (59) are

\[ t_1^{(0)} = 0.9, t_2^{(0)} = 1.1, t_3^{(0)} = 9.3 \times 10^{17}. \]  

(61)

Figure 8 shows residual graph for approximating roots of nonlinear function \( f_7(t) \) using simultaneous methods PJM10, MMN14D, and GFLM15D, respectively.

We observe that our method, namely, MMN14D and MMN14M, have better performance in terms of numerical results, CPU time, and residual errors as compared to PJM10, GFLM15D, and GFLM15M as shown in Tables 12 and 13 and residual graph in Figures 9(a) and 9(b).

Example 9 (see [36]). The solution of Legendre polynomial.

Legendre polynomial \( f_n(t) \) is the solution of the Legendre differential equation:

\[ (1 - t^2)\frac{d^2f_n(t)}{dt^2} - 2t\frac{df_n(t)}{dt} + n(n + 1)f_n(t) = 0, \quad n = 0, 1, \ldots. \]  

(66)

For Legendre polynomials, the recursive relation are

\[ f_0(t) = 1, f_1(t) = t, f_{n+2} = ((2n + 3)/(n + 2))t f_{n+1} - ((n + 1)/(n + 2)) f_n, n = 0, 1, \ldots \]  

Legendre polynomials are plotted in Figures 14(a) and 14(b). It is notable how the roots are clustered in the ends of the domain \([-1, 1]\), as shown in Figure 14(a). Here, we find the roots of Legendre polynomial of degree 10, i.e., \( f_{10}(t) \):

\[ f_{10}(t) = \frac{1}{256}(46189t^{10} - 109395t^8 \]
\[ + 90090t^6 - 303030t^4 + 3465t^2 - 63). \]  

(67)

We choose the following random initial approximations of (67):

\[ t_1^{(0)} = -1.04, t_2^{(0)} = -1.04, t_3^{(0)} = -0.7, t_4^{(0)} \]
\[ = -0.7, t_5^{(0)} = -0.13, \]  

(68)

\[ t_6^{(0)} = 1.13, t_7^{(0)} = 0.4, t_8^{(0)} = 0.4, t_9^{(0)} = 1.8, t_{10}^{(0)} = 0.8. \]

Figure 14(a) shows plot of Legendre polynomial from \( f_1(t) \) to \( f_{10}(t) \), while Figure 14(b) shows Legendre polynomial of degree 10 only, i.e., \( f_{10}(t) \).

Figure 10 shows the residual graph for approximating roots of nonlinear function \( f_{10}(t) \) using simultaneous methods PJM10, MMN14D, and GFLM15D, respectively.

Numerical results for linear combination of Legendre polynomials.

Although the roots of Legendre polynomials are real and lie in a specific interval, the roots of their linear combination need not be real may be complex. Some results on these linear combinations are given below:

\[ f_{10}(t) + 2f_5(t) + 3f_8(t) + 4f_7(t) + 5f_6(t) + 6f_5(t) + 7f_4(t) + 8f_3(t) + 9f_2(t) + 10f_1(t) + 11f_0(t) = 0, \]  

(69)
where

\[ f_0(t) = 1, \]
\[ f_1(t) = t, \]
\[ f_2(t) = \frac{1}{2} \left( 3t^2 - 1 \right), \]
\[ f_3(t) = \frac{1}{2} \left( 5t^3 - 3t \right), \]
\[ f_4(t) = \frac{1}{8} \left( 35t^4 - 30t^2 + 3 \right), \]
\[ f_5(t) = \frac{1}{8} \left( 63t^5 - 70t^3 + 15t \right), \]
\[ f_6(t) = \frac{1}{16} \left( 231t^6 - 315t^4 + 105t^2 - 5 \right), \]
\[ f_7(t) = \frac{1}{16} \left( 429t^7 - 693t^5 + 315t^3 - 35t \right), \]
\[ f_8(t) = \frac{1}{128} \left( 6435t^8 - 12012t^6 + 6930t^4 - 1260t^2 + 35 \right), \]
\[ f_9(t) = \frac{1}{128} \left( 12155t^9 - 25740t^7 + 18018t^5 - 4620t^3 + 315t \right), \]
\[ f_{10}(t) = \frac{1}{256} \left( 46189t^{10} - 109395t^8 + 90090t^6 - 30030t^4 + 3465t^2 - 63 \right). \]

\begin{table}[h]
\centering
\begin{tabular}{|c|c|}
\hline
Exact root upto 5 D.P & Approximated roots upto 29 D.P. \\
\hline
-0.97390 & -0.9739065286949645626082637874 \hline
-0.86506 & -0.865063663989953196915160 \hline
-0.67940 & -0.6794095682988538736610857195 \hline
-0.43339 & -0.4333953941292471910421193291 \hline
-0.14887 & -0.1488743389816312108848260011 \hline
0.14887 & 0.1488743389816312108848260011 \hline
0.43339 & 0.4333953941292471910421193291 \hline
0.67940 & 0.6794095682988538736610857195 \hline
0.86506 & 0.865063663989953196915160 \hline
0.97390 & 0.9739065286949645626082637874 \hline
\end{tabular}
\caption{Approximate root of \( f_{10}(t) \).}
\end{table}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|}
\hline
Exact root upto 5 D.P & Approximated roots upto 26 D.P. \\
\hline
-1.04343 - 0.13335i & -1.043435442746761499747031756 - 0.333524610738184557261497085i \hline
-1.04343 + 0.13335i & -1.043435442746761499747031756 + 0.333524610738184557261497085i \hline
-0.70072 - 0.35611i & -0.700723759899621090796461 - 0.356127938766541642307607797i \hline
-0.70072 + 0.35611i & -0.700723759899621090796461 + 0.356127938766541642307607797i \hline
-0.13755 - 0.45042i & -0.137557723741216815853244559 + 0.45042389439524673029309733i \hline
-0.13755 + 0.45042i & -0.137557723741216815853244559 - 0.45042389439524673029309733i \hline
0.46025 - 0.38015i & 0.460253980135463018864437575 - 0.380156169636539269997965262i \hline
0.46025 + 0.38015i & 0.460253980135463018864437575 + 0.380156169636539269997965262i \hline
0.89514 - 0.18245i & 0.8951465371395847081650910 - 0.1824552082554832683582900i \hline
0.89514 + 0.18245i & 0.8951465371395847081650910 + 0.1824552082554832683582900i \hline
\end{tabular}
\caption{Approximate root of (71) using MMN14D.}
\end{table}
Putting values of \( f_0(t), \ldots, f_{10}(t) \) in (69), we have

\[
\frac{46189}{256} t^{10} + \frac{12155}{64} t^9 - \frac{70785}{256} t^8 - \frac{4719}{16} t^7 + \frac{18249}{128} t^6 + \frac{4977}{32} t^5 - \frac{2905}{128} t^4 - \frac{415}{16} t^3 + \frac{1041}{256} t^2 + \frac{347}{64} t + \frac{2083}{256} = 0, \tag{71}
\]

or

\[
f_{11}(t) = \frac{46189}{256} t^{10} + \frac{12155}{64} t^9 - \frac{70785}{256} t^8 - \frac{4719}{16} t^7 + \frac{18249}{128} t^6 + \frac{4977}{32} t^5 - \frac{2905}{128} t^4 - \frac{415}{16} t^3 + \frac{1041}{256} t^2 + \frac{347}{64} t + \frac{2083}{256}. \tag{71}
\]

We choose the following random initial approximations of (71):

\[
t_1^{(0)} = -1.04, t_2^{(0)} = -1.04, t_3^{(0)} = -0.7, t_4^{(0)} = -0.7, t_5^{(0)} = -0.13,
\]

\[
t_6^{(0)} = 1.13, t_7^{(0)} = 0.4, t_8^{(0)} = 0.4, t_9^{(0)} = 1.8, t_{10}^{(0)} = 0.8. \tag{72}
\]

Figure 11 shows the residual graph for approximating roots of nonlinear function \( f_{11}(t) \) using simultaneous methods PM10, MMN14D, and GFLM15D, respectively.

We observe that our methods, namely, MMN14D and MMN14M, have better performance in terms of numerical results, CPU time, and residual error as compared to PM10, GFLM15D, and GFLM15M as shown in Tables 2–15 and residual graph in Figures 2–11, respectively.

5. Conclusion

We have developed here two-step simultaneous computer methods of order fourteen for solving nonlinear polynomial equations, one for determining all the distinct roots, namely, MMN14D, and the other for determining multiple roots of nonlinear polynomial equations, namely, MMN14M. From comparison of numerical results, as depicted in Tables 1–17, computational efficiency (Figures 1(a) and 1(d)) and graphical representations of residual errors are shown in Figures 2–11; we observe that our methods (11) of 14th order are superior in terms of efficiency, CPU time, and residual errors as compared to the Petkovic method PM10 and the Gargantini–Farmer–Loizou method GFLM15D and GFLM15M. Using the similar ways, we can introduce more higher order and efficient methods.

Data Availability

No data were used to support this study.

Disclosure

The statements made and views expressed are solely the responsibility of the author.

Conflicts of Interest

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

Authors’ Contributions

All authors contributed equally in the preparation of this manuscript.

Acknowledgments

The fourth author (YUG) would like to acknowledge that this publication was made possible by a grant from Carnegie Corporation of New York.

References

[1] M. Cosnard and P. Fraigniaud, “Finding the roots of a polynomial on a MIMD multicomputer,” Parallel Computing, vol. 15, no. 1–3, pp. 75–85, 1990.
[2] S. Kanno, N. Kjurkchiev, and T. Yamamoto, “On Some methods for the Simultaneous determination of polynomial zeros,” Japan Journal of Industrial and Applied Mathematics, vol. 13, pp. 267–288, 1995.
[3] P. D. Proinov and S. I. Cholakov, “Semilocal convergence of Chebyshev-like root-finding method for simultaneous approximation of polynomial zeros,” Applied Mathematics and Computation, vol. 236, pp. 669–682, 2014.
[4] B. Sendov, A. Andreev, and N. Kjurkchiev, Numerical Solutions of Polynomial Equations, Elsevier science, New York, NY, USA, 1994.
[5] M. N. O. Ikhide, “On the Weierstrass and some Petkovic-like methods for numerical determination of polynomial zeros,” International Journal of Computer Mathematics, vol. 12, pp. 1505–1519, 2003.
[6] N. A. Mir, R. Munee, and I. Jabeen, “Some families of two-step simultaneous methods for determining zeros of non-linear equations,” ISRN Applied Mathematics, pp. 1–11, 2011.
[7] A. W. M. Nourein, “An improvement on two iteration methods for simultaneous determination of the zeros of a polynomial,” Intern. J. Computer Math, pp. 241–252, 1977.
[8] S. I. Cholakov, “Local and semilocal convergence of Wang-Zheng’s method for simultaneous finding polynomial zeros,” Symmetry, vol. 2019, Article ID 736, 15 pages, 2019.
[9] P. D. Proinov and S. I. Ivanov, “On the convergence of Halley’s method for multiple polynomial zeros,” Mediterranean Journal of Mathematics, vol. 12, pp. 555–572, 2015.
[10] A. I. Iliev and Kh. I. Semerdzhiev, “Some generalizations of the Chebyshev method for simultaneous determination of all roots of polynomial equations,” Computational Mathematics and Mathematical Physics, vol. 39, no. 9, pp. 1384–1391, 1999.
[11] V. K. Kyncheva, V. V. Yotov, and S. I. Ivanov, “Convergence of Newton, Halley and Chebyshev iterative methods as methods for simultaneous determination of multiple
polynomial zeros,” *Applied Numerical Mathematics*, vol. 112, pp. 146–154, 2017.

[12] P. D. Proinov and M. T. Vasileva, “On the convergence of family of Weierstrass-type root-finding methods,” *Comptes Rendus de l’Academie Bulgare des Sciences*, vol. 68, pp. 697–704, 2015.

[13] X. Wang and L. Liu, “Modified Ostrowski’s method with eighth-order convergence and high efficiency index,” *Applied Mathematics Letters*, vol. 23, no. 5, pp. 549–554, 2010.

[14] T.-F. Li, D.-S. Li, Z.-D. Xu, and Y.-L. Fang, “New iterative methods for non-linear equations,” *Applied Mathematics and Computation*, vol. 197, no. 2, pp. 755–759, 2008.

[15] P. D. Proinov and S. I. Ivanov, “Convergence analysis of Sakurai-Torii-Sugiura iterative method for simultaneous approximation of polynomial zeros,” *Journal of Computational and Applied Mathematics*, vol. 357, pp. 56–70, 2019.

[16] N. Rafiq, S. Akram, N. A. Mir, and M. Shams, “Study of dynamical behavior and stability of iterative methods for non-linear equations with application in engineering,” *Mathematical Problems in Engineering*, vol. 2020, p. 20, 2020.

[17] G. H. Nedzhivov, “Iterative methods for simultaneous computing arbitrary number of multiple zeros of nonlinear equations,” *International Journal of Computer Mathematics*, pp. 994–1007, 2013.

[18] A. J. Sommese and C. W. Wampler, *The Numerical Solution of Systems of Polynomials Arising Inengineering and Science*, World Scientific, Singapore, 2005.

[19] R. Thukral, “A new fifth-order iterative method for finding multiple roots of nonlinear equations,” *American Journal of Computational Mathematics*, vol. 6, pp. 260–264, 2012.

[20] M. S. Petkovic, L. D. Petkovic, and J. Djunic, “On an efficient simultaneous method for finding polynomial zeros,” *Applied Mathematics Letters*, vol. 28, pp. 60–65, 2014.

[21] M. Farmer and G. Loizou, “An algorithm for the total or partial factorization of a polynomial,” *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 82, pp. 427–437, 1977.

[22] P. D. Proinov and M. D. Petkova, “Local and semilocal convergence of a family of multi-point Weierstrass-type root finding methods,” *Mediterranean Journal of Mathematics*, vol. 17, no. 4, p. 20, 2020.

[23] P. D. Proinov and M. T. Vasileva, “On the convergence of high-order Ehrlich-type iterative methods for approximating all zeros of a polynomial simultaneously,” *Journal of Inequalities and Applications*, vol. 336, pp. 1–25, 2015.

[24] P. D. Proinov and M. T. Vasileva, “On the convergence of high-order Gargantini-Farmer-Loizou type iterative methods for simultaneous approximation of polynomial zeros,” *Applied Mathematics and Computation*, vol. 361, pp. 202–214, 2019.

[25] P. K. Parida and D. K. Gupta, “An improved method for finding multiple roots and it’s multiplicity of nonlinear equations in R,” *Applied Mathematics and Computation*, vol. 202, pp. 498–503, 2008.

[26] T. Miyakoda, “Iterative methods for multiple zeros of a polynomial by clustering,” *Journal of Computational and Applied Mathematics*, vol. 28, pp. 315–326, 1989.

[27] F. Soleymani, D. K. R. Babajee, and T. Lotfi, “On a numerical technique for finding multiple zeros and its dynamic,” *Journal of the Egyptian Mathematical Society*, vol. 21, pp. 346–353, 2013.

[28] J. F. Traub, *Iterative Methods for the Solution of Equations*, Prentice Hall, New York, NY, USA, 1964.

[29] J. L. Lagouanelle, “Sur une mtode de calcul de l’ordre de multiplicit des zros d’un polynme,” *Comptes rendus de l’Academie des Sciences*, vol. 262, pp. 626–627, 1966.

[30] M. S. Petkovic, L. D. Petkovic, and J. Djunic, “Accelerating generators of iterative methods for finding multiple roots of nonlinear equations,” *Computers & Mathematics with Applications*, vol. 59, pp. 2784–2793, 2010.

[31] T. Johnson and W. Tucker, “Enclosing all zeros of an analytic function A rigorous approach,” *Journal of Computational and Applied Mathematics*, vol. 228, pp. 418–423, 2009.

[32] Z. Wu and X. Li, “A fourth-order modification of Newton’s method for multiple roots,” *IJRRAS*, vol. 10, pp. 166–170, 2012.

[33] M. R. Farmer, “Computing the zeros of polynomials using the Divide and Conquer approach,” Ph.D. thesis, Department of Computer Science and Information Systems, Birkbeck, University of London, London, UK, 2014.

[34] R. L. Fournier, *Basic Transport Phenomena in Biomedical Engineering*, Taylor & Francis, New York, NY, USA, 2007.

[35] W. M. Saltzman, *Drug Delivery: Engineering Principal for Drug Therapy*, Oxford University Press, New York, NY, USA, 2001.

[36] G. Karniadakis and S. Sherwin, *Spectral/hp Element Methods for Computational Fluid Dynamics* by Karniadakis and Sherwin, Oxford University Press, Oxford, UK, 2005.