Research Article

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

Naila Rafiq, Mudassir Shams, Nazir Ahmad Mir, and Yaé Ulrich Gaba

Department of Mathematics, National University of Modern Languages, Islamabad, Pakistan
Department of Mathematics and Statistics, Riphah International University I-14, Islamabad 44000, Pakistan
Quantum Leap Africa (QLA), AIMS Rwanda Centre, Remera Sector KN 3, Kigali, Rwanda
Institut de Mathématiques et de Sciences Physiques (IMSP/UAC), Laboratoire de Topologie Fondamentale, Computationnelle et Leurs Applications (Lab-ToFoCApp), BP 613 Porto-Novo, Benin
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

Abstract

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] Ikhide [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 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 $n$: 
\[ f(t) = \prod_{i=1}^{m} (t - \zeta_i)^{\sigma_i}, \quad (1) \]

with multiple real or complex exact root \( \zeta_1, \ldots, \zeta_n \) of respective unknown multiplicities \( \sigma_1, \ldots, \sigma_n (\sigma_1 + \cdots + \sigma_n = m) \). Generally, the multiplicity of roots is not given in advance. However, research studies are working on numerical methods which approximate the unknown multiplicity of roots, see, e.g., [25–31].

2. Construction of Simultaneous Computer Methods for Multiple Roots

Considering two-step fourth-order Newton’s method [32] for finding multiple roots of nonlinear polynomial equation (1),

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

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:

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

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

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

This implies

\[
\begin{align*}
    f(t_i) &= \prod_{i=1}^{m} (t_i - t_i), \\
    f'(t_i) &= \sum_{i=1}^{m} \prod_{i \neq k}^{m} (t_i - t_i).
\end{align*}
\]

or

\[
\begin{align*}
    f(t_i) &= \sum_{j=1}^{m} \frac{1}{(t_i - t_j)}, \\
    f'(t_i) &= \sum_{j=1}^{m} \frac{1}{(t_i - t_j)} - \sum_{j=1}^{m} \frac{1}{(t_i - t_j)}.
\end{align*}
\]

This gives

\[
\begin{align*}
    t - t_i &= \frac{1}{\sum_{j=1}^{m} \frac{1}{(t_i - t_j)}},
\end{align*}
\]

where

\[
\begin{align*}
    N_i(t_i) &= \frac{f(t_i)}{f'(t_i)}.
\end{align*}
\]

or

\[
\begin{align*}
    f(t_i) &= \prod_{i=1}^{m} \frac{1}{(t_i - t_j)}, \\
    f'(t_i) &= \prod_{i=1}^{m} \frac{1}{(t_i - t_j)} - \prod_{j=1}^{m} \frac{1}{(t_i - t_j)}.
\end{align*}
\]

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

\[
\begin{align*}
    \sigma_i f(t_i) &= \frac{\sigma_i}{f'(t_i)} = \frac{\sigma_i}{(N_i(t_i))} - \frac{\sigma_i}{\sum_{j=1}^{m} (t_i - t_j)},
\end{align*}
\]

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 \( t_i \) by \( z_i \) in (8), we have

\[
\begin{align*}
    \sigma_i f(t_i) &= \frac{\sigma_i}{f'(t_i)} = \frac{\sigma_i}{(N_i(t_i))} - \frac{\sigma_i}{\sum_{j=1}^{m} (t_i - t_j)}.\end{align*}
\]
\[ z_j = y_j - \sigma_j \left( 1 + \left( \frac{f(y_j)}{f(t_j)} \right)^{2\sigma_j} \right) \frac{f(y_j)}{f'(y_j)}, \]
\[ y_j = t_j - \sigma_j \frac{f(t_j)}{f'(t_j)}. \]

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

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 \( \xi_1, \ldots, \xi_n \) 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
\[
\epsilon_i = \xi_i^{(k)} - \xi_i, \\
\epsilon_i' = \xi_i^{(k)} - \xi_i, \\
\epsilon_i'' = u_i^{(k)} - \xi_i,
\]
be the errors in \( t_i^{(k)}, y_i^{(k)}, \) and \( u_i^{(k)} \) approximations, respectively. Consider the first step of (11):
\[
y_i^{(k)} = t_i^{(k)} - \frac{\sigma_i}{(\sigma_i/N(t_i^{(k)})) - \sum_{j \neq i}^{n} ((\sigma_i/(t_i^{(k)} - z_j^{(k)}))}
\]
where
\[
N(t_i^{(k)}) = \frac{f(t_i^{(k)})}{f'(t_i^{(k)})}
\]

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

Thus, for multiple roots, we have, from (11),
\[
y_i^{(k)} = t_i^{(k)} - \frac{\sigma_i}{\left( (\sigma_i/t_i^{(k)} - \xi_i) + \sum_{j \neq i}^{n} ((\sigma_j/t_i^{(k)} - \xi_j)) \right)} - \sum_{j \neq i}^{n} \left( (\sigma_j/(t_i^{(k)} - z_j)) \right).
\]
\[
\xi_i^{(k)} - \xi_i = t_i^{(k)} - \xi_i - \frac{\sigma_i}{\left( (\sigma_i/t_i^{(k)} - \xi_i) + \sum_{j \neq i}^{n} ((\sigma_j/t_i^{(k)} - z_j)) \right)} - \sum_{j \neq i}^{n} \left( (\sigma_j/(t_i^{(k)} - z_j)) \right).
\]
\[
\epsilon_i' = \frac{\sigma_i}{\left( (\sigma_i/t_i^{(k)} - \xi_i) + \sum_{j \neq i}^{n} ((\sigma_j/t_i^{(k)} - z_j)) \right)} + \sum_{j \neq i}^{n} \left( (\sigma_j/(t_i^{(k)} - z_j)) \right).
\]
where \((Z_j^{(k)} - \zeta_j) = \epsilon_j^5\) [19] and \(E_i = (-\sigma_j)/(t_j^{(k)})\). Thus,

\[
\epsilon'_i = \frac{\epsilon_i^2 \sum_{j \in E_i} \epsilon_j^5}{\sigma_i + \epsilon_i \sum_{j \in E_i} \epsilon_j^5} \tag{17}
\]

If it is assumed that all errors \(\epsilon_j\) \((j = 1, 2, 3, \ldots)\) are of the same order as, say \(\epsilon\) = \(O[\epsilon]\), then, from (17), we have

\[
\epsilon'_i = O(\epsilon)^7. \tag{18}
\]

From the second equation of (11),

\[
u_i^{(k)} = y_i^{(k)} - \frac{\sigma_i}{\left(\sigma_i/(y_i^{(k)} - \zeta_i) - \sum_{j \in E_i} \left(\sigma_j/(y_j^{(k)} - y_j^{(k)})\right)\right)} \tag{19}
\]

This implies

\[
\epsilon''_i = \epsilon'_i - \frac{\sigma_i}{\left(\sum_{j \in E_i} \left(\sigma_j/(y_j^{(k)} - y_j^{(k)}) + \zeta_j\right)\right)} \left(\sum_{j \in E_i} \left(\sigma_j/(y_j^{(k)} - y_j^{(k)})\right)\right)
\]

\[
= \epsilon'_i - \frac{\sigma_i \epsilon'_i}{\sigma_i + \epsilon_i \sum_{j \in E_i} \epsilon_j^5 F_i} \tag{20}
\]

where \(F_i = \left(y_i^{(k)} - \zeta_i\right)/(y_i^{(k)} - y_j^{(k)})\)

This implies

\[
\epsilon'_i = \epsilon'_i - \frac{\sigma_i \epsilon'_i}{\sigma_i + \epsilon_i \sum_{j \in E_i} \epsilon_j^5 F_i} \tag{21}
\]

where \(\epsilon'_i = O(\epsilon)^7\), thus,

\[
\epsilon'_i = \epsilon'_i - \frac{\sigma_i \epsilon'_i}{\sigma_i + \epsilon_i \sum_{j \in E_i} \epsilon_j^5 F_i} = \epsilon'_i - \epsilon_i \left(\frac{\sum_{j \in E_i} \epsilon_j^5 F_i}{\sigma_i + \epsilon_i \sum_{j \in E_i} \epsilon_j^5 F_i}\right)
\]

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

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

For a given polynomial of degree \( n \), the efficiency of an iterative method can be estimated using the efficiency index given by

Here, we compare the computational efficiency and 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. 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 \( n \), the number of division, multiplication, addition, and subtraction per iteration for all roots is denoted by \( AS_m, M_m, \) and \( D_m \). The cost of computation can be calculated as

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 \( AS_m, M_m, \) and \( D_m \). The cost of computation can be calculated as

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

Thus, (23) becomes

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

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 \( AS_m, M_m, \) and \( D_m \). The cost of computation can be calculated as

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

(a) \( \rho ((11), (X)) = \left( \frac{EF(11)}{EF(15)} \right) \times 100 \) (in percent),  

(b) \( \rho ((X), (11)) = \left( \frac{EF(15)}{EF(11)} \right) \times 100 \) (in percent),

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 \( \epsilon = 10^{-30} \) as a tolerance and use the following stopping criteria for estimating the roots:

\[ \varepsilon^{(k)} = \frac{f(x_{(k+1)})}{f(x_k)} < \epsilon, \]  

where \( \varepsilon^{(k)} \) 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 and 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:

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

Let

\[ \frac{\omega}{\rho} = t. \]

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

- $y_j^{(k)} = t_j^{(k)} - \sigma_j (f(t_j^{(k)}/f'(t_j^{(k)}))$ and $\sigma_j$ is the multiplicity of actual multiple roots $\zeta_j$.
- $z_j^{(k)} = y_j^{(k)} - \sigma_j (1 + (f(y_j^{(k)})/f'(y_j^{(k)}))^{(i+1)}) (f(y_j^{(k)})/f'(y_j^{(k)}))$.
- update $t_j^{(k+1)} = (\sigma_j) / ((\sigma_j) / N^j_{i+1}) - (\sum_{j=1}^n ((\sigma_j) / (t_j^{(k+1)} - z_j^{(k)})))$, $(i, j = 1, 2, \ldots, n)$

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).

### Table 2: Simultaneous finding of all distinct roots.

<table>
<thead>
<tr>
<th>Method</th>
<th>$n$</th>
<th>CPU</th>
<th>$e_1$</th>
<th>$e_2$</th>
<th>$e_3$</th>
<th>$e_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PJM10</td>
<td>3</td>
<td>0.076</td>
<td>$1.2e - 26$</td>
<td>$5.8e - 27$</td>
<td>$1.1e - 26$</td>
<td>$1.9e - 26$</td>
</tr>
<tr>
<td>GFLM15D</td>
<td>3</td>
<td>0.087</td>
<td>$1.0e - 40$</td>
<td>$3.6e - 8$</td>
<td>$3.4e - 8$</td>
<td>$3.0e - 14$</td>
</tr>
<tr>
<td>MMN14D</td>
<td>3</td>
<td>0.045</td>
<td>$1.1e - 63$</td>
<td>$1.0e - 63$</td>
<td>$2.0e - 64$</td>
<td>$2.0e - 64$</td>
</tr>
</tbody>
</table>

Figure 1: (a–d) Computational efficiency of MMN14M, PJM10, and GFLM15M.
Table 3: Simultaneous finding of all distinct roots.

<table>
<thead>
<tr>
<th>Method</th>
<th>n</th>
<th>CPU</th>
<th>$e_1$</th>
<th>$e_2$</th>
<th>$e_3$</th>
<th>$e_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PJM10</td>
<td>2</td>
<td>0.032</td>
<td>4.2e−44</td>
<td>8.8e−41</td>
<td>8.2e−42</td>
<td>4.1e−44</td>
</tr>
<tr>
<td>GFLM15D</td>
<td>2</td>
<td>0.057</td>
<td>4.9e−48</td>
<td>1.5e−35</td>
<td>6.3e−53</td>
<td>7.1e−60</td>
</tr>
<tr>
<td>MMN14D</td>
<td>2</td>
<td>0.023</td>
<td>1.8e−65</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0e−65</td>
</tr>
</tbody>
</table>

Table 4: Simultaneous finding of all distinct roots.

<table>
<thead>
<tr>
<th>Method</th>
<th>n</th>
<th>CPU</th>
<th>$e_1$</th>
<th>$e_2$</th>
<th>$e_3$</th>
<th>$e_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PJM10</td>
<td>2</td>
<td>0.147</td>
<td>1.4e−24</td>
<td>1.4e−19</td>
<td>2.0e−19</td>
<td>2.0e−19</td>
</tr>
<tr>
<td>GFLM15D</td>
<td>2</td>
<td>0.153</td>
<td>1.0e−63</td>
<td>1.2e−57</td>
<td>3.7e−55</td>
<td>1.2e−50</td>
</tr>
<tr>
<td>MMN14D</td>
<td>2</td>
<td>0.116</td>
<td>2.0e−64</td>
<td>1.1e−63</td>
<td>2.1e−64</td>
<td>1.1e−63</td>
</tr>
</tbody>
</table>

Table 5: Simultaneous finding of all distinct roots.

<table>
<thead>
<tr>
<th>Method</th>
<th>n</th>
<th>CPU</th>
<th>$e_1$</th>
<th>$e_2$</th>
<th>$e_3$</th>
<th>$e_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PJM10</td>
<td>3</td>
<td>0.067</td>
<td>3.0e−85</td>
<td>1.5e−86</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>GFLM15D</td>
<td>3</td>
<td>0.071</td>
<td>3.3e−19</td>
<td>1.1e−13</td>
<td>8.1e−65</td>
<td>7.1e−75</td>
</tr>
<tr>
<td>MMN14D</td>
<td>3</td>
<td>0.031</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Table 6: Simultaneous finding of all multiple roots.

<table>
<thead>
<tr>
<th>Method</th>
<th>n</th>
<th>CPU</th>
<th>$e_1$</th>
<th>$e_2$</th>
<th>$e_3$</th>
<th>$e_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>GFLM15M</td>
<td>2</td>
<td>1.013</td>
<td>5.3e−88</td>
<td>2.7e−69</td>
<td>6.2e−83</td>
<td>1.7e−52</td>
</tr>
<tr>
<td>MMN14M</td>
<td>2</td>
<td>0.172</td>
<td>8.7e−124</td>
<td>7.3e−80</td>
<td>1.5e−101</td>
<td>2.9e−70</td>
</tr>
</tbody>
</table>

Table 7: Simultaneous finding of all distinct roots.

<table>
<thead>
<tr>
<th>Method</th>
<th>n</th>
<th>CPU</th>
<th>$e_1$</th>
<th>$e_2$</th>
<th>$e_3$</th>
<th>$e_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PJM10</td>
<td>3</td>
<td>1.076</td>
<td>7.0e−24</td>
<td>1.2e−15</td>
<td>1.8e−15</td>
<td>1.6e−17</td>
</tr>
<tr>
<td>GFLM15D</td>
<td>3</td>
<td>1.340</td>
<td>1.6e−84</td>
<td>1.2e−69</td>
<td>1.8e−56</td>
<td>7.1e−80</td>
</tr>
<tr>
<td>MMN14D</td>
<td>3</td>
<td>0.815</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Table 8: Simultaneous finding of all multiple roots.

<table>
<thead>
<tr>
<th>Method</th>
<th>n</th>
<th>CPU</th>
<th>$e_1$</th>
<th>$e_2$</th>
<th>$e_3$</th>
<th>$e_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>GFLM15M</td>
<td>2</td>
<td>0.978</td>
<td>8.9e−150</td>
<td>7.0e−242</td>
<td>5.4e−355</td>
<td>1.3e−409</td>
</tr>
<tr>
<td>MMN14M</td>
<td>2</td>
<td>0.613</td>
<td>9.7e−298</td>
<td>2.1e−567</td>
<td>1.2e−385</td>
<td>8.1e−553</td>
</tr>
</tbody>
</table>

Table 9: Simultaneous finding of all distinct roots.

<table>
<thead>
<tr>
<th>Method</th>
<th>n</th>
<th>CPU</th>
<th>$e_1$</th>
<th>$e_2$</th>
<th>$e_3$</th>
<th>$e_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PJM10</td>
<td>5</td>
<td>1.643</td>
<td>2.8</td>
<td>0.4</td>
<td>13.7</td>
<td>1.3e−4</td>
</tr>
<tr>
<td>GFLM15D</td>
<td>5</td>
<td>1.987</td>
<td>4.7e−28</td>
<td>4.9e−29</td>
<td>1.1e−60</td>
<td>3.7e−65</td>
</tr>
<tr>
<td>MMN14D</td>
<td>5</td>
<td>0.915</td>
<td>8.1e−80</td>
<td>1.9e−76</td>
<td>3.0e−69</td>
<td>62e−71</td>
</tr>
</tbody>
</table>

Table 10: Simultaneous finding of all multiple roots.

<table>
<thead>
<tr>
<th>Method</th>
<th>n</th>
<th>CPU</th>
<th>$e_1$</th>
<th>$e_2$</th>
<th>$e_3$</th>
<th>$e_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>GFLM15M</td>
<td>4</td>
<td>1.132</td>
<td>2.8e−43</td>
<td>1.8e−73</td>
<td>4.0e−42</td>
<td>1.1e−62</td>
</tr>
<tr>
<td>MMN14M</td>
<td>4</td>
<td>1.031</td>
<td>1.2e−54</td>
<td>2.8e−68</td>
<td>1.8e−84</td>
<td>2.7e−133</td>
</tr>
</tbody>
</table>
Table 11: Simultaneous finding of all distinct roots.

<table>
<thead>
<tr>
<th>Method</th>
<th>n</th>
<th>CPU</th>
<th>$e_1$</th>
<th>$e_2$</th>
<th>$e_3$</th>
<th>$e_4$</th>
<th>$e_5$</th>
<th>$e_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PJM10</td>
<td>2</td>
<td>0.063</td>
<td>9.4e-5</td>
<td>9.4e-5</td>
<td>1.5e-25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GFLM15D</td>
<td>2</td>
<td>0.57</td>
<td>1.8e-7</td>
<td>1.8e-7</td>
<td>2.0e-79</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MMN14D</td>
<td>2</td>
<td>0.035</td>
<td>1.1e-13</td>
<td>1.1e-14</td>
<td>7.5e-140</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 12: Simultaneous finding of all distinct roots.

<table>
<thead>
<tr>
<th>Method</th>
<th>n</th>
<th>CPU</th>
<th>$e_1$</th>
<th>$e_2$</th>
<th>$e_3$</th>
<th>$e_4$</th>
<th>$e_5$</th>
<th>$e_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PJM10</td>
<td>5</td>
<td>1.043</td>
<td>1.3e-7</td>
<td>4.6e-10</td>
<td>3.5e-11</td>
<td>1.2e-12</td>
<td>5.9e-5</td>
<td>3.1e-7</td>
</tr>
<tr>
<td>GFLM15D</td>
<td>5</td>
<td>1.154</td>
<td>0.0</td>
<td>7.0e-144</td>
<td>0.0</td>
<td>3.6e-84</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>MMN14D</td>
<td>5</td>
<td>1.015</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Table 13: Simultaneous finding of all distinct roots.

<table>
<thead>
<tr>
<th>Method</th>
<th>n</th>
<th>CPU</th>
<th>$e_1$</th>
<th>$e_2$</th>
<th>$e_3$</th>
<th>$e_4$</th>
<th>$e_5$</th>
<th>$e_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>GFLM15D</td>
<td>5</td>
<td>2.154</td>
<td>1.3e-85</td>
<td>0.0</td>
<td>1.7e-84</td>
<td>3.0e-70</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>MMN14D</td>
<td>5</td>
<td>1.875</td>
<td>0.0</td>
<td>0.0</td>
<td>1.1e-145</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Table 14: Simultaneous finding of all distinct roots.

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

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

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

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 $f_i(t)$ and $f_{i+1}(t)$ 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.
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

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

(32)

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

$$p = \frac{1.397 \times 10^7}{1.2 \times 10^6} = 34.12 s^{-1}, \quad (p = \text{natural frequency})$$

we have

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

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

(34)

which yields the following velocity for car stability:

$$v_{t_1} = \frac{w_{t_1} D}{2\pi} = 463.869546976 \text{ km/hr},$$

$$v_{t_3} = \frac{w_{t_2} D}{2\pi} = 285.4026956 \text{ km/hr},$$

(35)

where $D = 20$ meter is the distance between peaks and $v_{t_1}$ and $v_{t_3}$ are the horizontal speeds of the car at times $t_1$ and $t_3$.

**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

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

(36)

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

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

(37)

Let

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

(38)

Then, substituting this value in (38), we get the value of maximum deflection. Use the following values in

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

(38)
Figure 9: (a, b) Residual error graph of \( f_8(t) \) and \( f_{8-1}(t) \) using PJM10, MMN14D, and GFLM15D.
computation:
\[ L = 600 \text{ cm}, E = 50,000 \text{kN/cm}^3, I = 30,000 \text{ cm}^4, \text{ and } \omega_o = 2.5 \text{kN/cm}. \]

The exact roots of (38) are
\[ \zeta_1 = -599.999, \zeta_2 = -268.328, \zeta_3 = 268.328, \zeta_4 = 599.999. \]

The initial estimations of (38) have been taken as
\[ t_1(0) = -900, t_2(0) = -400, t_3(0) = 400, t_4(0) = 1000. \]

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 \( t_1^{(2)} = -599.99999980800000006144 \) in (38) and get the deflection equal to \( 1 \times 10^{-9} \).

Figure 3 shows the residual graph for approximating roots of nonlinear function \( f_2(t) \) 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:

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

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:

\[ f_3(t) = 1.9520 \times 10^{-14} t^4 - 9.5838 \times 10^{-11} t^3 + 9.7215 \times 10^{-8} t^2 + 1.671 \times 10^{-4} t - 0.20597, \]

with exact roots

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

\[ \zeta_4 = 2536.837119 - 910.5010371i. \]

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

\begin{equation}
 t_1^{(0)} = -300 - 800i, \quad t_2^{(0)} = -300 + 800i, \quad t_3^{(0)} = 8000 - 1000i, \quad t_4^{(0)} = 8000 + 1000i.
\end{equation}
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
\[
f_4(t) = (t + 1)^{14} (t - 2)^{12} (t - 1 - i)^4 (t - 1 + i)^{10},
\]
with multiple exact roots,
\[
\zeta_1 = -1, \quad \zeta_2 = 2, \quad \zeta_3 = 1 + i, \quad \zeta_4 = 1 - i,
\]
of the multiplicity \( \sigma_1 = 14, \sigma_2 = 12, \sigma_3 = 4, \) 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).

Figures 5(a) and 5(b) show the residual graph for approximating roots of nonlinear function \( f_4(t), f_{4-1}(t) \) using simultaneous methods PJM10, MMN14D, MMN14M, GFLM15D, and GFLM15M, respectively.

**Example 5.** Multiple complex roots.
Consider
\[
f_5(t) = (t - 0.3 - 0.6i)^{100} (t - 0.1 - 0.7i)^{200} (t - 0.7 - 0.5i)^{300} (t - 0.3 - 0.4i)^{400},
\]
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,
\[
f_{5-1}(t) = (t - 0.3 - 0.6i)(t - 0.1 - 0.7i)
\]
\[
\cdot (t - 0.7 - 0.5i)(t - 0.3 - 0.4i).
\]

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).

Figures 6(a) and 6(b) show residual graph for approximating roots of nonlinear function \( f_5(t), f_{5-1}(t) \) using simultaneous methods PJM10, MMN14D, MMN14M, GFLM15D, and GFLM15M, respectively.

**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,
\[
f_{6-1}(t) = (t - 1)(t - 2)(t - 3)(t - 4).
\]

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).

Figures 7(a) and 7(b) show the residual graph for approximating roots of nonlinear function \( f_6(t), f_{6-1}(t) \) using simultaneous methods PJM10, MMN14D, MMN14M, GFLM15D, and GFLM15M, respectively.

**Example 7.** Fluid permeability in biogels [34].
Specific hydraulic permeability relates the pressure gradient to fluid velocity in porous medium (agarose gel or

![Figure 13: Uniform beam subject to a linearly increasing distributed load.](image-url)
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:

\[ k = \frac{R_e x^3}{20(1 - t)^4}, \]  

(57)

or

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

(58)

where \( k \) is specific hydraulic permeability, \( R_e \) radius of the fiber, and \( t \) is the porosity [35]. Using \( k = 0.4655 \) and \( R_e = 100 \times 10^{-6} \), 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

\[
\xi_1 = 0.9999999997, \quad \xi_2 = 1.000000000, \quad \xi_3 = 9.31 \times 10^{18}.
\]  

(60)

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

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

(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, has better performance in terms of numerical results, CPU time, and residual errors as compared to PJM10, GFLM15D, and GFLM15M as shown in Table 11 and residual graph in Figure 8.

**Example 8.** Consider

\[
f_8(t) = (t + 3)(t - 2i)^3(t^2 + 4t + 5)^2(t^2 + 4t + 5)^2,
\]  

(62)

with multiple exact roots,

\[
\xi_1 = -3, \quad \xi_2 = 2i, \quad \xi_3 = -2 \pm i, \quad \xi_4 = 2 \pm i,
\]  

(63)

of the multiplicity \( \sigma_1 = 1, \sigma_2 = 3, \sigma_3 = 2, \sigma_4 = 2, \) and \( \sigma_5 = 2, \) respectively. The initial estimations have been taken as

\[
t_1^{(0)} = 10.1, \quad t_2^{(0)} = 7.1, \quad t_3^{(0)} = 9.1, \quad t_4^{(0)} = 12.1, \quad t_5^{(0)} = 12.1.
\]  

(64)

For distinct roots,

\[
f_{8-1}(t) = (t + 3)(t - 2i)(t^2 + 4t + 5)(t^2 + 4t + 5).
\]  

(65)

Figures (a) and (b) show the residual graph for approximating roots of nonlinear functions \( f_8(t), f_{8-1}(t) \) using simultaneous methods PJM10, MMN14D, MMN14M, GFLM15D, and GFLM15M, respectively.

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 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:

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

(66)

For Legendre polynomials, the recursive relation are

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

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., \( f_{10}(t) \):

\[
f_{10}(t) = \frac{1}{256}(46189 t^{10} - 109395 t^8
\]

\[+ 90090 t^6 - 30030 t^4 + 3465 t^2 - 63).
\]  

(67)

(68)

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

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

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

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

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

**Table 16:** Approximate root of \( f_{10}(t) \).

<table>
<thead>
<tr>
<th>Exact root upto 5 D.P</th>
<th>Approximated roots upto 29 D.P</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.973 90</td>
<td>-0.973 906 528 694 964 562 608 263 787 4</td>
</tr>
<tr>
<td>-0.865 06</td>
<td>-0.865 063 366 639 899 953 319 691 516 0</td>
</tr>
<tr>
<td>-0.679 40</td>
<td>-0.679 409 568 298 853 873 661 085 719 5</td>
</tr>
<tr>
<td>-0.433 39</td>
<td>-0.433 395 394 129 247 191 042 119 329 1</td>
</tr>
<tr>
<td>-0.148 87</td>
<td>-0.148 874 338 981 631 210 884 826 001 1</td>
</tr>
<tr>
<td>0.148 87</td>
<td>0.148 874 338 981 631 210 884 826 001 1</td>
</tr>
<tr>
<td>0.433 39</td>
<td>0.433 395 394 129 247 191 042 119 329 1</td>
</tr>
<tr>
<td>0.679 40</td>
<td>0.679 409 568 298 853 873 661 085 719 5</td>
</tr>
<tr>
<td>0.865 06</td>
<td>0.865 063 366 639 899 953 319 691 516 0</td>
</tr>
<tr>
<td>0.973 90</td>
<td>0.973 906 528 694 964 562 608 263 787 4</td>
</tr>
</tbody>
</table>

**Table 17:** Approximate root of (71) using MMN14D.

<table>
<thead>
<tr>
<th>Exact root upto 5 D.P</th>
<th>Approximated roots upto 26 D.P</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1.043 43 - 0.13 335i</td>
<td>-1.043 435 442 746 761 499 747 031 756 - 0.1 333 524 610 738 184 557 261 497 085i</td>
</tr>
<tr>
<td>-1.043 43 + 0.13 335i</td>
<td>-1.043 435 442 746 761 499 747 031 756 + 0.1 333 524 610 738 184 557 261 497 085i</td>
</tr>
<tr>
<td>-0.700 72 - 0.35 611i</td>
<td>-0.700 723 075 598 996 210 090 769 046 1 - 0.3 561 127 938 766 541 462 037 607 779i</td>
</tr>
<tr>
<td>-0.700 72 + 0.35 611i</td>
<td>-0.700 723 075 598 996 210 090 769 046 1 + 0.3 561 127 938 766 541 462 037 607 779i</td>
</tr>
<tr>
<td>-0.137 55 - 0.45 042i</td>
<td>-0.137 557 723 741 216 815 585 324 455 9 + 0.4 504 238 943 083 516 027 302 894 733i</td>
</tr>
<tr>
<td>-0.137 55 + 0.45 042i</td>
<td>-0.137 557 723 741 216 815 585 324 455 9 - 0.4 504 238 943 083 516 027 302 894 733i</td>
</tr>
<tr>
<td>0.460 25 - 0.38 015i</td>
<td>0.460 253 798 901 354 630 188 644 377 5 - 0.3 801 561 966 365 592 562 991 798 562i</td>
</tr>
<tr>
<td>0.460 25 + 0.38 015i</td>
<td>0.460 253 798 901 354 630 188 644 377 5 + 0.3 801 561 966 365 592 562 991 798 562i</td>
</tr>
<tr>
<td>0.895 14 - 0.18 245i</td>
<td>0.895 146 653 711 935 684 708 165 091 0 - 0.1 824 543 520 825 548 326 683 582 900i</td>
</tr>
<tr>
<td>0.895 14 + 0.18 245i</td>
<td>0.895 146 653 711 935 684 708 165 091 0 - 0.1 824 543 520 825 548 326 683 582 900i</td>
</tr>
</tbody>
</table>
Putting values of $f_9(t)$, \ldots, $f_{10}(t)$ in (69), we have

\[ f_{11}(t) = \frac{46189}{256} t^{10} + \frac{12155}{64} t^{9} - \frac{70785}{256} t^{8} - \frac{4719}{16} t^{7} + \frac{18249}{128} t^{6} + \frac{4977}{32} t^{5} + \frac{2905}{128} t^{4} - \frac{415}{16} t^{3} + \frac{1041}{256} t^{2} + \frac{347}{64} t + \frac{2083}{256} = 0, \]

or (71)

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

\[ t_1^{(0)} = -1.04, t_2^{(0)} = -1.04, t_3^{(0)} = -0.7, t_4^{(0)} = -0.7, t_5^{(0)} = -0.13, \]
\[ t_6^{(0)} = 1.13, t_7^{(0)} = 0.4, t_8^{(0)} = 0.4, t_9^{(0)} = 1.8, t_{10}^{(0)} = 0.8. \]

(72)

Figure 11 shows the residual graph for approximating roots of nonlinear function $f_{11}(t)$ using simultaneous methods PJM10, 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 PJM10, 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 PJM10 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