A third-order modification of Newton’s method for multiple roots

Changbum Chun, Beny Neta

A R T I C L E   I N F O

Keywords:
Newton’s method
Multiple roots
Iterative methods
Nonlinear equations
Order of convergence
Root-finding

ABSTRACT

In this paper, we present a new third-order modification of Newton’s method for multiple roots, which is based on existing third-order multiple root-finding methods. Numerical examples show that the new method is competitive to other methods for multiple roots.

1. Introduction

Solving nonlinear equations is one of the most important problems in numerical analysis. In this paper, we consider iterative methods to find a multiple root \(a\) of multiplicity \(m\), i.e., \(f^{(j)}(x) = 0, j = 0, 1, \ldots, m - 1\), and \(f^{(m)}(x) \neq 0\), of a nonlinear equation \(f(x) = 0\).

Newton’s method is only of first order unless it is modified to gain the second order of convergence, see Schröder [1]. This modification requires a knowledge of the multiplicity. Traub [2] has suggested to use any method for \(f^{(m)}(x)\) or \(g(x) = f(x)/f_0(x)\). Any such method will require higher derivatives than the corresponding one for simple zeros. Also the first one of those methods require the knowledge of the multiplicity \(m\). In such a case, there are several other methods developed by Hansen and Patrick [3], Victory and Neta [4], Dong [5,6], Neta and Johnson [7], Neta [8] and Werner [9]. See also Neta [10]. Since in general one does not know the multiplicity, Traub [2] suggested a way to approximate it during the iteration. The way it is done is by evaluating the quotient

\[
\frac{x_{n-2} - x_n}{x_n - x_{n-1}}
\]

and rounding the number up.

For example, the quadratically convergent modified Newton’s method is (see [1])

\[
x_{n+1} = x_n - \frac{f_n}{f_0}
\]

(1)


\[
x_{n+1} = x_n - \frac{f_n}{\frac{m+1}{2m} f_n^2 - \frac{m}{2m} f_n}
\]

(2)

where \(f_0^{(i)}\) is short for \(f^{(i)}(x_n)\). Another third-order method was developed by Victory and Neta [4] and based on King’s fifth order method (for simple roots) [12].
\[ y_n = x_n - u_n, \]
\[ x_{n+1} = y_n - f(y_n) \frac{f_n + A f(y_n)}{f_n} \]
where
\[ A = \mu^{m-1} - \mu^{m+1}, \]
\[ B = -\mu^m (m-2)(m-1) + 1, \]
\[ \mu = \frac{m}{m-1} \]
and
\[ u_n = \frac{f_n}{f_n}. \]

Dong [5] has developed two third-order methods requiring two evaluations of \( f \) and one evaluation of \( f' \)
\[
\begin{align*}
    y_n &= x_n - \sqrt{m} u_n, \\
    x_{n+1} &= y_n - m \left( 1 - \frac{1}{m} \right)^{\frac{1}{m}} \frac{f(y_n)}{f_n}, \\
    y_n &= x_n - u_n, \\
    x_{n+1} &= y_n + \frac{u_n f(y_n)}{f(y_n) - (1 - \frac{1}{m}) f_n},
\end{align*}
\]
where \( u_n \) is given by (6).

Yet two other third-order methods developed by Dong [6], both require the same information and both based on a family of fourth order methods (for simple roots) due to Jarratt [13]:
\[
\begin{align*}
    y_n &= x_n - u_n, \\
    x_{n+1} &= y_n - \frac{u_n f(y_n)}{(m-1) f(y_n) + m f_n}, \\
    y_n &= x_n - \frac{m}{m-1} u_n, \\
    x_{n+1} &= y_n - \frac{u_n f(y_n)}{(1 + \frac{m}{n}) f(y_n) - f_n},
\end{align*}
\]
where \( u_n \) is given by (6).

Osada [14] has developed a third-order method using the second derivative,
\[ x_{n+1} = x_n - \frac{1}{2} m(m+1) u_n + \frac{1}{2} (m-1)^2 \frac{f_n}{f_n}, \]
where \( u_n \) is given by (6).

Neta and Johnson [7] have developed a fourth order method requiring one function- and three derivative-evaluation per step. The method is based on Jarratt’s method [15] given by the iteration
\[ x_{n+1} = x_n - \frac{f_n}{a f_n' + a_2 f(y_n) + a_3 f'(\eta_n)}, \]
where \( u_n \) is given by (6) and
\[ y_n = x_n - u_n, \]
\[ u_n = \frac{f_n}{f_n'}, \]
\[ \eta_n = x_n - b u_n - c u_n. \]

Neta and Johnson [7] give a table of values for the parameters \( a, b, c, a_1, a_2, a_3 \) for several values of \( m \). In the case \( m = 2 \) they found a method that will require only two derivative-evaluations \( (a_3 = 0) \). This was not possible for higher \( m \).

Neta [8] has developed a fourth order method requiring one function- and three derivative-evaluation per step. The method is based on Murakami’s method [16] given by the iteration
\[ x_{n+1} = x_n - a_1 u_n - a_2 v_n - a_3 w_3(x_n) - \psi(x_n), \]
where \( u_n \) is given by (6), \( v_n, y_n, \) and \( \eta_n \) are given by (13) and
\[ w_3(x_n) = \frac{f_n}{f(\eta_n)}, \]
\[ \psi(x_n) = \frac{f_n}{b f_n' + b f(y_n)}. \]
Neta [8] gives a table of values for the parameters $a, b, c, a_1, a_2, a_3, b_1, b_2$ for several values of $m$.

A method of order 1.5 requiring two function- and one derivative-evaluation is given by Werner [9]. It is only for double roots

$$ y_n = x_n - u_n, $$

$$ x_{n+1} = x_n - s_n u_n, $$

where

$$ s_n = \begin{cases} \frac{2}{1+\sqrt{1-4f(x_n)/s_n}}, & \text{if } f(y_n)/f_n \leq \frac{1}{4}, \\ \frac{1}{4}f_n/f(y_n), & \text{otherwise}. \end{cases} $$

Later we give a table comparing the efficiency of all known methods for multiple roots including our new ones we develop here. The informational efficiency, $E$, of a method of order $p$ using $d$ function/derivative evaluations is defined [2] as

$$ E = \frac{p}{d}. $$

The efficiency index, $I$, is defined as

$$ I = p^{1/d}. $$

It can be seen in Table 1 that our new method is competitive with the previously developed schemes.

There is an approach in [17] that uses any pair of existing methods in constructing new iterative methods or families of the same order in the case of simple roots. Now, a simple question arises: does the approach also work for multiple roots case? The answer indeed is affirmative as will be seen in the following section. To this end and for the sake of illustration, two methods for multiple roots are considered. One is Osada’s third-order method (11) and the other is the Euler–Chebyshev method of order three [2]

$$ x_{n+1} = x_n - \frac{m(3 - m)}{2} \frac{f(x_n)}{f'(x_n)} - m^2 \frac{f(x_n)^2 f''(x_n)}{2 f'(x_n)^3}. $$

The methods (11) and (19) are used to obtain a new modification of Newton’s method for multiple roots. The new method is shown to be cubically convergent by analysis of convergence, its performance and practical utility are demonstrated by numerical examples.

2. Development of method and convergence analysis

Now, we approximately equate the correcting terms of both methods (11) and (19) to obtain the following approximate expression:

$$ -\frac{1}{2} m(m + 1) \frac{f(x_n)}{f'(x_n)} + \frac{1}{2} (m - 1) \frac{f'(x_n)}{f''(x_n)} \approx -\frac{m(3 - m)}{2} \frac{f(x_n)}{f'(x_n)} - m^2 \frac{f(x_n)^2 f''(x_n)}{2 f'(x_n)^3}, $$

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$p$</th>
<th>$d$</th>
<th>$E$</th>
<th>$I$</th>
<th>$f_0$</th>
<th>$f_0^2$</th>
<th>$f_0^3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Werner [9] (16) $m = 2$</td>
<td>1.5</td>
<td>3</td>
<td>0.5</td>
<td>1.145</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Schröder [1] (1)</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1.414</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hansen and Patrick [3]</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1.442</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Halley (2)</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1.442</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Laguerre</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1.442</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hansen and Patrick [3]</td>
<td>3</td>
<td>4</td>
<td>.75</td>
<td>1.316</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Victory and Neta [4] (3)</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1.442</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dong [5] (7), (8)</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1.442</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dong [6] (9), (10)</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1.442</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Osada [14]</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1.442</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Neta and Johnson [7] (12) $m = 2$</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>1.414</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Neta and Johnson [7] (12) $m = 2$</td>
<td>4</td>
<td>3</td>
<td>1.333</td>
<td>1.587</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Neta [8]</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>1.414</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Neta [19] $m = 3$</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1.442</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Neta [19] $m = 3$</td>
<td>2</td>
<td>3</td>
<td>.667</td>
<td>1.259</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Neta [19]</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1.442</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Neta [19]</td>
<td>2.732</td>
<td>2</td>
<td>1.366</td>
<td>1.653</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Neta [19]</td>
<td>2.732</td>
<td>2</td>
<td>1.366</td>
<td>1.653</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chun and Neta (22)</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1.442</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

the informational efficiency is  
\[ \frac{f(x_0)f''(x_0)}{2f'(x_0)} = \frac{m - 1}{m} f'(x_0) \]  
which indicates that the order of convergence of the methods defined by (22) is at least three. This completes the proof.

We now use Maple [18] to collect all these expansions into (22) to have
\[ e_{n+1} = \frac{(m^2 + 3)C_1^2 - 2m(m - 1)C_2 e_n^3 + O(e_n^4)}{2m^2(m - 1)}, \]
where \( e_n = x_n - x \) and \( C_j = \frac{m!}{(m+j)!} \frac{f^{(m+j)}(x_0)}{f^{(m)}(x_0)} \).

**Proof.** Using Taylor expansion of \( f(x_n) \) about \( x \), we have
\[ f(x_n) = f^{(m)}(x) \frac{e_n^m}{m!} + 1 + C_1 e_n + C_2 e_n^2 + O(e_n^3), \]
\[ f'(x_n) = f^{(m)}(x) \frac{e_n^{m-1}}{(m-1)!} \left[ 1 + \frac{m + 1}{m} C_1 e_n + \frac{m + 2}{m} C_2 e_n^2 + O(e_n^3) \right], \]
\[ f''(x_n) = f^{(m)}(x) \frac{e_n^{m-2}}{(m-2)!} \left[ 1 + \frac{m + 1}{m} C_1 e_n + \frac{(m + 1)(m + 2)}{m(m - 1)} C_2 e_n^2 + O(e_n^3) \right], \]
where \( C_j = \frac{m!}{(m+j)!} \frac{f^{(m+j)}(x_0)}{f^{(m)}(x_0)} \) and \( e_n = x_n - x \).

Dividing (24) by (25) gives us
\[ f(x_n) f''(x_n) = e_n \left[ 1 - \frac{1}{m} C_1 e_n + \frac{(m + 1)C_1^2 - 2mC_2 e_n^2 + O(e_n^3)}{m^2} \right]. \]

We now use Maple [18] to collect all these expansions into (22) to have
\[ m(3 - m)f(x_n)f''(x_n) + (m - 1)^2 f'(x_n)^2 = \frac{e_n^{2m-2}}{(m - 1)!} \left[ 2(m - 1) + \frac{2m^3 + 2m^2 + 4m - 2m + 10}{m} C_1 e_n \right. \]
\[ + \frac{4m^2 + 4m - 2m + 10}{m} C_2 e_n^2 + \left. \frac{4m - 6}{m} C_3 e_n^3 \right], \]
\[ 2m^2f(x_n)f''(x_n) = \frac{e_n^{2m-2}}{(m - 1)!} \left[ 2m(m - 1) + 4m^2 C_1 e_n + \frac{2m(m + 1)C_1^2 + 4(m^2 + m + 1)C_2}{m} e_n^2 \right. \]
\[ + \left. \frac{4(m^2 + 2m + 3)C_3}{m} (m + 1)^2 C_2 e_n^3 \right]. \]

Now divide (30) by (29) and multiply by (28) and collect terms we have
\[ e_{n+1} = \frac{(m^2 + 3)C_1^2 - 2m(m - 1)C_2}{2m^2(m - 1)} e_n^3 + O(e_n^4), \]
which indicates that the order of convergence of the methods defined by (22) is at least three. This completes the proof. □

Note that the method requires one evaluation of the function, the first and the second derivative at each step, therefore
the informational efficiency is \( E = 1 \) and the efficiency index is \( I = 1.442. \)

In a similar fashion, the proposed approach can be continuously applied to produce various types of new approximations from
the other iterative methods for multiple roots, which can in turn be freely employed to find many new iterative methods or families for multiple roots.
3. Numerical examples

We present some numerical test results for various third-order multiple root-finding methods as well as our new methods and the Newton method in Table 3. Compared were the Newton method (1) (NM), Halley-like method (2) (HM), Osada’s method (11) (OM), the Euler–Chebyshev method (19) (ECM), and the method (22) (CM) introduced in this contribution. All computations were done using MAPLE with 128 digit floating point arithmetics \( Digits = 128 \). Displayed in Table 3 are the number of iterations (IT) required such that \( |f(x_n)| < 10^{-15} \), and the value of \( |f(x_n)| \) after the required iterations.

The following functions are used for the comparison and we display the approximate zeros \( x_i \) found up to the 28th decimal places

\[
f_1(x) = (x^3 + 4x^2 - 10)^3, \quad x_1 = 1.3652300134140968457608068290,
\]

\[
f_2(x) = (\sin^2 x - x^2 + 1)^2, \quad x_2 = 1.4044916482153412260350868178,
\]

\[
f_3(x) = (x^2 - e^x - 3x + 2)^5, \quad x_3 = 0.25753028543986076045536730494,
\]

\[
f_4(x) = (\cos x - x)^3, \quad x_4 = 0.82399851323156160416553128767,
\]

\[
f_5(x) = ((x - 1)^3 - 1)^6, \quad x_5 = 2,
\]

\[
f_6(x) = (xe^x - \sin^2 x + 3\cos x + 5)^4, \quad x_6 = -1.2076478271309189270094167584,
\]

\[
f_7(x) = (\sin x - x/2)^2, \quad x_7 = 1.8954942670339809471440357381.
\]

We also ran the second case with various initial points and average the number of iterations required for convergence. These results are given in Table 2.

The results presented in Tables 2 and 3 show that for the functions we tested, the new method introduced here have at least equal performance as compared to the other multiple root-finding methods of the same order, and also converge more rapidly than Newton’s method for multiple roots.

### Table 2

Comparison of various third-order multiple root-finding methods and Newton’s method. The last row gives the average number of iterations required for convergence when starting from various initial guesses.

<table>
<thead>
<tr>
<th>( x_0 )</th>
<th>NM</th>
<th>HM</th>
<th>OM</th>
<th>ECM</th>
<th>CM</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0001</td>
<td>46</td>
<td>23</td>
<td>37</td>
<td>68</td>
<td>35</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>11.6</td>
<td>7.6</td>
<td>9.3</td>
<td>12.1</td>
<td>8.9</td>
<td></td>
</tr>
</tbody>
</table>

### Table 3

Comparison of various third-order multiple root-finding methods and Newton’s method.

| \( f(x) \) | IT (|\( f(x_n) \)|) |
|---|---|
| \( f_1(x_0) = 2 \) | 6(8.49e-54) | 4(7.06e-49) | 4(6.47e-33) | 4(4.01e-38) | 4(4.01e-38) |
| \( f_1(x_0) = 1 \) | 6(491e-62) | 5(3.46e-57) | 5(3.50e-84) | 4(1.94e-38) | 4(1.94e-38) |
| \( f_2(x_0) = 2.3 \) | 7(7.31e-52) | 5(4.98e-57) | 5(2.02e-38) | 5(1.73e-47) | 5(4.55e-42) |
| \( f_2(x_0) = 2 \) | 7(5.61e-64) | 5(7.43e-77) | 5(3.35e-51) | 5(1.53e-63) | 5(4.09e-56) |
| \( f_3(x_0) = 0 \) | 4(1.03e-55) | 3(1.68e-53) | 3(5.83e-62) | 3(4.31e-58) | 3(1.71e-55) |
| \( f_3(x_0) = 1 \) | 4(3.46e-52) | 4(1.39e-85) | 4(2.01e-91) | 4(2.24e-89) | 4(1.93e-87) |
| \( f_4(x_0) = 1.7 \) | 5(6.04e-47) | 4(9.12e-43) | 4(1.17e-39) | 4(5.25e-41) | 4(5.25e-41) |
| \( f_4(x_0) = 1 \) | 5(1.02e-60) | 1(4.18e-85) | 4(1.42e-78) | 4(1.43e-81) | 4(1.43e-81) |
| \( f_5(x_0) = 3 \) | 6(2.70e-45) | 4(7.44e-45) | 5(3.12e-85) | 5(1.89e-94) | 4(3.56e-37) |
| \( f_5(x_0) = -1 \) | 10(5.23e-49) | 11(2.22e-65) | 24(7.0e-44) | 23(1.87e-52) | 5(2.67e-77) |
| \( f_6(x_0) = -2 \) | 8(5.62e-37) | 5(1.06e-61) | 6(5.09e-45) | 6(3.21e-64) | 6(2.83e-82) |
| \( f_6(x_0) = -1 \) | 6(5.61e-60) | 3(4.75e-35) | 5(1.56e-103) | 4(1.47e-47) | 4(9.70e-58) |
| \( f_7(x_0) = 1.7 \) | 6(3.80e-57) | 4(7.40e-47) | 5(1.81e-76) | 4(1.01e-37) | 5(1.03e-92) |
| \( f_7(x_0) = 2 \) | 5(2.09e-40) | 4(1.55e-65) | 4(3.45e-53) | 4(1.67e-59) | 4(8.23e-56) |
References