New third order nonlinear solvers for multiple roots

B. Neta

Naval Postgraduate School, Department of Applied Mathematics, Monterey, CA 93943, United States

Abstract

Two third order methods for finding multiple zeros of nonlinear functions are developed. One method is based on Chebyshev’s third order scheme (for simple roots) and the other is a family based on a variant of Chebyshev’s which does not require the second derivative. Two other more efficient methods of lower order are also given. These last two methods are variants of Chebyshev’s and Osada’s schemes. The informational efficiency of the methods is discussed. All these methods require the knowledge of the multiplicity.

Published by Elsevier Inc.

Keywords: Nonlinear equations; High order; Multiple roots; Fixed point; Chebyshev; Osada

1. Introduction

There is a vast literature on the solution of nonlinear equations and nonlinear systems, see for example Ostrowski [1], Traub [2], Neta [3] and references there. Here we develop several high order fixed point type methods to approximate a multiple root. There are several methods for computing a zero $\xi$ of multiplicity $m$ of a nonlinear equation $f(x) = 0$, see Neta [3]. Newton’s method is only of first order unless it is modified to gain the second order of convergence, see Rall [4] or Schröder [5]. This modification requires a knowledge of the multiplicity. Traub [2] has suggested to use any method for $f^{(m)}(x)$ or $g(x) = \frac{f(x)}{f'(x)}$. 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 [6], Victory and Neta [7], Dong [8,9], Neta and Johnson [10], Neta [11] and Werner [12]. Since in general one does not know the multiplicity, Traub [2] suggested a way to approximate it during the iteration.

For example, the quadratically convergent modified Newton’s method is (see [5])

$$x_{n+1} = x_n - \frac{f_n f_{n+1}}{f_n}$$

and the cubically convergent Halley’s method [13] is a special case of the Hansen and Patrick’s method [6].

E-mail address: bneta@nps.edu

doi:10.1016/j.amc.2008.01.031

\[ x_{n+1} = x_n - \frac{f_n}{2m} f'_{n+1} - \frac{f_n^2}{2f''_{n+1}}, \]  
(2)

where \( f^{(i)} \) is short for \( f^{(i)}(x_n) \). Another third order method was developed by Victory and Neta [7] and based on King’s fifth order method (for simple roots) [14]

\[ y_n = x_n - u_n, \]
\[ x_{n+1} = y_n - \frac{f(y_n) f_n + Af(y_n)}{f'_n} f_n + Bf(y_n), \]  
(3)

where
\[ A = \mu^{2m} - \mu^{m+1}, \]
\[ B = -\frac{\mu^m(m-2)(m-1) + 1}{(m-1)^2}, \]  
(4)
\[ \mu = \frac{m}{m-1}, \]  
(5)

and
\[ u_n = \frac{f_n}{f'_n}. \]  
(6)

Dong [8] has developed two third order methods requiring two evaluations of \( f \) and one evaluation of \( f' \)

\[
\begin{aligned}
\{ y_n &= x_n - \sqrt{mu_n}, \\
n_{n+1} &= y_n - m\left(1 - \frac{1}{\sqrt{m}}\right) \frac{f(y_n)}{f'_n}, \\
y_n &= x_n - u_n, \\
n_{n+1} &= y_n + \frac{u_n f(y_n)}{f(y_n) - (1-\frac{1}{\sqrt{m}}) f'_n}, \\
\}
\]  
(7)

where \( u_n \) is given by (6).

Yet two other third order methods developed by Dong [9], both require the same information and both based on a family of fourth order methods (for simple roots) due to Jarratt [15]:

\[
\begin{aligned}
\{ y_n &= x_n - u_n, \\
n_{n+1} &= y_n - \frac{f(y_n)}{(m+1)^{m+1} f(y_n) + \frac{m^2 f(y_n)}{(m-1)^2}} \frac{f'_n}{f''_{n+1}}, \\
y_n &= x_n - \frac{m}{m+1} u_n, \\
n_{n+1} &= y_n - \frac{m u_n f(y_n)}{f(y_n) - (1+\frac{1}{m}) f'(y_n) f'_n}, \\
\}
\]  
(9)

where \( u_n \) is given by (6).

Osada [16] has developed a third order method using the second derivative,

\[ x_{n+1} = x_n - \frac{1}{2} m(m+1) u_n + \frac{1}{2} (m-1)^2 \frac{f_n''}{f'_n}, \]  
(11)

where \( u_n \) is given by (6).

Neta and Johnson [10] have developed a fourth order method requiring one function- and three derivative-evaluation per step. The method is based on Jarratt’s method [17] given by the iteration

\[ x_{n+1} = x_n - \frac{f}{a_1 f' + a_2 f''(y_n) + a_3 f''(\eta_n)}, \]  
(12)

where \( u_n \) is given by (6) and

\[ y_n = x_n - a u_n, \]
\[ v_n = \frac{f_n}{f'(y_n)}, \]
\[ \eta_n = x_n - b u_n - c v_n. \]

Neta and Johnson [10] 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 [11] has developed a fourth order method requiring one function- and three derivative-evaluation per step. The method is based on Murakami’s method [18] given by the iteration

\[ x_{n+1} = x_n - a_1 u_n - a_2 v_n - a_3 w_3(x_n) - \psi(x_n), \] (14)

where \( u_n \) is given by (6), \( v_n \), \( y_n \), and \( \eta_n \) are given by (13) and

\[ w_3(x_n) = \frac{f_n}{f'(\eta_n)}, \]
\[ \psi(x_n) = \frac{b_1 f''_n + b_2 f'''(y_n)}{f'(y_n)}. \] (15)

Neta [11] 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 [12]. It is only for double roots

\[ y_n = x_n - u_n, \]
\[ x_{n+1} = x_n - s_n u_n, \] (16)

where

\[ s_n = \begin{cases} \frac{2}{1 + \sqrt{1 - 4 f(y_n)/f_n}} & \text{if } f(y_n)/f_n \leq \frac{1}{3}, \\ \frac{1}{2} f_n/f(y_n) & \text{otherwise}. \end{cases} \]

Later we give a table comparing the efficiency of these methods and of our new ones we develop here.

### 2. A new third order scheme

We would like to develop a new method for multiple roots based on Chebyshev’s method (see [19–21]). Chebyshev’s method is given for simple roots by

\[ x_{n+1} = x_n - u_n \left[ 1 + \frac{1}{2} u_n \frac{f_n'}{f_n} \right]. \] (17)

Let us take the following two-parameter family

\[ x_{n+1} = x_n - a u_n \left[ 1 + \beta u_n \frac{f_n'}{f_n} \right]. \] (18)

We now show how to choose the parameters \( a, \beta \) so that the method is of third order for the case of multiple roots.

Expand \( f(x_n) \) and \( f'(x_n) \) in Taylor series (truncated after the \( N \)th power, \( N > m \)) about the root \( \xi \), we have

\[ f(x_n) = f(x_n - \xi + \xi) = f(\xi + e_n) = \frac{f(m)(\xi)}{m!} \left( e_n^m + \sum_{i=m+1}^{N} A_i e_n^i \right) \] (19)

or...
\[ f(x_n) = \frac{f^{(m)}(\xi)}{m!} e_n^m \left( 1 + \sum_{i=m+1}^{N} B_{i-m} e_n^{i-m} \right), \] (20)

where

\[ A_i = \frac{m! f^{(i)}(\xi)}{i! f^{(m)}(\xi)}, \quad i > m, \] (21)

\[ B_{i-m} = A_i, \]

and

\[ e_n = x_n - \xi, \] (22)

\[ f'(x_n) = \frac{f^{(m)}(\xi)}{(m-1)!} e_n^{m-1} \left( 1 + \sum_{i=m+1}^{N} \frac{i}{m} B_{i-m} e_n^{i-m} \right). \] (23)

Then \( u_n \) will be (neglecting terms of order higher than 3)

\[ u_n = \frac{1}{m} e_n - \frac{B_1}{m^2} e_n^2 + \frac{(m+1)B_1^2 - 2mB_2}{m^3} e_n^3. \] (24)

Now expand the second derivative and get \( u_n f_n'' \)

\[ f''(x_n) = \frac{f^{(m)}(\xi)}{(m-2)!} e_n^{m-2} \left( 1 + \frac{m+1}{m-1} B_1 e_n + \frac{(m+1)(m+2)}{m(m-1)} B_2 e_n^2 + \frac{(m+2)(m+3)}{m(m-1)} B_3 e_n^3 \right), \] (25)

\[ u_n f_n'' = \frac{m-1}{m} + \frac{2}{m^2} B_1 e_n - \frac{3}{m^3} \left( \frac{(m+1)B_1^2 - 2mB_2}{m^3} e_n^2 + \frac{(m+3)(m+1)^2 B_1^3 - (3m+4)(m+3)mB_1B_2 + 3m^2(m+3)B_3}{m^4} e_n^3 \right). \] (26)

Now substitute (24) and (26) in (18)

\[ e_{n+1} = -\alpha m + m^2 + \alpha \beta (1 - m) e_n^2 + \frac{\alpha B_1 ((m-3) \beta + m)}{m^3} e_n^3 + \left[ \frac{2\alpha (m + \beta (m - 4)) B_2}{m^4} - \frac{\alpha (m(m+1) + \beta (m^2 - 3m - 6)) B_1^2}{m^4} \right] e_n^3. \] (27)

If we choose \( m \neq 3 \)

\[ \alpha = -\frac{m(m-3)}{2}, \] (28)

\[ \beta = -\frac{m}{m-3}, \]

we have the third order method

\[ x_{n+1} = x_n + \frac{m(m-3)}{2} u_n \left[ 1 - \frac{m}{m-3} u_n f_n'' \right] f_n'' \] (29)

with the error at \( x_{n+1} \) becomes

\[ e_{n+1} = \left( -\frac{1}{m} B_2 + \frac{3 + m}{2m^2 - B_1^2} \right) e_n^3. \] (30)

Clearly the case \( m = 3 \) need to be considered separately. In this case the method is only second order with the choice

\[ \beta = \frac{3 - \alpha}{\alpha} \] (31)
\[ x_{n+1} = x_n - \alpha u_n \left[ 1 + \frac{3 - \alpha}{\alpha} u_n \frac{f''}{f_n} \right] \] (32)

with the error at \( x_{n+1} \) becomes
\[ e_{n+1} = \frac{\alpha}{9} B_1 e_n^2 + \left( \frac{\alpha - 1}{3} B_2 + \frac{\alpha}{2} \frac{B_1}{27} \right) e_n^3. \] (33)

We can annihilate one of the terms in the coefficient of \( e_n^3 \) by choosing \( \alpha = 1 \) or \( \alpha = \frac{3}{4} \), but the method is only second order.

3. New methods not requiring second derivative

Here we develop 3 new methods not requiring second derivative. First is a third order method based on a modification of Chebyshev’s method. The one-parameter family of modified Chebyshev methods (see Kou and Li [23]) is given by
\[ x_{n+1} = x_n - u_n \left[ \frac{\theta^2 + \theta - 1}{\theta^2} + f(y_n) \frac{f''}{f_n} \right], \] (34)
where
\[ y_n = x_n - \theta u_n. \] (35)

Two special cases were discussed in [23]. The choice \( \theta = 1 \) yields the method
\[ x_{n+1} = x_n - u_n \left[ 1 + \frac{f(x_n - u_n)}{f_n} \right]. \] (36)

Another possibility is to choose \( \theta \) so that
\[ \theta^2 + \theta - 1 = 0 \]
i.e.
\[ \theta = -\frac{1 \pm \sqrt{5}}{2}. \] (37)

In this case, the method is
\[ x_{n+1} = x_n - \frac{2}{3 - \sqrt{5}} \frac{f(x_n - \theta u_n)}{f_n}. \] (38)

We now construct a family similar to (34) for the case of multiple roots. Let
\[ x_{n+1} = x_n - u_n \left[ \beta + \gamma f(y_n) \right], \] (39)
where
\[ y_n = x_n - \alpha u_n. \] (40)

Using Maple [22] to get all the Taylor series necessary, we have (up to third order terms)
\[ e_{n+1} = c_1 e_n + c_2 e_n^2 + c_3 e_n^3, \] (41)
where
\[ c_1 = \frac{m - \beta - \gamma \rho}{m}, \]
\[ c_2 = \frac{B_1 (m - \alpha) (\gamma \rho + \beta) - \gamma \rho x^2}{m^2 (m - \alpha)}. \] (42)
and
\[
\rho = \left(\frac{m - \alpha}{m}\right)^m.
\] (43)

Before we list \(c_3\), we choose \(\beta\) and \(\gamma\) to annihilate the coefficients \(c_1\) and \(c_2\)
\[
\beta = m - \frac{m(m - \alpha)}{\alpha^2},
\]
\[
\gamma = \frac{m(m - \alpha)}{\rho \alpha^2}.
\] (44)

In this case, we have
\[
c_3 = -\frac{m - \alpha}{m^2} B_2 + \frac{1}{2} \frac{m(m + 3) - 2\alpha(m + 1)}{m^2(m - \alpha)} B_1^2.
\] (45)

We can choose the last parameter, \(\alpha\), to annihilate the coefficient of \(B_1^2\), i.e.
\[
\alpha = 1 \frac{m(m + 3)}{2} \frac{1}{m + 1}.
\] (46)

The choice \(\alpha = m\) is not possible. Using the value of \(\alpha\) from (46) in (44), we have the following:
\[
\beta = \frac{m^3 + 4m^2 + 9m + 2}{(m + 3)^2},
\]
\[
\gamma = \frac{2^{m+1}(m^2 - 1)}{(m + 3)^2\left(\frac{m-1}{m+1}\right)}^m,
\]
\[
e_{n+1} = -\frac{1}{2} \frac{m - 1}{m(m + 1)} B_2 e_n^3.
\] (47)

The one parameter family (39), (40), and (44) is third order requiring two function- and one derivative-evaluation per step.

The second new method is based on the approximation given by Neta [24] for the second derivative in (29), i.e.
\[
f_n'' = \frac{6}{h^2}(f_{n-1} - f_n) + \frac{2}{h} f_n'_{n-1} + \frac{4}{h^2} f_n'_{n},
\] (48)

where \(h = x_n - x_{n-1}\). This will give the following method:
\[
x_{n+1} = x_n + \frac{m(m - 3)}{2} u_n \left[1 - \frac{m}{m - 3} u_n w(x_n)\right],
\] (49)

where
\[
w(x_n) = \frac{6(f_{n-1} - f_n) + 2h f_n'_{n-1} + 4h f_n'_{n}}{h^2 f_n'}.
\] (50)

This modified method is of order 2.732 for \(m \neq 3\) and it requires one function- and one derivative-evaluation per step. It also requires an additional starting value which we can be obtained using Newton’s method (first derivative is required anyway).

The third new method is based on the approximation (48) for the second derivative in (11), i.e.
\[
x_{n+1} = x_n - \frac{m(m + 1)}{2} u_n + \frac{(m - 1)^2}{2w(x_n)},
\] (51)

where \(w(x_n)\) is given by (50). This modified method is also of order 2.732 and it requires one function- and one derivative-evaluation per step. It also requires an additional starting value which we can be obtained using Newton’s method (first derivative is required anyway).

We now give the definitions of informational efficiency (see Traub [2])
\[ E = \frac{p}{d} \]
and efficiency index
\[ I = \frac{p^{1/d}}{d}. \]
where \( p \) is the order of the method and \( d \) is the number of function (and derivatives) evaluations per step. Clearly it is assumed that the cost of evaluating a function or any of the derivatives required is identical.

In Table 1 we list all methods known (to the author) for finding roots with multiplicity \( m \). It can be seen that the informational efficiency is almost always unity. The efficiency index for our methods here is the highest for a general \( m \). Of the two third order methods developed here, we prefer the second which has the same efficiency independent of the multiplicity. These two schemes are still preferable over the fourth order methods given by Neta and Johnson [10] and Neta [11] because we have a closed form formula for the parameters. The last two methods are of order lower than three but of higher efficiency.

4. Numerical experiments

In this section we report on numerical experiments using Halley’s method (2), Chebyshev’s method (29), the modified Chebyshev method (39) (with \( z = m/2 \)) all for multiple roots, the modified method (49) and the modified Osada scheme (51). In our first example we took a quadratic polynomial having a double roots at \( \xi = 1 \)
\[ f(x) = x^2 - 2x + 1. \]
Here we started with \( x_0 = 0 \) and the convergence achieved in 1 iteration for the first 3 methods and 2 iterations for the last two schemes. In the second example we took a polynomial having two double roots at \( \xi = \pm 1 \)
\[ f(x) = x^4 - 2x^2 + 1. \]
Starting at \( x_0 = 0.8 \) Halley’s and Chebyshev’s methods converged in 4 iterations but the modified Chebyshev required 3 iterations. The last two modified schemes required 5 iterations. When we start at \( x_0 = 0.6 \) all the methods required the same number of iterations as before. The results are given in Table 2, where we used \( a(-b) \) to denote \( a \times 10^{-b} \). Note that the modified Chebyshev was consistently better than the first two schemes. The last two schemes are slower since their order is less than three.

Table 1
Comparison of methods for multiple roots

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>( p )</th>
<th>( d )</th>
<th>( E )</th>
<th>( I )</th>
<th>( f' )</th>
<th>( f'' )</th>
<th>( f''' )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Werner [12] (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 [5] (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 [6]</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 [6]</td>
<td>3</td>
<td>4</td>
<td>0.75</td>
<td>1.316</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Victory and Neta [7] (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 [8] (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 [9] (9), (10)</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1.442</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Osada [16]</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 [10] (12) ( m \neq 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 [10] (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 [11]</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 (29) ( m \neq 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 (32) ( m = 3 )</td>
<td>2</td>
<td>3</td>
<td>0.667</td>
<td>1.259</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Neta (39)</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 (49)</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 (51)</td>
<td>2.732</td>
<td>2</td>
<td>1.366</td>
<td>1.653</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The next example is a polynomial with triple root at \( \xi = 1 \). Recall that Chebyshev’s method for \( m = 3 \) has a free parameter \( \alpha \). In this example we used \( \alpha = 3/2 \)

\[
f(x) = x^5 - 8x^4 + 24x^3 - 34x^2 + 23x - 6. \tag{56}
\]

The iteration starts with \( x_0 = 0 \) and the results are summarized in Table 3.

Another example with double root at \( \xi = 0 \) is

\[
f(x) = x^2 e^x. \tag{57}
\]

Starting at \( x_0 = 0.1 \) or at \( x_0 = 0.2 \) the first three methods converged in 2 iterations and the last two required 4–5 iterations. The results are given in Table 4.

### Table 2
Comparison of 5 methods for 4 different initial guesses for Example 2

| \( x_0 \) | Method | \( n \) | \( |f(x_n)| \) | \( n \) | \( |f(x_n)| \) | \( n \) | \( |f(x_n)| \) | \( n \) | \( |f(x_n)| \) |
|---|---|---|---|---|---|---|---|---|---|
| 0.8 | (2) | 4 | 0 | 4 | 0 | 3 | 0 | 5 | 0 |
| 0.6 | (29) | 6(−15) | 4 | 5.8(−15) | 3 | 2(−20) | 5 | 0 | 5 | 0 |

Note that \( a(−b) \) means \( a \times 10^{-b} \).

### Table 3
Results for Example 3

| Method | \( n \) | \( |f(x_n)| \) |
|---|---|---|
| (2) | 3 | 1(−18) |
| (29) | 3 | 1(−15) |
| (39) | 3 | 0 |
| (49) | 7 | 1(−14) |
| (51) | 9 | 0 |

Note that \( a(−b) \) means \( a \times 10^{-b} \).

### Table 4
Results for Example 4

| Method | \( x_0 = 0.1 \) | \( n \) | \( |f(x_n)| \) | \( x_0 = 0.2 \) | \( n \) | \( |f(x_n)| \) |
|---|---|---|---|---|---|---|
| (2) | 2 | 4(−26) | 2 | 8(−21) |
| (29) | 2 | 2(−22) | 2 | 3(−17) |
| (39) | 2 | 2(−22) | 2 | 3(−17) |
| (49) | 4 | 3(−15) | 5 | 5(−26) |
| (51) | 4 | 1(−14) | 5 | 1(−24) |

Note that \( a(−b) \) means \( a \times 10^{-b} \).

### Table 5
Results for Example 5

| Method | \( x_0 = 0 \) | \( n \) | \( |f(x_n)| \) | \( x_0 = 0.5 \) | \( n \) | \( |f(x_n)| \) |
|---|---|---|---|---|---|---|
| (2) | 3 | 1(−18) | 3 | 0 |
| (29) | 5 | 0 | 3 | 0 |
| (39) | 4 | 1(−18) | 3 | 0 |
| (49) | 5 | 0 | 5 | 1(−18) |
| (51) | 5 | 2(−18) | 5 | 1(−18) |

Note that \( a(−b) \) means \( a \times 10^{-b} \).
The last example having a double root at $\xi = 1$ is
\[ f(x) = 3x^4 + 8x^3 - 6x^2 - 24x + 19. \] (58)

Now we started with $x_0 = 0$ and $x_0 = 0.5$ and the results are summarized in Table 5. Again the last two schemes are slower to converge.

5. Conclusions

We have developed four new methods to obtain multiple roots. The first scheme is third order and it requires a special care in the case that the root is triple. The second family does not require the use of the second derivative. Instead we have another function evaluation at an off-step. The other two methods of order 2.732 are more efficient. The numerical experiments show the rapid convergence of the third order methods with a slight edge to (39). The last two methods require slightly more iterations but less function evaluations per step.

References
