A Family of Combined Iterative Methods for Solving Nonlinear Equations

Shuping Chen¹*, Youhua Qian²

¹College of Mathematics, Xiamen University of Technology, xiamen 361024, P. R. China
²College of Mathematics, Physics and Information Engineering, Zhejiang Normal University, Jinhua, 321004, P. R. China
*Corresponding author: shupingchen@126.com

Abstract In this article we construct some higher-order modifications of Newton’s method for solving nonlinear equations, which is based on the undetermined coefficients. This construction can be applied to any iteration formula. It can be found that per iteration the resulting methods add only one additional function evaluation, their order of convergence can be increased by two or three units. Higher order convergence of our methods is proved and corresponding asymptotic error constants are expressed. Numerical examples, obtained using Matlab with high precision arithmetic, are shown to demonstrate the convergence and efficiency of the combined iterative methods. It is found that the combined iterative methods produce very good results on tested examples, compared to the results produced by the existing higher order schemes in the related literature.

Keywords: Newton’s method, combined iterative methods, nonlinear equations, order of convergence, computational efficiency

Cite This Article: Shuping Chen, and Youhua Qian, “A Family of Combined Iterative Methods for Solving Nonlinear Equations.” American Journal of Applied Mathematics and Statistics, vol. 5, no. 1 (2017): 22-32. doi: 10.12691/ajams-5-1-5.

1. Introduction

A variety of problems in different fields of science and technology require to find the solution of nonlinear equations. Iterative methods such as Newton’s method are the most used technique. In this paper, we consider a new family of combined iterative methods to find a simple root ξ of a nonlinear equation \( f(x) = 0 \), where \( f \) is a real function \( f : I \subseteq \mathbb{R} \rightarrow \mathbb{R} \), defined in an open interval \( I \).

The well-known numerical method for the calculation of \( \xi \) is the classical Newton’s method as given by

\[
\begin{align*}
x_{n+1} &= x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 0, 1, \ldots,
\end{align*}
\]

where \( x_0 \) is an initial approximation that is sufficiently close to \( \xi \). The convergence order of the classical Newton's method is quadratic for simple roots and linear for multiple roots.

Recently, a number of authors including [1-6] have derived new variants of Newton’s methods that offer higher order convergence. These methods are frequently composed of more than two formulas and derived in different ways.

In [6] some fifth order modifications of Newton’s method which is extending a general form of third order method are considered. In a similar way, some sixth-order class of modified Ostrowski’s methods [7] that improves the order of convergence of Ostrowski’s method are presented as follows

\[
\begin{align*}
y_n &= x_n - \frac{f(x_n)}{f'(x_n)},
y_n &= y_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 0, 1, \ldots,
\end{align*}
\]

This is a three-step method. The first and the second equations of equation (3) compose a third-order method developed by the authors in [9].
It is worth noting that equations (2) and (3) use the technique that consists in applying a new function on the existing iterative schemes. Motivated by the activities in this direction, in this paper, our special attention is paid to the development of a general class of higher-order combined iterative methods. The important feature of our methods is only to add the evaluation of the function at another point. However, their convergence order can be improved and increased above the original level.

This paper is organized as follows. In section 2, the new methods are formulated, and the local convergence theorem is established. Some concrete iterative methods are discussed in section 3. In section 4, the new methods are verified through a number of numerical examples, comparisons of results are also reported to show the effectiveness of the present approach. Finally, the paper ends with conclusions in section 5.

2. The Methods and Analysis of Convergence

Let \( f(x; f(x), f'(x), f'(y)) \) be a function from \( R \rightarrow R \) with the information \( f, f' \) at \( x \) and \( y \), this means that the functions \( f(x), f'(x) \) and \( f'(y) \) at each iteration step are required to evaluate in the computation of \( \phi \). Now we consider the modification of Newton’s method as given below

\[
\begin{align*}
\phi(x_n; f(x_n), f'(x_n), f'(y_n)) = 0, \\
x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)},
\end{align*}
\]

(4)

where \( z_n = \phi(x; f(x), f'(x), f'(y)) \) represents any iterative method whose order of convergence is at least \( m \).

In recent years, a type of multipoint iterative methods have been proposed [9-11]. These methods can be viewed as obtained by approximating \( f'(x_n) \) with the expressions \( a_1(x_n), \ldots, a_k(x_n) \), where the function \( f'(z_n) \) is defined as

\[
f'(z_n) = \phi(x_n; a_1(x_n), \ldots, a_k(x_n)).
\]

(5)

Analogously, in order to derive the new methods, we consider the expression

\[
f'(z_n) = Af'(x_n) + Bf'(y_n) + Cf(z_n) + Df'(\frac{x_n + y_n}{2}),
\]

(6)

for application of the method of undetermined coefficients.

Expand the terms \( f'(z_n), f'(y_n), f'(\frac{x_n + y_n}{2}) \) and \( f(z_n) \) about the point \( x_n \) up to the third derivatives and collect terms. Upon comparing the coefficients of the derivatives of \( f \) at \( x_n \), we get

\[
A + B + \alpha C + D = 1,
\]

(7)

\[
C = 0,
\]

(8)

\[
B\beta + \frac{Ca^2}{2} + \frac{1}{2} D\beta = \alpha,
\]

(9)

\[
\frac{B\beta^2}{2} + \frac{Ca^3}{6} + \alpha \beta \beta^2 = \alpha^2,
\]

(10)

\[
where \( \alpha = z_n - x_n \) and \( \beta = y_n - x_n \).

Solving equations (7)-(10), we have

\[
A = \frac{\beta^2 - 3\alpha\beta + 2\alpha^2}{\beta^2},
\]

(11)

\[
B = \frac{\alpha\beta + 2\alpha^2}{\beta^2},
\]

(12)

\[
C = 0,
\]

(13)

\[
D = \frac{4(\alpha\beta - \alpha^2)}{\beta^2}.
\]

(14)

Substituting equations (11)-(14) into equation (6), we obtain

\[
f'(z_n) = \frac{\beta^2 - 3\alpha\beta + 2\alpha^2}{\beta^2} f'(x_n) \]

\[
- \frac{\alpha\beta - 2\alpha^2}{\beta^2} f'(y_n) + \frac{4(\alpha\beta - \alpha^2)}{\beta^2} f'(\frac{x_n + y_n}{2}).
\]

(15)

By using the arithmetic mean of \( f'(x_n) \) and \( f'(y_n) \) instead of the midpoint value \( f'(\frac{x_n + y_n}{2}) \) in equation (15), we have

\[
f'(z_n) = \frac{\beta^2 - \alpha\beta}{\beta^2} f'(x_n) + \frac{\alpha\beta}{\beta^2} f'(y_n).
\]

(16)

Substituting equation (16) into the second step of equation (4), we propose a higher-order family of combined iterative methods in the following form:

\[
\begin{align*}
\phi(x_n; f(x), f'(x), f'(y)) = 0, \\
x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.
\end{align*}
\]

(17)

For this family of methods, the following result can be established.

Theorem 2.1. Let us suppose that \( f(x) \) is a sufficiently differentiable function and \( f(x) \) has a simple zero \( \xi \). If the initial guess \( x_0 \) is close enough to \( \xi \) and iteration
function $z_n = \phi(x_n; f(x_n), f'(x_n), f'(y_n))$ satisfies condition

$$z_n - \xi = A n^m + O\left(n^{m+1}\right). \quad (18)$$

If $m < 3$, then the sequence $\{x_k\}$ generated by equation (17) is of order at least $2m$. If $m \geq 3$, then the sequence $\{x_k\}$ generated by equation (17) is of order at least $m + 3$.

Proof. Let $e_n = x_n - \xi$ and $d_n = z_n - \xi$. Using the Taylor expansion and taking into account $f'(\xi) = 0$, we arrive at

$$f(x_n) = f'(\xi)\left[e_n + c_2e_n^2 + c_3e_n^3 + c_4e_n^4 + c_5e_n^5 + c_6e_n^6 + c_7e_n^7 + O\left(e_n^8\right)\right]. \quad (19)$$

$$f'(x_n) = f'(\xi)\left[1 + 2c_2e_n + 3c_3e_n^2 + 4c_4e_n^3 + 5c_5e_n^4 + 6c_6e_n^5 + 7c_7e_n^6 + O\left(e_n^7\right)\right]. \quad (20)$$

where $c_k = \frac{f^{(k)}(\alpha)}{k!f'(\alpha)}$.

By simple calculations, we have

$$y_n = x_n - \frac{f(x_n)}{f'(x_n)} = \xi + c_2e_n^2 + 2\left(c_3 - c_2^2\right)e_n^3 + \left(4c_4^2 - 7c_3c_4 + 3c_4\right)e_n^4 + \left(-8c_4^2 - 20c_5c_4 - 6c_2^2 + 10c_2c_4 - 4c_5\right)e_n^5 + \left(16c_5^2 - 52c_3c_5 + 28c_4c_5 - 13c_2c_5 - 33c_3c_6\right)e_n^6 + \left(-32c_5^2 + 128c_4c_6 - 72c_2c_6\right)e_n^7 + O\left(e_n^8\right), \quad (21)$$

$$\beta = y_n - x_n = -e_n + c_2e_n^2 + 2\left(c_3 - c_2^2\right)e_n^3 + 4\left(4c_4^2 - 7c_3c_4 + 3c_4\right)e_n^4 + \left(-8c_4^2 - 20c_5c_4 - 6c_2^2 + 10c_2c_4 - 4c_5\right)e_n^5 + \left(16c_5^2 - 52c_3c_5 + 28c_4c_5 - 13c_2c_5 + 33c_3c_6\right)e_n^6 + \left(-32c_5^2 + 128c_4c_6 - 72c_2c_6\right)e_n^7 + O\left(e_n^8\right). \quad (25)$$

Hence, we obtain

$$\beta^2 = e_n^2 - 2c_2e_n^3 + \left(5c_2^2 - 4c_3\right)e_n^4 + 6\left(3c_2c_3 - 2c_2^3 - 4c_4\right)e_n^5 + \left(28c_4^2 - 62c_3c_4 + 26c_2c_4 + 16c_5 - 8c_5\right)e_n^6 + \left(-64c_5^2 + 188c_3c_5 - 88c_4c_5 + 34c_2c_5\right)e_n^7 + \left(-106c_2c_5^2 - 10c_6 + 46c_3c_6\right)e_n^8 + O\left(e_n^9\right), \quad (26)$$

$$\alpha\beta = e_n\left[1 + c_2d_n\right]e_n - \left(c_2 - 2\left(c_3 - c_2^2\right)d_n\right)e_n^2 - \left(2\left(c_3 - c_2^2\right) - \left(4c_4^2 - 7c_3c_4 + 3c_4\right)d_n\right)e_n^3 - \left(4c_4^2 - 7c_3c_4 + 3c_4\right)e_n^4 - \left(-8c_4^2 - 20c_5c_4 + 6c_2^2 + 10c_2c_4 - 4c_5\right)d_n^2e_n^4 + \left(8c_4^2 - 20c_5c_4 + 6c_2^2 + 10c_2c_4 - 4c_5\right)e_n^5 + \left(16c_5^2 - 52c_3c_5 + 28c_4c_5 - 13c_2c_5 + 33c_3c_6\right)e_n^6 + \left(-32c_5^2 + 128c_4c_6 - 72c_2c_6\right)e_n^7 + O\left(e_n^8\right), \quad (27)$$

We then have
\[ \beta^2 - \alpha \beta + c_n d_n - c_2 d_n e_n^2 - (c_2 + 2(c_3 - c_2^2) d_n) e_n^3 + \left((3c_2^2 - 2c_3) - (4c_3^2 - 7c_2 c_3 + 3c_4) d_n\right) e_n^4 + \left((11c_2 c_3 - 8c_3 - 3c_4) + (8c_2^4 - 4c_5 - 20c_2^2 c_3 + 6c_3^2 + 10c_2 c_4) d_n\right) e_n^5 + (20c_4^2 - 4c_5 - 42c_3^2 c_5 + 16c_4 c_5 + 10c_3^2) e_n^6 - (48c_5^2 - 136c_2^3 c_5 + 60c_3^2 c_5 - 21c_2 c_5 + 73c_2 c_5) + 5c_6 - 29c_5 c_4) e_n^7 + (11c_2^2 - 6c_2 c_4 + 4c_2^2 - 4c_3^2 + 5c_6 + 28c_3^2 c_5 + 16c_3 c_5 - 8c_2 c_5) - 20c_2 c_5^2 - 12c_2 c_4) e_n^8 + (16c_6^2 - 68c_2^2 c_5 + 40c_2 c_4 + 73c_2^2 c_3^2 - 21c_2 c_3^2 - 6c_4 - 12c_3^2 - 58c_2 c_3 c_4 + 10c_2 c_6 + 9c_4^2 + 16c_3 c_5) e_n^9 + (30)\]

Using equations (29) and (30), we have

\[ \begin{align*}
\beta^2 & - \alpha \beta + c_n d_n + (1 + 2c_2 d_n) e_n^2 \\
& - (c_2 - 2(c_3 + 2c_2^2) d_n) e_n^3 \\
& + \left((4c_2^2 - 2c_3) + (10c_3 - 11c_2 c_3 + 3c_4) d_n\right) e_n^4 \\
& + \left((11c_2 c_3 - 10c_3 - 3c_4) - (12c_2^2 c_3 - 6c_2^2 - 10c_2 c_4 + 4c_4) d_n\right) e_n^5 \\
& + (24c_4^2 - 39c_2^2 c_3 + 6c_3^2 + 16c_2 c_4 - 4c_3^2 + 5c_6 - 17c_3 c_4) e_n^6 \\
& - (56c_5^2 - 121c_2 c_3 + 60c_2 c_4 - 21c_2 c_5 + 41c_2 c_3 c_5 + 5c_6 - 17c_3 c_4) e_n^7 \\
& + (128c_6^2 - 358c_2^2 c_3 + 196c_2 c_4 + 212c_2 c_3^2 - 78c_2 c_3 - 6c_2 - 132c_2 c_3 c_4 + 26c_2 c_5) e_n^8 + (12c_4^2 - 18c_3^2 + 22c_2 c_3) e_n^9 + O\left(e_n^{10}\right).
\end{align*}\]

Thus, from equations (20), (22), (27) and (28), we attain

\[ \begin{align*}
\beta^2 - \alpha \beta + c_n d_n + (1 + 2c_2 d_n) e_n^2 \\
& - (c_2 - 2(c_3 + 2c_2^2) d_n) e_n^3 \\
& + \left((4c_2^2 - 2c_3) + (10c_3 - 11c_2 c_3 + 3c_4) d_n\right) e_n^4 + \left((11c_2 c_3 - 10c_3 - 3c_4) - (12c_2^2 c_3 - 6c_2^2 - 10c_2 c_4 + 4c_4) d_n\right) e_n^5 \\
& + (24c_4^2 - 39c_2^2 c_3 + 6c_3^2 + 16c_2 c_4 - 4c_3^2 + 5c_6 - 17c_3 c_4) e_n^6 - (56c_5^2 - 121c_2 c_3 + 60c_2 c_4 - 21c_2 c_5 + 41c_2 c_3 c_5 + 5c_6 - 17c_3 c_4) e_n^7 + (128c_6^2 - 358c_2^2 c_3 + 196c_2 c_4 + 212c_2 c_3^2 - 78c_2 c_3 - 6c_2 - 132c_2 c_3 c_4 + 26c_2 c_5) e_n^8 + (12c_4^2 - 18c_3^2 + 22c_2 c_3) e_n^9 + O\left(e_n^{10}\right) + \beta^2 f\left(x_n\right) + \alpha \beta f\left(y_n\right)
\end{align*}\]

In the same way, we attain

\[ \begin{align*}
\beta^2 f\left(z_n\right) & = f\left(\xi\right) e_n^2 \left[1 - 2c_2 e_n + \left(5c_2^2 - 4c_3\right) e_n^2 + 6\left(3c_2 c_3 - 2c_2 - c_4\right) e_n^3 + \left(28c_2^2 - 62c_2 c_3 + 26c_2 c_4 + 16c_2^2 - 8c_3\right) e_n^4 + \left(188c_3^2 c_3 - 64c_3^2 - 88c_4 c_3 + 34c_2 c_5 - 106c_2 c_3^2 - 10c_6 + 46c_3 c_4\right) e_n^5 + c_2 d_n - 2c_2 e_n d_n + \left(5c_3 - 4c_2 c_3\right) e_n^2 d_n + \left(6c_3 c_2 - 2c_4 - c_2 c_4\right) e_n^3 d_n + \left(144c_5 - 528c_2 c_3 + 264c_2 c_4 + 47c_2^2 c_3^2 - 114c_2 c_5 - 300c_2 c_3 c_4 + 42c_2 c_6 + 33c_2^4 - 12c_7 - 60c_3^3 + 60c_3 c_5\right) e_n^4 + O\left(e_n^{10}\right)\right].
\end{align*}\]

Dividing equation (32) by equation (31), we get

\[ \begin{align*}
\beta^2 f\left(x_n\right) + \alpha \beta f\left(y_n\right) & = d_n \left[1 + 3c_2 c_3 e_n^3 - c_2 d_n + O\left(e_n^{10}\right)\right].
\end{align*}\]

Since \(z_n\) is of order at least \(m\), there exists a constant \(A\) such that

\[ \begin{align*}
d_n = z_n - \xi = A e_n^m + O\left(e_n^{m+1}\right).
\end{align*}\]

Thus,
3. The Concrete Iterative Methods

This section describes some interesting studies based on different forms of \( f(x, f(x), f'(x), f''(y)) = x - \frac{f(x)}{f'(x)}. \)

Throughout the rest of this article, \( y_n \) is defined by equation (21).

3.1. Some Fourth-Order Methods

Case 3.1. For the function \( \phi \) defined by

\[
\phi(x, f(x), f'(x), f''(y)) = x - \frac{f(x)}{f'(x)},
\]

then we obtain the two-step Newton’s method

\[
\begin{align*}
z_n &= x_n - \frac{f(x_n)}{f'(x_n)}, \\
x_{n+1} &= z_n - \frac{f(z_n)}{f'(z_n)}.
\end{align*}
\]

(38)

(39)

It is easy to see that equation (39) is the well-known two-point fourth-order double-Newton method [12, 13].

Case 3.2. If we take a second order method from [14]

\[
z_n = x_n - \frac{f(x_n)}{f'(x_n) + \alpha' f(x_n)},
\]

(40)

where \( \alpha' \) is a parameter, then we obtain the new four-order method

\[
\begin{align*}
z_n &= x_n - \frac{f(x_n)}{f'(x_n) + \alpha' f(x_n)}, \\
x_{n+1} &= z_n - \frac{\beta^2 f(z_n)}{(\beta^2 - \alpha' \beta) f'(x_n) + \alpha' f'(y_n)}.
\end{align*}
\]

(41)

These two methods have order four and use four functional evaluations per step, and their efficiency index [15] is the same with the known methods of order two. However, numerical examples show that these modified methods may be efficient enough and have better performance as compared to the known methods of order two.

### 3.2. Some Sixth-Order Methods

Case 3.3. If we take a third-order variant of Newton’s method appeared in [9]

\[
z_n = x_n - \frac{2 f(x_n)}{f'(y_n) + f'(x_n)}
\]

(42)

Then, we get the new sixth-order method

\[
\begin{align*}
x_n &= x_n - \frac{2 f(x_n)}{f'(y_n) + f'(x_n)}, \\
x_{n+1} &= z_n - \frac{\beta^2 f(z_n)}{(\beta^2 - \alpha' \beta) f'(x_n) + \alpha f'(y_n)}.
\end{align*}
\]

(43)

Case 3.4. If we use the cubically convergent iterative scheme in [16]

\[
x_n = x_n - \frac{f(x_n)\left( f'(x_n) + f'(y_n) \right)}{2 f'(x_n) f'(y_n)},
\]

(44)

Then, the following expressions can be resulted

\[
\begin{align*}
x_n &= x_n - \frac{f(x_n)\left( f'(x_n) + f'(y_n) \right)}{2 f'(x_n) f'(y_n)}, \\
x_{n+1} &= z_n - \frac{\beta^2 f(z_n)}{(\beta^2 - \alpha' \beta) f'(x_n) + \alpha f'(y_n)}.
\end{align*}
\]

(45)

The new methods (43) and (45) have the efficiency index equal to \( \frac{1}{6} \approx 1.5651 \), which is better than \( \sqrt{2} \approx 1.4142 \) of Newton’s method and \( \frac{1}{3} \approx 1.4422 \) of the methods (42) and (44).

### 3.3. The Seventh-Order Method

Case 3.5. If we take the fourth-order Jarratt method [17] defined by

\[
y_n = x_n - \frac{2 f(x_n)}{3 f'(y_n)},
\]

(46)

\[
z_n = x_n - \frac{3 f'(y_n) + f'(x_n)}{6 f'(y_n) - 2 f'(x_n)} f'(y_n).
\]

Then, we obtain the new seventh-order method

\[
\begin{align*}
y_n &= x_n - \frac{2 f(x_n)}{3 f'(x_n)}, \\
z_n &= x_n - \frac{3 f'(y_n) + f'(x_n)}{6 f'(y_n) - 2 f'(x_n)} f'(x_n), \\
x_{n+1} &= z_n - \frac{\beta^2 f(z_n)}{(\beta^2 - \alpha \beta) f'(x_n) + \alpha f'(y_n)}.
\end{align*}
\]

(47)
Case 3.6. If we take the fourth-order method presented by Khattri and Abbasbandy in [18]

\[
y_n = x_n - \frac{2}{3} f(x_n) f'(x_n), \\
z_n = y_n - \left[ 1 + \frac{21}{8} \right] f'(y_n) f'(x_n) + \left( -\frac{9}{2} \right) \frac{f''(y_n)}{f'(x_n)} + \frac{15}{8} \left[ f'(y_n) \right]^3 f(x_n) \frac{1}{f'(x_n)}.
\]

Then, we obtain the new seventh-order method

\[
y_n = x_n - \frac{2}{3} f(x_n) f'(x_n), \\
z_n = y_n - \left[ 1 + \frac{21}{8} \right] f'(y_n) f'(x_n) + \left( -\frac{9}{2} \right) \frac{f''(y_n)}{f'(x_n)} + \frac{15}{8} \left[ f'(y_n) \right]^3 f(x_n) \frac{1}{f'(x_n)}, \\
x_{n+1} = z_n - \frac{\beta^2 f(z_n)}{\beta^2 - \alpha \beta} f'(x_n) + \alpha \beta f''(y_n).
\]

The methods defined by equations (47) and (49) require two function and two first derivative evaluations per iteration. Each method improves the order of convergence of the fourth-order method from four to seven with additional function evaluation at the point iterated by the fourth-order method. We have that the methods obtained by the formulae (47) and (49) have the efficiency index equal to \(7^{\frac{1}{2}} = 1.6266\), which is better than the fourth-order methods \(4^3 = 1.5874\). It should be pointed out that many new higher-order methods can be obtained by considering the different choices of \(z_n\) in (17).

4. Numerical Examples

In this section, the results of some numerical tests are given to demonstrate the convergence efficiencies of various iterative schemes. We employ the present methods (39), (41) (\(\alpha = 1\)), (43), (45), (47) and (49) denoted by INW, IKT, IWM, IOM, IJA and IKA, respectively to solve some nonlinear equations and compare with Newton’s method (NW), the method (KT) developed by Kanwar et al. [14], the method (WM) developed by Weerakoon et al. [9], the method (OM) by Özban [16], the Jarratt method (JA) and the method (KA) developed by Khattri et al. [18]. The equations \(f(x) = 0\) was solved using the following test functions with corresponding starting values \(x_0\):

\[
f_1(x) = \sin^2 x - x^2 + 1, \xi^*_1 = 1.4044916482153412260, \\
f_2(x) = x^2 - e^x - 3x + 2, \xi^*_2 = 0.25753028543986076046, \\
f_3(x) = \cos x - xe^x + x^2, \xi^*_3 = 0.63915409633200758106, \\
f_4(x) = \cos x - x, \xi^*_4 = 0.73908513321516064166.
\]

Numerical computations have been carried out using variable precision arithmetic, with 1000 digits, in Matlab 2013a. The stopping criterion is taken as \(|x_{n+1} - x_n| + |f(x_{n+1})| < 10^{-100}\). In cases where the exact solution was not available, we used the approximation \(\xi^*_n\), which was also calculated with 1000 digits. For simplicity, only 20 digits are displayed.

The computational order of convergence (COC) was given by (see [19])

\[
COC = \frac{\ln \left( \frac{|x_{k+1} - x_k|}{|x_k - x_{k-1}|} \right)}{\ln \left( \frac{|x_{k+1} - x_k|}{|x_{k-1} - x_{k-2}|} \right)}.
\]

Table 1 summarizes the results obtained by using the mentioned methods in order to estimate a root of nonlinear equations. For every function we specify the initial estimate \(x_0\), the number of iterations \(N\) required to meet the stopping criteria, the value of \(\left| f(x_{n+1}) \right|\) in the last iteration and the value of COC.

We also compare our methods with the fourth-order King’s method [20] (KM), the Ostrowski’s method [21] (OST), the sixth order methods given by Chun and Ham [7] (CM1 and CM2), Parhi and Gupta [8] (PM), and the eighth order method given by Grau-Sánchez [22] (MG) (see Table 2).

Table 1. Numerical comparisons of the existing methods and the present combined iterative methods

| Method | \(N\) | \(\left| f(x_{n+1}) \right|\) | CPU | COC |
|--------|------|-----------------|-----|-----|
| NW     | 9    | 3.4e-101        | 0.046 | 1.9998515889198838 |
| INW    | 5    | 3.4e-101        | 0.031 | 3.991493528005224 |
| KT     | 11   | 3.7e-168        | 0.047 | 2.000000000152457 |
| IKT    | 6    | 1.6e-234        | 0.203 | 3.941604179745691 |
| WM     | 7    | 7.5e-266        | 0.088 | 3.045328591919475 |
| IWM    | 5    | 1.2e-566        | 0.111 | 6.262699932136188 |
| OM     | 5    | 1.1e-186        | 0.531 | 3.010588446816082 |
| Method | N  | \( f(x_{n+1}) \) | CPU   | COC     |
|--------|----|-----------------|-------|---------|
| IOM    | 3  | 6.5e-123        | 0.063 | 6.570253392709460 |
| JA     | 6  | 1.4e-334        | 0.152 | 4.248708984793318  |
| IJA    | 4  | 6.0e-426        | 0.060 | 6.883153998959128  |
| KA     | 8  | 3.0e-244        | 0.203 | 3.944947185376285  |
| IKA    | 6  | 9.0e-728        | 0.125 | 6.896899302326081  |

\( f_1 \cdot x_0 = 2.3 \)

| Method | N  | \( f(x_{n+1}) \) | CPU   | COC     |
|--------|----|-----------------|-------|---------|
| NW     | 9  | 1.7e-104        | 0.031 | 1.999762551535668 |
| INW    | 5  | 1.7e-104        | 0.031 | 3.992705876948916 |
| KT     | 10 | 6.9e-110        | 0.031 | 2.00000006728449  |
| IKT    | 6  | 3.8e-296        | 0.078 | 3.973931308562368 |
| WM     | 6  | 5.8e-106        | 0.031 | 2.977548164561699 |
| IWM    | 5  | 1.0e-520        | 0.046 | 5.428706327611860 |
| OM     | 6  | 1.3e-213        | 0.031 | 3.004705987760414 |
| IOM    | 4  | 1.7e-129        | 0.054 | 5.756054092345167 |
| JA     | 6  | 1.4e-311        | 0.076 | 3.991853110427456 |
| IJA    | 5  | 6.0e-426        | 0.078 | 6.91059134100821  |
| KA     | 6  | 1.0e-234        | 0.094 | 3.929232615276695 |
| IKA    | 5  | 1.9e-1000       | 0.124 | 6.991599899128005 |

\( f_2 \cdot x_0 = 0 \)

| Method | N  | \( f(x_{n+1}) \) | CPU   | COC     |
|--------|----|-----------------|-------|---------|
| NW     | 8  | 8.9e-201        | 0.031 | 2.001221658759151 |
| INW    | 5  | 1.9e-402        | 0.031 | 4.036830839017272 |
| KT     | 8  | 4.9e-124        | 0.046 | 2.00000001565404  |
| IKT    | 5  | 1.4e-289        | 0.047 | 4.072048404246373 |
| WM     | 5  | 7.8e-106        | 0.033 | 2.989730807986798 |
| IWM    | 4  | 5.4e-271        | 0.046 | 5.998509512335663 |
| OM     | 5  | 4.3e-112        | 0.172 | 2.999984519710846 |
| IOM    | 4  | 3.5e-276        | 0.125 | 5.992519937559335 |
| JA     | 5  | 1.0e-286        | 0.057 | 3.987739193299696 |
| IJA    | 4  | 2.8e-827        | 0.061 | 6.997013442884977 |
| KA     | 5  | 1.6e-292        | 0.109 | 4.220027174017621 |
| IKA    | 4  | 5.3e-833        | 0.094 | 7.09530069185996  |

\( f_2 \cdot x_0 = 1 \)

| Method | N  | \( f(x_{n+1}) \) | CPU   | COC     |
|--------|----|-----------------|-------|---------|
| NW     | 8  | 1.7e-189        | 0.032 | 1.997942151132725 |
| INW    | 5  | 7.1e-380        | 0.032 | 3.725609846208910 |
| KT     | 9  | 4.1e-138        | 0.031 | 2.000000001854255 |
| IKT    | 5  | 2.7e-207        | 0.063 | 3.878358818775769 |
| WM     | 6  | 1.4e-201        | 0.062 | 3.038062042190221 |
| IWM    | 4  | 1.2e-201        | 0.031 | 5.825290885992024 |
| OM     | 6  | 3.4e-206        | 0.047 | 3.001050663973445 |
| IOM    | 4  | 3.1e-202        | 0.047 | 5.843786348256205 |
| JA     | 5  | 2.4e-258        | 0.062 | 3.722160566330791 |
| IJA    | 4  | 3.2e-633        | 0.062 | 6.833672690585321 |
| KA     | 5  | 5.6e-264        | 0.110 | 3.981598688393819 |
| IKA    | 4  | 5.5e-643        | 0.110 | 7.023459826814717 |

\( f_3 \cdot x_0 = 1 \)

| Method | N  | \( f(x_{n+1}) \) | CPU   | COC     |
|--------|----|-----------------|-------|---------|
| NW     | 9  | 1.3e-151        | 0.047 | 2.001106997169595 |
| INW    | 5  | 1.3e-151        | 0.031 | 3.999685578348938 |
| KT     | 9  | 1.3e-108        | 0.062 | 2.00000012840528  |
| Method | N  | $f(x_{n+1})$ | CPU  | COC     |
|--------|----|--------------|------|---------|
| IKT    | 5  | 7.2e-122     | 0.093| 3.998627364224366 |
| WM     | 6  | 2.9e-131     | 0.063| 3.000546669049605 |
| IWM    | 4  | 3.9e-133     | 0.062| 5.86966779516045  |
| OM     | 6  | 3.4e-186     | 0.234| 3.000513674703602 |
| IOM    | 4  | 2.8e-129     | 0.062| 5.983351099780604 |
| JA     | 6  | 3.4e-425     | 0.109| 3.990537088703305 |
| IJA    | 4  | 9.3e-444     | 0.106| 6.985005730502670 |
| KA     | 6  | 3.7e-302     | 0.172| 3.991952875605968 |
| IKA    | 5  | 0            | 0.125| 6.825746339976237 |

$f_3 \cdot x_0 = 0.5$

| Method | N  | $f(x_{n+1})$ | CPU  | COC     |
|--------|----|--------------|------|---------|
| NW     | 8  | 8.9e-122     | 0.046| 2.003980933435205 |
| INW    | 5  | 2.6e-243     | 0.047| 4.016633132132276 |
| KT     | 9  | 8e-184       | 0.047| 1.99999964933080 |
| IKT    | 5  | 7.0e-205     | 0.078| 4.031322751779069 |
| WM     | 6  | 8.7e-214     | 0.054| 3.044070100120532 |
| IWM    | 4  | 5.2e-203     | 0.078| 6.026855595538948 |
| OM     | 6  | 2.7e-292     | 0.062| 2.999983735018816 |
| IOM    | 4  | 1.6e-217     | 0.062| 5.996940163017289 |
| JA     | 5  | 5.6e-257     | 0.085| 3.995556784204996 |
| IJA    | 4  | 4.7e-723     | 0.101| 6.97121515532127  |
| KA     | 6  | 3.2e-400     | 0.141| 4.116625589705837 |
| IKA    | 4  | 4.8e-552     | 0.140| 6.989197183612720 |

$f_4 \cdot x_0 = 0$

| Method | N  | $f(x_{n+1})$ | CPU  | COC     |
|--------|----|--------------|------|---------|
| NW     | 9  | 1.2e-166     | 0.016| 1.998849110827749 |
| INW    | 5  | 1.2e-166     | 0.031| 3.999868264532630 |
| KT     | 11 | 3.0e-142     | 0.047| 2.00000000043006  |
| IKT    | 6  | 3.1e-220     | 0.046| 3.848261256965063 |
| WM     | 6  | 4.6e-189     | 0.031| 3.579101980170624 |
| IWM    | 4  | 4.5e-124     | 0.031| 6.409937874573192 |
| OM     | 6  | 3.1e-180     | 0.031| 2.998710147557952 |
| IOM    | 4  | 4.0e-183     | 0.047| 6.182149360844007 |
| JA     | 6  | 1.2e-388     | 0.047| 3.967349085468574 |
| IJA    | 4  | 3.7e-477     | 0.046| 6.909444215476527 |
| KA     | 7  | 4.9e-435     | 0.094| 3.60909292736953  |
| IKA    | 5  | 1.0e-851     | 0.109| 7.09217798320114  |

$f_4 \cdot x_0 = 1.7$

| Method | N  | $f(x_{n+1})$ | CPU  | COC     |
|--------|----|--------------|------|---------|
| NW     | 8  | 4.0e-130     | 0.031| 1.994301682099229 |
| INW    | 5  | 2.0e-260     | 0.047| 3.594716721885800 |
| KT     | 10 | 5.7e-177     | 0.031| 2.000000000480267 |
| IKT    | 5  | 6.9e-190     | 0.046| 3.999745340001081 |
| WM     | 6  | 1.2e-196     | 0.047| 3.150905120339132 |
| IWM    | 4  | 7.4e-148     | 0.032| 5.93006529346190  |
| OM     | 6  | 1.5e-177     | 0.031| 3.001592484975891 |
| IOM    | 4  | 6.2e-143     | 0.031| 5.886505869490367 |
| JA     | 6  | 1.4e-443     | 0.062| 3.665874280525460 |
| IJA    | 4  | 1.7e-484     | 0.054| 6.799062297633464 |
| KA     | 6  | 7.0e-428     | 0.078| 3.868210819761995 |
| IKA    | 4  | 1.5e-466     | 0.093| 6.808033648694732 |
Table 2. Numerical comparisons of the methods of KM, OST, PM, CM1, CM2, MG and the present methods

| Method | $N$ | $f(x_{n+1})$ | CPU |
|--------|-----|---------------|-----|
| $f_1$, $x_0 = 1$ | | | |
| INW | 5 | 3.4e-101 | 0.031 |
| IKT | 6 | 1.6e-234 | 0.203 |
| KM | 10 | 6.3e-301 | 0.251 |
| OST | 5 | 1.0e-109 | 0.062 |
| IWM | 5 | 1.2e-566 | 0.111 |
| IOM | 3 | 6.5e-123 | 0.063 |
| CM1 | 5 | 1.4e-226 | 0.098 |
| CM2 | 6 | 3.6e-245 | 0.090 |
| PM | 5 | 1.2e-566 | 0.068 |
| IJA | 4 | 6.0e-426 | 0.060 |
| IKA | 6 | 9.0e-728 | 0.125 |
| MG | 5 | 7.6e-345 | 0.121 |
| $f_1$, $x_0 = 2.3$ | | | |
| INW | 5 | 1.7e-104 | 0.031 |
| IKT | 6 | 3.8e-296 | 0.078 |
| KM | 6 | 1.0e-259 | 0.078 |
| OST | 6 | 8.3e-389 | 0.063 |
| IWM | 5 | 1.0e-520 | 0.046 |
| IOM | 4 | 1.7e-129 | 0.054 |
| CM1 | 5 | 1.6e-391 | 0.115 |
| CM2 | 5 | 1.4e-372 | 0.085 |
| PM | 5 | 1.0e-520 | 0.075 |
| IJA | 5 | 6.0e-426 | 0.078 |
| IKA | 5 | 1.9e-1000 | 0.124 |
| MG | 4 | 1.6e-130 | 0.320 |
| $f_2$, $x_0 = 0$ | | | |
| INW | 5 | 1.9e-402 | 0.031 |
| IKT | 5 | 1.4e-289 | 0.047 |
| KM | 5 | 1.9e-376 | 0.059 |
| OST | 5 | 1.1e-352 | 0.063 |
| IWM | 4 | 5.4e-271 | 0.046 |
| IOM | 4 | 3.5e-276 | 0.125 |
| CM1 | 4 | 8.2e-313 | 0.081 |
| CM2 | 4 | 4.2e-291 | 0.067 |
| PM | 4 | 5.4e-271 | 0.052 |
| IJA | 4 | 2.8e-827 | 0.061 |
| IKA | 4 | 5.3e-833 | 0.094 |
| MG | 4 | 1.5e-387 | 0.133 |
| $f_2$, $x_0 = 1$ | | | |
| INW | 5 | 7.1e-380 | 0.032 |
| IKT | 5 | 2.7e-207 | 0.063 |
| KM | 5 | 3.8e-281 | 0.080 |
| OST | 5 | 6.6e-258 | 0.047 |
| IWM | 4 | 1.2e-201 | 0.031 |
| IOM | 4 | 3.1e-202 | 0.047 |
| CM1 | 4 | 3.4e-194 | 0.074 |
| CM2 | 4 | 1.4e-194 | 0.062 |
| PM | 4 | 1.2e-201 | 0.059 |
| IJA | 4 | 3.2e-633 | 0.062 |
| IKA | 4 | 5.5e-643 | 0.110 |
| MG | 4 | 3.5e-326 | 0.129 |
| Method | $N$ | $|f(x_{n+1})|$ | CPU |
|--------|-----|----------------|-----|
| $f_3 \cdot x_0 = 1$ |
| INW | 5 | 1.3e-151 | 0.031 |
| IKT | 5 | 7.2e-122 | 0.093 |
| KM | 5 | 1.5e-103 | 0.196 |
| OST | 5 | 9.5e-187 | 0.094 |
| IWM | 4 | 3.9e-133 | 0.062 |
| IOM | 4 | 2.8e-129 | 0.062 |
| CM1 | 4 | 9.0e-124 | 0.102 |
| CM2 | 4 | 4.3e-118 | 0.098 |
| PM | 4 | 4.0e-133 | 0.131 |
| IJA | 4 | 9.3e-444 | 0.106 |
| IKA | 5 | 0 | 0.125 |
| MG | 4 | 1.1e-205 | 0.190 |
| $f_3 \cdot x_0 = 0.5$ |
| INW | 5 | 2.6e-243 | 0.047 |
| IKT | 5 | 7.0e-205 | 0.078 |
| KM | 5 | 1.1e-163 | 0.124 |
| OST | 5 | 1.0e-292 | 0.062 |
| IWM | 4 | 5.2e-203 | 0.078 |
| IOM | 4 | 1.6e-217 | 0.062 |
| CM1 | 4 | 1.2e-200 | 0.120 |
| CM2 | 4 | 7.4e-188 | 0.096 |
| PM | 4 | 5.3e-203 | 0.093 |
| IJA | 4 | 4.7e-723 | 0.101 |
| IKA | 4 | 4.8e-552 | 0.140 |
| MG | 4 | 2.0e-316 | 0.203 |
| $f_4 \cdot x_0 = 0$ |
| INW | 5 | 1.2e-166 | 0.031 |
| IKT | 6 | 3.1e-220 | 0.046 |
| KM | 6 | 1.8e-197 | 0.057 |
| OST | 5 | 5.5e-141 | 0.031 |
| IWM | 4 | 4.5e-124 | 0.031 |
| IOM | 4 | 4.0e-183 | 0.047 |
| CM1 | 5 | 3.1e-489 | 0.073 |
| CM2 | 5 | 1.6e-373 | 0.055 |
| PM | 4 | 4.5e-124 | 0.038 |
| IJA | 4 | 3.7e-477 | 0.046 |
| IKA | 5 | 1.0e-851 | 0.109 |
| KLW | 4 | | |
| MG | 4 | 8.8e-127 | 0.086 |
| $f_4 \cdot x_0 = 1.7$ |
| INW | 5 | 2.0e-260 | 0.047 |
| IKT | 5 | 6.9e-190 | 0.046 |
| KM | 5 | 6.8e-178 | 0.045 |
| OST | 5 | 4.4e-192 | 0.031 |
| IWM | 4 | 7.4e-148 | 0.032 |
| IOM | 4 | 6.2e-143 | 0.031 |
| CM1 | 4 | 1.5e-141 | 0.067 |
| CM2 | 4 | 1.0e-139 | 0.051 |
| PM | 4 | 7.3e-148 | 0.044 |
| IJA | 4 | 1.7e-484 | 0.054 |
| IKA | 4 | 1.5e-466 | 0.093 |
| MG | 4 | 7.2e-244 | 0.089 |
In Table 1 it is seen that our combined iterative methods generally arrive at the iterated solution with less number of iterations than the corresponding second, cubic and the fourth methods, so that the proposed methods improve the computational efficiency of the existing iterative methods. Our examples show that the combined iterative methods sometimes require more CPU time per iteration, compared to the existing methods. Although our methods require more time per iteration, they yield better numerical results. The numerical results in Table 1 show that for almost all of the test functions, our methods are well in accordance with the theory developed in section 2.

The test results in Table 2 show that for most of the functions we tested, the methods introduced in the present presentation for numerical tests have equal or better performance compared to the other methods of the same order. In each of these 8 test cases, the INW method outperformed the KM method in every case and it outperformed the OST method in 5 out of the 8 cases. Our IKT method outperformed KM method in 6 out of the 8 cases. We also implemented the three 6th order schemes of [7,8] using these 8 cases, and found that the IWM and IOM methods outperformed the PM, CM1 and CM2 methods in 7 out of the 8 cases.

In [22] a eighth-order method, denoted with MG was considered. The IJA method outperformed the MG method in 7 out of the 8 cases, and the IKA method outperformed the MG method in 4 out of the 8 cases. Besides, we can see that the local convergence property of the new methods depending on the structure of the tested functions and the choice of initial approximations.

5. Conclusion

The new modified Newton’s methods presented in this paper offer an increase rate of convergence over the existing methods. Unlike other higher-order methods, the distinct feature of such methods is only to add the existing methods. Unlike other higher-order methods, so that the proposed methods improve the computational efficiency of the existing iterative methods. Our examples show that the combined iterative methods sometimes require more CPU time per iteration, compared to the existing methods. Although our methods require more time per iteration, they yield better numerical results. The numerical results in Table 1 show that for almost all of the test functions, our methods are well in accordance with the theory developed in section 2.

The test results in Table 2 show that for most of the functions we tested, the methods introduced in the present presentation for numerical tests have equal or better performance compared to the other methods of the same order. In each of these 8 test cases, the INW method outperformed the KM method in every case and it outperformed the OST method in 5 out of the 8 cases. Our IKT method outperformed KM method in 6 out of the 8 cases. We also implemented the three 6th order schemes of [7,8] using these 8 cases, and found that the IWM and IOM methods outperformed the PM, CM1 and CM2 methods in 7 out of the 8 cases.

In [22] a eighth-order method, denoted with MG was considered. The IJA method outperformed the MG method in 7 out of the 8 cases, and the IKA method outperformed the MG method in 4 out of the 8 cases. Besides, we can see that the local convergence property of the new methods depending on the structure of the tested functions and the choice of initial approximations.

5. Conclusion

The new modified Newton’s methods presented in this paper offer an increase rate of convergence over the existing methods. Unlike other higher-order methods, the distinct feature of such methods is only to add the evaluation of the function at another point, while their order of convergence can be improved effectively. Our new combined iterative methods are relatively simple and robust, more high-order convergence methods can be constructed by using the family of methods (17).

Computational results for test functions, presented in Table 1 and Table 2, show that our methods are efficient and show at least equal or better performance as compared with other higher order (4th order, 6th order and 8th order) schemes or Newton’s method itself. Our methods show similar good performance for other functions.

Acknowledgements

The authors gratefully acknowledge the support of the National Natural Science Foundations of China (NNSFC) through grants No.11302184, 11202189, the Scientific Research Fund of Zhejiang Provincial Education Department through grant No. Y201121157 and the Scientific Research Foundation of Xiamen University of Technology through grant No. YKJJ3007R.