Afr. Mat. (2018) 29:435–449
https://doi.org/10.1007/s13370-018-0552-x

Several two-point with memory iterative methods for solving nonlinear equations

Neha Choubey[1] · **Bhavna Panday**[2] · **J. P. Jaiswal**[3]

Received: 31 August 2015 / Accepted: 24 January 2018 / Published online: 7 February 2018

Abstract In this article, our main motivation is to present two-step with memory iterative methods for solving nonlinear equations. We attempted to convert the existing fourth-order without memory method into a with memory method. Further acceleration of convergence order is attained by means of different approximations of self-accelerating parameters. The parameters are calculated by Hermite interpolating polynomial and applied to accelerate the order of convergence of the without memory methods. In particular, the R-order of the proposed two-step with memory iterative method is increased without any additional calculations and it possesses high computational efficiency. At the end, the theoretical results are confirmed by considering different numerical examples. Numerical comparisons specify that the new family is efficient and give tough competition to some existing with memory iterative methods.

Keywords Iterative method · With memory scheme · Hermite interpolation polynomial · Computational efficiency · Numerical result

Mathematics Subject Classification 34G20 · 74G15

✉ J. P. Jaiswal
asstprofjpmanit@gmail.com

Neha Choubey
nehachby2@gmail.com

Bhavna Panday
bhavna.nic@gmail.com

[1] Department of Mathematics, Oriental Institute of Science and Technology, Bhopal, M.P. 462021, India

[2] Department of Mathematics, Demonstration Multipurpose School Regional Institute of Education, Bhopal, M.P. 462013, India

[3] Department of Mathematics, Maulana Azad National Institute of Technology, Bhopal, M.P. 462003, India

1 Introduction

Solving nonlinear equation $f(x) = 0$, is a very significant problem in the real world. To determine the solution of nonlinear equations, many iterative methods have been projected (see[1–5]); these iterative methods have a vital area of research in numerical analysis as they have applications in numerous branches of pure and applied sciences. Beyond them the most famous one-point iterative without memory method is the Newton–Raphson method, given by

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \tag{1}$$

where x_0 is the initial guess for the exact solution of $f(x) = 0$, and $n = 0, 1, 2, \ldots$ This converges quadratically. One drawback of this method is the condition $f'(x_n) \neq 0$, which confines its applications in practice. To resolve this difficulty, Kumar et al. [6] developed a new one-point iterative method, specified by

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n) - \lambda_1 f(x_n)}. \tag{2}$$

Setting $\lambda_1 = 0$ in (2), we obtain Newton's method. The error expression for the above method is:

$$e_{n+1} = (\lambda_1 - c_2)e_n^2 + O(e_n^3), \tag{3}$$

where $e_n = x_n - \alpha$, $c_k = \frac{1}{k!}\frac{f^{(k)}(\alpha)}{f'(\alpha)}$, $k = 2, 3, \ldots$ and α is a root of $f(x) = 0$.

Subsequently, we talk about the one-point iterative method with memory. In the mapping $x_{n+1} = \phi(x_n; x_{n-1}, \ldots, x_{n-k})$, the iteration function ϕ is called the one-point iteration function with memory, since x_n is novel information, while $x_{k-1}, \ldots, x_{n-k}$ are reused information. The best known model one-point with memory method is the secant method, given by

$$x_{n+1} = x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n), n = 1, 2, 3 \ldots \tag{4}$$

Now, the next iteration function ϕ, defined through the mapping $x_{n+1} = \phi(x_n, w_1(x_n), \ldots, w_n(x_n))$, is termed the multipoint iteration function without memory [7], where the terms $w_1(x_n), \ldots, w_n(x_n)$, have common argument x_n. Let us classify one more iteration function ϕ which has arguments z_j, where every argument characterizes $n + 1$ quantities $x_j, w_1(x_j), \ldots, w_n(x_j)(n \geq 1)$. The iteration mapping $x_{n+1} = \phi(z_n, z_{n-1}, \ldots, z_{n-k})$ is called a multipoint iteration function with memory, where in each iterative step we must preserve information of the last n approximations x_j and for each approximation we are required to calculate n expressions $w_1(x_j), \ldots, w_n(x_j)$. Now a days lots of researchers are paying their attention on developing with memory iterative methods using one or more self accelerating parameters. There are nice contributions are available devoted to derivative free with memory iterative methods such as [8–11]. But very few with memory with derivative iterative methods are available in literature for solving nonlinear equations. The main objective of this paper is to work on multipoint iteration function with memory, which can improve the order of convergence of the without memory with derivative methods, without using any additional calculations and it has very high computational efficiency. In this paper, we present a new multipoint iterative method with memory, to solve nonlinear equations. Their convergence analysis is also discussed. In Sect. 2, we discuss

two-point iterative method with memory. This method is acquired by employing a self-accelerating parameter. This parameter is designed by the Hermite interpolating polynomial, where the R-order convergence of two-point method is increased from 4 to 4.5616, 4.7913 and 5, respectively. Section 3, corroborates theoretical results by means of different numerical examples.

2 Development and convergence analysis of with memory method

In the ensuing section, we will improve the convergence rate of the method given by Hafiz and Bahgat [12]. First, we consider the optimal fourth-order without memory scheme presented in [12],

$$\begin{aligned} y_n &= x_n - \frac{f(x_n)}{f'(x_n)}, \\ x_{n+1} &= y_n - \frac{f(y_n)P_1(x_n, y_n)}{2P_1^2(x_n, y_n) - f(y_n)P_2(x_n, y_n)}, \end{aligned} \tag{5}$$

where $P_1(x_n, y_n) = 2\left(\frac{f(y_n)-f(x_n)}{y_n-x_n}\right) - f'(x_n)$ and $P_2(x_n, y_n) = \frac{2}{y_n-x_n}\left(\frac{f(y_n)-f(x_n)}{y_n-x_n} - f'(x_n)\right)$. The error expression for iterative scheme (5) is

$$e_{n+1} = -c_2c_3e_n^4 + O(e_n^5). \tag{6}$$

Now, we use the idea considered in Kumar et al. [6] to change the first step of the above method and consider

$$\begin{aligned} y_n &= x_n - \frac{f(x_n)}{f'(x_n) - Tf(x_n)}, \\ x_{n+1} &= y_n - \frac{f(y_n)P_1(x_n, y_n)}{2P_1^2(x_n, y_n) - f(y_n)P_2(x_n, y_n)}. \end{aligned} \tag{7}$$

If we set $T = 0$, then we obtain the first step of the method (5). The error expressions for each sub-step of scheme (7),

$$\begin{aligned} e_{n,y} = y_n - \alpha &= (c_2 - T)e_n^2 + (2(T - c_2)c_2 - T^2 - 2c_2T + 2c_3)e_n^3 \\ &+ (-T^3 - 5Tc_2^2 + 4c_2^3 + 4Tc_3 - c_2(7c_3 - 3T^2) + 3c_4)e_n^4 \\ &+ O(e_n^5) \end{aligned} \tag{8}$$

and

$$e_{n+1} = (T - c_2)c_3e_n^4 + O(e_n^5), \tag{9}$$

where $c_j = \frac{f^{(j)}(\alpha)}{j!f'(\alpha)}$, for $j = 2, 3, \ldots$ and $T, \gamma \in R$. Replacing T by T_n in the scheme (7), we obtain the following with memory method

$$\begin{aligned} y_n &= x_n - \frac{f(x_n)}{f'(x_n) - T_nf(x_n)}, \\ x_{n+1} &= y_n - \frac{f(y_n)P_1(x_n, y_n)}{2P_1^2(x_n, y_n) - f(y_n)P_2(x_n, y_n)}, \end{aligned} \tag{10}$$

which is denoted by OM4(2.6). It is easy to recognize from Eq. (6) that the order of convergence of scheme (5) is four and after changing the first step of method (5) to that of the

first step of the scheme (7), we see that the order of scheme (7) is also four when $T \neq c_2$. Now, by taking $T = c_2 = f''(\alpha)/(2f'(\alpha))$, it can be established that the order of the method (7) would be 5. For this type of acceleration of convergence the exact values of $f'(\alpha)$ and $f''(\alpha)$ are not obtainable. Other than this we could replace the parameter T by T_n. To locate the values of the parameter, we can utilize the information accessible from the current and previous iteration and it satisfies $\lim_{n\to\infty} T_n = c_2 = f''(\alpha)/(2f'(\alpha))$, such that the fourth-order asymptotic convergence constant to be zero in the error expression (9). We consider the following formula for T_n:

Method 1:

$$T_n = \frac{H_2''(x_n)}{2f'(x_n)}, \tag{11}$$

where $H_2(x) = f(x_n) + f[x_n, x_n](x - x_n) + f[x_n, x_n, y_{n-1}](x - x_n)^2$ and $H_2''(x) = 2f[x_n, x_n, y_{n-1}]$.

Method 2:

$$T_n = \frac{H_3''(x_n)}{2f'(x_n)}, \tag{12}$$

where $H_3(x) = H_2(x) + f[x_n, x_n, y_{n-1}, x_{n-1}](x - x_n)^2(x - y_{n-1})$ and $H_3''(x) = 2f[x_n, x_n, y_{n-1}] + 2f[x_n, x_n, y_{n-1}, x_{n-1}](x_n - y_{n-1})$.

Method 3:

$$T_n = \frac{H_4''(x_n)}{2f'(x_n)}, \tag{13}$$

where $H_4(x) = H_3(x) + f[x_n, x_n, y_{n-1}, x_{n-1}, x_{n-1}](x - x_n)^2(x - y_{n-1})(x - x_{n-1})$ and $H_4''(x) = 4f[x_n, x_n, y_{n-1}] + (2f[x_n, x_n, y_{n-1}, x_{n-1}] - 2f[x_n, y_{n-1}x_{n-1}, x_{n-1}])(x_n - y_{n-1})$.

Note: The Hermite interpolation polynomial $H_m(x)(m = 2, 3, 4)$ satisfied the condition $H_m'(x_n) = f'(x_n)$ $(m = 2, 3, 4)$. So, $T_n = \frac{H_m''(x_n)}{2f'(x_n)}$ can be expressed as $T_n = \frac{H_m''(x_n)}{2H'(x_n)}$ $(m = 2, 3, 4)$.

Theorem 1 *Let H_m be the Hermite interpolating polynomial of degree m that interpolates a function f at interpolation nodes $x_n, x_n, t_0 \ldots t_{m-2}$ contained in an interval I and the derivative $f^{(m+1)}$ is continuous in I and the Hermite interpolating polynomial $H_m(x_n) = f(x_n)$, $H_m'(x_n) = f'(x_n)$, $H_m(t_j) = f(t_j)$ $(j = 0, 1, \ldots, m-2)$. Define the errors $e_{t,j} = t_j - \alpha (j = 0, 1, \ldots, m-2)$ and assume that*

(1) *all nodes $x_n, t_0, \ldots, t_{m-2}$ are sufficiently close to the zero α,*

(2) *the condition $e_n = O(e_{t,0} \ldots e_{t,m-2})$ holds. Then*

$$H_m''(x_n) = 2f'(\alpha)\left(c_2 - (-1)^{m-1}c_{m+1}\prod_{j=0}^{m-2} e_{t,j} + 3c_3 e_n\right), \tag{14}$$

$$T_n = \frac{H''(x_n)}{2f'(x_n)} \sim \left(c_2 - (-1)^{m-1}c_{m+1}\prod_{j=0}^{m-2} e_{t,j} + (3c_3 - 2c_2^2)e_n\right) \tag{15}$$

and

$$T_n - c_2 \sim \left(- (-1)^{m-1}c_{m+1}\prod_{j=0}^{m-2} e_{t,j} + (3c_3 - 2c_2^2)e_n\right). \tag{16}$$

Proof The error expression of the Hermite interpolation can be expressed like this

$$f(x) - H_m(x_n) = \frac{f^{(m+1)}(\xi)}{(m+1)!}(x - x_n)^2 \prod_{j=0}^{m-2}(x_n - t_j), (\xi \in I). \tag{17}$$

After differentiating (17) twice at the point $x = x_n$, we obtain

$$H_m''(x_n) = f''(x) - 2\frac{f^{(m+1)}(\xi)}{(m+1)!} \prod_{j=0}^{m-2}(x_n - t_j), (\xi \in I). \tag{18}$$

Taylor's series expansion of derivative of f at the point $x_n \in I$ and $\xi \in I$ about the zero α of f give

$$f'(x_n) = f'(\alpha)\big(1 + 2c_2e_n + 3c_3e_n^2 + O(e_n^3)\big), \tag{19}$$

$$f''(x_n) = f'(\alpha)\big(2c_2 + 6c_3e_n + O(e_n^2)\big), \tag{20}$$

and

$$f^{(m+1)}(\xi) = f'(\alpha)\big((m+1)!c_{m+1} + (m+2)!c_{m+2}e_\xi + O(e_\xi^2)\big), \tag{21}$$

where $e_\xi = \xi - \alpha$. Placing (20), (21) in (18), we find

$$H_m''(x_n) = 2f'(\alpha)\left(c_2 - (-1)^{m-1}c_{m+1}\prod_{j=0}^{m-2} e_{t,j} + 3c_3e_n\right), \tag{22}$$

This implies

$$\frac{H_m''(x_n)}{2f'(x_n)} \sim \left(c_2 - (-1)^{m-1}c_{m+1}\prod_{j=0}^{m-2} e_{t,j} + (3c_3 - 2c_2^2)e_n\right). \tag{23}$$

Thus

$$T_n \sim \left(c_2 - (-1)^{m-1}c_{m+1}\prod_{j=0}^{m-2} e_{t,j} + (3c_3 - 2c_2^2)e_n\right), \tag{24}$$

or

$$T_n - c_2 \sim \left(-(-1)^{m-1}c_{m+1}\prod_{j=0}^{m-2} e_{t,j} + (3c_3 - 2c_2^2)e_n\right). \tag{25}$$

The concept of R-order of convergence [13] and the subsequent declaration (see [14], p.287) will be applied to approximate the convergence order of the iterative method (10).

Theorem 2 *If the errors of approximations $e_j = x_j - \alpha$ obtained in an iterative root finding method IM satisfy*

$$e_{k+1} \sim \prod_{j=0}^{m-2}(e_{k-i})^{m_i}, \quad k \geq k(\{e_k\}),$$

then the R-order of convergence of IM, denoted with $O_R(IM, \alpha)$, satisfies the inequality $O_R(IM, \alpha) \geq s^*$, where s^* is the unique positive solution of the equation $s_{n+1} - \sum_{i=0}^{n} m_i s^{n-i} = 0$.

We state the following convergence theorem designed for iterative method with memory (10).

Theorem 3 *Let the varying parameter T_n in the iterative method* (10) *be calculated by* (11)–(13). *If an initial approximation x_0 is sufficiently close to a simple root α of $f(x)$, then the R-order of convergence of iterative methods* (11)–(10), (12)–(10) *and* (13)–(10) *with memory is at least* $(5+\sqrt{17})/2 \approx 4.5616$, $(5+\sqrt{21})/2 \approx 4.7913$ *and 5, respectively.*

Proof Let the sequence $\{x_n\}$ be generated by an iterative method (IM) converges to the root α of $f(x)$, with R-order $O_R(IM,\alpha) \geq r$, we write down

$$e_{n+1} \sim D_{n,r} e_n^r, \quad e_n = x_n - \alpha. \tag{26}$$

If we take $n \to \infty$, then $D_{n,r}$ tends to the asymptotic error constant D_r of IM. Consequently

$$e_{n+1} \sim D_{n,r}(D_{n-1,r} e_{n-1}^r)^r = D_{n,r} D_{n-1,r} e_{n-1}^{r^2}. \tag{27}$$

The subsequent error expression of the with memory method (10), can be attained through (8), (9) and the varying parameter T_n

$$e_{n,y} = y_n - \alpha \sim (T_n - c_2) e_n^2, \tag{28}$$

and

$$e_{n+1} = x_{n+1} - \alpha \sim B_{n,4}(T_n - c_2) e_n^4, \tag{29}$$

where $B_{n,4}$ is a varying quantity because of the self accelerating parameter T_n and it comes from (9). Here, we excluded higher order terms in (28) and (29).
Method 1. T_n is calculated by (11): It is similar to the derivation of (26). We suppose that the iterative sequence $\{y_n\}$ has the R-order p,

$$e_{n,y} \sim D_{n,p} e_n^p \sim D_{n,p}(D_{n-1,r} e_{n-1}^r)^p = D_{n,p} D_{n-1,r}^p e_{n-1}^{rp}. \tag{30}$$

Using Theorem 1 for $m = 2$ and $t_0 = y_{n-1}$, we obtain

$$T_n - c_2 \sim c_3 e_{t,0} = c_3 e_{n-1,y}. \tag{31}$$

Now from (28), (29) and (31), we obtain

$$e_{n,y} \sim c_3 e_{n-1,y}(D_{n-1,r} e_{n-1}^r)^2 \sim c_3 D_{n-1,p} D_{n-1,r}^2 e_{n-1}^{2r+p}, \tag{32}$$

and

$$\begin{aligned} e_{n+1} &\sim B_{n,4} c_3 e_{n-1,y} e_n^4 \sim B_{n,4} c_3 D_{n-1,p} e_{n-1}^p (D_{n-1,r} e_{n-1}^r)^4, \\ &\sim B_{n,4} c_3 D_{n-1,p} D_{n-1,r}^4 e_{n-1}^{4r+p}. \end{aligned} \tag{33}$$

The following system of equations, are obtained by comparing the components of e_{n-1} featuring in two pairs of relation (30)–(32) and (27)–(33),

$$\begin{aligned} 2r + p &= rp, \\ 4r + p &= r^2. \end{aligned} \tag{34}$$

Positive solution of system (34) is specified through $r = (5+\sqrt{17})/2$ and $p = (1+\sqrt{17})/2$. As a result, we obtain at least $(5+\sqrt{17})/2 \approx 4.5616$ is the R-order of with memory method (10)–(11).

Method 2. T_n is calculated by (12): Using Theorem 1 for $m = 3$, $t_0 = y_{n-1}$ and $t_1 = x_{n-1}$, we obtain

$$T_n - c_2 \sim -c_4 e_{t,0} e_{t,1} = -c_4 e_{n-1,y} e_{n-1}. \tag{35}$$

Now using (28), (29) and (34), we obtain

$$\begin{aligned} e_{n,y} &\sim (T_n - c_2) e_n^2 \sim -c_4 e_{n-1} e_{n-1,y} (D_{n-1,r} e_{n-1}^r)^2 \\ &\sim -c_4 D_{n-1,p} D_{n-1,r}^2 e_{n-1}^{2r+p+1}, \end{aligned} \tag{36}$$

and

$$\begin{aligned} e_{n+1} &\sim -B_{n,4} c_4 e_{n-1} e_{n-1,y} e_n^4 \sim -B_{n,4} c_4 e_{n-1} D_{n-1,p} e_{n-1}^p (D_{n-1,r} e_{n-1}^r)^4 \\ &\sim -B_{n,4} c_4 D_{n-1,p} D_{n-1,r}^4 e_{n-1}^{4r+p+1}. \end{aligned} \tag{37}$$

We obtain the following system of equations, by comparing the components of e_{n-1} featuring in two pairs of relation (30)–(36) and (27)–(37)

$$\begin{aligned} 2r + p + 1 &= rp, \\ 4r + p + 1 &= r^2. \end{aligned} \tag{38}$$

Positive solution of system (38) is specified through $r = (5+\sqrt{21})/2$ and $p = (1+\sqrt{21})/2$. As a result, we obtain at least $(5 + \sqrt{21})/2 \approx 4.7913$ is the R-order of the methods with memory (10)–(12).

Method 3. T_n is calculated by (13): Hermite interpolating polynomial $H_4(x)$ satisfies the condition $H_4(x_n) = f(x_n)$, $H_4'(x_n) = f'(x_n)$, $H_4(y_{n-1}) = f(y_{n-1})$, $H_4(x_{n-1}) = f(x_{n-1})$ and $H_4'(x_{n-1}) = f'(x_{n-1})$. The error expression of the Hermite interpolation can be expressed as follows:

$$f(x) - H_4(x) = \frac{f^{(5)}(\xi)}{5!}(x - x_n)^2 (x - x_{n-1})^2 (x - y_{n-1}), (\xi \in I). \tag{39}$$

After differentiating (39) twice at the point $x = x_n$, we obtain

$$H_4''(x_n) = f''(x_n) - 2\frac{f^{(5)}(\xi)}{5!}(x_n - x_{n-1})^2 (x_n - y_{n-1}), (\xi \in I). \tag{40}$$

Taylor's series expansion of derivatives of f at the point $x_n \in I$ and $\xi \in I$ about the zero α of f give

$$f'(x_n) = f'(\alpha)(1 + 2c_2 e_n + 3c_3 e_n^2 + O(e_n^3)), \tag{41}$$

$$f''(x_n) = f'(\alpha)(2c_2 + 6c_3 e_n + O(e_n^2)), \tag{42}$$

and

$$f^{(m+1)}(\xi) = f'(\alpha)\big((m+1)! c_{m+1} + (m+2)! c_{m+2} e_\xi + O(e_\xi^2)\big), \tag{43}$$

where $e_\xi = \xi - \alpha$.

Substituting (43) and (42) into (40), we obtain

$$H_4''(x_n) = 2f'(\alpha)(c_2 - c_5 e_{n-1,y} e_{n-1}^2 + 3c_3 e_n). \tag{44}$$

By means of (28) and (29), we have

$$e_{n-1,y} = y_{n-1} - \alpha \sim (T_{n-1} - c_2) e_{n-1}^2 \tag{45}$$

and

$$e_n = x_n - \alpha \sim B_{n-1,4}(T_{n-1} - c_2)e_{n-1}^4. \tag{46}$$

Then

$$e_{n-1,y}e_{n-1}^2 \sim (T_{n-1} - c_2)e_{n-1}^4 \sim \frac{1}{B_{n-1,4}}e_n, \tag{47}$$

after substituting (47) into (44), we obtain

$$H_4''(x_n) = 2f'(a)\left(c_2 + \left(3c_3 - \frac{c_5}{B_{n-1,4}}\right)e_n\right), \tag{48}$$

which implies

$$\frac{H_4''(x_n)}{2f'(x_n)} \sim c_2 + \left(3c_3 - 2c_2^2 - \frac{c_5}{B_{n-1,4}}\right)e_n. \tag{49}$$

And hence

$$T_n = \frac{H_4''(x_n)}{2f'(x_n)} \sim c_2 + \left(3c_3 - 2c_2^2 - \frac{c_5}{B_{n-1,4}}\right)e_n, \tag{50}$$

or

$$T_n - c_2 \sim \left(3c_3 - 2c_2^2 - \frac{c_5}{B_{n-1,4}}\right)e_n. \tag{51}$$

Substituting (51) into (29), we obtain

$$e_{n+1} \sim B_{n,4}\left(3c_3 - 2c_2^2 - \frac{c_5}{B_{n-1,4}}\right)e_n^5, \tag{52}$$

As a result, the R-order of the methods with memory (10)–(13) is at least 5. Thus the proof is over.

3 Results and discussion

Now we attempt to compare the two-step with memory method with our two-step method OM4(2.6). Wang and Zang [15] proposed two with memory methods. Both are two-step methods in the subsequent form:

$$\begin{aligned} y_n &= x_n - \frac{f(x_n)}{T_n f(x_n) + f'(x_n)}, \\ x_{n+1} &= y_n - \left(1 - \frac{f(y_n)}{2T_n f(x_n) + f'(x_n)}\right) \\ &\quad \times \left(1 + \frac{2f(y_n)}{f(x_n)} + a\left(\frac{2f(y_n)}{f(x_n)}\right)^2\right), \end{aligned} \tag{53}$$

where $a \in R$, which is denoted by XW41(16), and

$$\begin{aligned} y_n &= x_n - \frac{f(x_n)}{T_n f(x_n) + f'(x_n)}, \\ x_{n+1} &= y_n - \left(1 - \frac{f(y_n)}{2T_n f(x_n) + f'(x_n)}\right)\left(\frac{f(x_n) + (2+b)f(y_n)}{f(x_n) + bf(y_n)}\right), \end{aligned} \tag{54}$$

Table 1 Test functions and their roots

Non-linear function	Root
$f_1 = x^5 + x^4 + 4x^2 - 15$	1.3474...
$f_2 = x^3 - x^2 - 1$	1.4655...
$f_3 = x^2 - e^x - 3x + 2$	0.2575...
$f_4 = sin^2(x) - x^2 + 1$	−1.4044...

where $b \in R$, which is denoted by XW42(17). We are captivating the values of the self-accelerating parameter T_n for both methods in the following form
Method 4:

$$T_n = -\frac{H_2''(x_n)}{2f'(x_n)}, \tag{55}$$

where $H_2(x) = f(x_n) + f[x_n, x_n](x - x_n) + f[x_n, x_n, y_{n-1}](x - x_n)^2$ and $H_2''(x) = 2f[x_n, x_n, y_{n-1}]$.
Method 5:

$$T_n = -\frac{H_3''(x_n)}{2f'(x_n)}, \tag{56}$$

where $H_3(x) = H_2(x) + f[x_n, x_n, y_{n-1}, x_{n-1}](x - x_n)^2(x - y_{n-1})$ and $H_3''(x) = 2f[x_n, x_n, y_{n-1}] + 2f[x_n, x_n, y_{n-1}, x_{n-1}](x_n - y_{n-1})$.
Method 6:

$$T_n = -\frac{H_4''(x_n)}{2f'(x_n)}, \tag{57}$$

where $H_4(x) = H_3(x) + f[x_n, x_n, y_{n-1}, x_{n-1}, x_{n-1}](x - x_n)^2(x - y_{n-1})(x - x_{n-1}))$ and $H_4''(x) = 4f[x_n, x_n, y_{n-1}] + (2f[x_n, x_n, y_{n-1}, x_{n-1}] - 2f[x_n, y_{n-1}x_{n-1}, x_{n-1}])(x_n - y_{n-1})$.

Table 1 has four nonlinear test functions with their roots; two functions are considered from [16] and the other two are taken from [17]. The numerical results illustrated in Table 2 are in accordance with the theory developed in the paper. The absolute errors $|x_n - \alpha|$ are given for our proposed method OM4(2.6), where α is an exact root. All the computations have been performed using the programming package $MATHEMATICA$ 8. The computational order of convergence in [18] is approximated by means of

$$COC \approx \frac{ln|f(x_{n+1})/f(x_n)|}{ln|f(x_n)/f(x_{n-1})|},$$

to confirm the computational efficiency, which established all the theoretical rate of convergence. Our proposed method OM4(2.6) have been used to solve the nonlinear functions and the calculated results are compared with other existing methods of the same nature XW41(16) and XW42(17). Another effective approach to compare the efficiency of methods is CPU time used in the execution of program. At this moment, the CPU time has been calculated by means of the command "$TimeUsed[]$" in $MATHEMATICA$ 8. The CPU time depends on the specification of computer. The computer characteristics are Microsoft Windows 7 Intel Core i3-2330M CPU@ 2.20 GHz with 2 GB of RAM, 64-bit operating system throughout the paper. The mean CPU time is calculated by considering the mean of 30 performance

Table 2 Comparison of absolute error

Ex.	Method	*Guess*	$\lvert x_1-\alpha\rvert$	$\lvert x_2-\alpha\rvert$	$\lvert x_3-\alpha\rvert$	$\lvert x_4-\alpha\rvert$	*COC*	*CPU*
f_1	(53)–(55), $T_0=-0.01, a=8$	1.1	1.9608e−1	1.0145e−3	5.6074e−15	3.5746e−66	4.5475	1.051
	(53)–(56), $T_0=-0.01, a=8$		1.9608e−1	1.5029e−4	5.2406e−19	1.3766e−81	4.3286	1.192
	(53)–(57), $T_0=-0.01, a=8$		1.9608e−1	1.4189e−4	9.1099e−19	5.1407e−80	4.3155	1.146
	(54)–(55), $T_0=-0.01, b=-2$		2.6058e−3	1.8618e−12	1.6587e−54	2.3020e−246	4.5626	1.213
	(54)–(56), $T_0=-0.01, b=-2$		2.6058e−3	8.1833e−14	1.9534e−59	8.6022e−258	4.3478	1.132
	(54)–(57), $T_0=-0.01, b=-2$		2.6058e−3	3.3581e−14	4.6356e−61	1.8739e−264	4.3404	1.087
	(10)–(11), $T_0=-0.01$		2.5073e−3	1.4534e−12	6.5671e−55	3.6750e−248	4.5638	1.055
	(10)–(12), $T_0=-0.01$		2.5073e−3	1.4021e−13	3.2414e−58	2.0670e−252	4.3506	1.223
	(10)–(13), $T_0=-0.01$		2.5073e−3	3.4563e−14	5.9004e−61	5.5157e−264	4.3412	1.113
f_1	(53)–(55), $T_0=-0.01, a=8$	1.0	NC	–	–	–	–	–
	(53)–(56), $T_0=-0.01, a=8$		NC	–	–	–	–	–
	(53)–(57), $T_0=-0.01, a=8$		NC	–	–	–	–	–
	(54)–(55), $T_0=-0.01, b=-2$		1.1214e−2	1.6877e−9	4.3829e−41	3.6091e−185	4.5617	1.175
	(54)–(56), $T_0=-0.01, b=-2$		1.1214e−2	6.8673e−11	9.2680e−47	1.5103e−202	4.3432	1.203
	(54)–(57), $T_0=-0.01, b=-2$		1.1214e−2	4.8757e−11	3.5431e−47	6.8921e−204	4.3364	1.079
	(10)–(11), $T_0=-0.01$		8.8257e−3	5.2252e−10	3.1963e−43	7.7667e−195	4.5638	1.084
	(10)–(12), $T_0=-0.01$		8.8257e−3	6.8912e−11	1.9285e−46	4.5594e−201	4.3506	1.097
	(10)–(13), $T_0=-0.01$		8.8257e−3	1.9076e−11	6.2705e−49	1.5622e−211	4.3412	1.068

Table 2 continued

Ex.	Method	*Guess*	$\lvert x_1-\alpha\rvert$	$\lvert x_2-\alpha\rvert$	$\lvert x_3-\alpha\rvert$	$\lvert x_4-\alpha\rvert$	*COC*	*CPU*
f_2	(53)–(55), $T_0=-0.01, a=8$	1.2	1.9608e−1	9.2459e−2	3.4407e−7	9.8437e−31	4.3061	1.222
	(53)–(56), $T_0=-0.01, a=8$		1.9608e−1	4.4649e−2	1.6626e−8	2.3675e−36	4.3189	1.241
	(53)–(57), $T_0=-0.01, a=8$		1.9608e−1	4.4649e−2	8.8778e−9	1.8552e−37	4.2678	1.150
	(54)–(55), $T_0=-0.01, b=-2$		6.2006e−3	1.2291e−11	1.6706e−51	2.4526e−233	4.5610	1.166
	(54)–(56), $T_0=-0.01, b=-2$		6.2006e−3	1.6121e−12	2.1909e−54	2.1270e−236	4.3474	1.143
	(54)–(57), $T_0=-0.01, b=-2$		6.2006e−3	1.6121e−12	4.3818e−54	1.3613e−234	4.3427	1.249
	(10)–(11), $T_0=-0.01$		1.4358e−3	3.8462e−14	1.1624e−62	4.0834e−284	4.5642	1.237
	(10)–(12), $T_0=-0.01$		1.4358e−3	1.1326e−15	2.8485e−68	5.2766e−297	4.3486	1.162
	(10)–(13), $T_0=-0.01$		1.4358e−3	1.1326e−15	5.6970e−68	3.3770e−295	4.3448	1.255
f_2	(53)–(55), $T_0=-0.01, a=8$	1.1	NC	–	–	–	–	–
	(53)–(56), $T_0=-0.01, a=8$		NC	–	–	–	–	–
	(53)–(57), $T_0=-0.01, a=8$		NC	–	–	–	–	–
	(54)–(55), $T_0=-0.01, b=-2$		2.9502e−1	1.2176e−2	6.1832e−38	1.7358e−171	4.5590	1.250
	(54)–(56), $T_0=-0.01, b=-2$		2.9502e−1	3.2202e−2	5.6567e−40	1.4573e−173	4.3436	1.255
	(54)–(57), $T_0=-0.01, b=-2$		2.9502e−1	3.2202e−2	1.1315e−39	9.3309e−172	4.3371	1.153
	(10)–(11), $T_0=-0.01$		8.8257e−3	9.7124e−13	3.6430e−56	1.9366e−254	4.5658	1.140
	(10)–(12), $T_0=-0.01$		8.8257e−3	2.2043e−14	9.5108e−63	5.3752e−273	4.3471	1.257
	(10)–(13), $T_0=-0.01$		8.8257e−3	2.2043e−14	1.9022e−62	3.4401e−271	4.3430	1.081

Table 2 continued

Ex.	Method	*Guess*	$\lvert x_1-\alpha\rvert$	$\lvert x_2-\alpha\rvert$	$\lvert x_3-\alpha\rvert$	$\lvert x_4-\alpha\rvert$	*COC*	*CPU*
f_3	(53)–(55), $T_0=-0.01, a=8$	0.8	1.9441e−4	3.8295e−20	1.2404e−91	1.1427e−417	4.5606	1.695
	(53)–(56), $T_0=-0.01, a=8$		1.9441e−4	6.6785e−21	6.3522e−92	3.8518e−401	4.3538	1.755
	(53)–(57), $T_0=-0.01, a=8$		1.9441e−4	7.5777e−23	2.3882e−101	3.9617e−442	4.3411	1.715
	(54)–(55), $T_0=-0.01, b=-2$		1.8571e−4	3.1886e−20	5.4403e−92	2.6747e−419	4.5606	1.624
	(54)–(56), $T_0=-0.01, b=-2$		1.8571e−4	5.283e−21	2.8320e−92	1.1425e−402	4.3539	1.788
	(54)–(57), $T_0=-0.01, b=-2$		1.8571e−4	3.5157e−23	5.8886e−103	3.4537e−449	4.3400	1.966
	(10)–(11), $T_0=-0.01$		4.9908e−5	4.4597e−23	4.0414e−105	3.0930e−479	4.5600	1.708
	(10)–(12), $T_0=-0.01$		4.9908e−5	7.7522e−24	7.9976e−105	2.0001e−457	4.3538	1.653
	(10)–(13), $T_0=-0.01$		4.9908e−5	4.8867e−26	1.5919e−115	4.9905e−504	4.3415	1.655
f_3	(53)–(55), $T_0=-0.01, a=8$	0.5	2.2777e−5	9.9037e−24	9.9576e−110	3.6659e−500	4.5615	1.669
	(53)–(56), $T_0=-0.01, a=8$		2.2777e−5	2.6627e−25	1.6365e−111	8.0809e−487	4.3533	1.647
	(53)–(57), $T_0=-0.01, a=8$		2.2777e−5	4.4148e−27	4.1007e−120	3.5030e−524	4.3433	1.845
	(54)–(55), $T_0=-0.01, b=-2$		1.6051e−5	9.6240e−25	1.8263e−112	1.2519e−512	4.5617	1.571
	(54)–(56), $T_0=-0.01, b=-2$		1.6051e−5	6.4339e−26	3.8520e−114	2.9166e−498	4.3540	1.645
	(54)–(57), $T_0=-0.01, b=-2$		1.6051e−5	1.3818e−28	3.5200e−127	3.7519e−555	4.3408	1.834
	(10)–(11), $T_0=-0.01$		1.2311e−5	2.4198e−25	3.1240e−115	2.8997e−525	4.5615	1.593
	(10)–(12), $T_0=-0.01$		1.2311e−5	1.6241e−26	8.7937e−117	8.6234e−510	4.3539	1.793
	(10)–(13), $T_0=-0.01$		1.2311e−5	5.2064e−30	5.9547e−134	7.4537e−585	4.3380	1.839

Table 2 continued

Ex.	Method	*Guess*	$\lvert x_1-\alpha\rvert$	$\lvert x_2-\alpha\rvert$	$\lvert x_3-\alpha\rvert$	$\lvert x_4-\alpha\rvert$	*COC*	*CPU*
f_4	(53)–(55), $T_0=-0.01, a=8$	-1.8	1.6737e−3	3.8204e−15	2.2573e−68	3.5212e−311	4.5616	1.650
	(53)–(56), $T_0=-0.01, a=8$		1.6737e−3	2.2407e−15	1.7946e−67	2.4222e−294	4.3548	1.876
	(53)–(57), $T_0=-0.01, a=8$		1.6737e−3	6.0451e−16	5.6894e−70	7.9010e−305	4.3471	1.960
	(54)–(55), $T_0=-0.01, b=-2$		4.3395e−3	1.0769e−13	6.2802e−62	7.3861e−282	4.5596	1.813
	(54)–(56), $T_0=-0.01, b=-2$		4.3395e−3	4.6428e−14	4.0824e−62	4.2838e−271	4.3487	1.943
	(54)–(57), $T_0=-0.01, b=-2$		4.3395e−3	6.6075e−14	5.2837e−61	1.0796e−265	4.3461	1.733
	(10)–(11), $T_0=-0.01$		5.2677e−4	3.7184e−17	2.2498e−77	3.6554e−352	4.5632	1.610
	(10)–(12), $T_0=-0.01$		5.2677e−4	2.4248e−17	9.2344e−76	2.1740e−330	4.3586	1.648
	(10)–(13), $T_0=-0.01$		5.2677e−4	2.0492e−18	7.9623e−81	3.4330e−352	4.3481	1.747
f_4	(53)–(55), $T_0=-0.01, a=8$	-1.0	NC	–	–	–	–	–
	(53)–(56), $T_0=-0.01, a=8$		NC	–	–	–	–	–
	(53)–(57), $T_0=-0.01, a=8$		NC	–	–	–	–	–
	(54)–(55), $T_0=-0.01, b=-2$		1.4017e−2	1.3621e−10	7.1750e−48	8.9866e−216	4.5577	1.757
	(54)–(56), $T_0=-0.01, b=-2$		1.4017e−2	2.9423e−10	1.2182e−44	2.3476e−194	4.3543	1.749
	(54)–(57), $T_0=-0.01, b=-2$		1.4017e−2	3.3634e−10	2.4137e−44	9.5962e−193	4.3463	1.716
	(10)–(11), $T_0=-0.01$		2.6435e−3	4.9417e−15	1.3099e−67	1.3850e−307	4.5643	1.565
	(10)–(12), $T_0=-0.01$		2.6435e−3	3.7985e−15	8.3615e−66	9.1840e−287	4.3618	1.672
	(10)–(13), $T_0=-0.01$		2.6435e−3	1.0261e−16	2.4056e−73	1.8365e−319	4.3461	1.713

of the program. It can be observed from Table 2, that the results obtained by our proposed method are efficient and show better performance than other existing methods.

4 Conclusion

We have presented a multipoint with memory iterative method for finding simple roots of nonlinear equations with minimum computational effort, of improved convergence order by modifying the existing without memory method. Since our aim is to construct the method of higher order convergence without any additional calculation, so we have used three different approximations of self-correcting parameters, designed by Hermite interpolating polynomials in the fourth-order method to achieve higher order convergence. The R-order of convergence of new with memory iterative methods is increased from 4 to 4.5616, 4.7913 and 5 respectively, without any additional calculation. Our measure for the proposed method is also based on the efficiency index, computational order of convergence (COC) and CPU time. At the end the numerical results confirm validity of theoretical results.

References

1. Wang, X., Zhang, T.: Higher-order Newton-type iterative methods with and without memory for solving nonlinear equations. Math. Commun. **19**, 91–109 (2014)
2. Soleymani, F.: Some optimal iterative methods and their with memory variants. J. Egyp. Math. Soc. **21**, 133–141 (2013)
3. Jaiswal, J.P.: Some class of third- and fourth-order iterative methods for solving nonlinear equations. J. Appl. Math. **2014**, 1–17 (2014)
4. Cordero, A., Torregrosa, J.R., Vassileva, M.P.: A family of modified Ostrowski's method with optimal eighth-order of convergence. Appl. Math. Lett. **24**(12), 2082–2086 (2011)
5. Cordero, A., Lotfi, T., Mahdiani, K., Torregrosa, J.R.: Two optimal general classes of iterative methods with eighth-order. Act. Appl. Math. **134**(1), 61–74 (2014)
6. Kumar, S., Kanwar, V., Tomar, S.K., Singh, S.: Geometrically constructed families of Newton's method for unconstrained optimization and nonlinear equations. Int. J. Math. Math. Sci. Article ID 972537, 1–9 (2011)
7. Petkovic, M.S., Ilic, S., Dzunic, J.: Derivative free two point methods with and without memory for solving nonlinear equations. Appl. Math. Comput. **217**, 1887–1895 (2010)
8. Soleymani, F., Lotfi, T., Tavakoli, E., Khaksar Haghani, F.: Several iterative methods with memory using self-accelerators. Appl. Math. Comput. **254**, 452–458 (2015)
9. Lotfi, T., Assari, P.: New three- and four-parametric iterative with memory methods with efficiency index near 2. Appl. Math. Comput. **270**, 1004–1010 (2015)
10. Lotfi, T., Assari, P.: Two new three and four parametric with memory methods for solving nonlinear equations. Int. J. Ind. Math. **7**(3), 1–8 (2015)
11. Jaiswal, J.P.: Improved bi-parametric derivative free with memory family for solving nonlinear equations. J. Appl. Anal. Comput. **6**(1), 196–206 (2016)
12. Hafiz, M.A., Bahgat, M.S.M.: Solving nonlinear equations using two-step optimal methods. Annu. Rev. Chaos Theory Bifur. Dyn. Sys. **3**, 1–11 (2013)
13. Ortega, J.M., Rheinbolt, W.C.: In Iterative Solution of Nonlinear Equations in Several Variables. Academic Press, New York (1997)
14. Alefeld, G., Herzberger, J.: In Introduction to Interval Computation. Academic Press, New York (1983)
15. Wang, X., Zhang, T.: A new family of Newton-type iterative methods with and without memory for solving nonlinear equations. Calcolo **51**(1), 1–15 (2014)
16. Wang, X., Zhang, T.: Some Newton-type iterative methods with and without memory for solving nonlinear equations. Int. J. Comput. Methods **11**(5), 1–20 (2013)
17. Chun, C., Lee, M.Y.: A new optimal eighth-order family of iterative methods for the solution of nonlinear equations. Appl. Math. Comput. **223**, 506–519 (2013)

18. Weerakoon, S., Fernando, T.G.I.: A variant of Newton's method with accelerated third-order convergence. Appl. Math. Lett. **13**, 87–93 (2000)