mathematics

MDPI

Article

Using Matrix Eigenvalues to Construct an Iterative Method with the Highest Possible Efficiency Index Two

Malik Zaka Ullah [1], Vali Torkashvand [2], Stanford Shateyi [3,*] and Mir Asma [4]

[1] Mathematical Modeling and Applied Computation (MMAC) Research Group, Department of Mathematics, King Abdulaziz University, Jeddah 21589, Saudi Arabia; zmalek@kau.edu.sa
[2] Member of Young Researchers and Elite Club, Shahr-e-Qods Branch, Islamic Azad University, Tehran 37515-374, Iran; torkashvand1978@gmail.com
[3] Department of Mathematics and Applied Mathematics, School of Mathematical and Natural Sciences, University of Venda, P. Bag X5050, Thohoyandou 0950, South Africa
[4] Institute of Mathematical Sciences, Faculty of Science, University of Malaya, Kuala Lumpur 50603, Malaysia; miraszaka@gmail.com
* Correspondence: stanford.shateyi@univen.ac.za

Abstract: In this paper, we first derive a family of iterative schemes with fourth order. A weight function is used to maintain its optimality. Then, we transform it into methods with several self-accelerating parameters to reach the highest possible convergence rate 8. For this aim, we employ the property of the eigenvalues of the matrices and the technique with memory. Solving several nonlinear test equations shows that the proposed variants have a computational efficiency index of two (maximum amount possible) in practice.

Keywords: with-memory method; accelerator parameter; *R*-order convergence; eigenvalues

MSC: 65H05; 41A25

Citation: Ullah, M.Z.; Torkashvand, V.; Shateyi, S.; Asma, M. Using Matrix Eigenvalues to Construct an Iterative Method with the Highest Possible Efficiency Index Two. *Mathematics* **2022**, *10*, 1370. https://doi.org/10.3390/math10091370

Academic Editors: Alicia Cordero Barbero and Juan R. Torregrosa

Received: 9 March 2022
Accepted: 11 April 2022
Published: 20 April 2022

1. Introduction

This paper is concerned with the numerical solution of nonlinear problems having the structure $g(x) = 0$. In fact, we look at iterative approaches to solving nonlinear problems. It is famous that the celebrated Newton's scheme can define $x_{k+1} = x_k - \frac{g(x_k)}{g'(x_k)}$ with the local second convergence rate for simple roots while per iteration it requires one evaluation of the function and one of its first derivative. For some applications, one may refer to [1,2].

A wide range of problems which are not related to nonlinear equations at first sight can be expressed as finding the solution of nonlinear equations in special spaces (e.g., in operator form.) For example, finding approximate-analytic solutions to nonlinear stochastic differential equations [3] is possible via Chaplygin type solvers which are in fact some Newton's iteration in an appropriate operator environment in order to be imposed for solving such equations [4].

Let us recall that the efficiency of the iterative schemes [5] is calculated via $\sqrt[d]{p}$ wherein d is the number of functional evaluations per cycle and p is the speed rate. Besides, for multi-point without-memory iterative schemes, the optimal convergence order is 2^{n-1}, needing n functional evaluations per cycle [6].

Now some definitions are given which will be used later in this work.

A famous fourth-order two-point method without-memory is the King's method, which is given by [7]:

$$\begin{cases} y_k = x_k - \frac{g(x_k)}{g'(x_k)}, \gamma \in \mathbb{R}, \\ x_{k+1} = y_k - \frac{g(y_k)}{g'(x_k)} \frac{g(x_k)+\gamma g(y_k)}{g(x_k)+(\gamma-2)g(y_k)}. \end{cases} \tag{1}$$

The authors of [8] made the following three steps based on Ostrowski's method and the use of the technique of the Chun weight function [9]:

$$\begin{cases} y_k = x_k - \frac{g(x_k)}{g'(x_k)},\ \gamma \in \mathbb{R},\ t_k = \frac{g(z_k)}{g(x_k)},\ u(0) = u'(0) = 1, \\ z_k = y_k - \frac{g(y_k)}{g'(x_k)} \frac{g(x_k)+\gamma g(y_k)}{g(x_k)+(\gamma-2)g(y_k)}, \\ x_{k+1} = z_k - u(t_k) \frac{g[x_k,y_y]g(z_k)}{g[x_k,z_y]g[z_k,y_k]}. \end{cases} \tag{2}$$

Ostrowski's method proposed in [5] has fourth order of convergence as follows:

$$\begin{cases} y_k = x_k - \frac{g(x_k)}{g'(x_k)},\ k \geq 0, \\ x_{k+1} = y_k - \frac{g(y_k)(y_k-x_k)}{2g(y_k)-g(x_k)}. \end{cases} \tag{3}$$

This method can be rewritten as follows:

$$\begin{cases} y_k = x_k - \frac{g(x_k)}{g'(x_k)},\ k \geq 0, \\ x_{k+1} = y_k - \frac{g(y_k)}{g'(x_k)} \frac{g(x_k)}{g(x_k)-2g(y_k)}. \end{cases} \tag{4}$$

This method supports the optimality conjecture of Kung–Traub for the highest possible convergence order for methods without memory. Accordingly, the efficiency index of Newton's and Ostrowski's methods are: $\sqrt{2} \approx 1.414$ and $\sqrt[3]{4} \approx 1.587$, respectively.

In this work, we turn the famous Ostrowski method into a family of Steffensen-like methods, ref. [10]. This technique eliminates the disadvantage of calculating the function derivative. A three self-parameters family of optimal two-step methods is obtained, which uses the weight function technique to maintain the optimality of the without-memory methods. In addition, the matrix eigenvalue technique is employed to prove the convergence order of the proposed methods.

To explain the motivation of the current manuscript clearly we should address why we need to achieve such high precision results, for which applications? The answer is that we mostly do not need high precision. Actually the current study is useful in terms of a theoretical point of view by proposing a general family of methods with memory that possesses a 100% order improvement without any additional functional evaluations. In terms of the application point of view, we employ multiple precision arithmetic in numerical simulations just to re-check the order of convergence in numerical tests. In applications, clearly our method that has a higher order of convergence reaches the convergence radius faster, and gives the final solution in reasonable timing.

We describe the structure of the modified Ostrowski's methods two-step without memory in Section 2. The improvement of the convergence rate of this family is attained via employing several parameters of self-acceleration. Such parameters are computed per loop via the information from the current and the previous loops which help to accelerate the convergence without adding further incorporation of functional evaluations. The efficiency index of the new method is two (the highest efficiency index available). The theoretical proof is presented in Section 3. Computational pieces of evidence are brought forward in Section 4 and uphold the analytical results. Finally, we provide concluding remarks in Section 5.

2. Derivation of Methods and Convergence Analysis

By looking at relation (4), it can be seen that this method uses the derivative of the function in the first and second steps and this shows that the two-point family of schemes (4) achieves the fourth convergence rate employing three evaluations of func-

tions only (viz., $g(x_k)$, $g(y_k)$, and $g'(x_k)$) per full iteration. To derive new methods, we approximate $g'(x_k)$ given in one-step (4) as follows:

$$g'(x_k) \approx g[w_k, x_k] = \frac{g(w_k) - g(x_k)}{w_k - x_k},\ w_k = x_k + \beta g(x_k). \tag{5}$$

In what follows, the derivative $g'(x_k)$ in the second step will be estimated via $\frac{g[y_k, w_k]}{h(t_k)}$, where $h(t_k)$ is a differentiable function that relies on the real variable

$$t_k = \frac{g(y_k)}{g(x_k)}.$$

Thus, it is begun by the scheme (4), the approximation (5), and mentions the following two-point method:

$$\begin{cases} w_k = x_k + \beta g(x_k),\ y_k = x_k - \frac{g(x_k)}{g[w_k, x_k]},\ k \geq 0, \\ x_{k+1} = y_k - H(t_k)\frac{g(x_k)}{g(x_k) - 2g(y_k)}\frac{g(y_k)}{g[y_k, w_k]}. \end{cases} \tag{6}$$

The next theorem illustrates the weight function and under what conditions the convergence rate of (6) will achieve the optimal order four.

Theorem 1. *Let for the open interval D, the function $g : D \subset \mathbb{R} \to \mathbb{R}$ have a simple root $x^* \in D$. As long as the starting point x_0 is close enough to the exact root, then $\{x_k\}$ obtained via (6) tends to x^*. If H is a real function under the assumptions $H''(0) < \infty$, $H'(0) = -1$, $H(0) = 1$ and $\beta \neq 0$ then the fourth order of convergence can be obtained for (6).*

Proof. The proof of this theorem is similar to the way of proving convergence order for similar schemes in the literature, see e.g., ref. [11]. It is hence omitted and we bring the final error equation, which can be written as follows:

$$e_{k+1} = \frac{-1}{2}((1 + \beta g'(x^*))^2 c_2)((-2 + h2 + g'(x^*)\beta(2 + h2)c_2^2 + 2c_3))e_k^4 + O(e_k^5), \tag{7}$$

where $c_i = \frac{1}{i!}\frac{g^{(i)}(x^*)}{g'(x^*)}$ and using relations $h0 = H(0)$, $h1 = H'(0)$, $h2 = H''(0)$. Hence, the fourth-order convergence is established. The proof is ended. □

Some of the functions that satisfy Theorem 1 are as follows:

$$\begin{cases} H_1(t) = 1 - t,\ H_2(t) = \frac{1}{1+t}, \\ H_3(t) = (1 - \frac{t}{2})^2,\ H_4(t) = e^{-t}, \\ H_5(t) = \frac{1+2t}{1+3t},\ H_6(t) = \cos(t) - \sin(t), \\ H_7(t) = \arccos(t),\ H_8(t) = \frac{t^2+1}{1+t}, \\ H_9(t) = e^t - 2t. \end{cases} \tag{8}$$

By considering a new accelerator, the following two-step method can be obtained:

$$\begin{cases} w_k = x_k + \beta g(x_k),\ k \geq 0, \\ y_k = x_k - \frac{g(x_k)}{g[w_k, x_k] + \gamma g(w_k)}, \\ x_{k+1} = y_k - H(t_k)\frac{g(x_k)}{g(x_k) - 2g(y_k)}\frac{g(y_k)}{g[y_k, w_k] + \gamma g(w_k)}. \end{cases} \tag{9}$$

The method (6) can also be converted into a without memory method by adding two self-accelerator parameters to a three-parameter method as follows:

$$\begin{cases} w_k = x_k + \beta g(x_k),\ k \geq 0, \\ y_k = x_k - \frac{g(x_k)}{g[w_k, x_k] + \gamma g(w_k)}, \\ x_{k+1} = y_k - H(t_k)\frac{g(x_k)}{g(x_k) - 2g(y_k)}\frac{g(y_k)}{g[y_k, w_k] + \gamma g(w_k) + \lambda(y_k - x_k)(y_k - w_k)}. \end{cases} \tag{10}$$

Theorem 2. *Having similar conditions as in Theorem 1, then, the iteration methods (9) and (10) have fourth order of convergence and satisfy the following error equations, respectively:*

$$\begin{aligned} e_{k+1} = &-(1 + \beta g'(x^*))^2(\gamma + c_2)((\gamma + c_2)\gamma(1 + \beta g'(x^*)) \\ &+ (-1 + \beta g'(x^*))c_2 + c_3))e_k^4 + O(e_k^5), \end{aligned} \tag{11}$$

and

$$\begin{aligned} e_{k+1} = &-(1 + \beta g'(x^*))^2(\gamma + c_2)(g'(x^*)(1 + \beta g'(x^*))\gamma^2 - \lambda + g'(x^*) \\ &(2c_2 g'(x^*)\beta\gamma + (-1 + g'(x^*)\beta)c_2^2 + c_3))(g'(x^*))^{-1}e_k^4 + O(e_k^5). \end{aligned} \tag{12}$$

Proof. This is proved as in the Theorem 1; hence, it is omitted. □

We also note here that (12) can be rewritten as follows:

$$\begin{aligned} e_{k+1} = &-(1 + \beta g'(x^*))^2(\gamma + c_2)(g'(x^*)(1 + \beta g'(x^*))\gamma^2 - \lambda - g'(x^*)c_2^2 \\ &(g'(x^*)\beta + 1) - c_2^2(1 + g'(x^*)\beta) + c_3 g'(x^*)))(g'(x^*))^{-1}e_k^4 + O(e_k^5). \end{aligned} \tag{13}$$

3. Further Improvements via the Concept of Methods with Memory

3.1. One-Parametric Method

It is observed by (7) that the convergence rate order for the presented methods (6) is 4 when $\beta \neq \frac{1}{g'(x^*)}$. We could approximate the parameter β by β_k:

$$\beta_k = \frac{-1}{g'(x^*)} \approx -\frac{1}{N_3'(x_k)}, \tag{14}$$

where $N_3(x_k)$ are defined as follows:

$$N_3(x_k) = N_3(t; x_k, x_{k-1}, w_{k-1}, y_{k-1}). \tag{15}$$

Combining (6) with (14), one is able to propose a family of two-point Ostrowski–Steffensen-type methods with memory as comes next:

$$\begin{cases} \beta_k = \frac{1}{N_3'(x_k)},\ k = 1, 2, 3, \cdots, \\ w_k = x_k + \beta_k g(x_k),\ y_k = x_k - \frac{g(x_k)}{g[x_k, w_k]},\ k = 0, 1, 2, \cdots, \\ t_k = \frac{g(y_k)}{g(x_k)},\ H(0) = 1,\ H'(0) = 1, \\ x_{k+1} = y_k - H(t_k)\frac{g(y_k)}{g[y_k, w_k]}\frac{g(x_k)}{g(x_k) - 2g(y_k)}. \end{cases} \tag{16}$$

Theorem 3. *This theorem has similar conditions to Theorem 1. As long as β_k in (16) is computed recursively by (14), then the convergence R-order would be six.*

Proof. The matrix approach discussed initially in [12] is now used to obtain the rate of convergence for such an accelerated method. Recalling the lower bound for the rate of convergence in such cases on the single-step s-point procedure technique (14), i.e.,

$x_k = \varphi(x_{k-1}, x_{k-2}, \cdots, x_{k-s})$ would be the spectral radius of $M^{(s)} = (\mathfrak{m}_{ij})$, corresponding to the method by having:

$$\begin{cases} \mathfrak{m}_{i,i-1} = 1,\ i = 2, 3, \cdots, s, \\ \mathfrak{m}_{1,j} = \text{amount of information required at point} x_{k-j},\ j = 1, 2, \cdots, s, \\ \mathfrak{m}_{i,j} = 0 \text{ otherwise.} \end{cases} \tag{17}$$

Then the spectral radius of $M = M_1.M_2.\cdots.M_s$ would be the lower bound for the s-step method $\varphi = \varphi_1 \cdot \varphi_2 \circ \cdots \circ \varphi_s$. We can state each of the estimates x_{k+1}, y_k, and w_k as a function of available information $g(y_k), g(w_k)$ and $g(x_k)$ from the k-th iterate and $g(y_{k-1})$, $g(w_{k-1})$ and $g(x_{k-1})$ from the past iterate. From the relations (16) and (17), we create the corresponding matrices as follows:

$$x_{k+1} = \varphi_1(y_k, w_k, x_k, y_{k-1}); \Rightarrow M_1 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$y_k = \varphi_2(w_k, x_k, y_{k-1}, w_{k-1}); \Rightarrow M_2 = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$w_k = \varphi_3(x_k, y_{k-1}, w_{k-1}, x_{k-1}); \Rightarrow M_3 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Hence, we obtain

$$M = M_1 M_2 M_3 = \begin{pmatrix} 4 & 4 & 0 & 0 \\ 2 & 2 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix},$$

for which its eigenvalues are $(6, 0, 0, 0)$. It is derived that the rate of convergence would be six for the methods with memory (16). The proof is complete now. □

3.2. Two-Parametric Method

Now, similar to the prior case, we build the following derivative-free with the memory method from (9):

$$\begin{cases} \beta_k = \frac{1}{N_3'(x_k)},\ \gamma_k = \frac{-N_4''(w_k)}{2N_4'(w_k)},\ k = 1, 2, 3, \cdots, \\ y_k = x_k - \frac{g(x_k)}{g[x_k, w_k] + \gamma_k g(w_k)},\ w_k = x_k + \beta_k g(x_k),\ k = 0, 1, 2, \cdots, \\ t_k = \frac{g(y_k)}{g(x_k)},\ H(0) = 1,\ H'(0) = 1, \\ x_{k+1} = y_k - H(t_k) \frac{g(y_k)}{g[y_k, w_k] + \gamma_k g(w_k)} \frac{g(x_k)}{g(x_k) - 2g(y_k)}. \end{cases} \tag{18}$$

Theorem 4. *Let x_0 be an initial approach close enough to x^* of $g(x) = 0$. If the parameters β_k and γ_k will compute recursively, then the convergence R-order of (18) is at least 7.*

Proof. Using appropriate matrixes in the proof of Theorem 3 and substituting them into the goal matrix, we obtain that (18) has seventh order of convergence. In fact, the proof is similar to the proof of Theorem 3 and hence it is omitted. □

3.3. Tri-Parametric Method

The method (10) with memory could be expressed in what follows:

$$\begin{cases} \beta_k = \frac{1}{N_3'(x_k)}, \gamma_k = \frac{-N_4''(w_k)}{2N_4'(w_k)}, \lambda_k = \frac{N_5'''(y_k)}{6}, k = 1,2,3,\cdots, \\ y_k = x_k - \frac{g(x_k)}{g[x_k,w_k]+\gamma_k g(w_k)}, w_k = x_k + \beta_k g(x_k), k = 0,1,2,\cdots, \\ t_k = \frac{g(y_k)}{g(x_k)}, H(0) = 1, H'(0) = 1, \\ x_{k+1} = y_k - H(t_k)\frac{g(y_k)}{g[y_k,w_k]+\gamma_k g(w_k)+\lambda_k(y_k-x_k)(y_k-w_k)}\frac{g(x_k)}{g(x_k)-2g(y_k)}. \end{cases} \tag{19}$$

Now, we establish a theorem for determining the rate of (19).

Theorem 5. *With the hypotheses of Theorem 3 and that the three parameters β_k, γ_k and λ_k have been recursively calculated in (19), then the convergence rate of the with-memory method suggested in (19) is 7.53.*

Proof. From the relation (19) and similar to that used in Theorem 3, we construct the corresponding matrix as follows:

$$M = M_1M_2M_3 = \begin{pmatrix} 4 & 4 & 4 & 4 & 0 & 0 \\ 2 & 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix},$$

and its eigenvalues are $(\frac{1}{2}(7+\sqrt{65}), \frac{1}{2}(7-\sqrt{65}), 0, 0, 0, 0)$. Thus, $\frac{1}{2}(7+\sqrt{65}) \approx 7.53$ is the rate of convergence. □

We will present the process of the work in the following four sections until the maximum degree of convergence (i.e., 100%):

(I) Now, we consider three parameters' iterative methods as follows:

$$\begin{cases} \beta_k = \frac{1}{N_6'(x_k)}, \gamma_k = \frac{-N_7''(w_k)}{2N_7'(w_k)}, \lambda_k = \frac{N_8'''(y_k)}{6}, k = 2,3,4,\cdots, \\ y_k = x_k - \frac{g(x_k)}{g[x_k,w_k]+\gamma_k g(w_k)}, w_k = x_k + \beta_k g(x_k), k = 0,1,2,\cdots, \\ t_k = \frac{g(y_k)}{g(x_k)}, H(0) = 1, H'(0) = 1, \\ x_{k+1} = y_k - H(t_k)\frac{g(y_k)}{g[y_k,w_k]+\gamma_k g(w_k)+\lambda_k(y_k-x_k)(y_k-w_k)}\frac{g(x_k)}{g(x_k)-2g(y_k)}. \end{cases} \tag{20}$$

Theorem 6. *Having the same conditions as in Theorems 1 and 5, then (20) converges to x^* with the rate of convergence 7.77.*

Proof. In a similar fashion, one obtains that

$$M = \begin{pmatrix} 4 & 4 & 4 & 4 & 4 & 0 & 0 \\ 2 & 2 & 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix},$$

for which its eigenvalues are $\left(\frac{1}{2}(7+\sqrt{73}), \frac{1}{2}(7-\sqrt{73}), 0, 0, 0, 0, 0\right)$, which states that the order of the with-memory methods (20) is 7.77. □

(II) Now, we study tri-parametric iterative methods as follows:

$$\begin{cases} \beta_k = \frac{1}{N_9'(x_k)},\ \gamma_k = \frac{-N_{10}''(w_k)}{2N_{10}'(w_k)},\ \lambda_k = \frac{N_{11}'''(y_k)}{6},\ k = 3,4,5,\cdots, \\ y_k = x_k - \frac{g(x_k)}{g[x_k,w_k]+\gamma_k g(w_k)},\ w_k = x_k + \beta_k g(x_k),\ k = 0,1,2,\cdots, \\ t_k = \frac{g(y_k)}{g(x_k)},\ H(0) = 1,\ H'(0) = 1, \\ x_{k+1} = y_k - H(t_k)\frac{g(y_k)}{g[y_k,w_k]+\gamma_k g(w_k)+\lambda_k(y_k-x_k)(y_k-w_k)}\frac{g(x_k)}{g(x_k)-2g(y_k)}. \end{cases} \tag{21}$$

Theorem 7. *With the hypotheses of Theorem 3 and that the three parameters β, γ and λ have been recursively calculated in (21), then (21) converges to x^* with the convergence order 7.89.*

Proof. Proving this theorem is similar to that of Theorem 3. □

(III) Now, we consider three parameters' iterative methods as follows:

$$\begin{cases} \beta_k = \frac{1}{N_{12}'(x_k)},\ \gamma_k = \frac{-N_{13}''(w_k)}{2N_{13}'(w_k)}, \lambda_k = \frac{N_{14}'''(y_k)}{6},\ k = 4,5,6,\cdots, \\ y_k = x_k - \frac{g(x_k)}{g[x_k,w_k]+\gamma_k g(w_k)}, w_k = x_k + \beta_k g(x_k),\ k = 0,1,2,\cdots, \\ t_k = \frac{g(y_k)}{g(x_k)},\ H(0) = 1,\ H'(0) = 1, \\ x_{k+1} = y_k - H(t_k)\frac{g(y_k)}{g[y_k,w_k]+\gamma_k g(w_k)+\lambda_k(y_k-x_k)(y_k-w_k)}\frac{g(x_k)}{g(x_k)-2g(y_k)}, \end{cases} \tag{22}$$

where $N_{12}(x_k)$, $N_{13}(w_k)$ and $N_{14}(y_k)$ are defined as follows:

$$\begin{cases} N_{12}(x_k) = N_{12}(t; x_k, w_{k-1}, y_{k-1}, x_{k-1}, y_{k-2}, w_{k-2}, \cdots, w_{k-4}, y_{k-4}, x_{k-4}), \\ N_{13}(w_k) = N_{13}(t; w_k, x_k, w_{k-1}, y_{k-1}, x_{k-1}, y_{k-2}, w_{k-2}, \cdots, w_{k-4}, y_{k-4}, x_{k-4}), \\ N_{14}(y_k) = N_{14}(t; y_k, w_k, x_k, w_{k-1}, y_{k-1}, x_{k-1}, y_{k-2}, w_{k-2}, \cdots, w_{k-4}, y_{k-4}, x_{k-4}). \end{cases} \tag{23}$$

Theorem 8. *Having the same assumptions as in Theorem 1, then the proposed family with memory method defined by (22) has R-order 7.94.*

Proof. From the relation (22) and similar to that used in the previous section, we derive the associated matrices as comes next:

$$x_{k+1} = \varphi_1(y_k, w_k, x_k, y_{k-1}, w_{k-1}, x_{k-1}, y_{k-2}, w_{k-3}, x_{k-3});\ \Rightarrow$$

$$M_1 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

$$y_k = \varphi_2(w_k, x_k, y_{k-1}, w_{k-1}, x_{k-1}, y_{k-2}, w_{k-2}, x_{k-2}, y_{k-3});\ \Rightarrow$$

$$M_2 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

$$w_k = \varphi_3(x_k, y_{k-1}, w_{k-1}, x_{k-1}, y_{k-2}, w_{k-2}, x_{k-2}, y_{k-3}, w_{k-3}); \Rightarrow$$

$$M_3 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

So, we obtain

$$M = M_1 M_2 M_3 = \begin{pmatrix} 4 & 4 & 4 & 4 & 4 & 4 & 4 & 0 & 0 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

The only eigenvalue of the matrix M that is positive and real is the number 7.94. It follows that the convergence rate of (22) is 7.94. □

(IV) At the end of this section, we have presented the most important theorem of this paper, which has the highest degree of convergence of a Ostrowski-like two-point method, i.e., 7.97:

$$\begin{cases} \beta_k = \frac{1}{N'_{15}(x_k)}, \gamma_k = \frac{-N''_{16}(w_k)}{2N'_{16}(w_k)}, \lambda_k = \frac{N'''_{17}(y_k)}{6}, \ k = 5, 6, 7, \cdots, \\ y_k = x_k - \frac{g(x_k)}{g[x_k, w_k] + \gamma_k g(w_k)}, \ w_k = x_k + \beta_k g(x_k), \ k = 0, 1, 2, \cdots, \\ t_k = \frac{g(y_k)}{g(x_k)}, \ H(0) = 1, \ H'(0) = 1, \\ x_{k+1} = y_k - H(t_k) \frac{g(y_k)}{g[y_k, w_k] + \gamma_k g(w_k) + \lambda_k (y_k - x_k)(y_k - w_k)} \frac{g(x_k)}{g(x_k) - 2g(y_k)}. \end{cases} \tag{24}$$

Theorem 9. *With the hypotheses of Theorem 3 and that the three parameters β, γ and λ have recursively calculated, then (24) has R-order 7.97 ≈ 8 and its efficiency index is $7.97^{\frac{1}{3}} \approx 2$.*

Proof. From the relation (24) and similar to that used in the previous section, we construct the corresponding matrices as follows:

$$x_{k+1} = \varphi_1(y_k, w_k, x_k, y_{k-1}, w_{k-1}, x_{k-1}, y_{k-2}, w_{k-2}, x_{k-2}, y_{k-3});$$

$$\Rightarrow M_1 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

$$y_k = \varphi_2(w_k, x_k, y_{k-1}, w_{k-1}, x_{k-1}, y_{k-2}, w_{k-2}, x_{k-2}, y_{k-3}, w_{k-3});$$

$$\Rightarrow M_2 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

$$w_k = \varphi_3(x_k, y_{k-1}, w_{k-1}, x_{k-1}, y_{k-2}, w_{k-2}, x_{k-2}, y_{k-3}, w_{k-3}, x_{k-3}); \Rightarrow$$

$$M_3 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

Thus, we get

$$M = M_1 M_2 M_3 = \begin{pmatrix} 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 0 & 0 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix},$$

and its eigenvalues are $(7.97243, -0.48621 + 0.71846i, -0.48621 + 0.71846i, 0, 0, 0, 0, 0, 0, 0)$. This states that $7.97 \approx 8$ is the analytical order and the efficiency index is $7.97^{\frac{1}{3}} \approx 2$. □

4. Numerical Results

The principal purpose of numerical examples is to verify the validity of the theoretical developments through a variety of test examples using high accuracy computations by use of the Mathematica program. All computations were performed using Mathematica 11 [13].

In the tables, the abbreviations Div, TNE and Iter are used as follows:

TNE: Total Number of Evaluations required for a method to do the specified iterations;
Iter: The number of iterations;
The errors $|x_k - \alpha|$ of estimations to the simple zeros of $g_i(x)$, $i = 1, 2, 3$;
The computational order of convergence (r_c) [14] can be calculated via:

$$\begin{cases} r_c = \dfrac{\log|g(x_k)/g(x_{k-1})|}{\log|g(x_{k-1})/g(x_{k-2})|}, \\ \text{COC} = \dfrac{\log|(x_k - x^*)/(x_{k-1} - x^*)|}{\log|(x_{k-1} - x^*)/(x_{k-2} - x^*)|}. \end{cases} \tag{25}$$

We shall check the effectiveness of the new without and with memory methods. We employ the presented methods (6), (16), (18), (19) and (24) denoted by TM4, TM6, TM7, TM7.5 and TM8, respectively (for $\beta_0 = \gamma_0 = \lambda_0 = 0.01$), to solve some nonlinear equations. We compared our methods and some known methods as follows: Campos et al. (CCTVM) [15], Choubey–Jaiswal (CJM) [16], Chun's method (CM) [17], Cordero et al. (CLKTM) [18], Cordero et al. (CLTAM) [19], Jaiswal's method (JM) [20], Jarratt's method (JM) [21], Kung–Traub's method (KTM) [6], Maheshwari's method (MM) [22], Kansal et al.'s method (KKBM) [23], Lalehchini et al.'s method (LLMM) [24], Mohammadi et al.'s method (MLAM) [25], Ostrwoski's method (OM) [5], Soleymani et al.'s method (SLTKM) [11], Torkashvand–Kazemi (TKM) [26], Traub's method (TM) [27], Wang's method (WM) [28] and Zafar et al.'s method (ZYKZM) [29].

In Tables 1–7, we show the numerical results obtained by applying the different methods with a memory for approximating the solution of $g_i(x) = 0$, $i = 1, 2, 3$, given as follows:

$$\begin{cases} g_1(x) = x^5 + x^4 + 4x^2 - 15, \ x^* \approx 1.34, \ x_0 = 1.1, \\ g_2(x) = x^3 + 4x^2 - 10, \ x^* \approx 1.36, \ x_0 = 1, \\ g_3(x) = 10xe^{-x^2} - 1, \ x^* \approx 1.67, \ x_0 = 1. \end{cases}$$

Here, $\approx$ stands for an approximation of the solution that we have written only to provide an overview of the solution. All calculations are performed using 2000 floating point arithmetic in Wolfram Mathematica. This means that we care about very small numbers in terms of magnitude and we do not allow the programming package to consider them as zero automatically. Actually, higher orders can only be seen in the convergence phase and more clearly in high precision computing. The numerical results shown in Tables 1–7 confirmed the theoretical discussions and the efficiency of the proposed scheme under different choices of the weight functions.

Table 1. Comparison of various iterative schemes (first part).

Functions		**OM** [5]	**JM** [21]	**KTM** [6]	**MM** [22]	**CM** [17]
g_1, $x_0 = 1.1$	$\lvert x_{k+1} - x_k \rvert$	1.47E-43	3.75E-43	5.39E-31	1.08E-18	1.47E-43
	$\lvert g(x_{k+1}) \rvert$	9.19E-171	4.04E-169	5.40E-120	2.13E-703	9.19E-171
	Iter	4	4	4	4	4
	r_c	4.00	4.00	4.00	3.99	4.00
g_2, $x_0 = 1$	$\lvert x_{k+1} - x_k \rvert$	3.60E-47	3.60E-47	3.36E-38	3.85E-28	3.60E-47
	$\lvert g(x_{k+1}) \rvert$	2.45E-186	2.45E-186	4.37E-150	1.60E-109	2.45E-186
	Iter	4	4	4	4	4
	r_c	4.00	4.00	4.00	4.00	4.00
g_3, $x_0 = 1$	$\lvert x_{k+1} - x_k \rvert$	1.56E-29	4.40E-24	7.32E-28	3.43E-26	1.56E-29
	$\lvert g(x_{k+1}) \rvert$	1.35E-115	7.72E-94	1.33E-108	1.31E-101	1.35E-115
	Iter	4	4	4	4	4
	r_c	4.00	4.00	4.00	4.00	4.00

Table 2. Comparison of various iterative schemes (second part).

Functions		TM4 (6), $H_1(t)$	TM4 (6), $H_2(t)$	TM4 (6), $H_3(t)$	TM4 (6), $H_4(t)$	TM4 (6), $H_5(t)$
g_1, $x_0 = 1.1$	$\lvert x_{k+1} - x_k \rvert$	4.24E-12	0E-0	2.89E-10	7.74E-8	-
	$\lvert g(x_{k+1}) \rvert$	1.80E-45	6.39E-14	1.62E-37	1.86E-27	4.98E-12
	Iter	3	3	3	3	3
	r_c	3.99	4.11	4.00	4.00	3.83
g_2, $x_0 = 1$	$\lvert x_{k+1} - x_k \rvert$	7.94E-12	2.17E-9	6.57E-15	8.14E-15	-
	$\lvert g(x_{k+1}) \rvert$	6.12E-45	3.42E-35	1.47E-57	3.57E-59	7.01E-11
	Iter	3	3	3	3	3
	r_c	3.99	4.00	3.99	3.97	3.63
g_3, $x_0 = 1$	$\lvert x_{k+1} - x_k \rvert$	8.11E-9	6.13E-10	5.94E-9	3.76E-9	1.81E-8
	$\lvert g(x_{k+1}) \rvert$	9.42E-33	7.08E-39	2.01E-33	2.12E-34	4.87E-31
	Iter	3	3	3	3	3
	r_c	4.00	4.07	4.00	4.00	4.01

Table 3. Comparison of various iterative schemes (third part).

Functions		TM6 (16), $H_1(t)$	TM6 (16), $H_2(t)$	TM6 (16), $H_3(t)$	TM6 (16), $H_4(t)$	TM6 (16), $H_5(t)$
g_1, $x_0 = 1.1$	$\lvert x_{k+1} - x_k \rvert$	1.02E-90	9.59E-38	7.76E-85	1.60E-63	1.64E-36
	$\lvert g(x_{k+1}) \rvert$	1.05E-538	7.07E-221	1.98E-503	1.55E-381	1.82E-219
	Iter	4	4	4	4	4
	r_c	6.00	6.00	6.00	6.00	6.00
g_2, $x_0 = 1$	$\lvert x_{k+1} - x_k \rvert$	1.70E-100	1.09E-78	1.58E-125	8.85E-104	3.53E-28
	$\lvert g(x_{k+1}) \rvert$	2.03E-599	1.41E-468	1.28E-749	3.93E-619	1.58E-170
	Iter	4	4	4	4	4
	r_c	6.00	6.00	6.00	6.00	6.00
g_3, $x_0 = 1$	$\lvert x_{k+1} - x_k \rvert$	2.96E-84	1.97E-84	2.69E-84	2.43E-84	7.65E-85
	$\lvert g(x_{k+1}) \rvert$	1.246E-501	1.06E-502	6.92E-502	3.77E-502	3.67E-505
	Iter	4	4	4	4	4
	r_c	6.00	6.00	6.00	6.00	6.00

Table 4. Comparison of various iterative schemes (fourth part).

Functions		TM7 (18), $H_1(t)$	TM7 (18), $H_2(t)$	TM7 (18), $H_3(t)$	TM7 (18), $H_4(t)$	TM7 (18), $H_5(t)$
g_1, $x_0 = 1.1$	$\lvert x_{k+1} - x_k \rvert$	8.86E-130	1.66E-55	2.63E-119	2.95E-92	3.80E-56
	$\lvert g(x_{k+1}) \rvert$	3.69E-903	3.02E-383	7.51E-830	1.68E-640	9.89E-388
	Iter	4	4	4	4	4
	r_c	7.00	7.00	7.00	7.00	7.00
g_2, $x_0 = 1$	$\lvert x_{k+1} - x_k \rvert$	5.63E-147	6.47E-117	7.66E-186	1.89E-150	3.78E-54
	$\lvert g(x_{k+1}) \rvert$	3.62E-1033	6.95E-816	2.26E-1298	1.25E-1050	1.61E-376
	Iter	4	4	4	4	4
	r_c	7.00	7.00	7.00	7.00	7.00
g_3, $x_0 = 1$	$\lvert x_{k+1} - x_k \rvert$	3.32E-119	1.70E-119	2.83E-119	2.40E-119	3.43E-120
	$\lvert g(x_{k+1}) \rvert$	4.38E-827	3.99E-830	1.43E-828	4.52E-829	5.48E-835
	Iter	4	4	4	4	4
	r_c	7.00	6.99	7.00	7.00	7.00

Table 5. Comparison of various iterative schemes (fifth part).

Functions		TM7.5 (19), $H_1(t)$	TM7.5 (19), $H_2(t)$	TM7.5 (19), $H_3(t)$	TM7.5 (19), $H_4(t)$	TM7.5 (19), $H_5(t)$
g_1, $x_0 = 1.1$	$\lvert x_{k+1} - x_k \rvert$	1.08E-160	2.53E-69	5.73E-147	5.42E-114	2.62E-69
	$\lvert g(x_{k+1}) \rvert$	1.97E-1205	1.11E-518	7.52E-1102	7.82E-584	2.70E-516
	Iter	4	4	4	4	4
	r_c	7.51	7.51	7.50	7.50	7.47
g_2, $x_0 = 1$	$\lvert x_{k+1} - x_k \rvert$	1.58E-188	2.99E-148	3.39E-237	3.84E-192	1.21E-64
	$\lvert g(x_{k+1}) \rvert$	5.80E-1505	9.32E-1183	2.50E-1894	6.85E-1534	6.55E-514
	Iter	4	4	4	4	4
	r_c	8.00	8.00	8.00	8.00	8.00
g_3, $x_0 = 1$	$\lvert x_{k+1} - x_k \rvert$	4.23E-137	1.89E-137	3.49E-137	2.87E-137	2.78E-138
	$\lvert g(x_{k+1}) \rvert$	1.97E-1027	4.76E-1030	4.70E-1028	1.07E-1028	2.63E-1036
	Iter	4	4	4	4	4
	r_c	7.51	7.51	7.51	7.51	7.50

Table 6. Comparison of various iterative schemes (sixth part).

Functions		TM8 (24), $H_1(t)$	TM8 (24), $H_2(t)$	TM8 (24), $H_3(t)$	TM8 (24), $H_4(t)$	TM8 (24), $H_5(t)$
g_1, $x_0 = 1.1$	$\lvert x_{k+1} - x_k \rvert$	3.27E-167	2.00E-69	7.97E-152	5.40E-117	1.11E-69
	$\lvert g(x_{k+1}) \rvert$	4.02E-1331	8.04E-549	4.95E-1208	2.19E-929	7.19E-551
	Iter	4	4	4	4	4
	r_c	8.00	8.00	8.003	8.00	8.00
g_2, $x_0 = 1$	$\lvert x_{k+1} - x_k \rvert$	1.58E-188	2.99E-148	3.39E-237	3.84E-192	1.21E-64
	$\lvert g(x_{k+1}) \rvert$	5.80E-1505	9.32E-1183	2.50E-1894	6.85E-1534	6.55E-514
	Iter	4	4	4	4	4
	r_c	8.00	8.00	8.003	8.00	8.00
g_3, $x_0 = 1$	$\lvert x_{k+1} - x_k \rvert$	3.19E-144	1.42E-144	2.63E-144	2.16E-144	2.06E-145
	$\lvert g(x_{k+1}) \rvert$	1.31E-1149	2.04E-1152	2.83E-1150	5.79E-1151	3.95E-1159
	Iter	4	4	4	4	4
	r_c	8.00	8.00	8.003	8.00	8.00

Table 7. Comparison of various iterative schemes (seventh part).

Functions		CLKTM ($b = 1$) [18]	CLTAMM ($A = 1$) [19]	KKBM (a = 1) Cas 1 [23]	ZYKZM, *Method F*1 [29]	TM8 (24), $H_6(t)$
g_1, $x_0 = 1.1$	$\lvert x_{k+1} - x_k \rvert$	3.15E-84	2.32E-115	1.21E-106	2.46E-108	5.90E-111
	$\lvert g(x_{k+1}) \rvert$	4.20E-506	2.90E-802	3.44E-741	4.11E-810	4.50E-881
	Iter	4	4	4	4	4
	r_c	6.00	7.00	7.00	7.49	8.00
g_2, $x_0 = 1$	$\lvert x_{k+1} - x_k \rvert$	1.85E-100	2.26E-177	7.02E-157	1.96E-154360	1.47E-159
	$\lvert g(x_{k+1}) \rvert$	1.43E-599	4.91E-1416	1.23E-1095	3.15E-1232	3.19E-1273
	Iter	4	4	4	4	4
	r_c	6.00	8.00	7.00	8.00	8.00
g_3, $x_0 = 1$	$\lvert x_{k+1} - x_k \rvert$	3.11E-87	8.95E-112	1.85E-119	8.71E-136	4.69E-144
	$\lvert g(x_{k+1}) \rvert$	1.50E-520	2.64E-777	7.30E-830	4.42E-1025	2.83E-1148
	Iter	4	4	4	4	4
	r_c	6.00	6.99	6.99	7.51	8.00

The question may arise now of do we really need to use the small numbers (e.g., 2.83E-1148 which stands for 2.83×10^{-1148} in Table 7)? The answer is 'no'; to illustrate, we must state that sometimes in applications, results up to at most 100 digits are enough. However, here we used such a number of high floating point arithmetic on purpose to check the computational order of convergence (25). In fact, for higher order methods, the higher speed can only be seen in the number of meaningful decimal places when the method takes several iterations.

In addition, in Table 8 a comparison among various schemes is given, which again states that the proposed methods with memory possess a higher computational efficiency index and can be employed in solving nonlinear equations.

Table 8. Comparison improvement of convergence order of the proposed method with other schemes.

With Memory Methods	Number of Sub-Steps	Optimal Order	COC	Percentage Increase
CCTVM [15]	2	4.00	4.24	5.9%
CJM [16]	2	4.00	4.56	14.03%
CJM [16]	2	4.00	4.79	19.78%
CJM [16]	2	4.00	5.00	20%
CJM [16]	3	8.00	9.00	12.5%
CJM [16]	3	8.00	9.58	19.79%
CJM [16]	3	8.00	9.80	22.44%
CJM [16]	3	8.00	10.00	25%
CLKTM [18]	2	4.00	6.00	50%
CLTAMM [19]	2	4.00	7.00	75%
JM [20]	2	4.00	7.00	75%
JM [20]	3	8.00	14.00	75%
KKBM [23]	2	4.00	7.00	75%
LLMM [24]	2	4.00	6.32	57.93%
MLAM [25]	2	4.00	5.95	48.75%
SLTKM [11]	2	4.00	7.22	80.58%
SLTKM [11]	2	4.00	12.00	50%
TKM [26]	3	8.00	14.00	75%
TKM [26]	4	16.00	28.00	75%
TM [27]	1	2.00	2.41	20.5%
WM [28]	2	4.00	4.24	5.75%
WM [28]	2	4.00	4.45	11.23%
WZM [30]	2	4.00	4.56	14.03%
WZM [30]	3	8.00	10.13	26.64%
ZYKZM [29]	2	4.00	7.5	88.28%
(16)	2	4.00	6.00	50%
(18)	2	4.00	7.00	75%
(19)	2	4.00	7.53	88.28%
(20)	2	4.00	7.77	94.25%
(21)	2	4.00	7.89	97.25%
(23)	2	4.00	7.94	98.5%
(24)	2	4.00	7.97	99.25%

We end this section by pointing out that the extension of our methods for the system of nonlinear equations (see some application of nonlinear system in [31–33]) require the computation of a divided difference operator (DDO) which would be a dense matrix. This dense structure of the DDO matrix restricts the usefulness of such methods. Because of this, we consider our proposed methods only for the scalar case.

5. Conclusions

In this paper, we have used the idea of the weight function and have turned Ostrowski's method into an optimal order method. We have constructed the with-memory methods by using the same number of evaluations that do not require the calculation of a function derivative. Then, with the advent of accelerator parameters and eigenvalues of matrices, we created the with memory methods with higher orders. By interpolatory accelerator parameters, the methods with memory reached up to 100% convergence improvement. Numerical tests were intended to verify the better performance of the proposed methods over the others. Employing such an efficient numerical scheme for practical problems in solving stochastic differential equations [34] is worth investigating in future work.

Author Contributions: Conceptualization, M.Z.U.; Data curation, V.T.; Formal analysis, M.Z.U., S.S. and M.A.; Funding acquisition, S.S.; Investigation, M.Z.U., V.T. and M.A.; Methodology, M.Z.U. and V.T.; Project administration, V.T. and M.A.; Supervision, V.T. and S.S.; Visualization, M.A. All authors have read and agreed to the published version of the manuscript.

Funding: The Deanship of Scientific Research (DSR) at King Abdulaziz University, Jeddah, Saudi Arabia has funded this project, under grant no. (KEP-48-130-42).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: For data availability statement, we state that data sharing is not applicable to this article as no new data were used in this study. All used data has clearly been mentioned in the text.

Acknowledgments: The Deanship of Scientific Research (DSR) at King Abdulaziz University, Jeddah, Saudi Arabia has funded this project, under grant no. (KEP-48-130-42).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Liu, L.; Cho, S.Y.; Yao, J.C. Convergence analysis of an inertial Tseng's extragradient algorithm for solving pseudomonotone variational inequalities and applications. *J. Nonlinear Var. Anal.* **2021**, *5*, 627–644.
2. Alsaedi, A.; Broom, A.; Ntouyas, S.K.; Ahmad, B. Existence results and the dimension of the solution set for a nonlocal inclusions problem with mixed fractional derivatives and integrals. *J. Nonlinear Funct. Anal.* **2020**, *2020*, 28.
3. Itkin, A.; Soleymani, F. Four-factor model of quanto CDS with jumps-at-default and stochastic recovery. *J. Comput. Sci.* **2021**, *54*, 101434. [CrossRef]
4. Soheili, A.R.; Amini, M.; Soleymani, F. A family of Chaplygin-type solvers for Itô stochastic differential equations. *Appl. Math. Comput.* **2019**, *340*, 296–304. [CrossRef]
5. Ostrowski, A.M. *Solution of Equations and Systems of Equations*; Academic Press: New York, NY, USA, 1960.
6. Kung, H.T.; Traub, J.F. Optimal order of one-point and multipoint iteration. *J. ACM* **1974**, *21*, 643–651. [CrossRef]
7. King, R.F. A family of fourth order methods for nonlinear equations. *SIAM J. Numer. Anal.* **1973**, *10*, 876–879. [CrossRef]
8. Sharma, J.R.; Sharma, R. A new family of modified Ostrowski's methods with accelerated eighth order convergence. *Numer. Algorithms* **2010**, *54*, 445–458. [CrossRef]
9. Chun, C.; Lee, M.Y. A new optimal eighth-order family of iterative methods for the solution of nonlinear equations. *Appl. Math. Comput.* **2013**, *223*, 509–519. [CrossRef]
10. Torkashvand, V.; Lotfi, T.; Araghi, M.A.F. A new family of adaptive methods with memory for solving nonlinear equations. *Math. Sci.* **2019**, *13*, 1–20. [CrossRef]
11. Soleymani, F.; Lotfi, T.; Tavakoli, E.; Haghani, F.K. Several iterative methods with memory using self-accelerators. *Appl. Math. Comput.* **2015**, *254*, 452–458. [CrossRef]
12. Herzberger, J. Uber Matrixdarstellungen fur Iterationverfahren bei nichtlinearen Gleichungen. *Computing* **1974**, *12*, 215–222. [CrossRef]
13. Don, E. *Schaum's Outline of Mathematica*; McGraw-Hill Professional: New York, NY, USA, 2000.
14. Petković, M.S.; Neta, B.; Petkovixcx, L.D.; Džunixcx, J. *Multipoint Methods for Solving Nonlinear Equations*; Elsevier: Amsterdam, The Netherlands, 2013.
15. Campos, B.; Cordero, A.; Torregrosa, J.R.; Vindel, P. Stability of King's family of iterative methods with memory. *J. Comput. Appl. Math.* **2017**, *318*, 504–514. [CrossRef]
16. Choubey, N.; Jaiswal, J.P. Two- and three-point with memory methods for solving nonlinear equations. *Numer. Anal. Appl.* **2017**, *10*, 74–89. [CrossRef]
17. Chun, C. Some fourth-order iterative methods for solving nonlinear equations. *Appl. Math. Comput.* **2008**, *195*, 454–459. [CrossRef]
18. Cordero, A.; Lotfi, T.; Khoshandi, A.; Torregrosa, J.R. An efficient Steffensen-like iterative method with memory. *Bull. Math. Soc. Sci. Math. Roum.* **2015**, *58*, 49–58.
19. Cordero, A.; Lotfi, T.; Torregrosa, J.R.; Assari, P.; Mahdiani, K. Some new bi-accelarator two-point methods for solving nonlinear equations. *Comput. Appl. Math.* **2016**, *35*, 251–267. [CrossRef]
20. Jaiswal, J.P. Two efficient bi-parametric derivative free with memory methods for finding simple roots nonlinear equations. *J. Adv. Appl. Math.* **2016**, *1*, 203–210. [CrossRef]
21. Jarratt, P. Some fourth order multipoint iterative methods for solving equations. *Math. Comput.* **1966**, *20*, 434–437. [CrossRef]
22. Maheshwari, A.K. A fourth-order iterative method for solving nonlinear equations. *Appl. Math. Comput.* **2009**, *211*, 383–391. [CrossRef]
23. Kansal, M.; Kanwar, V.; Bhatia, S. Efficient derivative-free variants of Hansen-Patrick's family with memory for solving nonlinear equations. *Numer. Algorithms* **2016**, *73*, 1017–1036. [CrossRef]
24. Lalehchini, M.J.; Lotfi, T.; Mahdiani, K. On developing an adaptive free-derivative Kung and Traub's method with memory. *J. Math. Ext.* **2020**, *14*, 221–241.
25. Zadeh, M.M.; Lotfi, T.; Amirfakhrian, M. Developing two efficient adaptive Newton-type methods with memory. *Math. Methods Appl. Sci.* **2019**, *42*, 5687–5695. [CrossRef]

26. Torkashvand, V.; Kazemi, M. On an efficient family with memory with high order of convergence for solving nonlinear equations. *Int. J. Ind. Math.* **2020**, *12*, 209–224.
27. Traub, J.F. *Iterative Methods for the Solution of Equations*; Prentice Hall: New York, NY, USA, 1964.
28. Wang, X. An Ostrowski-type method with memory using a novel self-accelerating parameter. *J. Comput. Appl. Math.* **2018**, *330*, 710–720. [CrossRef]
29. Zafar, F.; Yasmin, N.; Kutbi, M.A.; Zeshan, M. Construction of tri-parametric derivative free fourth order with and without memory iterative method. *J. Nonlinear Sci. Appl.* **2016**, *9*, 1410–1423. [CrossRef]
30. Wang, X.; Zhu, M. Two iterative methods with memory constructed by the method of inverse interpolation and their dynamics. *Mathematics* **2020**, *8*, 1080. [CrossRef]
31. Zhao, Y.-L.; Zhu, P.-Y.; Gu, X.-M.; Zhao, X.-L.; Jian, H.-Y. A preconditioning technique for all-at-once system from the nonlinear tempered fractional diffusion equation. *J. Sci. Comput.* **2020**, *83*, 10. [CrossRef]
32. Zhao, Y.-L.; Gu, X.-M.; Ostermann, A. A preconditioning technique for an all-at-once system from volterra subdiffusion equations with graded time steps. *J. Sci. Comput.* **2021**, *88*, 11. [CrossRef]
33. Gu, X.M.; Wu, S.L. A parallel-in-time iterative algorithm for Volterra partial integro-differential problems with weakly singular kernel. *J. Comput. Phys.* **2020**, *417*, 109576. [CrossRef]
34. Ernst, P.A.; Soleymani, F. A Legendre-based computational method for solving a class of Itô stochastic delay differential equations. *Numer. Algorithms* **2019**, *80*, 1267–1282. [CrossRef]