mathematics

MDPI

Article

New Memory-Updating Methods in Two-Step Newton's Variants for Solving Nonlinear Equations with High Efficiency Index

Chein-Shan Liu [1] and Chih-Wen Chang [2,*]

[1] Center of Excellence for Ocean Engineering, National Taiwan Ocean University, Keelung 202301, Taiwan; csliu@mail.ntou.edu.tw
[2] Department of Mechanical Engineering, National United University, Miaoli 360302, Taiwan
* Correspondence: cwchang@nuu.edu.tw

Abstract: In the paper, we iteratively solve a scalar nonlinear equation $f(x) = 0$, where $f \in \mathcal{C}(I, \mathbb{R})$, $x \in I \subset \mathbb{R}$, and I includes at least one real root r. Three novel two-step iterative schemes equipped with memory updating methods are developed; they are variants of the fixed-point Newton method. A triple data interpolation is carried out by the two-degree Newton polynomial, which is used to update the values of $f'(r)$ and $f''(r)$. The relaxation factor in the supplementary variable is accelerated by imposing an extra condition on the interpolant. The new memory method (NMM) can raise the efficiency index (E.I.) significantly. We apply the NMM to five existing fourth-order iterative methods, and the computed order of convergence (COC) and E.I. are evaluated by numerical tests. When the relaxation factor acceleration technique is combined with the modified Džunić's memory method, the value of E.I. is much larger than that predicted by the paper [Kung, H.T.; Traub, J.F. *J. Assoc. Comput. Machinery* **1974**, *21*]. for the iterative method without memory.

Keywords: nonlinear equation; two-step iterative schemes; new memory updating method; relaxation factor; supplementary variable

MSC: 65H05; 65B99; 41A25

Citation: Liu, C.-S.; Chang, C.-W. New Memory-Updating Methods in Two-Step Newton's Variants for Solving Nonlinear Equations with High Efficiency Index. *Mathematics* **2024**, *12*, 581. https://doi.org/10.3390/math12040581

Academic Editor: Carlo Bianca

Received: 15 January 2024
Revised: 8 February 2024
Accepted: 13 February 2024
Published: 15 February 2024

1. Introduction

Three novel two-step iterative schemes with memory will be proposed to solve a given scalar nonlinear equation:

$$f(x) = 0, \quad f \in \mathcal{C}(I, \mathbb{R}), \quad x \in I \subset \mathbb{R}, \tag{1}$$

where I is an interval to include the real solution of $f = 0$.

For iteratively solving Equation (1),

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \ n = 0, 1, \ldots \tag{2}$$

is a famous Newton method (NM). Obviously, for the Newton method, $f(x)$ is required to be differentiable, even though the NM is still a popular iterative method to solve Equation (1), owing to its simplicity.

From Equation (2), we can define the following Newton iteration function:

$$\mathcal{N}(x) = x - \frac{f(x)}{f'(x)}. \tag{3}$$

It follows that

$$\mathcal{N}'(x) = \frac{f(x)f''(x)}{f'(x)^2}. \tag{4}$$

The critical point of the mapping $\mathcal{N}(x)$ has two sources: one is the root of $f(x) = 0$ and another is the zero point of $f''(x) = 0$, which causes $\mathcal{N}'(x) = 0$. When the iteration tends to the zero point of $f''(x) = 0$, the NM no longer converges to the real solution of $f(x) = 0$. For the function $f(x) = x^3 - 2x + 2 = 0$, as an example, we have

$$\mathcal{N}(x) = x - \frac{f(x)}{f'(x)} = \frac{2x^3 - 2}{3x^2 - 2}. \tag{5}$$

It follows that

$$\mathcal{N}(0) = 1,\ \mathcal{N}(1) = 0. \tag{6}$$

Because of $f''(0) = 0$ and $\mathcal{N}(1) = 0$, if the initial point x_0 is located near 0 and 1, the NM does not converge to the true solution $r = -1.769292354238631$ of $x^3 - 2x + 2 = 0$.

In summary, the NM possesses some drawbacks, such as sensitiveness to the initial guess, dividing by a nearly zero value in the denominator, and nonconvergence near to the critical values which are not the roots of $f(x) = 0$. In order to overcome these difficulties, we propose the following perturbation of the Newton method. Mathematically, Equation (1) can be written as

$$xf'(x) - \alpha xf(x) = xf'(x) - (1 + \alpha x)f(x), \tag{7}$$

where $\alpha = f''(r)/(2f'(r))$ is determined in [1]. By canceling $xf'(x)$ and $\alpha xf(x)$ on both sides of

$$xf'(x) - \alpha xf(x) = xf'(x) - \alpha xf(x) - f(x), \tag{8}$$

we can achieve

$$-f(x) = 0,$$

which is equivalent to Equation (1).

From Equation (8):

$$[f'(x_n) - \alpha f(x_n)]x_{n+1} = x_n f'(x_n) - \alpha x_n f(x_n) - f(x_n), \tag{9}$$

which upon dividing both sides by $f'(x_n) - \alpha f(x_n)$ preceding x_{n+1}, yields

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n) - \alpha f(x_n)}. \tag{10}$$

The iterative scheme (10) was developed to be a one-step continuation Newton-like method [2], and it was used as the first step in the multistep iterative schemes in [3–5]. Some dynamical analysis of Equation (10) can be seen in [6,7]. When $f'(x_n) = 0$, Equation (10) is still applicable, but Equation (2) is a failure. As pointed out by Wu [8], Equation (10) has some merits over the NM.

The following second-order boundary value problem (BVP) demonstrates the usefulness of Equation (1):

$$u''(y) - 3u'(y) + 2u(y) = 0,\ y \in (0,1), \tag{11}$$

$$u(0) = 1,\ \ u(1) = 0. \tag{12}$$

Upon letting

$$u(y) = v(y) - (x + 1)y + 1, \tag{13}$$

and using $v(0) = 0$ and $v(1) = x$ to render $u(0) = 1$ and $u(1) = 0$ automatically, we can transform Equations (11) and (12) to an initial value problem of the following ordinary differential equation (ODE):

$$v''(y) - 3v'(y) + 2v(y) - 2(x + 1)y + 3(x + 1) + 2 = 0, \tag{14}$$

$$v(0) = 0,\ \ v'(0) = 0; \tag{15}$$

the solution is endowed with an unknown value x, given by

$$v(y) = (3+x)e^y - (x+2)e^{2y} + (x+1)y - 1. \tag{16}$$

If a real value exists for x, then we have a real solution for $v(y)$. By imposing $v(1) = x$,

$$(3+x)e - (x+2)e^2 + x + 1 - 1 = x \tag{17}$$

results to the equation $f(x) = (3+x)e - (x+2)e^2 = 0$ for determining x; then, by Equations (16) and (13), the exact solution $u(y)$ can be obtained.

Sometimes $f(x)$ is obtained from a nonlinear ODE, rather than the linear ODE in Equation (11). We further consider a nonlinear BVP:

$$u''(y) = \frac{3}{2}u^2(y),\ y \in (0,1), \tag{18}$$

$$u(0) = 4,\ \ u(1) = 1. \tag{19}$$

One assumes an initial value $u'(0) = x$ with x being unknown and integrates Equation (18) with the initial conditions $u(0) = 4$ and $u'(0) = x$. The nonlinear equation $f(x) = u(1,x) - 1 = 0$ for satisfying the right-end boundary condition in Equation (19) is derived. Since $u(1,x)$ is an implicit function of x, the function $f(x)$ cannot be written out explicitly. In this nonlinear problem, when we apply the NM to solve $f(x) = u(1,x) - 1 = 0$, we encounter a difficulty to calculate $f'(x)$. Recently, Liu et al. [9] proposed a single-step memory-dependent method to solve $f(x) = 0$ by a Lie-symmetry formulation of Equation (18).

Consider the following one [10]:

$$\dot{\mathbf{Q}}(t) = k_e\dot{\mathbf{q}}(t) - \frac{k_e x}{Q_0}\mathbf{Q}(t), \tag{20}$$

$$\sqrt{(Q_0 - \|\mathbf{Q}(t)\|)^2 + x^2} - (Q_0 - \|\mathbf{Q}(t)\|) - x = 0, \tag{21}$$

which simulates the time-varying relation between stress $\mathbf{Q}$ and strain $\mathbf{q}$ of an elastic–perfectly plastic material. In the above, k_e is the elastic modulus and Q_0 is the yield stress of material. We need to solve the nonlinear scalar equation $f(x) = \sqrt{(Q_0 - \|\mathbf{Q}\|)^2 + x^2} - (Q_0 - \|\mathbf{Q}\|) - x = 0$ to determine x, but $\mathbf{Q}$ is governed by a system of first-order ODEs being coupled to x in Equation (20). The difficulty is exhibited by using the NM to solve Equation (21), where the transition from elastic phase $x = 0$ to plastic phase $x > 0$ is not smooth.

Many engineering problems and adapted mathematical methods have been proposed for solving nonlinear equations, e.g., a weighted density functional theory for an inhomogeneous 12-6 Lennard–Jones fluid and the Euler–Lagrange equation derived from the density functional theory of inhomogeneous fluids [11], the governing mathematical equations defining the physical features of the first-grade viscoelastic nanofluid flow and heat transfer models [12], and a specialized nonlinear Fredholm integral equation in the turbo-reactors industry [13].

If one attempts to obtain an approximate analytical solution of the nonlinear BVP, the functional iteration method may be a useful tool. A conventional functional iteration method is the Picard iteration method; however, it has a major disadvantage because of its slow convergence. In order to improve the convergence property, He [14,15] proposed the variational iteration method, which is a modification of the Picard iteration method for the second-order nonlinear initial value problem. Recently, Wang et al. [16] developed an accurate predictor–corrector Picard iteration method for solving nonlinear problems. By the Newton–Kurchatov method for solving nonlinear equations, Argyros et al. [17] addressed a semilocal analysis and derived the weaker sufficient semilocal convergence criteria, and Argyros and Shakhno [18] employed a local convergence in the Banach space valued equations.

The drawbacks and limitations of the NM as just mentioned above induced a lot of studies that continue to the present and render some modifications of the Newton method, like the Adomian decomposition method [19], decomposition method [20], the arithmetic mean quadrature method [21], contra-harmonic mean and quadrature method [22], power-means variants [23], modified homotopy perturbation method [24], generating function method [25], perturbation of Newton's method [26], and the variants of Bawazir's iterative methods [27].

To improve the low-order convergence of the one-step iterative scheme and to enhance the convergence order, many multistep iterative schemes were developed. One can refer to [28,29] for many discussions of the multistep iterative methods. According to the conjecture of Kung and Traub [30], the upper bound of the efficiency index (E.I.) for the optimal iterative scheme with m evaluations of functions is E.I. = $2^{1-1/m} < 2$. For $m = 2$, the NM is an optimal iterative scheme with E.I. = 1.414. With $m = 3$, the Halley method is not the optimal one with a low-value E.I. = 1.44225. The conjecture of Kung and Traub [30] is only applicable to the integer-order convergence scheme. The computed order of convergence (COC) proposed in [21] can be adopted to evaluate the convergence order of the iterative scheme. In this paper, we extend the Newton method by the idea of perturbations, which involve some optimal parameters determined by the convergence analysis. The COC of these two-step iterative schemes is larger than $p = 2^{m-1}$.

Liu et al. [31] verified that the following iterative scheme:

$$x_{n+1} = x_n - \frac{f(x_n)}{c + df(x_n)} \tag{22}$$

is of third-order convergence, if $c = f'(r)$ and $d = f''(r)/(2f'(r))$. By using the accelerating parameters,we can speed up the convergence. A more detailed analysis of the iterative scheme (22) can be seen in [1]. Equation (22) was addressed by Liu [32] from a two-dimensional approach together with the splitting technique. If we take $c = f'(r)$ and $d = 0$, Equation (22) is known to be a fixed-point Newton method:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(r)}. \tag{23}$$

Traub [33] developed a simple accelerating method by giving x_0 and γ_0:

$$\begin{cases} w_n = x_n + \gamma_n f(x_n), \\ x_{n+1} = x_n - \frac{f(x_n)}{f[x_n, w_n]}, \\ \gamma_{n+1} = -\frac{1}{f[x_n, x_{n+1}]}. \end{cases} \tag{24}$$

By incorporating the memory of x_n into account, the order of convergence can be raised. Traub's technique is a typical method with memory, in which the data that appeared in the previous iteration were adopted in the iteration. For recent progress of the memory method with accelerating parameters in the multistep iterative methods, one can refer to [5,27,34–40]. One major goal of this paper is to develop the two-step iterative schemes with a new memory method to determine the accelerating parameters by updating technique with the information at the current step.

2. Three New Two-Step Iterative Schemes

In 2010, Wang and Liu [41] proposed

$$\begin{cases} y_n = x_n - \frac{f(x_n)}{f'(x_n) - \alpha f(x_n)}, \\ x_{n+1} = y_n - \frac{f(y_n)}{f'(x_n) - \alpha f(x_n)}, \end{cases} \tag{25}$$

which is a two-step iterative scheme based on Equation (10). The error equation is

$$e_{n+1} = (\alpha^2 - 3a_2\alpha + 2a_2^2)e_n^3 + \mathcal{O}(e_n^4), \tag{26}$$

where $a_2 := f''(r)/(2f'(r))$ and $e_n = x_n - r$. Equation (25) is a two-step iterative scheme because it involves two variables, x_n and y_n, and two steps for computing x_{n+1}; the first step is computing y_n, and then x_n and y_n are inserted into the second step to compute x_{n+1}.

Wang and Zhang [42] developed a family of Newton-type two-step iterative schemes with a memory method for solving nonlinear equations whose R-convergence order is increased from 4 to 4.5616, 4.7913, and 5 depending on whether updating techniques are used in the accelerating parameters. Nowadays, most memory-accelerating methods do not take the differential term $f'(x)$ into the iterative schemes.

2.1. First New Two-Step Iterative Scheme

Instead of Equation (25), we consider an extension of Equation (23) to a two-step iterative scheme:

$$\begin{cases} y_n = x_n - \frac{f(x_n)}{f'(r) - \beta f(x_n)}, \\ x_{n+1} = y_n - \frac{f(y_n)}{f'(r) - \beta f(x_n)}, \end{cases} \tag{27}$$

where β is a parameter whose optimal value is to be determined. The first step is a variant of the so-called fixed-point Newton method $y_n = x_n - f(x_n)/f'(r)$.

Theorem 1. *The function $f : I \subset \mathbb{R} \mapsto \mathbb{R}$ is sufficiently differentiable on the domain I, and $r \in I$ is a simple root with $f(r) = 0$ and $f'(r) \neq 0$. If x_0 is sufficiently close to r within the radius of convergence, then the iterative scheme (27) for solving $f(x) = 0$ has fourth-order convergence:*

$$e_{n+1} = (a_2^3 - a_2a_3)e_n^4 + \mathcal{O}(e_n^5), \tag{28}$$

where the optimal value of β is given by

$$\beta = -a_2 = -\frac{f''(r)}{2f'(r)}. \tag{29}$$

Furthermore, the error of y_n reads as

$$e_{n,y} = y_n - r = (a_2^2 - a_3)e_n^3 + \mathcal{O}(e_n^4), \tag{30}$$

where $a_3 = f'''(r)/(6f'(r))$.

Proof. According to the assumption,

$$e_n = x_n - r \tag{31}$$

is a small quantity; by $e_{n+1} = x_{n+1} - r$ and Equation (31):

$$e_{n+1} = e_n + x_{n+1} - x_n. \tag{32}$$

Define

$$a_n := \frac{f^{(n)}(r)}{n!f'(r)},\ n = 2, \ldots; \tag{33}$$

as usual,

$$f(x_n) = f'(r)[e_n + a_2e_n^2 + a_3e_n^3 + a_4e_n^4 + \cdots]. \tag{34}$$

A straightforward computation renders

$$\beta f(x_n) - f'(r) = f'(r)[-1 + \beta e_n + \beta a_2 e_n^2 + \beta a_3 e_n^3 + \beta a_4 e_n^4], \tag{35}$$

$$d_n = y_n - x_n = \frac{f(x_n)}{\beta f(x_n) - f'(r)} = -e_n + b_2 e_n^2 + b_3 e_n^3 + b_4 e_n^4, \tag{36}$$

where

$$b_2 = -(\beta + a_2),\ b_3 = -(2\beta a_2 + \beta^2 + a_3),\ b_4 = -(\beta a_2^2 + 3\beta^2 a_2 + 2\beta a_3 + a_4). \tag{37}$$

It follows from Equations (31) and (36) that

$$e_{n,y} = y_n - r = y_n - x_n + x_n - r = d_n + e_n = b_2 e_n^2 + b_3 e_n^3 + b_4 e_n^4. \tag{38}$$

In terms of $e_{n,y}$, we can express $f(y_n)$ by

$$\begin{aligned} f(y_n) &= f'(r)[e_{n,y} + a_2 e_{n,y}^2 + \cdots] \\ &= f'(r)[b_2 e_n^2 + b_3 e_n^3 + b_4 e_n^4 + a_2(b_2 e_n^2 + b_3 e_n^3 + b_4 e_n^4)^2 + \cdots] \\ &= f'(r)[b_2 e_n^2 + b_3 e_n^3 + (b_4 + a_2 b_2^2) e_n^4 + \cdots]. \end{aligned} \tag{39}$$

By using the second one in Equation (27), subtracting both sides by r and from Equations (34), (38) and (39), we can derive

$$\begin{aligned} e_{n+1} &= e_{n,y} - \tfrac{f(y_n)}{f'(r) - \beta f(x_n)} \\ &= b_2 e_n^2 + b_3 e_n^3 + b_4 e_n^4 - \tfrac{b_2 e_n^2 + b_3 e_n^3 + (b_4 + a_2 b_2^2) e_n^4}{1 - \beta e_n - \beta a_2 e_n^2 - \beta a_3 e_n^3 - \beta a_4 e_n^4} \\ &= b_2 e_n^2 + b_3 e_n^3 + b_4 e_n^4 - [b_2 e_n^2 + b_3 e_n^3 + (b_4 + a_2 b_2^2) e_n^4] \\ &\times [1 + \beta e_n + \beta a_2 e_n^2 + (\beta e_n + \beta a_2 e_n^2)^2] \\ &= -\beta b_2 e_n^3 - (\beta b_3 + a_2 b_2^2 + \beta a_2 b_2 + \beta^2 b_2) e_n^4 + \mathcal{O}(e_n^5) \\ &= \beta(\beta + a_2) e_n^3 - (\beta b_3 + a_2 b_2^2 + \beta a_2 b_2 + \beta^2 b_2) e_n^4 + \mathcal{O}(e_n^5). \end{aligned} \tag{40}$$

If we take $\beta = -a_2$, then $b_2 = 0$, $b_3 = a_2^2 - a_3$, and then Equation (28) is derived.

If $\beta = -a_2$ is taken, Equation (38) reduces to

$$e_{n,y} = b_3 e_n^3 + b_4 e_n^4 = (a_2^2 - a_3) e_n^3 + (2a_2 a_3 - 2a_2^3 - a_4) e_n^4. \tag{41}$$

The proof of Theorem 1 is completed. □

Notice that the error Equation (28) is the same as the Ostrowski's two-step fourth-order optimal iterative scheme [43]. Since Equation (27) is different from Equation (25), the error Equation (28) is different from that in Equation (26); if $\alpha = a_2$, the iterative scheme (25) also has fourth-order convergence.

2.2. Second New Two-Step Iterative Scheme

Let

$$\begin{cases} y_n = x_n - \frac{f(x_n)}{f'(r) - \beta f(x_n)}, \\ x_{n+1} = y_n - \frac{f(y_n)}{f'(r) - \alpha f(x_n)}, \end{cases} \tag{42}$$

where β and α are two parameters whose optimal values are to be determined.

Theorem 2. *The function $f : I \subset \mathbb{R} \mapsto \mathbb{R}$ is sufficiently differentiable on the domain I, and $r \in I$ is a simple root with $f(r) = 0$ and $f'(r) \neq 0$. If x_0 is sufficiently close to r within the radius of convergence, then the iterative scheme (42) with $\alpha = 0$ for solving $f(x) = 0$ has fourth-order convergence:*

$$e_{n+1} = -a_2(\beta + a_2)^2 e_n^4 + \mathcal{O}(e_n^5). \tag{43}$$

If the optimal value of β is given by

$$\beta = -a_2 = -\frac{f''(r)}{2f'(r)}, \tag{44}$$

then the iterative scheme (42) with $\alpha = 0$ has fifth-order convergence.

Proof. By using the second one in Equation (42), subtracting both sides by r and from Equations (34), (38) and (39), we can derive

$$\begin{aligned} e_{n+1} &= e_{n,y} - \tfrac{f(y_n)}{f'(r)-\alpha f(x_n)} \\ &= b_2 e_n^2 + b_3 e_n^3 + b_4 e_n^4 - \tfrac{b_2 e_n^2 + b_3 e_n^3 + (b_4 + a_2 b_2^2) e_n^4}{1 - \alpha e_n - \alpha a_2 e_n^2 - \alpha a_3 e_n^3 - \alpha a_4 e_n^4} \\ &= b_2 e_n^2 + b_3 e_n^3 + b_4 e_n^4 - [b_2 e_n^2 + b_3 e_n^3 + (b_4 + a_2 b_2^2) e_n^4] \\ &\times [1 + \alpha e_n + \alpha a_2 e_n^2 + (\alpha e_n + \alpha a_2 e_n^2)^2] \\ &= -\alpha b_2 e_n^3 - (\alpha b_3 + a_2 b_2^2 + \alpha a_2 b_2 + \alpha^2 b_2) e_n^4 + \mathcal{O}(e_n^5) \\ &= \alpha(\beta + a_2) e_n^3 - (\alpha b_3 + a_2 b_2^2 + \alpha a_2 b_2 + \alpha^2 b_2) e_n^4 + \mathcal{O}(e_n^5). \end{aligned} \tag{45}$$

If we take $\alpha = 0$, then Equation (43) is derived after taking $b_2 = -(\beta + a_2)$. □

It is interesting that the iterative scheme (42) with $\alpha = 0$ is simpler than the iterative scheme (27), but its order of convergence is better.

2.3. Third New Two-Step Iterative Scheme

We further consider

$$\begin{cases} y_n = x_n - \frac{f(x_n)}{f'(r)-\beta f(x_n)}, \\ x_{n+1} = y_n - \frac{f(y_n)}{f'(r)-\beta f(y_n)}. \end{cases} \tag{46}$$

The second-step is enhanced by using $f'(r) - \beta f(y_n)$, rather than $f'(r) - \beta f(x_n)$ in Equation (27). Equation (46) is a two-step iterative scheme because it involves two variables, x_n and y_n, and two steps for computing x_{n+1}; the first step is computing y_n, and then y_n is inserted into the second step to compute x_{n+1}.

Theorem 3. *The function $f : I \subset \mathbb{R} \mapsto \mathbb{R}$ is sufficiently differentiable on the domain I, and $r \in I$ is a simple root with $f(r) = 0$ and $f'(r) \neq 0$. If x_0 is sufficiently close to r within the radius of convergence, then the iterative scheme (46) for solving $f(x) = 0$ is of the fourth-order convergence:*

$$e_{n+1} = -(\beta + a_2) b_2^2 e_n^4 + \mathcal{O}(e_n^5) = -(\beta + a_2)^3 e_n^4 + \mathcal{O}(e_n^5). \tag{47}$$

If $\beta = -a_2$, Equation (47) reduces to $e_{n+1} = \mathcal{O}(e_n^5)$. That is, the iterative scheme (46) is of the fifth-order convergence.

Proof. By using the second one in Equation (46), subtracting both sides by r and from Equations (38) and (39), we can derive

$$\begin{aligned} e_{n+1} &= e_{n,y} - \frac{f(y_n)}{f'(r)-\beta f(y_n)} \\ &= b_2e_n^2 + b_3e_n^3 + b_4e_n^4 - \frac{b_2e_n^2+b_3e_n^3+(b_4+a_2b_2^2)e_n^4}{1-\beta b_2e_n^2-\beta b_3e_n^3-\beta(b_4+a_2b_2^2)e_n^4} \\ &= b_2e_n^2 + b_3e_n^3 + b_4e_n^4 - [b_2e_n^2 + b_3e_n^3 + (b_4 + a_2b_2^2)e_n^4][1 + \beta b_2e_n^2 + \beta b_3e_n^3] \\ &= -(\beta + a_2)b_2^2e_n^4 + \mathcal{O}(e_n^5). \end{aligned} \tag{48}$$

The proof of Theorem 3 is completed. □

3. Four New Memory Methods

3.1. The First and Second New Memory Methods

The error Equation (40) is simpler than that in Equation (26). Theorem 1 indicates that the optimal value of β is $\beta = -f''(r)/(2f'(r))$. However, because the root r is itself an unknown value, $f'(r)$ and $f''(r)$ are not available; they are critical parameters to enhance the performance of the proposed two-step iterative schemes.

The memory-dependent techniques to obtain the suitable parameters' values were found in [34,44–48]. Let $A = f'(r)$ and $B = -\beta = f''(r)/(2f'(r))$. We develop a new memory method for updating the values of A and B with the current values. In Equation (27), there are only two current values, x_n and y_n, which are insufficient to update $f''(r)$. Therefore, we introduce a supplementary variable obtained by the fixed point Newton method:

$$w_n = x_n - \frac{f(x_n)}{A} = x_n - \frac{f(x_n)}{f'(r)}. \tag{49}$$

Then, with the three data (x_n, w_n, y_n), a second-degree Newton polynomial that is an interpolant is given by

$$\mathcal{N}_2(x) = f(x_n) + f[x_n, w_n](x - x_n) + f[x_n, w_n, y_n](x - x_n)(x - w_n), \tag{50}$$

where

$$f[x_n, w_n] = \frac{f(x_n) - f(w_n)}{x_n - w_n}, \ \ f[x_n, w_n, y_n] = \frac{f[x_n, w_n] - f[w_n, y_n]}{x_n - y_n}. \tag{51}$$

It is easy to derive $\mathcal{N}_2(y_n) = f(y_n)$, and

$$\mathcal{N}_2'(x_{n+1}) = f[x_n, w_n] + f[x_n, w_n, y_n](2x_{n+1} - x_n - w_n), \ \ \mathcal{N}_2''(x_{n+1}) = 2f[x_n, w_n, y_n]. \tag{52}$$

In general, $\mathcal{N}_2(x_{n+1}) \neq f(x_{n+1})$.

The new algorithm based on Theorem 1, namely the first new memory-updating method (FNMUM), is depicted by (i) giving x_0, a_0, b_0, and $c_0 = (a_0 + b_0)/2$ and computing A_0 and B_0 by

$$A_0 = f[a_0, b_0], \ B_0 = \frac{f(b_0) - 2f(c_0) + f(a_0)}{2A_0(c_0 - a_0)^2}, \tag{53}$$

(ii) for $n = 0, 1, \ldots$, perform the following computations until convergence:

$$w_n = x_n - \frac{f(x_n)}{A_n}, \tag{54}$$

$$y_n = x_n - \frac{f(x_n)}{A_n + B_nf(x_n)}, \tag{55}$$

$$x_{n+1} = y_n - \frac{f(y_n)}{A_n + B_nf(x_n)}, \tag{56}$$

$$A_{n+1} = f[x_n, w_n] + f[x_n, w_n, y_n](2x_{n+1} - x_n - w_n), \tag{57}$$

$$B_{n+1} = \frac{f[x_n, w_n, y_n]}{A_{n+1}}. \tag{58}$$

There exist three evaluations of functions for $f(x_n)$, $f(w_n)$, and $f(y_n)$ such that the optimal order of convergence is $p = 2^2 = 4$, and E.I. = 1.5874. The role of w_n, which does not engage in the iteration, is different from x_n and y_n; x_n and y_n are step variables used in the iteration in Equations (55) and (56), and w_n is computed from Equation (54) to provide an extra datum used in Equations (57) and (58) to update the values of A_{n+1} and B_{n+1}.

Therefore, the present parameters' updating technique is different from the memory-dependent accelerating techniques in [34,44–48]. In the FNMUM, no previous iteration values of w_{n-1} and y_{n-1} were used in addition to the initial values a_0 and b_0. Therefore, the new memory method can save much more computational cost than the previous memory-accelerating technique.

The second new memory-updating method (SNMUM) based on Theorem 2 is depicted by (i) giving x_0, a_0, b_0, and $c_0 = (a_0 + b_0)/2$ and computing A_0 and B_0 by

$$A_0 = f[a_0, b_0],\ B_0 = \frac{f(b_0) - 2f(c_0) + f(a_0)}{2A_0(c_0 - a_0)^2}, \tag{59}$$

(ii) for $n = 0, 1, \ldots$, perform the following computations until convergence:

$$w_n = x_n - \frac{f(x_n)}{A_n}, \tag{60}$$

$$y_n = x_n - \frac{f(x_n)}{A_n + B_n f(x_n)}, \tag{61}$$

$$x_{n+1} = y_n - \frac{f(y_n)}{A_n}, \tag{62}$$

$$A_{n+1} = f[x_n, w_n] + f[x_n, w_n, y_n](2x_{n+1} - x_n - w_n), \tag{63}$$

$$B_{n+1} = \frac{f[x_n, w_n, y_n]}{A_{n+1}}. \tag{64}$$

In the SNMUM, only the initial values a_0 and b_0 are guessed, and no previous iteration values of w_{n-1} and y_{n-1} were used, which renders it more computationally cost-effective than the previous memory-accelerating technique.

3.2. The Third New Memory Method

According to Theorem 3, the third new memory-updating method (TNMUM), is depicted by (i) giving x_0, a_0, b_0, and $c_0 = (a_0 + b_0)/2$ and computing A_0 and B_0 by

$$A_0 = f[a_0, b_0],\ B_0 = \frac{f(b_0) - 2f(c_0) + f(a_0)}{2A_0(c_0 - a_0)^2}, \tag{65}$$

(ii) for $n = 0, 1, \ldots$, perform the following computations until convergence:

$$w_n = x_n - \frac{f(x_n)}{A_n}, \tag{66}$$

$$y_n = x_n - \frac{f(x_n)}{A_n - B_n f(x_n)}, \tag{67}$$

$$x_{n+1} = y_n - \frac{f(y_n)}{A_n - B_n f(y_n)}, \tag{68}$$

$$A_{n+1} = f[x_n, w_n] + f[x_n, w_n, y_n](2x_{n+1} - x_n - w_n), \tag{69}$$

$$B_{n+1} = \frac{f[x_n, w_n, y_n]}{A_{n+1}}. \tag{70}$$

Similarly, in the TNMUM, only the initial values a_0 and b_0 are guessed, and no previous iteration values of w_{n-1} and y_{n-1} are used; it is quite computationally cost-effective.

3.3. The Accelerated Third New Memory Method

In the previous three new memory methods, the datum of $f(x_{n+1})$ at x_{n+1} was not used. We modify the supplementary variable by

$$w_n = x_n - \eta \frac{f(x_n)}{f'(r)}, \tag{71}$$

where η is a relaxation factor to be designed.

Then, we set

$$\mathcal{N}_2(x_{n+1}) = f(x_n) + f[x_n, w_n](x_{n+1} - x_n) + f[x_n, w_n, y_n](x_{n+1} - x_n)(x_{n+1} - w_n) = f(x_{n+1}), \tag{72}$$

where $\mathcal{N}_2(x)$ was given by Equation (50). Inserting

$$x_n - w_n = \eta \frac{f(x_n)}{f'(r)} \tag{73}$$

into Equation (72), we can derive

$$\begin{aligned} &f(x_n) + \tfrac{f'(r)[f(x_n)-f(w_n)]}{\eta f(x_n)}(x_{n+1} - x_n) \\ &+ \tfrac{1}{x_n - y_n}\left[\tfrac{f'(r)[f(x_n)-f(w_n)]}{\eta f(x_n)} - \tfrac{f(w_n)-f(y_n)}{x_n - y_n}\right] = f(x_{n+1}). \end{aligned} \tag{74}$$

Through some manipulations we can derive

$$\eta = \frac{f'(r)[f(x_n) - f(w_n)][(x_{n+1} - x_n)(x_n - y_n) + 1]}{f(x_n)\{[f(x_{n+1}) - f(x_n)](x_n - y_n) + f[w_n, y_n]\}}, \tag{75}$$

which can be used to update η.

The new algorithm based on Theorem 3 and Equations (71) and (75), namely the accelerated third new memory-updating method (ATNMUM), is depicted by (i) giving x_0, a_0, b_0, $c_0 = (a_0 + b_0)/2$ and η_0 and computing A_0 and B_0 by

$$A_0 = f[a_0, b_0],\ B_0 = \frac{f(b_0) - 2f(c_0) + f(a_0)}{2A_0(c_0 - a_0)^2}, \tag{76}$$

(ii) for $n = 0, 1, \ldots$, perform the following computations until convergence:

$$w_n = x_n - \eta_n \frac{f(x_n)}{A_n}, \tag{77}$$

$$y_n = x_n - \frac{f(x_n)}{A_n + B_n f(x_n)}, \tag{78}$$

$$x_{n+1} = y_n - \frac{f(y_n)}{A_n + B_n f(y_n)}, \tag{79}$$

$$A_{n+1} = f[x_n, w_n] + f[x_n, w_n, y_n](2x_{n+1} - x_n - w_n), \tag{80}$$

$$B_{n+1} = \frac{f[x_n, w_n, y_n]}{A_{n+1}}, \tag{81}$$

$$\eta_{n+1} = \frac{A_{n+1}[f(x_n) - f(w_n)][(x_{n+1} - x_n)(x_n - y_n) + 1]}{f(x_n)\{[f(x_{n+1}) - f(x_n)](x_n - y_n) + f[w_n, y_n]\}}. \tag{82}$$

In Equation (46), we take the free parameter $\beta = -B_n$.

In the ATNMUM, the initial values a_0, b_0, and η_0 are guessed, and no previous iteration values of w_{n-1} and y_{n-1} are used; it is quite computationally cost-effective.

In the above four iterative algorithms, FNMUM, SNMUM, TNMUM, and ATNMUM, the function $f(x)$ with $f \in \mathcal{C}(I, \mathbb{R})$ is sufficient. We suppose that the interval I includes at least one real root. In general, the nonlinear equation $f(x) = 0$ may possess many roots, which can be determined by the iterative schemes by giving different initial guesses of x_0.

4. A Simple Memory Approach to Existent Iterative Schemes

4.1. The WMM Method

Wang [36] modified the Ostrowski-type method with memory for self-accelerating a parameter λ, given by

$$\begin{cases} w_n = x_n - \frac{f(x_n)}{f'(x_n)}, \\ y_n = w_n - \lambda(w_n - x_n)^2, \\ x_{n+1} = y_n - \frac{f(y_n)}{2f[x_n, y_n] - f'(x_n)}, \end{cases} \tag{83}$$

whose error equation was proven to be

$$e_{n+1} = (a_2 - \lambda)(a_2^2 - a_3 - a_2\lambda)e_n^4 + \mathcal{O}(e_n^5). \tag{84}$$

Substituting the first one into the second one in Equation (83), it is indeed a two-step method:

$$\begin{cases} y_n = x_n - \frac{f(x_n)}{f'(x_n)} - \lambda \frac{f^2(x_n)}{f'(x_n)^2}, \\ x_{n+1} = y_n - \frac{f(y_n)}{2f[x_n, y_n] - f'(x_n)}. \end{cases} \tag{85}$$

Upon taking $B = \lambda = a_2$ to be an updating parameter, the order can be raised by the Wang memory method (WMM), which is depicted by (i) giving x_0, a_0, b_0, and $c_0 = (a_0 + b_0)/2$ and computing B_0 by

$$B_0 = \frac{f(b_0) - 2f(c_0) + f(a_0)}{2f[a_0, b_0](c_0 - a_0)^2}, \tag{86}$$

(ii) perform for $n = 0, 1, \ldots$,

$$w_n = x_n - \frac{f(x_n)}{f'(x_n)}, \tag{87}$$

$$y_n = w_n - B_n(w_n - x_n)^2, \tag{88}$$

$$x_{n+1} = y_n - \frac{f(y_n)}{2f[x_n, y_n] - f'(x_n)}, \tag{89}$$

$$B_{n+1} = \frac{f[x_n, w_n, y_n]}{f[x_n, w_n] + f[x_n, w_n, y_n](2x_{n+1} - x_n - w_n)}. \tag{90}$$

There exist four evaluations of functions for $f(x_n)$, $f(w_n)$, $f(y_n)$, and $f'(x_n)$.

4.2. The ZLHMM Method

Zheng et al. [49] modified the Steffensen-type method with an accelerating parameter γ:

$$\begin{cases} w_n = x_n + \gamma f(x_n), \\ y_n = x_n - \frac{f(x_n)}{f[x_n, w_n]}, \\ x_{n+1} = y_n - \frac{f(y_n)}{f[x_n, y_n] + (y_n - x_n) f[x_n, w_n, y_n]}, \end{cases} \tag{91}$$

whose error equation is

$$e_{n+1} = (1 + f'(r)\gamma)^2 a_2(a_2^2 - a_3)e_n^4 + \mathcal{O}(e_n^5). \tag{92}$$

Let $A = f'(r)$ and $\gamma = -1/A$. The ZLH memory method (ZLHMM) reads as (i) giving x_0, a_0, b_0 and computing A_0 by

$$A_0 = f[a_0, b_0], \tag{93}$$

(ii) for $n = 0, 1, \ldots$,

$$w_n = x_n - \frac{f(x_n)}{A_n}, \tag{94}$$

$$y_n = x_n - \frac{f(x_n)}{f[x_n, w_n]}, \tag{95}$$

$$x_{n+1} = y_n - \frac{f(y_n)}{f[x_n, y_n] + (y_n - x_n) f[x_n, w_n, y_n]}, \tag{96}$$

$$A_{n+1} = f[x_n, w_n] + f[x_n, w_n, y_n](2x_{n+1} - x_n - w_n). \tag{97}$$

There exist three evaluations of functions for $f(x_n)$, $f(w_n)$, and $f(y_n)$.

4.3. The CCTMM Method

Chicharro et al. [37] proposed a two-step iterative scheme:

$$\begin{cases} y_n = x_n - \frac{f(x_n)}{f[x_n, w_n]}, \\ x_{n+1} = y_n - H(t_n) \frac{f(y_n)}{f[y_n, w_n]}, \end{cases} \tag{98}$$

where $w_n = x_n + \gamma f(x_n)$, $t_n = f(y_n)/f(x_n)$, and $H(0) = H'(0) = 1$, and $H''(0) < \infty$. They derived the following error equation:

$$e_{n+1} = -\frac{a_2}{2}[1 + \gamma f'(r)]^2[\{-6 + \gamma f'(r)(H''(0) - 2) + H''(0)\}a_2^2 + 2a_3]e_n^4 + \mathcal{O}(e_n^5). \tag{99}$$

If we take $\gamma = -1/f'(r)$, the convergence at least increases by one order.

Let $A = f'(r)$. The CCT memory method (CCTMM) reads as (i) giving x_0, a_0, b_0 and computing A_0 by Equation (93), and (ii) for $n = 0, 1, \ldots$,

$$w_n = x_n - \frac{f(x_n)}{A_n}, \tag{100}$$

$$y_n = x_n - \frac{f(x_n)}{f[x_n, w_n]}, \tag{101}$$

$$t_n = \frac{f(y_n)}{f(x_n)}, \tag{102}$$

$$x_{n+1} = y_n - H(t_n) \frac{f(y_n)}{f[y_n, w_n]}, \tag{103}$$

$$A_{n+1} = f[x_n, w_n] + f[x_n, w_n, y_n](2x_{n+1} - x_n - w_n). \tag{104}$$

In the numerical test, we take $H(t) = 1 + t$. Three evaluations of functions for $f(x_n)$, $f(w_n)$, and $f(y_n)$ are needed.

4.4. The DMM Method

In 2013, Džunić [34] proposed a two-step iterative scheme:

$$\begin{cases} y_n = x_n - \frac{f(x_n)}{f[x_n, w_n] + p f(w_n)}, \\ x_{n+1} = y_n - g(t_n) \frac{f(y_n)}{f[y_n, w_n] + p f(w_n)}, \end{cases} \tag{105}$$

where $w_n = x_n + \gamma f(x_n)$, $t_n = f(y_n)/f(x_n)$, and $g(0) = g'(0) = 1$, and $|g''(0)| < \infty$. Džunić [34] verified that the convergence order is at least seven when the parameters

γ and p are accelerated by the memory-dependent technique using the third-order and fourth-order Newton polynomials.

Here, we employ the second-order $\mathcal{N}_2(x)$ in Equation (50) to update γ and p. Let $A = f'(r) = -1/\gamma$ and $B = -p = a_2$. The new Džunić memory method (DMM) reads as (i) giving x_0, a_0, b_0, $c_0 = (a_0 + b_0)/2$ and computing A_0 and B_0 by Equation (53), as well as (ii) performing, for $n = 0, 1, \ldots$,

$$w_n = x_n - \frac{f(x_n)}{A_n}, \tag{106}$$

$$y_n = x_n - \frac{f(x_n)}{f[x_n, w_n] - B_n f(w_n)}, \tag{107}$$

$$t_n = \frac{f(y_n)}{f(x_n)}, \tag{108}$$

$$x_{n+1} = y_n - g(t_n)\frac{f(y_n)}{f[y_n, w_n] - B_n f(w_n)}, \tag{109}$$

$$A_{n+1} = f[x_n, w_n] + f[x_n, w_n, y_n](2x_{n+1} - x_n - w_n), \tag{110}$$

$$B_{n+1} = \frac{f[x_n, w_n, y_n]}{A_{n+1}}. \tag{111}$$

In the numerical test, we take $g(t) = 1 + t$. Three evaluations of functions for $f(x_n)$, $f(w_n)$, and $f(y_n)$ are needed.

Like the accelerated third new memory method in Section 3.3, we propose a modification of DMM (MDMM). The new memory method of MDMM reads as (i) giving x_0, a_0, b_0, $c_0 = (a_0 + b_0)/2$, η_0 and computing A_0 and B_0 by Equation (53), as well as (ii) performing, for $n = 0, 1, \ldots$,

$$w_n = x_n - \frac{\eta_n f(x_n)}{A_n}, \tag{112}$$

$$y_n = x_n - \frac{f(x_n)}{f[x_n, w_n] - B_n f(w_n)}, \tag{113}$$

$$t_n = \frac{f(y_n)}{f(x_n)}, \tag{114}$$

$$x_{n+1} = y_n - g(t_n)\frac{f(y_n)}{f[y_n, w_n] - B_n f(w_n)}, \tag{115}$$

$$A_{n+1} = f[x_n, w_n] + f[x_n, w_n, y_n](2x_{n+1} - x_n - w_n), \tag{116}$$

$$B_{n+1} = \frac{f[x_n, w_n, y_n]}{A_{n+1}}, \tag{117}$$

$$\eta_{n+1} = \frac{A_{n+1}[f(x_n) - f(w_n)][(x_{n+1} - x_n)(x_n - y_n) + 1]}{f(x_n)\{[f(x_{n+1}) - f(x_n)](x_n - y_n) + f[w_n, y_n]\}}. \tag{118}$$

4.5. The CLBTMM Method

In 2015, Cordero et al. [50] proposed a two-step iterative scheme:

$$\begin{cases} y_n = x_n - \frac{f(x_n)}{f[x_n, w_n] + \lambda f(w_n)}, \\ x_{n+1} = y_n - \frac{f(y_n)}{f[x_n, y_n] + (y_n - x_n) f[x_n, w_n, y_n]}, \end{cases} \tag{119}$$

where $w_n = x_n + \gamma f(x_n)$. They argued that the memory-dependent method possesses at least seven-order convergence.

Let $A = f'(r) = -1/\gamma$ and $B = -\lambda = a_2$. The new CLBT memory method (CLBTMM) reads as (i) giving x_0, a_0, b_0, $c_0 = (a_0 + b_0)/2$ and computing A_0 and B_0 by Equation (53), as well as (ii) performing, for $n = 0, 1, \ldots$,

$$w_n = x_n - \frac{f(x_n)}{A_n}, \tag{120}$$

$$y_n = x_n - \frac{f(x_n)}{f[x_n, w_n] - B_n f(w_n)}, \tag{121}$$

$$x_{n+1} = y_n - \frac{f(y_n)}{f[x_n, y_n] + (y_n - x_n) f[x_n, w_n, y_n]}, \tag{122}$$

$$A_{n+1} = f[x_n, w_n] + f[x_n, w_n, y_n](2x_{n+1} - x_n - w_n), \tag{123}$$

$$B_{n+1} = \frac{f[x_n, w_n, y_n]}{A_{n+1}}. \tag{124}$$

Three evaluations of functions for $f(x_n)$, $f(w_n)$, and $f(y_n)$ are needed.

5. Numerical Verifications of the New Memory-Updating Method

We give some examples to assess the performance of the proposed iterative methods by the numerically computed order of convergence (COC), which is approximated by [21,51]

$$\text{COC1} := \frac{\ln|(x_{n+1} - r)/(x_n - r)|}{\ln|(x_n - r)/(x_{n-1} - r)|}, \quad \text{COC2} := \frac{\ln|f(x_{n+1})/f(x_n)|}{\ln|f(x_n)/f(x_{n-1})|}. \tag{125}$$

The triple (x_{n+1}, x_n, x_{n-1}) is the last three values. If $\ln|(x_{n+1} - r)/(x_n - r)|$ and $\ln|f(x_{n+1})/f(x_n)|$ are not computable due to $x_{n+1} = r$ and $f(x_{n+1}) = 0$, we can shift the triple forward to (x_n, x_{n-1}, x_{n-2}).

To save space, we test three examples by

$$f_1(x) = \exp(x^2 + 7x - 30) - 1, \tag{126}$$

$$f_2(x) = (x - 1)(x^6 + 1/x^6 + 4)\sin(x^2), \tag{127}$$

$$f_3(x) = (x - 1)^3 - 1. \tag{128}$$

The corresponding solutions are, respectively, $r_1 = 3$, $r_2 = 1$, and $r_3 = 2$.

As noticed by Džunić [34], $f_2(x)$ shows a nontrivial behavior since two relatively close roots, $r = -1.772453850905516$ and $r = 1.772453850905516$, appear, and a singularity is close to the sought root $r = 1$. In practice, we solved $f_2(x) = 0$ by the Newton method. With $x_0 = 0.6$, the NM spent 12 iterations to find $r = 1$; with $x_0 = \pm 2$; the NM does not converge to $r = \pm 1.772453850905516$ within 500 iterations.

With three evaluations of functions, the optimal order of convergence of the two-step iterative scheme without memory is $p = 4$ and E.I. = 1.5874. In Table 1, we list the number of iterations (NI) to satisfy $f_i = 0$, $i = 1, 2, 3$, where, as expected, the value of E.I. is near or larger than 1.5874. The roots of $f_3 = 0$ are triple; however, FNMUM can still quickly find the solution $x = 2$. Through the new memory-updating method, the convergence is significantly increased by several orders.

Table 1. The NI, COC1, COC2, and E.I. for the first new memory-updating method (FNMUM).

Functions	x_0	$[a_0, b_0]$	NI	COC1	COC2	E.I. = $(\text{COC1})^{1/3}$
f_1	2.9	$[2.9, 3.1]$	5	3.973	4.095	1.584
f_2	0.5	$[0.5, 1.5]$	4	5.274	5.295	1.741
f_3	1.5	$[0.5, 1]$	5	4.566	4.621	1.659

Compared with the Newton method mentioned above, for $f_2(x) = 0$, the FNMUM with $x_0 = 0.6$ spent five iterations to find $r = 1$; with $x_0 = -2$, the FNMUM spent eight iterations for $r = -1.772453850905516$; and with $x_0 = 2$, it took six iterations for $r = 1.772453850905516$.

For $f_1(x) = 0$, the exact values of $f'(r) = 13$ and $f''(r) = 171$ are obtained; hence, we can estimate the errors of parameters' values by ERR1 = $|A_n - 13|$ and ERR2 = $|B_n - 171/26| = |B_n - 6.576923076923077|$. Table 2 demonstrates that A_n and B_n tended to exact ones.

Table 2. The values of A_n and B_n tend to exact ones.

n	1	2	3	4
ERR1	4.15	3.45	0.259	1.55×10^{-4}
ERR2	0.802	2.86	1.85×10^{-2}	4.69×10^{-2}

In Table 3, we list the number of iterations (NI) to satisfy $f_i = 0$, $i = 1, 2, 3$, where the value of E.I. is near or larger than 1.5874. As expected, the second new memory-updating method significantly increased the COC by more than 4.7 for $f_1(x) = 0$ and $f_2(x) = 0$. But for $f_3(x) = 0$, The SNMUM is weak.

Table 3. The NI, COC1, COC2, and E.I. for the second new memory-updating method (SNMUM).

Functions	x_0	$[a_0, b_0]$	NI	COC1	COC2	E.I. = $(\text{COC1})^{1/3}$
f_1	2.9	$[2.8, 3.1]$	4	4.711	4.713	1.676
f_2	0.5	$[0.5, 1.5]$	4	5.439	5.459	1.759
f_3	1.5	$[1.5, 2.5]$	5	3.627	3.692	1.536

In Table 4, we list the results for the third new memory-updating method (TNMUM). As expected, the COCs are greater than four for $f_2(x) = 0$ and $f_3(x) = 0$.

Table 4. The NI, COC1, COC2, and E.I. for the third new memory-updating method (TNMUM).

Functions	x_0	$[a_0, b_0]$	NI	COC1	COC2	E.I. = $(\text{COC1})^{1/3}$
f_1	2.9	$[2.95, 3.01]$	5	3.677	3.794	1.544
f_2	0.5	$[-0.5, 1.4]$	4	4.313	4.334	1.628
f_3	1.5	$[-0.5, 1.5]$	5	4.839	5.015	1.691

In Table 5, we list the results for the accelerated third new memory-updating method (ATNMUM). As expected, the COCs are larger than those in Table 3.

Table 5. The NI, COC1, COC2, and E.I. for the accelerated third new memory-updating method (ATNMUM).

Functions	x_0	$[A_0, B_0]$	η_0	NI	COC1	COC2	E.I. = $(\text{COC1})^{1/3}$
f_1	2.9	$[13.97, 6.35]$	-2	4	6.913	6.959	1.905
f_2	0.5	$[14.44, -0.33]$	1.5	4	10.296	10.354	2.176
f_3	2.5	$[1.75, 0.86]$	2	5	5.635	5.625	1.780

We can estimate the errors by ERR1=$|A_n - 13|$ and ERR2=$|\mathcal{N}_2(x_n) - f(x_n)|$ for the solution of $f_1(x) = 0$ by using the ATNMUM. Table 6 demonstrates that A_n tends to an exact one, and $\mathcal{N}_2(x_n)$ quickly approaches to $f(x_n)$ owing to the design of the relaxation factor in Equation (75).

Table 6. The values of A_n and $\mathcal{N}_2(x_n)$ tend to exact ones.

n	1	2	3
ERR1	4.22	7.82×10^{-2}	1.60×10^{-4}
ERR2	5.72×10^{-2}	2.92×10^{-6}	4.43×10^{-10}

In Table 7, we list the number of iterations (NI) for solving $f_i = 0,\ i = 1, 2, 3$ by the WMM, of which four evaluations of functions are required; hence, we take E.I. = $(\text{COC1})^{1/4}$. The good performance of the new memory method can be seen, which raises COC=4 to values in the range $[5.592, 7.576]$.

Table 7. The NI, COC1, COC2, and E.I. for the WMM method.

Functions	x_0	$[a_0, b_0]$	NI	COC1	COC2	E.I. = $(\text{COC1})^{1/4}$
f_1	2.9	$[2.9, 3.2]$	5	6.447	6.369	1.594
f_2	0.6	$[0.5, 1.5]$	3	5.592	4.459	1.538
f_3	1.5	$[0.5, 1]$	5	7.576	6.796	1.659

In Table 8, we list the number of iterations (NI) for solving $f_i = 0,\ i = 1, 2, 3$ by the ZLHMM method.

Table 8. The NI, COC1, COC2, and E.I. for the ZLHMM method.

Functions	x_0	$[a_0, b_0]$	NI	COC1	COC2	E.I. = $(\text{COC1})^{1/3}$
f_1	2.9	$[2.8, 3.2]$	4	5.388	7.291	1.753
f_2	0.6	$[0.6, 2.5]$	6	5.854	5.754	1.802
f_3	1.5	$[0.5, 1]$	4	5.011	4.743	1.711

In Table 9, we list the number of iterations (NI) to satisfy $f_i = 0,\ i = 1, 2, 3$, where, as expected, the value of E.I. is near or larger than 1.7. Moreover, with the same initial value $x_0 = 2.5$, the presented COC1 and NI for $f_3(x) = 0$ are better than those computed by Chicharro et al. [37], where NI=6 and ACOC=4.476. The values 1.705, 1.736, and 1.986 are also better than the E.I. = $8^{1/4} = 1.682$ obtained by the eighth-order optimal iterative scheme with four evaluations of functions.

Table 9. The NI, COC1, COC2, and E.I. for the CCTMM method.

Functions	x_0	$[a_0, b_0]$	NI	COC1	COC2	E.I. = $(\text{COC1})^{1/3}$
f_1	2.9	$[2.5, 3.1]$	3	4.959	5.695	1.705
f_2	0.2	$[0.2, 1.5]$	4	5.236	4.967	1.736
f_3	2.5	$[-0.5, 1]$	4	7.833	6.299	1.986

In Table 10, the presented COC2 for $f_2(x) = 0$ is better than that computed by Džunić [34], where COC = 6.9 is smaller than 7.3. Even if we do not use the full memory information, the performance is better than that in [34].

Table 10. The NI, COC1, COC2, and E.I. for the DMM method.

Functions	x_0	$[a_0, b_0]$	NI	COC1	COC2	E.I. = $(\text{COC1})^{1/3}$
f_1	2.7	$[2.7, 3.2]$	4	11.775	8.861	2.275
f_2	0.6	$[0.5, 2]$	5	7.408	7.300	1.949
f_3	2.5	$[0.5, 3]$	3	5.755	6.479	1.792

In Table 11, the presented COC2 for $f_2(x) = 0$ is better than that computed by Džunić [34], where COC = 6.9 is smaller than 11.959. Even if we do not use the full memory information, the performance is better than that in [34]. Compared with Table 10, the high performance was gained in Table 11 by introducing a relaxation factor in the modified Džunić's memory method. The COC2 = 21.009 is abnormal for the solution of $f_3(x) = 0$.

Table 11. The NI, COC1, COC2, and E.I. for the MDMM method.

Functions	x_0	$[a_0, b_0]$	η_0	NI	COC1	COC2	E.I. = $(\text{COC1})^{1/3}$
f_1	2.7	[2.7, 3.2]	-0.2	4	14.894	13.222	2.460
f_2	0.6	[0.5, 2]	0.15	5	10.658	11.959	2.200
f_3	2.5	[0.5, 3]	-0.47	3	6.533	21.009	1.869

In Table 12, the presented COC2 = 8.53 for $f_4(x) = x^3 + 4x^2 - 10 = 0$ with the root $r_4 = 1.365230013414097$ is better than that computed in [52], where COC=7 is smaller than 8.53. Even if we do not use the full memory information, the performance is better than that in [52], where $\gamma_n = -1/\mathcal{N}_3'(x_{n+1})$ and $\lambda_n = -\mathcal{N}_3''(w_{n+1})/[2\mathcal{N}_3'(w_{n+1})]$ were used for the memory-dependent function $\mathcal{N}_3(x; x_n, y_{n-1}, w_{n-1}, x_{n-1})$ in Equation (119). $\mathcal{N}_3(x)$ is the third-degree Newton interpolation polynomial.

Table 12. The NI, COC1, COC2, and E.I. for the CLBTMM method.

Functions	x_0	$[a_0, b_0]$	NI	COC1	E.I. = $(\text{COC1})^{1/3}$	COC2	E.I. = $(\text{COC2})^{1/3}$
f_1	2.8	[2.8, 3.2]	4	5.512	1.766	8.762	2.062
f_3	1.5	$[-1, 3]$	3	7.519	1.959	9.579	2.124
f_4	-0.5	[0.5, 2]	3	5.975	1.815	8.530	2.043
f_5	1.8	$[-0.5, 2.5]$	3	10.309	2.176	9.686	2.132

The presented COC2 = 9.579 for $f_3(x) = (x-1)^3 - 1 = 0$ is better than that computed in [50], where COC = 6.9041 was obtained, which used $\gamma_n = -1/\mathcal{N}_3'(x_{n+1})$, $\lambda_n = -\mathcal{N}_4''(w_{n+1})/[2\mathcal{N}_4'(w_{n+1})]$, $\mathcal{N}_3(x; x_n, y_{n-1}, x_{n-1}, w_{n-1})$, and $\mathcal{N}_4(x; w_n, x_n, y_{n-1}, w_{n-1}, x_{n-1})$ in Equation (119). $\mathcal{N}_3(x)$ and $\mathcal{N}_4(x)$ are, respectively, the third-degree and fourth-degree Newton interpolation polynomials. For $f_5(x) = (x-2)(x^6 + x^3 + 1)e^{-x^2} = 0$ with the root $r_5 = 2$, the presented COC2 = 9.686 is better than the COC = 6.9727 computed in [50].

6. Enhancing the Updating Speed of Parameters

To further accelerate the updating speed of the parameters A_n and B_n, according to [52], we can take $A_n = \mathcal{N}_3'(x_{n+1})$ and $B_n = -\mathcal{N}_3''(w_{n+1})/[2\mathcal{N}_3'(w_{n+1})]$. The memory-dependent method of CLBTM reads as (i) giving x_0, a_0, b_0, $c_0 = (a_0 + b_0)/2$ and giving or computing A_0 and B_0 by Equation (53), and $w_0 = x_0 - f(x_0)/A_0$, as well as (ii) performing, for $n = 0, 1, \ldots$,

$$y_n = x_n - \frac{f(x_n)}{f[x_n, w_n] - B_n f(w_n)}, \tag{129}$$

$$x_{n+1} = y_n - \frac{f(y_n)}{f[x_n, y_n] + (y_n - x_n) f[x_n, w_n, y_n]}, \tag{130}$$

$$\begin{aligned} A_{n+1} &= f[x_n, w_n] + f[x_n, w_n, y_n](2x_{n+1} - w_n - x_n) + f[x_n, w_n, y_n, x_{n+1}] \\ &\times [(x_{n+1} - w_n)(x_{n+1} - y_n) + (x_{n+1} - x_n)(x_{n+1} - y_n) + (x_{n+1} - x_n)(x_{n+1} - w_n)], \\ w_{n+1} &= x_{n+1} - \tfrac{f(x_{n+1})}{A_{n+1}}, \end{aligned} \tag{131}$$

$$\begin{aligned} C_{n+1} &= 2f[x_n, w_n, y_n] + f[x_n, w_n, y_n, x_{n+1}][(6w_{n+1} - 2x_n - 2w_n - 2y_n) \\ D_{n+1} &= f[x_n, w_n] + f[x_n, w_n, y_n](2w_{n+1} - w_n - x_n) + f[x_n, w_n, y_n, x_{n+1}] \\ &\times[(w_{n+1} - w_n)(w_{n+1} - y_n) + (w_{n+1} - x_n)(w_{n+1} - y_n) + (w_{n+1} - x_n)(w_{n+1} - w_n)], \\ B_{n+1} &= \frac{C_{n+1}}{2D_{n+1}}, \end{aligned} \tag{132}$$

where

$$f[x_n, w_n, y_n, x_{n+1}] = \frac{f[x_n, w_n, y_n] - f[w_n, y_n, x_{n+1}]}{x_n - x_{n+1}}. \tag{133}$$

In Table 13, the presented COC2 = 9.157 for $f_4(x) = x^3 + 4x^2 - 10 = 0$ is better than that computed in [52]. The presented COC2 = 15.634 for $f_3(x) = (x-1)^3 - 1 = 0$ is better than that computed in [50], where COC = 6.9041 was obtained. The presented COC2 = 10.159 for $f_5(x) = (x-2)(x^6 + x^3 + 1)e^{-x^2} = 0$ is better than that computed in [50], where COC = 6.9727 was obtained.

Table 13. The NI, COC1, COC2, and E.I. for the CLBTM method.

Functions	x_0	$[A_0, B_0]$	NI	COC1	E.I. = (COC1)$^{1/3}$	COC2	E.I. = (COC2)$^{1/3}$
f_1	2.85	[17.16, 5.78]	3	6.339	1.851	9.306	2.103
f_3	1.6	[4, 0]	3	11.805	2.277	19.053	2.671
f_4	1	[−10, −0.1]	3	11.827	2.278	13.027	2.353
f_5	2.5	[1.08, −1.07]	3	9.896	2.147	18.327	2.637

Comparing Table 13 with Table 12, the convergence speed as reflected in the values of COC and E.I. is enhanced by the CLBTM; however, the complexity is increased compared with the iterative scheme CLBTMM in Section 4.5.

On this occasion, we can point out the difference between the new memory method (NMM) and the memory method (MM): in the MM, $f(x_{n+1})$ is computed, but it is not needed in the NMM. The supplementary variable w_{n+1} is updated by Equation (131) in the MM, but no update of w is required in the NMM; in the NMM, a lower-order $\mathcal{N}_2$ polynomial is sufficient, but for the MM, the higher orders polynomials of $\mathcal{N}_3, \mathcal{N}_4, \mathcal{N}_5$ are necessary. According to the authors, the computational cost of the NMM saves much more than the MM.

7. Concluding Remarks

Traub [33] was the first to develop a memory method from the Steffnensen's iterative scheme:

$$\begin{cases} w_n = x_n - \gamma_n f(x_n), \\ x_{n+1} = x_n - \frac{f(x_n)}{f[x_n, w_n]}, \\ \gamma_{n+1} = f[x_n, x_{n+1}] \end{cases}$$

By giving x_0 and γ_0, the above iterative scheme can be initiated. We notice that in the proposed new memory-updating method, we do not need to compute $f(x_{n+1})$, which saves one more evaluation of the functions than Traub's memory method.

In [34], the accelerating technique is based on the memory of $(w_n, x_n, y_{n-1}, w_{n-1}, x_{n-1})$ by

$$\gamma_n = -\frac{1}{\mathcal{N}_3'(x_n)}, \quad p_n = -\frac{\mathcal{N}_4''(w_n)}{2\mathcal{N}_4'(w_n)}.$$

In addition to the cost of storing the information in the previous iteration, there is an expensive computational cost for computing $\mathcal{N}_3(x_n)$, $\mathcal{N}_4(w_n)$, and $\mathcal{N}_4''(w_n)$. Since the work

in [34], there has been extensive literature on this memory style using similar techniques. The new memory approach for updating the accelerating parameters proposed in this paper without using the information at the previous iteration, which takes the current values of (x_n, w_n, y_n) into $\mathcal{N}_2(x_{n+1})$ for updating A_n and B_n, is more computationally cost-effective than the previous techniques. The three new two-step iterative schemes developed together with four updating techniques worked very well to quickly solve nonlinear equations. Numerical examples revealed that without computing extra functions, the new memory-updating methods can raise several orders of convergence and significantly enhance the values of E.I.

We introduced an accelerating technique in the third new memory method by imposing $\mathcal{N}_2(x_{n+1}) = f(x_{n+1})$ to determine the relaxation factor. The values of COC and E.I. are greatly raised by this accelerated third new memory-updating method. When the relaxation factor acceleration technique was combined with the modified Džunić's memory method, very high values of COC = 13.222 and E.I. = 2.46 were achieved. High performance is achieved by the proposed two-step iterative methods to find the root of nonlinear equations.

The novelties involved in this paper are as follows:

- Developing three novel two-step iterative schemes with simple forms, which are derivative-free.
- A second-degree Newton polynomial was used to update two critical parameters, $f'(r)$ and $f''(r)$, greatly saving computation cost by evaluating three functions and using the new memory method (NMM).
- The NMM was applied to five existing two-step iterative schemes, WMM, ZLHMM, CCTMM, DMM, and CLBTMM, with the high values of E.I. all being larger than 1.5874.
- The new idea of imposing $\mathcal{N}_2(x_{n+1}) = f(x_{n+1})$ to determine the relaxation factor was developed, whose resulting E.I. $\in [1.78, 2.176]$ is larger than E.I. = 1.5874 of the fourth-order optimal iterative scheme.
- Combining the relaxation factor acceleration technique and Džunić's new memory method, very high values of COC and E.I. were achieved.

Author Contributions: Conceptualization, C.-S.L.; Methodology, C.-S.L. and C.-W.C.; Software, C.-S.L. and C.-W.C.; Validation, C.-S.L. and C.-W.C.; Formal analysis, C.-S.L. and C.-W.C.; Investigation, C.-S.L. and C.-W.C.; Resources, C.-W.C.; Data curation, C.-S.L. and C.-W.C.; Writing–original draft, C.-S.L.; Writing–review and editing, C.-W.C.; Visualization, C.-S.L. and C.-W.C.; Supervision, C.-S.L.; Project administration, C.-S.L.; Funding acquisition, C.-W.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding authors. The data are not publicly available due to restrictions privacy.

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

Abbreviations

The following abbreviations are used in this manuscript:

ATNMUM	Accelerated third new memory-updating method
BVP	Boundary value problem
CCTMM	Chicharro, Cordero, and Torregrosa's memory method
CLBTM	Cordero, Lotfi, Bakhtiari, and Torregrosa's method
CLBTMM	Cordero, Lotfi, Bakhtiari, and Torregrosa's memory method
COC	Computed order of convergence
DMM	Džunić's memory method
E.I.	Efficiency index
FNMUM	First new memory-updating method
MDMM	Modification of Džunić's memory method
NI	number of iterations
NM	Newton method

NMM	New memory method
ODE	Ordinary differential equation
SNMUM	Second new memory-updating method
TNMUM	Third new memory-updating method
WMM	Wang memory method
ZLHMM	Zheng, Li, and Huang's memory method

References

1. Liu, C.S.; El-Zahar, E.R.; Chang, C.W. A two-dimensional variant of Newton's method and a three-point Hermite interpolation: Fourth- and eighth-order optimal iterative schemes. *Mathematics* **2023**, *11*, 4529. [CrossRef]
2. Wu, X.Y. A new continuation Newton-like method and its deformation. *Appl. Math. Comput.* **2000**, *112*, 75–78. [CrossRef]
3. Lee, M.Y.; Kim, Y.I.; Magrenãń, A.A. On the dynamics of tri-parametric family of optimal fourthorder multiple-zero finders with a weight function of the principal m^{th} root of a function-function ratio. *Appl. Math. Comput.* **2017**, *315*, 564–590.
4. Zafar, F.; Cordero, A.; Torregrosa, J.R. Stability analysis of a family of optimal fourth-order methods for multiple roots. *Numer. Algor.* **2019**, *81*, 947–981. [CrossRef]
5. Thangkhenpau, G.; Panday, S.; Mittal, S.K.; Jäntschi, L. Novel parametric families of with and without memory iterative methods for multiple roots of nonlinear equations. *Mathematics* **2023**, *14*, 2036. [CrossRef]
6. Singh, M.K.; Singh, A.K. A derivative free globally convergent method and its deformations. *Arab. J. Math.* **2021**, *10*, 481–496. [CrossRef]
7. Singh, M.K.; Argyros, I.K. The dynamics of a continuous Newton-like method. *Mathematics* **2022**, *10*, 3602. [CrossRef]
8. Wu, X.Y. Newton-like method with some remarks. *Appl. Math. Comput.* **2007**, *118*, 433–439. [CrossRef]
9. Liu, C.S.; Chang, C.W.; Kuo, C.L. Memory-accelerating methods for one-step iterative schemes with Lie-symmetry method solving nonlinear boundary value problem. *Symmetry* **2024**, *16*, 120. [CrossRef]
10. Liu, C.S. Elastoplastic models and oscillators solved by a Lie-group differential algebraic equations method. *Int. J. Non-Linear Mech.* **2015**, *69*, 93–108. [CrossRef]
11. Yu, Y.X. A novel weighted density functional theory for adsorption, fluid-solid interfacial tension, and disjoining properties of simple liquid films on planar solid surfaces. *J. Chem. Phys.* **2009**, *131*, 024704. [CrossRef] [PubMed]
12. Alazwari, M.A.; Abu-Hamdeh, N.H.; Goodarzi, M. Entropy optimization of first-grade viscoelastic nanofluid flow over a stretching sheet by using classical Keller-box scheme. *Mathematics* **2021**, *9*, 2563. [CrossRef]
13. Khan, F.A.; Aldhabani, M.S.; Alamer, A.; Alshaban, E.; Alamrani, F.M.; Mohammed, H.I.A. Almost nonlinear contractions under locally finitely transitive relations with applications to integral equations. *Mathematics* **2023**, *11*, 4749. [CrossRef]
14. He, J.H. Variational iteration method—A kind of non-linear analytical technique: Some examples. *Int. J. Non-linear Mech.* **1999**, *34*, 699–708. [CrossRef]
15. He, J.H. Variational iteration method for autonomous ordinary systems. *Appl. Math. Comput.* **2000**, *114*, 115–123. [CrossRef]
16. Wang, X.; He, W.; Feng, H.; Atluri, S.N. Fast and accurate predictor-corrector methods using feedback-accelerated Picard iteration for strongly nonlinear problems. *Comput. Model. Eng. Sci.* **2024**, *139*, 1263–1294. [CrossRef]
17. Argyros, I.K.; Shakhno, S.M.; Yarmola, H.P. Extended semilocal convergence for the Newton-Kurchatov method. *Mat. Stud.* **2020**, *53*, 85–91.
18. Argyros, I.K.; Shakhno, S.M. Extended local convergence for the combined Newton-Kurchatov method under the generalized Lipschitz conditions. *Mathematics* **2019**, *7*, 207. [CrossRef]
19. Abbasbandy, S. Improving Newton-Raphson method for nonlinear equations by modified Adomian decomposition method. *Appl. Math. Comput.* **2003**, *145*, 887–893. [CrossRef]
20. Chun, C. Iterative methods improving Newton's method by the decomposition method. *Comput. Math. Appl.* **2005**, *50*, 1559–1568. [CrossRef]
21. Weerakoon, S.; Fernando, T.G.I. A variant of Newton's method with accelerated third-order convergence. *Appl. Math. Lett.* **2000**, *13*, 87–93. [CrossRef]
22. Morlando, F. A class of two-step Newton's methods with accelerated third-order convergence. *Gen. Math. Notes* **2015**, *29*, 17–26.
23. Ogbereyivwe, O.; Umar S.S. Behind Weerakoon and Fernando's scheme: Is Weerakoon and Fernando's scheme version computationally better than its power-means variants? *FUDMA J. Sci.* **2023**, *7*, 368–371. [CrossRef]
24. Saqib, M.; Iqbal, M. Some multi-step iterative methods for solving nonlinear equations. *Open J. Math. Sci.* **2017**, *1*, 25–33. [CrossRef]
25. Zhanlav, T.; Chuluunbaatar, O.; Ulziibayar, V. Generating function method for constructing new iterations. *Appl. Math. Comput.* **2017**, *315*, 414–423. [CrossRef]
26. Argyros, I.K.; Regmi, S.; Shakhno, S.; Yarmola, H. Perturbed Newton methods for solving nonlinear equations with applications. *Symmetry* **2022**, *14*, 2206 [CrossRef]
27. Chanu, W.H.; Panday, S.; Thangkhenpau, G. Development of optimal iterative methods with their applications and basins of attraction. *Symmetry* **2022**, *14*, 2020. [CrossRef]
28. Petkovic, M.; Neta, B.; Petkovic, L.; Džunić, J. *Multipoint Methods for Solving Nonlinear Equations*; Elsevier: Amsterdam, The Netherlands, 2013.

29. Amat, S.; Busquier, S. *Advances in Iterative Methods for Nonlinear Equations*; SEMA-SEMAI Springer series; Springer: Cham, Switzerland, 2016.
30. Kung, H.T.; Traub, J.F. Optimal order of one-point and multi-point iterations. *J. Assoc. Comput. Machinery* **1974**, *21*, 643–651. [CrossRef]
31. Liu, C.S.; Hong, H.K.; Lee, T.L. A splitting method to solve a single nonlinear equation with derivative-free iterative schemes. *Math. Comput. Simul.* **2021**, *190*, 837–847. [CrossRef]
32. Liu, C.S. A new splitting technique for solving nonlinear equations by an iterative scheme. *J. Math. Res.* **2020**, *12*, 40–48. [CrossRef]
33. Traub, J. F. *Iterative Methods for the Solution of Equations*; Prentice-Hall: New York, NY, USA, 1964.
34. Džunić, J. On efficient two-parameter methods for solving nonlinear equations. *Numer. Algor.* **2013**, *63*, 549–569.
35. Lotfi, T.; Soleymani, F.; Noori, Z.; KJlJçman, A.; Khaksar Haghani, F. Efficient iterative methods with and without memory possessing high efficiency indices. *Discr. Dyna. Natu. Soc.* **2014**, *2014*, 912796. [CrossRef]
36. Wang, X. An Ostrowski-type method with memory using a novel self-accelerating parameter. *J. Comput. Appl. Math.* **2018**, *330*, 710–720 [CrossRef]
37. Chicharro, F.I.; Cordero, A.; Torregrosa, J.R. Dynamics of iterative families with memory based on weight functions procedure. *Appl. Math. Comput.* **2019**, *354*, 286–298. [CrossRef]
38. Torkashvand, V.; Kazemi, M.; Moccari, M. Sturcture a family of three-step with-memory methods for solving nonlinear equations and their dynamics. *Math. Anal. Convex Optim.* **2021**, *2*, 119–137.
39. Sharma, E.; Panday, S.; Mittal, S.K.; Joit, D.M.; Pruteanu, L.L.; Jäntschi, L. Derivative-free families of with- and without-memory iterative methods for solving nonlinear equations and their engineering applications. *Mathematics* **2023**, *14*, 4512. [CrossRef]
40. Thangkhenpau, G.; Panday, S.; Bolundut, L.C.; Jäntschi, L. Efficient families of multi-point iterative methods and their self-acceleration with memory for solving nonlinear equations. *Symmetry* **2023**, *15*, 1546. [CrossRef]
41. Wang, H.; Liu, H. Note on a cubically convergent Newton-type method under weak conditions. *Acta Appl. Math.* **2010**, *110*, 725–735 [CrossRef]
42. Wang, X.; Zhang, T. A new family of Newton-type iterative methods with and without memory for solving nonlinear equations. *Calcolo* **2014**, *51*, 1–15. [CrossRef]
43. Ostrowski, A.M. *Solutions of Equations and System Equations*; Academic Press: New York, NY, USA, 1960.
44. Wang, X. A family of Newton-type iterative methods using some special self-accelerating parameters. *Int. J. Comput. Math.* **2018**, *95*, 2112–2127. [CrossRef]
45. Jain, P.; Chand, P.B. Derivative free iterative methods with memory having higher R-order of convergence. *Int. J. Nonl. Sci. Numer. Simul.* **2020**, *21*, 641–648. [CrossRef]
46. Zhou, X.; Liu, B. Iterative methods for multiple roots with memory using self-accelerating technique. *J. Comput. Appl. Math.* **2023**, *428*, 115181. [CrossRef]
47. Džunić, J.; Petković, M.S.; Petković, L.D. Three-point methods with and without memory for solving nonlinear equations. *Appl. Math. Comput.* **2012**, *218*, 4917–4927.
48. Džunić, J.; Petković, M.S. On generalized multipoint root-solvers with memory. *J. Comput. Appl. Math.* **2012**, *236*, 2909–2920.
49. Zheng, Q.; Li, J.; Huang, F. Optimal Steffensen-type families for solving nonlinear equations. *Appl. Math. Comput.* **2011**, *217*, 9592–9597. [CrossRef]
50. Cordero, A.; Lotfi, T.; Bakhtiari, P.; Torregrosa, J.R. An efficient two-parameter family with memory for nonlinear equations. *Numer. Algor.* **2015**, *68*, 323–335. [CrossRef]
51. Petković, M.S. Remarks on "On a general class of multipoint root-finding methods of high computational efficiency." *SIAM J. Numer. Anal.* **2011**, *49*, 1317–1319. [CrossRef]
52. Torkashvand, V.; Kazemi, M. On an efficient family with memory with high order of convergence for solving nonlinear equations. *Int. J. Indus. Math.* **2020**, *12*, IJIM-1260.