mathematics

MDPI

Article

Improving the Computational Efficiency of a Variant of Steffensen's Method for Nonlinear Equations

Fuad W. Khdhr [1], Rostam K. Saeed [1] and Fazlollah Soleymani [2,*]

[1] Department of Mathematics, College of Science, Salahaddin University, Erbil, Iraq; fuad.khdhr@su.edu.krd (F.W.K.); rostam.saeed@su.edu.krd (R.K.S.)

[2] Department of Mathematics, Institute for Advanced Studies in Basic Sciences (IASBS), Zanjan 45137-66731, Iran

* Correspondence: fazlollah.soleymani@gmail.com

Received: 21 January 2019; Accepted: 18 March 2019; Published: 26 March 2019

Abstract: Steffensen-type methods with memory were originally designed to solve nonlinear equations without the use of additional functional evaluations per computing step. In this paper, a variant of Steffensen's method is proposed which is derivative-free and with memory. In fact, using an acceleration technique via interpolation polynomials of appropriate degrees, the computational efficiency index of this scheme is improved. It is discussed that the new scheme is quite fast and has a high efficiency index. Finally, numerical investigations are brought forward to uphold the theoretical discussions.

Keywords: iterative methods; Steffensen's method; R-order; with memory; computational efficiency

1. Introduction

One of the commonly encountered topics in computational mathematics is to tackle solving a nonlinear algebraic equation. The equation can be presented as in the scalar case $f(x) = 0$, or more complicated as a system of nonlinear algebraic equations. The procedure of finding the solutions (if it exists) cannot be done analytically. In some cases, the analytic techniques only give the real result while its complex zeros should be found and reported. As such, numerical techniques are a viable choice for solving such nonlinear problems. Each of the existing computational procedures has their own domain of validity with some pros and cons [1,2].

Two classes of methods with the use of derivatives and without the use of derivatives are known to be useful depending on the application dealing with [3]. In the derivative-involved methods, a larger attraction basin along with a simple coding effort for higher dimensional problems is at hand which, in derivative-free methods, the area of choosing the initial approximations is smaller and extending to higher dimensional problems is via the application of a divided difference operator matrix, which is basically a dense matrix. However, the ease in not computing the derivative and, subsequently, the Jacobians, make the application of derivative-free methods more practical in several problems [4–7].

Here, an attempt is made at developing a computational method which is not only efficient in terms of the computational efficiency index, but also in terms of larger domains for the choice of the initial guesses/approximations for starting the proposed numerical method.

The Steffensen's method [8] for solving nonlinear scalar equations has quadratic convergence for simple zeros and given by:

$$\begin{cases} x_{k+1} = x_k - \dfrac{f(x_k)}{f}, \\ \qquad\qquad [x_k, w_k] \\ w_k = x_k + \beta f(x_k),\ \beta \in R\backslash\{0\},\ k \geq 0, \end{cases} \tag{1}$$

where the two-point divided difference is defined by:

$$f[x_k, w_k] = \frac{f(x_k) - f(w_x)}{x_k - w_k},$$

This scheme needs two function evaluations per cycle. Scheme (1) shows an excellent tool for constructing efficient iterative methods for nonlinear equations. This is because it is derivative-free with a free parameter. This parameter can, first of all, enlarge the attraction basins of Equation (1) or any of its subsequent methods and, second, can directly affect the improvement of the R-order of convergence and the efficiency index.

Recalling that Kung and Traub conjectured that the iterative method without memory based on m functions evaluation per iteration attain the optimal convergence of order 2^{m-1} [9,10].

The term "with memory" means that the values of the function associated with the computed approximations of the roots are used in subsequent iterations. This is unlike the term "without memory" in which the method only uses the current values to find the next estimate. As such, in a method with memory, the calculated results up to the desired numbers of iterations should be stored and then called to proceed.

Before proceeding the given idea to improve the speed of convergence, efficiency index, and the attraction basins, we provide a short literature by reviewing some of the existing methods with accelerated convergence order. Traub [11] proposed the following two-point method with memory of order 2.414:

$$\begin{cases} x_{k+1} = x_k - \dfrac{f(x_k)}{f[x_k, x_k + \beta_k f(x_k)]}, \\ \beta_k = \dfrac{-1}{f[x_k, z_{k-1}]}, \end{cases} \tag{2}$$

where $z_{k-1} = x_{k-1} + \beta_{k-1} f(x_{k-1})$, and $\beta_0 = -\text{sign}(f'(x_0))$ or $-\dfrac{1}{f[x_0, x_0 + f(x_0)]}$. This is one of the pioneering and fundamental methods with memory for solving nonlinear equations.

Džunić in [12] suggested an effective bi-parametric iterative method with memory of $\frac{1}{2}\left(3 + \sqrt{17}\right)$ R-order of convergence as follows:

$$\begin{cases} w_k = x_k + \beta_k f(x_k), \\ \beta_k = -\dfrac{1}{N_2'(x_k)},\ \zeta_k = -\dfrac{N_3''(w_k)}{2N_3'(w_k)},\ k \geq 1, \\ x_{k+1} = x_k - \dfrac{f(x_k)}{f[x_k, w_k] + \zeta_k f(w_k)}\ k \geq 0. \end{cases} \tag{3}$$

Moreover, Džunić and Petković [13] derived the following cubically convergent Steffensen-like method with memory:

$$\begin{cases} x_{k+1} = x_k - \dfrac{f(x_k)}{f[x_k, x_k + \beta_k f(x_k)]}, \\ \beta_k = \dfrac{-1}{f[x_k, z_{k-1}] + f[x_k, x_{k-1}] + f[x_{k-1}, z_{k-1}]}, \end{cases} \tag{4}$$

where $z_{k-1} = x_{k-1} + \beta_{k-1} f(x_{k-1})$ depending on the second-order Newton interpolation polynomial.

Various Steffensen-type methods are proposed in [14–17].

In fact, it is possible to improve the performance of the aforementioned method by considering several more sub-steps and improve the computational efficiency index via multi-step iterative methods. However, this procedure is more computational burdensome. Thus, the motivation here is to know that is it possible to improve the performance of numerical methods in terms of the computational efficiency index, basins of attraction, and the rate of convergence without adding more sub-steps and propose a numerical method as a one-step solver.

Hence, the aim of this paper is to design a one-step method with memory which is quite fast and has an improved efficiency index, based on the modification of the one-step method of Steffensen (Equation (1)) and increase the convergence order to 3.90057 without any additional functional evaluations.

The rest of this paper is ordered as follows: In Section 2, we develop the one-point Steffensen-type iterative scheme (Equation (1)) with memory which was proposed by [18]. We present the main goal in Section 3 by approximating the acceleration parameters involved in our contributed scheme by Newton's interpolating polynomial and, thus, improve the convergence R-order. The numerical reports are suggested in Section 4 to confirm the theoretical results. Some discussions are given in Section 5.

2. An Iterative Method

The following iterative method without memory was proposed by [18]:

$$\begin{cases} w_k = x_k - \beta f(x_k), \\ x_{k+1} = x_k - \dfrac{f(x_k)}{f[x_k, w_k]} \left(1 + \zeta \dfrac{f(w_k)}{f[x_k, w_k]}\right), \ \zeta \in \mathbb{R}, \end{cases} \tag{5}$$

with the following error equation to improve the performance of (1) in terms of having more free parameters:

$$e_{k+1} = -\left(-1 + \beta f'(\alpha)\right)(c_2 - \zeta)e_k^2 + O\left(e_k^3\right), \tag{6}$$

where $c_i = \frac{1}{i!}\frac{f^{(i)}(\alpha)}{f'(\alpha)}$. Using the error Equation (6), to derive Steffensen-type iterative methods with memory, we calculate the following parameters: $\beta = \beta_k$, $\zeta = \zeta_k$, by the formula:

$$\begin{cases} \beta_k = \dfrac{1}{\overline{f'}(x_\alpha)}, \\ \zeta_k = \bar{c}_2, \end{cases} \tag{7}$$

for $k = 1, 2, 3, \cdots$, while $\overline{f'}(x_\alpha)$, $\bar{c}_2$ are approximations to $f(\alpha)$ and c_2, respectively; where α is a simple zero of $f(x)$. In fact, Equation (7) shows a way to minimize the asymptotic error constant of Equation (6) by making this coefficient closer and closer to zero when the iterative method is converging to the true solution.

The initial estimates β_0 and ζ_0 must be chosen before starting the process of iterations. We state the Newton's interpolating polynomial of fourth and fifth-degree passing through the saved points as follows:

$$\begin{cases} N_4(t) = N_4(t; x_k, w_{k-1}, x_{k-1}, w_{k-2}, x_{k-2}), \\ N_5(t) = N_5(t; w_k, x_k, w_{k-1}, x_{k-1}, w_{k-2}, x_{k-2}). \end{cases} \tag{8}$$

Recalling that $N(t)$ is an interpolation polynomial for a given set of data points also known as the Newton's divided differences interpolation polynomial because the coefficients of the polynomial are calculated using Newton's divided differences method. For instance, here the set of data points for $N_4(t)$ are $\{\{x_k, f(x_k)\}, \{w_{k-1}, f(w_{k-1})\}, \{x_{k-1}, f(x_{k-1})\}, \{w_{k-2}, f(w_{k-2})\}, \{x_{k-2}, f(x_{k-2})\}\}$.

Now, using some modification on Equation (5) we present the following scheme:

$$\begin{cases} w_k = x_k - \beta_k f(x_k), \\ \beta_k = \dfrac{1}{N_4'(x_k)}, \ \zeta_k = \dfrac{N_5''(w_k)}{2N_5'(w_k)}, \ k \geq 2, \\ x_{k+1} = x_k - \dfrac{f(x_k)}{f[x_k, w_k]} \left(1 + \zeta_k \dfrac{f(w_k)}{f[x_k, w_k]}\right), \ k \geq 0. \end{cases} \tag{9}$$

Noting that the accelerator parameters β_k, ζ_k are getting updated and then used in the iterative method right after the second iterations, viz, $k \geq 2$. This means that the third line of Equation (9) is imposed at the beginning and after that the computed values are stored and used in the subsequent iterates. For $k = 1$, the degree of Newton interpolation polynomials would be two and three. However, for $k \geq 2$, interpolations of degrees four and five as given in Equation (8) can be used to increase the convergence order.

Additionally speaking, this acceleration of convergence would be attained without the use any more functional evaluations as well as imposing more steps. Thus, the proposed scheme with memory (Equation (9)) can be attractive for solving nonlinear equations.

3. Convergence Analysis

In this section, we show the convergence criteria of Equation (9) using Taylor's series expansion and several extensive symbolic computations.

Theorem 1. *Let the function $f(x)$ be sufficiently differentiable in a neighborhood of its simple zero α. If an initial approximation x_0 is necessarily close to α. Then, R-order of convergence for the one-step method (Equation (9)) with memory is 3.90057.*

Proof. The proof is done using the definition of the error equation as the difference between the k-estimate and the exact zero along with symbolic computations. Let the sequence $\{x_k\}$ and $\{w_k\}$ have convergence orders r and p, respectively. Namely,

$$e_{k+1} \sim e_k^r, \tag{10}$$

and:

$$e_{w,k} \sim e_k^p, \tag{11}$$

Therefore, using Equations (10) and (11), we have:

$$e_{k+1} \sim e_k^r \sim e_{k-1}^{r^2} \sim e_{k-2}^{r^3}, \tag{12}$$

and:

$$e_{w,k} \sim e_k^p \sim (e_{k-1}^r)^p \sim e_{k-2}^{pr^2}. \tag{13}$$

The associated error equations to the accelerating parameters β_k and ζ_k for Equation (9) can now be written as follows:

$$e_{w,k} \sim (-1 + \beta_k f'(\alpha))e_k \ , \tag{14}$$

and:

$$e_{k+1} \sim -(-1 + \beta_k f'(\alpha))(c_2 - \zeta_k)e_k^2. \tag{15}$$

On the other hand, by using a symbolic language and extensive computations one can find the following error terms for the involved terms existing in the fundamental error Equation (6):

$$-1 + \beta_k f'(\alpha) \sim c_5 e_{k-2} e_{k-1} e_{w,k-1} e_{w,k-2}, \tag{16}$$

$$c_2 - \zeta_k \sim c_6 e_{k-2} e_{k-1} e_{w,k-1} e_{w,k-2} \tag{17}$$

Combining Equations (14)–(17), we get that:

$$e_{w,k} \sim e_{k-2}^{r^2+pr+r+p+1}, \tag{18}$$

$$e_{k+1} \sim e_{k-2}^{2(r^2+pr+r+p+1)}. \tag{19}$$

We now compare the left and right hand side of Equations (12)–(19) and Equations (13)–(18), respectively. Thus, we have the following nonlinear system of equations in order to find the final R-orders:

$$\begin{cases} r^2p - \left(r^2 + pr + r + p + 1\right) = 0, \\ r^3 - 2\left(r^2 + pr + r + p + 1\right) = 0. \end{cases} \tag{20}$$

The positive real solution of (20) is $r = 3.90057$ and $p = 1.9502$. Therefore, the convergence R-order for Equation (9) is 3.90057. □

Since improving the convergence R-order is useless if the whole computational method is expensive, basically researcher judge on a new scheme based upon its computational efficiency index which is a tool in order to provide a trade-off between the whole computational cost and the attained R-order. Assuming the cost of calculating each functional evaluation is one, we can use the definition of efficiency index as $EI = p^{1/\theta}$, θ is the whole computational cost [19].

The computational efficiency index of Equation (9) is $3.90057^{\frac{1}{2}} \approx 1.97499 \approx 2$, which is clearly higher than efficiency index $2^{\frac{1}{2}} \approx 1.4142$ of Newton's and Steffensen's methods, $3.56155^{\frac{1}{2}} \approx 1.8872$ of (3) $3^{1/2} \approx 1.73205$ of Equation (4).

However, this improved computational efficiency is reported by ignoring the number of multiplication and division per computing cycle. By imposing a slight weight for such calculations one may once again obtain the improved computational efficiency of (9) in contrast to the existing schemes of the same type.

4. Numerical Computations

In this section, we compare the convergence performance of Equation (9), with three well-known iterative methods for solving four test problems numerically carried out in Mathematica 11.1. [20].

We denote Equations (1), (3), (5) and (9) with SM, DZ, PM, M4, respectively. We compare the our method with different methods, using $\beta_0 = 0.1$ and $\zeta_0 = 0.1$. Here, the computational order of convergence (coc) has been computed by the following formula [21]:

$$coc = \frac{\ln|(f(x_k)/f(x_{k-1})|}{\ln|(f(x_{k-1})/f(x_{k-2})|} \tag{21}$$

Recalling that using a complex initial approximation, one is able to find the complex roots of the nonlinear equations using (9).

Experiment 1. *Let us consider the following nonlinear test function:*

$$f_1(x) = (x - 2\tan(x))\left(x^3 - 8\right), \tag{22}$$

where $\alpha = 2$ *and* $x_0 = 1.7$.

Experiment 2. *We take into account the following nonlinear test function:*

$$f_2(x) = (x - 1)\left(x^{10} + x^3 + 1\right)\sin(x), \tag{23}$$

where $\alpha = 1$ *and* $x_0 = 0.7$.

Experiment 3. *We consider the following test problem now:*

$$f_3(x) = \tfrac{-x^3}{2} + 2\tan^{-1}(x) + 1, \tag{24}$$

where $\alpha \approx 1.8467200$ *and* $x_0 = 4$.

Experiment 4. *The last test problem is taken into consideration as follows:*

$$f_4(x) = \tan^{-1}(\exp(x+2)+1) + \tanh(\exp(-x\cos(x))) - \sin(\pi x), \tag{25}$$

where $\alpha \approx -3.6323572\cdots$ *and* $x_0 = -4.1$.

Tables 1–4 show that the proposed Equation (9) is of order 3.90057 and it is obviously believed to be of more advantageous than the other methods listed due to its fast speed and better accuracy.

For better comparisons, we present absolute residual errors $|f(x)|$, for each test function which are displayed in Tables 1–4. Additionally, we compute the computational order of convergence. Noting that we have used multiple precision arithmetic considering 2000 significant digits to observe and the asymptotic error constant and the coc as obviously as possible.

The results obtained by our proposed Equation (M4) are efficient and show better performance than other existing methods.

A significant challenge of executing high-order nonlinear solvers is in finding initial approximation to start the iterations when high accuracy calculating is needed.

Table 1. Result of comparisons for the function f_1.

Methods	$\lvert f_1(x_3)\rvert$	$\lvert f_1(x_4)\rvert$	$\lvert f_1(x_5)\rvert$	$\lvert f_1(x_6)\rvert$	**coc**
SM	4.1583	3.0743	1.4436	0.25430	2.00
DZ	0.13132	2.0026×10^{-7}	1.0181×10^{-27}	7.1731×10^{-99}	3.57
PM	1.8921×10^{-6}	4.5864×10^{-24}	1.0569×10^{-88}	7.5269×10^{-318}	3.55
M4	9.1741×10^{-6}	3.3242×10^{-26}	4.4181×10^{-103}	1.1147×10^{-404}	3.92

Table 2. Result of comparisons for the function f_2.

Methods	$\lvert f_2(x_5)\rvert$	$\lvert f_2(x_6)\rvert$	$\lvert f_2(x_7)\rvert$	$\lvert f_2(x_8)\rvert$	**coc**
SM	–	–	–	–	–
DZ	0.14774	0.0016019.	1.3204×10^{-10}	1.5335×10^{-35}	3.56
PM	2.1191×10^{-10}	8.0792×10^{-35}	1.9037×10^{-121}	3.7062×10^{-430}	3.56
M4	5.9738×10^{-15}	4.1615×10^{-57}	1.7309×10^{-220}	1.8231×10^{-857}	3.90

Table 3. Result of comparisons for the function f_3.

Methods	$\lvert f_3(x_3)\rvert$	$\lvert f_3(x_4)\rvert$	$\lvert f_3(x_5)\rvert$	$\lvert f_3(x_6)\rvert$	**coc**
SM	0.042162	0.00012627	1.1589×10^{-9}	9.7638×10^{-20}	2.00
DZ	1.0219×10^{-11}	4.4086×10^{-44}	1.6412×10^{-157}	1.5347×10^{-562}	3.57
PM	7.9792×10^{-8}	3.712×10^{-30}	4.9556×10^{-108}	2.9954×10^{-386}	3.57
M4	4.4718×10^{-6}	2.9187×10^{-25}	4.7057×10^{-101}	1.0495×10^{-395}	3.89

To discuss further, mostly based on interval mathematics, one can find a close enough guess to start the process. There are some other ways to determine the real initial approximation to start the process. An idea of finding such initial guesses given in [22] is based on the useful commands in Mathematica 11.1 NDSolve [] for the nonlinear function on the interval $D = [a, b]$.

Following this the following piece of Mathematica code could give a list of initial approximations in the working interval for Experiment 4:

```
ClearAll[“Global`*”]

(*Defining the nonlinear function.*)
f[x_]:=ArcTan[Exp[x+2]+1]+Tanh[Exp[−x Cos[x]]]−Sin[Pi x];

(*Defining the interval.*)
a=−4.; b=4.;

(*Find the list of initial estimates.*)
Zeros = Quiet@Reap[soln=y[x]/.First[NDSolve[{y’[x]
==Evaluate[D[f[x],x]],y[b]==(f[b])},y[x],{x,a,b},
Method->{“EventLocator”,”Event”->y[x], “EventAction”:>Sow[{x,y[x]}]}]][[2,1]];
initialPoints = Sort[Flatten[Take[zeros,Length[zeros],1]]]
```

To check the position of the zero and the graph of the function, we can use the following code to obtain Figure 1.

```
Length[initialPoints]
Plot[f[x],{x,a,b}, Epilog->{PointSize[Medium], Red, Point[zeros]},PlotRange->All,  PerformanceGoal-
>“Quality”, PlotStyle->{Thick, Blue}]
```

Table 4. Result of comparisons for the function f_4.

Methods	$\lvert f_4(x_3)\rvert$	$\lvert f_4(x_4)\rvert$	$\lvert f_4(x_5)\rvert$	$\lvert f_4(x_6)\rvert$	**coc**
SM	0.00001166	3.7123×10^{-10}	3.7616×10^{-19}	3.8622×10^{-37}	2.00
DZ	1.6×10^{-13}	6.9981×10^{-47}	1.0583×10^{-164}	7.0664×10^{-585}	3.57
PM	3.0531×10^{-11}	3.2196×10^{-38}	3.7357×10^{-134}	6.5771×10^{-476}	3.56
M4	2.5268×10^{-13}	1.5972×10^{-49}	2.8738×10^{-191}	1.6018×10^{-744}	3.90

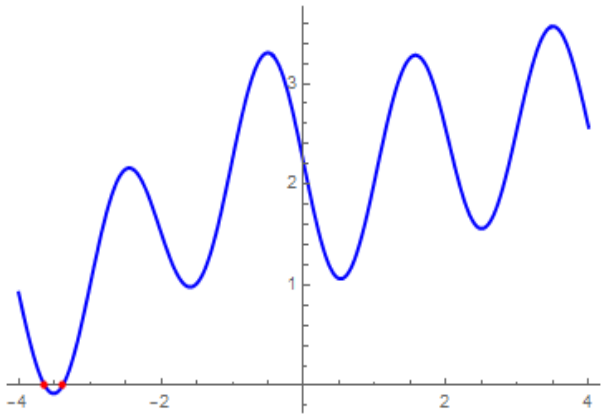

Figure 1. The plot of the nonlinear function in Experiment 4 along with its roots colored in red.

As a harder test problem, for the nonlinear function $g(x) = 2x + 0.5\sin(20\pi\, x) - x^2$, we can simply find a list of estimates as initial guesses using the above piece of codes as follows: $\{-0.185014, -0.162392, -0.0935912, -0.0535277, 6.73675 \times 10^{-9}, 0.0533287, 0.0941576, 0.160021, 0.188066, 0.269075, 0.279428, 1.76552, 1.78616, 1.8588, 1.89339, 1.95294, 2., 2.04692, 2.10748, 2.13979, 2.2228, 2.22471\}$. The plot of the function in this case is brought forward in Figure 2.

We observe that the two self-accelerating parameters β_0 and ζ_0 have to be selected before the iterative procedure is started. That is, they are calculated by using information existing from the present and previous iterations (see, e.g., [23]). The initial estimates β_0 and ζ_0 should be preserved as precise small positive values. We use $\beta_0 = \zeta_0 = 0.1$ whenever required.

After a number of iterates, the (nonzero) free parameters start converging to a particular value which makes the coefficient of Equation (6) zero as well as make the numerical scheme to converge with high R-order.

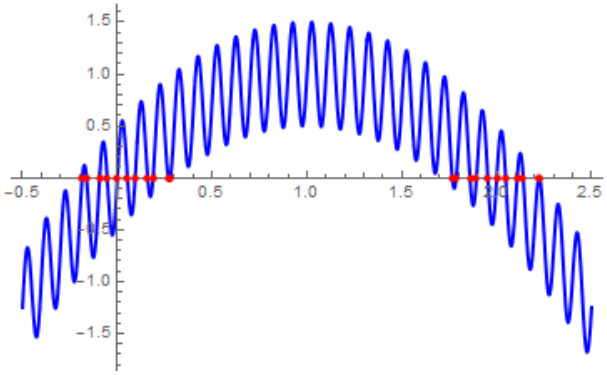

Figure 2. The behavior of the function g and the position of its roots (the red dots show the location of the zeros of the nonlinear functions).

5. Ending Comments

In this paper, we have constructed a one-step method with memory to solve nonlinear equations. By using two self-accelerator parameters our scheme equipped with Newton's interpolation polynomial without any additional functional calculation possesses the high computational efficiency index 1.97499, which is higher than many of the existing methods.

The efficacy of our scheme is confirmed by some of numerical examples. The results in Tables 1–4 shows that our method (Equation (M4)) is valuable to find an adequate estimate of the exact solution of nonlinear equations.

Author Contributions: The authors contributed equally to this paper.

Funding: This research received no external funding.

Acknowledgments: The authors are thankful to two anonymous referees for careful reading and valuable comments which improved the quality of this manuscript.

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

References

1. Cordero, A.; Hueso, J.L.; Martinez, E.; Torregrosa, J.R. Steffensen type methods for solving nonlinear equations. *J. Comput. Appl. Math.* **2012**, *236*, 3058–3064. [CrossRef]
2. Soleymani, F. Some optimal iterative methods and their with memory variants. *J. Egypt. Math. Soc.* **2013**, *21*, 133–141. [CrossRef]
3. Praks, P.; Brkić, D. Choosing the optimal multi-point iterative method for the Colebrook Flow friction equation. *Processes* **2018**, *6*, 130. [CrossRef]
4. Zafar, F.; Cordero, A.; Torregrosa, J.R. An efficient family of optimal eighth-order multiple root finders. *Mathematics* **2018**, *6*, 310. [CrossRef]
5. Saeed, R.K.; Aziz, K.M. An iterative method with quartic convergence for solving nonlinear equations. *Appl. Math. Comput.* **2008**, *202*, 435–440. [CrossRef]
6. Saeed, R.K. Six order iterative method for solving nonlinear equations. *World Appl. Sci. J.* **2010**, *11*, 1393–1397.
7. Torkashvand, V.; Lotfi, T.; Araghi, M.A.F. A new family of adaptive methods with memory for solving nonlinear equations. *Math. Sci.* **2019**, 1–20.
8. Noda, T. The Steffensen iteration method for systems of nonlinear equations. *Proc. Jpn. Acad.* **1987**, *63*, 186–189.
9. Kung, H.T.; Traub, J.F. Optimal order of one-point and multipoint iteration. *J. Assoc. Comput. Math.* **1974**, *21*, 634–651. [CrossRef]

10. Ahmad, F. Comment on: On the Kung-Traub conjecture for iterative methods for solving quadratic equations. *Algorithms* **2016**, *9*, 30. [CrossRef]
11. Traub, J.F. *Iterative Methods for the Solution of Equations*; Prentice-Hall: Englewood Cliffs, NJ, USA, 1964.
12. Džunić, J. On efficient two-parameter methods for solving nonlinear equations. *Numer. Algorithms* **2013**, *63*, 549–569. [CrossRef]
13. Džunić, J.; Petković, M.S. On generalized biparametric multipoint root finding methods with memory. *J. Comput. Appl. Math.* **2014**, *255*, 362–375. [CrossRef]
14. Zheng, O.; Wang, J.; Zhang, P.L. A Steffensen-like method and its higher-order variants. *Appl. Math. Comput.* **2009**, *214*, 10–16. [CrossRef]
15. Lotfi, T.; Tavakoli, E. On a new efficient Steffensen-like iterative class by applying a suitable self-accelerator parameter. *Sci. World J.* **2014**, *2014*, 769758. [CrossRef]
16. Zheng, O.P.; Zhao, L.; Ma, W. Variants of Steffensen-Secant method and applications. *Appl. Math. Comput.* **2010**, *216*, 3486–3496. [CrossRef]
17. Petković, M.S.; Ilić, S.; Džunić, J. Derivative free two-point methods with and without memory for solving nonlinear equations. *Appl. Math. Comput.* **2010**, *217*, 1887–1895.
18. Khaksar Haghani, F. A modiffied Steffensen's method with memory for nonlinear equations. *Int. J. Math. Model. Comput.* **2015**, *5*, 41–48.
19. Howk, C.L.; Hueso, J.L.; Martinez, E.; Teruel, C. A class of efficient high-order iterative methods with memory for nonlinear equations and their dynamics. *Math. Meth. Appl. Sci.* **2018**, 1–20. [CrossRef]
20. Cliff, H.; Kelvin, M.; Michael, M. *Hands-on Start to Wolfram Mathematica and Programming with the Wolfram Language*, 2nd ed.; Wolfram Media, Inc.: Champaign, IL, USA, 2016; ISBN 9781579550127.
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. Soleymani, F.; Shateyi, S. Two optimal eighth-order derivative-free classes of iterative methods. *Abstr. Appl. Anal.* **2012**, *2012*, 318165. [CrossRef]
23. Zaka, M.U.; Kosari, S.; Soleymani, F.; Khaksar, F.H.; Al-Fhaid, A.S. A super-fast tri-parametric iterative method with memory. *Appl. Math. Comput.* **2016**, *289*, 486–491.