An Efficient Derivative Free One-Point Method with Memory for Solving Nonlinear Equations

Janak Raj Sharma¹, Sunil Kumar¹ and Clemente Cesarano²,*

¹ Department of Mathematics, Sant Longowal Institute of Engineering and Technology, Longowal, Sangrur 148106, India
² Section of Mathematics, International Telematic University UNINETTUNO, Corso Vittorio Emanuele II, 39, 00186 Roma, Italy
* Correspondence: c.cesarano@uninettunouniversity.net

Received: 5 June 2019; Accepted: 4 July 2019; Published: 6 July 2019

Abstract: We propose a derivative free one-point method with memory of order 1.84 for solving nonlinear equations. The formula requires only one function evaluation and, therefore, the efficiency index is also 1.84. The methodology is carried out by approximating the derivative in Newton’s iteration using a rational linear function. Unlike the existing methods of a similar nature, the scheme of the new method is easy to remember and can also be implemented for systems of nonlinear equations. The applicability of the method is demonstrated on some practical as well as academic problems of a scalar and multi-dimensional nature. In addition, to check the efficacy of the new technique, a comparison of its performance with the existing techniques of the same order is also provided.

Keywords: nonlinear equations; iteration methods; one-point methods; order of convergence

MSC: 65H05; 65H10; 41A58

1. Introduction

In this study, we consider the problem of solving the nonlinear equations \( F(x) = 0 \); wherein \( F : D \subset \mathbb{R}^m \to \mathbb{R}^m \) is a univariate function when \( m = 1 \) or multivariate function when \( m > 1 \) on an open domain \( D \), by iterative methods. Univariate function is usually denoted by \( f(x) \).

Newton’s method [1–3] is one of the basic one-point methods which has quadratic convergence and requires one function and one derivative evaluation per iteration but it may diverge if the derivative is very small or zero. To overcome this problem, researchers have also proposed some derivative free one-point methods, for example, the Secant method [2], the Traub method [2], the Muller method [4,5], the Jarratt and Nudds method [6] and the Sharma method [7]. These methods are classified as one-point methods with memory whereas Newton’s method is a one-point method without memory (see Reference [2]). All the above mentioned one-point methods with memory require one function evaluation per iteration and possess order of convergence 1.84 except Secant which has order 1.62.

In this paper, we develop a new efficient one-point method with memory of convergence order 1.84 by using rational linear interpolation. The method consists of deriving the coefficients of a rational function that goes through by three points. Then, the derived coefficients are substituted into the derivative of the considered rational function which, when used in Newton’s scheme, gives the new scheme. The formula uses one function evaluation per step and has an efficiency index equal to the efficiency of the aforementioned methods of the same order. However, the main advantages of new method over the existing ones are its simplicity and suitability to solve systems of nonlinear equations.
The contents of the paper are organized as follows: in Section 2, the new method is developed and its convergence is discussed; in Section 3, some numerical examples are considered to verify the theoretical results and to compare the performance of proposed technique with existing techniques; the proposed method is generalized for solving the system of nonlinear equations in Section 4.

2. The Method and Its Convergence

Our aim is to develop a derivative-free iterative method by Newton’s scheme

\[ x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 0, 1, 2, \ldots, \]

wherein \( f'(x_n) \) is approximated by using the rational linear function

\[ R(t) = \frac{t - x_n + a}{b(t - x_n) + c}, \]

such that

\[ R(x_{n-2}) = f(x_{n-2}), \quad R(x_{n-1}) = f(x_{n-1}), \quad R(x_n) = f(x_n), \quad n = 2, 3, 4, \ldots. \]

Imposing the conditions (3) in (2), we have that

\[
\begin{align*}
(b(x_{n-2} - x_n) + c) f(x_{n-2}) &= x_{n-2} - x_n + a, \\
(b(x_{n-1} - x_n) + c) f(x_{n-1}) &= x_{n-1} - x_n + a, \\
 cf(x_n) &= a.
\end{align*}
\]

Then

\[
\begin{align*}
b(x_{n-2} - x_n) f(x_{n-2}) + c(f(x_{n-2}) - f(x_n)) &= x_{n-2} - x_n, \\
b(x_{n-1} - x_n) f(x_{n-1}) + c(f(x_{n-1}) - f(x_n)) &= x_{n-1} - x_n,
\end{align*}
\]

equivalently

\[
\begin{align*}
b f(x_{n-2}) + c f[x_{n-2}, x_n] &= 1, \\
b f(x_{n-1}) + c f[x_{n-1}, x_n] &= 1,
\end{align*}
\]

where \( f[s, t] = \frac{f(s) - f(t)}{s - t} \) is the Newton first order divided difference.

Solving for \( b \) and \( c \), we obtain that

\[
b = \frac{f[x_{n-2}, x_n] - f[x_{n-1}, x_n]}{f[x_{n-2}, x_n] f(x_{n-1}) - f[x_{n-1}, x_n] f(x_{n-2})},
\]

and

\[
c = \frac{f(x_{n-2}) - f(x_{n-1})}{f[x_{n-1}, x_n] f(x_{n-2}) - f[x_{n-2}, x_n] f(x_{n-1})}.
\]

Some simple calculations yield

\[
R'(x_n) = \frac{1 - b f(x_n)}{c} = \frac{f[x_{n-1}, x_n] f[x_{n-2}, x_n]}{f[x_{n-2}, x_{n-1}]}.
\]
Assuming that \( f'(x_0) \) is approximately equal to \( R'(x_0) \), then, in view of (9) the method (1) can be presented as

\[
x_{n+1} = x_n - \frac{f[x_{n-2},x_{n-1}]}{f[x_{n-1},x_n]} f(x_n).
\]

(10)

The scheme (10) defines a one-point method with memory and requires one function evaluation per iteration.

In the following theorem, we shall find the order of convergence of (10). We use the concept of \( R \)-order of convergence given by Ortega and Rheinboldt [8]. Suppose \( \{x_n\} \) is a sequence of approximation generated by an iteration method. If the sequence converges to a zero \( \alpha \) of \( f \) with \( R \)-order \( \geq r \), then we write

\[
e_{n+1} \sim e_n^r.
\]

(11)

**Theorem 1.** Suppose that \( f(x), f'(x), f''(x) \) and \( f'''(x) \) are continuous in the neighborhood \( \mathbb{D} \) of a zero (say, \( \alpha \)) of \( f \). If the initial approximations \( x_0, x_1 \) and \( x_2 \) are sufficiently close to \( \alpha \), then the \( R \)-order of convergence of the method (10) is 1.84.

**Proof.** Let \( e_n = x_n - \alpha, e_{n-1} = x_{n-1} - \alpha \) and \( e_{n-2} = x_{n-2} - \alpha \) be the errors in the \( n \)-th, \( n-1 \)-th and \( n-2 \)-th iterations, respectively. Using Taylor’s expansions of \( f(x_n), f(x_{n-1}) \) and \( f(x_{n-2}) \) about \( \alpha \) and taking into account that \( f(\alpha) = 0 \) and \( f'(\alpha) \neq 0 \), we have that

\[
f(x_n) = f'(\alpha)[e_n + A_2e_n^2 + A_3e_n^3 + \cdots],
\]

(12)

\[
f(x_{n-1}) = f'(\alpha)[e_{n-1} + A_2e_{n-1}^2 + A_3e_{n-1}^3 + \cdots],
\]

(13)

\[
f(x_{n-2}) = f'(\alpha)[e_{n-2} + A_2e_{n-2}^2 + A_3e_{n-2}^3 + \cdots],
\]

(14)

where \( A_1 = 1 \) and \( A_i = (1/i!)f^{(i)}(\alpha)/f'(\alpha), \ i = 2, 3, 4, \ldots \)

Using Equations (12) and (13), we have

\[
f[x_{n-1}, x_n] = f'(\alpha)(1 + A_2(e_n + e_{n-1}) + A_3(e_n^2 + e_{n-1}^2 + e_ne_{n-1}) + \cdots).
\]

(15)

Similarly we can obtain

\[
f[x_{n-2}, x_{n-1}] = f'(\alpha)(1 + A_2(e_{n-2} + e_{n-1}) + A_3(e_{n-2}^2 + e_{n-1}^2 + e_{n-2}e_{n-1}) + \cdots).
\]

(16)

\[
f[x_{n-3}, x_{n-2}] = f'(\alpha)(1 + A_2(e_{n-3} + e_{n-2}) + A_3(e_{n-3}^2 + e_{n-2}^2 + e_{n-3}e_{n-2}) + \cdots).
\]

(17)

Using Equations (12), (15)–(17) in (10), we obtain that

\[
e_{n+1} = e_n - \frac{e_n + A_2e_n^2 + A_3e_n^3 + A_2(e_{n-2}e_n + e_{n-1}e_n) + A_3(e_ne_{n-1}e_{n-2})}{1 + A_2(e_n + e_{n-2}) + A_2(e_n + e_{n-1}) + A_2^2(e_n^2 + e_ne_{n-1} + e_{n-1}e_n) + \cdots}.
\]

that is

\[
e_{n+1} \sim e_ne_{n-1}e_{n-2}.
\]

(18)

From (11), we have that

\[
e_n \sim e_{n+1}^{\frac{1}{2}},
\]

(19)

\[
e_{n-1} \sim e_n^{\frac{1}{2}}
\]

(20)

and

\[
e_{n-2} \sim e_{n-1}^{\frac{1}{2}} \sim e_n^{\frac{1}{2}}.
\]

(21)
Combining (18), (20) and (21), it follows that
\[ e_{n+1} \sim e_n^{\frac{1}{2} + \frac{1}{2^n}} = e_n^{r + \frac{1}{2}}. \] (22)

Comparison of the exponents of \( e_n \) on the right hand side of (11) and (22) leads to
\[ r^3 - r^2 - r = 0, \]
which has a unique positive real root 1.84. That means the order \( r \) of method (10) is 1.84. \( \Box \)

Remark 1. According to Ostrowski’s formula [9] for the efficiency measure of an iterative method of order \( r \); if \( c \) is the computational cost measured in terms of the number of evaluations of the function \( f \) and its derivatives that are required for each iteration, then the efficiency index of the method is given by \( r^{1/c} \). Thus the efficiency index of Newton’s method is 1.414, the Secant method is 1.62 whereas in the case of the Muller, Jarratt-Nudds, Traub, Sharma and new methods (10) this index is 1.84.

3. Numerical Results

We check the performance of the new method (10), now denoted by NM, using the computational software package Mathematica [10] with multiple-precision arithmetic. For comparison purposes, we consider the Muller method (MM), the Traub method (TM), the Jarratt-Nudds method (JNM) and the Sharma method (SM). These methods are given as follows:

**Muller method (MM):**
\[ x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{2a_2}{a_1 \pm \sqrt{a_1^2 - 4a_0a_2}}, \]
where
\[ a_0 = \frac{1}{D} [(x_n - x_{n-2})(f(x_n) - f(x_{n-1}))(x_n - x_{n-1})(f(x_n) - f(x_{n-2}))], \]
\[ a_1 = \frac{1}{D} [(x_n - x_{n-2})^2(f(x_n) - f(x_{n-1}))(x_n - x_{n-1})^2(f(x_n) - f(x_{n-2}))], \]
\[ a_2 = f(x_n), \]
\[ D = (x_n - x_{n-1})(x_n - x_{n-2})(x_{n-1} - x_{n-2}). \]

**Traub method (TM):**
\[ x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n) - f'(x_{n-1})} = x_n - \frac{f(x_n)f(x_{n-1})}{f(x_n) - f(x_{n-2})} \left( \frac{1}{f(x_n, x_{n-1})} - \frac{1}{f(x_{n-1}, x_{n-2})} \right). \]

**Jarratt-Nudds method (JNM):**
\[ x_{n+1} = x_n + \frac{(x_n - x_{n-1})(x_n - x_{n-2})f(x_n)(f(x_{n-1}) - f(x_{n-2}))}{(x_n - x_{n-1})(f(x_{n-2}) - f(x_{n}))f(x_{n-1}) + (x_n - x_{n-2})(f(x_{n}) - f(x_{n-1}))f(x_{n-2})}. \]

**Sharma method (SM):**
\[ x_{n+1} = x_n - \frac{2f(x_n)(b f(x_n) - d)}{c \pm \sqrt{c^2 - 4af(x_n)(b f(x_n) - d)}}, \]
where
\[ c = \frac{a(h_1 \delta_2 - h_2 \delta_1) + b \delta_1 \delta_2 (h_1 \delta_1 - h_2 \delta_2)}{\delta_2 - \delta_1}, \]
\[ d = \frac{a(h_2 - h_1) + b (h_2 \delta_2^2 - h_1 \delta_1^2)}{\delta_2 - \delta_1}, \]
\[ h_k = x_n - x_{n-k}, \]
\[ \delta_k = \frac{f(x_n) - f(x_{n-k})}{x_n - x_{n-k}}, \quad k = 1, 2. \]

We consider five examples for numerical tests as follows:

**Example 1.** Let us consider Kepler's equation; \( f_1(x) = x - \alpha_1 \sin(x) - K = 0 \), where \( 0 \leq \alpha_1 < 1 \) and \( 0 \leq K \leq \pi \). A numerical study, based on different values of parameters \( K \) and \( \alpha_1 \), has been performed in [11]. We solve the equation taking \( K = 0.1 \) and \( \alpha_1 = 0.25 \). For this set of values the solution \( \alpha \) is \( 0.13320215082857313... \). The numerical results are shown in Table 1.

**Table 1.** Comparison of performance of methods for function \( f_1(x) \), taking \( x_0 = 0.5, x_1 = -0.3 \), \( x_2 = 0.1 \).

| Methods | \( n \) | \( |x_4 - x_3| \) | \( |x_5 - x_4| \) | \( |x_6 - x_5| \) | COC | CPU-Time |
|---------|-------|----------------|----------------|----------------|-----|----------|
| MM      | 7     | 2.860 (−4)     | 2.271 (−7)     | 1.184 (−13)   | 1.82| 0.0952   |
| TM      | 7     | 2.832 (−4)     | 2.247 (−7)     | 1.143 (−13)   | 1.82| 0.0944   |
| JNM     | 7     | 2.846 (−4)     | 2.259 (−7)     | 1.163 (−13)   | 1.82| 0.1246   |
| SM \( (a = 1, b = 1) \) | 7 | 2.850 (−4) | 2.262 (−7) | 1.169 (−13) | 1.82 | 0.1107 |
| SM \( (a = 1, b = 2) \) | 7 | 2.845 (−4) | 2.258 (−7) | 1.162 (−13) | 1.82 | 0.0973 |
| SM \( (a = 1, b = −1) \) | 7 | 2.897 (−4) | 2.302 (−7) | 1.239 (−13) | 1.82 | 0.0984 |
| NM      | 7     | 2.670 (−4)     | 2.116 (−7)     | 1.013 (−13)   | 1.82| 0.0921   |

**Example 2.** Next, we consider isentropic supersonic flow across a sharp expansion corner (see Reference [12]). The relationship between the Mach number before the corner \( (i.e., M_1) \) and after the corner \( (i.e., M_2) \) is expressed by

\[ f_2(x) = b^{1/2} \left( \tan^{-1} \left( \frac{M_1^2-1}{b} \right)^{1/2} - \tan^{-1} \left( \frac{M_1^2-1}{b} \right)^{1/2} \right) - \left( \tan^{-1} (M_2^2-1)^{1/2} - \tan^{-1} (M_1^2-1)^{1/2} \right) - \delta, \]

where \( b = \frac{\gamma+1}{\gamma-1} \) and \( \gamma \) is the specific heat ratio of the gas. We take values \( M_1 = 1.5, \gamma = 1.4 \) and \( \delta = 10^0 \). The solution \( \alpha \) of this problem is \( 1.8411294068501996... \). The numerical results are shown in Table 2.

**Table 2.** Comparison of performance of methods for function \( f_2(x) \), taking \( x_0 = 1.1, x_1 = 2.3, x_2 = 1.7 \).

| Methods | \( n \) | \( |x_4 - x_3| \) | \( |x_5 - x_4| \) | \( |x_6 - x_5| \) | COC | CPU-Time |
|---------|-------|----------------|----------------|----------------|-----|----------|
| MM      | 7     | 8.212 (−3)     | 3.223 (−5)     | 3.369 (−9)     | 1.83| 0.3312   |
| TM      | 7     | 8.228 (−3)     | 4.906 (−5)     | 6.104 (−9)     | 1.83| 0.3434   |
| JNM     | 7     | 8.220 (−3)     | 4.048 (−5)     | 4.636 (−9)     | 1.83| 0.3163   |
| SM \( (a = 1, b = 1) \) | 7 | 8.215 (−3) | 3.537 (−5) | 3.841 (−9) | 1.83 | 0.3754 |
| SM \( (a = 1, b = 2) \) | 7 | 8.217 (−3) | 3.752 (−5) | 4.175 (−9) | 1.83 | 0.3666 |
| SM \( (a = 1, b = −1) \) | 7 | 8.207 (−3) | 2.724 (−5) | 2.660 (−9) | 1.83 | 0.3627 |
| NM      | 7     | 8.228 (−3)     | 4.905 (−5)     | 5.395 (−9)     | 1.83| 0.3024   |
Example 3. Consider the equation governing the L-C-R circuit in electrical engineering [13]

\[ L \frac{d^2 q}{dt^2} + R \frac{dq}{dt} + \frac{q}{C} = 0, \]

whose solution \( q(t) \) is

\[ q(t) = q_0 e^{-\frac{Rt}{2L}} \cos \left( \sqrt{\frac{1}{LC} - \left( \frac{R^2}{2L} \right)^2} t \right), \]

where at \( t = 0, q = q_0 \).

A particular problem as a case study is given as: Assuming that the charge is dissipated to 1 percent of its original value \( (q/q_0 = 0.01) \) in \( t = 0.05 \) s, with \( L = 5 \) Henry and \( C = 10^{-4} \) Farad. Determine the proper value of \( R \)?

Using the given numerical data, the problem is given as

\[ f_3(x) = e^{-0.005x} \cos \left( \sqrt{2000 - 0.01x^2} (0.05) \right) - 0.01 = 0, \]

where \( x = R \). Solution of this problem is, \( a = 328.15142908514817 \ldots \) Numerical results are displayed in Table 3.

**Table 3.** Comparison of performance of methods for function \( f_3(x) \), taking \( x_0 = 430, x_1 = 200, x_2 = 315. \)

| Methods  | \( n \) | \( |x_4 - x_3| \) | \( |x_5 - x_4| \) | \( |x_6 - x_5| \) | COC | CPU-Time |
|----------|--------|----------------|----------------|----------------|-----|---------|
| MM       | 7      | 4.446(−1)      | 2.182(−3)      | 3.303(−8)      | 1.83 | 0.2075  |
| TM       | 7      | 6.515(−1)      | 4.078(−3)      | 1.381(−7)      | 1.84 | 0.2185  |
| JNM      | 7      | 1.259(−1)      | 1.767(−4)      | 2.058(−10)     | 1.83 | 0.1721  |
| SM \((a = 1, b = 1)\) | 7 | 4.446(−1) | 2.182(−3) | 3.303(−8) | 1.83 | 0.2126 |
| SM \((a = 1, b = 2)\) | 7 | 4.446(−1) | 2.182(−3) | 3.303(−8) | 1.83 | 0.1979 |
| SM \((a = 1, b = −1)\) | 7 | 4.446(−1) | 2.182(−3) | 3.303(−8) | 1.83 | 0.2034 |
| NM       | 7      | 1.818(−1)      | 3.112(−4)      | 6.976(−10)     | 1.83 | 0.1568  |

Example 4. Law of population growth is given as (see References [14,15])

\[ \frac{dN(t)}{dt} = \lambda N(t) + v, \]

where \( N(t) = \) population at time \( t, \lambda = \) constant birth rate of population and \( v = \) constant immigration rate.

The solution of this differential equation is given by

\[ N(t) = N_0 e^{\lambda t} + \frac{v}{\lambda} (e^{\lambda t} - 1), \]

where \( N_0 \) is initial population.

A particular problem for the above model can be formulated as: Suppose that a certain population consists of 1,000,000 people initially. Further suppose that 435,000 people immigrate into the community in the first year and 1,564,000 people are present at the end of one year. Determine the birth rate \( (\lambda) \) of this population.

To find the birth rate, we will solve the equation

\[ f_4(x) = 1564 - 1000 e^x - \frac{435}{x} (e^x - 1) = 0, \]
wherein \( x = \lambda \). Solution of this problem is, \( \alpha = 0.10099792968574979 \ldots \). The numerical results are given in Table 4.

### Table 4. Comparison of performance of methods for function \( f_4(x) \), taking \( x_0 = -0.4, x_1 = 0.5, x_2 = 0.05 \).

| Methods | \( n \) | \( |x_4 - x_3| \) | \( |x_5 - x_4| \) | \( |x_6 - x_5| \) | COC | CPU-Time |
|---------|------|--------------|--------------|--------------|-----|---------|
| MM      | 7    | 1.464(−3)    | 4.958(−6)    | 5.588(−11)   | 1.83| 0.1924  |
| TM      | 7    | 3.061(−3)    | 1.670(−5)    | 7.659(−10)   | 1.84| 0.1884  |
| JNM     | 7    | 7.093(−4)    | 1.013(−6)    | 2.558(−12)   | 1.82| 0.1811  |
| SM \((a = 1, b = 1)\) | 7    | 3.061(−3)    | 1.670(−5)    | 7.659(−10)   | 1.84| 0.2194  |
| SM \((a = 1, b = 2)\) | 7    | 3.061(−3)    | 1.670(−5)    | 7.659(−10)   | 1.84| 0.2033  |
| SM \((a = 1, b = −1)\) | 7    | 3.061(−3)    | 1.670(−5)    | 7.659(−10)   | 1.84| 0.1875  |
| NM      | 7    | 1.980(−3)    | 1.078(−6)    | 8.160(−12)   | 1.85| 0.1727  |

**Example 5.** Next, we consider an example of academic interest, which is defined by

\[
|f_5(x)| = \begin{cases} 
 x^3 \ln x^2 + x^5 - x^4, & x \neq 0, \\
 0, & x = 0.
\end{cases}
\]

It has three zeros. Note that \( \alpha = 0 \) is the multiple zero of multiplicity 3. We consider the zero \( \alpha = 1 \) in our work. Numerical results are displayed in Table 5.

### Table 5. Comparison of performance of methods for function \( f_5(x) \), taking \( x_0 = 1.2, x_1 = 0.9, x_2 = 1.05 \).

| Methods | \( n \) | \( |x_4 - x_3| \) | \( |x_5 - x_4| \) | \( |x_6 - x_5| \) | COC | CPU-Time |
|---------|------|--------------|--------------|--------------|-----|---------|
| MM      | 8    | 3.206(−3)    | 5.057(−5)    | 2.720(−8)    | 1.84| 0.0943  |
| TM      | 9    | 1.090(−2)    | 9.025(−4)    | 5.898(−6)    | 1.83| 0.0821  |
| JNM     | 8    | 4.915(−3)    | 1.525(−4)    | 1.955(−7)    | 1.85| 0.0798  |
| SM \((a = 1, b = 1)\) | 8    | 1.010(−2)    | 7.066(−4)    | 3.901(−6)    | 1.85| 0.0942  |
| SM \((a = 1, b = 2)\) | 8    | 1.048(−2)    | 7.960(−4)    | 4.777(−6)    | 1.83| 0.0933  |
| SM \((a = 1, b = −1)\) | 9    | 1.185(−2)    | 1.188(−3)    | 9.274(−6)    | 1.83| 0.0931  |
| NM      | 8    | 1.930(−3)    | 4.728(−5)    | 1.766(−8)    | 1.85| 0.0775  |

Numerical results shown in Tables 1–5 contain the required iterations \( n \), computed estimated error \( |x_{n+1} - x_n| \) in first three iterations (wherein \( \lambda \) denotes \( \lambda \times 10^{-6} \)), computational order of convergence (COC) and CPU time (CPU-time) are measured during the execution of the program. Computational order of convergence (COC) is computed by using the formula [16]

\[
\text{COC} = \frac{\ln(|x_{n+1} - x_n|/|x_n - x_{n-1}|)}{\ln(|x_n - x_{n-1}|/|x_{n-1} - x_{n-2}|)}.
\]

The necessary iterations \( n \) are obtained so as to satisfy the criterion \( |x_{n+1} - x_n| + |f(x_n)| < 10^{-100} \). The first two initial approximations \( x_0 \) and \( x_1 \) are chosen arbitrarily, whereas third \( x_2 \) is taken as the average of these two. From the numerical results displayed in Tables 1–5, we can conclude that the accuracy of the new method (NM) is either equal to or better than existing methods. Moreover, it requires less CPU-time compared with that of existing methods. This character makes it more efficient than the existing ones.
4. Generalized Method

We end this work with a method for solving a system of nonlinear equations \( F(x) = 0; F : \mathbb{D} \subset \mathbb{R}^m \rightarrow \mathbb{R}^m \) is the given nonlinear function \( F = (f_1, f_2, \ldots, f_m)^T \) and \( x = (x_1, \ldots, x_m)^T \). The divided difference \( F[x, y] \) of \( F \) is a matrix of order \( m \times m \) (see Reference [2], p. 229) with elements

\[
F[x, y]_{ij} = \frac{f_i(x_1, \ldots, x_j, y_{j+1}, \ldots, y_m) - f_i(x_1, \ldots, x_j, y_j, \ldots, y_m)}{x_j - y_j}, \quad 1 \leq i, j \leq m. \tag{23}
\]

Keeping in mind (10), we can write the corresponding method for the system of nonlinear equations as:

\[
x^{(n+1)} = x^{(n)} - F[x^{(n-1)}, x^{(n)}]^{-1} F[x^{(n-2)}, x^{(n-1)}] F[x^{(n-2)}, x^{(n)}]^{-1} F(x^{(n)}), \tag{24}
\]

where \( F[\cdot, \cdot]^{-1} \) is the inverse of the divided difference operator \( F[\cdot, \cdot] \).

**Remark 2.** The computational efficiency of an iterative method for solving the system \( F(x) = 0 \) is calculated by the efficiency index \( E = r^{1/C} \), (see Reference [17]), where \( r \) is the order of convergence and \( C \) is the total cost of computation. The cost of computation \( C \) is measured in terms of the total number of function evaluations per iteration and the number of operations (that means products and divisions) per iteration. The various evaluations and operations that contribute to the cost of computation are described as follows. For the computation of \( F \) in any iterative function we evaluate \( m \) scalar functions \( f_i \), \( 1 \leq i \leq m \) and when computing a divided difference \( F[x, y] \), we evaluate \( m(m - 1) \) scalar functions, wherein \( F(x) \) and \( F(y) \) are evaluated separately. Furthermore, one has to add \( m^2 \) divisions from any divided difference. For the computation of an inverse linear operator, a linear system can be solved that requires \( m(m - 1)(2m - 1)/6 \) products and \( m(m - 1)/2 \) divisions in the LU decomposition process, and \( m(m - 1) \) products and \( m \) divisions in the solution of two triangular linear systems. Thus, taking into account the above considerations of evaluations and operations for the method (24), we have that

\[
C = \frac{2}{3} m^3 + 8 m^2 - \frac{8}{3} m \quad \text{and} \quad E = 1.84 \sqrt[3]{\frac{2}{m^3 + 24m^2 - 8m}}.
\]

Next, we apply the generalized method on the following problems:

**Example 6.** The following system of \( m \) equations (selected from [18]) is considered:

\[
\sum_{j=1, j \neq i}^{m} x_j - e^{-x_i} = 0, \quad 1 \leq i \leq m.
\]

In particular, we solve this problem for \( m = 10, 30, 50, 100 \) by selecting initial guesses \( x^{(0)} = \{2, 2, \ldots, 2\}^T \), \( x^{(1)} = \{-1, -1, \ldots, -1\}^T \) and \( x^{(2)} = \{\frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2}\}^T \) towards the corresponding solution:

\[
a = \{0.100488400337, 0.100488400337, \ldots, 0.100488400337\}^T, \\
a = \{0.033351667835, 0.033351667835, \ldots, 0.033351667835\}^T, \\
a = \{0.020003975040, 0.020003975040, \ldots, 0.020003975040\}^T, \\
a = \{0.010000498387, 0.010000498387, \ldots, 0.010000498387\}^T.
\]

Numerical results are displayed in Table 6.
Table 6. Performance of new method (NM) for Example 6.

<table>
<thead>
<tr>
<th>m</th>
<th>n</th>
<th>|x_4 - x_3|</th>
<th>|x_5 - x_4|</th>
<th>|x_6 - x_5|</th>
<th>COC</th>
<th>CPU-Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>7</td>
<td>1.139(-2)</td>
<td>7.395(-5)</td>
<td>1.310(-9)</td>
<td>1.84</td>
<td>1.935</td>
</tr>
<tr>
<td>30</td>
<td>7</td>
<td>6.776(-3)</td>
<td>1.937(-5)</td>
<td>5.181(-11)</td>
<td>1.84</td>
<td>16.832</td>
</tr>
<tr>
<td>50</td>
<td>7</td>
<td>5.251(-3)</td>
<td>9.485(-6)</td>
<td>9.630(-12)</td>
<td>1.84</td>
<td>57.704</td>
</tr>
<tr>
<td>100</td>
<td>7</td>
<td>3.691(-3)</td>
<td>3.463(-6)</td>
<td>9.131(-13)</td>
<td>1.84</td>
<td>407.912</td>
</tr>
</tbody>
</table>

Example 7. Consider the system of m equations (selected from Reference [19]):

\[
\tan^{-1}(x_i) + 1 - 2 \sum_{j=1, j \neq i}^{m} x_j^2 = 0, \quad 1 \leq i \leq m.
\]

Let us solve this problem for \(m = 10, 30, 50, 100\) with initial values \(x^{(0)} = \{-1, -1, \ldots, -1\}^T\), \(x^{(1)} = \{0, 0, \cdots, 0\}^T\), \(x^{(2)} = \{-0.5, -0.5, \cdots, -0.5\}^T\) towards the corresponding solutions:

\[
\alpha = \{-0.209906976944 \ldots, -0.209906976944 \ldots, -0.209906976944 \ldots\}^T,
\]

\[
\alpha = \{-0.123008700800 \ldots, -0.123008700800 \ldots, -0.123008700800 \ldots\}^T,
\]

\[
\alpha = \{-0.068590313107 \ldots, -0.068590313107 \ldots, -0.068590313107 \ldots\}^T.
\]

Numerical results are displayed in Table 7.

Table 7. Performance of new method (NM) for Example 7.

<table>
<thead>
<tr>
<th>m</th>
<th>n</th>
<th>|x_4 - x_3|</th>
<th>|x_5 - x_4|</th>
<th>|x_6 - x_5|</th>
<th>COC</th>
<th>CPU-Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>9</td>
<td>7.661(-2)</td>
<td>1.423(-2)</td>
<td>1.304(-4)</td>
<td>1.84</td>
<td>3.386</td>
</tr>
<tr>
<td>30</td>
<td>10</td>
<td>4.195(-1)</td>
<td>5.623(-2)</td>
<td>6.824(-3)</td>
<td>1.84</td>
<td>25.600</td>
</tr>
<tr>
<td>50</td>
<td>10</td>
<td>6.603(-1)</td>
<td>5.572(-2)</td>
<td>1.354(-2)</td>
<td>1.84</td>
<td>87.531</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
<td>1.076</td>
<td>2.307(-2)</td>
<td>1.106(-2)</td>
<td>1.84</td>
<td>593.691</td>
</tr>
</tbody>
</table>

In Tables 6 and 7 we have shown the results of the new method only, because the other methods are not applicable for nonlinear systems. We conclude that there are numerous one-point iterative methods for solving a scalar equation \(f(x) = 0\). Contrary to this fact, such methods are rare for multi-dimensional cases, that is, for approximating the solution of \(F(x) = 0\). Since the method uses first divided difference, a drawback of the method is that if at some stage (say \(j\)) the denominator \(x_j = y_j\) in the Formula (23), then the method may fail to converge. However, this situation is rare since we have applied the method successfully on many other different problems. In the present work, an attempt has been made to develop an iterative scheme which is equally suitable for both categories viz. univariate and multivariate functions.

Author Contributions: The contribution of all the authors has been equal. All of them have worked together to prepare the manuscript.

Funding: This research received no external funding.

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

References
3. Kumar, D.; Sharma, J.R.; Cesarano, C. An efficient class of Traub-Steffensen-type methods for computing multiple zeros. *Axioms* 2019, 8, 65. [CrossRef]


© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).