Calcolo (2016) 53:451–473
DOI 10.1007/s10092-015-0157-9

Simple yet efficient Newton-like method for systems of nonlinear equations

Janak Raj Sharma[1] · **Rangan K. Guha**[1]

Received: 25 March 2015 / Accepted: 23 September 2015 / Published online: 13 October 2015

Abstract We present a three-step iterative method of convergence order five for solving systems of nonlinear equations. The methodology is based on Newton's and Newton-like iterations. Hence, the name Newton-like method. Computational efficiency of the new method is considered and compared with well-known existing methods. Numerical tests are performed on some problems of different nature, which confirm robust and efficient convergence behavior of the proposed technique. Moreover, theoretical results concerning order of convergence and computational efficiency are verified in the numerical problems. It is shown that, in general, the new method is more efficient than the existing counterparts.

Keywords Nonlinear equations · Newton's method · Multipoint methods · Order of convergence · Computational efficiency

Mathematics Subject Classification 65H10 · 65Y20 · 41A58

1 Introduction

The construction of fixed point methods for solving nonlinear equations and systems of nonlinear equations is an interesting and challenging task in numerical analysis and many applied scientific branches. An immense importance of this topic has led

✉ Janak Raj Sharma
jrshira@yahoo.co.in

Rangan K. Guha
rangankguha@yahoo.com

[1] Department of Mathematics, Sant Longowal Institute of Engineering and Technology, Sangrur, Longowal 148106, India

to the development of many numerical methods, most frequently of iterative nature (see [1,16,20,22,26]). With the advancement of computer hardware and software, the problem of solving nonlinear equations by numerical methods has gained an additional importance. In this paper, we consider the problem of finding solution of the system of nonlinear equations $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ by iterative methods of a high order of convergence. This problem can be precisely stated as to find a vector $\mathbf{r} = (r_1, r_2, \ldots, r_n)^T$ such that $\mathbf{F}(\mathbf{r}) = \mathbf{0}$, where $\mathbf{F} : \mathrm{D} \subseteq \mathbb{R}^n \to \mathbb{R}^n$ is the given nonlinear vector function $\mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_n(\mathbf{x}))^T$ and $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$. The solution vector $\mathbf{r}$ of $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ can be obtained as a fixed point of some function $\phi : \mathbb{R}^n \to \mathbb{R}^n$ by means of the fixed point iteration

$$\mathbf{x}^{(k+1)} = \phi(\mathbf{x}^{(k)}), \quad k = 0, 1, 2, \ldots.$$

One of the basic procedures for solving systems of nonlinear equations is the quadratically convergent Newton's method (see [16,20,26]), which is given as,

$$\mathbf{x}^{(k+1)} = \phi_1^{(2)}(\mathbf{x}^{(k)}) = \mathbf{x}^{(k)} - \mathbf{F}'(\mathbf{x}^{(k)})^{-1}\mathbf{F}(\mathbf{x}^{(k)}), \tag{1}$$

where $\mathbf{F}'(\mathbf{x})^{-1}$ is the inverse of first Fréchet derivative $\mathbf{F}'(\mathbf{x})$ of the function $\mathbf{F}(\mathbf{x})$. From the computational point of view the Newton's method requires the evaluations of one $\mathbf{F}$, one $\mathbf{F}'$ and one matrix inversion (i.e. inverse Fréchet derivative) per iteration. Throughout the paper, we use the abbreviation $\phi_i^{(p)}$ to denote an ith iterative function of convergence order p.

To improve the order of convergence of Newton's method, a number of higher order methods have been proposed in literature. For example, Cordero and Torregrosa [2], Frontini and Sormani [10], Grau et al. [11], Homeier [13], and Noor and Waseem [19] have developed third order methods each requiring one $\mathbf{F}$, two $\mathbf{F}'$ and two matrix inversions per iteration. Cordero and Torregrosa have also derived two third-order methods in [3]. One of the methods requires one $\mathbf{F}$ and three $\mathbf{F}'$ whereas other requires one $\mathbf{F}$ and four $\mathbf{F}'$ evaluations per iteration. Both the methods also require two matrix inversions in each iteration. Darvishi and Barati in [7], and Potra and Pták in [23] have proposed third order methods which use two $\mathbf{F}$, one F' and one matrix inversion. Cordero et al. developed a fourth order method in [4], which uses two $\mathbf{F}$, two $\mathbf{F}'$ and one matrix inversion. Cordero et al. in [5] have implemented fourth order Jarratt's method [14] for scalar equations to systems of equations which requires one $\mathbf{F}$, two $\mathbf{F}'$ and two matrix inversions. Darvishi and Barati [8] presented a fourth order method requiring two $\mathbf{F}$, three $\mathbf{F}'$ and two matrix inversions per iteration. Grau et al. presented a fourth order method in [11] utilizing three $\mathbf{F}$, one $\mathbf{F}'$ and one matrix inversion. Neta [18] proposed a fourth order method using three $\mathbf{F}$, one $\mathbf{F}'$ and one matrix inversion. Sharma et al. [24] developed a fourth order method requiring one $\mathbf{F}$, two $\mathbf{F}'$ and two matrix inversions.

In quest of more fast algorithms, researchers have also proposed fifth and sixth order methods in [4–6,11,25]. The fifth order methods by Cordero et al. [5,6] and Grau et al. [11] require four evaluations namely, two $\mathbf{F}$ and two $\mathbf{F}'$ per iteration. The fifth order method by Cordero et al. [4] requires three $\mathbf{F}$ and two $\mathbf{F}'$. In addition, the fifth order

method in [4] requires one matrix inversion, in [5,6] three and in [11] two matrix inversions. One sixth order method by Cordero et al. [5] uses two $\mathbf{F}$ and two $\mathbf{F}'$ while other sixth order method [6] uses three $\mathbf{F}$ and two $\mathbf{F}'$. The sixth order methods, apart from the mentioned evaluations, also require two matrix inversions per one iteration. Sharma and Gupta [25] proposed a fifth order method requiring two $\mathbf{F}$, two $\mathbf{F}'$ and two matrix inversions per one iteration.

The main goal of this paper is to develop iterative method of high computational efficiency, which may assume a high convergence order and low computational cost. To do so, we here propose a method with fifth order of convergence by employing the iterative scheme that utilizes the number of function evaluations and inverse operators as minimum as possible. In this way, we attain low computational cost and hence an increased computational efficiency. Moreover, we show that the proposed methods are efficient than existing methods in general.

Contents of the paper are summarized as follows. Some preliminary results are presented in Sect. 2. In Sect. 3, we describe the basic method for solving scalar equations. The method developed in Sect. 3 is generalized for systems of equations in Sect. 4. Here, the convergence behavior showing fifth order of convergence is also analyzed. Computational efficiency of the new method is studied and then compared with some well-known existing methods in Sect. 5. In Sect. 6, we present various numerical examples to confirm the theoretical results and to compare convergence properties of the proposed method with existing methods. Concluding remarks are given in Sect. 7.

2 Preliminary results

2.1 Order of convergence

Let $\{\mathbf{x}^{(k)}\}_{k\geqslant 0}$ be a sequence in $\mathbb{R}^n$ which converges to $\mathbf{r}$. Then, convergence is called of order p, $p > 1$, if there exists M, $M > 0$, and k_0 such that

$$\left|\left|\mathbf{x}^{(k+1)} - \mathbf{r}\right|\right| \leqslant M \left|\left|\mathbf{x}^{(k)} - \mathbf{r}\right|\right|^p \quad \forall \quad k \geqslant k_0$$

or

$$\left|\left|\mathbf{e}^{(k+1)}\right|\right| \leqslant M \left|\left|\mathbf{e}^{(k)}\right|\right|^p \quad \forall \quad k \geqslant k_0,$$

where $\mathbf{e}^{(k)} = \mathbf{x}^{(k)} - \mathbf{r}$. The convergence is called linear if $p = 1$ and there exists M such that $0 < M < 1$.

2.2 Error equation

Let $\mathbf{e}^{(k)} = \mathbf{x}^{(k)} - \mathbf{r}$ be the error in the kth iteration, we call the relation

$$\mathbf{e}^{(k+1)} = L(\mathbf{e}^{(k)})^p + O((\mathbf{e}^{(k)})^{p+1}),$$

as the error equation. Here, p is the order of convergence, L is a p-linear function, i.e. $L \in \mathcal{L}(\mathbb{R}^n \times \overset{p-times}{\cdots} \times \mathbb{R}^n, \mathbb{R}^n)$, $\mathcal{L}$ denotes the set of bounded linear functions.

2.3 Computational order of convergence

Let $\mathbf{x}^{(k-1)}$, $\mathbf{x}^{(k)}$ and $\mathbf{x}^{(k+1)}$ be the three consecutive iterations close to the zero $\mathbf{r}$ of $\mathbf{F}(\mathbf{x})$. Then, the computational order of convergence can be approximated using the formula (see [15,21])

$$\rho_k = \frac{\log\left(\left|\left|\mathbf{F}(\mathbf{x}^{(k)})\right|\right| / \left|\left|\mathbf{F}(\mathbf{x}^{(k-1)})\right|\right|\right)}{\log\left(\left|\left|\mathbf{F}(\mathbf{x}^{(k-1)})\right|\right| / \left|\left|\mathbf{F}(\mathbf{x}^{(k-2)})\right|\right|\right)}.$$

2.4 Computational efficiency

Computational efficiency of an iterative method is measured by the efficiency index $E = p^{1/C}$ (see [11]), where p is the order of convergence and C is the computational cost given by

$$C(\mu_0, \mu_1, n) = P_0(n)\mu_0 + P_1(n)\mu_1 + P(n).$$

Here, $P_0(n)$ represents the number of evaluations of scalar functions $(f_1, f_2, \ldots, f_n)$ used in the evaluations of $\mathbf{F}$, $P_1(n)$ is the number of evaluations of scalar functions of $\mathbf{F}'$, i.e. $\frac{\partial f_i}{\partial x_j}$, $1 \leqslant i, j \leqslant n$, $P(n)$ represents the number of products or quotients needed per iteration, and μ_0 and μ_1 are ratios between products and evaluations required to express the value of $C(\mu_0, \mu_1, n)$ in terms of products.

In case of iterative methods for scalar equations $f(x) = 0$, the cost C is measured as the number of new pieces of information required by the method per iterative step. A 'piece of information' typically is any evaluation of the function f or one of its derivatives.

3 Basic method

In what follows, we shall develop the scheme for solving scalar equation $f(x) = 0$, then based on this we shall state the generalized form for systems of nonlinear equations. Let us consider the scheme

$$\begin{aligned} y_k &= x_k - \frac{f(x_k)}{f'(x_k)}, \\ z_k &= y_k - a\frac{f(y_k)}{f'(x_k)}, \\ x_{k+1} &= y_k - b\frac{f(y_k)}{f'(x_k)} - c\frac{f(z_k)}{f'(x_k)}, \end{aligned} \tag{2}$$

where a, b and c are parameters to be determined from the following convergence theorem.

Theorem 1 *Let the function $f : I \to \mathbb{R}$ be a real valued function on I, where I is a neighborhood of a simple zero r of $f(x)$. Assume that $f(x)$ is sufficiently differentiable in I. If an initial approximation x_0 is sufficiently close to r, then the local order of convergence of method (2) is at least 5, if $a = 5,\ b = 9/5$ and $c = 1/5$.*

Proof Let $e_k = x_k - r$, $e_{y_k} = y_k - r$ and $e_{z_k} = z_k - r$ be the errors in the kth iteration. Using the fact that $f(r) = 0$, $f'(r) \neq 0$, we write the Taylor's series expansions of the functions $f(x_k)$, $f'(x_k)$, $f(y_k)$ and $f(z_k)$ about r as follows

$$f(x_k) = f'(r)\big(e_k + A_2e_k^2 + A_3e_k^3 + A_4e_k^4 + A_5e_k^5 + O(e_k^6)\big), \tag{3}$$

$$f'(x_k) = f'(r)\big(1 + 2A_2e_k + 3A_3e_k^2 + 4A_4e_k^3 + 5A_5e_k^4 + 6A_6e_k^5 + O(e_k^6)\big), \tag{4}$$

$$f(y_k) = f'(r)\big(e_{y_k} + A_2e_{y_k}^2 + A_3e_{y_k}^3 + A_4e_{y_k}^4 + A_5e_{y_k}^5 + O(e_{y_k}^6)\big), \tag{5}$$

$$f(z_k) = f'(r)\big(e_{z_k} + A_2e_{z_k}^2 + A_3e_{z_k}^3 + A_4e_{z_k}^4 + A_5e_{z_k}^5 + O(e_{z_k})^6\big), \tag{6}$$

where $A_j = (1/j!)f^{(j)}(r)/f'(r),\ j = 2, 3, 4, \ldots$

Let $e_{k+1} = x_{k+1} - r$ be the error in the $(k+1)$th iteration. Substituting equations (4), (5) and (6) in the last step of (2) and then simplifying, we obtain the error equation as

$$\begin{aligned} e_{k+1} = e_{y_k} - e_{y_kz_k} - A_2(be_{y_k}^2 + ce_{z_k}^2) + 2A_2\big(e_{y_kz_k} + A_2(be_{y_k}^2 + ce_{z_k}^2)\big)e_k - e_{y_kz_k}\big((4A_2^2 - 3A_3)e_k^2 \\ + 4(2A_2^3 - 3A_2A_3 + A_4)e_k^3 + O(e_k^4)\big), \end{aligned} \tag{7}$$

where $e_{y_kz_k} = be_{y_k} + ce_{z_k}$.

Substitution of (3) and (4) in the first step of (2) yields

$$\begin{aligned} e_{y_k} = A_2e_k^2 + (-2A_2^2 + 2A_3)e_k^3 + (4A_2^3 - 7A_2A_3 + 3A_4)e_k^4 \\ - 2(4A_2^4 - 10A_2^2A_3 + 3A_3^2 + 5A_2A_4 - 2A_5)e_k^5 + O(e_k)^6. \end{aligned} \tag{8}$$

Then using (4), (5) and (8) in the second step of (2), we get

$$\begin{aligned} e_{z_k} =& K_1A_2e_k^2 + 2(K_2A_2^2 + K_1A_3)e_k^3 + (K_3A_2^3 + 7K_2A_2A_3 + 3K_1A_4)e_k^4 \\ &+ 2(K_4A_2^4 - K_5A_2^2A_3 + 3K_2A_3^2 + 5K_2A_2A_4 + 2K_1A_5)e_k^5 + O(e_k)^6, \end{aligned} \tag{9}$$

where $K_1 = -a+1,\ K_2 = 2a-1,\ K_3 = -13a+4,\ K_4 = 19a-4,\ K_5 = 32a-10$. Combining (7), (8) and (9), it follows that

$$\begin{aligned} e_{k+1} =& L_1A_2e_k^2 - 2(L_2A_2^2 - L_1A_3)e_k^3 + (L_3A_2^3 - 7L_2A_2A_3 + 3L_1A_4)e_k^4 \\ &+ 2(L_4A_2^4 - 2L_5A_2^2A_3 - 3L_2A_3^2 - 5L_2A_2A_4 + 2L_1A_5)e_k^5 + O(e_k)^6, \end{aligned} \tag{10}$$

where $L_1 = (a-1)c - b + 1,\ L_2 = (3a-2)c - 2b + 1,\ L_3 = -(a^2 - 27a + 13)c + 13b - 4,\ L_4 = (5a^2 - 52a + 19)c + 19b - 4$ and $L_5 = (a^2 - 33a + 16)c + 16b - 5$.

In order to find the parameters a, b and c it will be sufficient to equate the coefficients of e_k^2, e_k^3 and e_k^4 to zero. Thus, we have the following system of equations:

$$\begin{aligned}(a-1)c - b + 1 &= 0,\\ (3a-2)c - 2b + 1 &= 0,\\ -(a^2 - 27a + 13)c + 13b - 4 &= 0.\end{aligned}$$

Solving the above system of equations, we get $a = 5$, $b = 9/5$ and $c = 1/5$. Putting these values of a, b and c in (10), the final error equation of the proposed scheme is given as

$$e_{k+1} = \left(14A_2^4 + 4A_2^2A_3\right)e_k^5 + O\left(e_k\right)^6. \tag{11}$$

This error equation shows that the order of convergence of method (2) is five, which completes the proof of Theorem 1. □

Remark 1 The proposed scheme is a multipoint method without memory, which is based on four function evaluations (namely three f and one f') and possesses fifth order convergence. Applying the Definition 2.4 of computational efficiency for scalar case, we have $E = 5^{1/4} \approx 1.495$. So, the efficiency is better than Newton's method ($E \approx 1.414$). However, according to Kung-Traub hypothesis [17] multipoint methods without memory based on n function evaluations can achieve order of convergence 2^{n-1}. For example, with four function evaluations a method of optimal eighth order convergence can be developed. In this case the efficiency is, $E \approx 1.682$. Therefore, the presented scheme for finding zero of a univariate function is not an efficient one.

4 Generalized method

The novel feature of the method (2) is its simple design which makes it easily implemented to systems of nonlinear equations. Moreover, the method may prove to be efficient for systems of equations. Solving a system of equations involves the computations such as evaluations of vector function, Fréchet derivative and its inverse, matrix multiplication and so on. Each of these evaluations requires a different amount of computational work. Among these the most expensive is the evaluation of inverse Fréchet derivative. For systems, therefore, the computational efficiency can not be measured by considering only the number of function and derivative evaluations. Keeping these facts in view, here our motive is to generalize the scheme (2) for solving systems of nonlinear equations.

Let us consider the problem of solving the system of equations $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ by an iterative method based on the scheme (2). Thus, writing the corresponding formula with $a = 5$, $b = 9/5$ and $c = 1/5$ for system of equations

$$\begin{aligned}
\mathbf{y}^{(k)} &= \mathbf{x}^{(k)} - \mathbf{F}'\left(\mathbf{x}^{(k)}\right)^{-1}\mathbf{F}\left(\mathbf{x}^{(k)}\right),\\
\mathbf{z}^{(k)} &= \mathbf{y}^{(k)} - 5\mathbf{F}'\left(\mathbf{x}^{(k)}\right)^{-1}\mathbf{F}\left(\mathbf{y}^{(k)}\right),\\
\mathbf{x}^{(k+1)} &= \mathbf{y}^{(k)} - \frac{9}{5}\mathbf{F}'\left(\mathbf{x}^{(k)}\right)^{-1}\mathbf{F}\left(\mathbf{y}^{(k)}\right) - \frac{1}{5}\mathbf{F}'\left(\mathbf{x}^{(k)}\right)^{-1}\mathbf{F}\left(\mathbf{z}^{(k)}\right). \qquad (12)
\end{aligned}$$

This is a scheme which uses Newton's iteration (1) in the first step and Newton-like iterations in the subsequent steps. For this reason the scheme is called as Newton-like method. From now on this method is denoted by $\phi_1^{(5)}$. It is clear that $\phi_1^{(5)}$ uses three $\mathbf{F}$, one $\mathbf{F}'$ and one matrix inversion per iteration. In order to analyze the convergence properties, we recall the following result of Taylor's expression on vector functions (see [20]).

Lemma 1 *Let the function $\mathbf{F} : D \subseteq \mathbb{R}^n \to \mathbb{R}^n$ be p-times Fréchet differentiable in a convex set $D \subseteq \mathbb{R}^n$, then for any $\mathbf{x}, \mathbf{h} \in \mathbb{R}^n$ the following expression holds:*

$$\mathbf{F}(\mathbf{x}+\mathbf{h}) = \mathbf{F}(\mathbf{x}) + \mathbf{F}'(\mathbf{x})\mathbf{h} + \frac{1}{2!}\mathbf{F}''(\mathbf{x})\mathbf{h}^2 + \frac{1}{3!}\mathbf{F}'''(\mathbf{x})\mathbf{h}^3 + \cdots + \frac{1}{(p-1)!}\mathbf{F}^{(p-1)}(\mathbf{x})\mathbf{h}^{p-1} + \mathbf{R}_p,$$

where

$$\left|\left|\mathbf{R}_p\right|\right| \leqslant \frac{1}{p!}\sup_{0<t<1}\left|\left|\mathbf{F}^{(p)}(\mathbf{x}+t\mathbf{h})\right|\right| ||\mathbf{h}||^p \text{and} \quad \mathbf{h}^p = (\mathbf{h}, \mathbf{h}, \overset{p-times}{\cdots}, \mathbf{h}).$$

The following theorem gives the convergence order of the proposed method.

Theorem 2 *Let the function $\mathbf{F} : D \subseteq \mathbb{R}^n \to \mathbb{R}^n$ be sufficiently Fréchet differentiable in an open neighborhood D of its zero $\mathbf{r}$. Suppose that $\mathbf{F}'(\mathbf{x})$ is continuous and non-singular in $\mathbf{r}$. If an initial approximation $\mathbf{x}^{(0)}$ is sufficiently close to $\mathbf{r}$, then the local order of convergence of proposed Newton-like method ($\phi_1^{(5)}$) is 5.*

Proof Taylor expansion of $\mathbf{F}(\mathbf{x})$ around $\mathbf{x}^{(k)}$ is

$$\begin{aligned}
\mathbf{F}(\mathbf{x}) = \mathbf{F}\left(\mathbf{x}^{(k)}\right) &+ \mathbf{F}'\left(\mathbf{x}^{(k)}\right)\left(\mathbf{x}-\mathbf{x}^{(k)}\right) + \frac{1}{2!}\mathbf{F}''\left(\mathbf{x}^{(k)}\right)\left(\mathbf{x}-\mathbf{x}^{(k)}\right)^2\\
&+ \frac{1}{3!}\mathbf{F}'''\left(\mathbf{x}^{(k)}\right)\left(\mathbf{x}-\mathbf{x}^{(k)}\right)^3 + \frac{1}{4!}\mathbf{F}^{(iv)}\left(\mathbf{x}^{(k)}\right)\left(\mathbf{x}-\mathbf{x}^{(k)}\right)^4\\
&+ \frac{1}{5!}\mathbf{F}^{(v)}(\mathbf{x}^{(k)})\left(\mathbf{x}-\mathbf{x}^{(k)}\right)^5 + O\left(\left\|(\mathbf{x}-\mathbf{x}^{(k)})^6\right\|\right). \qquad (13)
\end{aligned}$$

Let $\mathbf{e}^{(k)} = \mathbf{x}^{(k)} - \mathbf{r}$ and assuming that $\mathbf{F}'(\mathbf{x}^{(k)})^{-1}$ exists, then setting $\mathbf{x} = \mathbf{r}$ and using $\mathbf{F}(\mathbf{r}) = \mathbf{0}$ in (13),

$$\begin{aligned}
\mathbf{F}\left(\mathbf{x}^{(k)}\right) = \mathbf{F}'\left(\mathbf{x}^{(k)}\right)\mathbf{e}^{(k)} &- \frac{1}{2!}\mathbf{F}''\left(\mathbf{x}^{(k)}\right)(\mathbf{e}^{(k)})^2 + \frac{1}{3!}\mathbf{F}'''\left(\mathbf{x}^{(k)}\right)\left(\mathbf{e}^{(k)}\right)^3\\
&- \frac{1}{4!}\mathbf{F}^{(iv)}\left(\mathbf{x}^{(k)}\right)(\mathbf{e}^{(k)})^4 + \frac{1}{5!}\mathbf{F}^{(v)}\left(\mathbf{x}^{(k)}\right)(\mathbf{e}^{(k)})^5 + O\left(\left\|\left(\mathbf{e}^{(k)}\right)^6\right\|\right). \qquad (14)
\end{aligned}$$

Pre-multiplying (14) by $\mathbf{F}'(\mathbf{x}^{(k)})^{-1}$,

$$\mathbf{F}'\left(\mathbf{x}^{(k)}\right)^{-1}\mathbf{F}\left(\mathbf{x}^{(k)}\right) = \mathbf{e}^{(k)} - \frac{1}{2!}\mathbf{G}\left(\mathbf{x}^{(k)}\right)(\mathbf{e}^{(k)})^2 + \frac{1}{3!}\mathbf{H}\left(\mathbf{x}^{(k)}\right)(\mathbf{e}^{(k)})^3 - \frac{1}{4!}\mathbf{J}\left(\mathbf{x}^{(k)}\right)(\mathbf{e}^{(k)})^4 + \frac{1}{5!}\mathbf{F}'\left(\mathbf{x}^{(k)}\right)^{-1}\mathbf{F}^{(v)}(\mathbf{x}^{(k)})(\mathbf{e}^{(k)})^5 + O\left(\left\|\left(\mathbf{e}^{(k)}\right)^6\right\|\right), \tag{15}$$

where $\mathbf{G}(\mathbf{x}^{(k)}) = \mathbf{F}'(\mathbf{x}^{(k)})^{-1}\mathbf{F}''(\mathbf{x}^{(k)})$, $\mathbf{H}(\mathbf{x}^{(k)}) = \mathbf{F}'(\mathbf{x}^{(k)})^{-1}\mathbf{F}'''(\mathbf{x}^{(k)})$ and $\mathbf{J}(\mathbf{x}^{(k)}) = \mathbf{F}'(\mathbf{x}^{(k)})^{-1}\mathbf{F}^{(iv)}(\mathbf{x}^{(k)})$.
Taylor expansion of $\mathbf{F}(\mathbf{y}^{(k)})$ around $\mathbf{x}^{(k)}$ is

$$\mathbf{F}(\mathbf{y}^{(k)}) = \mathbf{F}(\mathbf{x}^{(k)}) + \mathbf{F}'(\mathbf{x}^{(k)})\left(\mathbf{y}^{(k)} - \mathbf{x}^{(k)}\right) + \frac{1}{2!}\mathbf{F}''(\mathbf{x}^{(k)})\left(\mathbf{y}^{(k)} - \mathbf{x}^{(k)}\right)^2 + \frac{1}{3!}\mathbf{F}'''(\mathbf{x}^{(k)})(\mathbf{y}^{(k)} - \mathbf{x}^{(k)})^3 + \frac{1}{4!}\mathbf{F}^{(iv)}\left(\mathbf{x}^{(k)}\right)\left(\mathbf{y}^{(k)} - \mathbf{x}^{(k)}\right)^4 + \frac{1}{5!}\mathbf{F}^{(v)}\left(\mathbf{x}^{(k)}\right)\left(\mathbf{y}^{(k)} - \mathbf{x}^{(k)}\right)^5 + O\left(\left\|\left(\mathbf{y}^{(k)} - \mathbf{x}^{(k)}\right)^6\right\|\right). \tag{16}$$

Using first step of (12) and then value of $\mathbf{F}'(\mathbf{x}^{(k)})^{-1}\mathbf{F}\left(\mathbf{x}^{(k)}\right)$ from (15) in (16), we get

$$\begin{aligned}\mathbf{F}(\mathbf{y}^{(k)}) = {} & \frac{1}{2}\mathbf{F}''(\mathbf{x}^{(k)})(\mathbf{e}^{(k)})^2 - \left(\frac{1}{2}\mathbf{F}''(\mathbf{x}^{(k)})\mathbf{G}(\mathbf{x}^{(k)}) + \frac{1}{6}\mathbf{F}'''(\mathbf{x}^{(k)})\right)\left(\mathbf{e}^{(k)}\right)^3 \\ & + \left[\frac{1}{2}\mathbf{F}''(\mathbf{x}^{(k)})\left(\frac{1}{4}\mathbf{G}(\mathbf{x}^{(k)})^2 + \frac{1}{3}\mathbf{H}(\mathbf{x}^{(k)})\right) + \frac{1}{4}\mathbf{F}'''(\mathbf{x}^{(k)})\mathbf{G}(\mathbf{x}^{(k)}) + \frac{1}{24}\mathbf{F}^{(iv)}(\mathbf{x}^{(k)})\right](\mathbf{e}^{(k)})^4 \\ & + \left[\frac{1}{2}\mathbf{F}''(\mathbf{x}^{(k)})\left(\frac{1}{12}\mathbf{J}(\mathbf{x}^{(k)}) + \frac{1}{6}\mathbf{G}(\mathbf{x}^{(k)})\mathbf{H}(\mathbf{x}^{(k)})\right) + \frac{1}{6}\mathbf{F}'''(\mathbf{x}^{(k)})\left(\frac{1}{2}\mathbf{H}(\mathbf{x}^{(k)}) + \frac{3}{4}\mathbf{G}(\mathbf{x}^{(k)})^2\right)\right. \\ & \left. + \frac{1}{12}\mathbf{F}^{(iv)}\left(\mathbf{x}^{(k)}\right)\mathbf{G}(\mathbf{x}^{(k)}) + \frac{1}{120}\mathbf{F}^{(v)}(\mathbf{x}^{(k)})\right]\left(\mathbf{e}^{(k)}\right)^5 + O\left(\left\|\left(\mathbf{e}^{(k)}\right)^6\right\|\right).\end{aligned} \tag{17}$$

Taylor expansion of $\mathbf{F}(\mathbf{z}^{(k)})$ around $\mathbf{x}^{(k)}$ is

$$\mathbf{F}(\mathbf{z}^{(k)}) = \mathbf{F}(\mathbf{x}^{(k)}) + \mathbf{F}'(\mathbf{x}^{(k)})(\mathbf{z}^{(k)} - \mathbf{x}^{(k)}) + \frac{1}{2!}\mathbf{F}''(\mathbf{x}^{(k)})\left(\mathbf{z}^{(k)} - \mathbf{x}^{(k)}\right)^2 + \frac{1}{3!}\mathbf{F}'''\left(\mathbf{x}^{(k)}\right)\left(\mathbf{z}^{(k)} - \mathbf{x}^{(k)}\right)^3 + \frac{1}{4!}\mathbf{F}^{(iv)}\left(\mathbf{x}^{(k)}\right)\left(\mathbf{z}^{(k)} - \mathbf{x}^{(k)}\right)^4 + \frac{1}{5!}\mathbf{F}^{(v)}\left(\mathbf{x}^{(k)}\right)\left(\mathbf{z}^{(k)} - \mathbf{x}^{(k)}\right)^5 + O\left(\left\|\left(\mathbf{z}^{(k)} - \mathbf{x}^{(k)}\right)^6\right\|\right). \tag{18}$$

Combining the first two steps of (12), we obtain $\mathbf{z}^{(k)} - \mathbf{x}^{(k)} = -\mathbf{F}'(\mathbf{x}^{(k)})^{-1}\left(\mathbf{F}\left(\mathbf{x}^{(k)}\right) + 5\mathbf{F}(\mathbf{y}^{(k)})\right)$. Then using this value in (18), it follows that

$$\begin{aligned}\mathbf{F}(\mathbf{z}^{(k)}) = {} & -5\mathbf{F}(\mathbf{y}^{(k)}) + \frac{1}{2}\mathbf{F}''(\mathbf{x}^{(k)})(\mathbf{e}^{(k)})^2 - \left(\frac{1}{6}\mathbf{F}'''(\mathbf{x}^{(k)}) - 2\mathbf{F}''(\mathbf{x}^{(k)})\mathbf{G}(\mathbf{x}^{(k)})\right)\left(\mathbf{e}^{(k)}\right)^3 \\ & - \left(\frac{1}{2}\mathbf{F}''(\mathbf{x}^{(k)})\mathbf{G}(\mathbf{x}^{(k)})^2 + \frac{2}{3}\mathbf{F}''\left(\mathbf{x}^{(k)}\right)\mathbf{H}\left(\mathbf{x}^{(k)}\right) + \mathbf{F}'''\left(\mathbf{x}^{(k)}\right)\mathbf{G}(\mathbf{x}^{(k)}) - \frac{1}{24}\mathbf{F}^{(iv)}(\mathbf{x}^{(k)})\right)\left(\mathbf{e}^{(k)}\right)^4\end{aligned}$$

$$-\left(\frac{35}{8}\mathbf{F}''\left(\mathbf{x}^{(k)}\right)\mathbf{G}(\mathbf{x}^{(k)})^3+\frac{1}{2}\mathbf{F}''\left(\mathbf{x}^{(k)}\right)\mathbf{G}(\mathbf{x}^{(k)})\mathbf{H}(\mathbf{x}^{(k)})-\frac{5}{4}\mathbf{F}''\left(\mathbf{x}^{(k)}\right)\mathbf{H}(\mathbf{x}^{(k)})\mathbf{G}\left(\mathbf{x}^{(k)}\right)$$
$$-\frac{1}{6}\mathbf{F}''\left(\mathbf{x}^{(k)}\right)\mathbf{J}(\mathbf{x}^{(k)})+\frac{3}{4}\mathbf{F}'''\left(\mathbf{x}^{(k)}\right)\mathbf{G}(\mathbf{x}^{(k)})^2-\frac{1}{3}\mathbf{F}'''\left(\mathbf{x}^{(k)}\right)\mathbf{H}\left(\mathbf{x}^{(k)}\right)-\frac{1}{3}\mathbf{F}^{(iv)}(\mathbf{x}^{(k)})\mathbf{G}(\mathbf{x}^{(k)})$$
$$+\frac{1}{120}\mathbf{F}^{(v)}(\mathbf{x}^{(k)})\Big)\left(\mathbf{e}^{(k)}\right)^5+O\left(\left\|\left(\mathbf{e}^{(k)}\right)^6\right\|\right). \tag{19}$$

Combining all the three steps of (12), we can write

$$\mathbf{x}^{(k+1)}=\mathbf{x}^{(k)}-\mathbf{F}'(\mathbf{x}^{(k)})^{-1}\left(\mathbf{F}(\mathbf{x}^{(k)})+\frac{9}{5}\mathbf{F}(\mathbf{y}^{(k)})+\frac{1}{5}\mathbf{F}(\mathbf{z}^{(k)})\right). \tag{20}$$

Using (14), (17) and (19) in (20), we obtain the error equation as

$$\mathbf{e}^{(k+1)}=\Big(\frac{7}{8}\mathbf{G}(\mathbf{x}^{(k)})^4+\frac{1}{6}\mathbf{G}(\mathbf{x}^{(k)})^2\mathbf{H}(\mathbf{x}^{(k)})-\frac{1}{4}\mathbf{G}(\mathbf{x}^{(k)})\mathbf{H}(\mathbf{x}^{(k)})\mathbf{G}(\mathbf{x}^{(k)})$$
$$+\frac{1}{4}\mathbf{H}(\mathbf{x}^{(k)})\mathbf{G}(\mathbf{x}^{(k)})^2\Big)\left(\mathbf{e}^{(k)}\right)^5+O\left(\left\|\left(\mathbf{e}^{(k)}\right)^6\right\|\right). \tag{21}$$

This equation shows that proposed method ($\phi_1^{(5)}$) for system of equations possesses fifth order of convergence. This completes the proof of Theorem 2. □

5 Computational efficiency

In order to assess the computational efficiency of derived method, we will consider all possible number of evaluations that contribute to the total cost of computation. For example, to compute $\mathbf{F}$ in any iterative method we evaluate n scalar functions, whereas the number of scalar evaluations is n^2 for any new derivative $\mathbf{F}'$. In addition, we must include the amount of computational work required to evaluate the inverse of a matrix. Instead of computing the inverse operator we solve a linear system, where we have $n(n-1)(2n-1)/6$ products and $n(n-1)/2$ quotients in the LU decomposition, and $n(n-1)$ products and n quotients in the resolution of two triangular linear systems. Moreover, we must add n^2 products for the multiplication of a matrix with a vector or of a matrix by a scalar and n products for the multiplication of a vector by a scalar. We suppose that a quotient is equivalent to l products.

Computational efficiency of the presented Newton-like method $\phi_1^{(5)}$ is compared with some well-known fourth and fifth order methods. For example, fourth order generalized Jarratt's method [5], fourth order method by Cordero et al. [4], and fifth order methods by Cordero et al. [5,6], Grau et al. [11] and Sharma and Gupta [25]. The existing mentioned methods are given as follows:

Fourth order Generalized Jarratt method ($\phi_1^{(4)}$):

$$\mathbf{y}^{(k)}=\mathbf{x}^{(k)}-\frac{2}{3}\mathbf{F}'(\mathbf{x}^{(k)})^{-1}\mathbf{F}(\mathbf{x}^{(k)}),$$
$$\mathbf{x}^{(k+1)}=\mathbf{x}^{(k)}-\frac{1}{2}\Big[3\mathbf{F}'(\mathbf{y}^{(k)})-\mathbf{F}'(\mathbf{x}^{(k)})\Big]^{-1}\Big[3\mathbf{F}'(\mathbf{y}^{(k)})+\mathbf{F}'(\mathbf{x}^{(k)})\Big]\mathbf{F}'(\mathbf{x}^{(k)})^{-1}\mathbf{F}(\mathbf{x}^{(k)}).$$

Fourth order method by Cordero et al. ($\phi_2^{(4)}$):

$$\begin{aligned}
\mathbf{y}^{(k)} &= \phi_1^{(2)}(\mathbf{x}^{(k)}),\\
\mathbf{x}^{(k+1)} &= \mathbf{y}^{(k)} - \left[2\mathbf{F}'(\mathbf{x}^{(k)})^{-1} - \mathbf{F}'(\mathbf{x}^{(k)})^{-1}\mathbf{F}'(\mathbf{y}^{(k)})\mathbf{F}'(\mathbf{x}^{(k)})^{-1}\right]\mathbf{F}(\mathbf{y}^{(k)}).
\end{aligned}$$

Fifth order method by Cordero et al. ($\phi_2^{(5)}$):

$$\begin{aligned}
\mathbf{y}^{(k)} &= \mathbf{x}^{(k)} - \frac{2}{3}\mathbf{F}'(\mathbf{x}^{(k)})^{-1}\mathbf{F}(\mathbf{x}^{(k)}),\\
\mathbf{z}^{(k)} &= \mathbf{x}^{(k)} - \frac{1}{2}\left[3\mathbf{F}'(\mathbf{y}^{(k)}) - \mathbf{F}'(\mathbf{x}^{(k)})\right]^{-1}\left[3\mathbf{F}'(\mathbf{y}^{(k)}) + \mathbf{F}'(\mathbf{x}^{(k)})\right]\mathbf{F}'(\mathbf{x}^{(k)})^{-1}\mathbf{F}(\mathbf{x}^{(k)}),\\
\mathbf{x}^{(k+1)} &= \mathbf{z}^{(k)} - \left[\alpha\mathbf{F}'(\mathbf{x}^{(k)}) + (1-\alpha)\mathbf{F}'(\mathbf{y}^{(k)})\right]^{-1}\mathbf{F}(\mathbf{z}^{(k)}),
\end{aligned}$$

where $\alpha \in \mathbb{R} - \{-1/2\}$.
Fifth order method by Cordero et al. ($\phi_3^{(5)}$):

$$\begin{aligned}
\mathbf{y}^{(k)} &= \phi_1^{(2)}(\mathbf{x}^{(k)}),\\
\mathbf{z}^{(k)} &= \mathbf{x}^{(k)} - 2\left[\mathbf{F}'(\mathbf{y}^{(k)}) + \mathbf{F}'(\mathbf{x}^{(k)})\right]^{-1}\mathbf{F}(\mathbf{x}^{(k)}),\\
\mathbf{x}^{(k+1)} &= \mathbf{z}^{(k)} - \mathbf{F}'(\mathbf{y}^{(k)})^{-1}\mathbf{F}(\mathbf{z}^{(k)}).
\end{aligned}$$

Fifth order method by Grau et al. ($\phi_4^{(5)}$):

$$\begin{aligned}
\mathbf{y}^{(k)} &= \phi_1^{(2)}(\mathbf{x}^{(k)}),\\
\mathbf{z}^{(k)} &= \mathbf{x}^{(k)} - \frac{1}{2}\left[\mathbf{F}'(\mathbf{x}^{(k)})^{-1} + \mathbf{F}'(\mathbf{y}^{(k)})^{-1}\right]\mathbf{F}(\mathbf{x}^{(k)}),\\
\mathbf{x}^{(k+1)} &= \mathbf{z}^{(k)} - \mathbf{F}'(\mathbf{y}^{(k)})^{-1}\mathbf{F}(\mathbf{z}^{(k)}).
\end{aligned}$$

Fifth order Sharma-Gupta method ($\phi_5^{(5)}$):

$$\begin{aligned}
\mathbf{y}^{(k)} &= \mathbf{x}^{(k)} - \frac{1}{2}\mathbf{F}'(\mathbf{x}^{(k)})^{-1}\mathbf{F}(\mathbf{x}^{(k)}),\\
\mathbf{z}^{(k)} &= \mathbf{x}^{(k)} - \mathbf{F}'(\mathbf{y}^{(k)})^{-1}\mathbf{F}(\mathbf{x}^{(k)}),\\
\mathbf{x}^{(k+1)} &= \mathbf{z}^{(k)} - \left[2\mathbf{F}'(\mathbf{y}^{(k)})^{-1} - \mathbf{F}'(\mathbf{x}^{(k)})^{-1}\right]\mathbf{F}(\mathbf{z}^{(k)}).
\end{aligned}$$

Let us denote efficiency indices of $\phi_i^{(p)}$ by $E_i^{(p)}$ and computational cost by $C_i^{(p)}$. Then using the Definition 2.4 of computational efficiency while taking into account all the possible evaluations discussed above, we have

$$C_1^{(4)} = n\mu_0 + 2n^2\mu_1 + \frac{n}{3}\left(2n^2 + 9n + 1 + 3l(n+1)\right) \quad \text{and} \quad E_1^{(4)} = 4^{1/C_1^{(4)}}. \tag{22}$$

$$C_2^{(4)} = 2n\mu_0 + 2n^2\mu_1 + \frac{n}{6}\left(2n^2 + 21n - 11 + 3l(n+5)\right) \quad \text{and} \quad E_2^{(4)} = 4^{1/C_2^{(4)}}. \tag{23}$$

$$C_1^{(5)} = 3n\mu_0 + n^2\mu_1 + \frac{n}{6}\left(2n^2 + 15n + 1 + 3l(n+5)\right) \quad \text{and} \quad E_1^{(5)} = 5^{1/C_1^{(5)}}. \tag{24}$$

$$C_2^{(5)} = 2n\mu_0 + 2n^2\mu_1 + \frac{n}{2}\left(2n^2 + 11n - 1 + 3l(n+1)\right) \quad \text{and} \quad E_2^{(5)} = 5^{1/C_2^{(5)}}. \tag{25}$$

$$C_3^{(5)} = 2n\mu_0 + 2n^2\mu_1 + \frac{n}{2}\left(2n^2 + 3n - 3 + 3l(n+1)\right) \quad \text{and} \quad E_3^{(5)} = 5^{1/C_3^{(5)}}. \tag{26}$$

$$C_4^{(5)} = 2n\mu_0 + 2n^2\mu_1 + \frac{n}{3}\left(2n^2 + 6n - 5 + 3l(n+2)\right) \quad \text{and} \quad E_4^{(5)} = 5^{1/C_4^{(5)}}. \tag{27}$$

$$C_5^{(5)} = 2n\mu_0 + 2n^2\mu_1 + \frac{n}{3}\left(2n^2 + 9n - 5 + 3l(n+3)\right) \quad \text{and} \quad E_5^{(5)} = 5^{1/C_5^{(5)}}. \tag{28}$$

Here μ_0 and μ_1 are the ratios between products and evaluations as stated in the definition of computational efficiency in Sect. 2.

5.1 Efficiency comparison

To compare the computational efficiencies of the iterative methods, say $\phi_i^{(p)}$ against $\phi_j^{(q)}$, we consider the ratio

$$R_{i,j}^{p,q} = \frac{\log E_i^{(p)}}{\log E_j^{(q)}} = \frac{C_j^{(q)}\log(p)}{C_i^{(p)}\log(q)}. \tag{29}$$

It is clear that if $R_{i,j}^{p,q} > 1$, the iterative method $\phi_i^{(p)}$ is more efficient than $\phi_j^{(q)}$. Note that the boundary between two computational efficiencies is given by $R_{i,j}^{p,q} = 1$, this boundary is expressed by the equation μ_0 written as a function of μ_1, n and l; $(\mu_1, \mu_0) \in (0, +\infty) \times (0, +\infty)$, n is a positive integer $\geqslant 2$ and $l \geqslant 1$.

$\phi_1^{(5)}$ versus $\phi_1^{(4)}$ case:

Using (22) and (24) in (29) the boundary $R_{1,1}^{5,4} = 1$, expressed in μ_0 as function of μ_1, n and l, is given by

$$\mu_0 = \frac{6(s-2r)n\mu_1 + (s-2r)n^2 + 3(5s-6r)n + (s-2r) + 3(s-2r)ln + 3(5s-r)l}{6(r-3s)},$$

where $r = \log(5)$ and $s = \log(4)$. A comparison between the efficiencies $E_1^{(5)}$ and $E_1^{(4)}$ can be made in the (μ_1, μ_0)-plane. In Fig. 1, we present some boundary lines

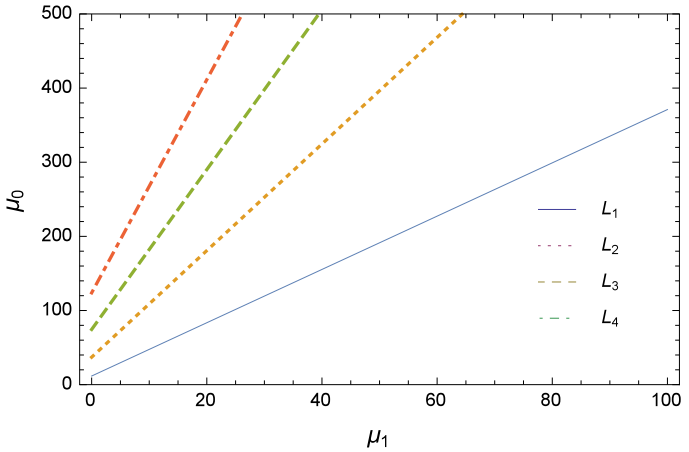

Fig. 1 *Boundary lines* in the (μ_1, μ_0)-plane for E_1^5 and E_1^4

L_i $(i = 1, 2, 3, 4)$ in the (μ_1, μ_0)-plane corresponding to $n = 5$, 10, 15 and 20 taking $l = 2.66$ in each case. Reason for selecting the value 2.66 for l will be clear in the next section. These boundaries are the straight lines with positive slope, where $E_1^{(5)} > E_1^{(4)}$ on the right (below) and $E_1^4 > E_1^5$ on the above (left) of each line.

$\phi_1^{(5)}$ *versus* $\phi_2^{(4)}$ *case*:

The boundary $R_{1,2}^{5,4} = 1$, calculated by using (23) and (24) in (29), is expressed by

$$\mu_0 = \frac{6(s-2r)n\mu_1 + 2(s-r)n^2 + 3(5s-7r)n + (s+11r) + 3(s-r)ln + 15(s-r)l}{6(2r-3s)}.$$

In order to compare the efficiencies $E_1^{(5)}$ and $E_2^{(4)}$, here we also draw some particular boundaries L_i $(i = 1, 2, 3, 4)$ in the (μ_1, μ_0)-plane using the same set of values of n and l as in the previous case. These boundaries are the straight lines with positive slopes, where $E_1^{(5)} > E_2^{(4)}$ on the right and $E_2^{(4)} > E_1^{(5)}$ on the left of each line (see Fig. 2).

$\phi_1^{(5)}$ *versus* $\phi_2^{(5)}$ *case*:

For this case the boundary $R_{1,2}^{5,5} = 1$ is given as

$$\mu_0 = n\mu_1 + \frac{2}{3}(n^2 - 1) + 3n + ln - l.$$

The comparison between the efficiencies $E_1^{(5)}$ and $E_2^{(5)}$ can be made in the (μ_1, μ_0)-plane. Thus, we draw some particular boundaries L_i $(i = 1, 2, 3, 4)$ in the (μ_1, μ_0)-plane using the values of n and l considered in the previous cases. The boundaries are the straight lines with positive slope, where $E_1^{(5)} > E_2^{(5)}$ on the right (below) and $E_2^5 > E_1^5$ on the above (left) of each line (see Fig. 3).

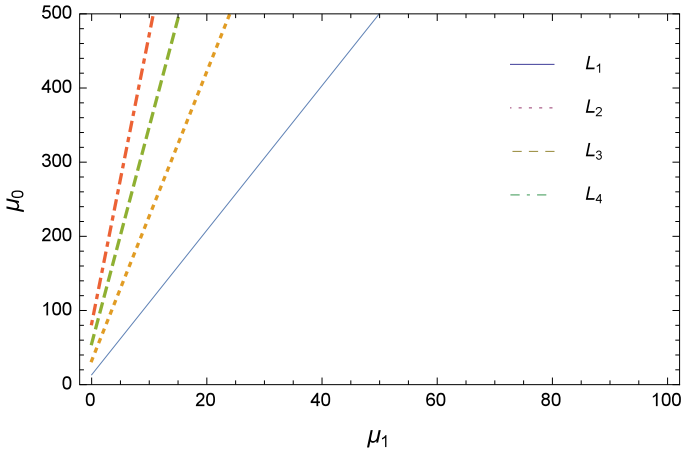

Fig. 2 *Boundary lines* in the (μ_1, μ_0)-plane for E_1^5 and E_2^4

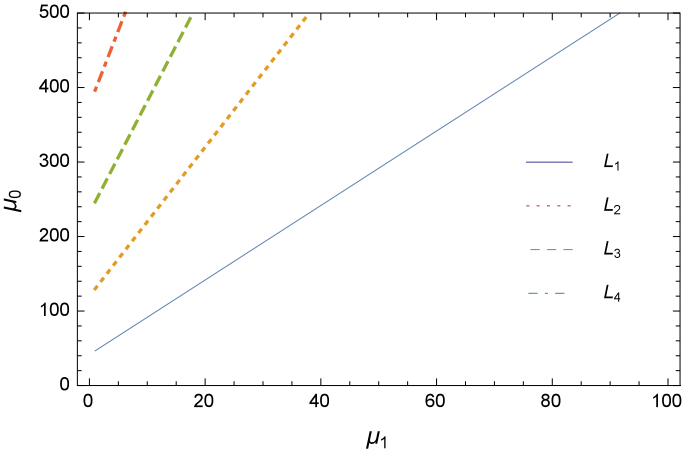

Fig. 3 *Boundary lines* in the (μ_1, μ_0)-plane for E_1^5 and E_2^5

$\phi_1^{(5)}$ versus $\phi_3^{(5)}$ case:

The boundary $R_{1,3}^{5,5} = 1$ is given by

$$\mu_0 = n\mu_1 + \frac{1}{3}(2n^2 - 5) - n + ln - l.$$

Here also we show some boundary lines L_i $(i = 1, 2, 3, 4)$ in the (μ_1, μ_0)-plane using the values of n and l as in the previous cases for comparing the efficiencies $E_1^{(5)}$ and $E_3^{(5)}$. Such boundaries are straight lines with positive slopes, where $E_1^{(5)} > E_3^{(5)}$ on the right and $E_3^{(5)} > E_1^{(5)}$ on the left of each line (see Fig. 4).

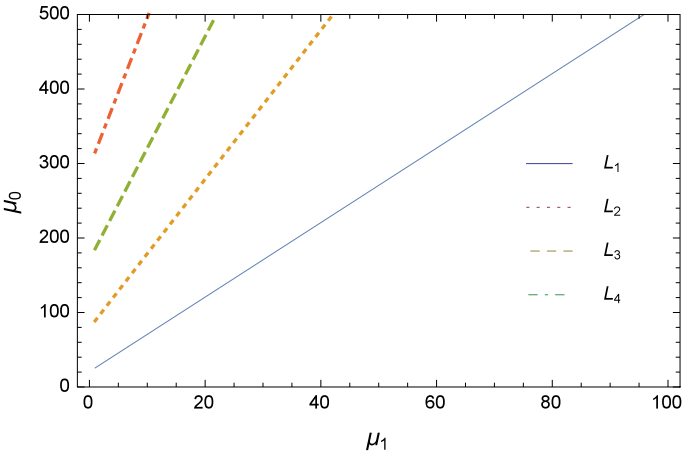

Fig. 4 *Boundary lines* in the (μ_1, μ_0)-plane for E_1^5 and E_3^5

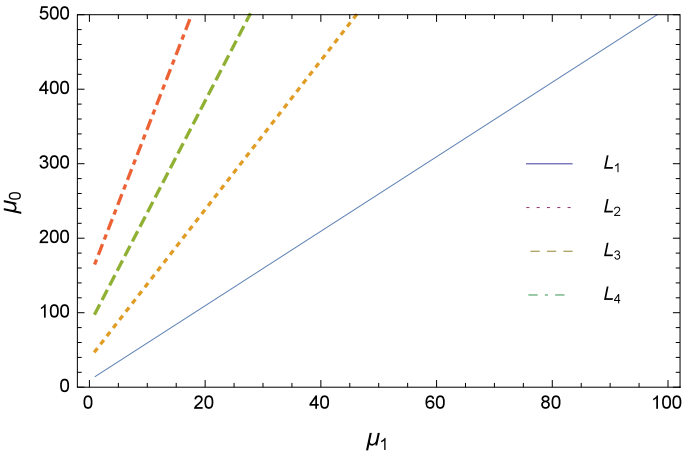

Fig. 5 *Boundary lines* in the (μ_1, μ_0)-plane for E_1^5 and E_4^5

$\phi_1^{(5)}$ versus $\phi_4^{(5)}$ case:

The boundary $R_{1,4}^{5,5} = 1$ is expressed by

$$\mu_0 = n\mu_1 + \frac{1}{6}\left(2n^2 - 3n - 11 + 3ln - 3l\right).$$

We draw the boundaries L_i $(i = 1, 2, 3, 4)$ in the (μ_1, μ_0)-plane using the same set of values of n and l as in the previous case. These boundaries are the straight lines with positive slope, where $E_1^{(5)} > E_4^{(5)}$ on the right and $E_4^{(5)} > E_1^{(5)}$ on the left of each line (see Fig. 5).

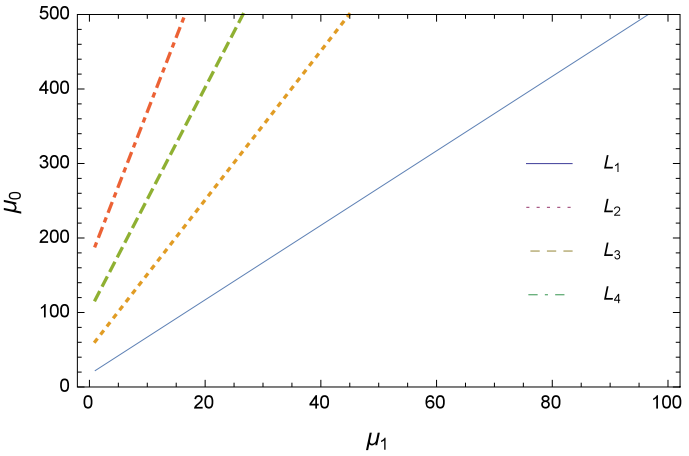

Fig. 6 *Boundary lines* in the (μ_1, μ_0)-plane for E_1^5 and E_5^5

$\phi_1^{(5)}$ versus $\phi_5^{(5)}$case:

For this case the boundary $R_{1,5}^{5,5} = 1$ is given by

$$\mu_0 = n\mu_1 + \frac{1}{6}\big(2n^2 + 3n - 11 + 3ln + 3l\big).$$

We draw the boundaries L_i $(i = 1, 2, 3, 4)$ in the (μ_1, μ_0)-plane using the previously considered values of n and l in order to compare the efficiencies $E_1^{(5)}$ and $E_5^{(5)}$. The boundaries are the straight lines with positive slope, where $E_1^{(5)} > E_5^{(5)}$ on the right and $E_5^{(5)} > E_1^{(5)}$ on the left of each line (see Fig. 6).

Below we summarize the above proved results:

Theorem 3 *For $\mu_0 > 0$, $\mu_1 > 0$, $l \geqslant 1$ and $n \geqslant 2$ we have:*

(i) $E_1^{(5)} > E_1^{(4)}$ *for* $\mu_0 < m_1$,
(ii) $E_1^{(5)} > E_2^{(4)}$ *for* $\mu_0 < m_2$,
(iii) $E_1^{(5)} > E_2^{(5)}$ *for* $\mu_0 < m_3$,
(iv) $E_1^{(5)} > E_3^{(5)}$ *for* $\mu_0 < m_4$,
(v) $E_1^{(5)} > E_4^{(5)}$ *for* $\mu_0 < m_5$,
(vi) $E_1^{(5)} > E_5^{(5)}$ *for* $\mu_0 < m_6$,

where

$$m_1 = \frac{6(s-2r)n\mu_1+(s-2r)n^2+3(5s-6r)n+(s-2r)+3(s-2r)ln+3(5s-r)l}{6(r-3s)},$$

$$m_2 = \frac{6(s-2r)n\mu_1+2(s-r)n^2+3(5s-7r)n+(s+11r)+3(s-r)ln+15(s-r)l}{6(2r-3s)},$$

$$m_3 = n\mu_1 + \frac{2}{3}(n^2-1) + 3n + ln - l,$$

$$m_4 = n\mu_1 + \frac{1}{3}(2n^2 - 5) - n + ln - l,$$
$$m_5 = n\mu_1 + \frac{1}{6}\left(2n^2 - 3n - 11 + 3ln - 3l\right),$$
$$m_6 = n\mu_1 + \frac{1}{6}\left(2n^2 + 3n - 11 + 3ln + 3l\right).$$

Remark 2 It is clear from Figs. 1, 2, 3, 4, 5 and 6 that the efficiency region of the proposed method ($\phi_1^{(5)}$) increases in size with increasing value of n as compared with the efficiency regions of existing methods which decrease in size. That means the efficiency of new method is greater than the efficiency of existing methods in a wide region of (μ_1, μ_0)-plane with increasing n. This shows that the proposed method is more efficient, especially in case of the systems with large dimensions.

Remark 3 It has been seen that, in general, the presented method is more efficient than the second and third order methods. For this reason we have not included such lower order methods in the comparison of computational efficiencies.

6 Numerical results

In order to illustrate the convergence behavior and computational efficiency of the new scheme $\phi_1^{(5)}$, we consider some numerical examples and compare the performance with existing methods, namely fourth order ($\phi_i^{(4)}$, $i = 1, 2$) and fifth order ($\phi_j^{(5)}$, $j = 2, 3, 4, 5$) methods. The computations are performed in the programming package Mathematica [27] using multiple-precision arithmetic with 4096 digits. For every method, we analyze the number of iterations (k) needed to converge to the solution such that the stopping criterion $||\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}|| + ||\mathbf{F}(\mathbf{x}^{(k)})|| < 10^{-300}$ is satisfied. In numerical results, we also include CPU time used in the execution of program which is computed by the Mathematica command "TimeUsed[ ]".

The results of Theorem 3 are also verified through numerical examples. In order to do this, we need an estimation of the factors μ_0 and μ_1. To claim this estimation, we express the cost of the evaluation of elementary functions in terms of products, which depends on the computer, the software and the arithmetics used (see, for example [9]). In Table 1, the elapsed CPU time (measured in milliseconds) in the computation of elementary functions and an estimation of the cost of the elementary functions in product units are displayed. The programs are performed in the processor Intel (R) Core

Table 1 CPU time and estimation of computational cost of elementary functions, where $x = \sqrt{3} - 1$ and $y = \sqrt{5}$

Functions	xy	x/y	$\sqrt{x}$	e^x	$\ln(x)$	$\sin(x)$	$\cos(x)$	$\arccos(x)$	$\arctan(x)$
CPU time	0.0495	0.1317	0.0601	3.8934	3.7347	4.8079	4.7908	7.8906	7.6487
Cost	1	2.66	1.22	78.74	75.52	97.23	96.88	159.57	154.68

(TM) i5-520M CPU @ 2.40 GHz (64-bit Machine) Microsoft Windows 7 Home Premium 2009 and are complied by computational software program *Mathematica* using multiple-precision arithmetic. It can be observed from Table 1 that for this hardware and the software, the computational cost of division with respect to multiplication is, $l = 2.66$.

For numerical tests we consider the following problems:

Problem 1 Consider the system of two equations (selected from [25]):

$$\begin{cases} x_1 + e^{x_2} - \cos x_2 = 0, \\ 3x_1 - \sin x_1 - x_2 = 0, \end{cases}$$

with initial value $\mathbf{x}^{(0)} = \{-1.5, 1.5\}^T$ towards the solution: $\mathbf{r} = \{0, 0\}^T$. For this problem the corresponding values of parameters μ_0 and μ_1 (calculated by using the Table 1) are 136.93 and 68.21 which we use in (22)–(28) for computing computational costs and efficiency indices, and also to verify the results of theorem 3. Observe that the other parameters are $(n, l) = (2, 2.66)$.

Problem 2 Consider the Gauss-Legendre quadrature formula:

$$\int_0^1 f(x)dx = \sum_{j=1}^{m} \omega_j f(x_j),$$

where x_j and ω_j are called abscissas and weights, respectively. The abscissas and weights are symmetrical with respect to the middle point of the interval. There being $2m$ unknowns, $2m$ relations between them are necessary so that the formula is exact for all polynomials of degree not exceeding $2m - 1$. Thus we consider

$$f(x) = \sum_{i=0}^{2m-1} c_i x^i.$$

Then,

$$\int_0^1 f(x)dx = \int_0^1 \Big(\sum_{i=0}^{2m-1} c_i x^i \Big) dx = \sum_{i=0}^{2m-1} \frac{c_i}{i+1}.$$

Also,

$$\int_0^1 f(x)dx = \sum_{j=1}^{m} \sum_{i=0}^{2m-1} \omega_j c_i x_j^i.$$

But both the above last equations are identical for all values of c_i, hence comparing coefficients of c_i, we obtain the following system of $2m$ equations in $2m$ unknowns x_j and w_j $(j = 1, 2, \ldots, m)$:

$$\sum_{j=1}^{m} \omega_j x_j^i - \frac{1}{i+1} = 0, \quad i = 0, 1, 2, \ldots, 2m-1.$$

In particular, we solve this problem for $m = 2$ so that $n = 4$ by choosing the initial value $\mathbf{x}^{(0)} = \{x_1^{(0)}, x_2^{(0)}, \omega_1^{(0)}, \omega_2^{(0)}\}^T = \{0, 1, 1, 1\}^T$ towards the solution:

$$\mathbf{r} = \{x_1, x_2, \omega_1, \omega_2\}^T = \{0.21132486540518712\ldots, 0.78867513459481288\ldots, 0.5, 0.5\}^T.$$

For this problem the corresponding values of parameters μ_0 and μ_1, calculated by using the Table 1, are 1.5 and 0.75. Note that the other parameters are $(n, l) = (4, 2.66)$.

Problem 3 The boundary value problem (see [20]):

$$u'' + a^2(u')^2 + 1 = 0, \quad u(0) = 0, \quad u(1) = 0,$$

is studied. Consider the following partitioning of the interval [0, 1]:

$$t_0 = 0 < t_1 < t_2 < \cdots < t_{m-1} < t_m = 1, \quad t_{j+1} = t_j + h, \quad h = 1/m.$$

Let us define $u_0 = u(t_0) = 0, u_1 = u(t_1), \ldots, u_{m-1} = u(t_{m-1}), u_m = u(t_m) = 1$. If we discretize the problem by using the numerical formulae for first and second derivatives

$$u'_k = \frac{u_{k+1} - u_{k-1}}{2h}, \quad u''_k = \frac{u_{k-1} - 2u_k + u_{k+1}}{h^2}, \quad (k = 1, 2, 3, \ldots, m-1),$$

we obtain a system of $m - 1$ nonlinear equations in $m - 1$ variables:

$$u_{k-1} - 2u_k + u_{k+1} + a^2(u_{k+1} - u_{k-1})^2 + h^2 = 0, \quad (k = 1, 2, 3, \ldots, m-1).$$

In particular, we solve this problem for $m = 9$ so that $n = 8$ by selecting $\mathbf{u}^{(0)} = \{-1, -1, \ldots, -1\}^T$ as the initial value and $a = 2$. The solution of this problem is,

$$\begin{aligned}\mathbf{r} = \{&0.068510993237298025\ldots, 0.11210808478752376\ldots, 0.13846581959470837\ldots,\\ &0.15096779807292381\ldots, 0.15096779807292381\ldots, 0.13846581959470837\ldots,\\ &0.11210808478752376\ldots, 0.068510993237298025\ldots\}^T\end{aligned}$$

and concrete values of the parameters (n, l, μ_0, μ_1) are $(8, 2.66, 2, 0.125)$.

Problem 4 Consider the system of thirteen equations (selected from [12]):

$$\sum_{j=1, j\neq i}^{13} x_j - e^{-x_i} = 0, \quad 1 \leq i \leq 13,$$

with initial value $\mathbf{x}^{(0)} = \{2.5, 2.5, \ldots, 2.5\}^T$ towards the solution:

$$\begin{aligned}\mathbf{r} = \{&0.077146207613064638\ldots, 0.077146207613064638\ldots, \ldots,\\ &0.077146207613064638\ldots\}^T.\end{aligned}$$

Table 2 Comparison of the performances of methods

Methods	k	ρ_k	$C_i^{(p)}$	$E_i^{(p)}$	CPU-time
Problem 1					
$\phi_1^{(4)}$	7	4.00000	853.50	1.001626	0.32067
$\phi_2^{(4)}$	7	4.00000	1125.02	1.001233	0.38301
$\phi_1^{(5)}$	6	5.00000	1126.04	1.001430	0.33458
$\phi_2^{(5)}$	6	5.00000	1146.34	1.001405	0.34845
$\phi_3^{(5)}$	6	5.00000	1128.34	1.001427	0.34367
$\phi_4^{(5)}$	6	5.00000	1124.68	1.001432	0.32165
$\phi_5^{(5)}$	6	5.00000	1134.00	1.001420	0.34780
Problem 2					
$\phi_1^{(4)}$	6	4.00000	175.20	1.007944	0.17334
$\phi_2^{(4)}$	7	4.00000	153.88	1.009050	0.17156
$\phi_1^{(5)}$	6	5.00000	139.88	1.011572	0.11612
$\phi_2^{(5)}$	6	5.00000	265.80	1.006073	0.24166
$\phi_3^{(5)}$	6	5.00000	197.80	1.008170	0.22189
$\phi_4^{(5)}$	5	6.00000	167.84	1.010733	0.15645
$\phi_5^{(5)}$	6	5.00000	194.48	1.008310	0.19067
Problem 3					
$\phi_1^{(4)}$	7	4.00000	759.52	1.001827	0.43865
$\phi_2^{(4)}$	7	4.00000	566.32	1.002451	0.42886
$\phi_1^{(5)}$	6	5.00000	526.32	1.003063	0.24433
$\phi_2^{(5)}$	6	5.00000	1195.28	1.001347	0.50267
$\phi_3^{(5)}$	6	5.00000	931.28	1.001730	0.47458
$\phi_4^{(5)}$	6	6.00000	716.80	1.002503	0.35889
$\phi_5^{(5)}$	6	5.00000	802.08	1.002009	0.36223
Problem 4					
$\phi_1^{(4)}$	6	4.00000	5531.01	1.000251	3.58862
$\phi_2^{(4)}$	6	4.00000	5705.73	1.000243	3.65267
$\phi_1^{(5)}$	5	5.00000	5562.71	1.000289	2.52756
$\phi_2^{(5)}$	5	5.00000	7940.69	1.000203	4.42524
$\phi_3^{(5)}$	5	5.00000	7251.69	1.000222	4.24344
$\phi_4^{(5)}$	5	5.00000	6394.21	1.000252	3.53235
$\phi_5^{(5)}$	5	5.00000	6597.79	1.000244	3.59426
Problem 5					
$\phi_1^{(4)}$	7	4.00000	13484.40	1.000103	37.24354
$\phi_2^{(4)}$	7	4.00000	12459.80	1.000111	35.17520

Table 2 continued

Methods	k	ρ_k	$C_i^{(p)}$	$E_i^{(p)}$	CPU-Time
$\phi_1^{(5)}$	6	5.00000	12092.60	1.000133	17.09776
$\phi_2^{(5)}$	6	5.00000	19630.60	1.000082	35.45393
$\phi_3^{(5)}$	6	5.00000	18010.60	1.000089	33.61846
$\phi_4^{(5)*}$	6	5.00000	15035.20	1.000107	30.01342
$\phi_5^{(5)}$	6	5.00000	15488.40	1.000104	30.73324

* $\phi_4^{(5)}$ converges to solution $\mathbf{r}_2$, whereas the rest converge to $\mathbf{r}_1$

For this problem values of the parameters (n, l, μ_0, μ_1) are $(13, 2.66, 78.74, 6.057)$.

Problem 5 Lastly, considering the system of twenty equations (selected from [25]):

$$x_i - \cos\left(2x_i - \sum_{j=1}^{20} x_j\right) = 0, \quad 1 \le i \le 20.$$

In this problem a closer choice of initial approximation to the required solution is very much needed since the problem has many solution vectors with the same value of each component of magnitude less than one in every solution vector. That means each solution vector satisfies $\|\mathbf{r}\| = \sqrt{\sum_{i=1}^{20} |r_i|^2} < \sqrt{20}$. The two solutions of this problem are given by,

$$\mathbf{r}_1 = \{-0.89797814194212824\ldots, -0.89797814194212824\ldots, \ldots, -0.89797814194212824\ldots\}^T$$

and

$$\mathbf{r}_2 = \{-0.57671512524652449\ldots, -0.57671512524652449\ldots, \ldots, -0.57671512524652449\ldots\}^T.$$

We choose the initial approximation $\mathbf{x}^{(0)} = \{-1, -1, \ldots, -1\}^T$ to solve the problem. The concrete values of parameters used in (22)–(28) are $(n, l, \mu_0, \mu_1) = (20, 2.66, 96.88, 4.862)$.

In Table 2, we exhibit numerical results obtained for the considered problems 1–5 by implementing the methods $\phi_i^{(4)}$, $(i = 1, 2)$ and $\phi_j^{(5)}$, $(j = 1, 2, \ldots, 5)$. Displayed in the table are the necessary iterations (k), the computational order of convergence (ρ_k), the computational cost $(C_i^{(p)})$ in terms of products, the computational efficiency $(E_i^{(p)})$ and the mean elapsed CPU time (CPU-Time). Computational cost and efficiency are calculated according to the corresponding expressions given by (22)–(28) using the values of parameters n, μ_0 and μ_1 as shown in the end of each problem and taking $l = 2.66$ in each case.

Table 3 Comparison between $E_i^{(4)}$ $(i = 1, 2)$, $E_j^{(5)}$ $(j = 1, 2, \ldots, 5)$ according to Theorem 3

Problem	μ_0	m_1	m_2	m_3	m_4	m_5	m_6	Comparison between efficiencies as per Theorem 3					
								(i)	(ii)	(iii)	(iv)	(v)	(vi)
1	136.93	100.18	269.71	147.08	138.08	136.25	140.91	$E_1^{(5)} < E_1^{(4)}$	$E_1^{(5)} > E_2^{(4)}$	$E_1^{(5)} > E_2^{(5)}$	$E_1^{(5)} > E_3^{(5)}$	$E_1^{(5)} < E_4^{(5)}$	$E_1^{(5)} > E_5^{(5)}$
2	1.5	10.14	15.79	32.98	15.98	8.49	15.15	$E_1^{(5)} > E_1^{(4)}$	$E_1^{(5)} > E_2^{(4)}$	$E_1^{(5)} > E_2^{(5)}$	$E_1^{(5)} > E_3^{(5)}$	$E_1^{(5)} > E_4^{(5)}$	$E_1^{(5)} > E_5^{(5)}$
3	2	26.16	26.18	85.62	52.62	25.81	36.47	$E_1^{(5)} > E_1^{(4)}$	$E_1^{(5)} > E_2^{(4)}$	$E_1^{(5)} > E_2^{(5)}$	$E_1^{(5)} > E_3^{(5)}$	$E_1^{(5)} > E_4^{(5)}$	$E_1^{(5)} > E_5^{(5)}$
4	78.74	114.65	199.15	261.66	208.66	142.70	158.36	$E_1^{(5)} > E_1^{(4)}$	$E_1^{(5)} > E_2^{(4)}$	$E_1^{(5)} > E_2^{(5)}$	$E_1^{(5)} > E_3^{(5)}$	$E_1^{(5)} > E_4^{(5)}$	$E_1^{(5)} > E_5^{(5)}$
5	96.88	193.73	271.84	473.78	392.78	244.01	266.67	$E_1^{(5)} > E_1^{(4)}$	$E_1^{(5)} > E_2^{(4)}$	$E_1^{(5)} > E_2^{(5)}$	$E_1^{(5)} > E_3^{(5)}$	$E_1^{(5)} > E_4^{(5)}$	$E_1^{(5)} > E_5^{(5)}$

The mean elapsed CPU time is calculated by taking the mean of 50 performances of the program, wherein we use $||\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}|| + ||\mathbf{F}(\mathbf{x}^{(k)})|| < 10^{-300}$ as the stopping criterion in single performance of the program.

From the numerical results, we can observe that like the existing methods the present method shows consistent convergence behavior. Calculated values of the computational order of convergence displayed in the third column of Table 2 verify the theoretical fifth order of convergence proved in Sect. 4. The existing method $\phi_4^{(5)}$ by Grau et al. converges to the solution with sixth order convergence in the second and third problems consisting of algebraic equations. In order to confirm that the results obtained in Table 2 are also in accordance with the results of Theorem 3, we find m_i, $(i = 1, 2 \cdots, 6)$ using the values of l, n, and μ_1 for each numerical problem. The results are presented in Table 3. Comparing the efficiency results of Table 2 with that of Table 3, we find that the results are compatible with each other for each numerical problem, and hence theorem 3 is verified. From the numerical values of the efficiency $(E_i^{(p)})$ and elapsed CPU Time (CPU-Time), we can observe that the method with large efficiency uses less computing time than the method with small efficiency. This shows that the efficiency results are in complete agreement with the CPU time utilized in the execution of program. Moreover, the results of efficiency and CPU-time confirm the robust and efficient nature of the new method.

7 Concluding remarks

In the foregoing study, we have developed a fifth-order iterative method for solving nonlinear equations. Scheme of the method is very simple which consists of three steps. Of these three steps the first step is Newton's step and last two are Newton-like steps. Hence, the name Newton-like method. Taylor's expansion is used to prove the local convergence order of the method. The computational efficiency is discussed exhaustively. Then a comparison between the efficiencies of new method with some existing methods is performed. It is shown that the proposed method is more efficient than existing methods, especially when applied for solving the systems of large dimensions. To illustrate the new technique five numerical examples are presented and completely solved. Computational results have justified robust and efficient convergence behavior of the proposed method. Similar numerical experimentations, carried out for a number of problems of different type, confirmed the above conclusions to a large extent.

References

1. Argyros, I.K.: Computational theory of iterative methods, series: studies in computational mathematics, vol. 15. Elsevier Publishing Company, New York (2007)
2. Cordero, A., Torregrosa, J.R.: Variants of Newton's method for functions of several variables. Appl. Math. Comput. **183**, 199–208 (2006)
3. Cordero, A., Torregrosa, J.R.: Variants of Newton's method using fifth-order quadrature formulas. Appl. Math. Comput. **190**, 686–698 (2007)
4. Cordero, A., Martínez, E., Torregrosa, J.R.: Iterative methods of order four and five for systems of nonlinear equations. J. Comput. Appl. Math. **231**, 541–551 (2009)

5. Cordero, A., Hueso, J.L., Martińez, E., Torregrosa, J.R.: A modified Newton-Jarratt's composition. Numer. Algor. **55**, 87–99 (2010)
6. Cordero, A., Hueso, J.L., Martińez, E., Torregrosa, J.R.: Increasing the convergence order of an iterative method for nonlinear systems. Appl. Math. Lett. **25**, 2369–2374 (2012)
7. Darvishi, M.T., Barati, A.: A third-order Newton-type method to solve system of nonlinear equations. Appl. Math. Comput. **187**, 630–635 (2007)
8. Darvishi, M.T., Barati, A.: A fourth-order method from quadrature formulae to solve systems of nonlinear equations. Appl. Math. Comput. **188**, 257–261 (2007)
9. Fousse, L., Hanrot, G., Lefvre, V., Plissier, P., Zimmermann, P.: MPFR: a multiple-precision binary floating-point library with correct rounding. ACM Trans. Math. Software. **33 (2) 15**, Art. 13 (2007)
10. Frontini, M., Sormani, E.: Third-order methods from quadrature formulae for solving systems of nonlinear equations. Appl. Math. Comput. **149**, 771–782 (2004)
11. Grau-Sánchez, M., Grau, À., Noguera, M.: On the computational efficiency index and some iterative methods for solving systems of nonlinear equations. J. Comput. Appl. Math. **236**, 1259–1266 (2011)
12. Grau-Sánchez, M., Noguera, M., Amat, S.: On the approximation of derivatives using divided difference operators preserving the local convergence order of iterative methods. J. Comput. Appl. Math. **237**, 363–372 (2013)
13. Homeier, H.H.H.: A modified Newton method with cubic convergence: the multivariable case. J. Comput. Appl. Math. **169**, 161–169 (2004)
14. Jarratt, P.: Some fourth order multipoint iterative methods for solving equation. Math. Comput. **20**, 434–437 (1966)
15. Jay, L.O.: A note on Q-order of convergence. BIT **41**, 422–429 (2001)
16. Kelley, C.T.: Solving Nonlinear Equations with Newton's Method. SIAM, Philadelphia, PA (2003)
17. Kung, H.T., Traub, J.F.: Optimal order of one-point and multipoint iteration. J. ACM **21**, 643–651 (1974)
18. Neta, B.: A new iterative method for the solution of systems of nonlinear equations. Z. Ziegler (ed.) Approximation Theory and Applications, pp. 249–263. Acdemic Press, New York (1981)
19. Noor, M.A., Waseem, M.: Some iterative methods for solving a system of nonlinear equations. Comput. Math. Appl. **57**, 101–106 (2009)
20. Ortega, J.M., Rheinboldt, W.C.: Iterative Solution of Nonlinear Equations in Several Variables. Academic Press, New York (1970)
21. Petković, M.S.: On a general class of multipoint root-finding methods of high computational efficiency. SIAM J. Numer. Anal. **49**, 1317–1319 (2011)
22. Petković, M.S., Neta, B., Petković, L.D., Džunić, J.: Multipoint Methods for Solving Nonlinear Equations. Elsevier, Boston (2013)
23. Potra, F.-A., Pták, V.: Nondiscrete Induction and Iterarive Processes. Pitman Publishing, Boston (1984)
24. Sharma, J.R., Guha, R.K., Sharma, R.: An efficient fourth order weighted-Newton method for systems of nonlinear equations. Numer. Algor. **62**, 307–323 (2013)
25. Sharma, J.R., Gupta, P.: An efficient fifth order method for solving systems of nonlinear equations. Comput. Math. Appl. **67**, 591–601 (2014)
26. Traub, J.F.: Iterative Methods for the Solution of Equations. Prentice-Hall, Englewood Cliffs, New Jersy (1964)
27. Wolfram, S.: The Mathematica Book, 5th edn. Wolfram Media, (2003)