Bangmod International Journal of
Mathematical and Computational Science
https://bangmodmcs.wordpress.com
ISSN : 2408-154X
Bangmod Int. J. Math. & Comp. Sci. Vol. 6, No. 1 & 2, 2020, 55 – 70

EFFICIENT TWO-STEPS ITERATIVE METHOD FOR SOLVING SYSTEM OF NONLINEAR EQUATIONS

Yau Balarabe Musa [1,*], **Muhammad Yusuf Waziri**[2], **Muhammad Aslam Noor**[3]
[1] *Department of Mathematics and computer Sciences, Faculty of Natural and Applied Science, Sule Lamido University, Kafin Hausa, Jigawa, Nigeria. Numerical Optimization Research Group, Bayero University, Kano, Nigeria.*
E-mails: yaumusa.jsu@gmail.com
[2] *Department of Mathematical Sciences, Faculty of physical Science, Bayero University Kano, Kano, Nigeria.*
E-mails: mywaziri.mth@buk.edu.ng
[3] *Department of Mathematics, COMSATS University Islamabad, Islamabad, Pakistan.*
E-mails: noormaslam@gmail.com.

**Corresponding author.*

Abstract In this paper, we present a new iterative method for solving nonlinear system of equations using Levenberg-Marquardt technique. The proposed method is a two-steps and proved to be globally convergent. The global convergence is achieved by incoperating the proposed algorithm with nonmonotone line search. The fourth-order of the scheme was obtained through computation of its computational order of convergence (COC). The strategy being a regularized method, solves most of the test functions that are singular in nature. Comparison for computation of efficiency index shows that the proposed method is robust in both classical or traditional efficiency index (TEI) and flops-like efficiency index (FEI). The convergence properties of the proposed method are also presented. In terms of less number of iterations and fast computing time, our proposed technique competes with other existing fourth-order methods nicely. Almost all the numerical performances conducted on some benchmark problems, have shown that the proposed algorithm is very efficient and promising.

MSC: 65H10, 65K05, 65F22, 65F35
Keywords: Nonlinear system of equations, Levenberg-Marquardt method, Regularization, Matrix norm, Traditional efficiency index, Flops-like efficiency index, Global convergence

Submission date: 23.05.2020 / Acceptance date: 13.12.2020 / Available online 31.12.2020

1. Introduction

We consider the system of nonlinear equatios,

$$\mathbf{F}(\mathbf{x}) = 0, \tag{1.1}$$

where

$$F(\mathbf{x}) = [\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3, ..., \mathbf{f}_n]^T \text{ and } \mathbf{x} = [x_1, x_2, ..., x_n]^T \in \mathbb{R}^n$$

is a nonlinear mapping and continuously differentiable. This problem is one of the cornerstones in computation and applied mathematics such as physics, engineering, technology, economics, industries, etc [2, 3, 5, 6, 13–15]

For decades, researchers were more interested in finding the approximate solution of (1.1), using Newton method and its invariants[1]. Halley[16] established a new exact and easy strategy for finding the roos of nonlinear equations. Traub[27] came up with iterative methods for solution of equations and was able to prove that the convergence order of the Newton's method is 2.
Waziri et al [28] achieved a local convergence by implementing a two-step derivative-free diagonally Newton method for large-scale nonlinear equations. Amini and Faramarz [2] used a modified two-steps Levenberg-Marquardt method where they achieved cubic convergence. Amini and Rostam [3] introduced a new nonmonotone third-order Armijor type line search which guarantees the global convergence of a modified Lenvenberg-Marquardt method for solving of nonlinear equations.

In recent times, higher-order iterative techniques have been introduced and successifully implemented in order to increase the rate of convergence. Alicia et al [12], have produced increasing the order of convergence of an iterative scheme and achieved higher-order of convergence. Artdiello et al [4], have developed design of higher-order iterative methods for nonlinear system of equations by using weight function procedure. Sharma and Guha [22] presented an efficient fourth-order Weighted-Newton method for nonlinear system of equations. Waziri et al[29] developed a new multi-step fixed newton method for solving large-scale systems of nonlinear equations. Singh [23] implemented an iterative strategy using three-step and achieved fourth-order convergence. The main draw back in the scheme is that at each step, one has to compute the Jacobian matrix and its inverse, which is itself a very difficult. We notice that if the Jacobian matrix is singular, then the method of Singh fails. To over come the defiency, we use the regularization technique of Levenberg-Marquardt method and regularized the Singh's scheme. We suggest a modified two-step method with nonmonotone line search for solving nonlinear system of equations. This nonmonotone line search is what guarantees the global convergence of our proposed method. Some examples are given to illustrate the efficiency of the regularized method. The global convergence of the proposed method is investigated. Comparison between Singh's and other existing fourth-order iterative methods shows that the performance of our method is better.

The paper is organized as follows. In Section 2, preliminaries and basic concepts were presented . In Section 3, we present our method and implementation of the proposeed algorithm. Global convergence of our method is analysed in section 4. in section 5, we give the numerical results in which our method is compared with Singh's scheme and other existing methods. Conclusion is presented in the last section.

2. Preliminaries and Basic concept.

Consider the system of nonlinear equation

$$\begin{aligned} &\mathbf{f}_1(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n) = 0, \\ &\mathbf{f}_2(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n) = 0, \\ &\quad \vdots \\ &\mathbf{f}_n(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n) = 0, \end{aligned} \tag{2.1}$$

where each function $\mathbf{f}_i, i = 1, 2, 3..., n$, maps a vector $\mathbf{x} \in \mathbb{R}^n$ of n-dimensional space $\mathbb{R}^n$ to the real line $\mathbb{R}$. The above system is representated in (1.1) and the components $f_i, i = 1, 2, 3, ..., n$ are coordinate functions of F. For the sake of simplicity, we denote the Jacobian matrix, by

$$J_k = \nabla F(\mathbf{x}^k).$$

The well known method for solving (1.1) is the Newton's method. The iterate of this strategy is given by

$$x^{k+1} = x^k - J_k^{-1} F(x^k), \tag{2.2}$$

it gives quadratic convergence for a sufficiently accurate starting point if J_k^{-1} exists. One of the weaknesses of the Newton's method is computation and inverting the Jacobian matrix at each eteration. Gauss-Newton method is a modification of Newton's method for finding the minimum of a function. The iterate is given as

$$x^{k+1} = x^k - (J_k^T J_k)^{-1} J_k^T F(x^k). \tag{2.3}$$

The Gauss-Newton method has an advantage that second derivatives which can be challenging, are never required[7].

The main shortcoming of the Gauss-Newton method is that when the computed Jacobian at each step becomes singuar, the algorithm stops. To overcome this problem, in 1944 [19], Kenneth Levenberg came up with an algorithm which solved the ill-possed problem of the system in both Newton and Gauss-Newton strategies. The classical Levenberg-Marquardt method computes the search directiond_k as

$$d_k = -(J_k^T J_k + \mu_k I)^{-1} J_k^T F(x^k), \tag{2.4}$$

where $\mu_k > 0$ is called the Levenberg-Marquardt parameter and I is the identity matrix of order $n \times n$

Definition 2.1. [23] (Computational order of convergence): Let β be a zero of the function $\mathbf{F}$ and suppose that x^{k-1}, x^k and x^{k+1} are three consecutive iterations close to β. Then, the computational order of convergence is approximated using the formula

$$p \approx COC = \frac{\log\left(\|(x^{k+1})-(x^k)\|/\|(x^k)-(x^{k-1})\|\right)}{\log\left(\|(x^k)-(x^{k-1})\|/\|(x^{k-1})-(x^{k-2})\|\right)}$$

Definition 2.2 (Global convergence)**.** An algorithm is said to be globally convergent if for any choosen initial point $\mathbf{x}^0$, the sequence $\{\mathbf{x}^k\}_{k=0}$ generated by the algorithm, converges to a point for which a necessary condition of optimality holds.

3. Method and Algorithm.

In this section, a two-steps scheme for solving system of nonlinear equations is presented.

$$\begin{aligned}
&\text{For } x \in \mathbb{R}^n,\\
&y^k = x^k + d_k\\
d_k = -\big(J_k^T J_k + \mu_k I\big)^{-1} J_k^T F(x^k), \qquad &\mu_k = \tfrac{\|J_k}{\|} 10k^k + \|F(x^k)^2\|,\\
&\tilde{d_k} = -\big(J(y^k)^T J(y^k) + \mu_k I\big)^{-1} J(y^k)^T F(x^k + d_k).\\
&x_{k+1} = x^k + \alpha_1 d_k + \alpha_2^2 \tilde{d}_k,
\end{aligned} \tag{3.1}$$

where α_1, α_2 positive real numbers.
In this paper, the scheme is denoted as ELM.
We now present the implementation of our proposed Algorithm 3.1.
Step 1. Input $x^0 \in \mathbb{R}^n$, $\mu_0 > 0$, $\delta \in [0, 2]$, $J_0 = I$, σ_1, σ_2, $\sigma_3 > 0$ and $\tau, \rho \in (0.1)$. Let $k := 0$
Step 2. If $\|J_k^T F(x^k)\| = 0$, then stop. Else compute d_k and $\tilde{d_k}$ by,

$$d_k = -\big(J_k^T J_k + \mu_k I\big)^{-1} J_k^T F(x^k), \qquad \mu_k = \frac{\|J_k\|}{10k^k + \|F(x^k)\|^2},$$

where, $||J_k||_F = \sqrt{tr(J_k^2)}$, and $tr(J_k^2)$, is the trace of the square of the matrix J_k,

$$\tilde{d_k} = -\big(J(y^k)^T J(y^k) + \mu_k I\big)^{-1} J(y^k)^T F(y^k). \qquad y^k = x^k + d_k.$$

Step 3. If

$$\|F(x^k + d_k + \tilde{d_k})\| \le \rho \|F(x^k)\|, \tag{3.2}$$

then $\alpha = 1$ and go to **Step** 5. Else go to **step**
Step 4. Compute $\alpha_k = \max\{1^0, \tau^1, \tau^2, ...,\}$ where $\alpha_k = \tau^i : i = 0, 1, 2, 3..., n$, satisfying

$$\begin{aligned}
&\|F(x^k + d_k + \tilde{d_k})\|^2 - \|F(x^k)\|^2\\
&\le -\sigma_1 \alpha^2 \|d_k\|^2 - \sigma_2 \alpha^2 \|\tilde{d_k}\|^2 - \sigma_3 \alpha^2 \|F(x^k)\|^2 + \epsilon \|F(x^k)\|^2,
\end{aligned}$$

where, $\{\epsilon_k\}$ is a given positive sequence such that

$$\sum_{k=0}^{\infty} \epsilon_k < \infty. \tag{3.3}$$

Step 5. Compute $F(y^k)$,

$$\begin{aligned}
x^{k+1} &= x^k - \big[\alpha_1 A F(x^k) + \alpha_2^2 B F(y^k)\big]\\
&= x^k + \alpha_1 d_k + \alpha_2^2 \tilde{d}_k
\end{aligned} \tag{3.4}$$

where,

$$A = \big(J_k^T J_k + \mu_k I\big)^{-1} J_k^T \text{ and } B = \big(J(y^k)^T J(y^k) + \mu_k I\big)^{-1} J(y^k)^T$$

Step 6. Let $k = k + 1$ and go to **Step** 2

Remark 3.1. It follows this

1. We observe that as $\alpha \to 0^+$, the left-hand side of the inequality (3.3) goes to zero but the right-hand tends to $\epsilon_k\|F(x^k)\|$ which is positive. The algorithm is well defined since for any sufficiently $\alpha > 0$, the inequality (8) is satisfied.

2. The parameter $\mu_k = \frac{\|J_k\|}{10k^k+\|F(x^k)\|}$ used in our proposed algorithm was suggested from our parameter derived in [30].

4. Convergence Analysis.

Definition 4.1. Define the level set $L_0 = \{x : \left\|F(x^k)\right\| \leq \epsilon^{\frac{1}{2}} \left\|F(x^0)\right\|\}$,
where ϵ is a positive constant such that

$$\sum_{k=0}^{\infty} \epsilon_k \leq \epsilon < \infty. \tag{4.1}$$

Lemma 4.2. $\big[[13], \text{Lemma3.3}\big]$: *Let $\{\alpha_k\}$ and $\{r_k\}$ be two positive sequences satisfying*

$$\alpha_{k+1} \leq (1+r_k)\alpha_k + r_k \text{ and } \sum_{k=0}^{\infty} r_k < \infty. \text{ Then } \{\alpha_k\} \text{ converges.}$$

Lemma 4.3. $[[20], \text{Lemma2.1}]$. *Let the sequence $\{x^k\}$ be generated by Algorithm 3.1, then the sequenc $\{\|F_k\|\}$ converges and $x \in L_0$ for all $k \geq 0$.*

Proof. The Proof is similar to that of Lemm 4.2, but for the sake of completeness we write the proof. Now, from (3.2) and (3.3)

$$\|F(x^k + d_k + \tilde{d_k})\| \leq \rho\|F_k\| \tag{4.2}$$

and

$$\|F(x+\alpha d_k + \alpha^2\tilde{d_k})\|^2 - \|F_k\|^2 \leq -\sigma_1\alpha^2\|d_k\|^2 - \sigma_2\alpha^2\|\tilde{d_k}\| - \sigma_3\alpha^2\|F_k\|^2 + \epsilon_k\|F_k\|^2 \tag{4.3}$$

where $\{\epsilon_k\}$ is a given positive sequence such that

$$\sum_{k=0}^{\infty} \epsilon_k < \infty, \tag{4.4}$$

, we have,

$$\|F(x^{k+1})\| \leq (1+\epsilon_k)\|F(x^k)\|^2. \tag{4.5}$$

Inequalities (4.1), (4.2) and Lemma 4.2, imply that the sequence $\{\|F(x^k)\|^2\}$ converges. Hence, the sequence $\{\|F(x^k)\|\}$ also converges. Similarly, from definition 4.1 and (4.1), we can work out the following

$$\begin{aligned}\|F(x^{k+1})\| &\leq (1+\epsilon_k)^{\frac{1}{2}}\|F(x^k)\| \leq ... \leq \Pi_{i=0}^{k}(1+\epsilon_k)^{\frac{1}{2}}\|F(x^0)\| \\ &\leq \sum_{i=0}^{k}\frac{1}{k+1}(1+\epsilon_k)^{\frac{k+1}{2}}\|F(x^0)\| \\ &\leq (1+\tfrac{\epsilon}{k+1})^{\frac{k+1}{2}}\|F(x^0)\| \\ &\leq e^{\frac{\epsilon}{2}}\|F(x^0)\|.\end{aligned}$$

The above inequality implies that $x \in L_0$ for all k. Therefore, the proof is complete.

By Lemma 4.3, it can clearly be shown that the sequence$\{\|F(x^k)\|\}$ is bounded. Meaning, there exists a constant $M > 0$ such that

$$\|F(x^k)\| \leq M, \forall k \geq 0. \tag{4.6}$$

Now, we use the following definition as our assumption.

Definition 4.4. The matrix J_k and the function F are said to be Lipschitz continuous if there exists positive constants L_1 and L_2 called Lipschitz constants such that

$$\|\nabla F(y) - \nabla F(x)\| \leq L_1\|y - x\|, \forall x, y \in \mathbb{R}^n \tag{4.7}$$

and

$$\|F(y) - F(x)\| \leq L_2\|y - x\|, \forall x, y \in \mathbb{R}^n. \tag{4.8}$$

Since The matrix J_k is a linear operator, we have,

$$\|J_k\| \leq L, \forall x \in \mathbb{R}^n, \tag{4.9}$$

and some $L > 0$. We now present the global convergence of our algorithm.

Definition 4.5. Suppose that $F(x)$ and $\nabla F(x)$ in (4.7) and (4.8) hold. Then Algorithm 3.1 terminates in finite iteration or satisfies

$$\lim_k \inf \|J_k^T F(x^k)\| = 0. \tag{4.10}$$

Proof. We can use contradiction to prove the theorem. Suppose it is not true, it implies that the exists an integer $\tilde{k}$ such that

$$\|J_k^T F(x^k)\| \geq \tau, \forall k \geq \tilde{k}. \tag{4.11}$$

and this implies that

$$\|F(x^k)\| \geq \tau_1, \tag{4.12}$$

for sufficiently large k, the inequality holds for some positive constant τ_1.

If equation (3.2) holds for in fine k, then $\|F(x^k)\| \to 0$, which is a contradiction to (4.12). By (3.2) and (3.3), we have,

$$\begin{aligned}\|F(x^{k+1})\| &\leq \Pi_{i\in G_k}(1+\epsilon_i)^{\frac{1}{2}}\Pi_{i\in H_k}\rho\|F(x^0)\| \\ &= \Pi_{i\in G_k}(1+\epsilon_i)^{\frac{1}{2}}\rho^{|H_k|}\|F(x^0)\| \\ &\leq e^{\frac{\epsilon}{2}}\rho^{|H_k|}\|F(x^0)\| \to 0, \text{ as } k \to \infty.\end{aligned}$$

We will assume that (3.2) holds for only finite k. Thus, it can clearly be obtain from (3.3) that

$$\sum_{k=0}^{\infty}\alpha_k^2\|d_k\|^2 < \infty, \quad \sum_{k=0}^{\infty}\alpha_k^2\|\tilde{d}_k\|^2 < \infty \text{ and } \quad \sum_{k=0}^{\infty}\alpha_k^2\|F(x^k)\|^2 < \infty.$$

Therefore, these strong inequalities imply that

$$\sum_{k=0}^{\infty}\alpha_k\|d_k\| = 0, \quad \sum_{k=0}^{\infty}\alpha_k\|\tilde{d}_k\| = 0 \text{ and } \quad \sum_{k=0}^{\infty}\alpha_k\|F(x^k)\| = 0.$$

The inequalities above and (4.12) yeild

$$\lim_{k=0}^{\infty}\alpha_k = 0. \tag{4.13}$$

Now, set $\bar{\alpha} = \frac{\alpha}{r}$. Thus, from (3.3), we have

$$\begin{aligned}
&\|F(x^k + \bar{\alpha_k} d_k + \bar{\alpha_k}^2 \tilde{d_k})\|^2 - \|F(x^k)\|^2 \\
&\geq -\sigma_1 \bar{\alpha}_k^2 \|d_k\|^2 - \sigma_2 \bar{\alpha}_k^2 \|\tilde{d}_k\|^2 - \sigma_3 \bar{\alpha}_k^2 \|F(x^k)\|^2 + \epsilon_k \|F(x^k)\|^2 \\
&\geq -\sigma_1 \bar{\alpha}_k^2 \|d_k\|^2 - \sigma_2 \bar{\alpha}_k^2 \|\tilde{d}_k\|^2 - \sigma_3 \bar{\alpha}_k^2 \|F(x^k)\|^2,
\end{aligned}$$

Thus, we have

$$\begin{aligned}
&\bar{\alpha_k}\big(\sigma_1 \|d_k\|^2 + \sigma_2 \|\tilde{d}_k\|^2 + \sigma_3 \|F(x^k)\|^2\big) \\
&\geq -\big(\|F(x^k + \bar{\alpha_k} d_k + \bar{\alpha_k}^2 \tilde{d_k})\|^2 - \|F(x^k)\|^2\big) \\
&= -\big(2F(x^k)^T \big(F(x^k + \bar{\alpha_k} d_k + \bar{\alpha_k}^2 \tilde{d_k}) - F(x^k)\big) \\
&\quad + \left\|F(x^k) + \bar{\alpha_k} d_k + \bar{\alpha_k}^2 \tilde{d_k}) - F(x^k)\right\|^2 \big) \\
&\geq -2F(x_k)^T \big(F(x^k + \bar{\alpha_k} d_k + \bar{\alpha_k}^2 \tilde{d_k}) - F(x^k)\big) - \eta_1 \bar{\alpha_k}^2 \big(\|d_k\|^2 + \left\|\tilde{d_k}\right\|^2 \big),
\end{aligned}$$

where η_1 is a constant and (4.14) uses (4.7), the since $\bar{\alpha_k} < \frac{1}{r}$.

The following expression can be estimated. $F(x^k)^T \big(F(x^k + \bar{\alpha_k} d_k + \bar{\alpha_k}^2 \tilde{d_k}) - F(x^k)\big)$. It is worth noting that

$$\begin{aligned}
&F(x^k)^T \big(F(x^k + \bar{\alpha_k} d_k + \bar{\alpha_k}^2 \tilde{d_k}) - F(x^k)\big) \\
&= F(x^k)^T \big(F(x^k + \bar{\alpha_k} d_k + \bar{\alpha_k}^2 \tilde{d_k}) - F(x^k) - F(x^k)\big(x^k + \bar{\alpha_k} \tilde{d_k}\big) \\
&\quad + F(x^k)^T \big(F(x^k + \bar{\alpha_k} d_k) - F(x^k)\big) \\
&\leq LM \bar{\alpha_k} \|\tilde{d_k}\| + F(x^k)^T J_k \bar{\alpha_k} d_k + F(x^k)^T \int_0^1 \big(J(x^k + t\bar{\alpha_k} d_k) - J_k\big) \\
&\leq 2LM \bar{\alpha}^2 \|\tilde{d_k}\| - \bar{\alpha_k} d_k^T \big(J_k^T J_k + \mu_k I\big) d_k.
\end{aligned} \tag{4.14}$$

The first inequality comes from (13) and (15), while the last one from (6) and (14). Similarly, from (21) up to (22) means there exists a positive constant η_2 such that

$$\bar{\alpha_k} = \frac{d_k^T \big(J_k^T J_k + \mu_k I\big) d_k}{\eta_2 \big(\|d_k\|^2 + \|\tilde{d_k}\|^2 \|F(x^k)\|^2 + \|\tilde{d_k}\|} \geq \frac{\mu_k d_k^T d_k}{\eta_2 \big(\|d_k\|^2 + \|\tilde{d_k}\|^2 \left\|F(x^k)\right\|^2 + \|\tilde{d_k}\|}. \tag{4.15}$$

Now, on the boundedness of d_k, we use the SVD of the matrix J_k. Let the SVD of the Jacobiam matrix be
$J_k = UDV^T$,
where U and V are two orthorgonal matrices and D is a diagonal matrix with nonnegative diagonal enteries which are its singular values, $\sigma_i \geq 0; i = 1, 2, 3, ..., n$. Then, we have

$$\begin{aligned}
\left\|\big(J_k^T J_k + \mu_k I\big)^{-1}\right\| &= \|V(D^2 + \mu_k I)^{-1} V^T\| \\
&= \|(D^2 + \mu_k I)^{-1}\| \\
&= \max_{i=1,2,3,\ldots,n} \big(\sigma_i^2 + \mu_k I\big)^{-1} \\
&\leq \mu_k^{-1}.
\end{aligned}$$

Now, from (15), (16), **Step 2** of the algorithm and the above inequality, we have

$$\begin{aligned}\|d_k\| = \|(J_k^TJ_k+\mu_kI)^{-1}J_k^TF(x^k)\| \leq \left\|\left(J_k^TJ_k+\mu_kI\right)^{-1}\right\| \|J_k^T\| \|F(x^k)\| \leq LM\mu_k^{-1}\\ \leq LM\left(10k^k+M^2\right)\end{aligned} \tag{4.16}$$

where

$$\mu_k^{-1} = \frac{1ok^k + \|F(x^k)\|^2}{\|J_k\|}. \tag{4.17}$$

Similarly, we can obtain from (3.1), (4.8) and (4.12) that

$$\begin{aligned}\left\|\tilde{d}_k\right\| &= \|(J_k^TJ_k+\mu_kI)^{-1}J_k^TF(y^k)\|\\ &\leq \|(J_k^TJ_k+\mu_kI)^{-1}J_k^TF(y^k)-F(x^k)\| + \|(J_k^TJ_k+\mu_kI)^{-1}J_k^TF(x^k)\|\\ &\leq L^2\mu_k^{-1}\|d_k\| + \|d_k\| \leq \left(L(10k^k+M^2\right)\|d_k\|.\end{aligned} \tag{4.18}$$

Thus, if $\lim\inf_{k\to\infty}\|d_k\| = 0$, then we have from **Step 2** and (13) that

$$\lim\inf_{k\to\infty}\|J_k^TF(x^k)\| = \lim\inf_{k\to\infty}\|(J_k^TJ_k+\mu_kI)d_k\| = 0.$$

This is a contradiction to (18). Hence, there exists a contant τ_2 such that $\lim\inf_{k\to\infty}\|d_k\| \geq \tau_2$, which in addition to (4.6), (4.12), (4.16) and (4.18) imply that there exists a contradiction to (4.13). Hence the proof is complete.

5. Numerical Experiments.

In this section, we present some numerical experiments to show the robustness of our strategy in solving system of nonlinear equations. All the numerical results reported in this section, were computed and implemented in MATLAB R2014a 8.3.0.532 (64-bit) and tested on a 64-bit 1.73 GHz Q740 core 17 processor and 4GB RAM window 10. We tested our algorithm on numerical problems stated below and comparisons were made with existing methods used in [23]. We compare Algorithm 3.1 denoted as ELM, OM_4 [23], $M_{4.1}$[22] and M44[25] The main stopping criterion is $\|J_k^TF_k\| \leq 10^{-8}$. If the iteration $k \geq 1000$, the test failed and is denoted as '–'. The parameters in the algorithm are set as $\mu_0 = 0.5$, $\epsilon = 10^{-8}$, $\sigma_1 = 10^{-4}$, $\sigma_2 = 0.25$, $\sigma_3 = 0.75$, $\tau = 0.05$, $\rho = 0.02$. The numerical results obtained are depicted inTables 1 , 2, 3 and 4. The meaning of the notations used in the table 1, are stated as follows. "P": stands for Benchmark problem. $'x0'$: means initial starting point. "n": stands for dimension of the problem. $''\#iter''$ is the total number of iterations. cpu: stands for processing time recorded in seconds. "x": Solution reached by a given solver. $\|J_k^TF_k\|$: Stands for the value at which the solution is reached at point x_k . NaN: indicates that the method diverges. '–': Shows that the iterations is more than 1000 and no solution is found."s": Means the given solver fails due to singularity i.e the Jacobian matrix generated is singular. In Table 2, COC: stands for computational order of convergence. TNFE: Stands for total number of function evaluations for both F and F'. In Tabble 4, TEI: Stands for Traditional Efficiency Index and FEI, means Flops-like

Efficiency index. In the figures, $M_{1.4}, M_{4.4}$ and OM_4 are represented as M14, M44 and OM4, respectively.

5.1. Nonlinear system of equations

In Table 1, we use the following nonlinear system of equations for comparing ELM and other fourth-order existing methods.

F1: Rosenbrock function (n is even) [More et al., 1981].

$$f_{2i-1}(x) = 10(x_{2i} - x_{i-1}^2)$$
$$f_{2i}(x) = 1 - x_{2i-1}^2$$
$$x_0 = (1, -1, 1, ..., 1)$$
$$i = 1, 2, 3, ..., n/2$$

F2: Strictly convex function I [Raydan, 1997].

$$f_i(x) = \sqrt{(e^{x_i} - x_i)}$$
$$x_0 = (1/n, 2/n, 3/n, ..., 1)$$
$$i = 1, 2, 3, ..., n.$$

F3: Strictly convex function II [Raydan, 1997].

$$f_i(x) = \tfrac{i}{10}(e^{x_i} - x_i),$$
$$x_0 = (1, 1, 1, ..., 1)$$
$$i = 1, 2, 3, ..., n.$$

F4: Trigonometric diagonal exponential function [Luksan and vicek, 2003].

$$h = 1/(n+1)$$
$$f_1(x) = x_{1)}e^{(cos(h(x_1+x_2)},$$
$$f_i(x) = x_i - e^{(cos(hx_{i-1}+x_i+x_{i+1})},$$
$$i = 2, 3, 4, ..., n-1$$
$$x0 = (1.5, 1.5, ..., 1.5)$$

F5: Trigonometric exponential function [Luksan and vicek, 2003].

$$f_1(x) = 3x_1^3 + 2x_2 - 5 + sin(x_1 - x_2)sin(x_1 + x_2),$$
$$f_i(x) = 3x_i^3 + 2x_{i+1} - 5 + sin(x_i - x_{i+1})sin(x_i + x_{i+1}) + 4x_i - x_{i-1}e^{(x_{i-1}-x_i)},$$
for $1 < i < n$
$$f_n(x) = 4x_n - x_{n-1}e^{(x_{n-1}-x_n)} - 3$$
$$x_0 = (4, 4, 4, ..., 4)$$
$$i = 1, 2, 3, ..., n.$$

F6: Exponential function [La Cruz et al., 2004]

$$f_1(x) = e^{x_1-1}.$$
$$f_i(x) = i(e^{x_1-1}),$$
$$x_0 = (0.5, 0.5, 0.5, ..., 0.5).$$
$$i = 2, 3, ..., n.$$

F7: Singular Broyden funtion [Luksan and Vicek, 2003.]

$$f_1(x) = ((3 - 2x_1)x_1 - 2x_2 + 1)^2$$
$$f_i(x) = ((3 - 2x_i)x_i - x_{i-1} - 2x_{i+1} + 1)^2,$$
$$f_n(x) = ((3 - x_n)x_n - x_{n-1} + 1)^2$$
$$x_0 = (-1, -1, -1, ..., -1).$$
$$i = 2, 3, ..., n-1.$$

F8: Trigonometric function [More et al., 1981].

$$f_i(x) = n - \sum_{j=1}^{n} cos(x_j) + i(1 - cos(x_i)) - sin(x_{i,}),$$
$$i = 2, 3, ..., n.$$
$$x_0 = (1 - 1/n, 1 - 2/n, 1 - 3/n, ...0)$$

9. F9: Vector function(Generalized function of Spedicator).

$$f_i(x) = \sum_i^n (x_i^2 sin(x_i)) - x_i^4 + sin(x_i^2),$$
$$i = 2, 3, ..., n.$$
$$x_0 = (0.5, 0.5, 0.5, ..., 0.5).$$

10. F10: Brown and Dennis function.

$f_i(x) = (x_i + (i/5)x_{i+1} - e^{(i/5)^2}) + x_{i+2} + x_{i+3}sin(i/5) - cos(i/5)^2$, for $i = 1$

$$x0 = (15, 5, -5, -1)$$

Table 1. Numerical results of M41, M44, OM1 and ELM for problem F1-F10

			M41				M44				OM4				ELM			
P	n	x0	*#iter*	cpu	x'	$\|J_k^T F(x^k)\|$	#iter	cpu	x'	$\|J_k^T F(x^k)\|$	#iter	cpu	x'	$\|J_k^T F(x^k)\|$	#iter	cpu	x'	$\|J_k^T F(x^k)\|$
	10		15	0.007838	5.25e-04	3.34e-10	21	0.016049	7.10e-04	8.43e-10	3	0.002266	4.74e-07	8.18e-17	3	0.007774	5.4ee-04	5.08e-10
	100		16	0.084444	3.61e-04	3.43e-10	22	0.145501	4.90e-04	8.68e-10	3	0.018782	4.74e-07	2.59e-16	3	0.122468	2.49e-04	1.56e-10
F1	200	$(1, -1, 1, ..., 1)^t$	16	0.355277	3.61e-04	4.85e-10	23	0.543634	3.30e-04	4.00e-10	3	0.063229	4.74e-07	3.66e-16	3	0.433151	2.77e-04	3.03e-10
	300		16	0.918445	3.61e-04	5.94e-10	23	1.409837	3.30e-04	4.89e-10	4	0.189492	4.74e-07	4.48e-16	4	1.145931	3.24e-04	5.90e-10
	500		16	3.960224	3.61e-04	7.67e-10	23	5.751119	3.30e-04	6.32e-10	5	0.806104	4.74e-07	5.78e-16	4	5.991453	2.28e-04	2.68e-10
	10		720	0.381068	NaN	Na	10	0.006132	1.69e-11	3.70e-10	5	0.002747	4.08e-15	1.26e-14	5	0.00798	-1.90e+00	5.04e-14
	100		738	4.167025	NaN	NaN	9	0.050089	7.64e-11	7.64e-10	5	0.018901	1.00e-13	1.03e-12	5	0.004524	4.16e-11	1.32e-11
F2	200	$(1/n, 2/n, 3/n, ..., 1)^t$	744	14.56816	NaN	NaN	9	0.215749	3.70e-11	5.33e-10	5	0.066605	2.75e-13	3.89e-13	5	0.047856	2.61e-11	2.61e-11
	300		748	42.22197	NaN	Na	9	0.560666	2.22e-11	3.90e-10	5	0.490845	1.66e-13	2.88e-13	5	0.459225	8.46e-12	1.47e-11
	500		751	188.6564	NaN	NaN	9	2.30e+00	2.2804e-10	3.35e-10	6	1.197088	5.07e-14	8.85e-14	5	1.732335	5.68e-13	1.27e-11
	10		1	S	S	S	1	S	S	S	1	S	S	S	8	0.041183	-1.13	5.04e-14
	100		1	S	S	S	1	S	S	S	1	S	S	S	113	1.423668	-1.13	1.13e-14
F3	200	$(1, 1, 1, ..., 1)t$	1	S	S	S	1	S	S	S	1	S	S	S	NaN	NaN	NaN	NaN
	300		1	S	S	S	1	S	S	S	1	S	S	S	NaN	NaN	NaN	NaN
	500		1	S	S	S	1	S	S	S	1	S	S	S	NaN	NaN	NaN	NaN
	10		NaN	NaN	NaN	NaN	10	0.016046	2.41	7.12e-10	5	0.015913	2.41	6.65e-10	5	0.007954	2.48	9.06e-12
	100		NaN	NaN	NaN	NaN	12	0.120581	2.71	1.52e-10	5	0.049925	2.71	7.58e-11	5	0.342597	2.71	3.19e-15
F4	200	$(1.5, 1.5, 1.5, ..., 1.5)t$	NaN	NaN	NaN	NaN	12	0.79738	2.71	2.41e-10	6	0.292601	2.71	6.89e-11	7	2.625341	2.71	4.30e-14
	300		NaN	NaN	NaN	NaN	12	2.167812	2.71	3.02e-10	7	0.769334	2.71	3.36e-12	7	10.00061	2.71	2.30e-14
	500		NaN	NaN	NaN	NaN	12	6.606776	2.71	3.94e-10	7	1.692597	2.71	2.79e-10	8	43.95724	2.71	6.28e-16
	10		1	S	S	S	1	S	S	S	1	S	S	S	7	0.01243	1	9.92e-10
	100		1	S	S	S	1	S	S	S	1	S	S	S	7	0.253455	1	6.80e-10
F5	200	$(4, 4, 4, ..., 4)^t$	1	S	S	S	1	S	S	S	1	S	S	S	7	0.535404	1	3.79e-14
	300		1	S	S	S	1	S	S	S	1	S	S	S	7	1.036635	1	5.87e-10
	500		1	S	S	S	1	S	S	S	1	S	S	S	8	4.125789	1	3.68e-14
	10		1	S	S	S	1	S	S	S	1	S	S	S	17	0.057884	1	8.44e-10
	100		1	S	S	S	1	S	S	S	1	S	S	S	65	53.16407	1	8.44e-10
F6	200	$(0.5, 0.5, 0.5, ..., 0.5)t$	1	S	S	S	1	S	S	S	1	S	S	S	129	321.4642	1	7.05e-10
	300		1	S	S	S	1	S	S	S	1	S	S	S	–	–	–	–
	500		1	S	S	S	1	S	S	S	1	S	S	S	–	–	–	–
	10		NaN	NaN	Nan	NaN	11	0.005716	1.50e-11	4.75e-10	5	0.003026	1.87e-12	5.78e-12	5	0.004739	1.16e-16	3.72e-14
	100		NaN	NaN	NaN	NaN	12	0.06566	1.80e-11	1.88e-10	5	0.028834	1.87e-11	1.83e-11	5	0.076151	6.62e-16	6.66e-15
F7	200	$(-1, -1, -1, ..., -1)t$	NaN	NaN	NaN	NaN	12	0.241737	1.87e-11	2.65e-10	6	0.109214	1.87e-11	2.59e-11	5	0.199589	2.78e-15	3.14e-15
	300		NaN	NaN	NaN	NaN	12	0.683208	1.87e-11	3.25e-10	6	0.298054	1.82e-12	3.17e-11	6	0.83209	1.45e-16	3.85e-15
	500		NaN	NaN	NaN	NaN	12	2.943035	1.87e-11	4.19e-10	7	1.273314	1.82e-12	4.09e-11	6	3.914426	5.80e-16	1.29e-13
	10		1	S	S	S	1	S	S	S	1	S	S	S	11	0.015858	3.08e-02	1.15e-13
	100		1	S	S	S	1	S	S	S	1	S	S	S	13	0.277447	3.88e-04	3.68e-11
F8	200	$(1 - 1/n, 1 - 2/n, 1 - 3/n, ...0)t$	1	S	S	S	1	S	S	S	1	S	S	S	13	0.764857	2.59e-13	1.49e-11
	300		1	S	S	S	1	S	S	S	1	S	S	S	13	3.132658	4.40e-05	2.21e-11
	500		1	S	S	S	1	S	S	S	1	S	S	S	17	18.49659	1.59e-05	3.02e-10
	10		11	0.009591	4.01e-04	6.01e-10	9	0.025303	3.60e-04	4.43e-10	4	0.004237	7.57e-04	7.58e-16	5	0.007645	2.43e-04	1.79e-10
	100		11	0.1808	2.73e-04	6.41e-10	11	0.148885	2.49e-04	4.85e-10	5	0.076537	2.78E-04	1.29e-11	5	0.136648	2.47-04	6.31e-10
F9	200	$(0.5, 0.5, 0.5, ..., 0.5)t$	13	0.562116	2.52e-04	7.47e-10	12	0.551576	2.16e-04	4.72e-10	5	0.159649	1.76e-04	4.37e-12	5	0.421302	2.50e-04	9.94e-10
	300		14	1.505676	2.14e-04	5.88e-10	12	1.369161	2.50e-04	9.73e-10	5	0.382522	4.67e-05	6.15e-11	6	1.138028	1.31e-04	1.75e-10
	500		14	5.758127	2.02e-04	7.05e-10	13	5.613862	2.12e-04	8.20e-10	6	1.333342	1.18e-04	9.79e-10	6	3.03265	1.35e-04	2.68e-10
	10		–	–	–	–	11	0.007509	1.00e+00	9.85e-10	–	–	–	–	6	0.005128	1.00e+00	3.32e-11
	100		–	–	–	–	12	0.063294	1.00e+00	3.89e-10	–	–	–	–	7	0.060971	1.00e+00	7.11e-14
F10	200	$(15, 5, -5, -1)t$	–	–	–	–	12	0.277783	1.00e+00	5.50e-10	–	–	–	–	7	0.219662	1.00e+00	1.58e-10
	300		–	–	–	–	12	0.722181	1.00e+00	6.74e-10	–	–	–	–	8	0.642896	1.00e+00	1.38e-13
	500		–	–	–	–	12	3.059641	1.00e+00	8.70e-10	–	–	–	–	9	2.923045	1.00e+00	2.76e-12

The reported numerical results in Table 1 above, indicate that OM4 and ELM compete in terms of less number of iteration and cpu-time whlie solving F1 and F2, while M41 diverges for F2 . Methods M14, M44 and OM4 fail to solve F3, F5, F6 and F8 due to singularity attained by their respective Jacobians at initial stage. M41 fails to solve F2, F4 , F7 and F9 due to divergence. Our proposed method imerges superior with regards to those function where the other three methods fail due to singularity.

5.2. Performance Profile

Below are the figures indicating the performances of our proposed method in

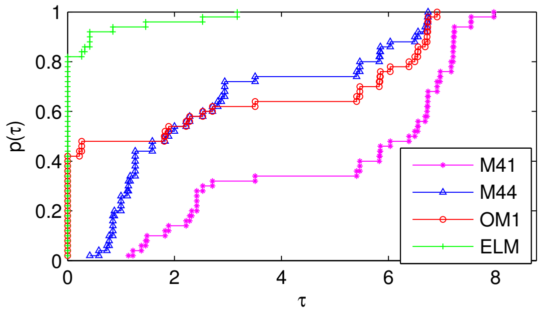

Figure 1. Performance profile of M41, M44, OM4 and ELM. methods with respect to number of iterations for problem 1-10

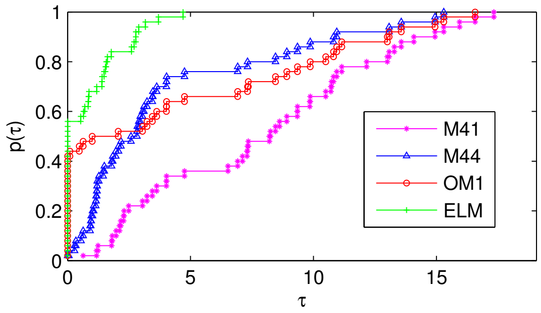

Figure 2. Performance profile of M41, M44, OM4 and ELM. methods with respect to cpu-time for problem 1-10

In Table 2, we used the following nonlinear test functions:
P1: Corresponds to Example 4.1.2[23]. P2: Corresponds to Example 4.1.3[23].
P3: Nonlinear problem:

$$f_1(x) = 2x_1^2 + sin(x_1) - 1,$$
$$f_i(x) = -2x_{i-1}^2 + 2x_i + 2sin(x_i) - 1,$$
$$i = 2, 3, 4, ..., n-1,$$
$$x0 = (1, 1, ...)$$

P4: Exponential trigonometric function:

$$\begin{aligned}
&h = 1/(n+1),\\
&f_1 = x_1 e^{cos(h(x_1+x_2))},\\
&f_i(x) = x_i - e^{cos(h(x_{(i-1}+x_i+x_{i+1}))},\\
&f_n(x) = x_n - e^{cos(h(x_{n-1}+x_n))}\\
&i = 2, 3, 4, ..., n-1\\
&x_0 = (1.5, 1.5, ...).
\end{aligned}$$

TABLE 2. Comparison of M41, M44, OM1 and ELM as fourth-order iterative methods for problem P1-P4

Method	x0	$\#Iter$	$\|F(x^1)\|$	$\|F(x^2)\|$	$\|F(x^3)\|$	COC	TNFE
P1:							
M4.1	(0.5,0.5,0.5)	4	2.00e-02	2.00e-07	1.00e-35	3.9949	76
M4.4		4	1.00e-02	1.00e-10	7.00e-43	3.9955	76
OM4		4	4.00e-02	3.00e-08	1.00e-32	3.9931	78
ELM		4	8.90e-01	8.636-07	1.00e-31	3.9954	78
P2:	(1.2,-1.8,0.1)						
M4.1		6	1.00e-02	7.00e-08	1.00e-37	3.9764	46
M4.4		6	1.00e-03	7e-08	1.00e-37	3.9624	46
OM4		5	1.00e-03	1.00e-11	2.00e-49	4.0056	52
ELM		5	1.00e-04	1.00e-12	2.00e-52	4.0043	54
P3:	(1,1,1)						
M4.1		6	1.00e-01	6.00e-08	1.00e-34	3.9751	62
M4.4		6	2.00e-02	4.00e-08	2.00e-34	3.9633	62
OM4		5	1.00e-03	2.00e-10	2.00e-50	3.9986	64
ELM		5	1.00e-03	1.00e-11	1.00e-52	3.9994	64
P4:	(1.5,1.5,1.5)						
M4.1		7	1.00e-02	3.00e-06	2.00e-35	3.9994	46
M4.4		6	2.00e-02	6.00e-09	7.00e-43	3.9993	46
OM4		5	3.00e-02	2.00e-07	1.00e-32	4.0046	52
ELM		5	6.50e-01	3.00e-09	2.00e-31	4.0043	54

5.3. COMPUTATIONAL EFFICIENCY INDEX.

The well known methods or techniques mostly used for computing the computational efficiency index are the Classical Efficiency Index or Traditional Efficiency Index and Flops-like Efficiency Index, denoted by TEI and FEI respectively. The two are known famous techniques or efficiency index for nonlinear ystem of equations. (I) Classical or traditional efficiency index is defined by TEI= $p^{\frac{1}{v}}$, where p is the order of convergence and v is the total computation cost per iteration in terms of number of function evaluations. (II) The Flops-like efficiency index [23], FEI= $p^{\frac{1}{V}}$, where V represents the total computational cost per iteration along with LU decomposition having solution of two triangular system(based on flops).

Table 3, below, summarizes the requirements for each solver and comparison of the efficiency indices of the four methods.

TABLE 3. Comparison Efficiency Indices of different methods.

Iterative methods	M41	M44	OM1	ELM
Number of steps:	2	2	3	2
Rate of convergence:	4	4	4	4
Number of function evaluations:	$n + n^2$	$n + n^2$	$3n + n^2$	$2n + 2n^2$
Traditional Efficiency Index(TEI):	$4^{1/(n+2n^2)}$	$4^{1/(n+2n^2)}$	$4^{1/(3n+n^2)}$	$4^{1/(2n^2+2n)}$
Flops-like Efficiency index(FEI):	$4^1/((10n3/3) + 4n^2 + n)$	$4^{1/((7n^3/3)+4n^2+n)}$	$4^1/((2n^3/3) + 7n^2 + 3n)$	$4^{1/((2n3/3)+8n^2+2n)}$

TABLE 4. Traditional and Flops Efficiency Index for the four methods from $n = 2$ to $n = 20$.

Variable	M41		M44		OM1		ELM	
n	TEI	FEI	TEI	FEI	TEI	FEI	TEI	FEI
2	1.15e+00	1.03e+00	1.15e+00	1.02e+00	1.15e+00	1.04e+00	1.12e+00	1.03e+00
3	1.91e-01	3.10e-02	1.91e-01	3.92e-02	2.22e-01	1.03e-01	1.67e-01	3.03e-02
4	1.11e-01	1.42e-02	1.11e-01	1.84e-02	1.11e-01	5.88e-02	1.00e-01	1.40e-02
5	7.27e-02	7.70e-02	7.27e-02	1.01e-02	1.00e-01	3.81e-02	6.67e-02	7.60e-03
6	5.13e-02	4.60e-03	5.13e-02	6.10e-03	7.41e-02	9.70e-03	4.76e-02	4.60e-03
7	3.81e-02	3.00e-03	3.81e-02	4.00e-03	5.71e-02	2.67e-02	3.57e-02	3.00e-03
8	2.94e-02	2.00e-03	2.94e-02	2.70e-03	4.55e-02	1.97e-02	2.78e-02	1.40e-03
9	2.34e-02	1.40e-03	2.34e-02	2.00e-03	3.70e-02	1.20e-02	2.22e-02	1.10e-03
10	1.90e-02	1.10e-03	1.90e-02	1.50e-03	3.08e-02	9.80e-03	1.82e-02	1.10e-03
11	1.58e-02	8.11e-04	1.58e-02	1.10e-03	2.60e-02	8.10e-03	1.52e-02	8.00e-04
12	1.33e-02	6.30e-04	1.33e-02	9.00e-04	2.22e-02	6.80e-03	1.28e-02	6.00e-04
13	1.14e-02	4.99e-04	1.14e-02	7.00e-04	1.92e-02	5.80e-03	1.10e-02	5.00e-04
14	9.90e-03	4.02e-04	9.90e-03	6.00e-04	1.68e-02	5.00e-03	9.50e-03	4.00e-04
15	8.60e-03	3.29e-04	8.60e-03	5.00e-04	1.48e-02	4.40e-03	8.30e-03	3.00e-04
16	7.60e-03	2.72e-04	7.60e-03	4.00e-04	1.32e-02	3.80e-03	7.40e-03	3.00e-04
17	6.70e-03	2.28e-04	6.70e-03	3.00e-04	1.18e-02	3.40e-03	6.50e-03	2.00e-04
18	6.00e-03	1.93e-04	6.00e-03	3.00e-04	1.06e-02	3.00e-03	5.80e-03	2.00e-04
19	5.40e-03	1.64e-04	5.40e-03	2.00e-04	9.60e-03	2.70e-03	5.30e-03	2.00e-04
20	4.90e-03	1.41e-04	4.90e-03	2.00e-04	8.70e-03	2.50e-03	4.80e-03	1.00e-04

Table 4, is the comparison of traditional efficiency and Flops-like efficiency indices for $n = 2$ to $n = 20$ for the four methods. Reports from the results in the Table and figures 3 and 4, the shows that ELM is very competative and efficient.

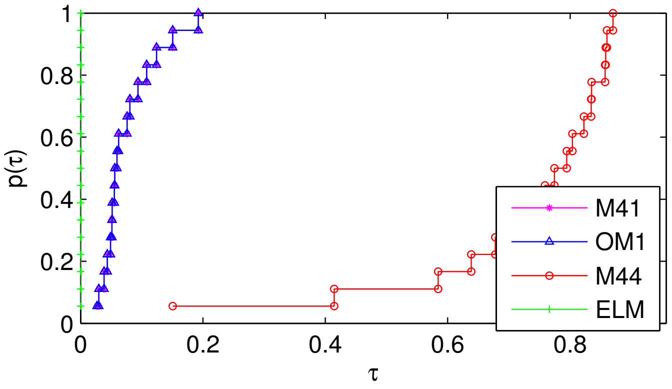

FIGURE 3. Performance profile of M41, M44, OM4 and ELM. methods with respect to traditional efficiency index for $n = 2$ to $n = 20$

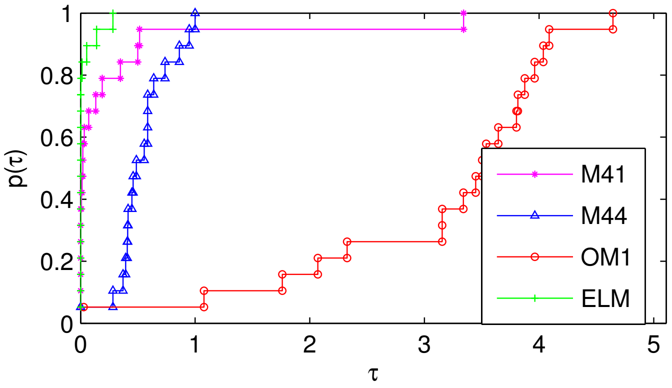

FIGURE 4. Performance profile of M41, M44, OM4 and ELM. methods with respect to Flops-like efficiency index for $n = 2$ to $n = 20$

6. CONCLUSION

In this paper, we have proposed a two- step fourth-order globally convergent method. The method shows its robustness more especially when solving singular problems. It also competes with some existing fourth-order methods in terms of less number of iteration and cpu-time. It numerically excels in both traditional efficiency as well as flops-like efficiency indices. Though it involves derivative computation but is highly efficient for solving medium-scale problems.

Conflict of interest: The authours declare that they have no conflict of interest.

Acknowledgement
The authors are much grateful to the Tertiary Education Trust Fund (TETFund), No 6, Zambezi Crescent, Off Aguiyi Ironsi St, Maitama, Abuja, Nigeria, for sponsoring fellowship and bench-work of Yau Balarabe Musa at both Bayero University, Kano, Nigeria and COMSATS University Islamabad, Islamabad Pakistan respectively. Appreciation to the Vice Chancellor, Sule Lamido University, Jigawa state, Nigeria, for moral and conceptual supports, and the Rector, COMSATS university Islamabad, Islamabad Pakistan for providing the research facilities and the environment.

References

[1] S. Abbasbandy. Improving Newton-Raphson method for nonlinear equations by modified Adomian decomposition method. Appl. Math. Comput., 145 (2003), pp. 887-893.
[2] K. Amini and F. Rostamii. A modified two steps Levenberg-Marquardt method for nonlinear equations. J. Comput. Appl. Math. 288(2015), 341-350.
[3] K. Amini and F. Rostami. Three-steps modified LevenbergMarquardt method with a new line search for systems of nonlinear equations. J. Comput. Appl. Math., 300(2016), 30-42.
[4] S. Artidiello, A. Cordero, J. R. Torregrosa and M. Pencova. Design of high-order iterative methods for nonlinear systems by using weight function procedure. J. Comput. Appl. Math. 1(2015), 1-12.
[5] M.S. Bahgat and M.A Hafiz (2012): *An Efficient two- step Iterative Method For Solving System of Nonlinear Equations*, Journal of Mathematical Research, 5, pp 3456-3476.
[6] P. Bhuvaneswari and G.P Ramesh. Levenberg-Marquardt algorithm to identify the fault analysis for industrial applications. Inter. J. Engin. and Tech. 7 (2018), 141-150.
[7] C.G. Broyden. Quasi-Newton methods and their applications to function minimisation. J. Math. Comput. 21(1967), 577-593.
[8] L. Chen. A high modified Levenberg-Marquardt method for system of nonlinear equations with fourth ordre convergence. Appl. Math. Comput. 285(2016), 79-93.
[9] Y. Chen and Y. Gao. Levenberg-Marquardt Method for the Eigenvalue Complementarity Problem. J. Math. Compt. 6(2014), 39-56.
[10] K. Christian. (2001): *An active set-type Newton method for constrained nonlinear systems*, M.C. Ferris, O.L. Mangasarian and J.-S. Pang (eds.):Complementarity: Applications,Algorithms and Extensions, Kluwer Academic Publishers, pp. 179-200.
[11] C. Chun. A new iterative method for solving nonlinear equations. Appl. Math. Comput., 187 (2006), 415-422.
[12] A. Cordero, J. R. Torregrosa and M. P. Vassilera. Increasing the order of convergence of iterative schemes for solving nonlinear systems. J. Comput. Appl. Math. 252(2013), 86-94.
[13] J. E. Dennis and J. J. More. A characterization of super linear convergence and its application to quasi- Newton methods. Math. Comput. 28(1974), 549-560.
[14] S. Gratton, A. S. Lawless and N. K. Nichols. Approximate Gauss-Newton Methods for Nonlinear Least Squares Problems. SIAM J. Optim. 18(1)(2007), 106-132
[15] M.A. Hafiz, S.M.H. Algoria. New ninth and seventh order methods for solving nonlinear equations. Eur. Sci. J., 8 (27) (2012), 83-95.
[16] E. Halley. A new, exact and easy method for finding the roots of equations generally, and that without any previous reduction. Philos. R. Soc. London, 18 (1964), 136-148.

[17] T. Kogan, L. Sapir, A. Sapir, A. Sapir. To the question of efficiency of iterative methods. Appl. Math. Lett., 66 (2017), 40-46.

[18] A. Kumar, P. Maroju, R. Behl, D.K. Gupta, S.S. Motsa. A family of higher order iterations free from second derivative for nonlinear equations J. Comput. Appl. Math. (2017), 10.1016/j.cam.2017.07.005.

[19] K. Levenberg. A method for the Solution of certain non-linear problems in least squares. Quart. Appl. Math. 2 (1914), 164-168.

[20] [10] D. Li and M. Fukushima, A globally and superlinearly convergent Gauss-Newton-based BFGS method for symmetric nonlinear equations, SIAM J. Numer. Anal., 37 (1999), 152-172.

[21] K. Amini, F. Rostami and G. Carist . An efficient LevenbergMarquardt method with a new LM parameter for systems of nonlinear equations, J. Math. Prog. Operat. Res. 67 (2018), 637-650. Mathematics, 4 (2) (2016), p. 22, 10.3390/math4020022.

[22] J. R. Sharma and R. K. Guha. An efficient fourth-order Weighted-Newton method for nonlinear system of equations. Numer. Algorithm.63(2013), 307-323 .

[23] A. Singh. On a three-step efficient fourth-order method for computing the numerical solution of system of nonlinear equations and its application. Natl. Acad. sci. (2019), 1-8. https://doi.org/10.1007/s40010.019-00617-4.

[24] O. S. Solaiman and I. Hashim. Two-steps efficient six-order iterative methods for solving nonlinear equations. J. King Sau. Univ. (2018), 1-5. https://doi.org/10.1016/j.jksus.2018.03.021.

[25] M. M. Solaymani, F. S. Shateyi and S. S. S. Mosta. On a new method for computing the numerical solution of system of nonlinear equations.J. Appl. Maht. Article ID 751975.

[26] L. Song and Yan Gao. A smoothing Levenberg-Marquardt method for nonlinear complementarity problems. Numer Algor. 79 (2018). 1305-1321.

[27] J.F. Traub, Iterative Methods for Solution of Equations. Prentice-Hall, Englewood, Cliffs, NJ (1964).

[28] M.Y. Waziri, W.J. Leong, M. Mamat and A.U. Moyi, Two-step derivative-free diagonally newtons method for large-scale nonlinear equations. World Appl. Sci.J. (Special Issue of Applied Math). 21(2013), 86-94.

[29] M.Y. Waziri, Ibrahim Saidu and Aisha Halliru A new multi-step fixed newtons method for solving large-scale systems of nonlinear equations. IUP J. Comput. Math. 1(2011), 19-28.

[30] Y. B. Musa, M. Y. Waziri and A. S. Halliru.On Computing the Regularization Parameter for the Levenberg-Marquardt Method Via the Spectral Radius Approach to Solving Systems of Nonlinear Equations. J. Numer. Math. Stoch., 9 (1)(2017), 80-94,.

References should be cited in the text as, e.g., [5], [6, p. 213], or [6, Theorem 2.1]. It is suggested that the theorem and page numbers be used for referencing instead of just a number representing an entire book or paper, and also that only those papers or books which are actually cited in the text be listed as references.