Journal of
Linear and Topological Algebra
Vol. 05*, No.* 03*,* 2016*,* 187- 198

J Journal of Linear and Topological Algebra
L T A

Solving systems of nonlinear equations using decomposition technique

M. Nili Ahmadabadi[a*], F. Ahmad[b], G. Yuan[c], Xian. Li[c]

[a] *Department of Mathematics, Najafabad Branch, Islamic Azad University, Najafabad, Iran.*
[b] *Departament de Física i Enginyeria Nuclear, Universitat Politècnica de Catalunya, Comte d'Urgell 187, 08036 Barcelona, Spain.*
[c] *College of Mathematics and Information Science, Guangxi University, Nanning, Guangxi, 530004, P.R. China.*
[d] *School of Mathematics and Computing Science, Guangxi Colleges and Universities Key Laboratory of Data Analysis and Computation, Guilin University of Electronic Technology, Guilin, Guangxi, China, 541004.*

Received 10 July 2016; Revised 11 September 2016; Accepted 20 September 2016.

Abstract. A systematic way is presented for the construction of multi-step iterative method with frozen Jacobian. The inclusion of an auxiliary function is discussed. The presented analysis shows that how to incorporate auxiliary function in a way that we can keep the order of convergence and computational cost of Newton multi-step method. The auxiliary function provides us the way to overcome the singularity and ill-conditioning of the Jacobian. The order of convergence of proposed p-step iterative method is $p + 1$. Only one Jacobian inversion in the form of LU-factorization is required for a single iteration of the iterative method and in this way, it offers an efficient scheme. For the construction of our proposed iterative method, we used a decomposition technique that naturally provides different iterative schemes. We also computed the computational convergence order that confirms the claimed theoretical order of convergence. The developed iterative scheme is applied to large scale problems, and numerical results show that our iterative scheme is promising.

© 2016 IAUCTB. All rights reserved.

Keywords: Systems of nonlinear equations, decomposition, order of convergence, higher order methods, computational efficiency.

[*] Corresponding author.
E-mail address: mneely59@hotmail.com (M. Nili Ahmadabadi).

Print ISSN: 2252-0201
Online ISSN: 2345-5934

© 2016 IAUCTB. All rights reserved.
http://jlta.iauctb.ac.ir

1. Introduction

Nature is nonlinear, and that is why we find a large class of nonlinear problems in different disciplines of science, engineering, and technology. Most of the nonlinear phenomena are modeled in the form of nonlinear ordinary and partial differential equations. The discretization of nonlinear ordinary and partial differential equations provides the system of nonlinear equations. But many real-world scenarios can be modeled in the form nonlinear algebraic system of equations. So constructing efficient iterative algorithms for solving system of nonlinear equations is an effective area of research. Let $\mathbf{F}(\mathbf{x}) = [F_1(\mathbf{x}), F_1(\mathbf{x}), \cdots, F_n(\mathbf{x})]^T = \mathbf{0}$ be a system of nonlinear equations, where $\mathbf{x} = [x_1, x_2, \cdots, x_n]^T$ and $\mathbf{0}$ is a column vector of zeros of size n. When we talk about to find simple zeros of system of nonlinear equations, the first classical iterative method that comes into mind is the Newton-Raphson method [1, 2] that can be written as

$$\begin{cases} \mathbf{x}_0 = \text{initial guess} \\ \mathbf{F}'(\mathbf{x}_k)\,\boldsymbol{\phi} = \mathbf{F}(\mathbf{x}_k) \\ \mathbf{x}_{k+1} = \mathbf{x}_k - \boldsymbol{\phi}, \quad k = 0, 1, 2 \cdots . \end{cases} \tag{1}$$

The implementation of Newton method requires the non-singularity of Jacobian $\mathbf{F}'(\cdot)$ during iteration and, of course, the ill-conditioning of Jacobian also affects the convergence.

Recently, to improve the order of convergence of Newton method[1, 2], Gutirrez[8] and Sharma [20] proposed more robust and efficient methods. In reference [21], a new class of order five method has been proposed. Recently, some researchers have reported higher order class of multi-step iterative method[3–6] for solving system of nonlinear equations. The multi-step iterative method has the benefit that they utilize the LU factorization information of Jacobian and solve the involved system of linear equations repeatedly and provides a high order of convergence with less number of function evaluations. However, the proposed higher-order iterative methods are futile unless they have low computational cost. Therefore, the aim of developing new algorithms is to achieve as high as possible convergence order requiring as small as possible the evaluations of functions, derivatives and matrix inversions. Our work is a generalization of [7] to higher dimensions, i.e., systems of nonlinear equations. In the next section, we will describe our approach in more details.

The paper is organized as follows: Iterative methods for solving system of nonlinear equations are proposed in Section 2. In Section 3, convergence analysis is given. Finally, the results of numerical experiments of the proposed algorithm are reported in Section 4, and conclusions can get in Section 5.

2. Iterative methods

The main aim of this section is to give our method. To obtain high order iterative methods for the system of nonlinear equations, we generalize [7]. Assume that $\boldsymbol{\gamma}$ is an initial guess in the vicinity of a root of the system of nonlinear equations. Let $\mathbf{G} : \mathbb{R}^n \to \mathbb{R}^n$ be an auxiliary function such that

$$\begin{cases} \mathbf{F}(\mathbf{x}) \odot \mathbf{G}(\mathbf{x}) = \mathbf{0} \ \text{ or} \\ F_i(\mathbf{x})\, G_i(\mathbf{x}) = 0, \quad i = 1, 2, \cdots, n, \end{cases} \tag{2}$$

where $\odot$ is the point-wise multiplication and $\mathbf{G}(\mathbf{x}) = [G_1(\mathbf{x}), G_2(\mathbf{x}), \cdots, G_n(\mathbf{x})]^T$. We assume that $\mathbf{G}(\mathbf{x}) \neq \mathbf{0}$ at least for the roots of $\mathbf{F}(\mathbf{x}) = \mathbf{0}$. However, It is advisable that $\mathbf{G}(\mathbf{x}) \neq \mathbf{0}$ and $\det(\mathbf{G}'(\mathbf{x})) \neq \mathbf{0}$ for all values of $\mathbf{x}$ With the help of a multi-dimensional Taylor series, we can rewrite (2) as

$$F_i(\boldsymbol{\gamma})\, G_i(\boldsymbol{\gamma}) + [\nabla F_i(\boldsymbol{\gamma})\, G_i(\boldsymbol{\gamma}) + F_i(\boldsymbol{\gamma})\, \nabla G_i(\boldsymbol{\gamma})]^T (\mathbf{x} - \boldsymbol{\gamma}) + H_i(\mathbf{x}) = 0, \quad i = 1, 2, \cdots, n, \tag{3}$$

i.e.

$$\begin{bmatrix} F_1(\boldsymbol{\gamma})G_1(\boldsymbol{\gamma}) \\ \vdots \\ F_n(\boldsymbol{\gamma})G_n(\boldsymbol{\gamma}) \end{bmatrix} + \begin{bmatrix} [\nabla F_1(\boldsymbol{\gamma})G_1(\boldsymbol{\gamma}) + F_1(\boldsymbol{\gamma})\nabla G_1(\boldsymbol{\gamma})]^T \\ \vdots \\ [\nabla F_n(\boldsymbol{\gamma})G_n(\boldsymbol{\gamma}) + F_n(\boldsymbol{\gamma})\nabla G_n(\boldsymbol{\gamma})]^T \end{bmatrix} (\mathbf{x} - \boldsymbol{\gamma}) + \begin{bmatrix} h_1(\mathbf{x}) \\ \vdots \\ h_n(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{4}$$

The above equation implies that

$$\begin{cases} [H_i(\mathbf{x})]_{i=1}^n = -[F_i(\boldsymbol{\gamma})\, G_i(\boldsymbol{\gamma})]_{i=1}^n - \left[[\nabla F_i(\boldsymbol{\gamma}) G_i(\boldsymbol{\gamma}) + F_i(\boldsymbol{\gamma})\, \nabla G_i(\boldsymbol{\gamma})]^T\right]_{i=1}^n (\mathbf{x} - \boldsymbol{\gamma}) \quad \text{or} \\ [H_i(\mathbf{x})]_{i=1}^n = [F_i(\mathbf{x})\, G_i(\boldsymbol{\gamma})]_{i=1}^n - [F_i(\boldsymbol{\gamma})\, G_i(\boldsymbol{\gamma})]_{i=1}^n - \left[[\nabla F_i(\boldsymbol{\gamma}) G_i(\boldsymbol{\gamma}) + F_i(\boldsymbol{\gamma})\, \nabla G_i(\boldsymbol{\gamma})]^T\right]_{i=1}^n (\mathbf{x} - \boldsymbol{\gamma}), \end{cases} \tag{5}$$

and

$$\begin{aligned} \mathbf{x} = \boldsymbol{\gamma} &- \left[[\nabla F_i(\boldsymbol{\gamma})\, G_i(\boldsymbol{\gamma}) + F_i(\boldsymbol{\gamma})\, \nabla G_i(\boldsymbol{\gamma})]^T\right]_{i=1}^{n\ -1} [F_i(\boldsymbol{\gamma})\, G_i(\boldsymbol{\gamma})]_{i=1}^n \\ &- \left[[\nabla F_i(\boldsymbol{\gamma})\, G_i(\boldsymbol{\gamma}) + F_i(\boldsymbol{\gamma})\, \nabla G_i(\boldsymbol{\gamma})]^T\right]_{i=1}^{n\ -1} [H_i(\mathbf{x})]_{i=1}^n. \end{aligned} \tag{6}$$

For simplicity in the following discussion, we express (6) as

$$\mathbf{x} = \mathbf{c} + N(\mathbf{x}). \tag{7}$$

Where

$$\mathbf{c} = \boldsymbol{\gamma} - \left[[\nabla F_i(\boldsymbol{\gamma})\, G_i(\boldsymbol{\gamma}) + F_i(\boldsymbol{\gamma})\, \nabla G_i(\boldsymbol{\gamma})]^T\right]_{i=1}^{n\ -1} [F_i(\boldsymbol{\gamma})\, G_i(\boldsymbol{\gamma})]_{i=1}^n, \tag{8}$$

and

$$N(\mathbf{x}) = -\left[[\nabla F_i(\boldsymbol{\gamma})\, G_i(\boldsymbol{\gamma}) + F_i(\boldsymbol{\gamma})\, \nabla G_i(\boldsymbol{\gamma})]^T\right]_{i=1}^{n\ -1} [H_i(\mathbf{x})]_{i=1}^n \tag{9}$$

is a nonlinear function.

In what follows, we adopt decomposition technique[23]. We look for a solution in the form

$$\mathbf{x} = \sum_{i=0}^{\infty} \mathbf{x}_i. \tag{10}$$

At the same time, the nonlinear operator N can be decomposed as

$$N(\mathbf{x}) = N(\mathbf{x}_0) + \sum_{i=1}^{\infty} \left\{ N\left(\sum_{j=0}^{i} \mathbf{x}_j\right) - N\left(\sum_{j=0}^{i-1} \mathbf{x}_j\right) \right\}. \tag{11}$$

Combining (7), (10) and (11), we deduce

$$\sum_{i=0}^{\infty} \mathbf{x}_i = \mathbf{c} + N(\mathbf{x}_0) + \sum_{i=1}^{\infty} \left\{ N\left(\sum_{j=0}^{i} \mathbf{x}_j\right) - N\left(\sum_{j=0}^{i-1} \mathbf{x}_j\right) \right\}. \tag{12}$$

From the structure of (12), we assume the following iterative scheme

$$\begin{cases} \mathbf{x}_0 = \mathbf{c}, \\ \mathbf{x}_1 = N(\mathbf{x}_0), \\ \mathbf{x}_2 = N(\mathbf{x}_0 + \mathbf{x}_1) - N(\mathbf{x}_0), \\ \vdots \\ \mathbf{x}_{m+1} = N\left(\sum_{j=0}^{m} \mathbf{x}_j\right) - N\left(\sum_{j=0}^{m-1} \mathbf{x}_j\right), \qquad m = 1, 2, \cdots , \end{cases} \tag{13}$$

Sum both sides of (13), we can obtain that

$$\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_{m+1} = N(\mathbf{x}_0 + \mathbf{x}_1 + \cdots + \mathbf{x}_m), \quad m = 1, 2, \cdots . \tag{14}$$

From (10) and (13), there are

$$\mathbf{x} = \mathbf{c} + \sum_{i=1}^{\infty} \mathbf{x}_i. \tag{15}$$

By [7], we know that series $\sum_{i=0}^{\infty} \mathbf{x}_i$ converges absolutely and uniformly to a unique solution of Eq. (**??**). Using (8) and (13), we can easily deduce that

$$\mathbf{x}_0 = \mathbf{c} = \boldsymbol{\gamma} - \left[[\nabla F_i(\boldsymbol{\gamma})\, G_i(\boldsymbol{\gamma}) + F_i(\boldsymbol{\gamma})\, \nabla G_i(\boldsymbol{\gamma})]^T \right]_{i=1}^{n}{}^{-1} [F_i(\boldsymbol{\gamma})\, G_i(\boldsymbol{\gamma})]_{i=1}^{n}. \tag{16}$$

Similar to [7], by (5), (9) and (13), we observe that

$$[H_i(\mathbf{x}_0)]_{i=1}^{n} = [F_i(\mathbf{x}_0) G_i(\boldsymbol{\gamma})]_{i=1}^{n}, \tag{17}$$

and

$$\mathbf{x}_1 = N(\mathbf{x}_0) = -\left[[\nabla F_i(\boldsymbol{\gamma})\, G_i(\boldsymbol{\gamma}) + F_i(\boldsymbol{\gamma})\, \nabla G_i(\boldsymbol{\gamma})]^T \right]_{i=1}^{n}{}^{-1} [F_i(\mathbf{x}_0) G_i(\boldsymbol{\gamma})]_{i=1}^{n}. \tag{18}$$

Let

$$\mathbf{w}_m = \mathbf{x}_0 + \mathbf{x}_1 + \cdots + \mathbf{x}_m. \tag{19}$$

We note that $\mathbf{x}$ is approximated by $\mathbf{w}_m$, and

$$\mathbf{x} = \lim_{m\to\infty} \mathbf{w}_m. \tag{20}$$

For $m = 0$, the approximation becomes

$$\mathbf{x} \approx \mathbf{w}_0 = \mathbf{x}_0 = \boldsymbol{\gamma} - \big[[\nabla F_i(\boldsymbol{\gamma})\, G_i(\boldsymbol{\gamma}) + F_i(\boldsymbol{\gamma})\, \nabla G_i(\boldsymbol{\gamma})]^T\big]_{i=1}^{n}{}^{-1} [F_i(\boldsymbol{\gamma})\, G_i(\boldsymbol{\gamma})]_{i=1}^{n}, \tag{21}$$

which suggests that we can use the following one-step iterative method to solve system of nonlinear equations. The idea of Algorithm 2.1 comes from reference [23].

Algorithm 2.1

Given $\mathbf{x}_0$(which can be computed by (21)), we can get the approximate solution $\mathbf{x}_{k+1}$ by the following way

$$\big[[\nabla F_i(\mathbf{x}_k)\, G_i(\mathbf{x}_k) + F_i(\mathbf{x}_k)\, \nabla G_i(\mathbf{x}_k)]^T\big]_{i=1}^{n} (\mathbf{x}_{k+1} - \mathbf{x}_k) = -[F_i(\mathbf{x}_k)\, G_i(\mathbf{x}_k)]_{i=1}^{n},$$

for all $k = 0, 1, 2, \cdots$. For $m = 1$, the approximation becomes

$$\begin{aligned} \mathbf{x} \approx W_1 = \mathbf{x}_0 + \mathbf{x}_1 = \boldsymbol{\gamma} - \big[[\nabla F_i(\boldsymbol{\gamma})\, G_i(\boldsymbol{\gamma}) + F_i(\boldsymbol{\gamma})\, \nabla G_i(\boldsymbol{\gamma})]^T\big]_{i=1}^{n}{}^{-1} [F_i(\boldsymbol{\gamma})\, G_i(\boldsymbol{\gamma})]_{i=1}^{n} \\ - \big[[\nabla F_i(\boldsymbol{\gamma})\, G_i(\boldsymbol{\gamma}) + F_i(\boldsymbol{\gamma})\, \nabla G_i(\boldsymbol{\gamma})]^T\big]_{i=1}^{n}{}^{-1} [F_i(\mathbf{x}_0) G_i(\boldsymbol{\gamma})]_{i=1}^{n}. \end{aligned} \tag{22}$$

By (22), we can deduce the following two-step iterative method.

Algorithm 2.2

For a given $\mathbf{x}_0$, the approximate solution $\mathbf{x}_{k+1}$ can be calculated by solving the following two equations:

$$\big[[\nabla F_i(\mathbf{x}_k)\, G_i(\mathbf{x}_k) + F_i(\mathbf{x}_k)\, \nabla G_i(\mathbf{x}_k)]^T\big]_{i=1}^{n} (\mathbf{y}_k - \mathbf{x}_k) = -[F_i(\mathbf{x}_k)\, G_i(\mathbf{x}_k)]_{i=1}^{n},$$

$$\big[[\nabla F_i(\mathbf{x}_k)\, G_i(\mathbf{x}_k) + F_i(\mathbf{x}_k)\, \nabla G_i(\mathbf{x}_k)]^T\big]_{i=1}^{n} (\mathbf{x}_{k+1} - \mathbf{y}_k) = -[F_i(\mathbf{y}_k)\, G_i(\mathbf{x}_k)]_{i=1}^{n},$$

for all $k = 0, 1, 2, \cdots$. Similar to the previous cases, for $m = 2$, the approximation becomes

$$\begin{aligned} \mathbf{x} \approx W_2 = \mathbf{x}_0 + \mathbf{x}_1 + \mathbf{x}_2 = \boldsymbol{\gamma} - \big[[\nabla F_i(\boldsymbol{\gamma})\, G_i(\boldsymbol{\gamma}) + F_i(\boldsymbol{\gamma})\, \nabla G_i(\boldsymbol{\gamma})]^T\big]_{i=1}^{n}{}^{-1} [F_i(\boldsymbol{\gamma})\, G_i(\boldsymbol{\gamma})]_{i=1}^{n} \\ - \big[[\nabla F_i(\boldsymbol{\gamma})\, G_i(\boldsymbol{\gamma}) + F_i(\boldsymbol{\gamma})\, \nabla G_i(\boldsymbol{\gamma})]^T\big]_{i=1}^{n}{}^{-1} [H_i(\mathbf{x}_0 + \mathbf{x}_1)]_{i=1}^{n} \\ = \boldsymbol{\gamma} - \big[[\nabla F_i(\boldsymbol{\gamma})\, G_i(\boldsymbol{\gamma}) + F_i(\boldsymbol{\gamma})\, \nabla G_i(\boldsymbol{\gamma})]^T\big]_{i=1}^{n}{}^{-1} [F_i(\boldsymbol{\gamma})\, G_i(\boldsymbol{\gamma})]_{i=1}^{n} \\ - \big[[\nabla F_i(\boldsymbol{\gamma})\, G_i(\boldsymbol{\gamma}) + F_i(\boldsymbol{\gamma})\, \nabla G_i(\boldsymbol{\gamma})]^T\big]_{i=1}^{n}{}^{-1} [(F_i(\mathbf{x}_0 + \mathbf{x}_1) + F_i(\mathbf{x}_0)) G_i(\boldsymbol{\gamma})]_{i=1}^{n}. \end{aligned} \tag{23}$$

Thus, we can easily establish the following three-step iterative method.

Algorithm 2.3

For a given $\mathbf{x}_0$ compute the approximate solution $\mathbf{x}_{k+1}$ from

$$\begin{aligned}
&\left[[\nabla F_i(\mathbf{x}_k)\, G_i(\mathbf{x}_k) + F_i(\mathbf{x}_k)\, \nabla G_i(\mathbf{x}_k)]^T\right]_{i=1}^n (\mathbf{y}_k - \mathbf{x}_k) = -[F_i(\mathbf{x}_k)\, G_i(\mathbf{x}_k)]_{i=1}^n\\
&\left[[\nabla F_i(\mathbf{x}_k)\, G_i(\mathbf{x}_k) + F_i(\mathbf{x}_k)\, \nabla G_i(\mathbf{x}_k)]^T\right]_{i=1}^n (\mathbf{z}_k - \mathbf{y}_k) = -[F_i(\mathbf{y}_k)\, G_i(\mathbf{x}_k)]_{i=1}^n\\
&\left[[\nabla F_i(\mathbf{x}_k)\, G_i(\mathbf{x}_k) + F_i(\mathbf{x}_k)\, \nabla G_i(\mathbf{x}_k)]^T\right]_{i=1}^n (\mathbf{x}_{k+1} - \mathbf{z}_k) = -[F_i(\mathbf{x}_k)\ G_i(\mathbf{x}_k)]_{i=1}^n,
\end{aligned}$$

for all $k = 0, 1, 2, \cdots$.

3. Convergence analysis

The convergence proofs of multi-step iterative with frozen Jacobian can be find in [3, 25]. In the following section, we will only show that, our prosed multi-step iterative schemes can be written in frozen Jacobian form. Once we have shown, we enjoy the convergence proofs from [3, 25]. We define a new function $\mathbf{Q}(\boldsymbol{\theta}, \mathbf{x})$ with help of an auxiliary function $\mathbf{G}(\boldsymbol{\theta})$, such that $\mathbf{Q}(\boldsymbol{\theta}, \mathbf{x}) = \operatorname{diag}(\mathbf{G}(\boldsymbol{\theta}))\, \mathbf{F}(\mathbf{x})$, where $\operatorname{diag}(\cdot)$ is the diagonal matrix. The first order Fréchet derivative of $\mathbf{Q}(\mathbf{x}, \mathbf{x})$ can be computed as

$$\begin{aligned}
\mathbf{Q}'(\mathbf{x}, \mathbf{x}) &= \operatorname{diag}(\mathbf{G}(\mathbf{x}))\, \mathbf{F}'(\mathbf{x}) + \operatorname{diag}(\mathbf{F}(\mathbf{x}))\ \mathbf{G}'(\mathbf{x})\\
&= \operatorname{diag}(\mathbf{G}(\mathbf{x}))\ \Big(\mathbf{F}'(\mathbf{x}) + \operatorname{diag}(\mathbf{G}(\mathbf{x}))^{-1} \operatorname{diag}(\mathbf{F}(\mathbf{x}_k))\ \mathbf{G}'(\mathbf{x})\Big).
\end{aligned} \tag{24}$$

Using Newton method to solve $\mathbf{Q}(\mathbf{x}, \mathbf{x}) = \mathbf{0}$, we state

$$\begin{aligned}
\mathbf{x}_{k+1} &= \mathbf{x}_k - \mathbf{Q}'(\mathbf{x}_k, \mathbf{x}_k)^{-1}\, \mathbf{Q}(\mathbf{x}_k, \mathbf{x}_k)\\
&= \mathbf{x}_k - \Big(\Big(\mathbf{F}'(\mathbf{x}_k) + \operatorname{diag}(\mathbf{G}(\mathbf{x}_k))^{-1} \operatorname{diag}(\mathbf{F}(\mathbf{x}_k))\ \mathbf{G}'(\mathbf{x}_k)\Big)\Big)^{-1}\\
&\quad (\operatorname{diag}(\mathbf{G}(\mathbf{x}_k)))^{-1}\ \mathbf{Q}(\mathbf{x}_k, \mathbf{x}_k).
\end{aligned} \tag{25}$$

Since $(\operatorname{diag}(\mathbf{G}(\mathbf{x})))^{-1}\ \mathbf{Q}(\mathbf{x}, \mathbf{x}) = (\operatorname{diag}(\mathbf{G}(\mathbf{x})))^{-1}\ diag(\mathbf{G}(\mathbf{x}))\, \mathbf{F}(\mathbf{x}) = \mathbf{F}(\mathbf{x})$, (25) can be written as

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \Big(\Big(\mathbf{F}'(\mathbf{x}_k) + \operatorname{diag}(\mathbf{G}(\mathbf{x}_k))^{-1} \operatorname{diag}(\mathbf{F}(\mathbf{x}_k))\ \mathbf{G}'(\mathbf{x})\Big)\Big)^{-1}\ \mathbf{F}(\mathbf{x}_k). \tag{26}$$

The convergence order of Newton method is at least two, hence we conclude that the convergence order of (26) is at least two. The convergence order of frozen Jacobian schemes (27) and (28) are three and four respectively (see [25]).

$$\begin{aligned}
\mathbf{y}_k &= \mathbf{x}_k - \mathbf{Q}(\mathbf{x}_k, \mathbf{x}_k)^{-1}\, \mathbf{Q}(\mathbf{x}_k, \mathbf{x}_k)\\
\mathbf{x}_{k+1} &= \mathbf{y}_k - \mathbf{Q}(\mathbf{x}_k, \mathbf{x}_k)^{-1}\, \mathbf{Q}(\mathbf{x}_k, \mathbf{y}_k).
\end{aligned} \tag{27}$$

$$\begin{aligned}\mathbf{y}_k &= \mathbf{x}_k - \mathbf{Q}(\mathbf{x}_k,\mathbf{x}_k)^{-1}\,\mathbf{Q}(\mathbf{x}_k,\mathbf{x}_k)\\ \mathbf{z}_k &= \mathbf{y}_k - \mathbf{Q}(\mathbf{x}_k,\mathbf{x}_k)^{-1}\,\mathbf{Q}(\mathbf{x}_k,\mathbf{y}_k)\\ \mathbf{x}_{k+1} &= \mathbf{y}_k - \mathbf{Q}(\mathbf{x}_k,\mathbf{x}_k)^{-1}\,\mathbf{Q}(\mathbf{x}_k,\mathbf{z}_k).\end{aligned}\tag{28}$$

The simplified form of (27) and (28) are in order

$$\begin{aligned}\mathbf{y}_k &= \mathbf{x}_k - \Big(\Big(\mathbf{F}'(\mathbf{x}_k) + \operatorname{diag}\left(\mathbf{G}(\mathbf{x}_k)\right)^{-1}\operatorname{diag}\left(\mathbf{F}(\mathbf{x}_k)\right)\ \mathbf{G}'(\mathbf{x})\Big)\Big)^{-1}\ \mathbf{F}(\mathbf{x}_k)\\ \mathbf{x}_{k+1} &= \mathbf{y}_k - \Big(\Big(\mathbf{F}'(\mathbf{x}_k) + \operatorname{diag}\left(\mathbf{G}(\mathbf{x}_k)\right)^{-1}\operatorname{diag}\left(\mathbf{F}(\mathbf{x}_k)\right)\ \mathbf{G}'(\mathbf{x})\Big)\Big)^{-1}\ \mathbf{F}(\mathbf{y}_k).\end{aligned}\tag{29}$$

$$\begin{aligned}\mathbf{y}_k &= \mathbf{x}_k - \Big(\Big(\mathbf{F}'(\mathbf{x}_k) + \operatorname{diag}\left(\mathbf{G}(\mathbf{x}_k)\right)^{-1}\operatorname{diag}\left(\mathbf{F}(\mathbf{x}_k)\right)\ \mathbf{G}'(\mathbf{x})\Big)\Big)^{-1}\ \mathbf{F}(\mathbf{x}_k)\\ \mathbf{z}_k &= \mathbf{y}_k - \Big(\Big(\mathbf{F}'(\mathbf{x}_k) + \operatorname{diag}\left(\mathbf{G}(\mathbf{x}_k)\right)^{-1}\operatorname{diag}\left(\mathbf{F}(\mathbf{x}_k)\right)\ \mathbf{G}'(\mathbf{x})\Big)\Big)^{-1}\ \mathbf{F}(\mathbf{y}_k)\\ \mathbf{x}_{k+1} &= \mathbf{z}_k - \Big(\Big(\mathbf{F}'(\mathbf{x}_k) + \operatorname{diag}\left(\mathbf{G}(\mathbf{x}_k)\right)^{-1}\operatorname{diag}\left(\mathbf{F}(\mathbf{x}_k)\right)\ \mathbf{G}'(\mathbf{x})\Big)\Big)^{-1}\ \mathbf{F}(\mathbf{z}_k).\end{aligned}\tag{30}$$

Similar to (30), we can get an p-step frozen Jacobian scheme (31) with $p+1$ convergence order as follows:

$$\begin{aligned}\mathbf{y}_{1,k} &= \mathbf{x}_k - \Big(\Big(\mathbf{F}'(\mathbf{x}_k) + \operatorname{diag}\left(\mathbf{G}(\mathbf{x}_k)\right)^{-1}\operatorname{diag}\left(\mathbf{F}(\mathbf{x}_k)\right)\ \mathbf{G}'(\mathbf{x})\Big)\Big)^{-1}\ \mathbf{F}(\mathbf{x}_k)\\ \mathbf{y}_{2,k} &= \mathbf{y}_{1,k} - \Big(\Big(\mathbf{F}'(\mathbf{x}_k) + \operatorname{diag}\left(\mathbf{G}(\mathbf{x}_k)\right)^{-1}\operatorname{diag}\left(\mathbf{F}(\mathbf{x}_k)\right)\ \mathbf{G}'(\mathbf{x})\Big)\Big)^{-1}\ \mathbf{F}(\mathbf{y}_{1,k})\\ \mathbf{y}_{3,k} &= \mathbf{y}_{2,k} - \Big(\Big(\mathbf{F}'(\mathbf{x}_k) + \operatorname{diag}\left(\mathbf{G}(\mathbf{x}_k)\right)^{-1}\operatorname{diag}\left(\mathbf{F}(\mathbf{x}_k)\right)\ \mathbf{G}'(\mathbf{x})\Big)\Big)^{-1}\ \mathbf{F}(\mathbf{y}_{2,k})\\ &\vdots\\ \mathbf{x}_{k+1} &= \mathbf{y}_{p-1,k} - \Big(\Big(\mathbf{F}'(\mathbf{x}_k) + \operatorname{diag}\left(\mathbf{G}(\mathbf{x}_k)\right)^{-1}\operatorname{diag}\left(\mathbf{F}(\mathbf{x}_k)\right)\ \mathbf{G}'(\mathbf{x})\Big)\Big)^{-1}\ \mathbf{F}(\mathbf{y}_{p-1,k}).\end{aligned}\tag{31}$$

Choice of $\mathbf{G}(\cdot)$ is free, However, $\mathbf{G}(\cdot)$ and determinant of its first order Fréchet derivative do not vanish during the course of iterative process. We can choose $\mathbf{G}(\mathbf{x}) = exp(-\boldsymbol{\alpha}\odot\mathbf{x})$, where $\boldsymbol{\alpha} = [\alpha_1,\alpha_2,\cdots,\alpha_n]^T$. The Fréchet derivative of $\mathbf{G}(\mathbf{x})$ is $\mathbf{G}'(\mathbf{x}) = -\operatorname{diag}(\boldsymbol{\alpha})\operatorname{diag}\left(\mathbf{G}(\mathbf{x})\right)$, and (26) can be written as

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{x}_k - \Big(\Big(\mathbf{F}'(\mathbf{x}_k) - \operatorname{diag}\left(\mathbf{G}(\mathbf{x}_k)\right)^{-1}\operatorname{diag}\left(\mathbf{G}(\mathbf{x}_k)\right)\ \operatorname{diag}(\boldsymbol{\alpha})\operatorname{diag}\left(\mathbf{F}(\mathbf{x}_k)\right)\Big)\Big)^{-1}\mathbf{F}(\mathbf{x}_k)\\ \mathbf{x}_{k+1} &= \mathbf{x}_k - \big(\big(\mathbf{F}'(\mathbf{x}_k) - \operatorname{diag}\left(\boldsymbol{\alpha}\odot\mathbf{F}(\mathbf{x}_k)\right)\big)\big)^{-1}\ \mathbf{F}(\mathbf{x}_k).\end{aligned}\tag{32}$$

Notice that the iterative schemes (29) and (30) correspond to Algorithm 2.2 and Algorithm 2.3.

4. Computational cost

The single iteration of p-step frozen Jacobian can be written as

$$\left\{\begin{array}{l} \mathbf{x}_0 = \text{initial guess} \\ \mathbf{A} = \mathbf{F}'(\mathbf{x}_0) + \text{diag}\left(\mathbf{G}(\mathbf{x}_0)\right)^{-1} \text{diag}\left(\mathbf{F}(\mathbf{x}_0)\right)\, \mathbf{G}'(\mathbf{x}_0) \\ \text{for }\; k = 1, p \\ \quad \mathbf{A}\,\boldsymbol{\phi} = \mathbf{F}(\mathbf{x}_{k-1}) \\ \quad \mathbf{x}_k = \mathbf{x}_{k-1} - \boldsymbol{\phi} \\ \text{end} \\ \mathbf{x}_0 = \mathbf{x}_p. \end{array}\right. \tag{33}$$

If we choose $\mathbf{G}(\cdot)$ in way that the Jacobian of $\mathbf{G}(\cdot)$ becomes diagonal matrix then $\text{diag}\left(\mathbf{G}(\mathbf{x}_0)\right)^{-1} \text{diag}\left(\mathbf{F}(\mathbf{x}_0)\right)\, \mathbf{G}'(\mathbf{x}_0)$ can be written as

$$\text{diag}\left(\mathbf{G}(\mathbf{x}_0)\right)^{-1} \text{diag}\left(\mathbf{F}(\mathbf{x}_0)\right)\, \mathbf{G}'(\mathbf{x}_0) = \text{diag}\left(\left[F_i(\mathbf{x}_0)/G_i(\mathbf{x}_0)\, G'_{ii}(\mathbf{x}_0)\right]_{i=1}^{n}\right).$$

When $\mathbf{G}'(\cdot)$ is a diagonal matrix, the computational cost of (33) is given in Table 1.

Number of steps	p
Number of function evaluations	$p+2$
Number of function evaluations (when $\text{diag}(\mathbf{G}(\mathbf{x}_0))^{-1}\, \mathbf{G}'(\mathbf{x}_0)$ is constant vector)	p
Number of Jacobian evaluations	1
Number of LU factors	1
Number of solutions of lower and upper triangular systems	p
Number of vector-vector point-wise multiplications	1
Number of vector-vector point-wise divisions	1

Table 1. Computational cost of multi-step iterative method (33) per iteration when $\mathbf{G}'(\cdot)$ is a diagonal matrix.

Notice that we can write the Newton p-step iterative method as

$$\left\{\begin{array}{l} \mathbf{x}_0 = \text{initial guess} \\ \mathbf{A} = \mathbf{F}'(\mathbf{x}_0) \\ \text{for }\; k = 1, p \\ \quad \mathbf{A}\,\boldsymbol{\phi} = \mathbf{F}(\mathbf{x}_{k-1}) \\ \quad \mathbf{x}_k = \mathbf{x}_{k-1} - \boldsymbol{\phi} \\ \text{end} \\ \mathbf{x}_0 = \mathbf{x}_p. \end{array}\right. \tag{34}$$

5. Numerical experiments

In all numerical simulations, we use the following definition of computational convergence order

$$\text{COC} = \frac{log\left(||\mathbf{F}(\mathbf{x}_{k+1})||_\infty / ||\mathbf{F}(\mathbf{x}_k)||_\infty\right)}{log\left(||\mathbf{F}(\mathbf{x}_k)||_\infty / ||\mathbf{F}(\mathbf{x}_{k-1})||_\infty\right)}. \tag{35}$$

The only difference between Newton frozen Jacobian multi-step iterative method (34) and our newly proposed multi-step iterative method (33) is the introduction of an additive factor $\operatorname{diag}(\mathbf{G}(\mathbf{x}_0))^{-1}\operatorname{diag}(\mathbf{F}(\mathbf{x}_0))\,\mathbf{G}'(\mathbf{x}_0)$ in the Jacobian $\mathbf{F}'(\mathbf{x})$ of target system of nonlinear equations. The main purpose of numerical testing is to check that what is the effect of newly introduced term in the usual Jacobian. So, we make comparison between Newton frozen Jacobian multi-step iterative method (34) and multi-step iterative method (33). In all numerical simulation we choose $\mathbf{G}(\cdot)$ in way that the Jacobian $\mathbf{G}'(\cdot)$ becomes a diagonal matrix. If the Jacobian of the auxiliary function $\mathbf{G}(\cdot)$ is a diagonal matrix then the computational cost of (33) and (34) is almost same. It means that for same number of iterations and multi-steps, the simulation time for both methods is not significantly different. As the selection of auxiliary function $\mathbf{G}(\cdot)(\neq \mathbf{0})$ is open, we have made numerical experiments for different choices of $\mathbf{G}(\cdot)$. Xinyuan wu [26] has already reported $\mathbf{G}(\mathbf{x}) = \exp(\boldsymbol{\alpha} \odot \mathbf{x})$.

Problem 1

We consider the follow problem that has singular Jacobian at its root

$$\begin{aligned} &F_i(\mathbf{x}) = x_i\, x_{i+1}, \;\; i \in \{1, 2, \cdots, n-1\} \\ &F_n(\mathbf{x}) = x_n\, x_1. \end{aligned} \tag{36}$$

The Jacobian of (36) is

$$\begin{aligned} &F'_{i,j} = \delta_{i,j}\, x_{i+1} + \delta_{i+1,j}\, x_i, \;\; i \in \{1, 2, \cdots, n-1\} \\ &F'_{n,1} = x_n \\ &F'_{n,n} = x_1. \end{aligned} \tag{37}$$

Clearly the root of (36) is zeros vector of size n. We can not implement Newton method for solving (36) because at the root Jacobian becomes singular. But we can solve problem (36) with our proposed iterative method (33). Table (2) shows that our proposed iterative method is convergence even we have singular Jacobian at the root. When the constant value of parameter alpha tends to -1, we enjoy fast reduction in the infinity norm of error.

iter	$\lvert\lvert \mathbf{x}_k - \mathbf{0} \rvert\rvert_\infty$	$\lvert\lvert \mathbf{x}_k - \mathbf{0} \rvert\rvert_\infty$
	$\alpha_i = 0.1$	$\alpha_i = -0.999999$
1	5.24e-1	1e-6
5	3.44e-2	6.25e-8
10	1.08e-4	1.95e-9
15	3.37e-5	6.10e-11
20	1.05e-6	1.91e-12
25	3.29e-8	5.96e-14
27	8.22e-9	1.49e-14

Table 2. Problem 1: Number of steps = 1, Size of the problem = 4, $G_i = exp(-\alpha_i\ x_i)$, Initial guess: $x_i = 1$

Problem 2

The second problem involved trigonometry functions.

$$F_i(\mathbf{x}) = 2\Big(n + i\,(1-\cos x_j) - \sin x_j - \sum_{j=1}^{n} \cos x_j\Big)(2\sin x_j - \cos x_j), \;\; 1 \leqslant i \leqslant n. \tag{38}$$

In this problem, we reported different expression for $\mathbf{G}(\cdot)$ and computed the infinity norm of $\mathbf{F}(\cdot)$ for each iteration. Table 3 shows that in all choices of $\mathbf{G}(\cdot)$, the achieved accuracy of iterative method (33) is better than that of (34). The best achieved accuracy of (33) for problem 2 is 5.93×10^{-5698} and $G_i(\mathbf{x}) = -\left(1 + x_i^3\right)$. It means $\mathbf{G}(\mathbf{x})$ do not have single choice of $exp\,(\boldsymbol{\alpha} \odot \mathbf{x})$ as it was reported in [26]. The only key point in the selection of auxiliary function is that it should not be zeros during iteration. In the all cases, we choose three number of steps for both multi-step iterative methods. Hence, the theoretical order of convergence for both methods is at least four. The computed computational convergence order confirms the theoretical claim.

Definition of $\mathbf{G}(\cdot)$	$G_i = -\left(1 + x_i^4\right)$	$G_i = -\left(1 + x_i^3\right)$	$G_i = exp\,(-x_i)$	
Methods	(33)	(33)	(33)	(34)
Norm of residue	$\lVert\mathbf{F}(\mathbf{x}_k)\rVert_\infty$	$\lVert\mathbf{F}(\mathbf{x}_k)\rVert_\infty$	$\lVert\mathbf{F}(\mathbf{x}_k)\rVert_\infty$	$\lVert\mathbf{F}(\mathbf{x}_k)\rVert_\infty$
Iter 1	8.57e+1	5.77e+1	6.1e+1	9.46e+1
2	2.38e+0	3.79e+0	3.42e+1	6.58e+1
3	4.41e-2	1.03e-4	5.72e+0	5.73e+1
4	5.30e-9	3.32e-21	4.26e-3	5.99e-1
5	1.14e-36	3.96e-88	1.33e-15	4.38e-9
6	2.46e-147	2.92e-355	1.54e-65	1.27e-41
7	5.29e–590	8.45e-1424	2.82e-265	8.96e-172
8	1.14e-2360	red**5.93e-5698**	3.13e-1064	2.23e-692
COC	4	4	4	4

Table 3. Problem 2: Number of steps = 3, Size of the problem = 10, Initial guess: $x_i = \frac{101}{100n}$

Problem 3

The third problem consists system of nonlinear equations that admits $\mathbf{1} = [1, 1, \cdots, 1]^T$ as a solution.

$$\begin{aligned} F_i(\mathbf{x}) &= x_i^2\, x_{i+1} - 1, \;\; i = 1, 2, 3, \cdots, n-1, \\ F_n(\mathbf{x}) &= x_{50}^2\, x_1 - 1 \end{aligned} \tag{39}$$

In all selection of $\mathbf{G}(\cdot)$, we obtained better accuracy in the solution compare to (34). Table 4 tells that the highest achieved accuracy is against the selection of $G_i = exp\,(-2\,x_i)$. Interestingly, for problem 3, we have a higher order of convergence than six for different values of parameter $\boldsymbol{\alpha}$. The number of steps for both multi-step iterative methods in the present case is five, and hence the theoretical convergence order is at least six. The computational order of convergence of our proposed iterative method become eleven, and we achieve high order accuracy in numerical solution.

Definition of $\mathbf{G}(\cdot)$	$G_i = exp\,(-0.5\,x_i)$	$G_i = exp\,(-x_i)$	$G_i = exp\,(-2\,x_i)$	
Methods	(33)	(33)	(33)	(34)
Norm of residue	$\|\|\mathbf{F}(\mathbf{x}_k)\|\|_\infty$	$\|\|\mathbf{F}(\mathbf{x}_k)\|\|_\infty$	$\|\|\mathbf{F}(\mathbf{x}_k)\|\|_\infty$	$\|\|\mathbf{F}(\mathbf{x}_k)\|\|_\infty$
Iter 1	8.50e-3	1.43e-3	3.49e-2	3.36e-2
2	3.80e-15	5.69e-24	1.0e-22	8.19e-11
3	3.11e-89	8.84e-167	1.19e-236	1.98e-62
4	9.46e-534	1.92e-1166	7.33e-2601	3.97e-372
5	7.49e-3201	4.43e-8164	red**3.72e-28607**	2.56e-2230
COC	6	7	11	6

Table 4. Problem 3: Number of steps = 5, Size of the problem = 100, Initial guess: $x_i = 1.5$

Problem 4

In problem 4, we solve Broyden Tridiagonal function

$$\begin{aligned} F_1(\mathbf{x}) &= (3 - 0.5\,x_1)\,x_1 - 2\,x_2 + 1 \\ F_n(\mathbf{x}) &= (3 - 0.5\,x_n)\,x_n - x_{n-1} + 1 \\ F_i(\mathbf{x}) &= (3 - 0.5\,x_i)\,x_i - x_{i-1} + 2x_{i+1} + 1, \quad i = 2, 3, \cdots, n-1. \end{aligned} \tag{40}$$

The number of steps is two for both multi-step iterative methods, and it means that the theoretical order of convergence is three. Again the performance of our proposed iterative method is better to compare to that of (34). Table 5 shows the successive error reduction in the solution of problem 4.

Definition of $\mathbf{G}(\cdot)$	$G_i = exp\,(0.1\,x_i)$	$G_i = exp\,(0.01\,x_i)$	$G_i = 1 + 0.1\;(cos(x_i))^2$	
Methods	(33)	(33)	(33)	(34)
Norm of residue	$\|\|\mathbf{F}(\mathbf{x}_k)\|\|_\infty$	$\|\|\mathbf{F}(\mathbf{x}_k)\|\|_\infty$	$\|\|\mathbf{F}(\mathbf{x}_k)\|\|_\infty$	$\|\|\mathbf{F}(\mathbf{x}_k)\|\|_\infty$
Iter 1	1.13e-2	3.38e-2	1.35e-2	4.41e-2
2	2.91e-9	1.76e-7	6.71e-9	2.56e-7
3	7.13e-29	2.34e-23	1.20e-27	6.30e-23
4	1.11e-87	9.90e-71	7.06e-84	2.11e-69
5	red**4.31e-264**	7.30e-213	1.46e-252	7.65e-209
COC	3	3	3	3

Table 5. Problem 4: Number of steps = 5, Size of the problem = 200, Initial guess: $x_i = -1$

6. Conclusion

The proposed frozen Jacobian multi-step iterative method is obtained via the generalization of article [7]. The classical Newton frozen Jacobian p-step iterative method [3, 25] has convergence order at least $p + 1$. We introduce an additive term in the frozen Jacobian in a way that we keep the computational cost and convergence order of Newton multi-step iterative method. The inclusion of auxiliary function benefits us in two ways, first remove the issue of Jacobian singularity and second provides a fast reduction in the solution error. We have shown that the different choices of the auxiliary function can provide us better accuracy and in some cases, we attain a very high order of convergence. The claimed theoretical order of convergences is confirmed by computing the computational order of convergences.

Acknowledgement

This work is supported by the Guangxi Science Fund for Distinguished Young Scholars (No. 2015GXNSFGA139001) and China NSF (Grant No. 11261006).

References

[1] J. M. Ortega, W. C. Rheinboldt, Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, NewYork, 1970.

[2] C. T. Kelley, Solving Nonlinear Equations with Newtons Method, SIAM, Philadelphia, 2003.

[3] F. Ahmad, E. Tohidi, J. A. Carrasco, A parameterized multi-step Newton method for solving systems of nonlinear equations. Numerical Algorithms, (2015), 1017-1398.

[4] F. Ahmad, E. Tohidi, M. Z. Ullah, J. A. Carrasco, Higher order multi-step Jarratt-like method for solving systems of nonlinear equations: Application to PDEs and ODEs, Computers & Mathematics with Applications, 70 (4) (2015), 624-636. http://dx.doi.org/10.1016/j.camwa.2015.05.012.

[5] M. Z. Ullah, S. Serra-Capizzano, F. Ahmad, An efficient multi-step iterative method for computing the numerical solution of systems of nonlinear equations associated with ODEs, Applied Mathematics and Computation, 250 (2015), 249-259. http://dx.doi.org/10.1016/j.amc.2014.10.103.

[6] E. S. Alaidarous, M. Z. Ullah, F. Ahmad, A.S. Al-Fhaid, An Efficient Higher-Order Quasilinearization Method for Solving Nonlinear BVPs ,Journal of Applied Mathematics, vol. 2013 (2013), Article ID 259371, 11 pages.

[7] F. Ahmed Shah, M, Aslam Noor, Some numerical methods for solving nonlinear equations by using decomposition technique, Appl. Math. Comput. 251 (2015) 378-386.

[8] J. M. Gutirrez, M. A. Hernndez, A family of Chebyshev-Halley type methods in banach spaces. Bull. Aust. Math. Soc. 55 (1997) 113-130.

[9] M. Palacios, Kepler equation and accelerated Newton method, J. Comput. Appl. Math. 138 (2002) 335-346.

[10] S. Amat, S. Busquier, J. M. Gutierrez, Geometrical constructions of iterative functions to solve nonlinear equations. J. Comput. Appl. Math. 157 (2003) 197-205.

[11] M. Frontini, E. Sormani, Some variant of Newtons method with third-order convergence, Appl. Math. Comput. 140 (2003) 419-426.

[12] M. Frontini, E. Sormani, Third-order methods from quadrature formulae for solving systems of nonlinear equations, Appl. Math. Comput. 149 (2004) 771-782.

[13] H. H. H. Homeier, A modified Newton method with cubic convergence: the multivariable case. J. Comput. Appl. Math. 169 (2004) 161-169.

[14] M.T. Darvishi, A. Barati, A fourth-order method from quadrature formulae to solve systems of nonlinear equations. Appl. Math. Comput. 188 (2007) 257-261.

[15] A. Cordero, J.R. Torregrosa, Variants of Newtons method using fifth-order quadrature formulas. Appl. Math. Comput. 190 (2007) 686-698.

[16] M. A. Noor, M. Waseem, Some iterative methods for solving a system of nonlinear equations. Comput. Math. Appl. 57 (2009) 101-106.

[17] M. Grau-Sanchez, A. Grau, M. Noguera, Ostrowski type methods for solving systems of nonlinear equations. Appl. Math. Comput. 218 (2011) 2377-2385.

[18] J.A. Ezquerro, M.Grau-Sanchez, A. Grau, M.A. Hernandez, M. Noguera, N. Romero, On iterative methods with accelerated convergence for solving systems of nonlinear equations, J. Optim. Theory Appl. 151 (2011) 163-174.

[19] A. Cordero, J.L. Hueso, E. Martinez, J.R. Torregrosa, Increasing the convergence order of an iterative method for nonlinear systems, Appl. Math. Lett. 25 (2012) 2369-2374.

[20] J.R. Sharma, R.K. Guha, R. Sharma, An efficient fourth order weighted-Newton method for systems of nonlinear equations, Numer. Algorithms 62 (2013) 307-323.

[21] X Xiao,H Yin, A new class of methods with higher order of convergence for solving systems of nonlinear equations, Appl. Math. Comput. 264 (2015) 300-309.

[22] E. D. Dolan and J. J. Moré, Benchmarking optimization software with performance profiles, Mathematical Programming, 91(2002), 201-213.

[23] V. Daftardar-Gejji, H. Jafari, An iterative method for solving nonlinear functional equations. J. Math. Anal. Appl. 316 (2006) 753-763.

[24] J. H. He, Variational iteration method-some recent results and new interpretations. J. Comput. Appl. Math. 207 (2007) 3-17.

[25] Sergio Amat, Sonia Busquier, ngela Grau, Miquel Grau-Sanchez, Maximum efficiency for a family of Newton-like methods with frozen derivatives and some applications. Applied Mathematics and Computation 219 (2013) 7954-7963.

[26] Xinyuan Wu, Note on the improvement of Newtons method for system of nonlinear equations. Applied Mathematics and Computation 189 (2007) 1476-1479 .