A New Accurate and Efficient Iterative Numerical Method for Solving the Scalar and Vector Nonlinear Equations: Approach Based on Geometric Considerations

Grégory Antoni
Aix-Marseille Université, IFSTTAR, LBA UMR T24, 13016 Marseille, France

Correspondence should be addressed to Grégory Antoni; antoni.gregory@yahoo.fr

Received 31 March 2016; Accepted 12 June 2016

1. Introduction

Solving both nonlinear equations and systems is a situation very often encountered in various fields of formal or physical sciences. For instance, solid mechanics is a branch of physics where the resolution of problems governed by nonlinear equations and systems occurs frequently [1–10]. In most cases, Newton method (also known as Newton-Raphson algorithm) is most commonly used for approximating the solutions of scalar and vector nonlinear equations [11–13]. But, over the years, several other numerical methods have been developed for providing iteratively the approximate solutions associated with nonlinear equations and/or systems [14–25]. Some of them present the advantage of having both high accuracy and strong efficiency using a numerical procedure based on an enhanced Newton-Raphson algorithm [26].

In this study, we propose to improve the iterative procedure developed in previous works [27, 28] for finding numerically the solution of both scalar and vector nonlinear equations. This study is decomposed as follows: (i) in Section 2, a new numerical geometry-based root-finding algorithm coupled with a stationary-type iterative method (such as Jacobi or Gauss-Seidel) is presented with the aim of solving any system of nonlinear equations [29, 30]; (ii) in Section 3, the numerical predictive abilities associated with the proposed iterative method are tested on some examples and also compared with other algorithms [31, 32].

2. New Iterative Numerical Method for Solving the Scalar and Vector Nonlinear Equations Based on a Geometric Approach

2.1. Problem Statement. We consider vector-valued function $F : X = \text{trans}(x_1, x_2, x_3, \ldots, x_n) \in \Omega \subset \mathbb{R}^n \rightarrow F(X) \in \mathbb{R}^n$, which is continuous and infinitely differentiable (i.e., $F \in C^\infty(\Omega)$), checking the following equation:

$$F(X) = 0,$$

(1)

where $X = \text{trans}(x_1, x_2, x_3, \ldots, x_n)$ denotes the vector-valued variable (with $n \in \mathbb{N}^*$), $x_i$ is $i$th component associated with vector $X$ (with $1 \leq i \leq n$), $\text{trans}(\cdot)$ is the transpose operator associated with the variable $\cdot$, and $C^\infty(\Omega)$ denotes the class of infinitely differentiable functions in domain $\Omega$. It should be mentioned that: (i) the nonlinear function $F = \text{trans}(F_1, F_2, F_3, \ldots, F_n) \in \mathbb{R}^n$ has a unique solution $X^*$ on domain $\Omega$ which is an open subset of $\mathbb{R}^n$, that is, $\exists X^* \in \Omega \subset \mathbb{R}^n$ such as $F(X^*) = 0 \in \mathbb{R}^n$; (ii) the case of scalar equation ($f$) with only one variable ($x$) is obtained when $n = 1$, that is, $F : X \in \Omega \subset \mathbb{R} \rightarrow F(X) \in \mathbb{R} \Leftrightarrow f : x \in \Omega \subset \mathbb{R} \rightarrow f(x) \in \mathbb{R}$. 

This paper deals with a new numerical iterative method for finding the approximate solutions associated with both scalar and vector nonlinear equations. The iterative method proposed here is an extended version of the numerical procedure originally developed in previous works. The present study proposes to show that this new root-finding algorithm combined with a stationary-type iterative method (e.g., Gauss-Seidel or Jacobi) is able to provide a longer accurate solution than classical Newton-Raphson method. A numerical analysis of the developed iterative method is addressed and discussed on some specific equations and systems.
Equation (1) can also be rewritten as system \((\mathcal{S}_{ml})\) of \(n\)-scalar nonlinear equations, that is,

\[
\begin{align*}
F_1(x_1, x_2, x_3, \ldots, x_i, \ldots, x_n) &= 0 \\
F_2(x_1, x_2, x_3, \ldots, x_i, \ldots, x_n) &= 0 \\
F_3(x_1, x_2, x_3, \ldots, x_i, \ldots, x_n) &= 0 \\
&\vdots \\
F_i(x_1, x_2, x_3, \ldots, x_i, \ldots, x_n) &= 0 \\
&\vdots \\
F_n(x_1, x_2, x_3, \ldots, x_i, \ldots, x_n) &= 0
\end{align*}
\]

with \(1 \leq i \leq n\),

where \(F_i\) denotes the \(i\)th component associated with vector-valued function \(F\) (see (1)), that is, the nonlinear system \((\mathcal{S}_{nl})\) (2). It should be noted that: (i) in the case of \(i \in [1, n]\) (with \(n > 1\)), the nonlinear system \((\mathcal{S}_{nl})\) (2) has a unique solution set \(\{x_1^*, x_2^*, \ldots, x_i^*, \ldots, x_n^*\}\) such as \(F_i(x_1^*, x_2^*, \ldots, x_i^*, \ldots, x_n^*) = 0\); (ii) in the case of \(n = 1\), nonlinear system \((\mathcal{S}_{nl})\) (2) is transformed into scalar nonlinear equation \((\mathcal{S}_{s})\) which has a unique solution \(x^*\) such as \(f(x^*) = 0\); (iii) (1) and (2) are mathematically equivalent, that is, \(x^* = \text{trans}(x_1^*, x_2^*, \ldots, x_i^*, \ldots, x_n^*)\).

With the aim of numerically solving system \((\mathcal{S}_{nl})\) (2), we adopt a Root-Finding Algorithm (RFA) coupled with a Stationary Iterative Procedure (SIP) such as Jacobi or Gauss-Seidel [26, 30]. The use of any SIP allows to reduce the considered nonlinear system \((\mathcal{S}_{nl})\) to a successive set of nonlinear equations with only one variable and therefore it can be solved with a RFA [30]. In the present study, we propose an extended version of RFA already developed in [27, 28] and combined with a Jacobi or Gauss-Seidel type iterative procedure for dealing any system of nonlinear equations.

2.2. Stationary Iterative Procedures (SIPs) with Root-Finding Algorithms (RFAs)

2.2.1. Jacobi and Gauss-Seidel Iterative Procedures. In order to solve a system of nonlinear equations, any RFA can be used if it is combined with a SIP (i.e., Jacobi or Gauss-Seidel) [26, 29, 30]. A Jacobi or Gauss-Seidel type procedure applied to nonlinear system \((\mathcal{S}_{nl})\) (1) can be described as follows:

\[
F_i(x_1^{k+1}; \Delta_i) = 0 \quad \forall i = 1, \ldots, n \quad \forall k = 0, \ldots, m
\]

with

(i) in the case of the Jacobi procedure:

\[
\Delta_i = \{x_1^k, x_2^k, x_3^k, \ldots, x_{i-1}^k, x_{i+1}^k, \ldots, x_n^k\}
\]

(ii) in the case of the Gauss-Seidel procedure:

\[
\Delta_i = \{x_1^{k+1}, x_2^{k+1}, x_3^{k+1}, \ldots, x_{i-1}^{k+1}, x_{i+1}^k, \ldots, x_n^k\}
\]

where \(x^k\) (resp., \(x^{k+1}\)) denotes the \(k\)th (resp., \((k + 1)\)th) iteration associated with the variable \(*\) \((m \in \mathbb{N})\), \(\Delta_i\) is the set of kept constant variables, and \(\dagger\) represents one set of variables \(\dagger\).

2.3. Used Root-Finding Algorithm (RFA). In previous works [27, 28], a root-finding algorithm (RFA) has been developed for approximating the solutions of scalar nonlinear equations. The new RFA presented here is an extended version to that previously developed taking into account some geometric considerations. In this paper, we propose to use a RFA coupled with Jacobi and Gauss-Seidel type procedures for iteratively solving nonlinear system \((\mathcal{S}_{nl})\). Hence, we adopt a new RFA for finding approximate solution \(x_i^{k+1}\) (when \(i\) is fixed and with the known set \(\Delta_i\)) associated with each nonlinear equation \(F_i\) of system \((\mathcal{S}_{nl})\) (see Section 2.1). For each nonlinear equation \(F_i\), parametrized by the set of variables \(\Delta_i\), depending only on one variable \(x_i^{k+1}\), we introduce the exact and inexact local curvature associated with the curve representing the nonlinear equation in question.

The used RFA is based on the following main steps (see [27] for more details):

(i) In the first step, we consider the iterative tangent and normal straight lines \((\mathcal{S}_{s}, \mathcal{N}_{s}) = (T_i(x_i; \Theta_i), N_i(x_i; \Theta_i))\) associated with nonlinear function \(F_i\) at point \(p_i^k = (x_i^k, \mathcal{F}_i^k)\) (see Figure 1):

\[
\mathcal{F}_i^k = \mathcal{F}_i^k(x_i - x_i^k) + \mathcal{F}_i^k
\]

\[
\mathcal{A}_i^k = -\frac{1}{\mathcal{F}_i^k} \left( x_i - x_i^k \right) + \mathcal{F}_i^k
\]

\[
\forall \mathcal{F}_i^k \neq 0 \forall i = 1, \ldots, n \forall k = 0, \ldots, m,
\]

where \(\mathcal{F}_i^k = F_i(x_i^k; \Delta_i)\) (resp., \(\mathcal{A}_i^k = F_i(x_i^k; \Delta_i)\)) denotes the value (resp., first-order derivative) of function \(F_i\) at point \(x_i^k, \Theta_i = \{\Delta_i, x_i^k\}\) is the set of known variables and \((T_i, N_i)\) are two functionals associated with \((\mathcal{S}_{s}, \mathcal{N}_{s})\).

(ii) In the second step, we introduce the iterative exact and inexact local curvature associated with the curve representing nonlinear function \(F_i^k\) at point \(p_i^k = (x_i^k, \mathcal{F}_i^k)\) (see Figure 1):

\[
\frac{1}{\mathcal{F}_i^{k+1}} = \frac{1}{\mathcal{F}_i^k} + \frac{1}{\left( \mathcal{F}_i^k \right)^{3/2}}
\]

\[
\forall \mathcal{F}_i^k \neq 0 \forall \mathcal{F}_i^{k+1} \neq 0 \forall i = 1, \ldots, n \forall k = 0, \ldots, m,
\]

where \(|*|\) denotes the absolute-value function associated with the variable \(*\), \(\mathcal{F}_i^k = \mathcal{F}_i(\Delta_i; \mathcal{F}_i^k)\) is the exact (\(\dagger = \text{ex}\)) or inexact (\(\dagger = \text{in}\)) radius
of the osculating $\sigma^k_i$ at point $x^k_i$, $R_i$ (with $\hat{t} = \text{ex, in}$) is functional associated with $\sigma_i^k$, and $\mathcal{F}_i'' = F_i''(x^k_i; \Delta_i)$ is the second-order derivative of function $F_i$ at point $x^k_i$. It should be noted that: (i) we consider the exact radius $\sigma^k_i$ associated with the true osculating circle $\sigma^k_i$ at point $p^{(k)}_k = (x^k_i, F^k_i)$ (see [33]); (ii) in line with [27, 28], we consider an inexact radius associated with the osculating circle $\sigma^k_i$ at point $p^{(k)}_k = (x^k_i, F^k_i)$, that is, $\sigma^k_i \neq \sigma^k_i$ (see (7)).

(iii) In the third step, we define the iterative center $\sigma^k_i = (\mathcal{F}^k_i, \mathcal{F}^k_i)$ associated with the exact and inexact osculating circle at point $x_i = x^k_i$, that is, (Figure 1)

\[
\begin{align*}
(x^k_i - \mathcal{F}^k_i)^2 + (x^k_i - \mathcal{F}^k_i)^2 &= (\mathcal{F}^k_i)^2 \\
(x^k_i - \mathcal{F}^k_i)^2 + (\mathcal{F}^k_i - \mathcal{F}^k_i)^2 &= (\mathcal{F}^k_i)^2 \\
(x^k_i - \mathcal{F}^k_i)^2 + \left(\frac{1}{\mathcal{F}^k_i} (x^k_i - \mathcal{F}^k_i)^2\right)^2 &= (\mathcal{F}^k_i)^2 \\
(\sigma^k_i - \mathcal{F}^k_i)^2 &= \left(\frac{(\mathcal{F}^k_i)^2}{1 + (\mathcal{F}^k_i)^2}\right)^2 (\mathcal{F}^k_i)^2
\end{align*}
\]

\[\forall \mathcal{F}^k_i \neq 0 \forall \mathcal{F}^k_i \neq 0 \forall i = 1, \ldots, n \forall k = 0, \ldots, m.\]

By taking (7) and (8), the iterative centers $\sigma^k_i = (\mathcal{F}^k_i, \mathcal{F}^k_i)$ are (with $\hat{t} = \text{ex, in}$)

\[
\begin{align*}
\sigma^k_i &= x^k_i + \text{sgn} \left( - \frac{\mathcal{F}^k_i}{\mathcal{F}^k_i} \right) \left| \frac{\mathcal{F}^k_i}{\mathcal{F}^k_i} \left( 1 + \left( \frac{\mathcal{F}^k_i}{\mathcal{F}^k_i} \right)^2 \right) \right| \\
\sigma^k_i &= x^k_i + \text{sgn} \left( - \frac{\mathcal{F}^k_i}{\mathcal{F}^k_i} \right) \left| \frac{\mathcal{F}^k_i}{\mathcal{F}^k_i} \left( 1 + \left( \frac{\mathcal{F}^k_i}{\mathcal{F}^k_i} \right)^2 \right) \right|
\end{align*}
\]



\[\forall \mathcal{F}^k_i \neq 0 \forall \mathcal{F}^k_i \neq 0 \forall i = 1, \ldots, n \forall k = 0, \ldots, m,\]

where $(\mathcal{F}^k_i, \mathcal{F}^k_i) = (\sigma^k_i, \sigma^k_i)$ is the iterative centers associated with the exact and inexact osculating circle $\sigma^k_i$ (with $\hat{t} = \text{ex, in}$) associated with each curve representing nonlinear function $\mathcal{F}^k_i$ at point $x^k_i$, $(\sigma^k_i, \sigma^k_i)$ are two functionals associated with $(\mathcal{F}^k_i, \mathcal{F}^k_i)$ and $\text{sgn}(\sigma)$ is sign function (i.e., $\text{sgn}(\sigma) = -1$ when $\sigma > 0$, $\text{sgn}(\sigma) = 1$ when $\sigma < 0$, and $\text{sgn}(\sigma) = 0$ when $\sigma = 0$).

(iv) In the fourth step, we introduce the iterative point $\sigma^k_i = \mathcal{F}^k_i(x_i^k; \Delta_i)$ such as $\mathcal{F}^k_i = 0$, that is,

\[
\sigma^k_i = x^k_i - \frac{\mathcal{F}^k_i}{\mathcal{F}^k_i}
\]

\[\forall \mathcal{F}^k_i \neq 0 \forall i = 1, \ldots, n \forall k = 0, \ldots, m,\]

where $\mathcal{F}^k_i$ is a functional associated with $\mathcal{F}^k_i$.

(v) In the fifth step, we define the iterative straight line $\mathcal{G}^k_i = G_i(x_i; \Xi_i)$ passing through two iterative
\[ x_{i}^{k+1} = x_i + \frac{1}{\Phi_i^k + 1/\Phi_i^k} \left[ \Phi_i^k \sigma_i^k + \frac{\sigma_i^k}{\Phi_i^k} + \mathcal{F}_i^k \right] \]  

where \( \Phi_i^k = P_i(\Xi_i) = \left( -\left(1/\mathcal{F}_i^{k'} \right) \left( x_i^k - x_i^k \right) + \mathcal{F}_i^k \right) \)  

\[ x_{i}^{k+1} = x_i^k + \frac{\mathcal{F}_i^k}{\Phi_i^k + 1/\Phi_i^k} \left[ 1 - \frac{\Phi_i^k}{\mathcal{F}_i^{k'}} \right] \]  

The new iterative method that we will thereafter appoint as “Adaptive Geometric-based Algorithm” (AGA) enables providing a more convenient approximate solution associated with a system of nonlinear equations of type (\( \delta_m \)), that is, (see Figure 2) 

\[ x_{i}^{k+1} = \begin{cases} \Gamma \left( x_i^k; \beta_i \right) & \text{if } [BC] \\ \alpha_i^k \sigma_i^k & \text{else} \end{cases} \]  

\( \forall i = 1, \ldots, n \ \forall k = 0, \ldots, m \)
with
\[
\Gamma(x^k_i; \beta_i) = x^k_i + \frac{\mathcal{R}^{k}}{\Phi_i^k + 1/\Phi_i^k} \left[ 1 - \frac{\Phi_i^k}{\mathcal{R}_i^k} \right]
\]
\[
\Phi_i^k = \left( -\frac{1}{\mathcal{R}_i^{k'}}(\mathcal{R}^{k'}_i - x^k_i) + \mathcal{R}^{k'}_i \right)^* \mathcal{R}^{k''}_i - \sigma_i^k,
\]
\[
\mathcal{R}_i^{k'} = x^k_i + \text{sgn} \left( -\frac{\mathcal{R}^{k}}{\mathcal{R}_i^{k'}} \right) \frac{\mathcal{R}^{k'}_i}{\mathcal{R}^{k''}_i \mathcal{R}^{k'}_i}
\]
\[
\forall \mathcal{R}_i^{k'} \neq 0 \forall \mathcal{R}_i^{k''} \neq 0 \forall i = 1, \ldots, n \forall k = 0, \ldots, m
\]
and for conditions [BC1], [BC2], [BC4], and [BC5]:
\[
\mathcal{M}_i^k = \begin{cases} 
1 + \left( \mathcal{R}_i^{k'} \right)^2 & \text{if [BC1] or [BC4]} \\
1 & \text{if [BC2] or [BC5]}
\end{cases}
\]
and for conditions [BC3] and [BC6]:
\[
\mathcal{M}_i^k = \begin{cases} 
1 + \left( \mathcal{R}_i^{k'} \right)^2 & \text{if } \dagger = \text{ex} \\
1 & \text{if } \dagger = \text{in},
\end{cases}
\]
where \( \Gamma \) denotes the fixed-point function [13] used to the
covered RFA (i.e., AGA) and \( \beta_i = \{\Delta_i, \mathcal{R}_i^{k} \} \) is the set of
known variables.

The different conditions [BC] associated with the proposed
RFA (i.e., AGA) are as follows:

(i) First condition [BC1] is
\[
\text{sgn} \left( x^k_i \right) = \text{sgn} \left( \Gamma \left( x^k_i, \mathcal{R}_i^{k}, \Delta_i \right) \right),
\]
\[
\text{sgn} \left( F_i \left( x^k_i, \Delta_i \right) \right) = \text{sgn} \left( F_i \left( \Gamma \left( x^k_i, \mathcal{R}_i^{k}, \Delta_i \right) \right) \right),
\]
\[
\forall i = 1, \ldots, n \forall k = 0, \ldots, m.
\]

(ii) Second condition [BC2] is
\[
\text{sgn} \left( x^k_i \right) = \text{sgn} \left( \Gamma \left( x^k_i, \mathcal{R}_i^{k}, \Delta_i \right) \right),
\]
\[
\text{sgn} \left( F_i \left( x^k_i, \Delta_i \right) \right) = \text{sgn} \left( F_i \left( \Gamma \left( x^k_i, \mathcal{R}_i^{k}, \Delta_i \right) \right) \right),
\]
\[
\forall i = 1, \ldots, n \forall k = 0, \ldots, m.
\]

(iii) Third condition [BC3] is
\[
\text{sgn} \left( x^k_i \right) = \text{sgn} \left( \Gamma \left( x^k_i, \mathcal{R}_i^{k}, \Delta_i \right) \right),
\]
\[
\text{sgn} \left( F_i \left( x^k_i, \Delta_i \right) \right) = \text{sgn} \left( F_i \left( \Gamma \left( x^k_i, \mathcal{R}_i^{k}, \Delta_i \right) \right) \right),
\]
\[
\forall i = 1, \ldots, n \forall k = 0, \ldots, m.
\]

(iv) Fourth condition [BC4] is
\[
\text{sgn} \left( F_i \left( \Gamma \left( x^k_i, \mathcal{R}_i^{k}, \Delta_i \right) \right) \right) < \left| F_i \left( \sigma_i^k; \Delta_i \right) \right|
\]
\[
\forall i = 1, \ldots, n \forall k = 0, \ldots, m.
\]

(v) Fifth condition [BC5] is
\[
\text{sgn} \left( F_i \left( \Gamma \left( x^k_i, \mathcal{R}_i^{k}, \Delta_i \right) \right) \right) < \left| F_i \left( \sigma_i^k; \Delta_i \right) \right|
\]
\[
\forall i = 1, \ldots, n \forall k = 0, \ldots, m.
\]

(vi) Sixth condition [BC6] is
\[
\text{sgn} \left( F_i \left( \Gamma \left( x^k_i, \mathcal{R}_i^{k}, \Delta_i \right) \right) \right) < \left| F_i \left( \sigma_i^k; \Delta_i \right) \right|
\]
\[
\forall i = 1, \ldots, n \forall k = 0, \ldots, m.
\]

3. Numerical Examples

3.1. Preliminary Remarks. In this section, we propose to
evaluate the predictive abilities associated with the numerical
iterative method developed in Section 2.3 (i.e., AGA) on some
eamples both in the case of scalar and vector nonlinear
ations. Hence, AGA is compared to other iterative Newton-
phson type methods [27, 28, 30–32] coupled with Jacobi (J) and Gauss-Seidel (GS) techniques. All the numerical
implementations associated with these presented iterative
methods have been made in Matlab software (see [26, 34–39]).

The used iterative methods for the different examples are
as follows (see [27, 28, 30–32]):

(i) Newton-Raphson Algorithm (NRA):
\[
X^{k+1} = X^k - \text{inv} \left( \varnothing^{(1)} \left( F(X^k) \right) \right) \cdot F(X^k)
\]
\[
\forall k = 0, \ldots, m,
\]
where \( \varnothing^{(1)} \left( F(X^k) \right) \) denotes the first-order differential
operator associated with nonlinear function \( F \) at point
\( X^k = (x^k_1, \ldots, x^k_n) \) and \( \text{inv}(\dagger) \) is the inverse transform
operator of \( \dagger \). It is important to highlight that NRA
can be used if and only if operator “\( \text{inv}(\varnothing^{(1)}(F(X^k))) \)”
exists, that is, \( \det(D[F(X_k)]) \neq 0 \), \( \forall X_k \in \Omega \subset \mathbb{R}^n \) (\( \det(\cdot) \) is determinant of operator \( \cdot \)).

(ii) Standard Newton's Algorithm (SNA):

\[
x_i^{k+1} = x_i^k - \frac{F_i(x_i^k; \Delta_i)}{F_i'(x_i^k; \Delta_i)}
\]

\( \forall F_i'(x_i^k; \Delta_i) \neq 0 \) \( \forall i = 1, \ldots, n \) \( \forall k = 0, \ldots, m \).

(iii) Third-order Modified Newton Method (TMNM):

\[
x_i^{k+1} = x_i^k - \frac{F_i(x_i^k; \Delta_i)}{F_i'(x_i^k; \Delta_i)} - \frac{F_i''(x_i^k; \Delta_i)}{2} \left( \frac{F_i'(x_i^k; \Delta_i)}{F_i'(x_i^k; \Delta_i)} \right)^2 
\]

\( \forall F_i'(x_i^k; \Delta_i) \neq 0 \) \( \forall i = 1, \ldots, n \) \( \forall k = 0, \ldots, m \).

In order to stop the iterative process associated with each considered algorithm, we consider three coupled types of criteria for dealing with nonlinear equations:

(i) For scalar-valued equations (Snl):

(a) (CIS) on the iteration number,

\( m \leq K_{\text{max}}^s \)

where \( K_{\text{max}}^s \) denotes the maximum number of iterations associated with scalar-valued equations.

(b) (C2S) on the residual error,

\[ \| f(x^{k+1}) \| \leq \epsilon_{\text{re}}^s \quad \forall k = 0, \ldots, m, \]

where \( \epsilon_{\text{re}}^s \) is the tolerance parameter associated with the residual error criterion for scalar-valued equations and \( \| \cdot \| \) is the absolute-value norm.

(c) (C3S) on the approximation error,

\[ \| x^{k+1} - x^k \| \leq \epsilon_{\text{ae}}^s \quad \forall k = 0, \ldots, m, \]

where \( \epsilon_{\text{ae}}^s \) is the tolerance parameter associated with the absolute error criterion for scalar-valued equations.

(ii) For vector-valued equations (Vnl):

(a) (CIV) on the iteration number, we adopt the same condition that (CIS), that is, \( m \leq K_{\text{max}}^v = K_{\text{max}}^v \) (where \( K_{\text{max}}^v \) denotes the maximum number of iterations associated with vector-valued equations).

(b) (C2V) on the residue error,

\[ \| F(x^k) \|_2 \leq \epsilon_{\text{re}}^v \]

or \( \| F(x^k) \|_2 \leq \epsilon_{\text{re}}^v \)

\( \forall k = 0, \ldots, m, \)

where \( \epsilon_{\text{re}}^v \) (resp., \( \epsilon_{\text{re}}^v \)) is the tolerance parameter associated with the residue error criterion for vector-valued equations and \( \| \cdot \|_p \) is the vector \( p \)-norm (here, \( p = 1, 2 \)). It is important to point out that \( \| \cdot \|_p = (\sum_{i=1}^n |Y_i|^p)^{1/p} = (|Y_1|^p + |Y_2|^p + \cdots + |Y_n|^p)^{1/p} \) is the so-called Euclidean norm.

(c) (C3V) on the approximation error,

\[ \| x^{k+1} - x^k \|_2 \leq \epsilon_{\text{ae}}^v \]

or \( \| x^{k+1} - x^k \|_2 \leq \epsilon_{\text{ae}}^v \)

\( \forall k = 0, \ldots, m, \)

where \( \epsilon_{\text{ae}}^v \) is the tolerance parameter associated with the absolute error criterion for vector-valued equations.

Here, for the stopping criteria (CIS, CIV), (C2S, C2V), and (C3S, C3V) associated with the iterative process, we consider: (i) the maximum number of iterations \( K_{\text{max}}^s = K_{\text{max}}^v \) = 20; (ii) the tolerance parameter \( \epsilon_{\text{re}}^s = \epsilon_{\text{re}}^v = 10^{-30} \) for the scalar-valued equations; (iii) the tolerance parameter \( \epsilon_{\text{re}}^v = \epsilon_{\text{ae}}^v = 10^{-30} \) (with \( i = 1, 2 \)) for the vector-valued equations.

3.2. Examples. We consider the following nonlinear equations:

(i) In case \( n = 1 \) (i.e., scalar-valued equations), one has the following:

**Example 1.** Consider the following:

\( \left( \phi_{nI}^1 \right) : f(x) = 10x^3 + 5 \log(x) - 2 = 0 \)

(34)

**Example 2.** Consider the following:

\( \left( \phi_{nI}^2 \right) : f(x) = 3 \exp(x) + x^3 \cos(x) - 30 = 0 \)

(35)

(ii) In case \( n = 2 \) (i.e., vector-valued equations), one has the following:

**Example 3.** Consider the following:

\( \left( \phi_{nI}^3 \right) : \begin{cases} F_1(x_1, x_2) = 3(x_1)^3 - 2x_2 - 5 = 0 \\ F_2(x_1, x_2) = (x_1)^2 - 2x_1(x_2)^2 = 0 \end{cases} \)

(36)
Consider the following:

**Example 4.** Consider the following:

\[
(x^{H})_{nl}: \begin{cases}
F_1(x_1, x_2) = x_1 \ln(x_2) + 4 (x_1)^3 - 5 = 0 \\
F_2(x_1, x_2) = (x_2)^4 + x_1 x_2 - 6 = 0
\end{cases}
\]

(37)

### 3.3. Results and Discussion.

All the numerical results of Examples 1–4 are shown in Figures 3–32. For Example 1 (resp., Example 2) with guest start point \(x_1^0 = 5\) (resp., \(x_1^0 = 10\), we can see that approximate solutions \(x^k_1\) provided by AGA with condition \([BC1]([BC3]\) (i.e., condition \([BC1]\)

is the same that \([BC3]\) where \(\dagger = \text{ex})\) are better than AGA with condition \([BC2]\), NRA/SNA (when \(n = 1\), NRA and SNA are the same), and TMNM. For Example 1 (resp., Example 2) in the case of guest start point \(x^0_1 = 10^{-2}\) (resp., \(x^0_1 = 1\), we can see that approximate solutions \(x^k_1\) provided by AGA with conditions \([BC4],[BC5],\) and \([BC6]\) in the first iterations are accurately better than NRA/SNA (when \(n = 1\), NRA and SNA are the same) and TMNM. For Example 3 with guest start point couple \((x^0_1, x^0_2) = (20, 20)\), we can observe that approximate solutions \((x^k_1, x^k_2)\) given by AGA using Gauss-Seidel (GS) or Jacobi (J) procedure with: (i) condition \([BC1]\) are more accurately than NRA and SNA; (ii) conditions \([BC2]\) and \([BC3]\) are accurately better than NRA (only in the first iterations) and SNA. In the case of first start point couple \((x^0_1, x^0_2) = (0.42, 0.42)\), we can see that approximate solutions \((x^k_1, x^k_2)\) provided by AGA using: (i) Gauss-Seidel (GS) procedure with condition \([BC6]\) give enough numerical accuracy than NRA and SNA; (ii) Gauss-Seidel (GS) procedure with conditions \([BC4]\) and \([BC5]\) offer much greater numerical accuracy than NRA and SNA; (iii) Jacobi (J) procedure with conditions \([BC4],[BC5],\) and \([BC6]\) are better than NRA (only in the first iterations) and SNA. For Example 4 with guest start point couple \((x^0_1, x^0_2) = (10, 10)\), we can see that approximate solutions \((x^k_1, x^k_2)\) given by AGA using: (i) Gauss-Seidel (GS) procedure with conditions \([BC1]\) and \([BC6]\) are more accurately than TMNM, and SNA; (ii) Gauss-Seidel (GS) procedure with condition \([BC2]\) are accurately better than TMNM and NRA (only in the first iterations) and SNA; (iii) Jacobi (J) procedure with conditions \([BC1],[BC2],\) and \([BC3]\) are more accurately numerically than NRA (only in the first iterations) and both TMNM and SNA. In the case of first start point couple \((x^0_1, x^0_2) = (0.5, 0.5)\), approximate solutions \((x^k_1, x^k_2)\) obtained by AGA using: (i) Gauss-Seidel (GS) procedure with conditions \([BC5]\) and \([BC6]\) are more accurately than SNA and NRA; (ii) Jacobi (J) procedure with the conditions \([BC4],[BC5],\) and \([BC6]\) are better than NRA (only in the first iterations) and SNA. The present work concerns a new numerical iterative method for approximating the solutions of both scalar and vector nonlinear equations. Based on an iterative procedure previously developed in a study, we propose here an extended form of this numerical algorithm including the use of a stationarity type iterative procedure in order to solve systems of nonlinear equations. A predictive numerical analysis associated with this proposed method for providing a more accurate approximate solution in regard to nonlinear equations and systems and can potentially provide a better or more suitable approximate solution than that of Newton-Raphson Algorithm (NRA).

### 4. Concluding Comments

The present work concerns a new numerical iterative method for approximating the solutions of both scalar and vector nonlinear equations. Based on an iterative procedure previously developed in a study, we propose here an extended form of this numerical algorithm including the use of a stationarity type iterative procedure in order to solve systems of nonlinear equations. A predictive numerical analysis associated with this proposed method for providing a more accurate approximate solution in regard to nonlinear equations and systems and can potentially provide a better or more suitable approximate solution than that of Newton-Raphson Algorithm (NRA).
Figure 5: Evolution of residue error (C2S) and approximation error (C3S) associated with \((E_{	ext{nl}})_{k}\) compared to \(k\)th iteration for Example 1 (where \(x^0 = 5\)) with NRA/SNA (black solid line with diamonds and squares), TMNM (green solid line with diamonds and squares), and AGA with condition \([BC1]/[BC3]\) (blue solid line with diamonds and squares) and condition \([BC2]\) (red solid line with diamonds and squares).

Figure 6: Evolution of residue error (C2S) and approximation error (C3S) associated with \((E_{	ext{nl}})_{k}\) compared to \(k\)th iteration for Example 1 (where \(x^0 = 10^{-2}\)) with NRA/SNA (black solid line with diamonds and squares), TMNM (green solid line with diamonds and squares), and AGA with condition \([BC4]\) (blue solid line with diamonds and squares) and condition \([BC5]/[BC6]\) (magenta solid line with diamonds and squares).

Figure 7: Evolution of approximate solutions \(x^k\) associated with \((E_{	ext{nl}})_{k}\) compared to \(k\)th iteration for Example 2 (where \(x^0 = 10\)) with NRA/SNA (black solid line with circles), TMNM (green solid line with circles), and AGA with condition \([BC1]/[BC3]\) (blue solid line with circles) and condition \([BC2]\) (red solid line with circles).

Figure 8: Evolution of approximate solutions \(x^k\) associated with \((E_{	ext{nl}})_{k}\) compared to \(k\)th iteration for Example 2 (where \(x^0 = 1\)) with NRA/SNA (black solid line with circles), TMNM (green solid line with circles), and AGA with condition \([BC4]\) (blue solid line with circles), condition \([BC5]\) (red solid line with circles), and condition \([BC6]\) (magenta solid line with circles).
Figure 9: Evolution of residue error (C2S) and approximation error (C2S) associated with $(\hat{\delta}_{nl}^{\text{II}})$ compared to $k$th iteration for Example 1 (where $x^0 = 10$) with NRA/SNA (black solid line with diamonds and squares), TMNM (green solid line with diamonds and squares), and AGA with condition $[BC1]/[BC3]$ (blue solid line with diamonds and squares) and condition $[BC2]$ (red solid line with diamonds and squares).

Figure 10: Evolution of residue error (C2S) and approximation error (C3S) associated with $(\hat{\delta}_{nl}^{\text{II}})$ compared to $k$th iteration for Example 2 (where $x^0 = 1$) with NRA/SNA (black solid line with diamonds and squares), TMNM (green solid line with diamonds and squares), and AGA with condition $[BC4]$ (blue solid line with diamonds and squares), condition $[BC5]$ (red solid line with diamonds and squares), and condition $[BC6]$ (magenta solid line with diamonds and squares).

Figure 11: Evolution of approximate solutions $(x_k^1, x_k^2)$ associated with $(\delta_{nl}^{\text{II}})$ compared to $k$th iteration for Example 3 (where $(x_0^1, x_0^2) = (20, 20)$) with NRA (black solid line with circles) and some other algorithms coupled with Gauss-Seidel (GS) procedure: SNA-GS (cyan solid line with circles), TMNM-GS (green solid line with circles), AGA-GS with condition $[BC1]$ (blue solid line with circles), condition $[BC2]$ (red solid line with circles), and condition $[BC3]$ (magenta solid line with circles).
Figure 12: Evolution of approximate solutions \((x_k^1, x_k^2)\) associated with \((\delta_{ij}^k)\) compared to \(k\)th iteration for Example 3 (where \((x_0^1, x_0^2) = (20, 20)\) with NRA (black solid line with circles) and some other algorithms coupled with Jacobi (J) procedure: SNA-J (cyan dashed line with circles), TMNM-J (green dashed line with circles), and AGA-J with condition [BC1] (blue dashed line with circles), condition [BC2] (red dashed line with circles), and condition [BC3] (magenta dashed line with circles).

Figure 13: Evolution of residue error using (C2V1) and (C2V2) conditions associated with \((\delta_{ij}^k)\) compared to \(k\)th iteration for Example 3 (where \((x_0^1, x_0^2) = (20, 20)\) with NRA (black solid line with diamonds) and some other algorithms coupled with Gauss-Seidel (GS) procedure: SNA-GS (cyan solid line with diamonds), TMNM-GS (green solid line with diamonds), and AGA-GS with condition [BC1] (blue solid line with diamonds), condition [BC2] (red solid line with diamonds), and condition [BC3] (magenta solid line with diamonds).

Figure 14: Evolution of residue error using (C2V1) and (C2V2) conditions associated with \((\delta_{ij}^k)\) compared to \(k\)th iteration for Example 3 (where \((x_0^1, x_0^2) = (20, 20)\) with NRA (black solid line with diamonds) and some other algorithms coupled with Jacobi (J) procedure: SNA-J (cyan dashed line with diamonds), TMNM-J (green dashed line with diamonds), and AGA-J with condition [BC1] (blue dashed line with diamonds), condition [BC2] (red dashed line with diamonds), and condition [BC3] (magenta dashed line with diamonds).
Figure 15: Evolution of the approximation error using (C3V1) and (C3V2) conditions associated with $(\delta_{\text{in}}^1)$ compared to $k$th iteration for Example 3 (where $(x_0^1, x_0^2) = (20, 20)$) with NRA (black solid line with squares) and some other algorithms coupled with Gauss-Seidel (GS) procedure: SNA-GS (cyan solid line with squares), TMNM-GS (green solid line with squares), and AGA-GS with condition [BC1] (blue solid line with squares), condition [BC2] (red solid line with squares), and condition [BC3] (magenta solid line with squares).

Figure 16: Evolution of the approximation error using (C3V1) and (C3V2) conditions associated with $(\delta_{\text{in}}^1)$ compared to $k$th iteration for Example 3 (where $(x_0^1, x_0^2) = (20, 20)$) with NRA (black solid line with squares) and some other algorithms coupled with Jacobi (J) procedure: SNA-J (cyan dashed line with squares), TMNM-J (green dashed line with squares), and AGA-J with condition [BC1] (blue dashed line with squares), condition [BC2] (red dashed line with squares), and condition [BC3] (magenta dashed line with squares).

Figure 17: Evolution of approximate solutions $(x_k^1, x_k^2)$ associated with $(\delta_{\text{in}}^1)$ compared to $k$th iteration for Example 3 (where $(x_0^1, x_0^2) = (0.42, 0.42)$) with NRA (black solid line with circles) and some other algorithms coupled with Gauss-Seidel (GS) procedure: SNA-GS (cyan solid line with circles) and AGA-GS with condition [BC4] (blue solid line with circles), condition [BC5] (red solid line with circles), and condition [BC6] (magenta solid line with circles).
Figure 18: Evolution of approximate solutions \((x_k^1, x_k^2)\) associated with \((\delta_k^1)\) compared to \(k\)th iteration for Example 3 (where \((x_0^1, x_0^2) = (0.42, 0.42)\)) with NRA (black solid line with diamonds and squares) and some other algorithms coupled with Jacobi (J) procedure: SNA-J (cyan dashed line with circles) and AGA-J with condition [BC4] (blue dashed line with circles), condition [BC5] (red dashed line with circles), and condition [BC6] (magenta dashed line with circles).

Figure 19: Evolution of residue error using (C2V1) and (C2V2) conditions associated with \((\delta_k^2)\) compared to \(k\)th iteration for Example 3 (where \((x_0^1, x_0^2) = (0.42, 0.42)\)) with NRA (black line) and some other algorithms coupled with Gauss-Seidel (GS) and Jacobi (J) procedures: SNA-GS/SNA-J (cyan line) and AGA-GS/AGA-J with condition [BC4] (blue line), condition [BC5] (red line), and condition [BC6] (magenta line).

Figure 20: Evolution of approximation error for (C3V1) and (C3V2) conditions associated with \((\delta_k^3)\) compared to \(k\)th iteration for Example 3 (where \((x_0^1, x_0^2) = (0.42, 0.42)\)) with NRA (black line) and some other algorithms coupled with Gauss-Seidel (GS) and Jacobi (J) procedures: SNA-GS/SNA-J (cyan line) and AGA-GS/AGA-J with condition [BC4] (blue line), condition [BC5] (red line), and condition [BC6] (magenta line).
Figure 21: Evolution of approximate solutions \((x_k^1, x_k^2)\) associated with \((\delta_{1}^{\text{II}})\) compared to \(k\)th iteration for Example 4 (where \((x_0^1, x_0^2) = (10, 10)\)) with NRA (black solid line with circles) and some other algorithms coupled with Gauss-Seidel (GS) procedure: SNA-GS (cyan solid line with circles), TMNM-GS (green solid line with circles), and AGA-GS with condition [BC1] (blue solid line with circles), condition [BC2] (red solid line with circles), and condition [BC3] (magenta solid line with circles).

Figure 22: Evolution of approximate solutions \((x_k^1, x_k^2)\) associated with \((\delta_{1}^{\text{II}})\) compared to \(k\)th iteration for Example 4 (where \((x_0^1, x_0^2) = (10, 10)\)) with NRA (black solid line with circles) and some other algorithms coupled with Jacobi (J) procedure: SNA-J (cyan dashed line with circles), TMNM-J (green dashed line with circles), and AGA-J with condition [BC1] (blue dashed line with circles), condition [BC2] (red dashed line with circles), and condition [BC3] (magenta dashed line with circles).

Figure 23: Evolution of residue error using (C2V1) and (C2V2) conditions associated with \((\delta_{1}^{\text{II}})\) compared to \(k\)th iteration for Example 4 (where \((x_0^1, x_0^2) = (10, 10)\)) with NRA (black solid line with diamonds) and some other algorithms coupled with Gauss-Seidel (GS) procedure: SNA-GS (cyan solid line with diamonds), TMNM-GS (green solid line with diamonds), and AGA-GS with condition [BC1] (blue solid line with diamonds), condition [BC2] (red solid line with diamonds), and condition [BC3] (magenta solid line with diamonds).
Figure 24: Evolution of residue error using (C2V1) and (C2V2) conditions associated with $(S_{II})$ compared to $k$th iteration for Example 4 (where $(x_0, x_1) = (10, 10)$) with NRA (black solid line with diamonds) and some other algorithms coupled with Jacobi (J) procedure: SNA-J (cyan dashed line with diamonds), TMNM-J (green dashed line with diamonds), and AGA-J with condition [BC1] (blue dashed line with diamonds), condition [BC2] (red dashed line with diamonds), and condition [BC3] (magenta dashed line with diamonds).

Figure 25: Evolution of the approximation error using (C3V1) and (C3V2) associated with $(S_{II})$ compared to $k$th iteration for Example 4 (where $(x_0, x_1) = (10, 10)$) with NRA (black solid line with squares) and some other algorithms coupled with Gauss-Seidel (GS) procedure: SNA-GS (cyan solid line with squares), TMNM-GS (green solid line with squares), and AGA-GS with condition [BC1] (blue solid line with squares), condition [BC2] (red solid line with squares), and condition [BC3] (magenta solid line squares).

Figure 26: Evolution of the approximation error using (C3V1) and (C3V2) conditions associated with $(S_{II})$ compared to $k$th iteration for Example 4 (where $(x_0, x_1) = (10, 10)$) with NRA (black solid line with squares) and some other algorithms coupled with Jacobi (J) procedure: SNA-J (cyan dashed line with squares), TMNM-J (green dashed line with squares), and AGA-J with condition [BC1] (blue dashed line with squares), condition [BC2] (red dashed line with squares), and condition [BC3] (magenta dashed line with squares).
Figure 27: Evolution of approximate solutions \((x_k^1, x_k^2)\) associated with \((\delta_{II}^n)\) compared to \(k\)th iteration for Example 4 (where \((x_0^1, x_0^2) = (0.5, 0.5)\)) with NRA (black solid line with circles) and some other algorithms coupled with Gauss-Seidel (GS) procedure: SNA-GS (cyan solid line with circles) and AGA-GS with condition [BC4] (blue solid line with circles), condition [BC5] (red solid line with circles), and condition [BC6] (magenta solid line with circles).

Figure 28: Evolution of approximate solutions \((x_k^1, x_k^2)\) associated with \((\delta_{II}^n)\) compared to \(k\)th iteration for Example 4 (where \((x_0^1, x_0^2) = (0.5, 0.5)\)) with NRA (black solid line with circles) and some other algorithms coupled with Jacobi (J) procedure: SNA-J (cyan dashed line with circles) and AGA-J with condition [BC4] (blue dashed line with circles), condition [BC5] (red dashed line with circles), and condition [BC6] (magenta dashed line with circles).

Figure 29: Evolution of the residue error using (C2V1) and (C2V2) conditions associated with \((\delta_{II}^n)\) compared to \(k\)th iteration for Example 4 (where \((x_0^1, x_0^2) = (0.5, 0.5)\)) with NRA (black solid line with diamonds) and some other algorithms coupled with Gauss-Seidel (GS) procedure: SNA-GS (cyan solid line with diamonds) and AGA-GS with condition [BC4] (blue solid line with diamonds), condition [BC5] (red solid line with diamonds), and condition [BC6] (magenta solid line with diamonds).
Figure 30: Evolution of residue error using (C2V1) and (C2V2) conditions associated with $\left( x_{01}^{(1)} \right)^{m}$ compared to $k$th iteration for Example 4 (where $(x_{01}^{0}, x_{02}^{0}) = (0.5, 0.5)$) with NRA (black solid line with diamonds) and some other algorithms coupled with Jacobi ($J$) procedure: SNA-J (cyan dashed line with diamonds) and AGA-J with condition $[BC4]$ (blue dashed line with diamonds), condition $[BC5]$ (red dashed line with diamonds), and condition $[BC6]$ (magenta dashed line with diamonds).

Figure 31: Evolution of approximation error using (C3V1) and (C3V2) conditions associated with $\left( x_{01}^{(1)} \right)^{m}$ compared to $k$th iteration for Example 4 (where $(x_{01}^{0}, x_{02}^{0}) = (0.42, 0.42)$) with NRA (black solid line with squares) and some other algorithms coupled with Gauss-Seidel ($GS$) procedure: SNA-GS (cyan solid line with squares) and AGA-GS with condition $[BC4]$ (blue solid line with squares), condition $[BC5]$ (red solid line with squares), and condition $[BC6]$ (magenta solid line with squares).

Figure 32: Evolution of the approximation error using (C3V1) and (C3V2) conditions associated with $\left( x_{01}^{(1)} \right)^{m}$ compared to $k$th iteration for Example 4 (where $(x_{01}^{0}, x_{02}^{0}) = (0.42, 0.42)$) with NRA (black solid line with squares) and some other algorithms coupled with Jacobi ($J$) procedure: SNA-J (cyan dashed line with squares) and AGA-J with condition $[BC4]$ (blue dashed line with squares), condition $[BC5]$ (red dashed line with squares), and condition $[BC6]$ (magenta dashed line with squares).
systems is tested, assessed and discussed on some specific examples.

Competing Interests
The author declares that there are no competing interests regarding the publication of this paper.

References
[1] O. C. Zienkiewicz and R. L. Taylor, The Finite Element Method: Solid Mechanics, vol. 2, Butterworth-Heinemann, 5th edition, 2000.
[2] T. Belytschko, W. K. Liu, and B. Moran, Nonlinear Finite Elements for Continua and Structures, John Wiley & Sons, New York, NY, USA, 2000.
[3] J. R. Huges, The Finite Element Method: Linear Static and Dynamic Finite Element Analysis, Dover Civil and Mechanical Engineering, Dover, New York, NY, USA, 2000.
[4] I. Doghri, Mechanics of Deformable Solids: Linear, Nonlinear, Analytical and Computational Aspects, Springer, Berlin, Germany, 2000.
[5] A. Curnier, Méthodes Numériques en Mécanique des Solides, Presses Polytechniques et Universitaires Romandes, 2000.
[6] M. Kojić and K.-J. Bathe, Inelastic Analysis of Solids and Structures, Computational Fluid and Solid Mechanics, Springer, Berlin, Germany, 2005.
[7] M. Bonnet and A. Frangi, Analyse des Solides Déformables par la Méthode des Éléments Finis, Editions de l'Ecole Polytechnique, Paris, France, 2007.
[8] J. Besson, G. Cailletaud, J. L. Chaboche, and S. Forest, Nonlinear Mechanics of Materials, vol. 167 of Solid Mechanics and Its Applications, Springer, New York, NY, USA, 2010.
[9] R. De Borst, M. A. Crisfield, J. J. C. Remmers, and C. V. Verhoosel, Non-Linear Finite Element Analysis of Solids and Structures, Computational Mechanics, Wiley-Blackwell, 2012.
[10] M. Bonnet, A. Frangi, and C. Rey, The Finite Element Method in Solid Mechanics, McGraw-Hill Education, New York, NY, USA, 2014.
[11] C. T. Kelley, Solving Nonlinear Equations with Newton's Method, vol. 1 of Fundamental Algorithms for Numerical Calculations, SIAM, Philadelphia, PA, USA, 2003.
[12] P. Deuflhard, Newton Methods for Nonlinear Problems. Computational Mathematics, vol. 35, Springer, New York, NY, USA, 2005.
[13] J. P. Dedieu, Points Fixes, Zéros et la Méthode de Newton, Mathématiques et Applications, Springer, 2006.
[14] R. W. Hamming, Numerical Methods for Scientists and Engineers, Dover, New York, NY, USA, 2nd edition, 1987.
[15] W. C. Rheinboldt, Methods for Solving Systems of Nonlinear Equations, CBMS- NSF Regional Conference Series in Applied Mathematics, Book 70, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 1987.
[16] C. T. Kelley, Iterative Methods for Linear and Nonlinear Equations. Number 16 in Frontiers in Applied Mathematics, SIAM, Philadelphia, PA, USA, 1995.
[17] M. T. Darvishi and A. Barati, "A third-order Newton-type method to solve systems of nonlinear equations," Applied Mathematics and Computation, vol. 187, no. 2, pp. 630–635, 2007.
[18] A. Golbabai and M. Javidi, "A third-order Newton type method for nonlinear equations based on modified homotopy perturbation method," Applied Mathematics and Computation, vol. 191, no. 1, pp. 199–205, 2007.
[19] M. A. Noor and M. Waseem, "Some iterative methods for solving a system of nonlinear equations," Computers and Mathematics with Applications, vol. 57, no. 1, pp. 101–106, 2009.
[20] S. Amat, S. Busquier, C. Bermúdez, and S. Plaza, "On two families of high order Newton type methods," Applied Mathematics Letters, vol. 25, no. 12, pp. 2209–2217, 2012.
[21] J. R. Sharma and H. Arora, "On efficient weighted-Newton methods for solving systems of nonlinear equations," Applied Mathematics and Computation, vol. 222, pp. 497–506, 2013.
[22] M. S. Petković, B. Neta, L. D. Petković, and J. Džunić, "Multipoint methods for solving nonlinear equations: a survey," Applied Mathematics and Computation, vol. 226, pp. 635–660, 2014.
[23] J. R. Sharma and P. Gupta, "An efficient fifth order method for solving systems of nonlinear equations," Computers and Mathematics with Applications, vol. 67, no. 3, pp. 591–601, 2014.
[24] A. Singh and J. P. Jaiswal, "An efficient family of optimal eighth-order iterative methods for solving nonlinear equations and its dynamics," Journal of Mathematics, vol. 2014, Article ID 569719, 14 pages, 2014.
[25] M.-D. Junjua, S. Akram, N. Yasmin, and F. Zafar, "A new Jarratt-type fourth-order method for solving nonlinear equations and applications," Journal of Applied Mathematics, vol. 2015, Article ID 805278, 14 pages, 2015.
[26] A. Quarteroni, R. Sacco, and F. Saleri, Méthodes Numériques Pour le Calcul Scientifique: Programmes en MATLAB, Springer, New York, NY, USA, 2000.
[27] G. Antoni, "A new iterative algorithm for approximating zeros of nonlinear scalar equations: geometry-based solving procedure," Asian Journal of Mathematics and Computer Research, vol. 10, no. 2, pp. 78–97, 2016.
[28] G. Antoni, "A geometry-based iterative algorithm for finding the approximate solution of systems of nonlinear equations," Asian Journal of Mathematics and Computer Research, In press.
[29] M. N. Vrahatis, G. D. Magoulas, and V. P. Plagianakos, "From linear to nonlinear iterative methods," Applied Numerical Mathematics, vol. 45, no. 1, pp. 59–77, 2003.
[30] G. Antoni, "A new Newton-type method for solving both nonlinear equations and systems," Asian Journal of Mathematics and Computer Research, vol. 6, no. 3, pp. 193–212, 2015.
[31] G. Antoni, "A new class of two-step iterative algorithms for finding roots of nonlinear equations," Asian Journal of Mathematics and Computer Research, vol. 7, no. 3, pp. 175–189, 2016.
[32] G. Antoni, "Some iterative algorithms with sub-steps to solve systems of non-linear equations," Asian Journal of Mathematics and Computer Research, vol. 9, no. 3, pp. 214–227, 2016.
[33] F. P. Miller, A. F. Vandome, and J. McBrewster, Frenet-Serret Formulas: Vector Calculus, Curve, Derivative, Euclidean Space, Kinematics, Darboux Frame, Differential Geometry of Curves, Affine Geometry of Curves, Alphascript, San Carlos, Calif, USA, 2010.
[34] G. W. Recktenwald, Numerical Methods with MATLAB: Implementations and Applications, Prentice Hall, New York, NY, USA, 2nd edition, 2000.
[35] J. H. Mathews and K. K. Fink, Numerical Methods Using MATLAB, Prentice Hall, New Jersey, NJ, USA, 4th edition, 2004.
[36] W. Y. Yang, W. Cao, T. S. Chung, and J. Morris, *Applied Numerical Methods Using MATLAB*, Wiley Interscience, 1st edition, 2005.

[37] J. Kiusalaas, *Numerical Methods in Engineering with MATLAB*, Cambridge University Press, Cambridge, UK, 2nd edition, 2009.

[38] R. Butt, *Introduction to Numerical Analysis Using MATLAB*, Jones & Bartlett Learning, 1st edition, 2009.

[39] S. C. Chapra, *Applied Numerical Methods with MATLAB for Engineers and Scientists*, McGraw-Hill Higher Education, 3rd edition, 2011.
Submit your manuscripts at
http://www.hindawi.com