Two-Step Fifth-Order Efficient Jacobian-Free Iterative Method for Solving Nonlinear Systems

Alicia Cordero *, Javier G. Maimó , Antmel Rodríguez-Cabral , And Juan R. Torregrosa

Posted Date: 11 September 2024

doi: 10.20944/preprints202409.0876.v1

Keywords: Nonlinear systems; Iterative processes; Convergence order; Efficiency

Article

Two-Step Fifth-Order Efficient Jacobian-Free Iterative Method for Solving Nonlinear Systems

Alicia Cordero [1], Javier G. Maimó [2], Antmel Rodríguez-Cabral [2] and Juan R. Torregrosa [1]

[1] Instituto de Matemática Multidisciplinar, Universitat Politècnica de València, Camino de Vera, s/n, 46022 Valencia, Spain
[2] Instituto Tecnológico de Santo Domingo (INTEC), Av. Los Próceres, Gala, Santo Domingo, Dominican Republic
* Correspondence: acordero@mat.upv.es

Abstract: This article introduces a novel two-step fifth-order Jacobian-free iterative method aimed at efficiently solving systems of nonlinear equations. The method leverages the benefits of Jacobian-free approaches, utilizing divided differences to circumvent the computationally intensive calculation of Jacobian matrices. This adaptation significantly reduces computational overhead and simplifies the implementation process while maintaining high convergence rates. We demonstrate that this method achieves fifth-order convergence under specific parameter settings, with a broad applicability across various types of nonlinear systems. The effectiveness of the proposed method is validated through a series of numerical experiments which confirm its superior performance in terms of accuracy and computational efficiency compared to existing methods.

Keywords: nonlinear systems; iterative processes; convergence order; efficiency

1. Introduction

Let $F(x) = 0$ be a nonlinear system of equations, $F : D \subset \mathbb{R}^n \to \mathbb{R}^n$, and the functions f_i for $i = 1, 2, \ldots, n$, are the coordinate components of F, expressed as $F(x) = (f_1(x), f_2(x), \ldots, f_n(x))^T$. Solving nonlinear systems is generally challenging, and solutions ξ are typically found by linearizing the problem or employing a fixed-point iteration function $G : D \subset \mathbb{R}^n \to \mathbb{R}^n$, leading to an iterative fixed-point method. Among the various root-finding techniques for nonlinear systems, Newton's method is the most well-known, which follows the second-order iterative procedure:

$$x^{(k+1)} = x^{(k)} - [F'(x^{(k)})]^{-1} F(x^{(k)}),\ k = 0, 1, \ldots,$$

being $F'(x^{(k)})$ the Jacobian matrix of F at the k-th iterate.

Recenty, many researchers have focused on developing iterative methods that outperform Newton's method in terms of both efficiency and order of convergence. Numerous approaches need the computation of F' at different points along each iteration. Nevertheless, calculating the Jacobian poses significant challenges, particularly in high-dimensional problems, where its computation can be costly or even impractical. In some instances, the Jacobian may not exist at all.

To address this issue, alternative approaches have been proposed, such as replacing the Jacobian matrix with a divided difference operator. One of the simplest alternatives is the multidimensional version of Steffensen's method, attributed to Samanskii [1,2], which substitutes the Jacobian in Newton's procedure with a first-order operator of divided differences:

$$x^{(k+1)} = x^{(k)} - [x^{(k)}, z^{(k)}; F]^{-1} F(x^{(k)}), \quad k = 0, 1, \ldots,$$

being $z^{(k)} = x^{(k)} + F(x^{(k)})$, and $[\cdot, \cdot; F] : \Omega \times \Omega \subset \mathbb{R}^n \times \mathbb{R}^n \to \mathcal{L}(\mathbb{R}^n)$ is the operator of divided differences related to F [3],

$$[y, x; F](y - x) = F(y) - F(x) \text{ for any } x, y \in \Omega.$$

 doi:10.20944/preprints202409.0876.v1

This substitution retains the 2-nd order of convergence while bypassing the calculation of the F'.

Although both Steffensen and Newton methods exhibit quadratic convergence, it has been shown that Steffensen's scheme is less stable than Newton's method, with stability depending more on the initial guess. This has been thoroughly analyzed in [4] and [5], where it was found that, for scalar cases $f(x) = 0$, derivative-free iterative methods become more stable when selecting $z = x + \alpha f(x)$ for small real values of α.

However, substituting the Jacobian with divided differences can result in lower convergence order for some iterative methods. For example, the multidimensional version of Ostrowski's fourth-order method (see [6], [13]):

$$\begin{aligned} y^{(k)} &= x^{(k)} - [F'(x^{(k)})]^{-1} F(x^{(k)}), \\ x^{(k+1)} &= y^{(k)} - \left[2[y^{(k)}, x^{(k)}; F] - F'(x^{(k)})\right]^{-1} F(y^{(k)}), \quad k = 0, 1, \ldots, \end{aligned}$$

achieves only cubic convergence if $F'(x)$ is replaced by $[y, x; F]$, as follows:

$$\begin{aligned} y^{(k)} &= x^{(k)} - [x^{(k)} + F(x^{(k)}), x^{(k)}; F]^{-1} F(x^{(k)}), \; k = 0, 1, \ldots, \\ x^{(k+1)} &= y^{(k)} - \left[2[y^{(k)}, x^{(k)}; F] - [x^{(k)} + F(x^{(k)}), x^{(k)}; F]\right]^{-1} F(y^{(k)}). \end{aligned}$$

Other fourth-order methods also loose their convergence order when the Jacobian is replaced with divided differences, such as Jarratt's scheme [7], Sharma's method [8], Montazeri's method [9], Ostrowski's vectorial extension ([13],[15]), and Sharma-Arora's fifth-order scheme [10]. In all these cases, Jacobian-free versions of the methods reduce to lower orders of convergence.

Nevertheless, Amiri et al. [11] demonstrated that using a specialized divided difference operator of the form $[x, x + G(x); F]$, where $G(x) = (f_1(x)^m, f_2(x)^m, \ldots, f_n(x)^m)^T$, $m \in \mathbb{N}$, as an approximation of the Jacobian matrix, may preserve the convergence order. By selecting an appropriate parameter m, the original fourth-order convergence of these methods can be maintained.

Despite the reduction in performance observed in some Jacobian-free methods, it is important to highlight that there are iterative methods that are successfully modified in their iterative expressions to preserve the order of convergence, even after fully transitioning to Jacobian-free formulations. This is the case of a combination of the Traub-Steffensen family of methods and a second step with divided differences operators, proposed by Behl et al. in [12]

$$\begin{aligned} y^{(k)} &= x^{(k)} - \left[u^{(k)}, x^{(k)}; F\right]^{-1} F(x^{(k)}), \quad k = 0, 1, 2, \ldots \\ x^{(k+1)} &= y^{(k)} - \left[y^{(k)}, x^{(k)}; F\right]^{-1} \left[u^{(k)}, x^{(k)}; F\right] \left[u^{(k)}, y^{(k)}; F\right]^{-1} F(y^{(k)}), \end{aligned} \tag{1}$$

where $u^{(k)} = x^{(k)} + \beta F(x^{(k)}), \beta \in \mathbb{R}$. The iterative schemes have a fourth order of convergence for every β, $\beta \neq 0$, for our purposes we choose $\beta = 1$ and this be called **Traub − Ste**.

Now, we consider several efficient vectorial iterative schemes existing in the literature to transform them in their Jacobian-free versions following the idea of Amiri et al. [11]. In the following sections, we compare these later schemes with our proposed procedures, in terms of efficiency and numerical performance. The first one is the vectorial extension of Ostrowski's scheme (see [13], [14] and [15], for instance),

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

 doi:10.20944/preprints202409.0876.v1

whose Jacobian-free version obtained by substituting the Jacobian matrix by the divided difference operator (with Amiri et al. approach [11], $m = 2$) is

$$\begin{aligned} y^{(k)} &= x^{(k)} - \left[u^{(k)}, x^{(k)}; F\right]^{-1} F(x^{(k)}), \\ x^{(k+1)} &= y^{(k)} - \left[2\left[y^{(k)}, x^{(k)}; F\right] - \left[u^{(k)}, x^{(k)}; F\right]\right]^{-1} F(y^{(k)}), \end{aligned} \tag{3}$$

where $\left[u^{(k)}, x^{(k)}, F\right] \approx F'(x^{(k)})$, and $u^{(k)} = x^{(k)} + \alpha G(x^{(k)})$ (with $m = 2$), $\alpha \in \mathbb{R}$. We denote this method as $\mathbf{Ostro_{01}}$.

Another fourth-order method proposed by Sharma in [16] using Jacobian matrices is

$$\begin{aligned} y^{(k)} &= x^{(k)} - \left[F'(x^{(k)})\right]^{-1} F(x^{(k)}), \\ x^{(k+1)} &= y^{(k)} - \left[3I - 2\left[F'(x^{(k)})\right]^{-1}[y^{(k)}, x^{(k)}; F]\right]\left[F'(x^{(k)})\right]^{-1} F(y^{(k)}), \end{aligned} \tag{4}$$

to which we apply the same Amiri's procedure performed for (2), getting its Jacobian-free partner

$$\begin{aligned} y^{(k)} &= x^{(k)} - \left[u^{(k)}, x^{(k)}; F\right]^{-1} F(x^{(k)}), \\ x^{(k+1)} &= y^{(k)} - \left[3I - 2\left[u^{(k)}, x^{(k)}; F\right]^{-1}[y^{(k)}, x^{(k)}; F]\right]\left[u^{(k)}, x^{(k)}; F\right]^{-1} F(y^{(k)}), \end{aligned} \tag{5}$$

that we denote by $\mathbf{M_{4,3}}$, where $\left[u^{(k)}, x^{(k)}, F\right] \approx F'(x^{(k)})$ and $u^{(k)} = x^{(k)} + \alpha G(x^{(k)})$, $m = 2$, firstly appeared in [11].

We finish with a sixth-order scheme [16], which is obtained by adding a step to the previous method (4),

$$\begin{aligned} y^{(k)} &= x^{(k)} - \left[F'(x^{(k)})\right]^{-1} F(x^{(k)}),\ k = 0, 1, \ldots, \\ z^{(k)} &= y^{(k)} - \left[3I - 2\left[F'(x^{(k)})\right]^{-1}[y^{(k)}, x^{(k)}; F]\right]\left[F'(x^{(k)})\right]^{-1} F(y^{(k)}), \\ x^{(k+1)} &= z^{(k)} - \left[3I - 2\left[F'(x^{(k)})\right]^{-1}[y^{(k)}, x^{(k)}; F]\right]\left[F'(x^{(k)})\right]^{-1} F(z^{(k)}). \end{aligned} \tag{6}$$

Similarly, its Jacobian-free version was constructed in [11] and denoted by $\mathbf{M_{6,3}}$,

$$\begin{aligned} y^{(k)} &= x^{(k)} - \left[u^{(k)}, x^{(k)}; F\right]^{-1} F(x^{(k)}),\ k = 0, 1, \ldots, \\ z^{(k)} &= y^{(k)} - \left[3I - 2\left[u^{(k)}, x^{(k)}; F\right]^{-1}[y^{(k)}, x^{(k)}; F]\right]\left[u^{(k)}, x^{(k)}; F\right]^{-1} F(y^{(k)}), \\ x^{(k+1)} &= z^{(k)} - \left[3I - 2\left[u^{(k)}, x^{(k)}; F\right]^{-1}[y^{(k)}, x^{(k)}; F]\right]\left[u^{(k)}, x^{(k)}; F\right]^{-1} F(z^{(k)}). \end{aligned} \tag{7}$$

where again $\left[u^{(k)}, x^{(k)}, F\right] \approx F'(x^{(k)})$ and $u^{(k)} = x^{(k)} + \alpha G(x^{(k)})$, $m = 2$. It should be noticed that in schemes (3), (5) and (7), we employed a quadratic element-by-element power of $F(x^{(k)})$ in the divided differences. This adjustment was essential for preserving the convergence order of the original method (see [11]). However, in our proposal, the order of convergence of the original schemes is held avoiding the computational cost of this element-by-element power.

Therefore, to avoid the calculation of Jacobian matrices, which can be a bottleneck in terms of computational efficiency especially for large systems, this article presents a two-step fifth-order efficient Jacobian-free iterative method that addresses these challenges by eliminating the need for

 doi:10.20944/preprints202409.0876.v1

direct Jacobian computation. Our approach is grounded in the use of divided differences and scalar accelerators recently developed in some very efficient schemes (using Jacobian matrices), [17,18]. This not only reduces the computational costs, but also accelerates the convergence with simpler iterative expressions. The proposed method's design and theoretical underpinnings are discussed, emphasizing its ability to achieve high-order convergence without the Jacobian calculations typically required.

In the Section 2, we develop a new parametric class of Jacobian-free iterative methods using scalar accelerators and demonstrate its theoretical order of convergence, depending on the values of the parameters involved. Subsequently, in Section 3 we carry out an efficiency analysis in which we compare our proposed method with the Jacobian-free versions of others previously cited in the literature. Finally, Section 4 presents practical results of these iterative methods applied to different nonlinear systems of equations.

2. Construction and Convergence of New Jacobian-Free Iterative Method

In 2023, Singh, Sharma and Kumar [18] proposed a family of iterative methods,

$$\begin{aligned} w^{(k)} &= x^{(k)} - F'(x^{(k)})^{-1}F(x^{(k)}), \quad k = 0, 1, 2, \ldots \\ x^{(k+1)} &= w^{(k)} - \left(p_1 + p_2 \frac{F(w^{(k)})^T F(w^{(k)})}{F(x^{(k)})^T F(x^{(k)})} \right) F'(w^{(k)})^{-1} F(w^{(k)}), \end{aligned} \tag{8}$$

where $\frac{F(y^{(k)})^T F(y^{(k)})}{F(x^{(k)})^T F(x^{(k)})}$ is a scalar accelerator that can be interpreted as $F(y^{(k)})^T F(y^{(k)}) = \|F(y^{(k)})\|^2$, and $F(x^{(k)})^T F(x^{(k)}) = \|F(x^{(k)})\|^2$, respectively. The real parameters p_1 and p_2 make the order of convergence of the method five if p_1=$p_2 = 1$, order four if $p_1 = 1$ and p_2 arbitrary and order two if $p_1 \neq 1$ and p_2 arbitrary. It is known that in many practical applications, computing the Jacobian matrix can be very resource-intensive and time-consuming, therefore, Jacobian-free methods are often preferred.

Making a modification to the scheme (8) by replacing the Jacobian matrices by specific divided differences, we obtain the following family:

$$\begin{aligned} y^{(k)} &= x^{(k)} - \left[u_x^{(k)}, x^{(k)}, F \right]^{-1} F(x^{(k)}), \quad k = 0, 1, 2, \ldots \\ x^{(k+1)} &= y^{(k)} - \left(p_1 + p_2 \frac{F(y^{(k)})^T F(y^{(k)})}{F(x^{(k)})^T F(x^{(k)})} \right) \left[u_y^{(k)}, y^{(k)}, F \right]^{-1} F(y^{(k)}), \end{aligned} \tag{9}$$

where $\left[u_x^{(k)}, x^{(k)}; F \right] \approx F'(x^{(k)})$, $\left[u_y^{(k)}, y^{(k)}; F \right] \approx F'(y^{(k)})$, $u_x^{(k)} = x^{(k)} + \alpha F(x^{(k)})$ and $u_y^{(k)} = y^{(k)} + \alpha F(y^{(k)})$. From now on, we will refer to our modified scheme as $MS(p_1, p_2)$.

The following result shows the error equations arising from method (9) for the possible parameter values, thereby demonstrating that the convergence results of the family (8) hold.

Theorem 1. *Let F be a differentiable enough function $F : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}^n$ defined in the open convex neighbourhood Ω of ξ, solution of $F(x) = 0$. Let us also consider an initial seed $x^{(0)}$ near enough to ξ and let $F'(x)$ be continuous and invertible at ξ. Then, the parametric class of iterative schemes presented in (9) locally converges for all $\alpha \in \mathbb{R}$, with the order of convergence given by:*

(a) Fifth-order convergence if $p_1 = p_2 = 1$, being the corresponding error equation

$$\begin{aligned} x^{(k+1)} - \xi = (M_4 - A_2(M_2M_1 + M_1M_2) - (C_1 + C_2)M_2 - ((M_1^2 + M_1)M_2 + M_2M_1 \\ - P^{-1}M_1^2Q)M_1)e^{(k)^5} + O(e^{(k)^6}). \end{aligned} \tag{10}$$

 doi:10.20944/preprints202409.0876.v1

(b) *Fourth-order convergence if* $p_1 = 1$, $p_2 \neq 1$, *being the corresponding error equation*

$$\begin{aligned} x^{(k+1)} - \xi = {} & (-A_2M_1^2 + C_1M_1 - p_2M_1^3)e^{(k)^4} \\ & + (M_4 - p_1(A_2(M_2M_1 + M_1M_2) - (C_1 + C_2)M_2) - (p_2((M_1^2 + M_1)M_2 \\ & + M_2M_1 - P^{-1}M_1^2Q)M_1)e^{(k)^5} + O(e^{(k)^6}). \end{aligned} \tag{11}$$

(c) *Second-order convergence if* $p_1 \neq 1$, p_2 *arbitrary, being the corresponding error equation*

$$\begin{aligned} x^{(k+1)} - \xi = {} & M_1(1 - p_1)e^{(k)^2} + M_2(1 - p_1)e^{(k)^3} \\ & + (M_3(1 - p_1) - p_1A_2M_1^2 + p_1C_1M_1 - p_2M_1^3)e^{(k)^4} \\ & + (M_4 - p_1(A_2(M_2M_1 + M_1M_2) - (C_1 + C_2)M_2) - (p_2((M_1^2 + M_1)M_2 \\ & + M_2M_1 - P^{-1}M_1^2Q)M_1)e^{(k)^5} + O(e^{(k)^6}), \end{aligned} \tag{12}$$

being $A_j = \frac{1}{j!}\left[F'(\xi)\right]^{-1}F^{(j)}(\xi)$, $j = 2, 3, \ldots$, *and* C_i, M_i, $i = 1, 2, \ldots$, *are combinations of* A_j, *and denoting the error at iteration* k *by* $e^{(k)} = x^{(k)} - \xi$.

Proof. Let $e^{(k)} = x^{(k)} - \xi$ be the error at the k-th iteration and let $\xi \in \mathbb{R}^n$ be a solution of $F(x) = 0$. Then, expanding $F(x^{(k)})$ in the neighborhood of ξ, we have

$$\begin{aligned} F(x^{(k)}) &= F'(\xi)\left[e^{(k)} + A_2e^{(k)^2} + A_3e^{(k)^3} + A_4e^{(k)^4} + A_5e^{(k)^5} + A_6e^{(k)^6} + O\left(e^{(k)^7}\right)\right], \\ F'(x^{(k)}) &= F'(\xi)\left[I + 2A_2e^{(k)} + 3A_3e^{(k)^2} + 4A_4e^{(k)^3} + 5A_5e^{(k)^4} + 6A_6e^{(k)^5} + O\left(e^{(k)^6}\right)\right], \\ F''(x^{(k)}) &= F'(\xi)\left[2A_2 + 6A_3e^{(k)} + 12A_4e^{(k)^2} + 20A_5e^{(k)^3} + 30A_6e^{(k)^4} + O\left(e^{(k)^5}\right)\right], \\ F'''(x^{(k)}) &= F'(\xi)\left[6A_3 + 24A_4e^{(k)} + 60A_5e^{(k)^2} + 120A_6e^{(k)^3} + O\left(e^{(k)^4}\right)\right], \\ F^{(iv)}(x^{(k)}) &= F'(\xi)\left[24A_4 + 120A_5e^{(k)} + 360A_6e^{(k)^2} + O\left(e^{(k)^3}\right)\right], \\ F^{(v)}(x^{(k)}) &= F'(\xi)\left[120A_5 + 720A_6e^{(k)} + O\left(e^{(k)^2}\right)\right], \\ F^{(vi)}(x^{(k)}) &= F'(\xi)\left[720A_6 + O\left(e^{(k)}\right)\right]. \end{aligned} \tag{13}$$

Then, based on the formula of Genochi and Hermite (see [3]) we have

$$\begin{aligned} \left[u_x^{(k)}, x^{(k)}; F\right] = {} & F'(x^{(k)}) + \frac{1}{2!}F''(x^{(k)})(u_x^{(k)} - x^{(k)}) + \frac{1}{3!}F'''(x^{(k)})(u_x^{(k)} - x^{(k)})^2 + \frac{1}{4!}F^{(iv)}(x^{(k)})(u_x^{(k)} - x^{(k)})^3 \\ & + \frac{1}{5!}F^{(v)}(x^{(k)})(u_x^{(k)} - x^{(k)})^4 + \frac{1}{6!}F^{(vi)}(x^{(k)})(u_x^{(k)} - x^{(k)})^5 + O((u_{x^{(k)}} - x^{(k)})^6). \end{aligned} \tag{14}$$

 doi:10.20944/preprints202409.0876.v1

Taking into account that $u_x^{(k)} - x^{(k)} = \alpha F(x^{(k)})$, and performing a series expansion up to fifth-order, we get

$$\begin{aligned}
\alpha^2(F(x^{(k)}))^2 &= \alpha^2(F'(\xi))^2e^{(k)^2} + \alpha[(F'(\xi))^2A_2 + F'(\xi)A_2F'(\xi)]e^{(k)^3} \\
&+ \alpha^2[(F'(\xi))^2A_3 + F'(\xi)A_2F'(\xi)A_2 + F'(\xi)A_3F'(\xi)]e^{(k)^4} \\
&+ \alpha^2[(F'(\xi))^2A_4 + F'(\xi)A_2F'(\xi)A_3 + F'(\xi)A_3F'(\xi)A_2 + F'(\xi)A_4F'(\xi)]e^{(k)^5} + O(e^{(k)^6}),
\end{aligned}$$

$$\begin{aligned}
\alpha^3(F(x^{(k)}))^3 &= \alpha^3(F'(\xi))^3e^{(k)^3} + \alpha^3[(F'(\xi))^3A_2 + (F'(\xi))^2A_2F'(\xi) + F'(\xi)A_2(F'(\xi))^2]e^{(k)^4} \\
&+ \alpha^3[(F'(\xi))^3A_3 + (F'(\xi))^2A_2F'(\xi)A_2 + F'(\xi)A_2(F'(\xi))^2A_2 + (F'(\xi))^2A_3F'(\xi) \\
&+ F'(\xi)A_2F'(\xi)A_2F'(\xi) + F'(\xi)A_3(F'(\xi))^2]e^{(k)^5} + O(e^{(k)^6}),
\end{aligned} \tag{15}$$

$$\begin{aligned}
\alpha^4(F(x^{(k)}))^4 &= \alpha^4(F'(\xi))^4e^{(k)^4} + \alpha^4[(F'(\xi))^4A_2 + (F'(\xi))^3A_2F'(\xi) \\
&+ (F'(\xi))^2A_2(F'(\xi))^2 + F'(\xi)A_2((F'(\xi))^3]e^{(k)^5} + O(e^{k^6}),
\end{aligned}$$

$$\alpha^5(F(x^{(k)}))^5 = \alpha(F'(\xi))^5e^{(k)^5} + O(e^{(k)^6}).$$

By combining formulas (13), (15) in the Taylor series expansion (14) we obtain:

$$\left[u_x^{(k)}, x^{(k)}; F\right] = F'(\xi)\left[I + B_1e^k + B_2e^{(k)^2} + B_3e^{(k)^3} + B_4e^{(k)^4} + B_5e^{(k)^5} + O(e^{(k)^6})\right], \tag{16}$$

where

$$\begin{aligned}
B_1 &= 2A_2 + \alpha A_2F'(\xi), \\
B_2 &= 3A_3 + \alpha A_2F'(\xi)A_2 + 3\alpha A_3F'(\xi) + \alpha^2A_3(F'(\xi))^2, \\
B_3 &= 4A_4 + \alpha A_2F'(\xi)A_3 + 3\alpha A_3F'(\xi)A_2 + 6\alpha A_4F'(\xi) + \alpha^2A_3(F'(\xi))^2A_2 + \alpha^2A_3F'(\xi)A_2F'(\xi) \\
&+ \alpha^2 4A_4(F'(\xi))^2 + \alpha^3A_4(F'(\xi))^3, \\
B_4 &= 5A_5 + \alpha A_2F'(\xi)A_4 + 3\alpha A_3F'(\xi)A_3 + 6\alpha A_4F'(\xi)A_2 + 10\alpha A_5F'(\xi) + \alpha^2A_3(F'(\xi))^2 \\
&+ \alpha^2A_3F'(\xi)A_2F'(\xi)A_2 + \alpha^2A_3F'(\xi)A_3F'(\xi) + 4\alpha^2A_4(F'(\xi))^2A_2 + 4\alpha^2A_4F'(\xi)A_2F'(\xi) \\
&+ 10\alpha^2A_5(F'(\xi))^2 + \alpha^3A_4(F'(\xi))^3A_2 + \alpha^3A_4(F'(\xi))^2A_2F'(\xi) + \alpha^3A_4F'(\xi)A_2(F'(\xi))^2 \\
&+ 5\alpha^3A_5(F'(\xi))^3 + \alpha^4A_5(F'(\xi))^4,
\end{aligned}$$

and

$$\begin{aligned}
B_5 &= 6A_6 + \alpha A_2F'(\xi)A_5 + 3\alpha A_3F'(\xi)A_4 + 6\alpha A_4F'(\xi)A_3 + 10\alpha A_5F'(\xi)A_2 + 15\alpha A_6F'(\xi) \\
&+ \alpha^2A_3(F'(\xi))^2A_4 + \alpha^2A_3F'(\xi)A_2F'(\xi)A_3 + \alpha^2A_3F'(\xi)A_3F'(\xi)A_2 + \alpha^2A_3F'(\xi)A_4F'(\xi) \\
&+ 4\alpha^2A_4(F'(\xi))^2A_3 + 4\alpha^2A_4F'(\xi)A_2F'(\xi)A_2 + 4\alpha^2A_4F'(\xi)A_3F'(\xi) + 10\alpha^2A_5(F'(\xi))^2A_2 \\
&+ 10\alpha^2A_5F'(\xi)A_2F'(\xi) + 20\alpha^2A_6(F'(\xi))^2 + \alpha^3A_4(F'(\xi))^3A_3 + \alpha^3A_4(F'(\xi))^2A_2F'(\xi))A_2 \\
&+ \alpha^3A_4F'(\xi)A_2(F'(\xi))^2A_2 + \alpha^3A_4(F'(\xi))^2A_3F'(\xi) + \alpha^3A_4F'(\xi)A_2F'(\xi)A_2F'(\xi) \\
&+ \alpha^3A_4F'(\xi)A_3(F'(\xi))^2 + 5\alpha^3A_5(F'(\xi))^3A_2 + 5\alpha^3A_5(F'(\xi))^2A_2F'(\xi) + 5\alpha^3A_5F'(\xi)A_2(F'(\xi))^2 \\
&+ 15\alpha^3A_6(F'(\xi))^3 + \alpha^4A_5(F'(\xi))^4A_2 + \alpha^4A_5(F'(\xi))^3A_2F'(\xi) + \alpha^4A_5(F'(\xi))^2A_2(F'(\xi))^2 \\
&+ \alpha^4A_5F'(\xi)A_2(F'(\xi))^3 + 6\alpha^4A_6(F'(\xi))^4 + \alpha^5A_6(F'(\xi))^5.
\end{aligned}$$

 doi:10.20944/preprints202409.0876.v1

Next, we expand the inverse of the divided difference operator $\left[u_x^{(k)}, x^{(k)}; F\right]$, forcing it to satisfy $\left[u_x^{(k)}, x^{(k)}; F\right]^{-1}\left[u_x^{(k)}, x^{(k)}; F\right] = I$,

$$\left[u_x^{(k)}, x^{(k)}; F\right]^{-1} = \left[I + X_2 e^k + X_3 e^{(k)^2} + X_4 e^{(k)^3} + X_5 e^{(k)^4} + O(e^{(k)^5})\right]\left(F'(\xi)\right)^{-1}, \tag{17}$$

where

$$\begin{aligned} X_2 &= -B_1, \\ X_3 &= B_1^2 - B_2, \\ X_4 &= B_1 B_2 + B_2 B_1 - B_1^3 - B_3, \\ X_5 &= B_1 B_3 + B_3 B_1 + B_1^4 - B_4 - B_1^2 + B_2^2 - B_1 B_2 B_1 - B_2 B_1^2. \end{aligned} \tag{18}$$

By using the Taylor expansions of $F(x^{(k)})$ defined in (13) and $\left[u_x^{(k)}, x^{(k)}, F\right]^{-1}$ obtained in (17), we get the error equation for the first step:

$$y^{(k)} - \xi = M_1 e^{(k)^2} + M_2 e^{(k)^3} + M_3 e^{(k)^4} + M_4 e^{(k)^5} + O(e^{(k)^6}), \tag{19}$$

where

$$\begin{aligned} M_1 &= -(X_2 + A_2), \\ M_2 &= -(A_3 + X_2 A_2 + X_3), \\ M_3 &= -(A_4 + X_2 A_3 + X_3 A_2 + X_4), \\ M_4 &= -(A_5 + X_2 A_4 + X_3 A_3 + X_4 A_2 + X_5). \end{aligned} \tag{20}$$

Now, we find the error equation for the second step,

$$\begin{aligned} F(y^{(k)}) &= F'(\xi)\left[e_y^{(k)} + A_2 e_y^{(k)^2} + O(e_y^{(k)^3})\right], \\ F'(y^{(k)}) &= F'(\xi)\left[I + 2A_2 e_y^{(k)} + 3A_3 e_y^{(k)^2} + O(e_y^{(k)^3})\right], \\ F''(y^{(k)}) &= F'(\xi)\left[2A_2 + 6A_3 e_y^{(k)} + 12A_4 e_y^{(k)^2} + O(e_y^{(k)^3})\right], \\ F'''(y^{(k)}) &= F'(\xi)\left[6A_3 + O(e_y^{(k)})\right], \end{aligned} \tag{21}$$

from which arises

$$\begin{aligned} F(y^{(k)}) &= F'(\xi)\left[M_1 e^{(k)^2} + M_2 e^{(k)^3} + (M_3 + A_2 M_1^2) e^{(k)^4} + A_2(M_2 M_1 + M_1 M_2) e^{(k)^5} + O(e^{(k)^6})\right], \\ F'(y^{(k)}) &= F'(\xi)\Big[I + 2A_2 M_1 e^{(k)^2} + 2A_2 M_2 e^{(k)^3} + (2A_2 M_3 + 3A_3 M_1^2) e^{(k)^4} \\ &\quad + (2A_2 M_4 + 3A_3(M_1 M_2 + M_2 M_1)) e^{(k)^5} + O(e^{(k)^6})\Big], \\ F''(y^{(k)}) &= F'(\xi)\left[2A_2 + 6A_3 M_1 e^{(k)^2} + 6A_3 M_3 e^{(k)^3} + O(e^{(k)^4})\right], \\ F'''(y^{(k)}) &= F'(\xi)\left[6A_3 + O(e^{(k)})\right]. \end{aligned} \tag{22}$$

Then, following the process seen in (14), the expansion of the second difference operator is given by

$$\begin{aligned} \left[u_y^{(k)}, y^{(k)}; F\right] = F'(\xi)\Big[&I + (2A_2 M_1 + \alpha A_2 F'(\xi) M_1) e^{(k)^2} + (2A_2 M_2 + \alpha A_2 F'(\xi) M_2) e^{(k)^3} \\ &+ (2A_3 M_3 + 3\alpha A_3 M_1^2 + \alpha A_2 F'(\xi)(M_3 + A_2 M_2^2) + 3\alpha A_3 M_1 F'(\xi) M_1 \\ &+ \alpha A_3 F'(\xi) M_1 F'(\xi) M_1) e^{(k)^4} + (2A_2 M_4 + 3A_3(M_1 M_2 + M_2 M_1) \\ &+ \alpha A_2 F'(\xi) A_2(M_2 M_1 + M_1 M_2) + 3\alpha A_3 M_1 F'(\xi)(M_2 + M_1) \\ &+ \alpha A_3 F'(\xi)(M_1 F'(\xi) M_2 + M_2 F'(\xi) M_1)) e^{(k)^5} + O(e^{(k)^6})\Big], \end{aligned} \tag{23}$$

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 September 2024 doi:10.20944/preprints202409.0876.v1

that can be expressed as

$$\left[u_y^{(k)}, y^{(k)}; F\right] = F'(\xi)\left[I + C_1 e^{(k)^2} + C_2 e^{(k)^3} + C_3 e^{(k)^4} + C_4 e^{(k)^5} + O(e^{(k)^6})\right], \tag{24}$$

being

$$\begin{aligned}
C_1 &= 2A_2M_1 + \alpha A_2 F'(\xi) M_1,\\
C_2 &= 2A_2M_2 + \alpha A_2 F'(\xi) M_2,\\
C_3 &= 2A_3M_3 + 3\alpha A_3 M_1^2 + \alpha A_2 F'(\xi)(M_3 + A_2 M_2^2) + 3\alpha A_3 M_1 F'(\xi) M_1 + \alpha A_3 F'(\xi) M_1 F'(\xi) M_1,\\
C_4 &= 2A_2M_4 + 3A_3(M_1M_2 + M_2M_1) + \alpha A_2 F'(\xi) A_2(M_2M_1 + M_1M_2) + 3\alpha A_3 M_1 F'(\xi)(M_2 + M_1)\\
&\quad + \alpha A_3 F'(\xi)(M_1 F'(\xi) M_2 + M_2 F'(\xi) M_1).
\end{aligned} \tag{25}$$

Again, we get in a similar way as in (17),

$$\left[u_y^{(k)}, y^{(k)}; F\right]^{-1} = -C_1 e^{(k)^2} - C_2 e^{(k)^3} + O\left(e^{(k)^4}\right). \tag{26}$$

Now, we proceed to calculate the expansion of $\left[u_y^{(k)}, y^{(k)}; F\right]^{-1} F\left(y^{(k)}\right)$, obtaining

$$\begin{aligned}
\left[u_y^{(k)}, y^{(k)}; F\right]^{-1} F\left(y^k\right) =& M_1 e^{(k)^2} + M_2 e^{(k)^3} + (M_3 - A_2 M_1^2 - C_1 M_1) e^{(k)^4}\\
&+ (A_2(M_2M_1 + M_1M_2) - (C_1 + C_2) M_2) e^{(k)^5} + O(e^{(k)^6}).
\end{aligned} \tag{27}$$

According to *Theorem 1* proven by Singh, Sharma and Kurmar in [18], we have

$$\frac{F(y^{(k)})^T F(y^{(k)})}{F(x^{(k)})^T F(x^{(k)})} = \frac{P e_y^{(k)^2} + O(e_y^{(k)^3})}{P e^{(k)^2} + Q e^{(k)^3} + O(e^{(k)^4})}, \tag{28}$$

where

$$P = \sum_{i=1}^{n} m_i P_i, \text{ with } P_i = R_i^T \times R_i, Q = \sum_{i=1}^{n} m_i Q_i, \text{ with } Q_i = R_i^T H_i + H_i^T R_i,$$

and

$$R_i = \left(\frac{\partial f_i}{\partial x_1}, \frac{\partial f_i}{\partial x_2}, \ldots, \frac{\partial f_i}{\partial x_n}\right), \quad H_i = \frac{1}{2}\left(\frac{\partial^2 f_i}{\partial x_j \partial x_r}\right)_{n\times n}.$$

Using (19), we get

$$e_y^{(k)^2} = M_1^2 e^{(k)^4} + (M_1M_2 + M_2M_1) e^{(k)^5} + O(e^{(k)^6}). \tag{29}$$

After substituting (29) in (28) when performing the quotient, we obtain:

$$\frac{F(y^{(k)})^T F(y^{(k)})}{F(x^{(k)})^T F(x^{(k)})} = M_1^2 e^{(k)^2} + \left(M_1M_2 + M_2M_1 - P^{-1} M_1^2 Q\right) e^{(k)^3} + O(e^{(k)^4}). \tag{30}$$

Finally, fitting (19), (27) and the last result obtained in (30), we have

$$\begin{aligned}
x^{(k+1)} - \xi =& M_1(1 - p_1) e^{(k)^2} + M_2(1 - p_1) e^{(k)^3}\\
&+ (M_3(1 - p_1) - p_1 A_2 M_1^2 + p_1 C_1 M_1 - p_2 M_1^3) e^{(k)^4}\\
&+ (M_4 - p_1(A_2(M_2M_1 + M_1M_2) - (C_1 + C_2) M_2) - (p_2((M_1^2 + M_1) M_2\\
&+ M_2M_1 - P^{-1} M_1^2 Q) M_1) e^{(k)^5} + O(e^{(k)^6}).
\end{aligned} \tag{31}$$

 doi:10.20944/preprints202409.0876.v1

In this last result, it is easy to observe that when p_1 is different from one ($p_1 \neq 1$) the iterative method has order of convergence equal to two, since the term $e^{(k)^2}$ would not cancel out. However, when $p_1 = p_2 = 1$ both terms with $e^{(k)^2}$ and $e^{(k)^3}$ cancel out while the term $e^{(k)^4}$ is as follows

$$-A_2M_1^2 + C_1M_1 - M_1^3.$$

Let us remember that $C_1 = 2A_2M_1 + \alpha A_2F'(\xi)M_1$, so that

$$A_2M_1^2 + \alpha A_2F'(\xi)M_1^2 - M_1^3, \tag{32}$$

but since $M_1 = -(X_2 + A_2)$, then $X_2 = -B_1$, so $M_1 = -(-B_1 + A_2)$. Also, $B_1 = 2A_2 + \alpha A_2F'(\xi)$. Given that $M_1 = A_2 + \alpha A_2F'(\xi)$, replacing M_1 in equation (32), we get

$$(A_2 + \alpha A_2F'(\xi))M_1^2 - M_1^3 \quad \rightarrow \quad M_1^3 - M_1^3 = 0,$$

resulting in the error equation

$$\begin{aligned} e^{(k+1)} - \xi = (M_4 - A_2(M_2M_1 + M_1M_2) - (C_1 + C_2)M_2 - ((M_1^2 + M_1)M_2 + M_2M_1 \\ - P^{-1}M_1^2Q)M_1)e^{(k)^5} + O(e^{(k)^6}). \end{aligned} \tag{33}$$

From the above it is clear to see that if $p_1 = 1$ but $p_2 \neq 1$ only order four is reached. □

3. Efficiency Analysis

We have demonstrated the order of convergence of the proposed class of the iterative method for the different values of the parameters p_1, p_2. In this section, we perform a computational effort study considering the effort of solving the involved linear systems per iteration and the other computational cost (functional evaluations, amount of product/quotients,...), not only for the proposed class but also for some Jacobian-free schemes presented in the introductory section.

In order to get this aim, it is known that the needed operations (products/quotients) of solving a $n \times n$ linear system is

$$\frac{1}{3}n^3 + n^2 - \frac{1}{3}n.$$

However, if other linear systems with the same coefficient matrix are solved, then the cost upgrades only in n^2 operations each; for each divided difference we calculate n^2 quotients; for each functional evaluation of F at different points, a cost of n real evaluations; for each evaluation of a divided differences, $n^2 - n$ scalar evaluations; Indeed, a matrix–vector product needs n^2 product/quotients. Based on the above, the computational cost for each method appears in Table 1. From family (9), which we call $MS(p_1, p_2)$, we consider the fifth-order member $p_1 = p_2 = 1$ and its fourth-order partner $p_1 = 1, p_2 = -1$. They have the same computational cost, which will be reflected, along with the others, in Table 1.

Table 1. Computational effort of new and comparison schemes

Method	Complexity C
$MS(p_1, p_2)$	$\frac{2}{3}n^3 + 6n^2 - \frac{2}{3}n$
$Traub - Ste$	$n^3 + 10n^2 - 2n$
$Ostro_{01}$	$\frac{2}{3}n^3 + 6n^2 + \frac{1}{3}n$
$M_{4,3}$	$\frac{1}{3}n^3 + 8n^2 + \frac{2}{3}n$
$M_{6,3}$	$\frac{1}{3}n^3 + 11n^2 + \frac{5}{3}n$

The results presented in Table 1 show that the method with the highest computational cost is $Traub - Ste$, while those with intermediate costs are $Ostro_{01}$ and $MS(p_1, p_2)$, with the latter being

 doi:10.20944/preprints202409.0876.v1

slightly better. The ones that offer the lowest cost are $M_{4,3}$ and $M_{6,3}$, although the latter has sixth-order convergence, which is a factor to consider when obtaining the efficiency index.

In order to show more clearly how computational cost influences the efficiency index $I = p^{\frac{1}{C}}$, where p is the convergence order of the corresponding scheme (see [20]), we present Figure 1 and Figure 2 for different sizes of the nonlinear system to be solved.

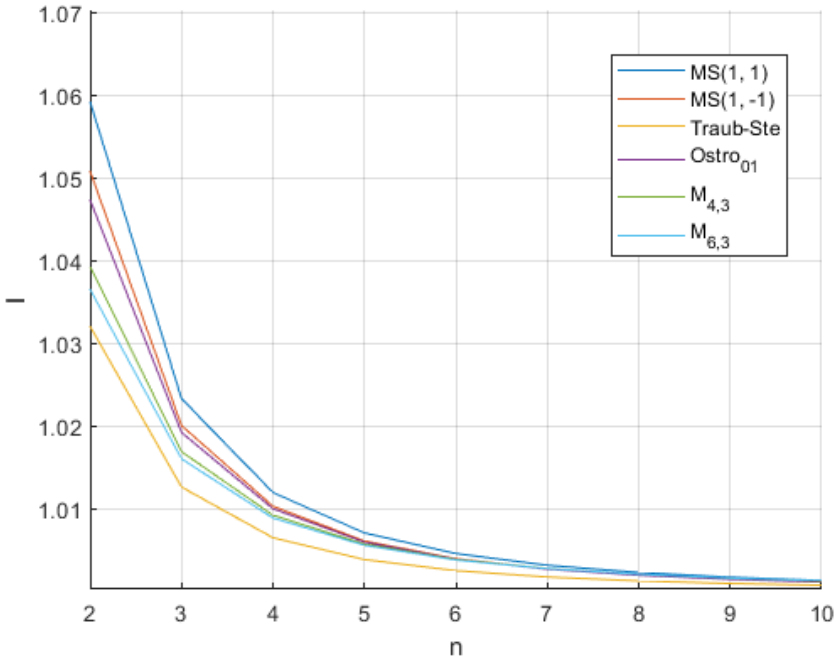

Figure 1. I indices for $MS(p_1, p_2)$ and comparison methods

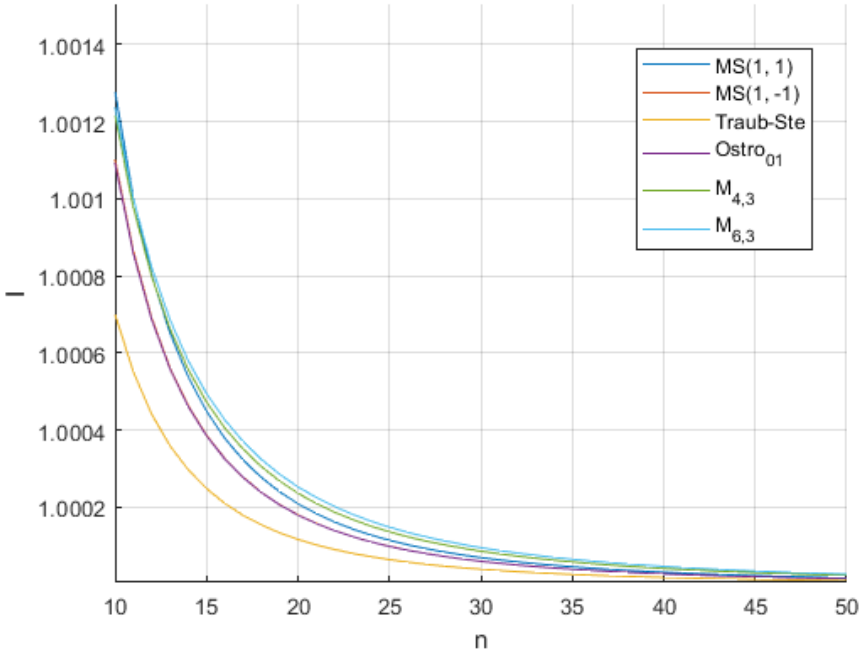

Figure 2. I indices for $MS(p_1, p_2)$ and comparison schemes

In Figure 1, we observe that for systems with dimensions $n = 2, 3, \ldots, 10$, the proposed class of vectorial iterative methods (9) shows a better computational efficiency than the other schemes, for both

 doi:10.20944/preprints202409.0876.v1

members of the family. On the other hand, as the system grows to sizes $n = 10, 11, \ldots, 50$ it becomes apparent that methods $M_{4,3}$ and $M_{6,3}$ yield better results. Let us notice that for sizes of $n < 11$, our method has better efficiency than $M_{6,3}$, and for sizes $n < 12$, it has better efficiency compared to $M_{4,3}$ (see Figure 2).

In the next section, we check the theoretical convergence order of our proposed method and assess the efficiency of different schemes on nonlinear systems of various sizes.

4. Numerical Results

In this section, we test numerically that the theoretical order holds for practical purposes in the proposed Jacobian-free class (9). Moreover, we compare it with the methods appearing in the efficiency section to show their accuracy and computational performance.

Below we show some nonlinear problems, including one whose related nonlinear function is not differentiable, in order to test the applicability of the methods on different kind of problems.

- We consider $F^1(x) = \left(f_1^1(x), f_2^1(x), \ldots, f_{25}^1(x)\right)^T$, where

$$\begin{aligned} f_i^1(x) &= x_i^2 x_{i+1} - 1, \quad i = 1, 2, \ldots, 24, \\ f_{25}^1(x) &= x_{25}^2 x_1 - 1. \end{aligned}$$

  whose solution is $\bar{\xi} = (1, 1, 1, \ldots, 1)$.
- We also have $F^2(x) = \left(f_1^2(x), f_2^2(x), \ldots, f_8^2(x)\right)^T$, where

$$f_i^2(x) = x_i - \cos\left(2x_i - \sum_{k=1}^{8} x_k\right), \quad i = 1, 2, \ldots, 8,$$

  being $\bar{\xi} \approx (0.5149,\ 0.5149,\ \ldots, 0.5149)$ its solution.
- We also consider $F^3(x) = \left(f_1^3(x), f_2^3(x), \ldots, f_5^3(x)\right)^T$, where

$$f_i^3(x) = \sum_{k=1}^{5} (x_k) - x_i - e^{(-x_i)}, \qquad i = 1, 2, 3, 4, 5,$$

  where the solution is $\bar{\xi} \approx (0.20389,\ 0.20389,\ 0.20389,\ \ldots, 0.20389)$.
- We test also $F^4(x) = \left(f_1^4(x), f_2^4(x), \ldots, f_5^4(x)\right)^T$, where

$$f_i^4(x) = \sum_{k=1}^{5} (x_k) - x_i - e^{(-x_i)} x_i, \qquad i = 1, 2, 3, 4, 5,$$

  whose solution is $\bar{\xi} = (0,\ 0,\ 0,\ \ldots\ , 0)$.
- $F^5(x) = \left(f_1^5(x), f_2^5(x), \ldots, f_{10}^5(x)\right)^T$, where

$$f_i^5(x) = x_i + 1 - 2\log\left(1 - x_i + \sum_{k=1}^{10} (x_k)\right), \qquad i = 1, 2, 3, \ldots, 10,$$

  being it solution $\bar{\xi} \approx (7.4370, 7.4370, \ldots, 7.4370)$.
- $F^6(x) = \left(f_1^6(x), f_2^6(x), \ldots, f_5^6(x)\right)^T$, where

$$f_i^6(x) = x_i + 1.5\sin\left(\sum_{k=1}^{5} (x_k) - x_i\right), \qquad i = 1, 2, 3, 4, 5,$$

  and there exist two solutions, $\bar{\xi}_1 \approx (-0.3004, -0.3004, \ldots, -0.3004)$, and $\bar{\xi}_2 \approx (0.4579, 0.4579, \ldots, 0.4579)$.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 September 2024 doi:10.20944/preprints202409.0876.v1

- $F^7(x) = \left(f_1^7(x), f_2^7(x)\right)^T$, where

$$F_i^7(x) = \arctan(x_i) + 1 - 2\left[\left(\sum_{k=1}^{2} x_k^2\right) - x_i^2\right] = 0, \qquad i = 1, 2,$$

  being $\bar{\xi} \approx (0.936, 0.936)$.
- We also test $F^8(x) = \left(f_1^8(x), f_2^8(x)\right)^T$, where

$$f_1^8(x) = log(|x_1|) + |x_2|,$$
$$f_2^8(x) = e^{(x_1)} + x_2 - 1,$$

  with solutions $\bar{\xi}_1 \approx (-0.6275, 0.4661)$ and $\bar{\xi}_2 \approx (0.5122, -0.669)$.

Numerical results have been obtained with Matlab2022b version, using 8000 digits in variable precision arithmetics, a processor AMD $A12 - 9720P$ RADEON R7, 12 COMPUTE CORES 4C, +8G-Ram, 2.70 GHz. These results are shown in Tables 2–13, including the following information, where the appearing norms are Euclidean:

- k: amount of iterations needed ("-" appears when the scheme does not converge or it needs more iterations than the maximum allowed).
- $\bar{\xi}$: obtained solution.
- Cpu-time: average time in seconds required by the iterative method to reach the solution of the problem when executed ten times.
- ρ: approximated computational order of convergence, ACOC, firstly appeaaring in [19]

$$\rho = \frac{\ln \frac{\left\|x^{(k+1)} - x^{(k)}\right\|}{\left\|x^{(k)} - x^{(k-1)}\right\|}}{\ln \frac{\left\|x^{(k)} - x^{(k-1)}\right\|}{\left\|x^{(k-1)} - x^{(k-2)}\right\|}}, \quad k = 2, 3, \ldots,$$

  (if ρ is not stable, then "-" appears in the table).
- $\epsilon_{aprox} = \left\|x^{(k+1)} - x^{(k)}\right\|$.
- $\epsilon_f = \left\|F\left(x^{(k+1)}\right)\right\|$. If ϵ_f or ϵ_{aprox} are very far from zero or we get infinity or NAN, then "-" appears in the table.

Regarding the stopping criterium, the iterative process ends when one of the following conditions is fulfilled:

(i) $\left\|F\left(x^{(k+1)}\right)\right\| < 10^{-100}$,
(ii) $\left\|x^{(k+1)} - x^{(k)}\right\| < 10^{-100}$,
(iii) 50 iterations are reached.

Table 2. Results for function F^1, using as seed $x^{(0)} = (1.5, 1.5, 1.5, \ldots, 1.5)$

Iterative method	k	ρ	ϵ_{aprox}	ϵ_f	$\bar{\xi}$	Cpu-time
$MS(1,-1)$	7	4.00	1.108e-51	4.624e-204	$\bar{\xi}_1$	191.7242
$MS(1,1)$	5	4.97	4.904e-24	3.995e-117	$\bar{\xi}_1$	143.1027
$Traub - Ste$	5	4.00	1.241e-35	1.517e-140	$\bar{\xi}_1$	227.8747
$Ostro_{01}$	-	-	-	-	-	-
$M_{4,3}$	6	4.00	7.338e-40	3.015e-158	$\bar{\xi}_1$	163.9808
$M_{6,3}$	5	5.96	1.336e-47	7.876e-284	$\bar{\xi}_1$	142.6446

In Table 2, $Ostro_{01}$ method reached the maximum number of iterations without converging to the solution, while the most notable scheme is $M_{6,3}$ in almost all aspects such as errors, iterations and

 doi:10.20944/preprints202409.0876.v1

computational time. On the other hand, although $MS(1,1)$ exhibits a lower computational time than $M_{4,3}$, the latter has an additional iteration, a relatively similar time, and better errors, proving that it might even be superior in terms of efficiency.

Table 3. Numerical results for F^2, $x^{(0)} = (1,\ 1,\ 1,\ \ldots,\ 1)$

Iterative method	k	ρ	ϵ_{aprox}	ϵ_f	$\bar{\xi}$	Cpu-time
$MS(1,-1)$	8	3.99	2.794e-29	4.092e-115	$\bar{\xi}_1$	73.3126
$MS(1,1)$	8	5.00	7.534e-74	2.162e-366	$\bar{\xi}_1$	72.6143
$Traub-Ste$	16	4.00	9.389e-68	3.459e-269	$\bar{\xi}_1$	317.9304
$Ostro_{01}$	15	4.00	6.341e-70	1.641e-278	$\bar{\xi}_1$	139.3808
$M_{4,3}$	7	4.00	8.510e-87	3.949e-346	$\bar{\xi}_1$	76.9114
$M_{6,3}$	6	6.02	4.223e-47	8.063e-281	$\bar{\xi}_1$	79.7898

In Table 3, we observe that our proposed schemes yield the best computational times, with $MS(1,1)$ method standing out for having the smallest error norm among all schemes. The $M_{4,3}$ method, while showing good overall performance in terms of errors, is more computationally expensive as it takes one iteration longer to reach the solution to the system and slightly more time compared to $MS(p1,p2)$ methods, indicating that the program takes more time to generate each iteration.

Table 4. Numerical results for F^3, $x^{(0)} = (0.5,\ 0.5,\ 0.5,\ 0.5,\ 0.5)$

Iterative method	k	ρ	ϵ_{aprox}	ϵ_f	$\bar{\xi}$	Cpu-time
$MS(1,-1)$	4	4.00	1.305e-55	2.980e-221	$\bar{\xi}_1$	12.8166
$MS(1,1)$	4	5.00	4.997e-101	1.143e-503	$\bar{\xi}_1$	12.6105
$Traub-Ste$	4	4.00	9.461e-68	1.372e-270	$\bar{\xi}_1$	18.9457
$Ostro_{01}$	4	4.00	1.420e-47	2.882e-189	$\bar{\xi}_1$	12.4768
$M_{4,3}$	4	4.00	3.440e-44	1.007e-175	$\bar{\xi}_1$	12.8707
$M_{6,3}$	3	6.07	1.482e-21	3.012e-127	$\bar{\xi}_1$	10.5536

Table 5. Numerical results for F^4, $x^{(0)} = (0.5,\ 0.5,\ 0.5,\ 0.5,\ 0.5)$

Iterative method	k	ρ	ϵ_{aprox}	ϵ_f	$\bar{\xi}$	Cpu-time
$MS(1,-1)$	4	3.99	7.838e-31	4.801e-121	$\bar{\xi}_1$	14.6365
$MS(1,1)$	4	5.00	2.864e-52	2.514e-258	$\bar{\xi}_1$	14.6532
$Traub-Ste$	4	4.00	8.656e-34	3.124e-133	$\bar{\xi}_1$	21.7138
$Ostro_{01}$	4	4.00	2.268e-35	6.443e-140	$\bar{\xi}_1$	14.0140
$M_{4,3}$	4	4.00	2.594e-47	9.227e-188	$\bar{\xi}_1$	14.8093
$M_{6,3}$	3	5.32	6.593e-26	7.254e-153	$\bar{\xi}_1$	12.0667

The results in Tables 4-5 are very balanced, with the $MS(1,1)$ method showing the best errors while the shortest computational time is achieved by the $M_{6,3}$ method. However, this last one takes more effort to generate an iteration because, despite producing one less iteration than the other iterative methods, their execution times are very similar.

Table 6. Numerical results for F^5, $x^{(0)} = (7,\ 7,\ 7,\ldots,7)$

Iterative method	k	ρ	ϵ_{aprox}	ϵ_f	$\bar{\xi}$	Cpu-time
$MS(1,-1)$	3	4.03	1.343e-24	1.076e-101	$\bar{\xi}_1$	58.6649
$MS(1,1)$	3	5.03	5.715e-36	1.098e-183	$\bar{\xi}_1$	58.6010
$Traub-Ste$	4	4.00	2.252e-97	1.397e-392	$\bar{\xi}_1$	138.3238
$Ostro_{01}$	4	4.00	1.317e-99	1.705e-401	$\bar{\xi}_1$	76.6871
$M_{4,3}$	3	4.00	1.490e-24	2.175e-101	$\bar{\xi}_1$	64.3101
$M_{6,3}$	3	6.01	1.565e-53	4.782e-326	$\bar{\xi}_1$	69.4223

 doi:10.20944/preprints202409.0876.v1

For Table 6, we note that the best computational times are obtained by the methods $MS(p1, p2)$. On the other hand, the best errors are obtained by $Ostro_{01}$ scheme, which has a competitive computational time considering it requires four iterations, similar to $Traub - Ste$. The latter has shown to be the one that requires the most time to converge.

Table 7. Numerical results for F^6 $x^{(0)} = (0.75, 0.75, 0.75, 0.75, 0.75)$

Iterative method	k	ρ	ϵ_{aprox}	ϵ_f	$\bar{\xi}$	Cpu-time
$MS(1,-1)$	6	4.00	5.425e-88	5.270e-349	$\bar{\xi}_1$	20.2517
$MS(1,1)$	5	4.97	5.922e-39	1.965e-190	$\bar{\xi}_1$	16.9191
$Traub - Ste$	20	4.00	9.536e-99	4.618e-391	$\bar{\xi}_2$	154.5691
$Ostro_{01}$	5	4.00	1.050e-33	2.164e-132	$\bar{\xi}_1$	16.5869
$M_{4,3}$	10	4.01	3.127e-35	1.500e-138	$\bar{\xi}_1$	51.2143
$M_{6,3}$	17	5.90	6.480e-23	2.076e-132	$\bar{\xi}_2$	105.4212

In Table 7, the best errors were obtained by $MS(1,-1)$ and $Traub - Ste$, which converged to different solutions, while in terms of performance, $MS(1,1)$ and $Ostro_{01}$ appeared to be better.

Table 8. Numerical results for F^7, $x^{(0)} = (0.25, 0.25)$

Iterative method	k	ρ	ϵ_{aprox}	ϵ_f	$\bar{\xi}$	Cpu-time
$MS(1,-1)$	5	4.00	5.203e-41	6.322e-161	$\bar{\xi}_1$	3.6588
$MS(1,1)$	4	5.00	5.282e-73	6.419e-362	$\bar{\xi}_1$	3.0030
$Traub - Ste$	6	4.00	6.615e-81	7.658e-321	$\bar{\xi}_1$	4.6891
$Ostro_{01}$	-	-	-	-	-	-
$M_{4,3}$	6	4.00	5.613e-98	3.787e-389	$\bar{\xi}_1$	3.7011
$M_{6,3}$	7	5.92	2.032e-32	5.723e-190	$\bar{\xi}_1$	4.4699

In Table 8, $Ostro_{01}$ method could not converge to the solution because it reached the maximum number of iterations, whereas the $MS(p1, p2)$ and $M_{4,3}$ methods showed better overall behavior.

Table 9. Numerical results for F^8, $x^{(0)} = (0.25, 0.25)$

Iterative method	k	ρ	ϵ_{aprox}	ϵ_f	$\bar{\xi}$	Cpu-time
$MS(1,-1)$	4	4.00	1.139e-54	9.420e-217	$\bar{\xi}_1$	2.9146
$MS(1,1)$	4	4.20	1.415e-72	2.272e-301	$\bar{\xi}_1$	2.8907
$Traub - Ste$	4	4.11	4.417e-48	1.391e-191	$\bar{\xi}_1$	3.1954
$Ostro_{01}$	6	4.00	2.520e-78	1.241e-310	$\bar{\xi}_1$	3.4855
$M_{4,3}$	7	4.02	1.401e-72	3.093e-288	$\bar{\xi}_1$	4.0501
$M_{6,3}$	6	6.27	4.634e-58	7.902e-346	$\bar{\xi}_1$	3.6475

In the Table 9, $MS(1,1)$ method stands out with the lowest computational time, being more efficient than the others, despite requiring a similar number of iterations and showing a convergence order of 4.20, close to the other methods. The $Traub - Ste$ method is less efficient, with a computational time of 3.1954 but with a larger errors. $Ostro_{01}$, is competitive only falling slightly behind in computational time. The methods $M_{4,3}$ and $M_{6,3}$ are the most expensive in terms of computational time, requiring 7 and 6 iterations respectively, with the highest time recorded for $M_{4,3}$. Despite their higher orders of convergence, both methods show lower overall efficiency compared to the MS and $Ostro01$ methods.

 doi:10.20944/preprints202409.0876.v1

Table 10. Numerical results for F^8, $x^{(0)} = (1.25, 1.25)$

Iterative method	k	ρ	ϵ_{aprox}	ϵ_f	$\bar{\xi}$	Cpu-time
$MS(1,-1)$	6	4.00	1.019e-65	6.029e-261	$\bar{\xi}_1$	4.1403
$MS(1,1)$	5	4.05	2.474e-42	4.929e-177	$\bar{\xi}_1$	3.3999
$Traub - Ste$	6	4.00	8.946e-86	2.341e-342	$\bar{\xi}_1$	4.4433
$Ostro_{01}$	8	4.00	2.050e-55	5.429e-219	$\bar{\xi}_1$	4.4246
$M_{4,3}$	-	-	-	-	-	-
$M_{6,3}$	-	-	-	-	-	-

In table 10, method $MS(1,1)$ stands out as the one with the lowest computational time, requiring fewer iterations compared to the other methods and showing a convergence order of 4.05. The $Traub - Ste$ method, despite having the same convergence order as $MS(1,-1)$, shows a slightly higher computational time and larger errors in the difference between iterates. $Ostro_{01}$, which requires more iterations, falls only slightly behind in computational time but remains competitive overall. Methods $M_{4,3}$ and $M_{6,3}$ do not converge to the solution, as division by zero occurred when calculating divided differences.

Table 11. Numerical results for F^8, $x^{(0)} = (2, 2)$

Iterative method	k	ρ	ϵ_{aprox}	ϵ_f	$\bar{\xi}$	Cpu-time
$MS(1,-1)$	10	4.00	6.679e-71	1.113e-281	$\bar{\xi}_1$	6.9568
$MS(1,1)$	8	4.28	4.387e-77	3.113e-321	$\bar{\xi}_1$	5.4126
$Traub - Ste$	15	4.00	4.978e-100	2.442e-397	$\bar{\xi}_2$	10.6624
$Ostro_{01}$	-	-	-	-	-	-
$M_{4,3}$	-	-	-	-	-	-
$M_{6,3}$	-	-	-	-	-	-

Method $MS(1,1)$ stands out as the one with the lowest computational time, requiring fewer iterations compared to the other methods and showing a convergence order of 4.28. $Traub - Ste$ scheme, although it has the same convergence order as $MS(1,-1)$, shows a significantly higher computational time but better results in terms of errors. $MS(1,-1)$, while requiring more iterations than $MS(1,1)$, still maintains a competitive time compared to $Traub - Ste$. Division by zero is the reason why the other methods could not reach the solution.

Table 12. Numerical results for F^8, $x^{(0)} = (2.25, 2.25)$

Iterative method	k	ρ	ϵ_{aprox}	ϵ_f	$\bar{\xi}$	Cpu-time
$MS(1,-1)$	16	4.00	2.821e-94	3.543e-375	$\bar{\xi}_1$	10.5084
$MS(1,1)$	6	4.19	8.959e-45	1.550e-184	$\bar{\xi}_2$	4.2948
$Traub - Ste$	-	-	-	-	-	-
$Ostro_{01}$	$k > 50$	-	-	-	-	-
$M_{4,3}$	-	-	-	-	-	-
$M_{6,3}$	-	-	-	-	-	-

In Table 12, it is observed that only the members of our proposed class converge, and they do it to different solutions. $MS(1,1)$ stands out as the most efficient in terms of time and iterations, while $MS(1,-1)$ yields the best errors. The method $Ostro_{01}$ do not converge because it exceeded the maximum number of iterations, while the other methods could not reach the solution due to division by zero during the computation of divided differences.

 doi:10.20944/preprints202409.0876.v1

Table 13. Numerical results for F^8, $x^{(0)} = (2.5, 2.5)$

Iterative method	k	ρ	ϵ_{aprox}	ϵ_f	$\bar{\xi}$	Cpu-time
$MS(1,-1)$	10	4.00	2.063e-86	6.071e-344	$\bar{\xi}_2$	6.8361
$MS(1,1)$	7	4.17	3.281e-52	6.327e-218	$\bar{\xi}_1$	4.7535
$Traub - Ste$	-	-	-	-	-	-
$Ostro_{01}$	-	-	-	-	-	-
$M_{4,3}$	-	-	-	-	-	-
$M_{6,3}$	-	-	-	-	-	-

The observations for Table 13 are the same as those in the previous one, but now method $Ostro_{01}$ does not converge to the solution due to a division by zero, just like the other methods that do not converge.

5. Conclusions

The fifth-order Jacobian-Free iterative method with scalar accelerators developed in this study (along with its fourth-order partners) has proven to be an efficient tool for solving nonlinear systems of equations. It preserves the convergence order of the original scheme, despite the elimination of Jacobian matrix calculations, with low computational cost. The substitution of these matrices with divided differences not only reduces computational complexity but also facilitates implementation, maintaining high precision and efficiency. The numerical results highlight its superiority in terms of performance compared to conventional methods, especially for not-so-large systems.

Author Contributions: Conceptualization, J.R.T. and A.C.; software, A.R.-C.; methodology, J.G.M.; validation, J.G.M., A.C. and J.R.T.; investigation, A.C.; formal analysis, J.R.; resources, A.R.-C.; writing—original draft preparation, A.R.-C.; visualization, J.G.M.; writing—review and editing, J.R.T. and A.C.; supervision, J.R.T. and A.C. All the authors have read and agreed to the published version of this paper.

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

References

1. Samanskii, V.: On a modification of the Newton method. Ukr. Math. J. **19**, 133–138 (1967).
2. Steffensen, J.F.: Remarks on iteration. Skand. Aktuarietidskr. **1**, 64–72 (1933).
3. Ortega, J.M.; Rheinboldt, W.C. Iterative Solution of Nonlinear Equations in Several Variables; Academic Press: Cambridge, MA, USA, 1970.
4. Chicharro, F.I., Cordero, A., Gutiérrez, J.M., Torregrosa, J.R.: Complex dynamics of derivative-free methods for nonlinear equations. Appl. Math. Comput. **219**, 7023–7035 (2013).
5. Cordero, A., Soleymani, F., Torregrosa, J.R., Shateyi, S.: Basins of attraction for various Steffensen-type methods. Appl. Math. **2014**, Article ID 539707 1–17 (2014).
6. Cordero, A., García-Maimó, J., Torregrosa, J.R., Vassileva, M.P.: Solving nonlinear problems by Ostrowski–Chun type parametric families. Math. Chem. **53**, 430–449 (2015).
7. Jarratt, P.: Some fourth order multipoint iterative methods for solving equations. Math. Comp. **20**, 434–437 (1966).
8. Sharma, J.R., Arora, H.: On efficient weighted-Newton methods for solving systems of nonlinear equations. Appl. Math. Comput. **222**, 497–506 (2013).
9. Montazeri, H., Soleymani, F., Shateyi, S., Motsa, S.S.: On a new method for computing the numerical solution of systems of nonlinear equations. Appl. Math. **2012** ID. 751975, 1–15 (2012).
10. Sharma, J.R., Arora, H.: Efficient derivative-free numerical methods for solving systems of nonlinear equations. Comp. Appl. Math. **35**, 269–284 (2016).
11. Amiri, A.R., Cordero, A., Darvishi, M.T., Torregrosa, J.R.: Preserving the order of convergence: Low-complexity Jacobian-free iterative schemes for solving nonlinear systems. Comput. Appl. Math. **337**, 87–97 (2018).
12. Behl, R., Cordero, A., Torregrosa, J. R., Bhalla, S., A New High-Order Jacobian-Free Iterative Method with Memory for Solving Nonlinear Systems. Mathematics **9**(17), 2122 (2021).

13. Ostrowski, A.M., Solutions of Equations and Systems of Equations, Academic Press, New York, London, 1966.
14. Abad, M.F., Cordero, A., Torregrosa, J.R., A family of seventh-order schemes for solving nonlinear systems, Bull. Math. Soc. Sci. Math. Roumanie **57**(105)(2), 133–145 (2014).
15. Cordero, A. , Maimó, J.G. ,Torregrosa, J.R., Vassileva, M.P., Solving nonlinear problems by Ostrowski-Chun type parametric families, Math. Chem. **53**, 430–449 (2015).
16. Sharma, J.R., Arora, H., On efficient weighted-Newton methods for solving systems of nonlinear equations, Applied Mathematics and Computation **222**, 497–506 (2013).
17. Cordero, A, Rojas–Hiciano, R. V., Torregrosa, J. R., Vassileva, M. P: A highly efficient class of optimal fourth-order methods for solving nonlinear systems. Numerical Algorithms **95**, 1879–1904 (2024), doi:10.1007/s11075-023-01631-9.
18. Singh, H., Sharma, J. R., Kumar, S.: A simple yet efficient two-step fifth-order weighted-Newton method for nonlinear models. Numerical Algorithms **93**, 203–225 (2023).
19. Cordero, A., Torregrosa, J.R., Variants of Newton's method using fifth order quadrature formulas. Appl. Math. Comput. **190**, 686–698 (2007).
20. Traub, I.F. Iterative Methods for the Solution of Equations; Prentice-Hall: Englewood Cliffs, NJ, USA, 1964.