Mathematics

https://www.aimspress.com/journal/Math

AIMS Mathematics, 10(12): 28926–28953.
DOI: 10.3934/math.20251273
Received: 04 August 2025
Revised: 15 September 2025
Accepted: 13 October 2025
Published: 10 December 2025

Research article

An effective double direction method for solving convex-constrained nonlinear equations with image restoration

Muhammad Abdullahi[1,2,*], Abdullah Al-Yaari[1], Auwal Bala Abubakar[3,4,5,*], Abubakar Sani Halilu[2,6] and Muhamad Afendee Muhamed[6]

[1] School of Mathematics and Statistics, HNP-LAMA, Central South University, Changsha, Hunan 410083, China

[2] Mathematical Innovation and Applications Research Group, Department of Mathematics, Sule Lamido University Kafin Hausa, PMB 048, Nigeria

[3] Department of Art and Science, George Mason University, Songdomunhwa-ro 119-4 Yeonsu-gu, Incheon 21985, Korea

[4] Numerical Optimization Research Group, Department of Mathematical Sciences, Faculty of Physical Sciences, Bayero University Kano, Kano, Nigeria

[5] Department of Mathematics and Applied Mathematics, Sefako Makgatho Health Sciences University, Ga-Rankuwa, Pretoria, Medunsa-0204, South Africa

[6] Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin, Kuala Terengganu 21300, Malaysia

* **Correspondence:** Email: 4.muhammad.abdullahi@gmail.com, ababubakar@buk.edu.ng.

Abstract: The restoration of medical imaging has recently gained significant attention due to its critical role in the medical sciences. This research introduces two efficient double-direction approaches for solving large-scale monotone nonlinear equations and addressing medical image restoration problems. The acceleration parameter in the first algorithm was derived by approximating the Jacobian matrix with a diagonal matrix, while the second algorithm integrates a correction parameter using the Picard-Mann hybrid iterative procedure. The study established the descent condition and demonstrated both the global convergence and R-linear convergence rate of the proposed method under favorable conditions. Numerical experiments showed that the procedure significantly enhances the numerical performance of the proposed methods. Additionally, one of the techniques was successfully applied to restore blurred medical images, highlighting its practical applicability in the field of medical imaging.

Keywords: medical image; convex-constrained; acceleration parameter; Picard-Mann; iterative procedure
Mathematics Subject Classification: 65K05, 90C52, 90C56, 94A08

1. Introduction

The system of nonlinear equations is essential in various scientific problems. These equations frequently arise in mathematical problems across multiple fields, such as natural sciences, social sciences, and engineering. A typical expression for a system of nonlinear equations is represented as follows:

$$F(v) = 0, \quad v \in \mathcal{D}, \tag{1.1}$$

where the set $\mathcal{D} \subseteq \mathbb{R}^n$ is a nonempty closed and convex set, and $F : \mathcal{D} \subseteq \mathbb{R}^n \to \mathbb{R}^n$ is a nonlinear map. In this article, $\mathbb{R}^n$ represents the n-dimensional real space equipped with the Euclidean norm $\|\cdot\|$.

Several studies have explored various iterative techniques to solve (1.1) for an extended period, including Newton's methods, and notably, quasi-Newton methods [2], Gauss-Newton methods [21], Levenberg-Marquardt methods [15, 31], and trust region methods [18]. The Newton method and its type are highly valued for their straightforward implementation and quadratic convergence. Their general iterative technique can be described as follows:

$$v_{k+1} = v_k + \alpha_k d_k, \quad k = 0, 1, \ldots \tag{1.2}$$

where v_{k+1} denotes the current iterate, while v_k refers to the preceding iterate. The step size, $\alpha_k > 0$, can be determined using an appropriate line search. The search direction d_k can be calculated using $d_k = -F(v_k) + \mathcal{P}_k$, where $\mathcal{P}_k = (I - (F(v_k)')^{-1})F(v_k)$, with I standing for the identity matrix and $F(v_k)'$ representing the Jacobian matrix at v_k. It is clear that when $\mathcal{P}_k = 0$, then d_k simplifies to the steepest descent direction. Despite their appealing properties, Newton's method and its type are not often utilized for large-scale problems. This necessity arises mainly from the requirement for iterative calculations and the storage of derivatives for the Jacobian matrix and its inverse within the system. However, in some cases, the derivatives of certain functions may not exist, a fact widely acknowledged in the field. To address this challenge, numerous studies have proposed matrix-free approaches, for more details (see [6, 14, 23]).

The double-direction method is another variant of the matrix-free method. The double-direction method incorporates two corrections in its iterative procedure. This means that if one correction fails during the iterative process, the second correction will still adjust the system. As a result, quite a number of studies have explored a double-direction approach [8, 9, 11] to address large-scale nonlinear equations. This concept was first introduced in [8], which utilized multi-step iteration to generate iterates. Later, [9] developed a multi-step algorithm specifically for minimizing a non-differentiable function using a double-direction technique building on the foundations laid by [7, 8].

Improving the theoretical and numerical convergence of the double-direction method is essential. Recent studies indicate that hybridizing the optimization algorithm with the Picard-Mann iterative process proposed in [13] significantly enhances its overall theoretical and numerical performance, for more details, see [10, 26]. These findings have inspired several authors to combine the Picard-Mann iterative process [13] with other optimization methods to develop more reliable approaches. For example, Petrovic et al. [22] introduced a hybrid method that integrates a dynamic line search for optimal step size with the Picard-Mann iterative process. Comprehensive numerical experiments signify that the hybrid method outperforms traditional approaches, especially for solving large-scale optimization problems. This indicates that the combination effectively accelerates convergence and improves the algorithm's robustness.

Similarly, Halilu et al. [11] introduced double-direction techniques that integrate hybrid techniques from [13] with an improved matrix-free method; the primary concept used in the initial algorithm is to approximate the Jacobian matrix and derive an acceleration parameter using Taylor series expansion, thereby proposing a robust derivative-free approach. Moreover, the second algorithm combines the derived acceleration parameter with a Picard-Mann hybrid iterative technique. These techniques are designed to effectively solve large-scale systems of nonlinear equations and exhibit global convergence under certain conditions. Numerical experiments validate the efficacy of these techniques, demonstrating notable performance improvements compared to existing methods.

Additionally, there exists a significant advancement in the realm of convex-constrained monotone nonlinear equations with the projection technique proposed in [25]. For instance, inspired by the projection technique [25] and the spectral gradient method [4], Yu et al. [28] improved the spectral gradient projection method presented in [33] by adapting it from unconstrained to constrained nonlinear monotone equations. The findings in [28] demonstrated that this new approach is highly effective, particularly for large-scale problems. Since its inception, it has attracted significant interest, leading to the development of various efficient methods for solving convex-constrained monotone equations (for more details, see [20, 24, 29]).

The existing optimization methods [11, 17, 22, 26] perform better when combined with the Picard-Mann iterative process [13]. For this reason, this research is motivated by the work of [11], the favorable characteristics of the Picard-Mann iterative process [13], and their absence for solving large-scale convex-constrained monotone nonlinear equations. The study aims to apply the concepts of the Picard-Mann iterative process to a double-direction approach for convex-constrained monotone nonlinear equations with application in medical image recovery problems.

The following are some of this paper's contributions.

- ♣ The study presents new double-direction techniques for solving the convex-constrained nonlinear equations.
- ♣ The two new derivative-free search directions were developed using an acceleration parameter, which was derived by approximating the Jacobian matrix. Additionally, the later direction was derived via the Picard-Mann iterative procedure.
- ♣ The proposed method satisfies the descent condition, the global convergence, and the R-linear convergence property under certain assumptions.
- ♣ The proposed algorithm is applied to solve the medical image recovery problem.

The research is structured as follows. Section 2 discusses the derivation of the proposed approaches. In Section 3, the convergence analysis of the proposed algorithms is presented. Section 4 includes some numerical experiments and image recovery implementations of the proposed methods. The article concludes in Section 5.

2. Preliminaries and the proposed

method This section presents the Picard-Mann-based iterative algorithm for solving convex-constrained nonlinear equations. It also presents the algorithm steps and some preliminaries. To begin with, the derivation, the study considers a general iterative scheme of the double-direction defined as follows:

$$v_{k+1} = v_k + \alpha_k \widehat{d_k^1} + \alpha_k^2 \widehat{d_k^2}, \tag{2.1}$$

where v_{k+1} represents the current iteration, and v_k represents the previous iteration, α_k is the corresponding line search, and $\widehat{d_k^1}$ and $\widehat{d_k^2}$ represent the respective search directions defined as follows:

$$\widehat{d_k^i} = \begin{cases} -\delta_k^{-1} F(v_k), & for \ \ i = 1, \\ -F(v_k), & for \ \ i = 2, \end{cases} \tag{2.2}$$

where δ_k is the acceleration parameter. Substituting (2.2) in (2.1), we obtain the following general scheme:

$$v_{k+1} = v_k - \mu_k \delta_k^{-1} F(v_k), \tag{2.3}$$

where $\mu_k = (\alpha_k + \alpha_k^2 \delta_k)$. Next, we derive the acceleration parameter δ_k. Although Taylor series expansion has been commonly used for this purpose (for more information, see [12, 17, 26]), we focus on using the Broyden update approximation. However, for large-scale dimensions, we cannot apply Broyden's update directly due to:

- accumulated errors in the Jacobian approximation from rank-one updates:
- instability and poor convergence when initial estimates are inaccurate or the problem is ill-conditioned.

To mitigate these shortcomings, we approximate the Broyden update with a scaled identity matrix $B_k \approx \delta_k I$, which simplifies computation and improves numerical stability. Now we consider Broyden's matrix updating formula given as follows:

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k) s_k^T}{s_k^T s_k}, \tag{2.4}$$

approximating the Jacobian matrix B_k defined in (2.4) as $B_k \approx \delta_k I$, where δ_k is the acceleration parameter and I stand for identity matrix. Therefore, (2.4) can be simplified to

$$\delta_{k+1} I = \delta_k I + \frac{(y_k - \delta_k s_k) s_k^T}{s_k^T s_k}. \tag{2.5}$$

The first choice of the proposed δ_{k+1} is obtained by multiplying (2.5) by $F(v_k)$ and then by s_k^T, and simplifying further, which yields

$$\delta_{k+1} = \delta_k + \frac{s_k^T (y_k - \delta_k s_k)}{s_k^T s_k} = \frac{s_k^T y_k}{s_k^T s_k}. \tag{2.6}$$

The second choice of the proposed δ_{k+1} is obtained by multiplying (2.5) by s_k and then by y_k^T, and simplifying further, which yields

$$\delta_{k+1} = \delta_k + \frac{y_k^T (y_k - \delta_k s_k)}{y_k^T s_k} = \frac{y_k^T y_k}{y_k^T s_k}. \tag{2.7}$$

The δ_{k+1} defined in (2.6) and (2.7) above can further be simplified as follows:

$$\delta_{k+1} = \max \left\{ \frac{s_k^T y_k}{s_k^T s_k}, \ \frac{y_k^T y_k}{y_k^T s_k} \right\}, \tag{2.8}$$

where $s_k = v_{k+1} - v_k$ and $y_k = F(v_{k+1}) - F(v_k) + \gamma s_k$, where $\gamma > 0$.
To this end, we present the first part of the proposed algorithm step for finding a solution to (1.1) (see Algorithm 1.1).

Algorithm 1.1:

```
Definitions:
Inputs: Initial point $v_0 \in \mathcal{D}$, initial search direction $\widehat{d_0^1} = -F(v_0)$; parameters $\rho \in (0,1)$,
  $\sigma,\ \gamma > 0$, $0 < w < 2$, $tol > 0$, initial scalar $\delta_0 = 1$, and $\xi > 0$.
Input: $v_0, \widehat{d_0^1}, \gamma > 0, \rho, \sigma, w, tol, \delta_0, \xi$
Output: Solution $\hat{v}$ such that $||F(\hat{v})|| \leq tol$
1: Set iteration counter $k \leftarrow 0$;
2: while $||F(v_k)|| > tol$ do
3:     Compute $F(v_k)$;
4:     if $||F(v_k)|| \leq tol$ then
5:         return $v_k$
6:     end
7:     Set step size $\alpha_k = \xi\rho^{m_i}$, where $m_i$ is the smallest nonnegative integer $m$ such that:
           $$-F(v_k + \mu_k\widehat{d_k^1})^T\widehat{d_k^1} \geq \sigma\mu_k||F(v_k + \mu_k d_k)||||\widehat{d_k^1}||^2, \quad (2.9)$$
        where $\mu_k = \alpha_k + \alpha_k^2\delta_k$;
8:     Set $z_k = v_k + \mu_k\widehat{d_k^1}$;
9:     if $z_k \in \mathcal{D}$ and $||F(z_k)|| \leq \varepsilon$ then
10:        return $z_k$
11:    end
12:    Compute step size:
           $$\lambda_k = \frac{F(z_k)^T(v_k - z_k)}{||F(z_k)||^2} \quad (2.10)$$
13:    Update iterate:
           $$v_{k+1} = P_{\mathcal{D}}[v_k - w\lambda_k F(z_k)] \quad (2.11)$$
14:    Compute $\delta_{k+1}$ using (2.8);
15:    Evaluate the new search direction $\widehat{d_{k+1}^1}$ using (2.2);
16:    Set $k \leftarrow k + 1$;
17: end
18: return $v_k$
```

Furthermore, the study presents a second algorithm by using a modify version of the iterative scheme defined in (2.1), and later deriving a search direction embedded with a correction parameter by means of the Picard-Mann iterative procedure. The correction parameter will significantly improve the overall numerical performance of the proposed algorithm. Consider the function $T(v_k)$ defined with respect to (2.3) as $T(v_k) = v_k - (\alpha_k - \alpha_k^2\delta_k)\delta_k^{-1}F(v_k)$. Therefore, the corresponding Picard-Mann is defined as follows:

$$\begin{cases} v_1 = v \in \mathcal{D}, \\ v_{k+1} = T(h_k) = h_k - \alpha_k \delta_k^{-1} F(v_k) - \alpha_k^2 F(v_k), \\ h_k = (1-\lambda)v_k + \lambda(v_k - \alpha_k \delta_k^{-1} F(v_k) - \alpha_k^2 F(v_k)) \\ \quad = v_k - \lambda(\alpha_k + \alpha_k^2 \delta_k)\delta_k^{-1} F(v_k), \quad k \in \mathbb{N}. \end{cases}$$

Substituting the third equation into the second equation above, we obtain a general iteration scheme as follows:

$$v_{k+1} = v_k - (\lambda + 1)\mu_k \delta_k^{-1} F(v_k), \tag{2.12}$$

where $(\lambda+1) \in (1,2)$, $\mu_k = \left(\alpha_k + \alpha_k^2 \delta_k\right)$, and α_k is the corresponding step length. Therefore from (2.12), the second search direction d_k is proposed as follows:

$$d_k = \begin{cases} -F(v_k), & for \;\; k = 0, \\ -(\lambda+1)\delta_k^{-1} F(v_k), & for \;\; k \geq 1. \end{cases} \tag{2.13}$$

Remark 2.1. *In the proposed search direction* (2.13)*, we introduce a modification of the direction in* (2.2)*. Specifically, when the acceleration parameter* δ_k *vanishes at some iteration, Eq* (2.2) *degenerates to the classical steepest descent direction, which is well known for its slow convergence. To circumvent this limitation, we incorporate a correction parameter* $(\lambda + 1) \in (1,2)$*, ensuring that the proposed search direction never coincides with the steepest descent direction at any iteration. This adjustment enhances the robustness of the method by mitigating the poor convergence behavior typically associated with steepest descent.*

Therefore, the algorithm steps for the proposed second algorithm are defined below (see Algorithm 2.1):

Algorithm 2.1:

Definitions:

Inputs: Initial iterate $v_0 \in \mathcal{D}$, initial search direction $d_0 = -F(v_0)$; parameters $\lambda, \rho \in (0,1)$, $\sigma,\ \gamma > 0$, $0 < w < 2$, $tol > 0$, $\gamma > 0$, $\xi > 0$, and initial scalar $\delta_0 = 1$.

Input: $v_0, d_0, \lambda, \rho, \sigma, w, tol, \delta_0, \xi$

Output: Solution $\hat{v}$ such that $\|F(\hat{v})\| \le tol$

```
1: Set iteration counter $k \leftarrow 0$;
2: while $\|F(v_k)\| > tol$ do
3:     Compute $F(v_k)$;
4:     if $\|F(v_k)\| \le tol$ then
5:         return $v_k$
6:     end
7:     Set step size $\alpha_k = \xi\rho^{m_i}$, where $m_i$ is the smallest nonnegative integer $m$ such that:
           $$-F(v_k + \mu_k d_k)^T d_k \ge \sigma\mu_k \|F(v_k + \mu_k d_k)\| \|d_k\|^2, \quad (2.14)$$
       where $\mu_k = \alpha_k + \alpha_k^2\delta_k$;
8:     Set $z_k = v_k + \mu_k d_k$;
9:     if $z_k \in \mathcal{D}$ and $\|F(z_k)\| \le \varepsilon$ then
10:        return $z_k$
11:    end
12:    Compute step size:
           $$\lambda_k = \frac{F(z_k)^T (v_k - z_k)}{\|F(z_k)\|^2}. \quad (2.15)$$
13:    Update iterate:
           $$v_{k+1} = P_{\mathcal{D}}[v_k - w\lambda_k F(z_k)]. \quad (2.16)$$
14:    Compute $\delta_{k+1}$ using (2.8);
15:    Evaluate the new search direction $d_{k+1}$ using (2.13);
16:    Set $k \leftarrow k + 1$;
17: end
18: return $v_k$
```

Let $\mathcal{D} \subseteq \mathbb{R}^n$ be a nonempty, closed, and convex set. Then for any $v \in \mathbb{R}^n$, its orthogonal projection onto $\mathcal{D}$, denoted by $P_{\mathcal{D}}[v]$, is defined by

$$P_{\mathcal{D}}[v] := arg\min\{\|v - y\| : y \in \mathcal{D}\}. \tag{2.17}$$

Suppose for any $v, y \in \mathbb{R}^n$, the mapping $P_{\mathcal{D}} : \mathbb{R}^n \to \mathcal{D}$ is known as the projection operator, which has a non-expansive property. Then, it holds that

$$\|P_{\mathcal{D}}[v] - P_{\mathcal{D}}[y]\| \le \|v - y\|, \ \forall v, y \in \mathbb{R}^n.$$

Consequently, we have

$$\|P_{\mathcal{D}}[v] - y\| \le \|v - y\|, \quad \forall y \in \mathcal{D}. \tag{2.18}$$

3. Convergence analysis

This section presents the global convergence of the proposed Algorithm 2.1. To proceed, consider the following assumptions and remark.

Assumption 3.1. *Consider the following assumptions:*

H1. The operator F is Lipschitz continuous, that is, there exists a positive constant $\mathcal{L} > 0$ such that

$$\|F(v) - F(y)\| \leq \mathcal{L}\|v - y\|, \quad \forall v, y \in \mathbb{R}^n. \tag{3.1}$$

H2. The solution set of (1.1) *is nonempty and is denoted by $\mathcal{D}^v$.*

H3. The mapping F is monotone; that is,

$$(F(v) - F(y))^T(v - y) \geq 0, \quad \forall v, y \in \mathbb{R}^n. \tag{3.2}$$

Remark 3.2. *Consider the following remark:*

(a) Using Assumption H3 and the definitions of s_k and y_k, we have

$$s_k^T y_k = s_k^T(F(v_{k+1}) - F(v_k)) + \gamma s_k^T s_k \geq \gamma\|s_k\|^2 > 0.$$

(b) From Assumption H1, and the definition of y_k, we have

$$\|y_k\| = \|F(v_{k+1}) - F(v_k) + \gamma s_k\| \leq \|F(v_{k+1}) - F(v_k)\| + \gamma\|s_k\| \leq \mathcal{L}\|s_k\| + \gamma\|s_k\| \leq (\mathcal{L} + \gamma)\|s_k\|.$$

(c) Case 1: If $\max\left\{\frac{s_k^T y_k}{s_k^T s_k}, \frac{y_k^T y_k}{y_k^T s_k}\right\} = \frac{s_k^T y_k}{s_k^T s_k}$, from Remark 3.2(a), we have

$$\delta_{k+1} = \frac{s_k^T y_k}{s_k^T s_k} \geq \frac{\gamma\|s_k\|^2}{\|s_k\|^2} = \gamma.$$

Since $\delta_{k+1} > 0$, this clearly shows that it's positive.

Case 2: If $\max\left\{\frac{s_k^T y_k}{s_k^T s_k}, \frac{y_k^T y_k}{y_k^T s_k}\right\} = \frac{y_k^T y_k}{y_k^T s_k}$, from Remark 3.2 (a) and (b), we have

$$\delta_{k+1} = \frac{y_k^T y_k}{y_k^T s_k} \leq \frac{\|y_k\|^2}{\gamma\|s_k\|^2} \leq \frac{(\mathcal{L} + \gamma)^2\|s_k\|^2}{\gamma\|s_k\|^2} = \frac{(\mathcal{L} + \gamma)^2}{\gamma}.$$

Therefore, for the two cases, the bounds for δ_{k+1}^{-1} are obtained as follows:

$$\gamma \leq \delta_{k+1} \leq \frac{(\mathcal{L} + \gamma)^2}{\gamma} = M.$$

Thus it implies

$$\frac{1}{M} \leq \frac{1}{\delta_{k+1}} \leq \frac{1}{\gamma}.$$

Hence $\delta_{k+1}^{-1} \in \left[\frac{1}{M}, \frac{1}{\gamma}\right]$.

Lemma 3.3. *Suppose Assumptions H1–H3 hold, and the sequence $\{v_k\}$ is generated by Algorithm 2.1, then the search direction d_k satisfies the following descent property:*

$$F(v_k)^T d_k \leq -p\|F(v_k)\|^2, \quad \forall k > 0. \tag{3.3}$$

Proof. For $k = 0$, from (2.13), we have $F(v_0)^T d_0 = -\|F(v_0)\|^2$, where $p = 1$.
For $k \geq 1$, from (2.6) and (2.13), and the fact that $\lambda \in (0, 1)$, we have

$$\begin{aligned} F(v_k)^T d_k &= -(\lambda + 1)\delta_{k+1}^{-1}\|F(v_k)\|^2 \\ &\leq -\frac{(\lambda + 1)}{M}\|F(v_k)\|^2 \\ &= -p\|F(v_k)\|^2, \end{aligned} \tag{3.4}$$

where $p := \frac{(\lambda+1)}{M}$. □

Next, we show the boundedness of the search direction.
For $k = 0$. From (2.13) and (3.17) we have $\|d_0\| = \|F(v_0)\|$.
For $k > 0$, using Remark 3.2(a), (2.13), and (3.17), we have

$$\|d_k\| = \left\|-(\lambda + 1)\delta_{k+1}^{-1}F(v_k)\right\| \leq (\lambda + 1)\left|\delta_{k+1}^{-1}\right|\|F(v_k)\| \leq \frac{(\lambda + 1)\|F(v_k)\|}{M} = \Omega.$$

Taking $\Omega = \frac{(\lambda+1)\|F(v_k)\|}{M}$, we have

$$\|d_k\| \leq \Omega, \quad \forall k \geq 0. \tag{3.5}$$

Lemma 3.4. *Let Assumptions H1–H3 be true, and the sequences $\{v_k\}$ and $\{z_k\}$ are generated by the Algorithm 2.1. Then there exists a step size μ_k satisfies the following inequality:*

$$\mu_k \geq \mu := \min\left\{\xi, \frac{\rho p}{(\mathcal{L} + \sigma\|F(z_k)\|)\,\frac{(\lambda+1)^2}{M^2}}\right\}. \tag{3.6}$$

Proof. If $\mu_k \neq \xi$, then by the definition of $\mu_k = (\alpha_k + \alpha_k^2\delta_k)$, $\rho^{-1}\mu_k$ does not satisfies (2.14), that is,

$$-F(v_k + \rho^{-1}\mu_k d_k)^T d_k < \sigma\rho^{-1}\mu_k\|F(z_k)\|\|d_k\|^2. \tag{3.7}$$

Therefore, using Assumption H1, combining (3.3) with (3.7), we obtain

$$\begin{aligned} p\|F(v_k)\|^2 &\leq -F(v_k)^T d_k \\ &= (F(v_k + \rho^{-1}\mu_k d_k) - F(v_k))^T d_k - F(v_k + \rho^{-1}\mu_k d_k)^T d_k \\ &\leq \mathcal{L}\rho^{-1}\mu_k\|d_k\|^2 + \sigma\rho^{-1}\mu_k\|F(z_k)\|\|d_k\|^2 \\ &= \mu_k(\mathcal{L} + \sigma\|F(z_k)\|)\rho^{-1}\|d_k\|^2. \end{aligned} \tag{3.8}$$

Therefore, from the boundedness of the search direction in (3.5), it is easy to see that (3.8), we have

$$\begin{aligned} \mu_k &> \frac{\rho p\|F(v_k)\|^2}{(\mathcal{L} + \sigma\|F(z_k)\|)\|d_k\|^2} \\ &\geq \frac{\rho p}{(\mathcal{L} + \sigma\|F(z_k)\|)\frac{(\lambda+1)^2}{M^2}}. \end{aligned} \tag{3.9}$$

□

Lemma 3.5. *Suppose Assumptions H1–H3 hold. Suppose the sequences* $\{v_k\}$ *and* $\{z_k\}$ *are generated by Algorithm 2.1, and then* $\{v_k\}$ *and* $\{z_k\}$ *are bounded. Additionally,*

$$\lim_{k\to\infty} \|v_k - z_k\| = 0, \tag{3.10}$$

and

$$\lim_{k\to\infty} \|v_{k+1} - v_k\| = 0. \tag{3.11}$$

Proof. The first part of the proof will focus on the boundedness of the sequences $\{v_k\}$ and $\{z_k\}$. Suppose $\widehat{v} \in \mathcal{D}^v$ is any solution of (1.1). Taking into consideration that F is monotone, we have

$$F(z_k)^T(v_k - \widehat{v}) \geq F(z_k)^T(v_k - z_k). \tag{3.12}$$

Utilizing the line search (2.14), and definition of z_k, we have

$$F(z_k)^T(v_k - \widehat{v}) \geq F(z_k)^T(v_k - z_k) \geq \sigma\mu_k^2\|F(z_k)\|\|d_k\|^2 > 0. \tag{3.13}$$

From the non-expensive projection (2.18), (2.16), and (3.13), it is easy to see that

$$\begin{aligned}
\|v_{k+1} - \widehat{v}\|^2 &= \|P_{\mathcal{D}}[v_k - w\lambda_k F(z_k)] - \widehat{v}\|^2 \\
&\leq \|v_k - w\lambda_k F(z_k) - \widehat{v}\|^2 \\
&= \|v_k - \widehat{v}\|^2 - 2w\lambda_k\langle F(z_k), v_k - \widehat{v}\rangle + w^2\lambda_k^2\|F(z_k)\|^2 \\
&\leq \|v_k - \widehat{v}\|^2 - 2w\lambda_k\langle F(z_k), v_k - z_k\rangle + w^2\lambda_k^2\|F(z_k)\|^2 \\
&= \|v_k - \widehat{v}\|^2 - w(2-w)\frac{(\langle F(z_k), (v_k - z_k)\rangle)^2}{\|F(z_k)\|^2} \qquad (3.14) \\
&\leq \|v_k - \widehat{v}\|^2 - \widehat{\sigma}\|v_k - z_k\|^4, \qquad (3.15)
\end{aligned}$$

where $\widehat{\sigma} = w(2-w)\sigma^2$, and $0 < w < 2$. From (3.14), we obtain

$$\|v_{k+1} - \widehat{v}\|^2 \leq \|v_k - \widehat{v}\|^2.$$

Thus, this implies the sequence $\{\|v_k - \widehat{v}\|\}$ is a decreasing sequence and convergent, thus $\{v_k\}$ is bounded. Furthermore, utilizing (3.14) and Assumptions H1–H3, we have

$$\|F(v_k)\| = \|F(v_k) - F(\widehat{v})\| \leq \mathcal{L}\|v_k - \widehat{v}\| \leq \mathcal{L}\|v_0 - \widehat{v}\|. \tag{3.16}$$

Letting $\mathcal{L}\|v_0 - \widehat{v}\| = N$, then we have

$$\|F(v_k)\| \leq N, \ \forall k \geq 0. \tag{3.17}$$

Since the sequences $\{v_k\}$ and $\{d_k\}$ are bounded by the definition of $\{z_k\}$ in Step 3 in Algorithm 2.1 it shows that the sequence $\{z_k\}$ is bounded. Therefore, using the same argument as in (3.16), we claim that there is a positive constant $\mathcal{P} > 0$ such that

$$\|F(z_k)\| \leq \mathcal{P}, \ \ \forall k \geq 0. \tag{3.18}$$

From (3.13), we obtain

$$(F(z_k)^T(v_k - z_k))^2 \leq \frac{\|F(z_k)\|^2}{w(2-w)}(\|v_k - \widehat{v}\|^2 - \|v_{k+1} - \widehat{v}\|^2). \tag{3.19}$$

Also, from (3.13) and (3.19), it is not difficult to see that

$$\mu_k^4\|F(z_k)\|^2\|d_k\|^4 \leq \frac{\|F(z_k)\|^2}{\sigma^2 w(2-w)}(\|v_k - \widehat{v}\|^2 - \|v_{k+1} - \widehat{v}\|^2). \tag{3.20}$$

Since the sequence $\{\|v_k - \widehat{v}\|^2\}$ is convergent, then using the facts that$\sigma > 0$, $0 < w < 2$, and $\{F(z_k)\}$ is bounded, and taking the limit on both sides of (3.20), we obtain

$$\lim_{k\to\infty} \mu_k^4\|d_k\|^4 \leq 0.$$

Thus, it holds that

$$\lim_{k\to\infty} \mu_k\|d_k\| = 0. \tag{3.21}$$

Therefore, combining with the definition of z_k implies that (3.10) was established. In addition, from the definition of λ_k and $P_{\mathcal{D}}[.]$ in (2.18), we have

$$\begin{aligned}
\|v_{k+1} - v_k\| &= \|P_{\mathcal{D}}[v_k - w\lambda_k F(z_k)] - v_k\| \\
&\leq \|v_k - w\lambda_k F(z_k) - v_k\| \\
&= \|w\lambda_k F(z_k)\| \\
&= w\|v_k - z_k\|,
\end{aligned}$$

which yields

$$\lim_{k\to\infty} \|v_{k+1} - v_k\| = 0.$$

This implies (3.11) is also established. Hence, the proof is complete. □

Theorem 3.6. *Suppose Assumptions H1–H3 hold with* $\{v_k\}$ *being the sequence of iterative points generated by Algorithm 2.1. Then* $\{v_k\}$ *converges to a solution of* (1.1).

Proof. First, it can be observed from (3.6) and (3.21) that $0 \leq \mu\|d_k\| \leq \mu_k\|d_k\| \to 0$. This consistently implies that $\lim\limits_{k\to\infty} \|d_k\| = 0$. This together with (3.3) yields,

$$0 \leq p\|F(v_k)\| \leq \|d_k\| \to 0,$$

which implies $\lim\limits_{k\to\infty} \|F(v_k)\| = 0$. Now (3.21), and the boundednes of the sequence $\{v_k\}$ shows that there exists at least one cluster point of $\{v_k\}$. Let $\{\widehat{v}\}$ be an accumulation point of $\{v_k\} \subset \mathcal{D}$ and let $\mathcal{K} \subset \{0, 1, 2, \ldots\}$ be an infinite indexing set for which

$$\lim_{k\to\infty,\ k\in\mathcal{K}} v_k = \widetilde{v} \in \mathcal{D}.$$

Since $\mathcal{D}$ is closed by Assumptions H1–H3, we have

$$0 = \lim_{k\to\infty} \|F(v_k)\| = \lim_{k\to\infty,\ k\in\mathcal{K}} \|F(v_k)\| = \|F(\widehat{v})\|,$$

which shows $\widetilde{v}$ to be a solution of (1.1). Also from Lemma 3.5, we know the sequence $\{\|v_k - \widehat{v}\|\}$ is convergent. Thus by letting $\widetilde{v} := \widehat{v}$, we obtain

$$\lim_{k\to\infty} \|v_k - \widetilde{v}\| = \lim_{k\to\infty,\ k\in\mathcal{K}} \|v_k - \widetilde{v}\| = 0.$$

Hence, the sequence $\{v_k\}$ converges to $\widetilde{v} \in \mathcal{D}^v$. □

4. Convergence rate

Assumption 4.1. *For any $\bar{v} \in \mathcal{D}'$, there exist a positive constants φ and ε such that*

$$\varphi dist(v, \mathcal{D}') \leq \|F(v)\|^2, \forall v \in N(\bar{v}, \varepsilon), \tag{4.1}$$

where $dist(v, \mathcal{D}')$ is the distance from v to the solution set $\mathcal{D}'$, and $N(\bar{v}, \varepsilon) = \{v \in \mathbb{R}^n : \|v - \bar{v}\| \leq \varepsilon\}$.

Theorem 4.2. *Let Assumptions 4.1 and H2 hold. Suppose the sequences $\{v_k\}$ and $\{z_k\}$ are generated by* ***Algorithm 2.1****. Therefore, the sequence $\{dist(v_k, \mathcal{D}')\}$ is Q-linearly convergence to 0.*

Proof. Let $h_k := \text{argmin}\{\|v_k - h\| : h \in \mathcal{D}'\}$. This means that h_k is the nearest solution to v_k,

$$\|v_k - h_k\| = dist(v_k, \mathcal{D}'), \quad \forall k \geq 0. \tag{4.2}$$

Combining (3.3) with the Cauchy-Schwarz inequality, we obtain

$$\|d_k\| \geq p\|F(v_k)\|. \tag{4.3}$$

From (3.15), (4.2), and (4.3), we have

$$\begin{aligned}
dist(v_{k+1}, \mathcal{D}')^2 &\leq \|v_{k+1} - h_k\|^2 \\
&\leq \|v_k - h_k\|^2 - \widehat{\sigma}\|v_k - z_k\|^4 \\
&\leq dist(v_k, \mathcal{D}')^2 - \widehat{\sigma}\mu_k^4\|d_k\|^4 \\
&\leq dist(v_k, \mathcal{D}')^2 - \widehat{\sigma}\mu_k^4 p^4\|F(v_k)\|^4 \\
&\leq dist(v_k, \mathcal{D}')^2 - \widehat{\sigma}\mu_k^4 p^4\varphi^2 dist(v_k, \mathcal{D}')^2 \\
&= (1 - \widehat{\sigma}p^4\varphi^2\mu_k^4) dist(v_k, \mathcal{D}')^2.
\end{aligned}$$

Taking $\varphi = \frac{1}{\sqrt{\widehat{\sigma}}p^2}$, we have $\widehat{\sigma}p^4\varphi^2\mu_k^4 \in (0, 1)$. Therefore, this indicates that the sequence $\{dist(x_k, \mathcal{D}')\}$ converges at a Q-linear rate to 0. □

Remark 4.3. *It is essential to note that when $\lambda = 0$, the proof of Algorithm 1.1 follows immediately.*

5. Numerical simulations

This section presents numerical results demonstrating the effectiveness of the proposed Algorithms 1.1 and 2.1. The study compares the proposed algorithms with other existing methods, as listed below. The MATLAB codes were developed using version 8.3.0 (R2023b) and run on a personal computer with a 2.80 GHz CPU, 16 GB of RAM, and 1 TB of memory. This analysis compared proposed Algorithm 1.1 defined in (2.2) and Algorithm 2.1 defined in (2.13), and the parameters used are: $\lambda = 0.4$ $\sigma = 10^{-4}$, $w = 1.76$, $\xi = 1$, $(\lambda + 1) = 1.2$, and $\rho = 0.90$. The search directions for the comparison algorithms and their respective parameters are outlined as follows:

(a) HDDM proposed in [11].

$$d_k = \begin{cases} -F(v_k), & \text{if } k = 0, \\ -t\delta_k^{-1} F(v_k), & \text{if } k \geq 1, \end{cases} \tag{5.1}$$

where $y_{k-1} = F(z_k) - F(v_{k-1}),\ \ s_k = z_k - v_k,\ \ \delta_{k+1} = \frac{\|s_k\|^2 \|y_k\|^2}{(\alpha_k + \alpha_k^2 \delta_k)^2 (y_k^T \delta_k^i)^2}$.

The parameters are given as follows: $t = 1.20,\ \sigma = 10^{-4}, \rho = 0.90,\ \xi = 1$.

(b) IMCNS proposed in [24].

$$d_k = \begin{cases} -F(v_k), & \text{if } k = 0, \\ -\eta F(v_k) + \beta_k s_{k-1}, & \text{if } k \geq 1, \end{cases} \tag{5.2}$$

where $y_{k-1} = F(v_k) - F_{k-1} + r s_{k-1},\ \ \beta_k = \frac{(\|y_{k-1}\|^2 - y_{k-1}^T s_{k-1}) s_{k-1}^T F(v_k)}{\theta_k},\ \ s_{k-1} = v_k - v_{k-1}$,
$\theta_k = \max\{\|s_{k-1}\|^2 \|y_{k-1}\|^2,\ N\|s_{k-1}\|^4\}, N > 0$.
The parameters are given as follows: $r = 0.001,\ \ \rho = 0.30,\ \ \sigma = 10^{-6},\ \ N = 1$.

(c) **Algorithm 2.1** proposed in [29]

$$d_k = \begin{cases} -F(v_k), & \text{if } \ k = 0, \\ -\left(1 + \beta_k \frac{F(v_k)^T d_{k-1}}{\|F(v_k)\|^2}\right) F(v_k) + \beta_k d_{k-1}, & \text{if } k \geq 1, \end{cases} \tag{5.3}$$

where $y_{k-1} = F(v_k) - F_{k-1},\ \ w_{k-1} = y_{k-1} + t_{k-1} d_{k-1},\ \ t_{k-1} = 1 + \max\left\{0, \frac{-d_{k-1}^T y_{k-1}}{\|d_{k-1}\|^2}\right\}$.
The parameters are given as follows: $\nu = 1.2,\ \ \rho = 0.55,\ \ \sigma = 10^{-6}$.

We conducted numerical simulations using five test problems, which are presented below. Each problem is addressed in five dimensions with five different initial points, referred to as #Dim. For each of the four methods, the iteration will be programmed to stop if any of the following conditions are satisfied:
(i) when $\|F(v_k)\| \leq 10^{-6}$, (ii) when $\|F(z_k)\| \leq 10^{-6}$, (iv) when the iterations count exceeds 1000.
The term #Itr represents the number of iterations, #Fev denotes the count of function evaluations, #Tm indicates the recorded CPU time in seconds (s), and #Nrm refers to the norm of the residual at the point of termination. The initial points are given as follows:
$v1 = (10, 10, \cdots, 10)^T,\ v2 = (0.1, 0.1, \cdots, 0.1)^T,\ v3 = \left(0, \frac{1}{2}, \frac{2}{3}, ..., 1 - \frac{1}{n}\right)^T,\ v4 = \left(1, \frac{1}{2}, \cdots, \frac{1}{n}\right)^T$,
$v5 = (5, 5, \cdots, 5)^T$.
The dimensions used for the numerical simulation are defined at n = 1000, 5000, 10000, 50000, and 100000. We utilize the following benchmark problem for our numerical simulation:
Problem 1. [32]

$$F_i(v) = 2v_i - \sin(v_i), \quad i = 1, 2, \ldots, n,$$

where the set $\mathcal{D} = \mathbb{R}_+^n$.
Problem 2. Modified exponential function [16]

$$F_1(v) = 2.5v_1 - e^{\frac{\cos(v_1 + v_2)}{n+1}}, \tag{5.4}$$

$$
\begin{aligned}
F_i(v) &= 2.5v_i - e^{\frac{\cos(v_{i-1}+v_i+v_{i+1})}{n+1}}, \quad i = 2, 3, \ldots, n-1,\\
F_n(v) &= 2.5v_n - e^{\frac{\cos(v_{n-1}+v_n)}{n+1}},
\end{aligned}
$$

where the set $\mathcal{D} = \mathbb{R}^n_+$.
Problem 3. [30]

$$
\begin{aligned}
F_1(v) &= 2v_1 + \sin(v_1) - 1,\\
F_i(v) &= 2v_{i-1} + 2v_i + 2\sin(v_i) - 1, \quad i = 2, 3, \ldots, n-1,\\
F_n(v) &= 2v_n + \sin(v_n) - 1,
\end{aligned}
\tag{5.5}
$$

where the set $\mathcal{D} = \mathbb{R}^n_+$.
Problem 4. [32]

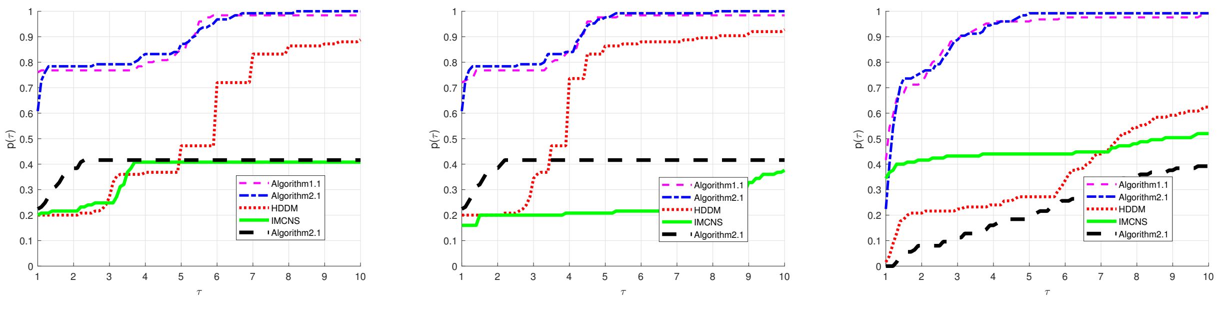
$$
\begin{aligned}
F_1(v) &= 2v_1 + \sin(v_1) - 1,\\
F_i(v) &= 2v_{i-1} + e^{\sin(v_i)} + 2v_i - 1, \quad i = 2, 3, \ldots, n-1,\\
F_n(v) &= 2v_n + e^{\sin(v_n)} - 1,
\end{aligned}
\tag{5.6}
$$

where the set $\mathcal{D} = \mathbb{R}^n_+$.
Problem 5. Modified exponential function [16]

$$
\begin{aligned}
F_1(v) &= v_1 - 2.5e^{\frac{\cos(v_1+v_2)}{n+1}},\\
F_i(v) &= v_i - 2.5e^{\frac{\cos(v_{i-1}+v_i+v_{i+1})}{n+1}}, \quad i = 2, 3, \ldots, n-1,\\
F_n(v) &= v_n - 2.5e^{\frac{\cos(v_{n-1}+v_n)}{n+1}},
\end{aligned}
\tag{5.7}
$$

where the set $\mathcal{D} = \mathbb{R}^n_+$.

The numerical information presented in Tables 1–5 is depicted graphically in Figure 1 using the profile performance presented in [19]. The performance profiles offer a detailed comparison of the methods, considering several aspects such as the #Itr, #Fev, and the #Tm.



(a) Performance profile of #Itr (b) Performance profile of #Fev (c) Performance profile of #Tm

Figure 1. Performance profiles comparing methods with respect to #Itr, #Fev, and #Tm.

Table 1. Numerical results for Algorithm 1.1, Algorithm 2.1, HDDM, IMCNS, and Algorithm 2.1 on Problem 1.

		Algorithm1.1				Algorithm2.1				HDDM				IMCNS				Algorithm2.1			
#Dim	InPoint	#Itr	#Fev	#Tm	#Nrm	#Itr	#Fev	#Tm	#Nrm	#Itr	#Fev	#Tm	#Nrm	#Itr	#Fev	#Tm	#Nrm	#Itr	#Fev	#Tm	#Nrm
	v1	1	2	0.006166	0.00000	1	2	0.008434	0.00000	6	8	0.019558	4.58E-07	22	51	0.013782	6.77E-06	117	118	0.499901	0.00000
	v2	1	2	0.002074	0.00000	1	2	0.002198	0.00000	4	6	0.006089	7.13E-07	18	43	0.0076	8.28E-06	119	120	0.354466	0.00000
1000	v3	1	2	0.002438	0.00000	1	2	0.002744	0.00000	5	7	0.00499	9.14E-07	22	51	0.006343	5.81E-06	106	107	0.38219	0.00000
	v4	6	7	0.008342	0.00000	4	5	0.010812	0.00000	36	38	0.032085	5.13E-08	1001	51360	1.9542	4.14E-05	102	103	0.343133	0.00000
	v5	1	2	0.007486	0.00000	1	2	0.009686	0.00000	5	7	0.008151	6.94E-07	23	54	0.009453	5.28E-06	117	118	0.351781	0.00000
	v1	1	2	0.003766	0.00000	1	2	0.002933	0.00000	7	9	0.017736	5.17E-08	23	58	0.018227	5.33E-06	92	93	1.479821	0.00000
	v2	1	2	0.002237	0.00000	1	2	0.002739	0.00000	5	7	0.009502	8.06E-08	19	50	0.01793	6.52E-06	94	95	1.469389	0.00000
5000	v3	1	2	0.002094	0.00000	1	2	0.002073	0.00000	6	8	0.013516	9.75E-08	23	58	0.019881	4.60E-06	85	86	1.331272	0.00000
	v4	6	7	0.010291	0.00000	4	5	0.008456	0.00000	39	41	0.114471	5.45E-07	1001	46133	8.1783	6.06E-05	82	83	1.354045	0.00000
	v5	1	2	0.002866	0.00000	1	2	0.002747	0.00000	6	8	0.01583	7.84E-08	24	61	0.026484	4.16E-06	92	93	1.468168	0.00000
	v1	1	2	0.004556	0.00000	1	2	0.006187	0.00000	7	9	0.03745	7.32E-08	23	58	0.045886	7.54E-06	84	85	2.335463	0.00000
	v2	1	2	0.00301	0.00000	1	2	0.00326	0.00000	5	7	0.017632	1.14E-07	19	50	0.038854	9.22E-06	86	87	2.33452	0.00000
10000	v3	1	2	0.003012	0.00000	1	2	0.003313	0.00000	6	8	0.022803	1.37E-07	23	58	0.043171	6.51E-06	79	80	2.250197	0.00000
	v4	6	7	0.022052	0.00000	5	6	0.021523	0.00000	41	43	0.263995	7.74E-08	1001	50826	17.9945	5.34E-05	76	77	2.341689	0.00000
	v5	1	2	0.004483	0.00000	1	2	0.004616	0.00000	6	8	0.032324	1.11E-07	24	61	0.034976	5.89E-06	85	86	2.422253	0.00000
	v1	1	2	0.022432	0.00000	1	2	0.021275	0.00000	7	9	0.135817	1.64E-07	24	66	0.15816	2.61E-06	71	72	8.740054	0.00000
	v2	1	2	0.010358	0.00000	1	2	0.013518	0.00000	5	7	0.101072	2.55E-07	20	58	0.13093	3.19E-06	72	73	9.029761	0.00000
50000	v3	1	2	0.011693	0.00000	1	2	0.015564	0.00000	6	8	0.108821	3.03E-07	24	66	0.1461	2.25E-06	68	69	8.639977	0.00000
	v4	6	7	0.09318	0.00000	4	5	0.072984	0.00000	43	45	1.146869	9.89E-08	1001	50485	93.2754	8.38E-05	64	65	8.457395	0.00000
	v5	1	2	0.023493	0.00000	1	2	0.020009	0.00000	6	8	0.135844	2.48E-07	25	69	0.25003	2.03E-06	71	72	8.389668	0.00000
	v1	1	2	0.044198	0.00000	1	2	0.036978	0.00000	7	9	0.277294	2.31E-07	24	66	0.43272	3.68E-06	66	67	13.57739	0.00000
	v2	1	2	0.024242	0.00000	1	2	0.023611	0.00000	5	7	0.172442	3.6E-07	20	58	0.33635	4.51E-06	68	69	13.65131	0.00000
100000	v3	1	2	0.030013	0.00000	1	2	0.022153	0.00000	6	8	0.217865	4.29E-07	24	66	0.39192	3.18E-06	64	65	13.57811	0.00000
	v4	6	7	0.17624	0.00000	4	5	0.145743	0.00000	49	51	2.416925	8.98E-08	1001	51013	189.3874	7.03E-05	60	61	13.30606	0.00000
	v5	1	2	0.047769	0.00000	1	2	0.043085	0.00000	6	8	0.249645	3.51E-07	25	69	0.43158	2.88E-06	66	67	13.70236	0.00000

Table 2. Numerical results for Algorithm 1.1, Algorithm 2.1, HDDM, IMCNS, and Algorithm 2.1 on Problem 2.

		Algorithm1.1				Algorithm2.1				HDDM				IMCNS				Algorithm2.1			
#Dim	InPoint	#Itr	#Fev	#Tm	#Nrm	#Itr	#Fev	#Tm	#Nrm	#Itr	#Fev	#Tm	#Nrm	#Itr	#Fev	#Tm	#Nrm	#Itr	#Fev	#Tm	#Nrm
	v1	1	2	0.00969	0.00000	1	2	0.001651	0.00000	6	8	0.009949	1.37E-07	1	2	0.003002	0.00000	1	2	0.00871	0.00000
	v2	1	2	0.000922	0.00000	1	2	0.000927	0.00000	5	7	0.006181	9.8E-07	1	2	0.001395	0.00000	1	2	0.001191	0.00000
1000	v3	1	2	0.001109	0.00000	1	2	0.000971	0.00000	5	7	0.007613	6.46E-07	1	2	0.00125	0.00000	1	2	0.001301	0.00000
	v4	1	2	0.001349	0.00000	1	2	0.001186	0.00000	7	9	0.007879	5.23E-07	1	2	0.001041	0.00000	1	2	0.003863	0.00000
	v5	1	2	0.001529	0.00000	1	2	0.001482	0.00000	5	7	0.006125	8.53E-07	1	3	0.001426	0.00000	1	2	0.0027	0.00000
	v1	1	2	0.002553	0.00000	1	2	0.002473	0.00000	6	8	0.02069	3.1E-07	1	2	0.001745	0.00000	1	2	0.005423	0.00000
	v2	1	2	0.001784	0.00000	1	2	0.001735	0.00000	6	8	0.018214	1.11E-07	1	2	0.001813	0.00000	1	2	0.002392	0.00000
5000	v3	1	2	0.001875	0.00000	1	2	0.001851	0.00000	6	8	0.021287	7.32E-08	1	2	0.001982	0.00000	1	2	0.002754	0.00000
	v4	1	2	0.002803	0.00000	1	2	0.002516	0.00000	8	10	0.028834	3.09E-07	1	2	0.002026	0.00000	1	2	0.008161	0.00000
	v5	1	2	0.002497	0.00000	1	2	0.002553	0.00000	6	8	0.019303	9.71E-08	1	3	0.002291	0.00000	1	2	0.005922	0.00000
	v1	1	2	0.004121	0.00000	1	2	0.003842	0.00000	6	8	0.047937	4.39E-07	1	2	0.002722	0.00000	1	2	0.008454	0.00000
	v2	1	2	0.003149	0.00000	1	2	0.002913	0.00000	6	8	0.051157	1.58E-07	1	2	0.002512	0.00000	1	2	0.007364	0.00000
10000	v3	1	2	0.002969	0.00000	1	2	0.003154	0.00000	6	8	0.045936	1.04E-07	1	2	0.003035	0.00000	1	2	0.004729	0.00000
	v4	1	2	0.004555	0.00000	1	2	0.004809	0.00000	8	10	0.069224	8.8E-07	1	2	0.004109	0.00000	1	2	0.01051	0.00000
	v5	1	2	0.004926	0.00000	1	2	0.004584	0.00000	6	8	0.040611	1.37E-07	1	3	0.00327	0.00000	1	2	0.009313	0.00000
	v1	1	2	0.022759	0.00000	1	2	0.02042	0.00000	6	8	0.220374	9.81E-07	1	2	0.011758	0.00000	1	2	0.041613	0.00000
	v2	1	2	0.016907	0.00000	1	2	0.014215	0.00000	6	8	0.21251	3.53E-07	1	2	0.01083	0.00000	1	2	0.019858	0.00000
50000	v3	1	2	0.013159	0.00000	1	2	0.013306	0.00000	6	8	0.211856	2.31E-07	1	2	0.015926	0.00000	1	2	0.017495	0.00000
	v4	1	2	0.020197	0.00000	1	2	0.022702	0.00000	9	11	0.312411	5E-07	1	2	0.008898	0.00000	1	2	0.062814	0.00000
	v5	1	2	0.025214	0.00000	1	2	0.02247	0.00000	6	8	0.203976	3.07E-07	1	3	0.011695	0.00000	1	2	0.044614	0.00000
	v1	1	2	0.043596	0.00000	1	2	0.038835	0.00000	7	9	0.542325	7.01E-08	1	2	0.021398	0.00000	1	2	0.081948	0.00000
	v2	1	2	0.023792	0.00000	1	2	0.029093	0.00000	6	8	0.448249	4.99E-07	1	2	0.023326	0.00000	1	2	0.029014	0.00000
100000	v3	1	2	0.022065	0.00000	1	2	0.02653	0.00000	6	8	0.474902	3.27E-07	1	2	0.02006	0.00000	1	2	0.034307	0.00000
	v4	1	2	0.038177	0.00000	1	2	0.051941	0.00000	10	12	0.714825	7.15E-08	1	2	0.018648	0.00000	1	2	0.108394	0.00000
	v5	1	2	0.043041	0.00000	1	2	0.040836	0.00000	6	8	0.459372	4.35E-07	1	3	0.020092	0.00000	1	2	0.082799	0.00000

Table 3. Numerical results for **Algorithm1.1**, **Algorithm2.1**, **HDDM**, **IMCNS**, and **Algorithm2.1** on Problem 3.

		Algorithm1.1				Algorithm2.1				HDDM				IMCNS				Algorithm2.1			
#Dim	InPoint	#Itr	#Fev	#Tm	#Nrm	#Itr	#Fev	#Tm	#Nrm	#Itr	#Fev	#Tm	#Nrm	#Itr	#Fev	#Tm	#Nrm	#Itr	#Fev	#Tm	#Nrm
	v1	35	37	0.044308	9.49E-07	39	41	0.060585	6.88E-07	107	109	0.206489	9.71E-07	1001	36042	2.8283	2.28E-05	76	77	0.227184	0.00000
	v2	34	36	0.02667	6.82E-07	37	39	0.035026	7.76E-07	103	105	0.157757	8.87E-07	1001	36450	2.8604	2.70E-05	78	79	0.215168	0.00000
1000	v3	35	37	0.026385	7.81E-07	39	41	0.031423	6.86E-07	113	115	0.176229	9.54E-07	1001	35332	2.6058	2.85E-05	77	78	0.234726	0.00000
	v4	357	359	0.248569	9.42E-07	65	67	0.04945	8.7E-07	607	609	0.890921	9.8E-07	1001	34388	2.5725	3.16E-05	71	72	0.213044	0.00000
	v5	36	38	0.029262	7.36E-07	38	40	0.032388	9.17E-07	110	112	0.167777	9.26E-07	1001	35881	2.6578	2.30E-05	76	77	0.212342	0.00000
	v1	38	40	0.094905	9.88E-07	40	42	0.127569	7.93E-07	110	112	0.621908	9.86E-07	1001	35045	12.5501	2.00E-05	64	65	1.021338	0.00000
	v2	35	37	0.088965	7.71E-07	36	38	0.09656	8.8E-07	109	111	0.662273	9.79E-07	1001	35125	13.6768	1.98E-05	66	67	0.945377	0.00000
5000	v3	36	38	0.096035	8.73E-07	39	41	0.111143	9.24E-07	114	116	0.6766	9.96E-07	1001	35252	12.9289	2.35E-05	66	67	1.0191	0.00000
	v4	714	716	1.829617	7.75E-07	65	67	0.187287	7.7E-07	1001	1002	5.613044	1653.292	1001	35273	12.9317	1.06E-04	60	61	0.93446	0.00000
	v5	37	39	0.108072	6.82E-07	48	50	0.13385	7.84E-07	113	115	0.642941	9.62E-07	1001	34929	12.6975	1.94E-05	64	65	1.041421	0.00000
	v1	37	39	0.249608	9.23E-07	42	44	0.287385	6.63E-07	111	113	1.502448	9.55E-07	1001	34940	21.6131	1.99E-05	60	61	1.734221	0.00000
	v2	36	38	0.228889	7.46E-07	37	39	0.234057	9.1E-07	108	110	1.406204	9.07E-07	1001	35062	21.4469	2.27E-05	63	64	1.768611	0.00000
10000	v3	52	54	0.324246	8.24E-07	52	54	0.325156	9.43E-07	114	116	1.485373	9.84E-07	1001	35138	21.369	2.03E-05	62	63	1.799623	0.00000
	v4	36	38	0.236142	7.19E-07	40	42	0.297641	7.02E-07	1001	1002	12.56563	2552.802	1001	36029	21.7949	1.38E-04	56	57	1.686394	0.00000
	v5	38	40	0.255969	9.22E-07	41	43	0.264666	7.49E-07	114	116	1.512958	8.85E-07	1001	35801	21.8349	2.33E-05	61	62	1.715508	0.00000
	v1	40	42	0.989062	7.53E-07	44	46	1.212873	8.28E-07	117	119	8.717152	8.87E-07	1001	34953	88.0366	2.13E-05	54	55	6.823937	0.00000
	v2	35	37	0.930503	6.89E-07	38	40	1.063361	8.76E-07	110	112	7.248286	9.37E-07	1001	35293	91.7405	2.60E-05	56	57	6.982676	0.00000
50000	v3	38	40	1.026036	7.16E-07	46	48	1.305871	8.76E-07	115	117	8.045671	9.7E-07	1001	34279	86.5934	1.76E-05	54	55	6.93878	0.00000
	v4	39	41	1.040771	7.05E-07	42	44	1.126637	9.73E-07	1001	1002	55.91245	6071.841	1001	36560	94.6624	1.52E-04	50	51	6.42041	0.00000
	v5	40	42	1.063314	6.44E-07	42	44	1.191891	6.63E-07	114	116	6.573694	9.77E-07	1001	35068	90.9093	2.32E-05	54	55	6.905323	0.00000
	v1	39	41	2.241317	6.42E-07	48	50	2.792107	8.55E-07	119	121	15.21714	9.01E-07	1001	35769	178.9046	2.21E-05	51	52	12.97943	0.00000
	v2	43	45	2.263902	7.16E-07	40	42	2.417634	7.02E-07	109	111	16.54639	9.88E-07	1001	35454	127.6801	2.62E-05	52	53	13.19645	0.00000
100000	v3	38	40	2.095556	7.24E-07	41	43	2.472406	9.31E-07	116	118	15.09289	9.39E-07	1001	34482	125.1482	2.01E-05	52	53	13.63579	0.00000
	v4	39	41	2.20373	6.6E-07	42	44	2.531499	7.61E-07	1001	1002	131.904	8658.251	1001	34560	125.5418	0.000188	47	48	12.24806	0.00000
	v5	44	46	2.51947	6.68E-07	46	48	2.752069	6.95E-07	114	116	14.64073	9.97E-07	1001	34904	126.8373	2.46E-05	51	52	12.86406	0.00000

Table 4. Numerical results for Algorithm 1.1, Algorithm 2.1, HDDM, IMCNS, and Algorithm 2.1 on Problem 4.

		Algorithm1.1				Algorithm2.1				HDDM				IMCNS				Algorithm2.1			
#Dim	InPoint	#Itr	#Fev	#Tm	#Nrm	#Itr	#Fev	#Tm	#Nrm	#Itr	#Fev	#Tm	#Nrm	#Itr	#Fev	#Tm	#Nrm	#Itr	#Fev	#Tm	#Nrm
	v1	1	2	0.002157	0.00000	1	2	0.002494	0.00000	8	10	0.024122	6.78E-07	47	187	0.030633	7.77E-06	73	74	0.233859	0.00000
	v2	1	2	0.001644	0.00000	1	2	0.001899	0.00000	6	8	0.011757	6.3E-07	33	132	0.011901	8.40E-06	74	75	0.226813	0.00000
1000	v3	1	2	0.001854	0.00000	1	2	0.001995	0.00000	6	8	0.012736	9.53E-08	37	148	0.013451	8.83E-06	74	75	0.221614	0.00000
	v4	335	336	0.248841	9.04E-07	579	580	0.413484	8.85E-07	712	714	1.258217	9.82E-07	74	275	0.024318	8.70E-06	71	72	0.220821	0.00000
	v5	1	2	0.001831	0.00000	1	2	0.001896	0.00000	8	10	0.0225	2.58E-07	46	183	0.016027	8.44E-06	73	74	0.232244	0.00000
	v1	1	2	0.004652	0.00000	1	2	0.004384	0.00000	7	9	0.055525	1.26E-07	48	191	0.049683	8.75E-06	62	63	0.999316	0.00000
	v2	1	2	0.004512	0.00000	1	2	0.004633	0.00000	5	7	0.037822	4.05E-07	34	136	0.047577	8.68E-06	64	65	1.017079	0.00000
5000	v3	1	2	0.005552	0.00000	1	2	0.005159	0.00000	6	8	0.041622	5.74E-08	38	152	0.043288	8.18E-06	63	64	1.009688	0.00000
	v4	794	795	2.180379	8.46E-07	155	156	0.423409	8.9E-07	1001	1002	6.764855	1671.892	80	319	0.11379	8.49E-06	60	61	0.950891	0.00000
	v5	1	2	0.004033	0.00000	1	2	0.004605	0.00000	7	9	0.052584	4.4E-07	48	191	0.059431	7.24E-06	62	63	0.975638	0.00000
	v1	1	2	0.00764	0.00000	1	2	0.011123	0.00000	7	9	0.116817	7.24E-08	49	195	0.1447	7.80E-06	59	60	1.767688	0.00000
	v2	1	2	0.012021	0.00000	1	2	0.007565	0.00000	5	7	0.09025	2.74E-07	35	140	0.10633	7.08E-06	60	61	1.789289	0.00000
10000	v3	1	2	0.01036	0.00000	1	2	0.008325	0.00000	6	8	0.090915	6.74E-08	38	152	0.083626	9.58E-06	59	60	1.758273	0.00000
	v4	1	2	0.008227	0.00000	1	2	0.008272	0.00000	1001	1002	14.70162	2216.919	82	327	0.22972	9.07E-06	56	57	1.724062	0.00000
	v5	1	2	0.007186	0.00000	1	2	0.008998	0.00000	7	9	0.110243	9.43E-08	48	191	0.1434	8.13E-06	59	60	1.767343	0.00000
	v1	1	2	0.046056	0.00000	1	2	0.042358	0.00000	7	9	0.492631	6.88E-07	50	199	0.61036	8.98E-06	52	53	6.913015	0.00000
	v2	1	2	0.043284	0.00000	1	2	0.043422	0.00000	5	7	0.366344	3.08E-07	36	144	0.43473	7.17E-06	53	54	6.840162	0.00000
50000	v3	1	2	0.047489	0.00000	1	2	0.053294	0.00000	7	8	0.433708	5.01E-07	39	156	0.49165	9.59E-06	52	53	6.951824	0.00000
	v4	1	2	0.045638	0.00000	1	2	0.04921	0.00000	1001	1002	66.49921	4043.215	111	409	1.3423	8.56E-06	50	51	6.687791	0.00000
	v5	1	2	0.043584	0.00000	1	2	0.035202	0.00000	6	8	0.40438	5.82E-07	49	195	0.62752	7.50E-06	52	53	6.7141	0.00000
	v1	1	2	0.100238	0.00000	1	2	0.087391	0.00000	7	9	1.079809	2.63E-07	51	203	1.3042	7.73E-06	50	51	13.27852	0.00000
	v2	1	2	0.07126	0.00000	1	2	0.080186	0.00000	5	7	0.760032	3.97E-07	36	144	0.92692	8.55E-06	50	51	13.04228	0.00000
100000	v3	1	2	0.096499	0.00000	1	2	0.092123	0.00000	7	8	0.938015	7.1E-07	40	160	1.0162	7.77E-06	50	51	13.13665	0.00000
	v4	1	2	0.099263	0.00000	1	2	0.098434	0.00000	1001	1002	141.9953	6137.676	124	437	2.8913	7.61E-06	47	48	12.53073	0.00000
	v5	1	2	0.088597	0.00000	1	2	0.095048	0.00000	6	8	0.905841	4.86E-07	49	195	1.1617	8.65E-06	50	51	13.18988	0.00000

Table 5. Numerical results for Algorithm 1.1, Algorithm 2.1, HDDM, IMCNS, and Algorithm 2.1 on Problem 5.

		Algorithm1.1				Algorithm2.1				HDDM				IMCNS				Algorithm2.1			
#Dim	InPoint	#Itr	#Fev	#Tm	#Nrm	#Itr	#Fev	#Tm	#Nrm	#Itr	#Fev	#Tm	#Nrm	#Itr	#Fev	#Tm	#Nrm	#Itr	#Fev	#Tm	#Nrm
	v1	34	36	0.042718	8.53E-07	33	35	0.044959	6.72E-07	7	9	0.018781	5.7E-08	23	74	0.016648	1.63E-06	523	525	0.754134	9.66E-08
	v2	32	34	0.023802	7.31E-07	36	38	0.028072	9.35E-07	6	8	0.011049	1.17E-07	21	68	0.009376	2.03E-06	390	392	0.534391	9.77E-08
1000	v3	25	27	0.018915	7.86E-07	28	30	0.023124	6.6E-07	5	7	0.008806	2.25E-07	1001	49355	2.9821	2.02E-05	380	382	0.519347	1E-07
	v4	34	36	0.025477	8.89E-07	32	34	0.026069	9.41E-07	8	10	0.013502	1.95E-07	29	92	0.011172	1.15E-06	732	734	1.025125	9.9E-08
	v5	34	36	0.025757	8.52E-07	33	35	0.026518	6.74E-07	6	8	0.010855	4.63E-07	22	71	0.009489	1.52E-06	479	481	0.655147	9.97E-08
	v1	36	38	0.10483	5.86E-07	35	37	0.113058	6.29E-07	7	9	0.040183	1.28E-07	23	74	0.037628	3.66E-06	468	470	3.014615	9.79E-08
	v2	33	35	0.106074	9.59E-07	38	40	0.105186	7.66E-07	6	8	0.043995	2.62E-07	21	68	0.03717	4.54E-06	403	405	2.654883	9.65E-08
5000	v3	26	28	0.076326	8.89E-07	29	31	0.086585	7.96E-07	5	7	0.03642	4.46E-07	13	152	0.041748	1.18E-06	343	345	2.104934	9.24E-08
	v4	36	38	0.107452	5.92E-07	35	37	0.114484	6.12E-07	9	11	0.055997	1.2E-07	31	98	0.044401	2.87E-06	729	731	4.700332	9.74E-08
	v5	36	38	0.107727	5.86E-07	35	37	0.117863	6.29E-07	7	9	0.053464	5.61E-08	22	71	0.039116	3.41E-06	447	449	2.919419	9.84E-08
	v1	36	38	0.236208	8.23E-07	35	37	0.245314	9.01E-07	7	9	0.106671	1.81E-07	23	74	0.076395	5.17E-06	466	468	5.300916	9.77E-08
	v2	34	36	0.242057	7.79E-07	39	41	0.316571	6.6E-07	6	8	0.086534	3.7E-07	21	68	0.066497	6.43E-06	373	375	4.486527	9.77E-08
10000	v3	27	29	0.192606	7.06E-07	30	32	0.253971	6.74E-07	5	7	0.077479	6.18E-07	12	131	0.069368	3.95E-06	320	322	3.79984	9.99E-08
	v4	36	38	0.252059	8.27E-07	35	37	0.277051	8.89E-07	9	11	0.124887	3.41E-07	32	101	0.097551	3.82E-06	746	748	8.909506	9.85E-08
	v5	36	38	0.223427	8.23E-07	35	37	0.291176	9.01E-07	7	9	0.089108	7.93E-08	22	71	0.073987	4.82E-06	422	424	5.013702	9.5E-08
	v1	38	40	1.164742	5.99E-07	37	39	1.628166	7.58E-07	7	9	0.451794	4.04E-07	24	84	0.33226	4.52E-06	431	433	21.41955	9.96E-08
	v2	35	37	1.05166	9.99E-07	40	42	1.350642	8.99E-07	6	8	0.479379	8.27E-07	22	78	0.32613	5.62E-06	358	360	18.29471	9.72E-08
50000	v3	28	30	0.846351	8.89E-07	31	33	1.024166	9.04E-07	6	8	0.44856	7.35E-08	13	153	0.39727	3.27E-06	185	187	9.566051	9.48E-08
	v4	38	40	1.187345	6E-07	37	39	1.19313	7.56E-07	10	12	0.618293	2.07E-07	34	107	0.50737	9.48E-06	722	724	37.72461	9.86E-08
	v5	38	40	1.142577	5.99E-07	37	39	1.252398	7.58E-07	7	9	0.440263	1.77E-07	23	81	0.33812	4.22E-06	400	402	20.95809	9.96E-08
	v1	38	40	2.348758	8.46E-07	38	40	2.526588	6.55E-07	7	9	1.055504	5.71E-07	24	84	0.67436	6.40E-06	404	406	44.99078	9.95E-08
	v2	36	38	2.335585	8.09E-07	41	43	2.902507	7.75E-07	7	9	0.97459	6.33E-08	22	78	0.62575	7.95E-06	312	314	31.99822	9.83E-08
100000	v3	29	31	1.937384	7.18E-07	32	34	2.264695	7.78E-07	6	8	0.906195	1.04E-07	13	153	0.84244	4.82E-06	295	297	29.12272	9.8E-08
	v4	38	40	2.469645	8.47E-07	38	40	2.890196	6.54E-07	10	12	1.328306	5.85E-07	36	120	1.0056	4.94E-06	736	738	71.72611	9.77E-08
	v5	38	40	2.536244	8.46E-07	38	40	2.880411	6.55E-07	7	9	1.02504	2.51E-07	23	81	0.62838	5.96E-06	380	382	35.69513	9.67E-08

To summarize the findings, Algorithm 2.1 outperforms the other three methods: Algorithm 1.1, HDDM, and IMCNS in Figure 1 above. In most test scenarios, it consistently shows superior performance in terms of iterations (#Itr), time taken (#Tm), and function evaluations (#Fev). Overall, this Figure showcased that Algorithms 1.1 and 2.1 are the most effective and reliable methods for solving optimization problems among those evaluated. This is highlighted by the higher curve in the Algorithms 1.1 and 2.1 profiles, indicating that they require (#Itr), (#Fev), and (#Tm) to find a solution across the five test problems.

The comprehensive results of our numerical experiments are presented in Tables 1–5. The methods analyzed in these tables include Algorithms 1.1 and 2.1, HDDM, and IMCNS, all of which are integral parts of our numerical simulation. We evaluate and record the following criteria: the number of iterations (#Itr), the number of evaluations (#Fev), the total time taken (#Tm), and the normalization metric (#Nrm).

In the following sections, we will discuss these metrics in detail and explain their significance. The consistent performance of the numerical simulations indicates that the proposed Algorithms 1.1 and 2.1 remain effective throughout Tables 1–5. However, in Table 3, the IMCNS method failed to converge to a solution, and the performance of Algorithm 2.1 in Table 3 was subpar. On the other hand, the performance of HDDM was consistently strong throughout the experiments. In fact, it is the only method that follows the efficiency demonstrated by the proposed Algorithms 1.1 and 2.1.

Table 6 provides a consolidated summary of the results from Tables 1–5, highlighting the relative performance of the five methods in terms of the number of iterations, function evaluations, processor time, and residual norms. The results clearly indicate that the proposed Algorithm 1.1 outperforms all other methods, achieving the lowest values across all performance measures. Proposed Algorithm 2.1 ranks second, consistently demonstrating the next-best performance in terms of iterations, function evaluations, processor time, and residual norms. The HDDM method follows as the third-best performer, showing moderate efficiency relative to the first two algorithms. By contrast, the IMCNS method exhibits the weakest performance among the five, as reflected by its significantly higher computational cost and poorer accuracy.

Table 6. Summary of the test results reported in Tables 1–5.

Methods	#Itr	#Fev	#Tm	#Nrm
Algorithm 1.1	32.032	33.632	0.345056	3.23294E-07
Algorithm 2.1	22.168	23.696	0.364256	3.27362E-07
HDDM	98.816	100.736	4.50992	264.0470989
IMCNS	267.008	9518.4	13.2575	1.45E-05
Algorithm 2.1	132.408	133.608	5.8258	1.95552E-08

5.1. Application of image recovery

Image recovery has seen significant advancements lately, particularly in restoring damaged or degraded images. This study aims to reconstruct the original image by mimicking the degradation process and utilizing methods for solving inverse problems. By incorporating factors such as additive Gaussian noise and applying a Gaussian blur operator, a restoration framework can be established, assuming that the observed image has undergone specific types of degradation.

$$b = Fv + \varepsilon, \tag{5.8}$$

where $v \in \mathbb{R}^{N\times 1}$ denotes the original image, $b \in \mathbb{R}^{N\times 1}$ is the adaptive noise $F \in \mathbb{R}^{N\times N}$, and $\varepsilon \in \mathbb{R}^{N}$ signifies the random additive noise. Due to the severe ill-conditioning of the matrix F, a commonly studied restoration model is formulated as follows:

$$\min_{v} \left\{ \frac{1}{2}\|Fv - b\|_2^2 + \tau\|v\|_1 \right\}, \tag{5.9}$$

where τ is a positive regularization parameter. It is important to note that there exists a well-established equivalence between the convex-constrained nonlinear equation (1.1) and the convex-constrained unconstrained problem represented with (5.9). (For more details, see [1, 3, 27] and the references therein.)

In this study, we analyzed four colored images: Heart (256 x 256), Intestine (256 x 256), Kidney (256 x 256), and Lungs (256 x 256). To simulate degradation, we applied a Gaussian blur with a standard deviation of 10^{-2} alongside a Gaussian noise operator. The objective was to compare the effectiveness of our proposed method, Algorithm 2.1, to other techniques like HDDM [11]. Evaluating various image restoration algorithms is a fascinating field of research. To measure the quality of the restored images, we used four primary metrics: Number of Iterations (Itr), CPU time (Tm), structural similarity index measure (SSIM), and signal-to-noise ratio (SNR), defined as follows:

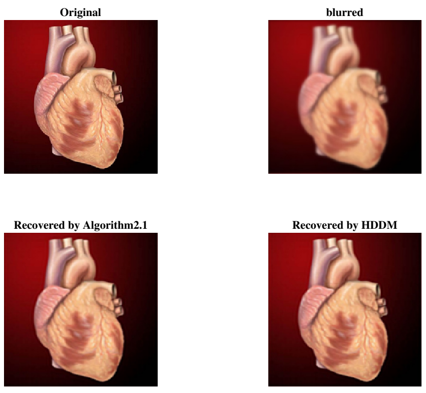
$$SNR = 20 \times \log_{10}\left(\frac{\|\widehat{v}\|}{\|v - \widehat{v}\|}\right),$$

where $\widehat{v}$ and v are the original and recovered images, respectively. The MATLAB code for calculating the SSIM index can be found at `https://www.cns.nyu.edu/~lcv/ssim/`. These metrics help assess how closely the recovered images match the original in terms of noise, visual fidelity, and quantitative analysis [3, 5, 27]. Higher SNR and SSIM values indicate a more effective method. Both algorithms' initial setup and stopping criteria are consistently defined as

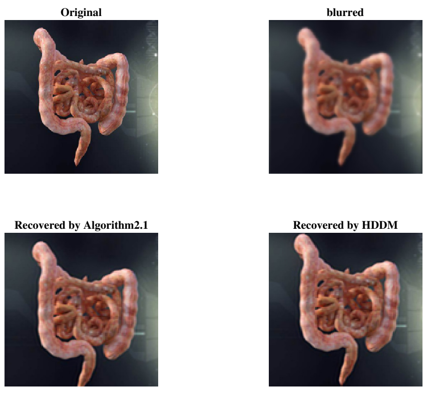
$$\frac{|f(x_k) - f(x_{k-1})|}{|f(x_{k-1})|} < 10^{-4}.$$

The original test images and images recovered through various methods are shown in Figures 2–4 below. We configured the implementation of the Algorithm 2.1 and HDDM method with the parameters set to $\mu = 0.04$, $\sigma = 0.00001$, $w = 1.2$, $(\lambda + 1) = 1.26$, and $\rho = 0.5$. Also, the Gaussian blur is of filter size 4×4. We varied the choice of the regularization and standard deviation parameters across three different experimental settings, as outlined below: Experiment 1: Standard deviation $\sigma = 0.00001$, and regularization parameter $\tau = 0.35$. Experiment 2: Standard deviation $\sigma = 25$, and regularization parameter $\tau = 25$. Experiment 3: Standard deviation $\sigma = 50$, and regularization parameter $\tau = 50$.

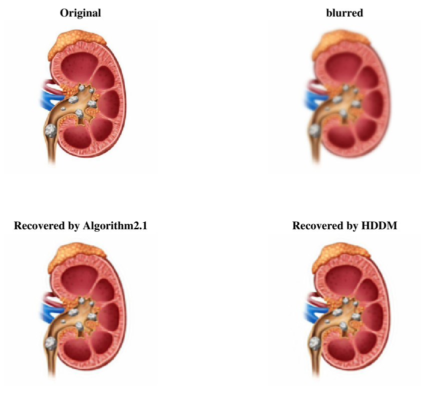
In the context of image restoration, Tables 6–9 indicate that Algorithm 2.1 outperforms the HDDM method. This is evidenced by Algorithm 2.1's fewer iterations (#Itr) and shorter time taken (#Tm), along with its higher #SNR and #SSIM. These factors demonstrate its superior effectiveness in recovering images.



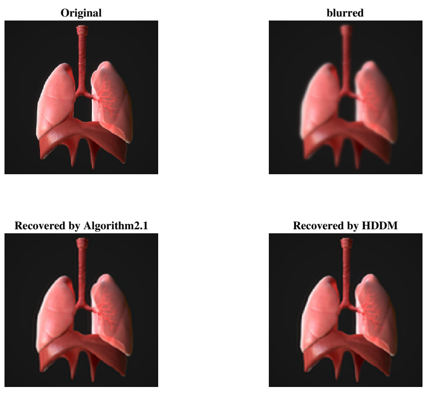
(a) Recovery by Algorithm 2.1 and HDDM



(b) Recovery by Algorithm 2.1 and HDDM



(c) Recovery by Algorithm 2.1 and HDDM

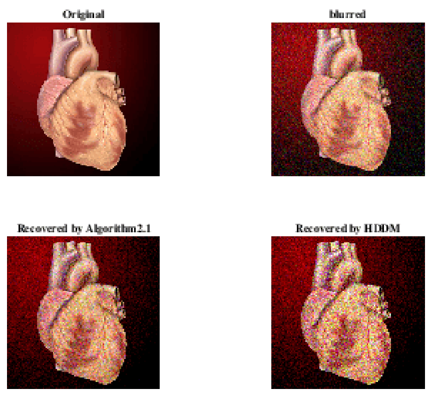


(d) Recovery by Algorithm 2.1 and HDDM

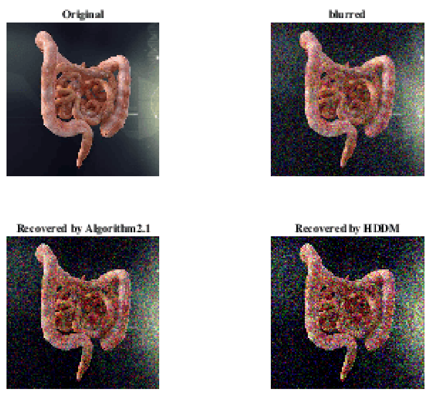
Figure 2. Experiment 1 results using Algorithm 2.1 and HDDM on heart, intestine, kidney, stone, and lung.

Table 7. Numerical simulation results for Experiment 1.

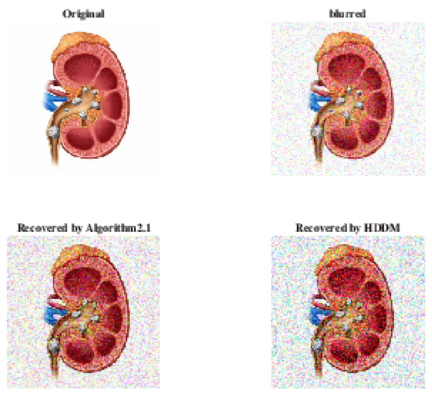
Images	Algorithm 2.1				HDDM			
	#Itr	#Tm	#SNR	#SSIM	#Itr	#Tm	#SNR	#SSIM
Experiment 1								
Heart	20	3.20	21.40	0.94	19	11.06	20.09	0.72
Intestine	37	8.20	21.02	0.92	35	8.19	21.00	0.91
Kidney	15	4.14	21.04	0.94	15	5.42	23.10	0.90
Lungs	18	5.32	22.00	0.97	33	6.77	20.29	0.97



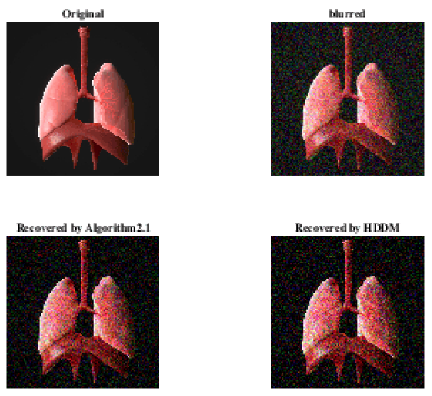
(a) Recovery by Algorithm 2.1 and HDDM



(b) Recovery by Algorithm 2.1 and HDDM



(c) Recovery by Algorithm 2.1 and HDDM

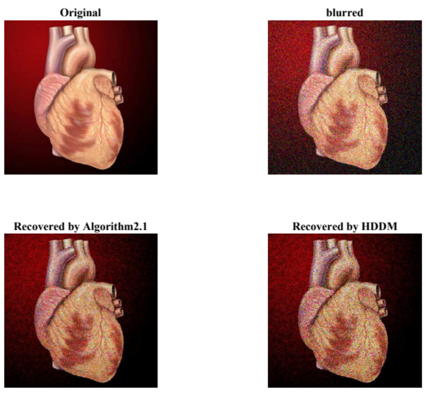


(d) Recovery by Algorithm 2.1 and HDDM

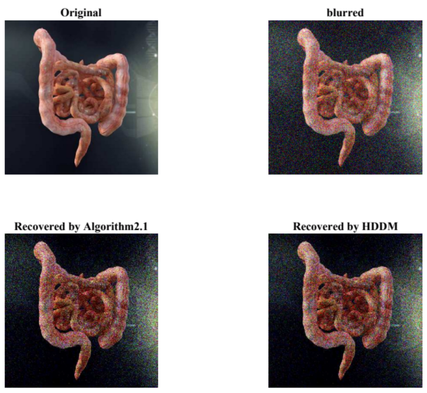
Figure 3. Experiment 2 results using Algorithm 2.1 and HDDM on heart, intestine, kidney, stone, and lung.

Table 8. Numerical simulation results for Experiment 2.

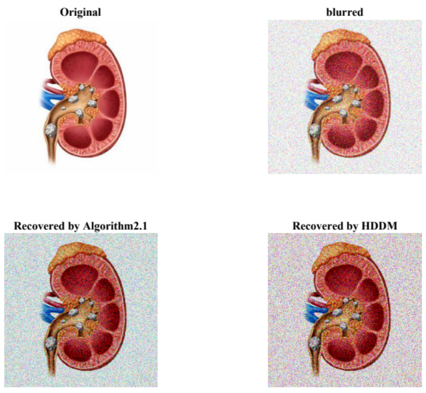
Images	Algorithm 2.1				HDDM			
	#Itr	#Tm	#SNR	#SSIM	#Itr	#Tm	#SNR	#SSIM
Experiment 1								
Heart	20	3.20	21.40	0.94	19	11.06	20.09	0.72
Intestine	37	8.20	21.02	0.92	35	8.19	21.00	0.91
Kidney	15	4.14	21.04	0.94	15	5.42	23.10	0.90
Lungs	18	5.32	22.00	0.97	33	6.77	20.29	0.97



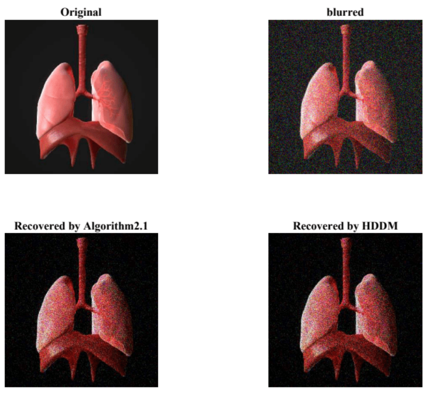
(a) Recovery by Algorithm 2.1 and HDDM



(b) Recovery by Algorithm 2.1 and HDDM



(c) Recovery by Algorithm 2.1 and HDDM



(d) Recovery by Algorithm 2.1 and HDDM

Figure 4. Experiment 3 results using Algorithm 2.1 and HDDM on heart, intestine, kidney, stone, and lung.

Table 9. Numerical simulation results for Experiment 3.

Images	Algorithm 2.1				HDDM			
	#Itr	#Tm	#SNR	#SSIM	#Itr	#Tm	#SNR	#SSIM
Experiment 3								
Heart	22	70.72	5.89	0.24	32	102.05	4.60	0.23
Intestine	48	189.94	2.47	0.05	53	190.41	2.00	0.04
Kidney	17	67.33	11.62	0.07	23	93.16	8.08	0.04
Lungs	28	115.39	4.06	0.07	47	188.20	2.95	0.06

6. Conclusions

This research presents two efficient algorithms for solving constrained monotone nonlinear equations. These techniques were developed by approximating a Jacobian matrix using a diagonal matrix via an acceleration parameter. The second algorithm was derived using the Picard-Mann iterative procedure, which enhances overall numerical convergence compared to existing algorithms. The proposed algorithms satisfy the descent condition, an integral requirement that aids in proving their global convergence. The study also demonstrates the R-linear convergence rate of the proposed method. The study conducted numerical simulations on large-scale problems, with results documented in Tables 1–5. These research findings indicate that Algorithm 2.1 solves all problems with the fewest iterations #Itr and time #Tm, while Algorithm 1.1 demonstrates efficiency in terms of #Fev. Figure 1 illustrates the numerical data trends. Notably, Algorithm 2.1 consistently outperforms the others. Moreover, the proposed algorithm performs excellently in application problems related to image recovery, successfully reconstructing images of the kidney, intestine, and lungs. In future work, the researchers plan to integrate inertial techniques to enhance the robustness of the numerical algorithms.

Author contributions

M. Abdullahi: Investigation, software, writing-original draft, methodology; A. Al-Yaari: Investigation, software, revising, methodology; A. B. Abubakar: Conceptualization, software, writing-review, and editing; A. S. Halilu: Conceptualization, revising, writing-review, and editing; M. A. Muhamed: Methodology, software, writing-review, and editing. All authors have read and approved the final version of the manuscript for publication.

Use of Generative-AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgment

The third author acknowledges with thanks the Department of Mathematics and Applied Mathematics at the Sefako Makgatho Health Sciences University.

Conflict of interest

The authors declare that there is no competing interest regarding the publication of this paper.

Supplementary materials

Details of the results of numerical / application experiments are provided in the following link `https://drive.google.com/drive/folders/1zP5i7h9sAoL1-Kub0K-AoZ_whFNxCurL?usp=sharing`.

References

1. M. Abdullahi, A. B. Abubakar, K. Muangchoo, Modified three-term derivative-free projection method for solving nonlinear monotone equations with application, *Numer. Algorithms,* **95** (2024), 1459–1474. https://doi.org/10.1007/s11075-023-01616-8
2. M. Abdullahi, A. S. Halılu, A. M. Awwal, N. Pakkaranang, On efficient matrix-free method via quasi-Newton approach for solving system of nonlinear equations, *Adv. Theor. Nonlinear Anal. Appl.,* **5** (2021), 568–579. Available from: `https://atnaea.org/index.php/journal/article/view/225`.
3. A. B. Abubakar, A. H. Ibrahim, M. Abdullahi, M. Aphane, J. Chen, A sufficient descent LS-PRP-BFGS-like method for solving nonlinear monotone equations with application to image restoration, *Numer. Algorithms,* **96** (2024), 1423–1464. https://doi.org/10.1007/s11075-023-01690-4
4. J. Barzilai, J. M. Borwein, Two-point step size gradient methods, *IMA J. Numer. Anal.,* **8** (1988), 141–148. https://doi.org/10.1093/imanum/8.1.141
5. A. C. Bovik, *Handbook of image and video processing*, 2Eds., Academic Press, New York, 2010. https://doi.org/10.1016/B978-0-12-374648-1.00069-2
6. R. H. Byrd, P. Lu, J. Nocedal, C. Zhu, A limited memory algorithm for bound constrained optimization, *SIAM J. Sci. Comput.,* **16** (1995), 1190–1208. https://doi.org/10.1137/0916069
7. N. Djuranovic-Milicic, M. Gardaševic-Filipovic, A multistep curve search algorithm in nonlinear optimization: Nondifferentiable convex case, *Facta Univer.-Ser. Math.,* **25** (2010), 11–24. https://doi.org/:10.2298/YJOR0801047D
8. N. Duranovic-Milicic, A multi-step curve search algorithm in nonlinear optimization, *Yugoslav J. Oper. Res.,* **18** (2008), 1.
9. N. Duranovic-Milicic, A multi-step curve search algorithm in nonlinear optimization, *Yugoslav J. Oper. Res.,***18** (2016), 1. https://doi.org/:10.2298/YJOR0801047D
10. A. S. Halilu, A. Majumder, M. Y. Waziri, K. Ahmed, Signal recovery with convex constrained nonlinear monotone equations through conjugate gradient hybrid approach, *Math. Comput. Simul.,* **187** (2021), 520–539. https://doi.org/10.1016/j.matcom.2021.03.020
11. A. S. Halilu, A. Majumder, M. Y. Waziri, A. M. Awwal, K. Ahmed, On solving double direction methods for convex constrained monotone nonlinear equations with image restoration, *Comput. Appl. Math.,* **40** (2021), 1–27. https://doi.org/10.1007/s40314-021-01624-1
12. A. S. Halilu, M. Y. Waziri, An improved derivative-free method via double direction approach for solving systems of nonlinear equations, *J. Ramanujan Math. Soc.,* **33** (2018), 75–89. https://doi.org/10.37727/jrms/2018/v33i1p75
13. S. H. Khan, A Picard–Mann hybrid iterative process, *J. Fix. Point Theory A.,* **2013** (2013), 1–10. https://doi.org/10.1186/1687-1812-2013-18
14. W. J. Leong, M. A. Hassan, M. Y. Waziri, A matrix-free quasi-Newton method for solving large-scale nonlinear systems, *Comput. Math. Appl.,* **62** (2011), 2354–2363. https://doi.org/10.1016/j.camwa.2011.07.023
15. D. Li, M. Fukushima, A globally and superlinearly convergent Gauss–Newton-based BFGS method for symmetric nonlinear equations, *SIAM J. Numerical Anal.,* **37** (1999), 152–172. https://doi.org/10.1137/S0036142998335704

16. J. K. Liu, S. J. Li, A projection method for convex constrained monotone nonlinear equations with applications, *Comput. Math. Appl.,* **70** (2015), 2442–2453. https://doi.org/10.1016/j.camwa.2015.09.014
17. J. K. Liu, B. Tang, T. Liu, Z. Yang, S. Liang, An accelerated double-step derivative-free projection method based algorithm using Picard–Mann iterative process for solving convex constrained nonlinear equations, *J. Compu. Appl. Math.,* **464** (2025), 116541. https://doi.org/10.1016/j.cam.2025.116541
18. J. Martínez, Practical quasi-Newton methods for solving nonlinear systems, *J. Comput. Appl. Math.,* **124** (2000), 97–121. https://doi.org/10.1016/S0377-0427(00)00434-9
19. E. D. Dolan, J. J. Moré, Benchmarking optimization software with performance profiles, *Math. Program.,* **91** (2002), 201–213. https://doi.org/10.1007/s101070100263
20. E. Nermeh, M. Abdullahi, A. S. Halilu, H. Abdullahi, Modification of a conjugate gradient approach for convex constrained nonlinear monotone equations with applications in signal recovery and image restoration, *Commun. Nonlinear Sci.,* **136** (2024), 108079. https://doi.org/10.1016/j.cnsns.2024.108079
21. J. Nocedal, S. J. Wright, *Numerical optimization,* Springer Science, 2006. http://dx.doi.org/10.1016/B978-0-12-775850-3.50017-0
22. M. Petrović, V. Rakočević, N. Kontrec, S. Panić, D. Ilić, Hybridization of accelerated gradient descent method, *Numer. Algorithms,* **79** (2018), 769–786. https://doi.org/10.1007/s11075-017-0460-4
23. B. Saheya, G. Chen, Y. Sui, C. Wu, A new Newton-like method for solving nonlinear equations, *SpringerPlus,* **5** (2016), 1–13. https://doi.org/10.1186/s40064-016-2909-7
24. S. B. Salihu, A. S. Halilu, M. Abdullahi, K. Ahmed, P. Mehta, S. Murtala, An improved spectral conjugate gradient projection method for monotone nonlinear equations with application, *J. Appl. Math. Comput.,* **70** (2024), 3879–3915. https://doi.org/10.1007/s12190-024-02121-4
25. M. V. Solodov, B. F. Svaiter, *A globally convergent inexact Newton method for systems of monotone equations*, in Reformulation: Nonsmooth, Piecewise Smooth, Semi-Smooth and Smoothing Methods, Springer, 1998, 355–369.
26. M. Y. Waziri, H. U. Muhammad, A. S. Halilu, K. Ahmed, Modified matrix-free methods for solving systems of nonlinear equations, *Optimization,* **69** (2020), 134–148. https://doi.org/10.1080/02331934.2020.1778689
27. Y. Xiao, H. Zhu, A conjugate gradient method to solve convex constrained monotone equations with applications in compressive sensing, *J. Math. Anal. Appl.,* **405** (2013), 310–319. https://doi.org/10.1016/j.jmaa.2013.04.017
28. Z. Yu, J. Lin, J. Sun, Y. Xiao, L. Liu, Z. Li, Spectral gradient projection method for monotone nonlinear equations with convex constraints, *Appl. Numer. Math.,* **59** (2009), 2416–2423. https://doi.org/10.1016/j.apnum.2009.04.004
29. N. Zhang, J. Liu, B. Tang, A three-term projection method based on spectral secant equation for nonlinear monotone equations, *JPN J. Ind. Appl. Math.,* **41** (2024), 617–635. https://doi.org/10.1007/s13160-023-00624-4
30. W. Zhou, D. Li, Limited memory BFGS method for nonlinear monotone equations, *J. Comput. Math.,* **25** (2007), 89–96.

31. W. Zhou, D. Li, A globally convergent BFGS method for nonlinear monotone equations without any merit functions, *Math. Comput.,* **77** (2008), 2231–2240. https://doi.org/10.1090/S0025-5718-08-02121-2
32. Z. Dai, X. Chen, F. Wen, A modified Perry's conjugate gradient method-based derivative-free method for solving large-scale nonlinear monotone equations, *Appl. Math. Comput.,* **270** (2015), 378–386. https://doi.org/10.1016/j.amc.2015.08.014
33. L. Zhang, W. Zhou, Spectral gradient projection method for solving nonlinear monotone equations, *J. Comput. Appl. Math.,* **196** (2006), 478–484. https://doi.org/10.1016/j.cam.2005.10.002