




Article

A Scaled Dai–Yuan Projection-Based Conjugate Gradient Method for Solving Monotone Equations with Applications

Ali Althobaiti ¹, Jamilu Sabi'u ², Homan Emadifar ³, Prem Junsawang ^{4,*} and Soubhagya Kumar Sahoo ⁵¹ Mathematics Department, College of Science, Taif University, Taif 21944, Saudi Arabia; aa.althobaiti@tu.edu.sa² Department of Mathematics, Faculty of Science, Yusuf Maitama Sule University, Kano P.M.B. 3099, Nigeria; jsabiu@nwu.edu.ng³ Department of Mathematics, Hamedan Branch, Islamic Azad University, Hamedan 1584743311, Iran; homan_emadi@yahoo.com⁴ Department of Statistics, Faculty of Science, Khon Kaen University, Khon Kaen 40002, Thailand⁵ Department of Mathematics, Institute of Technical Education and Research, Siksha O Anusandhan University, Bhubaneswar 751030, India; soubhagyakumarsahoo@soa.ac.in

* Correspondence: prem@kku.ac.th

Abstract: In this paper, we propose two scaled Dai–Yuan (DY) directions for solving constrained monotone nonlinear systems. The proposed directions satisfy the sufficient descent condition independent of the line search strategy. We also reasonably proposed two different relations for computing the scaling parameter at every iteration. The first relation is proposed by approaching the quasi-Newton direction, and the second one is by taking the advantage of the popular Barzilai–Borwein strategy. Moreover, we propose a robust projection-based algorithm for solving constrained monotone nonlinear equations with applications in signal restoration problems and reconstructing the blurred images. The global convergence of this algorithm is also provided, using some mild assumptions. Finally, a comprehensive numerical comparison with the relevant algorithms shows that the proposed algorithm is efficient.

Keywords: constrained monotone system; scaled conjugate gradient method; projection-based algorithm; quasi-Newton direction



Citation: Althobaiti, A.; Sabi'u, J.; Emadifar, H.; Junsawang, P.; Sahoo, S.K. A Scaled Dai–Yuan Projection-Based Conjugate Gradient Method for Solving Monotone Equations with Applications. *Symmetry* **2022**, *14*, 1401. <https://doi.org/10.3390/sym14071401>

Academic Editors: Włodzimierz Fechner and Jacek Chudziak

Received: 7 June 2022

Accepted: 4 July 2022

Published: 7 July 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Consider the constrained algebraic system

$$F(x) = 0, \quad x \in M, \quad (1)$$

where the function $F : \Omega \longleftrightarrow \mathbb{R}^n$ is a continuous mapping and is considered to be monotone, i.e.,

$$(F(x) - F(y))^T(x - y) \geq 0, \quad x, y \in M. \quad (2)$$

In addition, the set Ω is known to be a closed convex subset of \mathbb{R}^n . If $\Omega = \mathbb{R}^n$, the problem (1) is referred to as an unconstrained monotone nonlinear system. Both cases have been extensively studied by many researchers. The constrained system (1) appeared in many physical and mathematical applications, such as the financial forecasting problems [1], compressive sensing [2], Bregman distances [3], and monotone variational inequalities [4].

Moreover, the Newton method [5] and the quasi-Newton method [6,7] are well-known methods of solving the system of nonlinear equations. However, these methods are undesirable due to the Jacobian computation or its approximations at each iteration, as well as the storage requirements for storing the Jacobian matrix or its approximations at each iteration. Due to these limitations, these methods are unsuitable for large-scale problems involving both smooth and nonsmooth systems. The spectral gradient (SG) and conjugate

gradient (CG) methods are another set of methods, which are used to solve systems of nonlinear equations. These methods are matrix-free; they can successfully handle large-scale problems, see [8–15] and the references therein. The main iterative procedure of CG methods for the system of monotone equations given an initial point x_0 is

$$x_{k+1} = x_k + \alpha_k d_k, \tag{3}$$

where α_k is the step size that can be computed using the line search strategies. The term d_k is the spectral gradient direction defined by

$$d_k = -F(x_k), \quad k = 0, \quad d_k = -F(x_k) + \beta_k d_{k-1}, \quad k \geq 1, \tag{4}$$

the scalar parameter β_k is the parameter that differentiates the CG methods. Among the most efficient CG parameters is one proposed by Dai and Yuan [16], given by

$$\beta_k^{DY} = \frac{F_k^T F_k}{d_{k-1}^T y_{k-1}}. \tag{5}$$

where $F_k = F(x_k)$. Recently, the DY CG parameter and its modification have been used to solve the monotone nonlinear systems, for example, the two descent DY algorithm for monotone equations [17], the modified DY with sufficient descent property [18], the efficient DY-type spectral CG method [19], and the descent three-term DY CG method with application [20]. Most of these methods are efficient to some extent and apply to real-life applications. Motivated by these DY approaches, we will present a scaled, Dai–Yuan, projection-based conjugate gradient method to solve monotone equations with applications in signal and image recovery. In addition, we reasonably propose different relations to compute the scaling parameter at every iteration, namely, by tending the proposed direction to approach the quasi-Newton direction, and by taking advantage of the popular Barzilai–Borwein strategy.

The next section provides a detailed derivation of the proposed algorithm. Section 3 presents the convergence result. Section 4 provides the experimental findings in comparison with some existing algorithms. We provide the conclusion in the last section.

2. A Scaled Dai–Yuan CG Methods

This section will present spectral DY CG methods for solving monotone equations with the convex constrained. Among the well-known efficient is the one proposed by Dai and Young [16]. Moreover, the scaling strategy proved to be an efficient strategy for enhancing the performance of CG-based algorithms, see [21]. In this work, to enhance the performance of the DY CG parameter, we present the following scaled DY CG parameter

$$\beta_k^{SDY} = \sigma \beta_k^{DY}, \tag{6}$$

such that $|\sigma| \leq 1$. The main contribution of this work is to propose some reasonable ways of computing the scalar σ at every iteration.

2.1. Scaling Parameter Based on the Quasi-Newton Approach

Newton and quasi-Newton directions contain the full Jacobian information or its approximates. Recall that the quasi-Newton equation is defined as

$$y_k = B_{k+1} s_k, \tag{7}$$

where B_{k+1} is a positive definite and symmetric Jacobian estimate. Again, assuming that H_{k+1} is the inverse of B_{k+1} , the quasi-Newton direction is as follows:

$$d_{k+1} = -H_{k+1} F_{k+1}. \tag{8}$$

Using Equation (4), and Equation (6), the spectral DY search direction is given as

$$d_{k+1} = -F_{k+1} + \sigma \frac{F_{k+1}^T F_{k+1}}{d_k^T y_k} d_k, \quad k = 0, 1, \dots \quad (9)$$

Using the equality relation, and from Equations (8) and (9), we obtain

$$-H_{k+1} F_{k+1} = -F_{k+1} + \sigma \frac{F_{k+1}^T F_{k+1}}{d_k^T y_k} d_k. \quad (10)$$

Multiplying Equation (10) by $-s_k^T$ and B_{k+1} to obtain

$$s_k^T F_{k+1} = B_{k+1} s_k^T F_{k+1} - \sigma \frac{F_{k+1}^T F_{k+1}}{d_k^T y_k} B_{k+1} s_k^T d_k. \quad (11)$$

We further eliminate the matrix B_{k+1} in Equation (11) using the quasi-Newton Equation (7) to obtain

$$s_k^T F_{k+1} = y_k^T F_{k+1} - \sigma \frac{F_{k+1}^T F_{k+1}}{d_k^T y_k} y_k^T d_k. \quad (12)$$

Solving for σ in Equation (12) to obtain

$$\sigma_k = \frac{(y_k - s_k)^T F_{k+1}}{\|F_{k+1}\|^2}. \quad (13)$$

Furthermore, to achieve $|\sigma| \leq 1$ and to benefit from the nonnegative restriction of Polak–Ribière–Polyak, we proposed the following modified version of Equation (14)

$$\sigma_k^* = \min\{1, |\sigma|\}. \quad (14)$$

Moreover, to ensure the sufficient condition is satisfied, independent of the line search procedure, we proposed the following scaled DY CG direction

$$d_{k+1} = -\tau_k F_{k+1} + \sigma_k^* \beta_k^{DY} d_k, \quad k = 0, 1, \dots, \quad (15)$$

where

$$\tau_k = 1 + \sigma_k^* \frac{F_{k+1}^T d_k}{d_k^T y_k}. \quad (16)$$

2.2. Scaling Parameter Based on Barzilai–Borwein Approach

The most prevalent choices for the spectral scalars are those offered by Barzilai–Borwein [22], given as follows:

$$\gamma_k^1 = \frac{s_k^T s_k}{y_k^T s_k} \quad \text{and} \quad \gamma_k^2 = \frac{s_k^T y_k}{y_k^T y_k}. \quad (17)$$

Now, considering the relations (4) and (6), the direction (9) can be written as

$$d_0 = -F_0, \quad d_{k+1} = -A_{k+1} F_{k+1}, \quad k = 0, 1, \dots, \quad (18)$$

where A_{k+1} is defined by

$$A_{k+1} = I - \sigma \frac{F_{k+1} d_k^T}{d_k^T y_k}. \quad (19)$$

We aimed to take advantage of the Barzilai–Borwein [22] approach; hence, we put forward the parameter σ as the solution to the following minimization problem:

$$\min_{\sigma} \|A_{k+1} - \gamma_k I\|_F^2, \tag{20}$$

where $\|\cdot\|_F$ stands for the Frobenius matrix norm and

$$\gamma_k = \max\left\{\gamma_{\min}, \min\left\{\gamma_k^j, \gamma_{\max}\right\}\right\}, \quad j = 1, 2, \tag{21}$$

where the $0 < \gamma_{\min} < \gamma_{\max} < \infty$. Putting forward the relation $\|A\|_F^2 = \text{Trace}(A^T A)$, we obtained the following solution for (20) as follows

$$\sigma_k = (1 - \gamma_k) \frac{F_{k+1}^T y_k}{\|d_k\|^2 \beta_k^{DY}}. \tag{22}$$

Furthermore, to achieve $|\sigma| \leq 1$ and to benefit from the non-negative restriction of Polak–Ribière–Polyak, we proposed the following modified version of (22) as follows

$$\hat{\sigma}_k^* = \min\{1, |\sigma|\}. \tag{23}$$

Moreover, to ensure the sufficient condition is satisfied, independent of the line search procedure, we proposed the following scaled DY CG direction

$$d_{k+1} = -\hat{\tau}_k F_{k+1} + \hat{\sigma}_k^* \beta_k d_k, \quad k = 0, 1, \dots, \tag{24}$$

where

$$\hat{\tau}_k = 1 + \hat{\sigma}_k^* \frac{F_{k+1}^T d_k}{d_k^T y_k}. \tag{25}$$

Next, we define the projection operator $M_{\Omega} : \mathbb{R}^n \rightarrow \Omega$, given by

$$M_{\Omega}[x] = \arg \min\{\|x - z\| \mid z \in \Omega\}, \quad \forall x \in \mathbb{R}^n, \tag{26}$$

one of the appealing features of this operator is its non-expansive property, i.e.,

$$\|M_{\Omega}[x] - M_{\Omega}[z]\| \leq \|x - z\|, \quad \forall x, z \in \mathbb{R}^n. \tag{27}$$

Now, we present the spectral DY CG projection-based algorithm (Algorithm 1) for solving convex constrained monotone nonlinear equations.

Algorithm 1: The spectral DY CG projection-based algorithm (SDYCG)

Step 0 Initialize: $\epsilon \geq 0, a \in (0, 1), b > 0, c > 0, g > 0, 0 < \gamma_{\min} \leq \gamma_{\max}$ and $x_0 \in \mathbf{R}^n$. Set $k = 0$ and $d_0 = -F_0$.

Step 1 If $\|F_k\| \leq \epsilon$, stop; otherwise, go to Step 2.

Step 2 Compute the spectral DY CG directions d_k using (15) or (24), where $s_k = h_k - x_k$, and $y_k = F(h_k) - F_k + as_k$.

Step 3 Set $h_k = x_k + \alpha_k d_k$ and determine $\alpha_k = \max\{bc^i : i = 0, 1, 2, \dots\}$ satisfying

$$-F(h_k)^T d_k \geq g\alpha_k \|F(h_k)\| \|d_k\|^2 \tag{28}$$

Step 4 If $h_k \in \Omega$ and $\|F(h_k)\| = 0$ stop; otherwise

$$x_{k+1} = M_\Omega[x_k - v_k F(h_k)], \tag{29}$$

where

$$v_k = \frac{F(h_k)^T (x_k - h_k)}{\|F(h_k)\|^2}. \tag{30}$$

Step 5 Set $k = k + 1$ and then go to **Step 1**.

3. Global Convergence

This section presents the SDYCG algorithm’s global convergence result using the following assumptions:

- A1 The solution set Ω is non-empty.
- A2 The function F is Lipschitz continuous for Ω , i.e., for $L > 0$

$$\|F(x) - F(z)\| \leq L\|x - z\|, \quad \forall x, z \in \Omega. \tag{31}$$

- A3 The function F satisfies (2).

Remark:

1. The proposed search directions (15) and (24) satisfy the sufficient descent condition independent of the line-search procedure used.
2. Additionally, from the monotonicity assumption on F , we have

$$s_k^T y_k \geq as_k^T s_k. \tag{32}$$

Lemma 1. For all $k \geq 0$, the line search (28) is well-defined.

Proof. Assume that there exists $k_0 \geq 0$, such that, for any non-negative integer i ,

$$-F(x_{k_0} + bc^i d_{k_0})^T d_{k_0} < gbc^i \|F(x_{k_0} + bc^i d_{k_0})\| \|d_{k_0}\|^2. \tag{33}$$

Let $i \rightarrow \infty$ in (33), we have

$$-F(x_{k_0})^T d_{k_0} \leq 0. \tag{34}$$

From the sufficient condition of the HSGP direction, we have

$$-F(x_{k_0})^T d_{k_0} \geq \|F(x_{k_0})\|^2 > 0, \tag{35}$$

clearly, (34) contradicts (35), and so the proof is complete. \square

Lemma 2. If the sequences $\{x_k\}$ and $\{b_k\}$ are generated by HSGP algorithm, then

$$\lim_{k \rightarrow \infty} \alpha_k \|d_k\| = 0. \tag{36}$$

Proof. From the line search (28), we have

$$F(h_k)^T(x_k - h_k) = -F(h_k)^T d_k \geq \lambda \alpha_k^2 \|F(h_k)\| \|d_k\|^2 > 0. \tag{37}$$

Assume that $x^* \in \Omega$ such that $F(x^*) = 0$; using the monotonicity of F , we have

$$\begin{aligned} F(h_k)^T(x_k - x^*) &= F(h_k)^T(x_k - h_k) + F(h_k)^T(h_k - x^*) \\ &\geq F(h_k)^T(x_k - h_k) + F(x^*)^T(h_k - x^*) \\ &= F(h_k)^T(x_k - h_k). \end{aligned} \tag{38}$$

Using (37) and (38) to obtain

$$\begin{aligned} \|x_{k+1} - x^*\|^2 &= \|W_\Omega(x_k - v_k F(h_k)) - x^*\|^2 \\ &\leq \|x_k - v_k F(h_k) - x^*\|^2 \\ &= \|x_k - x^*\|^2 - 2v_k F(h_k)^T(x_k - x^*) + v_k^2 \|F(h_k)\|^2. \end{aligned} \tag{39}$$

Looking at the definition of v_k , and the Cauchy–Schwarz inequality in (39), we obtain

$$\begin{aligned} \|x_{k+1} - x^*\|^2 &\leq \|x_k - x^*\|^2 - 2v_k F(h_k)^T(x_k - h_k) + v_k^2 \|F(h_k)\|^2 \\ &\leq \|x_k - x^*\|^2 - \frac{(F(h_k)^T(x_k - h_k))^2}{\|F(h_k)\|^2} \\ &\leq \|x_k - x^*\|^2 - g^2 \|x_k - h_k\|^4, \end{aligned} \tag{40}$$

thus, we have

$$\|x_{k+1} - x^*\| \leq \|x_k - x^*\|, \quad \forall k \geq 0. \tag{41}$$

Therefore, $\{\|x_k - x^*\|\}$ is a decreasing sequence and convergent. Now, utilizing (41) and the Lipschitz continuity of F , we obtain

$$\|F(x_k)\| = \|F(x_k) - F(x^*)\| \leq L \|x_k - x^*\| \leq L \|x_0 - x^*\| = T. \tag{42}$$

Using (28), monotonicity of F , and the Cauchy–Schwarz inequality, we obtain

$$g \|F(h_k)\| \|x_k - h_k\|^2 \leq F(h_k)^T(x_k - h_k) \leq \|F(h_k)\| \|x_k - h_k\|, \tag{43}$$

which gives

$$g \|x_k - h_k\| \leq 1. \tag{44}$$

From (44), we can see that the sequence $\{h_k\}$ is bounded. From (40), we obtain

$$g^2 \sum_{k=0}^{\infty} \|x_k - h_k\|^4 \leq \sum_{k=0}^{\infty} (\|x_k - x^*\|^2 - \|x_{k+1} - x^*\|^2) < \infty. \tag{45}$$

This implies

$$\lim_{k \rightarrow \infty} \|x_k - h_k\| = \lim_{k \rightarrow \infty} \alpha_k \|d_k\| = 0. \tag{46}$$

The proof is now complete. \square

Lemma 3. *The directions generated by SDYCG algorithms are bounded.*

Proof. Starting with (15) for the case $k = 0$,

$$d_0 \leq \|-F_0\| = \|F_0\|. \tag{47}$$

Now, for $k \geq 0$, we obtain

$$\begin{aligned}
 \|d_k\| &= \left\| -\tau_k F_{k+1} + \sigma_k^* \beta_k^{DY} d_k \right\| \\
 &= \left\| -\left(1 + \sigma_k^* \frac{F_{k+1}^T d_k}{d_k^T y_k} \right) F_{k+1} + \sigma_k^* \frac{F_{k+1}^T F_{k+1}}{d_k^T y_k} d_k \right\| \\
 &\leq \|F_{k+1}\| + 2 \frac{\|F_{k+1}\|^2 \|s_k\|}{a \|s_k\|^2} \\
 &\leq T + 2 \frac{T^2}{a \|s_k\|} = \bar{T}
 \end{aligned}
 \tag{48}$$

The first inequality directly follows from the Cauchy–Swartz inequality, (32) and the main definition of h_k , while the second inequality directly follows from (42) and (46). The same result can be shown for the second search direction (24). \square

Theorem 1. *If the sequence $\{x_k\}$ is generated by SDYCG algorithm, then*

$$\liminf_{k \rightarrow \infty} \|F_k\| = 0.
 \tag{49}$$

Proof. Suppose that (49) is not true. Then, there exists $\omega > 0$, such that

$$\|F(x_k)\| \geq \omega, \quad \forall k \in \mathbb{N}.
 \tag{50}$$

Using the sufficient descent condition and the Cauchy–Schwartz inequality, we have

$$\|F_k\|^2 \leq -F_k^T d_k \leq \|F_k\| \|d_k\| \quad \forall k \in \mathbb{N}.
 \tag{51}$$

Hence,

$$\|d_k\| \geq \omega > 0.
 \tag{52}$$

Using Lemma 2 and the inequality (52), we obtain

$$\lim_{k \rightarrow \infty} \alpha_k = 0.
 \tag{53}$$

By line-search (28), there is α' , so that

$$-F(x_k + \alpha' d_k)^T d_k < g \alpha' \|F(x_k + \alpha')\| \|d_k\|^2.
 \tag{54}$$

From the fact that $\{x_k\}$ and $\{d_k\}$ are bounded, there is an accumulation point x^* , such that $\lim_{k \rightarrow \infty} x_k = x^*$, for $k \in B_1$. Similarly, there is a set $B_2 \subset B_1$ and an accumulation point d^* , such that $\lim_{k \rightarrow \infty} d_k = d^*$, for $k \in B_2$, where B_1, B_2 are infinite index sets. Hence, taking the limit as $k \rightarrow \infty$ in (54), we can obtain

$$F(x^*)^T d^* \geq 0.
 \tag{55}$$

Similarly, from the sufficient condition, taking the limit as $k \rightarrow \infty$ in the sufficient descent condition to obtain

$$F(x^*)^T d^* < 0.
 \tag{56}$$

Clearly, (55) and (56) provide a contradiction. \square

4. Numerical Experiment and Applications

This section comprises numerical experiments using the proposed algorithm to solve large-scale constrained monotone nonlinear systems and image and signal restoration problems.

4.1. Application to the Monotone Nonlinear Equations

The proposed two-direction algorithm was compared to the modified spectral gradient projection (MSGP) algorithm [23] and the derivative-free spectral projection (DFSP) [24]. Furthermore, for the proposed algorithm, we set the starting parameters as follows: $a = 0.1$, $g = 0.0001$, $c = 0.99$, $b = 1$, and $\epsilon = 10^{-9}$. Except for the stopping conditions, all of the compared algorithms were implemented using the published values. The following problems were tested:

Problem 1 ([25]).

$$F_1(x) = e^{x_1} - 1,$$

$$F_i(x) = e^{x_i} + x_{i-1} - 1, \quad i = 2, 3, \dots, n-1, \text{ and } \Omega = \mathbf{R}_+^n$$

Problem 2 ([25]).

$$F_i(x) = \log(|x_i| + 1) - \frac{x_i}{n}, \quad i = 2, 3, \dots, n,$$

$$\text{and } \Omega = \{x \in \mathbf{R}_+^n : \sum_{i=1}^n x_i \leq n, \quad x_i \geq -1, \quad i = 1, 2, \dots, n\}.$$

Problem 3 ([26]).

$$F_1(x) = \cos(x_1) - 9 + 3x_1 + 8 \exp(x_2)$$

$$F_i(x) = \cos(x_i) - 9 + 3x_i + 8 \exp(x_{i-2}), \text{ for } i = 2, 3, \dots, n, \text{ and } \Omega = \mathbf{R}_+^n$$

Problem 4 (This problem is from Reference).

$$F_i(x) = e^{x_i} - 1, \quad i = 1, 2, \dots, n, \text{ and } \Omega = \mathbf{R}_+^n.$$

Problem 5 ([21]).

$$F_i(x) = 4x_i + (x_{i+1} - 2x_i) - \frac{x_{i+1}^2}{3}, \text{ for } i = 1, 2, \dots, n-1,$$

$$F_n(x) = 4x_n + (x_{n-1} - 2x_n) - \frac{x_{n-1}^2}{3}, \text{ and } \Omega = \mathbf{R}_+^n.$$

Problem 6 ([27]).

$$F(x_i) = 2x_i - \sin|x_i|, \text{ for } i = 1, 2, 3, \dots, n, \text{ and } \Omega = \mathbf{R}_+^n.$$

From Tables 1–6, The NORM the norm of the function F at the stopping point. The following initial points are considered: $x_0^1 = (2, 2, \dots, 2)$, $x_0^2 = (1, \frac{1}{2}, \frac{1}{3}, \dots, \frac{1}{n})$, $x_0^3 = (1, 1, \dots, 1)$, $x_0^4 = (\frac{1}{n}, \frac{2}{n}, \dots, 1)$, $x_0^5 = (n - \frac{1}{n}, n - \frac{2}{n}, \dots, n - 1)$, $x_0^6 = (2, \frac{2}{2}, \frac{2}{3}, \dots, \frac{2}{n})$, $x_0^7 = (1 - 1, 1 - \frac{1}{2}, 1 - \frac{1}{3}, \dots, 1 - \frac{1}{n})$ and $x_0^8 = (-3, -3, \dots, -3)$.

For the sake of the numerical comparison, we used the three parameters, namely, the number of iterations (ITER), the number of function evaluations (FEV), and computational or CPU time (TIME). Starting with Problem 1, the proposed algorithm successfully solves the entire initial guess, with fewer ITER, FEV, and TIME. Additionally, the DFSP algorithm solved all the initial guesses, with a relatively high ITER and FEV compared to the SDYCG algorithm. We also observed that the MSGP algorithm failed for the three initial guesses at the start, and failed for four initial guesses regarding dimension increases, although it performed wonderfully for Problem 2 for all three parameters.

Table 1. Numerical comparison of SDYCG algorithm versus DFSP [24], and MSGP algorithms [23].

Problem 1		SDYCG(1)				SDYCG(2)				MSGP				DFSP			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
500	x_0^1	2	27	0.178052	0	2	27	0.030794	0	121	1538	0.102812	9.3×10^{-10}	7	73	0.051887	0
	x_0^2	2	27	0.007911	0	2	27	0.010431	0	91	1173	0.075893	8.58×10^{-10}	4	45	0.025909	0
	x_0^3	2	27	0.004025	0	2	27	0.004284	0	95	1200	0.146293	8.29×10^{-10}	6	69	0.013521	0
	x_0^4	3	40	0.008521	0	3	40	0.008234	0	Fail	Fail	Fail	Fail	5	57	0.011795	0
	x_0^5	2	27	0.006885	0	2	27	0.006878	0	Fail	Fail	Fail	Fail	5	56	0.011774	0
	x_0^6	2	27	0.005948	0	2	27	0.008259	0	57	725	0.042269	8.33×10^{-10}	6	72	0.014265	0
	x_0^7	2	27	0.004092	0	2	27	0.006461	0	Fail	Fail	Fail	Fail	5	56	0.01217	0
	x_0^8	5	66	0.009501	0	5	66	0.013534	0	2	18	0.004343	0	2	17	0.07065	0
1000	x_0^1	2	27	0.004634	0	2	27	0.008994	0	134	1706	0.322519	9.92×10^{-10}	8	84	0.022746	0
	x_0^2	2	27	0.009655	0	2	27	0.009192	0	91	1173	0.225743	8.59×10^{-10}	4	45	0.012467	0
	x_0^3	2	27	0.008619	0	2	27	0.009266	0	89	1122	0.212044	7.94×10^{-10}	9	78	0.019796	0
	x_0^4	3	40	0.00906	0	3	40	0.011755	0	Fail	Fail	Fail	Fail	5	56	0.01559	0
	x_0^5	2	27	0.006779	0	2	27	0.008895	0	Fail	Fail	Fail	Fail	5	56	0.015161	0
	x_0^6	2	27	0.004795	0	2	27	0.007495	0	57	725	0.138729	7.84×10^{-10}	6	72	0.019059	0
	x_0^7	2	27	0.006632	0	2	27	0.009641	0	Fail	Fail	Fail	Fail	5	56	0.015644	0
	x_0^8	5	66	0.017444	0	5	66	0.017055	0	2	18	0.004908	0	2	17	0.006929	0
10,000	x_0^1	2	27	0.047267	0	2	27	0.041266	0	104	1311	1.560793	9.44×10^{-10}	8	65	0.116677	0
	x_0^2	2	27	0.03194	0	2	27	0.039564	0	91	1173	1.997119	8.6×10^{-10}	4	45	0.080051	0
	x_0^3	2	27	0.031136	0	2	27	0.041833	0	98	1230	2.601348	9.08×10^{-10}	9	68	0.123208	0
	x_0^4	3	40	0.057081	0	3	40	0.056858	0	Fail	Fail	Fail	Fail	5	56	0.09662	0
	x_0^5	2	27	0.035342	0	2	27	0.040675	0	Fail	Fail	Fail	Fail	5	56	0.103659	0
	x_0^6	2	27	0.027089	0	2	27	0.038279	0	56	712	1.482753	9.8×10^{-10}	6	72	0.126239	0
	x_0^7	2	27	0.041587	0	2	27	0.04066	0	Fail	Fail	Fail	Fail	5	56	0.102553	0
	x_0^8	5	66	0.089053	0	5	66	0.111986	0	2	18	0.042366	0	3	30	0.054729	0

Table 1. Cont.

Problem 1		SDYCG(1)				SDYCG(2)				MSGP				DFSP			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
50,000	x_0^1	2	27	0.119851	0	2	27	0.192334	0	103	1293	10.94405	9.7×10^{-10}	10	89	0.690911	0
	x_0^2	2	27	0.158001	0	2	27	0.181709	0	91	1173	8.038891	8.6×10^{-10}	4	45	0.329212	0
	x_0^3	2	27	0.1872	0	2	27	0.195489	0	Fail	Fail	Fail	Fail	7	71	0.520311	0
	x_0^4	3	40	0.170595	0	3	40	0.294915	0	Fail	Fail	Fail	Fail	5	56	0.425746	0
	x_0^5	2	27	0.182155	0	2	27	0.19715	0	Fail	Fail	Fail	Fail	5	56	0.418211	0
	x_0^6	2	27	0.118753	0	2	27	0.188444	0	56	712	4.008244	9.74×10^{-10}	6	72	0.527691	0
	x_0^7	2	27	0.191926	0	2	27	0.199747	0	Fail	Fail	Fail	Fail	5	56	0.421186	0
	x_0^8	5	66	0.363679	0	5	66	0.467007	0	2	18	0.142288	0	3	30	0.223121	0
100,000	x_0^1	2	27	0.447002	0	2	27	0.423225	0	116	1456	17.30029	9.17×10^{-10}	9	67	1.114614	0
	x_0^2	2	27	0.282127	0	2	27	0.398508	0	91	1173	13.14327	8.6×10^{-10}	4	45	0.864917	0
	x_0^3	2	27	0.438826	0	2	27	0.439531	0	Fail	Fail	Fail	Fail	11	124	1.896561	0
	x_0^4	3	40	0.459044	0	3	40	0.630171	0	Fail	Fail	Fail	Fail	5	56	0.951345	0
	x_0^5	2	27	0.457132	0	2	27	0.441214	0	Fail	Fail	Fail	Fail	5	56	1.073445	0
	x_0^6	2	27	0.387432	0	2	27	0.398113	0	56	712	8.451864	9.73×10^{-10}	6	72	1.197623	0
	x_0^7	2	27	0.512523	0	2	27	0.424911	0	Fail	Fail	Fail	Fail	5	56	0.959261	0
	x_0^8	5	66	0.76242	0	5	66	0.992397	0	2	18	0.266698	0	3	30	0.545118	0

Table 2. Numerical comparison of SDYCG algorithm versus DFSP [24], and MSGP algorithms [23].

Problem 2		SDYCG(1)				SDYCG(2)				MSGP				DFSP			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
500	x_0^1	7	15	0.016995	1×10^{-10}	7	15	0.009055	1×10^{-10}	2	5	0.003311	0	2	5	0.006153	0
	x_0^2	11	34	0.010415	0	4	20	0.008933	0	3	7	0.004031	0	4	13	0.007909	0
	x_0^3	6	13	0.006106	1.9×10^{-10}	6	13	0.008533	1.9×10^{-10}	2	5	0.004184	0	17	35	0.016027	9.24×10^{-10}
	x_0^4	5	11	0.005878	0	5	11	0.008165	0	4	20	0.008457	0	5	12	0.007036	0
	x_0^5	5	11	0.005853	0	5	11	0.008164	0	4	20	0.007013	0	5	12	0.007406	0
	x_0^6	6	13	0.008509	0	4	9	0.006486	0	3	7	0.004293	0	4	9	0.007139	0
	x_0^7	5	11	0.006749	0	5	11	0.007685	0	4	20	0.00637	0	5	12	0.008024	0
	x_0^8	1	3	0.003592	0	1	3	0.004139	0	1	3	0.003345	0	1	3	0.003913	0
1000	x_0^1	7	15	0.009224	1.74×10^{-11}	7	15	0.011962	1.74×10^{-11}	2	5	0.00456	0	2	5	0.008143	0
	x_0^2	4	9	0.00694	0	4	20	0.011606	0	3	7	0.005106	0	4	13	0.009123	0
	x_0^3	6	13	0.009309	3.6×10^{-11}	6	13	0.010888	3.6×10^{-11}	2	5	0.004814	0	18	37	0.022215	5.1×10^{-10}
	x_0^4	5	11	0.009144	0	5	11	0.008701	0	4	20	0.009211	0	5	12	0.00947	0
	x_0^5	5	11	0.008489	0	5	11	0.008808	0	4	20	0.009244	0	5	12	0.009505	0
	x_0^6	3	7	0.005678	0	4	9	0.008407	0	3	7	0.004977	0	4	9	0.008368	0
	x_0^7	5	11	0.008087	0	5	11	0.009238	0	4	20	0.008604	0	5	12	0.009677	0
	x_0^8	1	3	0.0038	0	1	3	0.004987	0	1	3	0.005321	0	1	3	0.004467	0
10,000	x_0^1	7	15	0.045708	1.11×10^{-13}	7	15	0.043334	1.33×10^{-13}	2	5	0.02446	0	2	5	0.017714	0
	x_0^2	4	9	0.02586	0	4	20	0.04929	0	3	7	0.017497	0	4	13	0.03338	0
	x_0^3	6	13	0.038779	3.55×10^{-13}	6	13	0.043985	3.77×10^{-13}	2	5	0.015054	0	20	41	0.108359	3.63×10^{-10}
	x_0^4	5	11	0.032356	0	5	11	0.030417	0	4	20	0.047689	0	5	12	0.03619	0
	x_0^5	5	11	0.033364	0	5	11	0.034083	0	4	20	0.05494	0	5	12	0.038086	0
	x_0^6	3	7	0.022243	0	4	9	0.026586	0	3	7	0.027866	0	4	9	0.026896	0
	x_0^7	5	11	0.032746	0	5	11	0.031654	0	4	20	0.047107	0	5	12	0.036912	0
	x_0^8	1	3	0.009583	0	1	3	0.009103	0	1	3	0.010396	0	1	3	0.01129	0

Table 2. Cont.

Problem 2		SDYCG(1)				SDYCG(2)				MSGP			DFSP				
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
50,000	x_0^1	6	13	0.130026	4.01×10^{-10}	6	13	0.134782	4.01×10^{-10}	2	5	0.051851	0	21	43	0.445089	3.8×10^{-10}
	x_0^2	4	9	0.082732	0	4	20	0.159078	0	3	7	0.074055	0	4	13	0.117798	0
	x_0^3	6	13	0.109939	4.97×10^{-14}	6	14	0.136721	1.34×10^{-11}	2	5	0.051362	0	21	43	0.417691	4.89×10^{-10}
	x_0^4	5	11	0.096546	0	5	11	0.106255	0	4	20	0.163909	0	5	12	0.120821	0
	x_0^5	5	11	0.094329	0	5	11	0.10836	0	4	20	0.170905	0	5	12	0.114635	0
	x_0^6	3	7	0.058725	0	4	9	0.092347	0	3	7	0.069873	0	4	9	0.089748	0
	x_0^7	5	11	0.101219	0	5	11	0.113213	0	4	20	0.168554	0	5	12	0.118881	0
	x_0^8	1	3	0.026044	0	1	3	0.028872	0	1	3	0.027349	0	1	3	0.02784	0
100,000	x_0^1	6	13	0.229267	2.71×10^{-10}	6	13	0.267816	2.71×10^{-10}	2	5	0.104652	0	22	45	0.877074	4.89×10^{-10}
	x_0^2	4	9	0.145292	0	4	20	0.344569	0	3	7	0.131366	0	4	13	0.253001	0
	x_0^3	5	11	0.208255	9.27×10^{-10}	5	11	0.21464	9.27×10^{-10}	2	5	0.095152	0	21	43	0.848432	9.42×10^{-10}
	x_0^4	5	11	0.171397	0	5	11	0.214585	0	4	20	0.320318	0	5	12	0.233611	0
	x_0^5	5	11	0.167816	0	5	11	0.213382	0	4	20	0.34222	0	5	12	0.262368	0
	x_0^6	3	7	0.127268	0	4	9	0.17047	0	3	7	0.132036	0	4	9	0.199059	0
	x_0^7	5	11	0.162501	0	5	11	0.214841	0	4	20	0.321408	0	5	12	0.264622	0
	x_0^8	1	3	0.054556	0	1	3	0.051797	0	1	3	0.051647	0	1	3	0.055688	0

Table 3. Numerical comparison of SDYCG algorithm versus DFSP [24], and MSGP algorithms [23].

Problem 3		SDYCG(1)				SDYCG(2)				MSGP				DFSP			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
500	x_0^1	1	14	0.0239	0	1	14	0.00767	0	1	14	0.00708	0	2	26	0.010194	0
	x_0^2	1	14	0.007732	0	1	14	0.008647	0	3	40	0.01239	0	2	24	0.009397	0
	x_0^3	1	14	0.007985	0	1	14	0.007406	0	1	14	0.007757	0	2	24	0.010515	0
	x_0^4	1	14	0.007445	0	1	14	0.007337	0	Fail	Fail	Fail	Fail	5	63	0.019328	0
	x_0^5	1	14	0.007483	0	1	14	0.007494	0	3	40	0.011605	0	5	63	0.020952	0
	x_0^6	1	14	0.006083	0	1	14	0.007518	0	Fail	Fail	Fail	Fail	2	24	0.008674	0
	x_0^7	1	14	0.006988	0	1	14	0.006873	0	3	40	0.011505	0	5	63	0.019238	0
	x_0^8	2	27	0.016347	0	2	27	0.015556	0	1	12	0.006715	0	1	9	0.006439	0
1000	x_0^1	1	14	0.00895	0	1	14	0.009538	0	1	14	0.009487	0	2	26	0.01487	0
	x_0^2	1	14	0.007879	0	1	14	0.008877	0	3	40	0.01747	0	2	24	0.011346	0
	x_0^3	1	14	0.008755	0	1	14	0.00954	0	1	14	0.009751	0	2	24	0.01299	0
	x_0^4	1	14	0.009495	0	1	14	0.009276	0	Fail	Fail	Fail	Fail	5	63	0.030107	0
	x_0^5	1	14	0.00908	0	1	14	0.008924	0	3	40	0.016945	0	5	63	0.028482	0
	x_0^6	1	14	0.008038	0	1	14	0.008538	0	Fail	Fail	Fail	Fail	2	24	0.012681	0
	x_0^7	1	14	0.008972	0	1	14	0.009078	0	3	40	0.016225	0	5	63	0.027616	0
	x_0^8	2	27	0.021469	0	2	27	0.023114	0	1	12	0.008773	0	1	9	0.007666	0
10,000	x_0^1	1	14	0.036132	0	1	14	0.039536	0	1	14	0.042494	0	2	26	0.069214	0
	x_0^2	1	14	0.031156	0	1	14	0.038291	0	3	40	0.098848	0	2	24	0.059419	0
	x_0^3	1	14	0.038817	0	1	14	0.045399	0	1	14	0.038984	0	2	24	0.064127	0
	x_0^4	1	14	0.036851	0	1	14	0.047515	0	Fail	Fail	Fail	Fail	5	63	0.14985	0
	x_0^5	1	14	0.047204	0	1	14	0.041347	0	3	40	0.096622	0	5	63	0.146482	0
	x_0^6	1	14	0.031515	0	1	14	0.037743	0	Fail	Fail	Fail	Fail	2	24	0.056053	0
	x_0^7	1	14	0.03524	0	1	14	0.038517	0	3	40	0.094222	0	5	63	0.160358	0
	x_0^8	2	27	0.087458	0	2	27	0.092896	0	1	12	0.033203	0	1	9	0.024582	0

Table 3. Cont.

Problem 3		SDYCG(1)				SDYCG(2)				MSGP				DFSP			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
50,000	x_0^1	1	14	0.154012	0	1	14	0.148305	0	1	14	0.146243	0	2	26	0.258653	0
	x_0^2	1	14	0.128182	0	1	14	0.134815	0	3	40	0.384249	0	2	24	0.21526	0
	x_0^3	1	14	0.152506	0	1	14	0.146604	0	1	14	0.14895	0	2	24	0.229867	0
	x_0^4	1	14	0.134076	0	1	14	0.160741	0	Fail	Fail	Fail	Fail	5	63	0.62062	0
	x_0^5	1	14	0.15077	0	1	14	0.152467	0	3	40	0.297547	0	5	63	0.563123	0
	x_0^6	1	14	0.119791	0	1	14	0.129554	0	Fail	Fail	Fail	Fail	2	24	0.231704	0
	x_0^7	1	14	0.145513	0	1	14	0.150596	0	3	40	0.359077	0	5	63	0.675605	0
	x_0^8	2	27	0.331784	0	2	27	0.393617	0	1	12	0.090385	0	1	9	0.068757	0
100,000	x_0^1	1	14	0.247586	0	1	14	0.29132	0	1	14	0.229091	0	2	26	0.516286	0
	x_0^2	1	14	0.238248	0	1	14	0.261577	0	3	40	0.586183	0	2	24	0.349797	0
	x_0^3	1	14	0.252227	0	1	14	0.284209	0	1	14	0.230378	0	2	24	0.496748	0
	x_0^4	1	14	0.210766	0	1	14	0.314589	0	Fail	Fail	Fail	Fail	5	63	1.300943	0
	x_0^5	1	14	0.26294	0	1	14	0.294359	0	3	40	0.735556	0	5	63	1.678595	0
	x_0^6	1	14	0.204092	0	1	14	0.265777	0	Fail	Fail	Fail	Fail	2	24	0.454425	0
	x_0^7	1	14	0.234871	0	1	14	0.287143	0	3	40	0.70266	0	5	63	1.259794	0
	x_0^8	2	27	0.66677	0	2	27	0.815461	0	1	12	0.134747	0	1	9	0.175668	0

Table 4. Numerical comparison of SDYCG algorithm versus DFSP [24], and MSGP algorithms [23].

Problem 4		SDYCG(1)				SDYCG(2)				MSGP				DFSP			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
500	x_0^1	1	14	0.014439	0	1	14	0.006118	0	7	40	0.007555	0	21	54	0.012805	3.36×10^{-10}
	x_0^2	2	27	0.005311	0	2	27	0.007617	0	5	52	0.009625	0	6	66	0.012711	0
	x_0^3	1	14	0.004658	0	1	14	0.005807	0	6	35	0.006392	0	21	45	0.013598	3.34×10^{-10}
	x_0^4	2	27	0.006039	0	2	27	0.00744	0	93	1167	0.126211	9.31×10^{-10}	5	46	0.010058	0
	x_0^5	2	27	0.007802	0	2	27	0.007552	0	92	1154	0.14209	9.1×10^{-10}	6	60	0.012141	0
	x_0^6	2	27	0.006542	0	2	27	0.007702	0	31	385	0.044429	6.01×10^{-10}	5	50	0.010774	0
	x_0^7	2	27	0.007244	0	2	27	0.007673	0	92	1154	0.256753	9.1×10^{-10}	6	60	0.012402	0
	x_0^8	1	3	0.003637	0	1	3	0.003558	0	1	3	0.002732	0	1	3	0.003719	0
1000	x_0^1	1	14	0.007838	0	1	14	0.00718	0	8	56	0.021892	0	21	54	0.021797	4.76×10^{-10}
	x_0^2	2	27	0.009188	0	2	27	0.009234	0	5	52	0.021223	0	6	66	0.016802	0
	x_0^3	1	14	0.005851	0	1	14	0.006695	0	5	25	0.013147	0	21	45	0.016618	4.74×10^{-10}
	x_0^4	2	27	0.007817	0	2	27	0.009626	0	107	1347	0.226685	9.14×10^{-10}	5	46	0.013843	0
	x_0^5	2	27	0.008671	0	2	27	0.010016	0	107	1348	0.373635	9.05×10^{-10}	6	59	0.016737	0
	x_0^6	2	27	0.008867	0	2	27	0.008653	0	31	385	0.118865	5.98×10^{-10}	5	50	0.01304	0
	x_0^7	2	27	0.008939	0	2	27	0.009434	0	107	1348	0.222399	9.05×10^{-10}	6	59	0.01518	0
	x_0^8	1	3	0.004102	0	1	3	0.004062	0	1	3	0.003508	0	1	3	0.003979	0
10,000	x_0^1	1	14	0.022733	0	1	14	0.025232	0	7	33	0.076215	0	22	56	0.100193	4.57×10^{-10}
	x_0^2	2	27	0.034866	0	2	27	0.039607	0	5	52	0.058953	0	6	66	0.080801	0
	x_0^3	1	14	0.019616	0	1	14	0.023444	0	6	29	0.04933	0	22	47	0.081846	4.61×10^{-10}
	x_0^4	2	27	0.034486	0	2	27	0.03894	0	134	1693	2.21205	9.46×10^{-10}	6	59	0.077072	0
	x_0^5	2	27	0.034804	0	2	27	0.039725	0	134	1693	2.177277	9.54×10^{-10}	6	59	0.094729	0
	x_0^6	2	27	0.033743	0	2	27	0.038146	0	31	385	0.509865	5.96×10^{-10}	5	50	0.060477	0
	x_0^7	2	27	0.03721	0	2	27	0.04104	0	134	1693	2.145371	9.54×10^{-10}	6	59	0.083734	0
	x_0^8	1	3	0.008865	0	1	3	0.009299	0	1	3	0.008254	0	1	3	0.007881	0

Table 4. Cont.

Problem 4		SDYCG(1)				SDYCG(2)				MSGP				DFSP			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
50,000	x_0^1	1	14	0.076932	0	1	14	0.087556	0	7	33	0.202677	0	23	58	0.379123	3.14×10^{-10}
	x_0^2	2	27	0.135061	0	2	27	0.141828	0	5	52	0.289193	0	6	66	0.33015	0
	x_0^3	1	14	0.070861	0	1	14	0.080542	0	6	28	0.17549	0	23	49	0.333141	3.22×10^{-10}
	x_0^4	2	27	0.14223	0	2	27	0.147057	0	143	1813	9.16387	9.66×10^{-10}	6	59	0.299475	0
	x_0^5	2	27	0.140522	0	2	27	0.14851	0	143	1813	8.803139	9.68×10^{-10}	6	59	0.319691	0
	x_0^6	2	27	0.146285	0	2	27	0.139833	0	31	385	1.871516	5.96×10^{-10}	5	50	0.258848	0
	x_0^7	2	27	0.141941	0	2	27	0.146955	0	143	1813	7.50336	9.68×10^{-10}	6	59	0.311193	0
	x_0^8	1	3	0.027715	0	1	3	0.022438	0	1	3	0.017293	0	1	3	0.024975	0
100,000	x_0^1	1	14	0.160405	0	1	14	0.159961	0	7	33	0.360256	0	23	58	0.777767	4.51×10^{-10}
	x_0^2	2	27	0.268115	0	2	27	0.277038	0	5	52	0.534201	0	6	66	0.801637	0
	x_0^3	1	14	0.136486	0	1	14	0.156991	0	6	28	0.31303	0	23	49	0.728779	4.7×10^{-10}
	x_0^4	2	27	0.27761	0	2	27	0.297949	0	147	1865	16.4068	9.46×10^{-10}	6	59	0.729595	0
	x_0^5	2	27	0.256903	0	2	27	0.285998	0	147	1865	16.00851	9.47×10^{-10}	6	59	0.739932	0
	x_0^6	2	27	0.263076	0	2	27	0.283114	0	31	385	2.045793	5.96×10^{-10}	5	50	0.600872	0
	x_0^7	2	27	0.236317	0	2	27	0.287023	0	147	1865	13.37062	9.47×10^{-10}	6	59	0.748439	0
	x_0^8	1	3	0.040207	0	1	3	0.041483	0	1	3	0.042568	0	1	3	0.045916	0

Table 5. Numerical comparison of SDYCG algorithm versus DFSP [24], and MSGP algorithms [23].

Problem 5		SDYCG(1)				SDYCG(2)				MSGP				DFSP			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
500	x_0^1	1	14	0.021506	0	1	14	0.007261	0	6	41	0.011468	0	9	85	0.024421	9.27×10^{-11}
	x_0^2	1	14	0.008549	0	1	14	0.007703	0	8	92	0.019629	0	6	71	0.018679	0
	x_0^3	1	14	0.006508	0	1	14	0.007023	0	6	45	0.019822	0	9	85	0.025187	4.01×10^{-11}
	x_0^4	1	14	0.006233	0	1	14	0.00694	0	120	1522	0.293443	9.13×10^{-10}	5	58	0.018668	0
	x_0^5	1	14	0.006146	0	1	14	0.006566	0	121	1540	0.282529	9.12×10^{-10}	5	58	0.017161	0
	x_0^6	1	14	0.005834	0	1	14	0.006413	0	71	901	0.215492	8.85×10^{-10}	6	71	0.020237	0
	x_0^7	1	14	0.006965	0	1	14	0.007415	0	121	1540	0.499558	9.12×10^{-10}	5	58	0.017493	0
	x_0^8	1	3	0.004355	0	1	3	0.00417	0	1	3	0.003339	0	1	3	0.005034	0
1000	x_0^1	1	14	0.007873	0	1	14	0.007946	0	6	35	0.022229	0	9	85	0.030314	1.38×10^{-10}
	x_0^2	1	14	0.008308	0	1	14	0.007937	0	8	92	0.044925	0	6	71	0.027931	0
	x_0^3	1	14	0.00827	0	1	14	0.008861	0	6	42	0.013574	0	9	85	0.03155	5.69×10^{-11}
	x_0^4	1	14	0.008426	0	1	14	0.008309	0	141	1792	0.845762	9.32×10^{-10}	5	58	0.022619	0
	x_0^5	1	14	0.008186	0	1	14	0.008268	0	136	1733	0.860175	9.81×10^{-10}	5	58	0.020516	0
	x_0^6	1	14	0.007483	0	1	14	0.007944	0	71	901	0.225184	9.41×10^{-10}	6	71	0.026813	0
	x_0^7	1	14	0.006036	0	1	14	0.007867	0	136	1733	0.749835	9.81×10^{-10}	5	58	0.024208	0
	x_0^8	1	3	0.004891	0	1	3	0.004726	0	1	3	0.003693	0	1	3	0.004639	0
10,000	x_0^1	1	14	0.038383	0	1	14	0.035689	0	6	35	0.132843	0	9	85	0.219583	5.85×10^{-10}
	x_0^2	1	14	0.035904	0	1	14	0.034471	0	8	92	0.174444	0	6	71	0.170278	0
	x_0^3	1	14	0.038673	0	1	14	0.034298	0	7	45	0.090831	0	9	85	0.243069	1.85×10^{-10}
	x_0^4	1	14	0.034179	0	1	14	0.035263	0	Fail	Fail	Fail	Fail	5	58	0.140085	0
	x_0^5	1	14	0.038989	0	1	14	0.037575	0	145	1844	5.10125	9.92×10^{-10}	5	58	0.144594	0
	x_0^6	1	14	0.034517	0	1	14	0.036054	0	71	901	2.172107	9.93×10^{-10}	6	71	0.166404	0
	x_0^7	1	14	0.035881	0	1	14	0.03835	0	145	1844	4.761964	9.92×10^{-10}	5	58	0.138295	0
	x_0^8	1	3	0.01122	0	1	3	0.012854	0	1	3	0.01033	0	1	3	0.015678	0

Table 5. Cont.

Problem 5		SDYCG(1)				SDYCG(2)				MSGP				DFSP			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
50,000	x_0^1	1	14	0.149694	0	1	14	0.149678	0	7	40	0.601808	0	10	96	1.005554	6.02×10^{-11}
	x_0^2	1	14	0.147631	0	1	14	0.152282	0	8	92	0.824379	0	6	71	0.75667	0
	x_0^3	1	14	0.135184	0	1	14	0.149976	0	7	39	0.578662	0	9	85	0.940741	4.31×10^{-10}
	x_0^4	1	14	0.160898	0	1	14	0.144812	0	Fail	Fail	Fail	Fail	5	58	0.624748	0
	x_0^5	1	14	0.142256	0	1	14	0.154083	0	157	1995	18.94711	9.08×10^{-10}	5	58	0.612333	0
	x_0^6	1	14	0.155172	0	1	14	0.144641	0	71	901	5.746775	9.98×10^{-10}	6	71	0.762054	0
	x_0^7	1	14	0.140437	0	1	14	0.151539	0	157	1995	17.56905	9.08×10^{-10}	5	58	0.617805	0
	x_0^8	1	3	0.035545	0	1	3	0.038543	0	1	3	0.030092	0	1	3	0.040141	0
100,000	x_0^1	1	14	0.273227	0	1	14	0.29604	0	7	39	0.709299	0	10	96	1.96393	1.05×10^{-10}
	x_0^2	1	14	0.305759	0	1	14	0.28919	0	8	92	2.144781	0	6	71	2.116682	0
	x_0^3	1	14	0.265332	0	1	14	0.282444	0	7	40	0.935991	0	9	85	1.804135	6.28×10^{-10}
	x_0^4	1	14	0.280076	0	1	14	0.290106	0	Fail	Fail	Fail	Fail	5	58	1.264132	0
	x_0^5	1	14	0.26493	0	1	14	0.287867	0	Fail	Fail	Fail	Fail	5	58	1.640461	0
	x_0^6	1	14	0.272053	0	1	14	0.287873	0	71	901	19.87081	9.99×10^{-10}	6	71	1.57878	0
	x_0^7	1	14	0.276199	0	1	14	0.293467	0	Fail	Fail	Fail	Fail	5	58	1.646327	0
	x_0^8	1	3	0.066375	0	1	3	0.070481	0	1	3	0.055997	0	1	3	0.074297	0

Table 6. Numerical comparison of SDYCG algorithm versus DFSP [24], and MSGP algorithms [23].

Problem 6		SDYCG(1)				SDYCG(2)				MSGP				DFSP			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
500	x_0^1	1	14	0.022624	0	1	14	0.006415	0	5	31	0.009245	0	21	45	0.018109	8.78×10^{-10}
	x_0^2	2	27	0.008015	0	2	27	0.00818	0	5	54	0.013084	0	6	56	0.014451	0
	x_0^3	1	14	0.005822	0	1	14	0.005567	0	5	30	0.009862	0	20	42	0.016839	6.95×10^{-10}
	x_0^4	2	27	0.007301	0	2	27	0.009212	0	Fail	Fail	Fail	Fail	5	43	0.01177	0
	x_0^5	2	27	0.008102	0	2	27	0.008367	0	Fail	Fail	Fail	Fail	6	56	0.013896	0
	x_0^6	2	27	0.007734	0	2	27	0.008301	0	136	1704	0.29359	9.82×10^{-10}	6	57	0.013973	0
	x_0^7	2	27	0.008453	0	2	27	0.008059	0	Fail	Fail	Fail	Fail	6	56	0.013635	0
	x_0^8	2	27	0.008123	0	2	27	0.008796	0	1	7	0.003092	0	1	6	0.004603	0
1000	x_0^1	1	14	0.007487	0	1	14	0.007598	0	6	43	0.014542	0	22	47	0.022623	3.76×10^{-10}
	x_0^2	2	27	0.009685	0	2	27	0.00896	0	5	54	0.016321	0	6	56	0.018095	0
	x_0^3	1	14	0.007171	0	1	14	0.006439	0	6	39	0.014019	0	20	42	0.021935	9.87×10^{-10}
	x_0^4	2	27	0.010274	0	2	27	0.00929	0	Fail	Fail	Fail	Fail	6	56	0.018509	0
	x_0^5	2	27	0.010131	0	2	27	0.009105	0	Fail	Fail	Fail	Fail	6	57	0.0188	0
	x_0^6	2	27	0.011014	0	2	27	0.01066	0	136	1704	0.431967	9.88×10^{-10}	6	57	0.018837	0
	x_0^7	2	27	0.010828	0	2	27	0.009933	0	Fail	Fail	Fail	Fail	6	57	0.019487	0
	x_0^8	2	27	0.010742	0	2	27	0.010966	0	1	7	0.003346	0	1	6	0.004771	0
10,000	x_0^1	1	14	0.027581	0	1	14	0.029092	0	6	37	0.06886	0	23	49	0.113196	3.84×10^{-10}
	x_0^2	2	27	0.045375	0	2	27	0.046186	0	5	54	0.092398	0	6	56	0.093238	0
	x_0^3	1	14	0.027145	0	1	14	0.02555	0	6	31	0.061397	0	21	44	0.101149	9.68×10^{-10}
	x_0^4	2	27	0.051507	0	2	27	0.049899	0	Fail	Fail	Fail	Fail	6	56	0.096773	0
	x_0^5	2	27	0.055737	0	2	27	0.046071	0	Fail	Fail	Fail	Fail	6	56	0.096053	0
	x_0^6	2	27	0.048279	0	2	27	0.046802	0	136	1704	2.623827	9.93×10^{-10}	6	57	0.093759	0
	x_0^7	2	27	0.053576	0	2	27	0.047814	0	Fail	Fail	Fail	Fail	6	56	0.098486	0
	x_0^8	2	27	0.055324	0	2	27	0.053102	0	1	7	0.014141	0	1	6	0.015449	0

Table 6. Cont.

Problem 6		SDYCG(1)				SDYCG(2)				MSGP				DFSP			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
50,000	x_0^1	1	14	0.10161	0	1	14	0.101955	0	6	32	0.236257	0	23	49	0.445818	9.78×10^{-10}
	x_0^2	2	27	0.188288	0	2	27	0.18744	0	5	54	0.375475	0	6	56	0.382023	0
	x_0^3	1	14	0.102065	0	1	14	0.097173	0	5	23	0.176346	0	22	46	0.459486	6.89×10^{-10}
	x_0^4	2	27	0.201473	0	2	27	0.190649	0	Fail	Fail	Fail	Fail	5	43	0.311194	0
	x_0^5	2	27	0.194281	0	2	27	0.188893	0	Fail	Fail	Fail	Fail	5	43	0.321151	0
	x_0^6	2	27	0.189766	0	2	27	0.178816	0	136	1704	9.810722	9.94×10^{-10}	6	57	0.393791	0
	x_0^7	2	27	0.188747	0	2	27	0.189613	0	Fail	Fail	Fail	Fail	5	43	0.311808	0
	x_0^8	2	27	0.202092	0	2	27	0.198067	0	1	7	0.054461	0	1	6	0.048922	0
100,000	x_0^1	1	14	0.200437	0	1	14	0.207021	0	6	31	0.315882	0	24	51	0.93727	4.57×10^{-10}
	x_0^2	2	27	0.375252	0	2	27	0.374575	0	5	54	0.504352	0	6	56	0.931854	0
	x_0^3	1	14	0.1908	0	1	14	0.183819	0	5	23	0.440007	0	23	48	0.872302	3.05×10^{-10}
	x_0^4	2	27	0.355745	0	2	27	0.370361	0	Fail	Fail	Fail	Fail	5	43	0.709785	0
	x_0^5	2	27	0.356704	0	2	27	0.375388	0	Fail	Fail	Fail	Fail	5	43	0.653336	0
	x_0^6	2	27	0.36915	0	2	27	0.362355	0	136	1704	18.08103	9.94×10^{-10}	6	57	0.908398	0
	x_0^7	2	27	0.378124	0	2	27	0.371835	0	Fail	Fail	Fail	Fail	5	43	0.64292	0
	x_0^8	2	27	0.432772	0	2	27	0.448033	0	1	7	0.090026	0	1	6	0.091768	0

Moreover, for Problem 3, the proposed algorithm won in terms of the minimum ITER, FEV, and CPU time. In addition, the DFSP algorithm showed a high level of efficiency for the three parameters, but MSGP uniformly failed for three initial guesses across all five dimensions. For Problem 4, all the algorithms successfully and efficiently solved all the initial guesses. The SDYCG algorithm was the winner, followed by the DFSP algorithm and, lastly, the MSGP algorithm, for the three parameters. A similar observation can be made for Problems 5–6; it can also be seen the MSGP algorithm is relatively dimension-dependent, as the failure for the initial guesses increases as the dimension increases.

Furthermore, to ease the numerical comparisons in the above tables, we used the well-known Dolan and Moré technique [28] performance profile method. The three figures were plotted for ITER, TIME, and FVAL. Figures 1–3 show that, on average, the SDYCG algorithm has less ITER, computing time, and the number of function evaluations than the DFSP and MSGP algorithms.

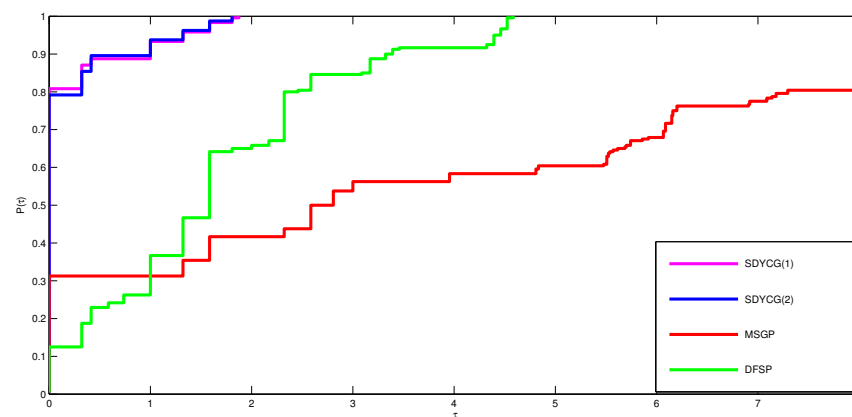


Figure 1. Performance profile of SDYCG algorithm versus MSGP algorithms [23], and DFSP [24] for number of iterations.

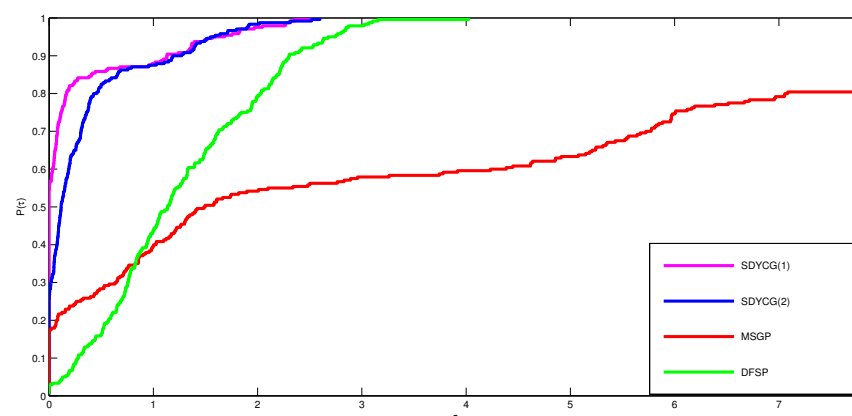


Figure 2. Performance profile of SDYCG algorithm versus MSGP algorithms [23], and DFSP [24] for the CPU time.

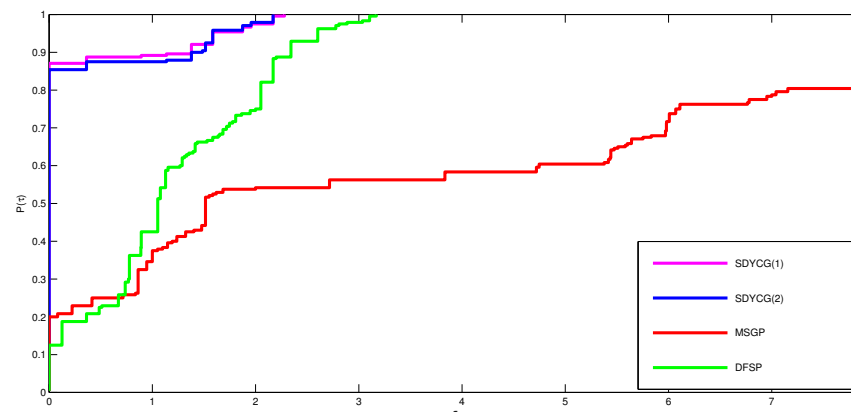


Figure 3. Performance profile of SDYCG algorithm versus MSGP algorithms [23], and DFSP [24] for number of function evaluations.

4.2. Application in Signal Recovery

This subsection considers the pursuit of denoising problem. Let x^* be the original sparse signal, and $f \in \mathbb{R}^s$ be an observation satisfying

$$m = Yx^*. \quad (57)$$

The standard sparse l_1 -regularization problem sparse term is represented as

$$\min w\|x\|_1 + \frac{1}{2}\|Yx - m\|_2^2, \quad (58)$$

this problem arises in compressive sensing, where w is a nonnegative parameter, $m \in \mathbb{R}^k$ is an observation, $Y \in \mathbb{R}^{k \times n}$ ($k \ll n$) is a linear operator, $x \in \mathbb{R}^n$, and $\|\cdot\|_1$, $\|\cdot\|_2$ represent the l_1 and l_2 norms, respectively, see [2,29–31]. The problem (58) can be transformed to the following monotone nonlinear system

$$F(u) = \min\{u, Mu + r\} = 0, \quad (59)$$

where $u = \begin{pmatrix} l \\ z \end{pmatrix}$, $r = we_{2n} + \begin{pmatrix} -Y^T m \\ Y^T m \end{pmatrix}$, $M = \begin{pmatrix} Y^T Y & -Y^T Y \\ -Y^T Y & Y^T Y \end{pmatrix}$, $e_n = (1, 1, \dots, 1)^T \in \mathbb{R}^n$ and $x = l - z$, for some $l \geq 0$ and $z \geq 0$. The function (59) is Lipschitz-continuous, and the monotone, for more detail information about the transformation of (59) from (58), see [32,33]. This was omitted to avoid repeats. The SDYCG algorithm will be used to address this problem. Similar methods have been employed to deal with this problem; see [34–36].

All codes in this work are written in Matlab R2014a and run on an HP core i5, 8th Gen personal computer. We carried out two restoration experiments: signal and image restoration. Further, we employed the direction (15) for signal restoration and the direction (24) for image restoration. Mean square error (MSE) is defined as

$$\text{MSE} = \frac{1}{n}\|x^* - x\|^2,$$

and the recovered image signal-to-ratio (SNR) as

$$\text{SNR} = 20 \times \log_{10} \left(\frac{\|x^*\|}{\|x - x^*\|} \right).$$

The primary purpose of the signal restoration experiment is to reconstruct a sparse signal with a length n from observations of length k . We initialized the following free

variables: $a = 0.4$, $g = 0.00001$, $c = 1$, and $b = 0.9$. We conducted the experiment on the signal of length $n = 4096$ from observations of length $k = 1024$ using 128 original signal's randomly nonzero components. Y is the Gaussian matrix produced by the MATLAB instruction $rand(s, n)$. Furthermore, the measurement w is noise distributed, that is, $w = Yx * + \theta$, where θ is Gaussian noise distributed normally with mean 0 and variance 10^{-4} . We used $x_0 = Y^T m$, $f(x) = w\|x\|_1 + \frac{1}{2}\|Yx - m\|_2^2$ as the merit function and stopped the iterations when

$$\frac{f_k - f_{k-1}}{f_{k-1}} < 10^{-4}. \quad (60)$$

In comparison to the PCG [37] and MSP [38] algorithms, we used the SDYCG algorithm and restored the signal to almost its original form in a few iterations, MSE, and CPU time. We run each code from the same starting point and used the same continuation strategy on parameter w . The convergence behavior of each algorithm was observed to achieve an accurate solution. As shown in Figures 4 and 5, which has been replicated more than a hundred times with remarkably similar results. The SDYCG algorithm recovers the sparse signal with a minimal processing time and minimal MSE error. Furthermore, for the objective function and MSE, the SDYCG method reduces faster than the PCG and MSP algorithms.

In addition, for the image restoration experiment, we used Y as a partial DWT matrix, with s rows chosen at random from the $n \times n$ DWT matrix. This type of encoding matrix Y does not need storage and allows for quick matrix-vector multiplications of Y and Y^T . As a result, it may be tested on large images without the need to store any matrix. $a = 0.2$, $g = 0.000001$, $c = 9$, and $b = 0.4$ are the parameter values. The initial parameter values are $a = 0.2$, $g = 0.000001$, $c = 9$, and $b = 0.4$. In this experiment, we utilized the standard Lena colour image with a size of 512×512 and the colour baby image with a size of 256×256 . In contrast, for performance comparison, we considered the well-known PSGM [34], CGD [35], and TPRP [36] iterative algorithms. The iteration procedure of all algorithms began at $Y^T m$ and ended when (60) was less than 10^{-5} . Figures 6 and 7 show the original, blurred, and reconstructed images generated by each algorithm. As can be seen from the figures, all of the algorithms considered generated images of similar quality. However SDYCG was quicker. As a result, we can infer that SDYCG is the winner.

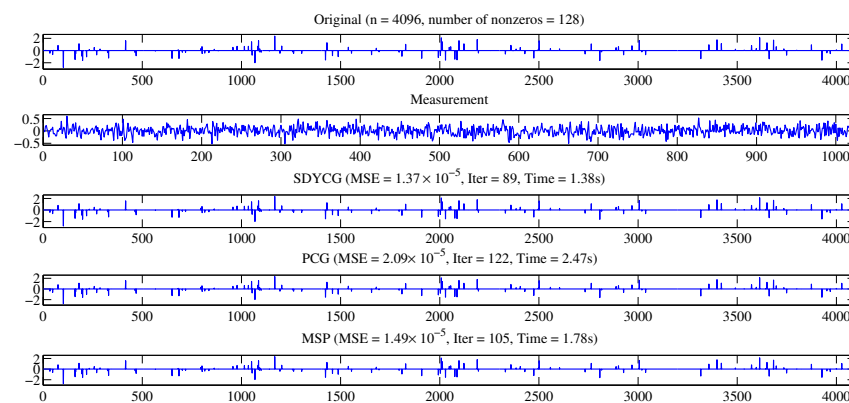


Figure 4. The original picture, measurement, and recovered signals using SDYCG, PCG, and MSP algorithms are shown from top to bottom.

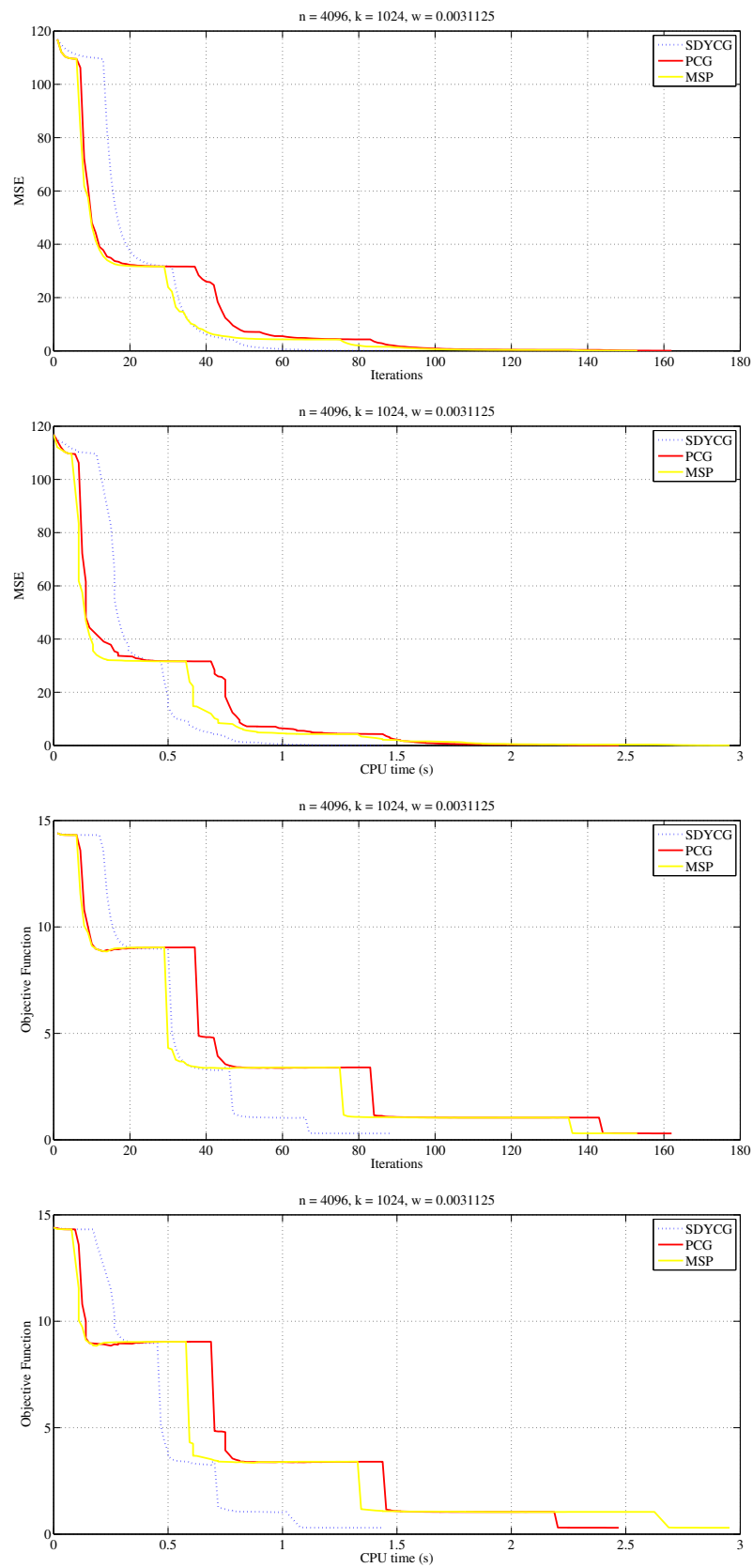


Figure 5. Comparison of the SDYCG, PCG, and MSP algorithms. The x -axes represent the number of iterations and the CPU time in seconds, respectively. MSE and objective function values are represented on the y -axes.

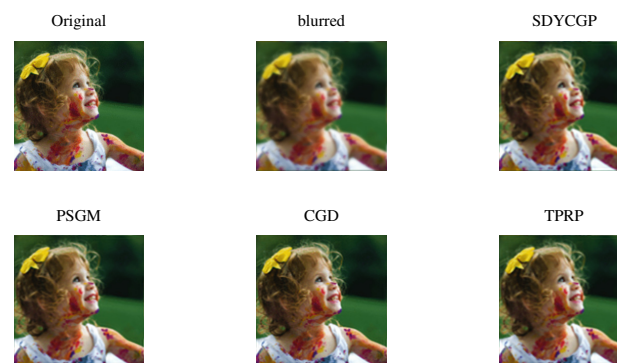


Figure 6. Original image, blurred image, restored image by SDYCG with ITER = 10, $\text{obj} = 8.128 \times 10^5$, TIME = 3.27, $\text{MSE} = 6.3421 \times 10^1$, SNR = 20.65, SSIM = 0.90, restored image by PSGM with ITER = 45, $\text{obj} = 7.365 \times 10^5$, TIME = 14.00, $\text{MSE} = 3.8961 \times 10^1$, SNR = 22.77, SSIM = 0.93, restored image by CGD with ITER = 713, $\text{obj} = 7.268 \times 10^5$, TIME = 167.89, $\text{MSE} = 4.4565 \times 10^1$, SNR = 22.19, SSIM = 0.93, and restored image by TPRP with ITER = 104, $\text{obj} = 7.445 \times 10^5$, TIME = 3049.66, $\text{MSE} = 3.8931 \times 10^1$, SNR = 22.77, SSIM = 0.92



Figure 7. Original image, blurred image, restored image by SDYCG with ITER = 25, $\text{obj} = 1.251 \times 10^6$, TIME = 5.67, $\text{MSE} = 6.7866 \times 10^1$, SNR = 22.55, SSIM = 0.85, restored image by PSGM with ITER = 29, $\text{obj} = 1.222 \times 10^6$, TIME = 6.03, $\text{MSE} = 5.8080 \times 10^1$, SNR = 23.23, SSIM = 0.87, restored image by CGD with ITER = 491, $\text{obj} = 1.214 \times 10^6$, TIME = 106.47, $\text{MSE} = 6.0494 \times 10^1$, SNR = 23.05, SSIM = 0.88, and restored image by TPRP with ITER = 41, $\text{obj} = 1.256 \times 10^6$, TIME = 4069.64, $\text{MSE} = 7.0079 \times 10^1$, SNR = 22.41, SSIM = 0.85

5. Conclusions

We suggested a scaled DY projection-based CG algorithm for solving convex constraint nonlinear monotone equations, with applications for signal and image restoration problems. Independent of the line search strategy that was employed, the proposed directions satisfied the sufficient descent condition. We also offer two alternate ways of determining the scaling parameter at each iteration, namely by tending the proposed direction to approach the quasi-Newton direction and by utilizing the popular Barzilai–Borwein [22] technique. The proposed algorithm's global convergence result was established using some mild assumptions. The robustness of the proposed algorithm was demonstrated by its ability to solve large-scale monotone nonlinear systems with convex constraints. These two proposed directions may be employed in all fields of CG method application, such as unconstrained optimization problems, control motion of robotic manipulators, etc.

Author Contributions: Methodology, software, supervision, A.A.; validation, writing-original draft preparation, investigation, J.S.; visualization, project administration, formal analysis, H.E.; writing-review and editing, funding acquisition, resources, P.J.; writing-review and editing, data curation, S.K.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Taif University Researches Supporting Project number (TURSP-2020/326), Taif University, Taif, Saudi Arabia.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Dai, Z.; Zhou, H.; Wen, F.; He, S. Efficient predictability of stock return volatility: The role of stock market implied volatility. *N. Am. J. Econ. Finance* **2020**, *52*, 101174. [\[CrossRef\]](#)
2. Figueiredo, M.A.; Nowak, R.D.; Wright, S.J. Gradient projection for sparse reconstruction, application to compressed sensing and other inverse problems. *IEEE J. Sel. Top. Signal Process.* **2007**, *1*, 586–597. [\[CrossRef\]](#)
3. Iusem, A.N.; Solodov, M.V. Newton-type methods with generalized distances for constrained optimization. *Optimization* **1997**, *41*, 257–278. [\[CrossRef\]](#)
4. Zhao, Y.B.; Li, D.H. Monotonicity of fixed point and normal mapping associated with variational inequality and its application. *SIAM J. Optim.* **2001**, *4*, 962–973. [\[CrossRef\]](#)
5. Ortega, J.M.; Rheinboldt, W.C. *Iterative Solution of Nonlinear Equations in Several Variables*; Academic Press: Cambridge, MA, USA, 1970.
6. Zhou, G.; Toh, K.C. Superlinear convergence of a Newton-type algorithm for monotone equations. *J. Optimiz. Theory App.* **2005**, *125*, 205–221. [\[CrossRef\]](#)
7. Zhou, W.J.; Li, D.H. A globally convergent BFGS method for nonlinear monotone equations without any merit functions. *Math. Comput.* **2008**, *77*, 2231–2240. [\[CrossRef\]](#)
8. Sabi'u, J.; Shah, A.; Waziri, M.Y.; Ahmed, K. Modified Hager-Zhang conjugate gradient methods via singular value analysis for solving monotone nonlinear equations with convex constraint. *Int. J. Comput. Methods* **2020**, *18*, 2050043. [\[CrossRef\]](#)
9. Sabi'u, J.; Shah, A.; Waziri, M.Y. A modified Hager-Zhang conjugate gradient method with optimal choices for solving monotone nonlinear equations. *Int. J. Comput. Math.* **2021**, *99*, 1–23. [\[CrossRef\]](#)
10. Sabi'u, J.; Shah, A. An efficient three-term conjugate gradient-type algorithm for monotone nonlinear equations. *RAIRO-Oper. Res.* **2021**, *55*, 1113. [\[CrossRef\]](#)
11. Waziri, M.Y.; Ahmed, K.; Sabi'u, J.; Halilu, A.S. Enhanced Dai-Liao conjugate gradient methods for systems of monotone nonlinear equations. *SeMA J.* **2021**, *78*, 15–51. [\[CrossRef\]](#)
12. Abubakar, A.B.; Sabi'u, J.; Kumam, P.; Shah, A. Solving nonlinear monotone operator equations via modified sr1 update. *J. Appl. Math. Comput.* **2021**, *67*, 1–31. [\[CrossRef\]](#)
13. Waziri, M.Y.; Hungu, K.A.; Sabi'u, J. Descent Perry conjugate gradient methods for systems of monotone nonlinear equations. *Numer. Algorithms* **2020**, *85*, 763–785. [\[CrossRef\]](#)
14. Waziri, M.Y.; Ahmed, K.; Sabi'u, J. A Dai-Liao conjugate gradient method via modified secant equation for system of nonlinear equations. *Arab. J. Math.* **2020**, *9*, 443–457. [\[CrossRef\]](#)
15. Sabi'u, J.; Shah, A.; Waziri, M.Y. Two optimal Hager-Zhang conjugate gradient methods for solving monotone nonlinear equations. *Appl. Numer. Math.* **2020**, *153*, 217–233. [\[CrossRef\]](#)
16. Dai, Y.H.; Yuan, Y. A nonlinear conjugate gradient method with a strong global convergence property. *SIAM J. Optim.* **1999**, *10*, 177–182. [\[CrossRef\]](#)
17. Waziri, M.Y.; Ahmed, K. Two Descent Dai-Yuan Conjugate Gradient Methods for Systems of Monotone Nonlinear Equations. *J. Sci. Comput.* **2022**, *90*, 1–53. [\[CrossRef\]](#)
18. Kambheera, A.; Ibrahim, A.H.; Muhammad, A.B.; Abubakar, A.B.; Hassan, B.A. Modified Dai-Yuan Conjugate Gradient Method with Sufficient Descent Property for Nonlinear Equations. *Thai J. Math.* **2022**, 145–167. Available online: <http://thaijmath.in.cmu.ac.th/index.php/thaijmath/article/viewFile/6026/354355047> (accessed on 6 June 2022).
19. Aji, S.; Kumam, P.; Awwal, A.M.; Yahaya, M.M.; Sitthithakerngkiet, K. An efficient DY-type spectral conjugate gradient method for system of nonlinear monotone equations with application in signal recovery. *AIMS Math.* **2021**, *6*, 8078–8106. [\[CrossRef\]](#)
20. Abdullahi, H.; Awasthi, A.K.; Waziri, M.Y.; Halilu, A.S. Descent three-term DY-type conjugate gradient methods for constrained monotone equations with application. *Comput. Appl. Math.* **2022**, *41*, 1–28. [\[CrossRef\]](#)
21. Sabi'u, J.; Aremu, K.O.; Althobaiti, A.; Shah, A. Scaled three-term conjugate gradient methods for solving monotone equations with application. *Symmetry* **2022**, *14*, 936. [\[CrossRef\]](#)
22. Barzilai, J.; Borwein, J.M. Two-point step size gradient methods. *IMA J. Numer. Anal.* **1988**, *8*, 141–148. [\[CrossRef\]](#)
23. Zheng, L.; Yang, L.; Liang, Y. A modified spectral gradient projection method for solving non-linear monotone equations with convex constraints and its application. *IEEE Access* **2020**, *8*, 92677–92686. [\[CrossRef\]](#)
24. Amini, K.; Famarazi, P.; Bahrami, S. A spectral conjugate gradient projection algorithm to solve the large-scale system of monotone nonlinear equations with application to compressed sensing. *Int. J. Comp. Math.* **2022**, 1–18. <https://doi.org/10.1080/00207160.2022.2047180>. [\[CrossRef\]](#)
25. La Cruz, W.; Martinez, J.; Raydan, M. Spectral residual method without gradient information for solving large-scale nonlinear systems of equations. *Math. Comput.* **2006**, *75*, 1429–1448. [\[CrossRef\]](#)

26. Halilu, A.S.; Majumder, A.; Waziri, M.Y.; Ahmed, K. Signal recovery with convex constrained nonlinear monotone equations through conjugate gradient hybrid approach. *Math Comput. Simul.* **2021**, *187*, 520–539. [[CrossRef](#)]
27. Liu, J.; Duan, Y. Two spectral gradient projection methods for constrained equations and their linear convergence rate. *J. Inequal. Appl.* **2015**, *2015*, 1–13. [[CrossRef](#)]
28. Dolan, E.D.; Moré, J.J. Benchmarking optimization software with performance profiles. *Math. Program.* **2002**, *91*, 201–213. [[CrossRef](#)]
29. Hale, E.T.; Yin, W.; Zhang, Y. A fixed-point continuation method for l_1 regularized minimization with applications to compressed sensing. *SIAM J. Optim.* **2008**, *19*, 1107–1130. [[CrossRef](#)]
30. Beck, A.; Teboulle, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imag. Sci.* **2009**, *2*, 183–202. [[CrossRef](#)]
31. Van den Berg, E.; Friedlander, M.P. Probing the pareto frontier for basis pursuit solutions. *SIAM J. Sci. Comput.* **2008**, *31*, 890–912. [[CrossRef](#)]
32. Xiao, Y.H.; Wang, Q.Y.; Hu, Q.J. Non-smooth equations based method for l_1 -norm problems with applications to compressed sensing. *Nonlinear Anal. TMA* **2011**, *74*, 3570–3577. [[CrossRef](#)]
33. Pang, J.S. Inexact Newton methods for the nonlinear complementarity problem. *Math. Program.* **1986**, *36*, 54–71. [[CrossRef](#)]
34. Awwal, A.M.; Kumam, P.; Mohammad, H.; Watthayu, W.; Abubakar, A.B. A Perry-type derivative-free algorithm for solving nonlinear system of equations and minimizing l_1 regularized problem. *Optimization* **2021**, *70*, 1231–1259. [[CrossRef](#)]
35. Xiao, Y.H.; Zhu, H. A conjugate gradient method to solve convex constrained monotone equations with applications in compressive sensing. *J. Math. Anal. Appl.* **2013**, *405*, 310–319. [[CrossRef](#)]
36. Ibrahim, A.H.; Deepho, J.; Abubakar, A.B.; Adamu, A. A three-term Polak-Ribière-Polyak derivative-free method and its application to image restoration. *Sci. Afr.* **2021**, *13*, e00880. [[CrossRef](#)]
37. Liu, J.K.; Lia, S.J. A projection method for convex constrained monotone nonlinear equations with applications. *Comput. Math. Appl.* **2015**, *70*, 2442–2453. [[CrossRef](#)]
38. Abubakar, A.B.; Kumam, P.; Mohammad, H.; Awwal, A.M. A Barzilai-Borwein gradient projection method for sparse signal and blurred image restoration. *J. Franklin Inst.* **2020**, *357*, 7266–7285. [[CrossRef](#)]