symmetry

MDPI

Article

Scaled Three-Term Conjugate Gradient Methods for Solving Monotone Equations with Application

Jamilu Sabi'u [1,2,*], Kazeem Olalekan Aremu [3,4], Ali Althobaiti [5] and Abdullah Shah [6]

[1] Department of Mathematics, Faculty of Science, Yusuf Maitama Sule University, Kano P.M.B. 3099, Nigeria
[2] Numerical Optimization Research Group, Bayero University, Gwarzo Road, Kano P.M.B. 3011, Nigeria
[3] Department of Mathematics and Applied Mathematics, Sefako Makgatho Health Sciences University, Ga-Rankuwa, Pretoria, Medunsa 0204, South Africa; aremukazeemolalekan@gmail.com or aremu.kazeem@udusok.edu.ng
[4] Department of Mathematics, Usmanu Danfodiyo University, Sokoto P.M.B. 2346, Nigeria
[5] Mathematics Department, College of Science, Taif University, P.O. Box 11099, Taif 21944, Saudi Arabia; aa.althobaiti@tu.edu.sa
[6] Department of Mathematics, COMSATS University, Islamabad Park Road, Islamabad 45550, Pakistan; abdullah_shah@comsats.edu.pk
* Correspondence: jsabiu@nwu.edu.ng

Abstract: In this paper, we derived a modified conjugate gradient (CG) parameter by adopting the Birgin and Martínez strategy using the descent three-term CG direction and the Newton direction. The proposed CG parameter is applied and suggests a robust algorithm for solving constrained monotone equations with an application to image restoration problems. The global convergence of this algorithm is established using some proper assumptions. Lastly, the numerical comparison with some existing algorithms shows that the proposed algorithm is a robust approach for solving large-scale systems of monotone equations. Additionally, the proposed CG parameter can be used to solve the symmetric system of nonlinear equations as well as other relevant classes of nonlinear equations.

Keywords: constrained monotone system; scaled three-term CG method; projection technique; descent condition

MSC: 90C30; 90C26

Citation: Sabi'u, J.; Aremu, K.O.; Althobait, A.; Shah, A. Scaled Three-Term Conjugate Gradient Methods for Solving Monotone Equations with Application. *Symmetry* **2022**, *14*, 936. https://doi.org/10.3390/sym14050936

Academic Editor: Alina Alb Lupas

Received: 19 March 2022
Accepted: 14 April 2022
Published: 5 May 2022

1. Introduction

Consider the system

$$F(x) = 0, \quad x \in \Omega, \tag{1}$$

such that the function $F : \Omega \longleftrightarrow \mathbf{R^n}$ is monotone and continuous. The monotonicity of F implies that

$$(F(x) - F(z))^T(x - z) \geq 0, \qquad x, z \in \Omega, \tag{2}$$

where Ω is a closed convex subset of $\mathbf{R^n}$. The system (1) arises in the areas of Bregman distances [1], financial forecasting problems [2], compressive sensing [3], and the monotone variational inequalities [4]. Additionally, whenever the Jacobian of F is symmetric, the system (1) is referred to as a symmetric system of nonlinear equations [5,6]. The symmetric system of nonlinear equations typically originates from the gradient mapping of unconstrained optimization problems and other relevant applications; more information on symmetric systems of equations can be found in [7,8], and the references therein. There are many methods or algorithms for solving the system of nonlinear equations in the literature. The most prominent ones included the Newton method [9] and the quasi-Newton methods [10,11]. These methods require the computation and the storage of the Jacobian matrix or its approximate at every iteration, as such, they are not efficient when dealing with large-scale problems and nonsmooth systems. These lead to the advance of the matrix-free

and derivative-free methods for solving the system of nonlinear equations, see [12–19] and the references therein.

Moreover, among the most robust matrix-free methods for solving the system (1) is the conjugate gradient (CG) method, which has the following iterative procedure:

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

where x_k is the initial guess, α_k is to be obtained using any line search procedures, and d_k is the CG search direction defined by

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

β_k is the CG parameter that differentiates the CG methods. Some recent CG methods for solving monotone equations with convex constraints are as follows: the efficient three-term conjugate gradient method [20], the iterative derivative-free method [21], the spectral gradient projection method [22], the hybrid gradient method [23], the Rivaie, Mustafa, Ismail, and Leong (RMIL) CG-type method [24], the modified double directions approach [25], the inertial two-hybrid spectral method [26], and the spectral projections CG method [27]. Each of these methods used the concept of the projection technique and addressed the large-scale system of monotone equations efficiently with some applications. Some recent papers on methods for solving convex constrained monotone systems can also be found in [28–31] and the references therein. Inspired by the success of these algorithms and the descent three-term conjugate gradient direction [32], we will propose a modified CG parameter by adopting the Birgin and Martínez [33] strategy, we will further use the proposed CG parameter and suggest a robust spectral three-term CG algorithm for solving the large-scale monotone equations with convex constraints with an application in image recovery.

The remaining paper is as follows: the next section is the derivation of the modified CG parameter and the proposed spectral three-term CG algorithm. Section 3 contains the global convergence analysis of the proposed algorithm. Section 4 provides the experimental comparison of the proposed algorithm with applications to the system of monotone equations with convex constraints and the image restoration problems. We finally give some concluding remarks with some future directions.

2. The Scaled Three-Term CG Method

The Birgin and Martínez [33] scaled two-term conjugate gradient (CG) method generates sequence x_k of approximate solutions of F, in which

$$x_{k+1} = x_k + \alpha_k d_k, \tag{5}$$

$$d_{k+1} = -\gamma_{k+1} F_{k+1} + \beta_k s_k, \tag{6}$$

where α_k satisfies some line search techniques, $F_k = F(x_k)$, β_k is a CG parameter, $s_k = x_{k+1} - x_k$ and γ_{k+1} is a scalar parameter or a matrix to be determined. The iterative process is initialized with an initial point x_0 and $d_0 = F_0$. Now, we proposed a scaled three-term CG direction for solving monotone equations based on the sufficient three-term descent direction [32], in which

$$d_{k+1} = -\gamma_{k+1} F_{k+1} + \beta_k s_k - \beta_k \frac{F_{k+1}^T s_k}{F_{k+1}^T p_k} p_k, \tag{7}$$

where p_k is any vector in $\mathbf{R^n}$. Observe that if $\gamma_{k+1} = 1$, then we get the default sufficient three-term CG direction based on the choice of the scalar β_k. On the other hand, if $\beta_k = 0$, we get another class of algorithms depending on γ_{k+1}(usually a scalar or a positive definite matrix). If γ_{k+1} is a positive scalar and $\beta_k \neq 0$, then we have a scaled three-term conjugate gradient method.

Moreover, we consider the following procedure to determine the parameter β_k. The Newton method for solving system of nonlinear equations generates the direction $d_{k+1} = -J_{k+1}^{-1}F_{k+1}$ at every iteration for $k \geq 1$. Thus, from the equality relation

$$-J_{k+1}^{-1}F_{k+1} = -\gamma_{k+1}F_{k+1} + \beta_k s_k - \beta_k \frac{F_{k+1}^T s_k}{F_{k+1}^T p_k} p_k, \tag{8}$$

we obtain

$$\beta_k = \frac{\left(s_k^T J_k \gamma_{k+1} F_{k+1} - s_k^T F_{k+1}\right) F_{k+1}^T p_k}{s_k^T J_k s_k F_{k+1}^T p_k - s_k^T J_k p_k F_{k+1}^T s_k}. \tag{9}$$

Using the Tailor's series expansion and adopting the Birgin and Martínez [33] strategy, we arrived at

$$\beta_k = \frac{\left(\gamma_{k+1} y_k - s_k\right)^T F_{k+1}}{y_k^T s_k F_{k+1}^T p_k - y_k^T p_k F_{k+1}^T s_k} F_{k+1}^T p_k \tag{10}$$

However, the denominator of (10) may be undefined if $y_k^T s_k F_{k+1}^T p_k$ is equal to $y_k^T p_k F_{k+1}^T s_k$. Hence, we approximated it with a positive constant $y_k^T s_k$ to have

$$\beta_k = \frac{\left(\gamma_{k+1} y_k - s_k\right)^T F_{k+1}}{y_k^T s_k} F_{k+1}^T p_k, \tag{11}$$

where $y_k = F_{k+1} - F_k + \sigma s_k$, $\sigma \in (0,1)$ and $s_k = x_{k+1} - x_k$.

Furthermore, inspired by the Birgin and Martínez algorithm [33], we consider the spectral gradient parameter γ_{k+1} as

$$\gamma_{k+1} = \frac{s_k^T s_k}{y_k^T s_k}. \tag{12}$$

The parameter γ_{k+1} is the inverse of the Rayleigh quotient and is always well-defined and positive since it guarantees that $y_k^T s_k > 0$ whenever F is monotone.

An operator $T_\Omega : \mathbf{R}^n \to \Omega$ is called a projection operator defined by

$$T_\Omega[x] = \arg\min\{\|x - z\| | z \in \Omega\}, \quad \forall x \in \mathbf{R}^n. \tag{13}$$

This operator satisfies the non-expansive property

$$\|T_\Omega[x] - T_\Omega[z]\| \leq \|x - z\|, \quad \forall x, z \in \mathbf{R}^n. \tag{14}$$

It is not difficult to see that $F_k^T d_k \leq -\|F_k\|^2$ for all $k \geq 0$ using simple algebra. The Lipschitz continuous assumption will be used in the following section to establish the global convergence of Algorithm 1.

Algorithm 1 Scaled Three-term CG method (STCG)

Step 0 Initialize: $\epsilon \geq 0$, $\zeta > 0$, $\lambda > 0$, $\tau > 0$, $\sigma \in (0,1)$, and $x_0 \in \mathbf{R^n}$. Set $k = 0$ and $d_0 = -F_0$.
Step 1 If $\|F_k\| \leq \epsilon$, stop, else go to Step 2.
Step 2 Compute the search direction d_k using (7) and (11).
Step 3 Set $m_k = x_k + \alpha_k d_k$ and determine the step-length $\alpha_k = \max\{\zeta\lambda^i : i = 0, 1, 2, \cdots\}$ satisfying

$$-F(m_k)^T d_k \geq \tau\alpha_k \|F(m_k)\| \|d_k\|^2. \tag{15}$$

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

$$x_{k+1} = P_\Omega[x_k - q_k F(m_k)], \tag{16}$$

where

$$q_k = \frac{F(m_k)^T(x_k - m_k)}{\|F(m_k)\|^2}. \tag{17}$$

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

3. Analysis of the Global Convergence

This section provides a global convergence analysis of the STCG algorithm under the assumption that the function F is Lipschitz continuous, i.e., for $c > 0$,

$$\|F(x) - F(z)\| \leq c\|x - z\|, \quad \forall x, z \in \mathbf{R}^n. \tag{18}$$

Lemma 1. *The line search procedure* (15) *is well-defined.*

Proof. Suppose that there is a constant $k_0 \geq 0$, such that (15) is not true for any non-negative integer i, then

$$-F(x_{k_0} + \zeta\lambda^i d_{k_0})^T d_{k_0} < \tau\lambda\zeta^i \|F(x_{k_0} + \zeta\lambda^i d_{k_0})\| \|d_{k_0}\|^2. \tag{19}$$

Allowing $i \to \infty$ in (19), we have

$$-F(x_{k_0})^T d_{k_0} \leq 0. \tag{20}$$

Additionally, from sufficient descent property of the STCG algorithm, we have

$$-F(x_{k_0})^T d_{k_0} \geq \|F(x_{k_0})\|^2 > 0. \tag{21}$$

Clearly (20) and (21) yield a contradiction. Therefore, the line search procedure (15) is well-defined. □

Lemma 2 ([34]). *If F is monotone and Lipschitz continuous, then*

$$\lim_{k\to\infty} \|x_k - m_k\| = \lim_{k\to\infty} \|\alpha_k d_k\| = 0, \tag{22}$$

$$\lim_{k\to\infty} \|x_{k+1} - x_k\| = 0, \tag{23}$$

$$\|x_k - \bar{x}\| \leq \|x_0 - \bar{x}\|, \quad \forall k \geq 0, \tag{24}$$

where $\bar{x}$ is any arbitrary solution of F.

The proof is skipped, because it is similar to the one given in [34].

Lemma 3. *The STCG direction generated using* (7) *and* (11) *is bounded.*

Proof. Assume that $\bar{x}$ is a solution of F, then

$$\|F(x_k)\| = \|F(x_k) - F(\bar{x})\| \leq c\|x_k - \bar{x}\| \leq c\|x_0 - \hat{x}\| = \kappa. \tag{25}$$

Again, from the monotone and Lipschitz assumptions on F

$$y_k^T s_k \geq \sigma\|s_k\|^2, \qquad \|y_k\| \leq (c+\sigma)\|s_k\|, \tag{26}$$

therefore,

$$\gamma_{k+1} \leq \frac{1}{\sigma}. \tag{27}$$

Choosing $p_k = F_{k+1}$, then from (11), (26), and (27), we get

$$\begin{aligned} |\beta_k| &\leq \frac{|\gamma_{k+1}|\|y_k\| + \|s_k\|}{\sigma\|s_k\|^2}\|F_{k+1}\|^3 \\ &\leq \frac{\frac{1}{\sigma}(c+\sigma)+1}{\sigma\|s_k\|}\|F_{k+1}\|^3. \end{aligned} \tag{28}$$

Hence, from the above and (7), we get

$$\begin{aligned} d_{k+1} &\leq \frac{1}{\sigma}\|F_{k+1}\| + 2\frac{\frac{1}{\sigma}(c+\sigma)+1}{\sigma\|s_k\|}\|F_{k+1}\|^3\|s_k\| \\ &\leq \frac{1}{\sigma}\kappa + 2\frac{\frac{1}{\sigma}(c+\sigma)+1}{\sigma}\kappa^3 = p. \end{aligned} \tag{29}$$

This shows the boundedness of the proposed direction. □

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

$$\liminf_{k\to\infty}\|F_k\| = 0. \tag{30}$$

Proof. Suppose that there exists a constant $a > 0$ such that

$$\|F_k\| > a, \quad \forall k \geq 0. \tag{31}$$

From the sufficient descent property and the Cauchy Schwarz inequality

$$\|F(x_k)\|^2 \leq -F(x_k)^T d_k \leq \|F(x_k)\|\|d_k\|, \quad \forall\, k \geq 0. \tag{32}$$

Thus,

$$\|d_k\| > 0. \tag{33}$$

On the other hand, $\|d_k\| \leq p \quad \forall k \geq 0$. Then it is obvious that for $\alpha_k \neq \zeta$, $\alpha_k\lambda^{-1}$ does not satisfy (15), i.e.,

$$-F(x_k + \lambda^{-1}\alpha_k d_k)^T d_k \geq \tau\lambda^{-1}\alpha_k\left\|F(x_k + \lambda^{-1}\alpha_k d_k)\right\|\|d_k\|^2, \tag{34}$$

also, by the sufficient descent property and letting $\bar{m}_k = x_k + \lambda^{-1}\alpha_k d_k$, we have

$$\begin{aligned} \|F_k\|^2 &\leq -F_k^T d_k = (F(z') - F_k)^T d_k - F(z')^T d_k \\ &\leq c\lambda^{-1}\alpha_k\|d_k\|^2 + \tau\lambda^{-1}\alpha_k\|F(\bar{m}_k)\|\|d_k\|^2, \end{aligned} \tag{35}$$

where the last inequality follows from (34) and the Cauchy Schwarz inequality. Hence from (35), we get

$$\alpha_k\|d_k\| \geq \frac{\lambda\|F_k\|^2}{(c + \tau\|F(\bar{m}_k)\|)\|d_k\|^2}\|d_k\|. \tag{36}$$

Using the boundedness of $\{x_k\}$ and $\{\alpha_k d_k\}$, we get that the sequence$\{x_k + \lambda^{-1}\alpha_k d_k\}$ is also bounded. Hence by the continuity of F, there is a constant ω, such that $\|F(\bar{m}_k)\| \leq \omega$. Thus

$$\alpha_k \|d_k\| \geq \frac{\lambda a^2}{(c+\omega)p}. \tag{37}$$

which clearly contradicts (22). Hence (30) is true. □

4. Numerical Experiment and Applications

This section presents a numerical experiment of the STCG algorithm on a system of monotone nonlinear equations with convex constraints, as well as its application to image restoration problems.

4.1. Application to the Monotone Nonlinear Equations

This subsection provides a numerical comparison of the STCG algorithm with some relevant algorithms for solving monotone nonlinear equations with convex constraints. We compared the proposed algorithm with the projection method for convex constrained monotone nonlinear equations with applications (PCG) [34], the derivative-free spectral projection algorithm (DFSP) [27], and the modified spectral gradient projection (MSGP) algorithm [22]. Moreover, for the proposed algorithm we set the following values for the initial parameters: $\sigma = 0.1$, $\lambda = 0.9$, $\zeta = 1$, $\tau = 0.0001$, and $\epsilon = 10^{-8}$. The remaining algorithms are implemented with the same parameters as implemented in the respective papers except for the stopping criteria. The following test problems are considered:

Problem 1. *This problem is from reference [35], given by*
$F_1(x) = e^{x_1} - 1$,
$F_i(x) = e^{x_i} + x_{i-1} - 1, \quad i = 2, 3, \cdots, n-1$, *and* $\Omega = \mathbf{R}^n_+$.

Problem 2. *This problem is from reference [35], given by*
$F_i(x) = \log(|x_i| + 1) - \frac{x_i}{n}, \quad i = 2, 3, \cdots, n$,
and $\Omega = \{x \in \mathbf{R}^n_+ : \sum_{i=1}^n x_i \leq n, \quad x_i \geq -1, \quad i = 1, 2, \cdots, n\}$.

Problem 3. *This problem is from reference [36], given by*
$F_1(x) = \cos(x_1) - 9 + 3x_1 + 8\exp(x_2)$
$F_i(x) = \cos(x_i) - 9 + 3x_i + 8\exp(x_{i-2})$, *for* $i = 2, 3, \ldots, n$, *and* $\Omega = \mathbf{R}^n_+$.

Problem 4. *This problem is from reference [35], given by*
$F_i(x) = \min(\min(|x_i|, x_i^2), \max(|x_i|, x_i^3)), \quad i = 1, 2 \ldots, n$, *and* $\Omega = \mathbf{R}^n_+$.

Problem 5. *This problem is from reference [35], given by*
$F_i(x) = e^{x_i} - 1, \quad i = 1, 2, \ldots, n$, *and* $\Omega = \mathbf{R}^n_+$.

From Tables 1–5, ITER stands for the number of iteration, TIME represents the computational time, and FVAL and NORM depict the number of function evaluations and the norm of the function value at the termination point respectively. We considered the initial points: $x_1 = (1, 1, \ldots, 1)$, $x_2 = (1, \frac{2}{2}, \frac{2}{3}, \ldots, \frac{2}{n})$, $x_3 = (0.01, 0.01, \ldots, 0.1)$, $x_4 = (\frac{1}{n}, \frac{2}{n}, \ldots, 1)$, $x_5 = (1 - \frac{1}{n}, 1 - \frac{2}{n}, \ldots, 0)$, $x_6 = (-1, -1, \ldots, -1)$, $x_7 = (n - \frac{1}{n}, n - \frac{2}{n}, \ldots, n - 1)$ and $x_8 = (\frac{1}{2}, 1, \frac{2}{3}, \ldots, \frac{2}{n})$.

Table 1. Numerical comparison of the STCG algorithm versus the PCG [34], DFSP [27], and MSGP [22] algorithms.

Problem 1		STCG				PCG				DFSP				MSGP			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
500	x_1	61	780	0.284719	8.26×10^{-9}	17	139	0.10215	0	6	69	0.031098	0	10	106	0.390938	0
	x_2	51	654	0.092112	7.86×10^{-9}	12	106	0.0335	0	7	83	0.014426	0	6	71	0.028397	0
	x_3	2	27	0.011366	0	39	130	0.03516	6.27×10^{-9}	4	41	0.006115	0	3	22	0.005823	0
	x_4	3	38	0.013019	0	81	994	0.126118	7.87×10^{-9}	5	57	0.011089	0	7	64	0.008197	0
	x_5	3	38	0.012648	0	13	109	0.021424	0	5	56	0.011844	0	7	64	0.007359	0
	x_6	2	19	0.00851	0	2	17	0.006121	0	2	17	0.005261	0	2	18	0.005216	0
	x_7	3	38	0.007233	0	13	109	0.021296	0	5	56	0.010039	0	7	64	0.011164	0
	x_8	6	58	0.017458	0	14	146	0.013858	0	6	72	0.013269	0	6	61	0.010591	0
1000	x_1	9	97	0.060604	0	FAIL	FAIL	FAIL	FAIL	9	78	0.019464	0	10	98	0.022113	0
	x_2	51	654	0.327119	7.86×10^{-9}	12	106	0.026258	0	7	83	0.019734	0	6	71	0.015524	0
	x_3	2	27	0.016096	0	13	102	0.028791	0	6	66	0.015238	0	4	35	0.009522	0
	x_4	3	38	0.016324	0	11	84	0.014168	0	5	56	0.014616	0	8	78	0.031971	0
	x_5	3	38	0.017409	0	11	84	0.024139	0	5	56	0.018121	0	8	78	0.023144	0
	x_6	2	19	0.01026	0	2	17	0.008221	0	2	17	0.008488	0	2	18	0.007901	0
	x_7	3	38	0.017569	0	11	84	0.024257	0	5	56	0.024643	0	8	78	0.022647	0
	x_8	6	57	0.022786	FAIL	FAIL	FAIL	FAIL	FAIL	6	72	0.020657	0	6	61	0.017588	0
10,000	x_1	45	555	1.140353	9.2×10^{-9}	14	105	0.082426	0	9	68	0.113286	0	11	104	0.223313	0
	x_2	51	654	1.385021	7.86×10^{-9}	12	106	0.136972	0	7	83	0.154083	0	6	71	0.09223	0
	x_3	2	27	0.036788	0	15	118	0.152166	0	5	49	0.07832	0	4	35	0.050341	0
	x_4	4	40	0.052326	0	14	110	0.084851	0	5	56	0.100839	0	7	63	0.090927	0
	x_5	4	29	0.075848	0	14	109	0.14573	0	5	56	0.084585	0	7	63	0.089563	0
	x_6	2	19	0.044899	0	2	17	0.026459	0	3	30	0.051177	0	2	18	0.02956	0
	x_7	4	29	0.069435	0	14	109	0.084996	0	5	56	0.091435	0	7	63	0.086863	0
	x_8	6	57	0.11967	0	FAIL	FAIL	FAIL	FAIL	6	72	0.114648	0	6	61	0.088411	0
50,000	x_1	6	45	0.261177	0	14	102	0.347122	0	7	71	0.493357	0	10	90	0.557246	0
	x_2	51	654	4.286989	7.86×10^{-9}	12	106	0.577771	0	7	83	0.596953	0	6	71	0.437155	0
	x_3	2	27	0.275629	0	22	209	0.847256	0	8	73	0.496097	0	5	40	0.251553	0
	x_4	2	25	0.14172	0	14	110	0.635446	0	5	56	0.393569	0	7	63	0.396589	0
	x_5	2	25	0.227892	0	14	110	0.621259	0	5	56	0.382475	0	7	63	0.396818	0
	x_6	2	19	0.174011	0	2	17	0.106832	0	3	30	0.203618	0	2	18	0.117185	0
	x_7	2	25	0.14487	0	14	110	0.491524	0	5	56	0.394082	0	7	63	0.392917	0
	x_8	6	57	0.408297	0	FAIL	FAIL	FAIL	FAIL	6	72	0.47525	0	6	61	0.37816	0

Table 1. *Cont.*

Problem 1		STCG				PCG				DFSP				MSGP			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
100,000	x_1	70	881	13.66074	8.42×10^{-9}	15	123	1.227476	0	11	124	1.686574	0	10	98	1.212834	0
	x_2	51	654	9.740805	7.86×10^{-9}	12	106	1.119876	0	7	83	1.210576	0	6	71	0.945403	0
	x_3	2	27	0.45712	0	52	591	4.82865	0	5	36	0.62634	0	5	37	0.460798	0
	x_4	2	25	0.489964	0	14	110	0.801824	0	5	56	0.869335	0	7	63	0.826982	0
	x_5	2	25	0.317064	0	14	110	1.075937	0	5	56	0.786943	0	7	63	0.868822	0
	x_6	2	19	0.406541	0	2	17	0.203338	0	3	30	0.504184	0	2	18	0.245538	0
	x_7	2	25	0.49201	0	14	110	1.203385	0	5	56	0.948483	0	7	63	0.829146	0
	x_8	6	57	1.095133	0	FAIL	FAIL	FAIL	FAIL	6	72	1.041105	0	6	61	0.855313	0

Table 2. Numerical comparison of the STCG algorithm versus the PCG [34], DFSP [27], and MSGP [22] algorithms.

Problem 2		STCG				PCG				DFSP				MSGP			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
500	x_1	2	5	0.005374	0	11	23	0.010948	5.14×10^{-9}	16	33	0.013891	3.08×10^{-9}	3	7	2.032566	0
	x_2	4	20	0.020759	0	4	14	0.007217	0	2	5	0.00451	0	2	5	0.004377	0
	x_3	2	5	0.005889	0	7	15	0.008634	2.65×10^{-9}	11	23	0.010986	9.07×10^{-9}	3	7	0.0044	0
	x_4	30	172	0.053898	0	5	11	0.004814	0	5	12	0.007259	0	3	7	0.00478	0
	x_5	15	149	0.043735	0	5	11	0.007061	0	5	12	0.006897	0	3	7	0.004881	0
	x_6	1	3	0.004408	0	1	3	0.002925	0	1	3	0.003868	0	1	3	0.003211	0
	x_7	15	149	0.046339	0	5	11	0.007912	0	5	12	0.006226	0	3	7	0.004804	0
	x_8	4	20	0.005596	0	4	9	0.00433	0	4	9	0.005803	0	4	20	0.008088	0
1000	x_1	2	5	0.006361	0	11	23	0.009104	6.68×10^{-9}	16	33	0.017005	5.66×10^{-9}	3	18	0.008546	0
	x_2	4	20	0.012093	0	4	14	0.010733	0	2	5	0.005167	0	2	5	0.004811	0
	x_3	2	5	0.006596	0	7	15	0.006435	3.06×10^{-9}	12	25	0.014617	3.3×10^{-9}	3	18	0.008887	0
	x_4	13	115	0.054277	0	5	11	0.009428	0	5	12	0.008852	0	3	7	0.006227	0
	x_5	6	13	0.043383	0	5	11	0.005846	0	5	12	0.008944	0	3	7	0.005588	0
	x_6	1	3	0.002764	0	1	3	0.004738	0	1	3	0.004072	0	1	3	0.003821	0
	x_7	6	13	0.026444	0	5	11	0.008963	0	5	12	0.009112	0	3	7	0.005492	0
	x_8	4	20	0.006369	0	4	9	0.008496	0	4	9	0.007132	0	4	20	0.009503	0

Table 2. *Cont.*

Problem 2		STCG				PCG				DFSP				MSGP			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
10,000	x_1	2	5	0.019078	0	12	25	0.067221	2.1×10^{-9}	18	37	0.080939	4.04×10^{-9}	3	7	0.018681	0
	x_2	4	20	0.055031	0	4	15	0.03359	0	2	5	0.014111	0	2	5	0.012974	0
	x_3	2	5	0.018543	0	7	15	0.024264	7.84×10^{-9}	12	25	0.057137	8.87×10^{-9}	3	18	0.037296	0
	x_4	6	13	0.045315	0	5	11	0.035584	0	5	12	0.032885	0	3	7	0.021448	0
	x_5	6	13	0.043383	0	5	11	0.020979	0	5	12	0.031453	0	3	7	0.016891	0
	x_6	1	3	0.010935	0	1	3	0.010224	0	1	3	0.009962	0	1	3	0.007533	0
	x_7	6	13	0.026444	0	5	11	0.032401	0	5	12	0.02918	0	3	7	0.01715	0
	x_8	4	20	0.05496	0	4	9	0.027178	0	4	9	0.025853	0	4	20	0.032959	0
50,000	x_1	2	5	0.084485	0	12	25	0.245072	4.67×10^{-9}	19	39	0.354658	5.43×10^{-9}	3	7	0.058291	0
	x_2	4	20	0.201987	0	4	15	0.086715	0	2	5	0.043553	0	2	5	0.042419	0
	x_3	2	5	0.041542	0	8	17	0.155248	1.85×10^{-9}	13	27	0.250225	5.86×10^{-9}	3	7	0.058758	0
	x_4	8	66	0.702038	0	5	11	0.108177	0	5	12	0.097718	0	3	7	0.048521	0
	x_5	4	31	0.328703	0	5	11	0.068881	0	5	12	0.113344	0	3	7	0.05755	0
	x_6	1	3	0.032938	0	1	3	0.028052	0	1	3	0.029898	0	1	3	0.020789	0
	x_7	4	31	0.332903	0	5	11	0.110806	0	5	12	0.106826	0	3	7	0.06394	0
	x_8	4	20	0.128546	0	4	9	0.054206	0	4	9	0.083519	0	4	20	0.12471	0
100,000	x_1	2	5	0.081947	0	12	25	0.471086	6.59×10^{-9}	20	41	0.657614	3.14×10^{-9}	3	7	0.111139	0
	x_2	4	20	0.294591	0	4	15	0.141274	0	2	5	0.096178	0	2	5	0.066017	0
	x_3	2	5	0.075869	0	8	17	0.319337	2.6×10^{-9}	13	27	0.43282	8.27×10^{-9}	3	18	0.239369	0
	x_4	4	34	0.674149	0	5	11	0.217111	0	5	12	0.20886	0	3	7	0.086976	0
	x_5	4	34	0.756062	0	5	11	0.224032	0	5	12	0.220614	0	3	7	0.10008	0
	x_6	1	3	0.06054	0	1	3	0.056049	0	1	3	0.054279	0	1	3	0.050571	0
	x_7	4	34	0.487762	0	5	11	0.1307	0	5	12	0.234348	0	3	7	0.09841	0
	x_8	4	20	0.266742	0	4	9	0.175585	0	4	9	0.155336	0	4	20	0.284555	0

Table 3. Numerical comparison of the STCG algorithm versus the PCG [34], DFSP [27], and MSGP [22] algorithms.

Problem 3		STCG				PCG				DFSP				MSGP			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
500	x_1	1	14	0.008851	0	11	68	0.019816	3.49×10^{-9}	2	24	0.003515	0	1	14	0.024658	0
	x_2	1	14	0.007385	0	12	136	0.029378	0	6	76	0.013465	0	9	118	0.018953	0
	x_3	1	14	0.008368	0	10	62	0.011243	1.85×10^{-9}	2	23	0.008741	0	3	18	0.004018	0
	x_4	1	14	0.045722	0	48	590	0.10955	0	5	63	0.018218	0	4	53	0.008094	0
	x_5	1	14	0.004585	0	10	98	0.022402	0	5	63	0.019463	0	3	40	0.010035	0
	x_6	9	36	0.017405	3.07×10^{-10}	1	7	0.003626	0	1	9	0.005846	0	1	12	0.004765	0
	x_7	1	14	0.005084	0	10	98	0.024546	0	5	63	0.024745	0	3	40	0.008947	0
	x_8	1	14	0.007788	0	5	11	0.007993	0	2	24	0.008523	0	3	40	0.009953	0
1000	x_1	1	14	0.012929	0	11	68	0.024155	4.93×10^{-9}	2	24	0.013413	0	1	14	0.006759	0
	x_2	1	14	0.008868	0	12	136	0.041538	0	6	76	0.033582	0	9	118	0.03269	0
	x_3	1	14	0.006702	0	10	62	0.021417	2.62×10^{-9}	2	23	0.01052	0	3	18	0.007584	0
	x_4	1	14	0.011304	0	14	148	0.04369	0	5	63	0.022801	0	4	53	0.022023	0
	x_5	1	14	0.010613	0	10	97	0.019306	0	5	63	0.028107	0	3	40	0.017749	0
	x_6	9	36	0.013732	4.33×10^{-10}	1	7	0.006027	0	1	9	0.007979	0	1	12	0.008322	0
	x_7	1	14	0.011291	0	10	97	0.032336	0	5	63	0.028723	0	3	40	0.01787	0
	x_8	1	14	0.009815	0	19	236	0.063194	0	2	24	0.011389	0	3	40	0.016378	0
10,000	x_1	1	14	0.046693	0	12	74	0.143705	1.63×10^{-9}	2	24	0.060277	0	1	14	0.029536	0
	x_2	1	14	0.02509	0	12	136	0.129926	0	6	76	0.143179	0	9	118	0.230534	0
	x_3	1	14	0.053757	0	10	62	0.117018	8.28×10^{-9}	2	23	0.052114	0	3	29	0.070871	0
	x_4	1	14	0.046307	0	13	135	0.226083	0	5	63	0.144242	0	4	53	0.102726	0
	x_5	1	14	0.026234	0	10	96	0.095035	0	5	63	0.127534	0	4	53	0.103745	0
	x_6	9	36	0.116424	1.37×10^{-9}	1	7	0.017484	0	1	9	0.023845	0	1	12	0.029127	0
	x_7	1	14	0.045158	0	10	96	0.100895	0	5	63	0.14972	0	4	53	0.112623	0
	x_8	1	14	0.024566	0	18	223	0.199282	0	2	24	0.054492	0	3	40	0.087067	0
50,000	x_1	1	14	0.18426	0	12	74	0.604309	3.65×10^{-9}	2	24	0.231208	0	1	14	0.133505	0
	x_2	1	14	0.157724	0	12	136	0.933096	0	6	76	0.673996	0	9	118	1.090432	0
	x_3	1	14	0.110849	0	11	68	0.561177	1.94×10^{-9}	2	23	0.203924	0	3	18	0.159291	0
	x_4	1	14	0.201193	0	13	135	0.929651	0	5	63	0.626326	0	4	53	0.47979	0
	x_5	1	14	0.175146	0	10	96	0.400642	0	5	63	0.556109	0	4	53	0.457753	0
	x_6	9	36	0.482154	3.07×10^{-9}	1	7	0.06678	0	1	9	0.097355	0	1	12	0.110101	0
	x_7	1	14	0.195195	0	10	96	0.67851	0	5	63	0.587922	0	4	53	0.439318	0
	x_8	1	14	0.155814	0	362	725	8.665941	9.93×10^{-9}	2	24	0.244387	0	3	40	0.337612	0

Table 3. *Cont.*

Problem 3		STCG				PCG				DFSP				MSGP			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
100,000	x_1	1	14	0.357991	0	12	74	0.675105	5.17×10^{-9}	2	24	0.509043	0	1	14	0.300253	0
	x_2	1	14	0.321884	0	12	136	1.658807	0	6	76	1.545788	0	9	118	2.269149	0
	x_3	1	14	0.368256	0	11	68	0.998355	2.74×10^{-9}	2	23	0.415931	0	3	29	0.561295	0
	x_4	1	14	0.365994	0	12	122	1.529508	0	5	63	1.297313	0	4	53	1.032402	0
	x_5	1	14	0.376287	0	10	96	1.339508	0	5	63	1.36258	0	4	53	1.12956	0
	x_6	9	36	1.077033	4.34×10^{-9}	1	7	0.124462	0	1	9	0.183223	0	1	12	0.220294	0
	x_7	1	14	0.381831	0	10	96	1.237283	0	5	63	1.279186	0	4	53	1.068634	0
	x_8	1	14	0.193508	0	451	903	17.88253	9.97×10^{-9}	2	24	0.473854	0	3	40	0.869784	0

Table 4. Numerical comparison of the STCG algorithm versus the PCG [34], DFSP [27], and MSGP [22] algorithms.

Problem 4		STCG				PCG				DFSP				MSGP			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
500	x_1	1	3	0.004658	0	10	22	0.008221	1.8×10^{-9}	1	3	0.004394	0	1	3	1.812187	0
	x_2	45	520	0.185998	0	14	93	0.006089	0	25	74	0.050085	1.76×10^{-10}	34	91	0.054258	9.91×10^{-9}
	x_3	1	3	0.00478	0	9	20	0.00517	2×10^{-9}	15	32	0.008077	3.22×10^{-9}	2	5	0.00525	0
	x_4	9	96	0.035615	0	9	19	0.014835	0	5	46	0.004736	0	92	1164	0.512712	9.99×10^{-9}
	x_5	10	109	0.072025	0	9	19	0.017347	0	6	60	0.011102	0	94	1190	0.474454	9.86×10^{-9}
	x_6	1	3	0.004958	0	1	3	0.005073	0	1	3	0.002327	0	1	3	0.00421	0
	x_7	14	139	0.049063	0	9	19	0.010113	0	6	60	0.012052	0	94	1190	0.476848	9.86×10^{-9}
	x_8	30	325	0.180583	0	5	11	0.011729	0	7	39	0.023268	0	5	33	0.024018	0
1000	x_1	1	3	0.004922	0	10	22	0.009612	2.54×10^{-9}	1	3	0.004198	0	1	3	0.004174	0
	x_2	47	546	0.332351	0	14	93	0.016606	0	26	73	0.082564	3.54×10^{-10}	46	115	0.109098	9.61×10^{-9}
	x_3	1	3	0.003611	0	9	20	0.008701	2.83×10^{-9}	15	32	0.012827	4.56×10^{-9}	2	5	0.006527	0
	x_4	4	31	0.043305	0	9	19	0.019201	0	5	46	0.013441	0	128	1632	1.280052	9.96×10^{-9}
	x_5	4	31	0.038655	0	9	19	0.024678	0	6	59	0.015477	0	128	1632	0.94554	9.95×10^{-9}
	x_6	1	3	0.006836	0	1	3	0.006031	0	1	3	0.003819	0	1	3	0.004479	0
	x_7	4	31	0.038201	0	9	19	0.014653	0	6	59	0.015738	0	128	1632	1.081112	9.95×10^{-9}
	x_8	30	325	0.321325	0	5	11	0.016327	0	7	39	0.031242	0	5	33	0.028514	0

Table 4. *Cont.*

Problem 4		STCG					PCG			DFSP				MSGP			
10,000	x_1	1	3	0.014354	0	10	22	0.030634	8.04×10^{-9}	1	3	0.010948	0	1	3	0.009406	0
	x_2	45	520	1.594754	0	14	94	0.0689	0	6	57	0.066085	0	77	177	0.729764	1×10^{-8}
	x_3	1	3	0.015226	0	9	20	0.026621	8.96×10^{-9}	16	34	0.053622	4.33×10^{-9}	2	5	0.017532	0
	x_4	4	31	0.180938	0	9	19	0.073933	5.84×10^{-10}	6	59	0.071969	0	5	51	0.05935	0
	x_5	4	31	0.320807	0	9	19	0.05876	1.76×10^{-9}	6	59	0.067016	0	5	51	0.057587	0
	x_6	1	3	0.034752	0	1	3	0.015392	0	1	3	0.010347	0	1	3	0.011976	0
	x_7	4	31	0.24721	0	9	19	0.094012	1.76×10^{-9}	6	59	0.071367	0	5	51	0.063661	0
	x_8	30	325	2.706975	0	7	15	0.045114	0	5	50	0.057792	0	6	66	0.073877	0
50,000	x_1	1	3	0.048857	0	11	24	0.120884	1.93×10^{-9}	1	3	0.023601	0	1	3	0.033587	0
	x_2	47	546	11.85282	0	14	94	0.287916	0	6	57	0.260533	0	85	193	3.342553	9.94×10^{-9}
	x_3	1	3	0.03657	0	10	22	0.109335	2.15×10^{-9}	16	34	0.211381	9.69×10^{-9}	3	18	0.269339	0
	x_4	4	41	0.732836	4.41×10^{-10}	9	19	0.137568	3.02×10^{-9}	6	59	0.28187	0	5	51	0.246078	0
	x_5	4	41	1.441536	4.41×10^{-10}	9	19	0.148562	3×10^{-9}	6	59	0.32215	0	5	51	0.230736	0
	x_6	1	3	0.086636	0	1	3	0.052435	0	1	3	0.022742	0	1	3	0.025453	0
	x_7	4	41	1.297597	4.41×10^{-10}	9	19	0.343581	3×10^{-9}	6	59	0.295197	0	5	51	0.25135	0
	x_8	30	325	9.67262	0	362	725	11.39927	9.93×10^{-9}	5	50	0.230919	0	6	66	0.297773	0
100,000	x_1	1	3	0.057776	0	11	24	0.251555	2.73×10^{-9}	1	3	0.053633	0	1	3	0.028851	0
	x_2	45	520	25.59638	0	14	94	0.589869	0	6	57	0.540931	0	86	195	4.930206	9.84×10^{-9}
	x_3	1	3	0.126164	0	10	22	0.202703	3.04×10^{-9}	17	36	0.461083	4.12×10^{-9}	3	18	0.396615	0
	x_4	4	42	3.008887	1.56×10^{-11}	9	19	0.31304	4.25×10^{-9}	6	59	0.545789	0	5	51	0.491155	0
	x_5	4	42	2.617622	1.56×10^{-11}	9	19	0.564087	4.23×10^{-9}	6	59	0.616111	0	5	51	0.481329	0
	x_6	1	3	0.252864	0	1	3	0.051067	0	1	3	0.046437	0	1	3	0.047267	0
	x_7	4	42	2.030463	1.56×10^{-11}	9	19	0.713459	4.23×10^{-9}	6	59	0.569277	0	5	51	0.470185	0
	x_8	34	377	16.45906	0	451	903	27.56836	9.97×10^{-9}	5	50	0.496654	0	6	66	0.607154	0

Table 5. Numerical comparison of the STCG algorithm versus the PCG [34], DFSP [27], and MSGP [22] algorithms.

Problem 5		STCG				PCG				DFSP				MSGP			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
500	x_1	10	27	0.010574	6.64×10^{-9}	10	22	0.010898	1.8×10^{-9}	19	41	0.004176	3.72×10^{-9}	3	10	0.009295	0
	x_2	5	60	0.021266	0	14	93	0.017962	0	6	57	0.013308	0	5	49	0.011538	0
	x_3	1	14	0.004491	0	9	20	0.009033	2×10^{-9}	15	32	0.015205	3.22×10^{-9}	3	8	0.002287	0
	x_4	3	33	0.007824	0	14	101	0.019996	0	5	46	0.014318	0	5	51	0.007657	0
	x_5	3	33	0.00706	0	14	97	0.017572	0	6	60	0.016291	0	5	51	0.011089	0
	x_6	1	3	0.004294	0	1	3	0.004387	0	1	3	0.004597	0	1	3	0.003946	0
	x_7	3	33	0.009805	0	14	97	0.010927	0	6	60	0.018994	0	5	51	0.010937	0
	x_8	4	49	0.00847	0	12	81	0.014564	0	5	50	0.015699	0	6	66	0.012359	0
1000	x_1	10	27	0.010574	9.39×10^{-9}	10	22	0.012155	2.54×10^{-9}	19	41	0.021854	5.27×10^{-9}	3	21	0.009133	0
	x_2	5	60	0.013321	0	14	93	0.012273	0	6	57	0.021194	0	5	49	0.013766	0
	x_3	1	14	0.006824	0	9	20	0.011087	2.83×10^{-9}	15	32	0.018919	4.56×10^{-9}	3	19	0.007292	0
	x_4	3	33	0.020399	0	14	99	0.022258	0	5	46	0.010385	0	5	51	0.014627	0
	x_5	3	33	0.009816	0	14	97	0.013263	0	6	59	0.021793	0	5	51	0.016709	0
	x_6	1	3	0.003477	0	1	3	0.004652	0	1	3	0.004924	0	1	3	0.003584	0
	x_7	3	33	0.009404	0	14	97	0.022413	0	6	59	0.022199	0	5	51	0.01455	0
	x_8	4	49	0.012058	0	12	81	0.011575	0	5	50	0.019057	0	6	66	0.015567	0
10,000	x_1	11	29	0.051525	2.7×10^{-9}	10	22	0.042853	8.04×10^{-9}	20	43	0.097315	5.12×10^{-9}	3	10	0.017224	0
	x_2	5	60	0.07864	0	14	94	0.103893	0	6	57	0.056325	0	5	49	0.060214	0
	x_3	1	14	0.019719	0	9	20	0.025003	8.96×10^{-9}	16	34	0.080901	4.33×10^{-9}	3	8	0.0185	0
	x_4	4	35	0.040926	0	14	98	0.109376	0	6	59	0.101109	0	5	51	0.061669	0
	x_5	4	35	0.043046	0	14	98	0.062041	0	6	59	0.10234	0	5	51	0.064411	0
	x_6	1	3	0.006305	0	1	3	0.008124	0	1	3	0.009881	0	1	3	0.006958	0
	x_7	4	35	0.054718	0	14	98	0.063321	0	6	59	0.106607	0	5	51	0.063783	0
	x_8	4	49	0.055413	0	12	81	0.089283	0	5	50	0.084264	0	6	66	0.082033	0
50,000	x_1	11	29	0.172819	6.04×10^{-9}	11	24	0.10662	1.93×10^{-9}	21	45	0.406622	3.58×10^{-9}	3	10	0.060921	0
	x_2	5	60	0.306848	0	14	94	0.387555	0	6	57	0.255425	0	5	49	0.263969	0
	x_3	1	14	0.063016	0	10	22	0.147784	2.15×10^{-9}	16	34	0.28772	9.69×10^{-9}	3	8	0.045655	0
	x_4	2	20	0.095903	0	14	98	0.261356	0	6	59	0.412698	0	5	51	0.245782	0
	x_5	2	20	0.087273	0	14	98	0.264178	0	6	59	0.315475	0	5	51	0.252724	0
	x_6	1	3	0.018449	0	1	3	0.022432	0	1	3	0.027422	0	1	3	0.019779	0
	x_7	2	20	0.090757	0	14	98	0.260123	0	6	59	0.237437	0	5	51	0.241624	0
	x_8	4	49	0.285972	0	12	81	0.358104	0	5	50	0.323593	0	6	66	0.346647	0

Table 5. *Cont.*

Problem 5		STCG				PCG				DFSP				MSGP			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
100,000	x_1	11	29	0.332679	8.54×10^{-9}	11	24	0.206588	2.73×10^{-9}	21	45	0.525293	5.22×10^{-9}	3	21	0.215806	0
	x_2	5	60	0.602989	0	14	94	0.767629	0	6	57	0.723098	0	5	49	0.538745	0
	x_3	1	14	0.153681	0	10	22	0.195123	3.04×10^{-9}	17	36	0.644117	4.12×10^{-9}	3	19	0.195524	0
	x_4	2	20	0.200692	0	14	98	0.602653	0	6	59	0.785726	0	5	51	0.516432	0
	x_5	2	20	0.182723	0	14	98	0.801786	0	6	59	0.76921	0	5	51	0.476088	0
	x_6	1	3	0.036697	0	1	3	0.046373	0	1	3	0.05628	0	1	3	0.03321	0
	x_7	2	20	0.203324	0	14	98	0.499816	0	6	59	0.760231	0	5	51	0.547949	0
	x_8	4	49	0.474225	0	12	81	0.560271	0	5	50	0.63667	0	6	66	0.593046	0

Moreover, a numerical comparison of Problem 1 indicates that the PCG algorithm failed on one initial point despite a large number of iterations. The STCG algorithm recorded fewer iterations compared to the PCG, DFSP, and MSGP algorithms. In terms of the number of iterations, the STCG method won over 90% of the time for Problems 3 and 5. Whereas the for remaining problems, the STCG algorithm performed more efficiently with more than 50% less number of iterations. However, when comparing the number of function evaluations and computing time, the STCG algorithm is likewise highly efficient for Problems 1, 3, and 5. On average, the STCG algorithm is competing with the remaining algorithms in terms of function evaluation and computing time. In general, the STCG algorithm is the most efficient across all problems, followed by the MSGP algorithm and the other two algorithms.

Furthermore, we simplified the numerical comparison by visualizing the performance profiles of the number of iterations, computation time, and the number of function evaluations using the well-known Dolan and Moré technique [37]. According to Figures 1–3, the STCG method is best in terms of the number of iterations and is competing with both the DFSP and MSGP algorithms for the minimal number of function evaluations and computing time.

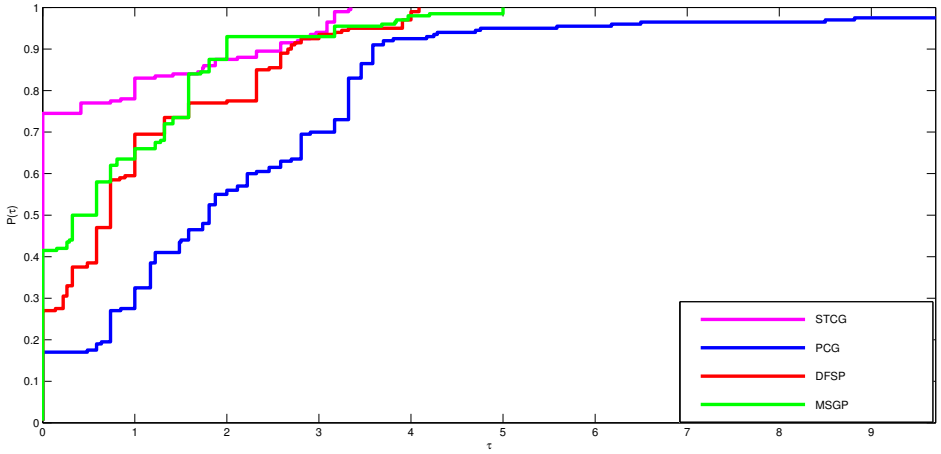

Figure 1. Performance profile of the the STCG algorithm versus the PCG [34], DFSP [27], and MSGP [22] algorithms for number of iterations.

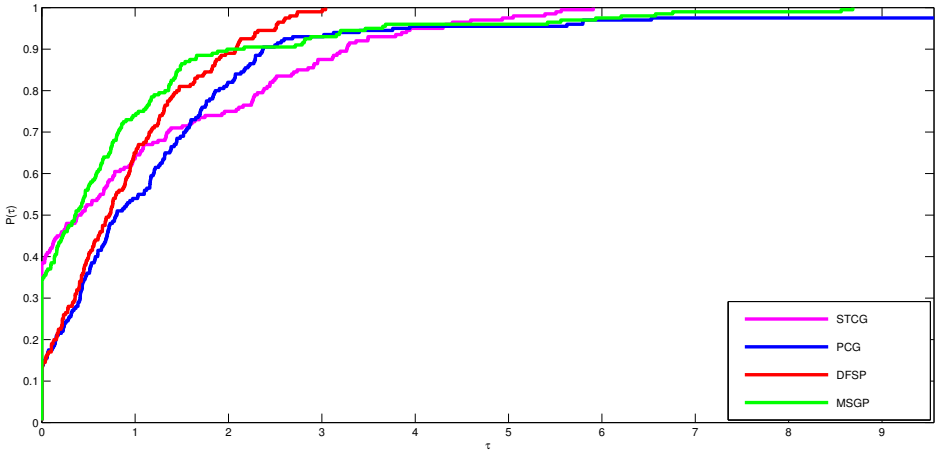

Figure 2. Performance profile of the STCG algorithm versus the PCG [34], DFSP [27], and MSGP [22] algorithms for CPU time.

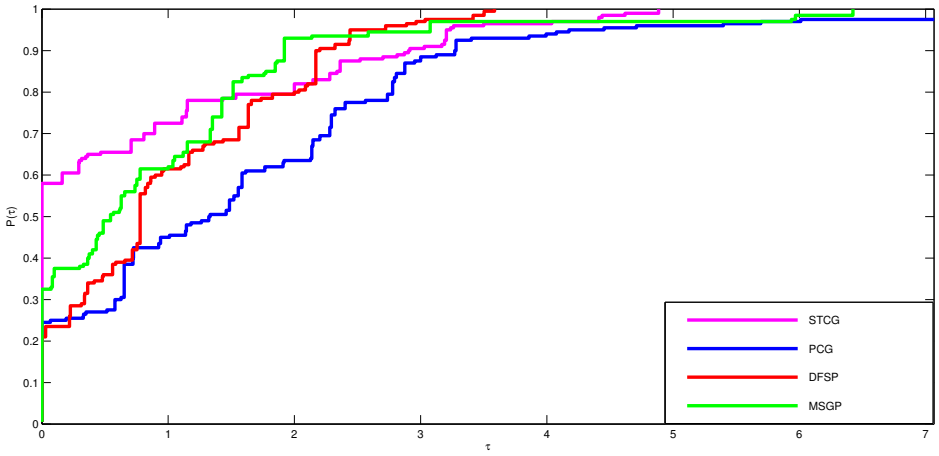

Figure 3. Performance profile of the STCG algorithm versus the PCG [34], DFSP [27], and MSGP [22] algorithms for number of function evaluations.

4.2. Application in Signal Recovery

The pursuit of the denoising problem that arises in compressive sensing is described by:

$$\min s\|x\|_1 + \frac{1}{2}\|Qx - r\|_2^2, \tag{38}$$

where $r \in \mathbf{R}^k$ is an observation, $x \in \mathbf{R}^n$, $Q \in \mathbf{R}^{k\times n}(k << n)$ is a linear operator, s is a nonnegative parameter, and $\|.\|_1$, $\|.\|_2$ represent the l_1 and l_2 norms respectively, see [3,38–40]. This problem can be reformulated into the basic bound-constrained quadratic problem and further into the following system of monotone nonlinear equations

$$F(z) = \min\{z, Av + b\} = 0. \tag{39}$$

For more detail about the transformation (39), Lipschitz continuity, and monotonicity of $F(z)$, see [41,42].

We have made an experiment with the image restoration problem. In the experiment, we considered four different algorithms, namely the STCG algorithm, the Perry-type derivative-free method for solving nonlinear system of equations (PSGM) [43], the conjugate gradient method for convex constrained monotone equations with applications in compressive sensing (CGD) [44], and the three-term Polak–Ribiére–Polyak derivative-free method (TPRP) [45]. We applied these algorithms to four different benchmark problems, namely the fox, Lena, horse, and the frog. All the experiments in this work are coded using Matlab R2014a and run on a personal computer HP core i5, 8th Gen. We used the mean of squared error (MSE) and signal-to-ratio (SNR) to measure the quality of the restored images

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

and,

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

According to Figures 4–7, the STCG algorithm restored all of the images with fewer iterations and less computation time than the PSGM, CGD, and TPRP methods. In addition, when compared to the CGD and TPRP algorithms, the PSGM approach is highly promising in terms of the number of iterations and computing time. Moreover, when compared to the other three algorithms, the STCG has the smallest MSE. Furthermore, the TPRP algorithm has the highest SNR values in all of the problems considered, followed by the CGD, PSGM, and STCG algorithms. These findings revealed that the STCG algorithm is a

great alternative for dealing with image restoration problems with fewer iterations, MSE, and computation time.

Figure 4. Original image, blurred image, restored image by STCG with time: 0.58 s, iterations: 7, MSE: 1.6745×10^2, and SNR: 20.86, restored image by PSGM with time: 1.05 s, iterations: 18, MSE: 7.3735×10, and SNR: 24.42, restored image by CGD with time: 33.36 s, iterations: 474, MSE: 7.4512×10, and SNR: 24.38, and restored image by TPRP with time: 250.69 s, iterations: 58, MSE: 7.7598×10, and SNR: 24.20.

Figure 5. Original image, blurred image, restored image by STCG with time: 0.59 s, iterations: 7, MSE: 1.6745×10^2, and SNR: 20.86, restored image by PSGM with time: 1.94 s, iterations: 26, MSE: 5.8083×10, and SNR: 18.38, restored image by CGD with time: 33.36 s, iterations: 491, MSE: 7.4512×10, and SNR: 23.23, and restored image by TPRP with time: 809.14 s, iterations: 101, MSE: 5.9071×10, and SNR: 23.16.

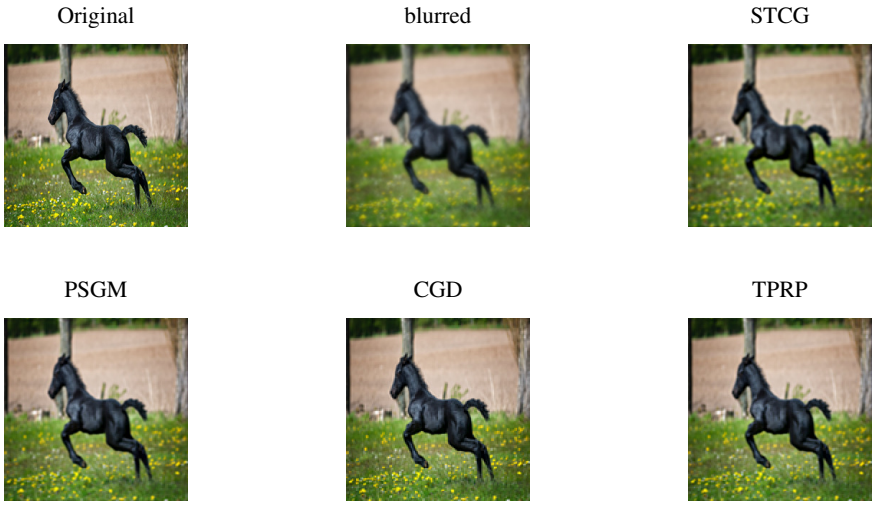

Figure 6. Original image, blurred image, restored image by STCG with time: 0.86 s, iterations: 7, MSE: 1.9173×10^2, and SNR: 17.48, restored image by PSGM with time: 0.95 s, iterations: 20, MSE: 5.8083×10, and SNR: 18.38, restored image by CGD with time: 40.16 s, iterations: 645, MSE: 1.0997×10^2, and SNR: 19.90, and restored image by TPRP with time: 552.25 s, iterations: 175, MSE: 1.0642×10^2, and SNR: 20.04.

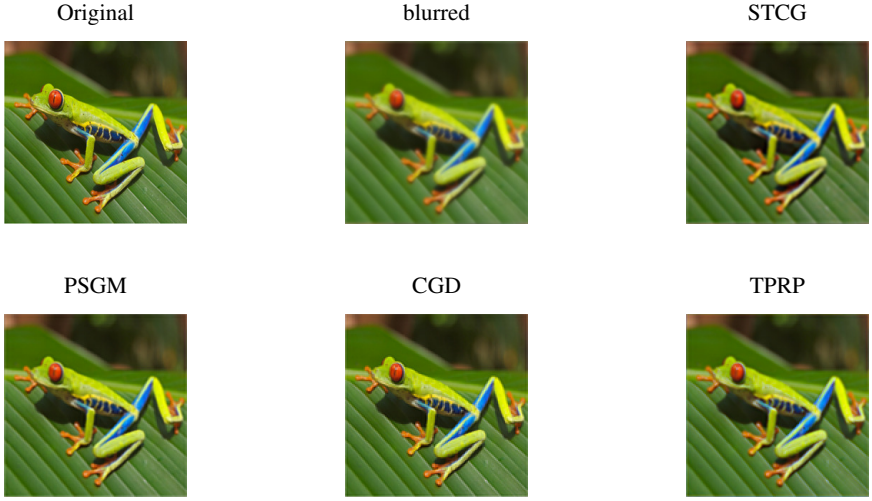

Figure 7. Original image, blurred image, restored image by STCG with time: 0.38 s, iterations: 7, MSE: 1.9752×10^2, and SNR: 12.06, restored image by PSGM with time: 1.44 s, iterations: 27, MSE: 1.3110×10^2, and SNR: 13.84, restored image by CGD with time: 61.20 s, iterations: 935, MSE: 1.2850×10^2, and SNR: 13.92, and restored image by TPRP with time: 770.35 s, iterations: 512, MSE: 1.2772×10^2, and SNR: 13.95.

5. Conclusions

We proposed a modified conjugate gradient parameter by combining the descent three-term CG direction with the well-known Newton direction by adopting the Birgin and Martínez strategy. We also demonstrated the efficacy of the proposed CG parameter by proposing a scaled three-term conjugate gradient algorithm for solving the system of monotone equations with convex constraints with an application in image restoration problems. Furthermore, we showed the global convergence analysis of the proposed algorithm using the Lipchitz continuous assumption. The proposed algorithm is a viable alternative for solving convex constraint monotone equations with fewer iterations. It is also demonstrated that the suggested technique can restore blurred pictures with less MSE, iterations, and computational time. We further highlight that the proposed CG parameter

can be used in many fields of CG method applications, such as the symmetric system of nonlinear equations, unconstrained optimization problems, and many more. It is vital to note that the method's stability analysis will be taken into account in our future work.

Author Contributions: Formal analysis, K.O.A.; investigation, J.S.; software, A.A.; supervision, A.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.

Acknowledgments: The second author acknowledges with thanks, the Department of Mathematics and Applied Mathematics at the Sefako Makgatho Health Sciences University.

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

References

1. Iusem, A.N.; Solodov, M.V. Newton-type methods with generalized distances for constrained optimization. *Optimization* **1997**, *41*, 257–278. [CrossRef]
2. 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. Financ.* **2020**, *52*, 101174. [CrossRef]
3. Figueiredo, M.; Nowak, R.; 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]
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. Li, D.; Fukushima, M. A Globally and Superlinearly Convergent Gauss-Newton-Based BFGS Method for Symmetric Nonlinear Equations. *SIAM J. Numer. Anal.* **1999**, *37*, 152–172. [CrossRef]
6. Waziri, M.Y.; Sabi'u, J. A derivative-free conjugate gradient method and its global convergence for solving symmetric nonlinear equations. *Int. J. Math. Math. Sci.* **2015**, *2015*, 961487. [CrossRef]
7. Sabi'u, J.; Muangchoo, K.; Shah, A.; Abubakar, A.B.; Aremu, K.O. An inexact optimal hybrid conjugate gradient method for solving symmetric nonlinear equations. *Symmetry* **2021**, *13*, 1829. [CrossRef]
8. Sabi'u, J.; Muangchoo, K.; Shah, A.; Abubakar, A.B.; Jolaoso, L.O. A modified PRP-CG type derivative-free algorithm with optimal choices for solving large-scale nonlinear symmetric equations. *Symmetry* **2021**, *13*, 234. [CrossRef]
9. Ortega, J.M.; Rheinboldt, W.C. *Iterative Solution of Nonlinear Equations in Several Variables*; Academic Press: Cambridge, MA, USA, 1970.
10. Zhou, G.; Toh, K.C. Superlinear convergence of a Newton-type algorithm for monotone equations. *J. Optimiz. Theory Appl.* **2005**, *125*, 205–221. [CrossRef]
11. 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]
12. 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]
13. 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.* **2022**, *99*, 332–354. [CrossRef]
14. Sabi'u, J.; Shah, A. An efficient three-term conjugate gradient-type algorithm for monotone nonlinear equations. *RAIRO Oper. Res.* **2021**, *55*, S1113–S1127. [CrossRef]
15. 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]
16. 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*, 343–373. [CrossRef]
17. 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]
18. 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]
19. 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]

20. Gao, P.; He, C. An efficient three-term conjugate gradient method for nonlinear monotone equations with convex constraints. *Calcolo* **2018**, *55*, 1–17. [CrossRef]
21. Liu, J.; Feng, Y. A derivative-free iterative method for nonlinear monotone equations with convex constraints. *Numer. Algorithms* **2019**, *82*, 245–262. [CrossRef]
22. 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]
23. 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]
24. Koorapetse, M.; Kaelo, P.; Lekoko, S.; Diphofu, T. A derivative-free RMIL conjugate gradient projection method for convex constrained nonlinear monotone equations with applications in compressive sensing. *Appl. Numer. Math.* **2021**, *165*, 431–441. [CrossRef]
25. Halilu, A.S.; Majumder, A.; Waziri, M.Y.; Awwal, A.M.; Ahmed, K. On solving double direction methods for convex constrained monotone nonlinear equations with image restoration. *Comput. Appl. Math.* **2021**, *40*, 1–27. [CrossRef]
26. Aji, S.; Kumam, P.; Awwal, A.M.; Yahaya, M.M.; Kumam, W. Two hybrid spectral methods with inertial effect for solving system of nonlinear monotone equations with application in robotics. *IEEE Access* **2021**, *9*, 30918–30928. [CrossRef]
27. Amini, K.; Faramarzi, 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. Comput. Math.* **2022**, 2047180. [CrossRef]
28. Waziri, M.Y.; Ahmed, K.; Halilu, A.S. A modified PRP-type conjugate gradient projection algorithm for solving large-scale monotone nonlinear equations with convex constraint. *J. Comput. Appl. Math.* **2022**, *406*, 114035. [CrossRef]
29. 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]
30. Waziri, M.Y.; Ahmed, K.; Halilu, A.S.; Sabi'u, J. Two new Hager–Zhang iterative schemes with improved parameter choices for monotone nonlinear systems and their applications in compressed sensing. *RAIRO Oper. Res.* **2022**, *56*, 239–273. [CrossRef]
31. Meli, E.; Morini, B.; Porcelli, M.; Sgattoni, C. Solving nonlinear systems of equations via spectral residual methods: Stepsize selection and applications. *J. Sci. Comput.* **2022**, *90*, 1–41. [CrossRef]
32. Narushima, Y.; Yabe, H.; Ford, J.A. A three-term conjugate gradient method with sufficient descent property for unconstrained optimization. *SIAM J. Optim.* **2011**, *21*, 212–230. [CrossRef]
33. Birgin, E.; Martinez, J.M. A spectral conjugate gradient method for unconstrained optimization. *Appl. Math. Optim.* **2001**, *43*, 117–128. [CrossRef]
34. 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]
35. 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]
36. Hu, Y.; Wei, Z. A modified Liu-Storey conjugate gradient projection algorithm for nonlinear monotone equations. *Int. Math. Forum.* **2014**, *9*, 1767–1777. [CrossRef]
37. Dolan, E.D.; More, J.J. Benchmarking optimization software with performance profiles. *Math. Program.* **2002**, *91*, 201–213. [CrossRef]
38. 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]
39. Beck, A.; Teboulle, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sci.* **2009**, *2*, 183–202. [CrossRef]
40. Van den Berg, E.; Friedlander, M.P. Probing the pareto frontier for basis pursuit solutions. *SIAM J. Sci. Comput.* **2008**, *31*, 890–912. [CrossRef]
41. 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. Theory Methods Appl.* **2011**, *74*, 3570–3577. [CrossRef]
42. Pang, J.S. Inexact Newton methods for the nonlinear complementarity problem. *Math. Program.* **1986**, *36*, 54–71. [CrossRef]
43. 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]
44. Xiao, Y.H.; Zhu, H. A conjugate gradient method to solve convex constrained monotone equations with applications in compressive sensing. *J. Math. Anal.* **2013**, *405*, 310–319. [CrossRef]
45. Ibrahim, A.H.; Deepho, J.; Abubakar, A.B.; Adamu, A. A three-term Polak-Ribiére-Polyak derivative-free method and its application to image restoration. *Sci. Afr.* **2021**, *13*, e00880.