Filomat 39:34 (2025), 12167–12187
https://doi.org/10.2298/FIL2534167S

Published by Faculty of Sciences and Mathematics,
University of Niš, Serbia
Available at: http://www.pmf.ni.ac.rs/filomat

Improving communication in conjugate gradient solvers using regularized variable s-step approach

Hojjatollah Shokri Kaveh[a], Masoud Hajarian[a,*], Anthony T. Chronopoulos[b,c]

[a] *Department of Applied Mathematics, Faculty of Mathematical Sciences, Shahid Beheshti University, Tehran, Iran*
[b] *The Shanghai Institute for Mathematics and interdisciplinary sciences research institute (SIMIS). Shanghai 200433, China*
[c] *Department of Computer Science, Faculty of Computer Sciences, The University of Texas at San Antonio, Texas, USA*

Abstract. The signal (or image) recovery problem typically involves solving a linear system with one or multiple right-hand sides. In this paper, we propose algorithms for addressing these types of linear systems. The conjugate gradient (CG) method is widely utilized to solve symmetric positive definite (SPD) linear systems arising from signal problems. Communication significantly impacts the performance of this algorithm. This paper presents algorithms aimed at reducing communication while solving linear systems with single and multiple right-hand sides, utilizing both CG and cooperative CG algorithms. Subsequently, the performance of the proposed algorithms is enhanced by employing techniques such as varying the parameter s in s-step algorithms and adjusting the parameters in regularization. Finally, the results of these algorithms and the effectiveness of the techniques used are demonstrated through several numerical examples.

1. Introduction

Krylov subspace methods (KSMs) are one of the most efficient iterative methods to solve large and sparse linear systems arising from engineering problems [4, 8, 12, 27, 28, 37, 49, 50]. The CG method is one of the first KSM, introduced by Hestenes and Stiefel in 1952 for solving the linear system

$$Ax = b, \tag{1}$$

where $A \in R^{n\times n}$ is SPD matrix and $x, b \in R^n$. Over the years, new versions of the CG algorithm have emerged, each offering benefits such as improved accuracy, faster convergence, and synchronization capabilities [34].

2020 *Mathematics Subject Classification*. Primary 65F18, 65F10; Secondary 65J10.
Keywords. Conjugate gradient method, Multiple right-hand sides, s-step algorithm, Variable s-step CG, Regularization method.
Received: 10 April 2025; Revised: 08 September 2025; Accepted: 14 October 2025
Communicated by Dijana Mosić
The work of A.T. Chronopoulos was supported by the Research Start-up Grant 2302-SRFP-2024-0056 of the Shanghai Institute for Mathematics and Interdisciplinary Sciences (SIMIS).
* Corresponding author: Masoud Hajarian
Email addresses: h_shokrikaveh@sbu.ac.ir, hojatshokri110@gmail.com (Hojjatollah Shokri Kaveh), m_hajarian@sbu.ac.ir, masoudhajarian@gmail.com (Masoud Hajarian), antony.tc@gmail.com (Anthony T. Chronopoulos)
ORCID iDs: https://orcid.org/0000-0003-2759-2061 (Hojjatollah Shokri Kaveh), https://orcid.org/0000-0002-5549-9270 (Masoud Hajarian), https://orcid.org/0000-0002-0094-1017 (Anthony T. Chronopoulos)

When implementing CG or KSMs to solve linear systems, considerable time is often required to transfer data between layers of main memory and fast memory. Therefore, it is crucial to develop algorithms that optimize communication between memory layers and processors, ultimately reducing the overall runtime of the algorithm. Communication-avoiding (CA) algorithms, which are based on s-step iterative methods, play a pivotal role in this optimization. To further enhance the computational efficiency of algorithms, various techniques are employed to minimize communication costs. Initially, Rosendale introduced methods aimed at this purpose [45]. Chronopoulos and Gear later expanded upon these methods with their s-step technique [13], which was subsequently extended to address non-symmetric problems [15].

In efforts to minimize communication in numerical linear algebra algorithms [5], researchers have redefined matrix multiplication and powers in innovative ways. Both parallel and sequential matrix decomposition methods were proposed to reduce communication optimally [16]. The GMRES communication-avoiding algorithm utilized $ILU(0)$ preconditioning, yielding positive results [18]. Additionally, the s-step CG method was utilized to further reduce communication through graph representations of matrix problems [31]. The s-step block Orthomin(k) technique, also known as BOSOmin(s,b,k), was introduced in [14]; specifically, for $k = 1$, BOSOmin(s,b,1) corresponds to the block s-step conjugate residual (CR) method.

Walker et al. explored enhancements to the stability of the GMRES method by improving the bases in traditional KSMs using the modified Gram-Schmidt (MGS) and Householder's orthogonalization processes [19, 47]. The MGS process was applied for s-step methods in [15]. It was noted in [15] that using a monomial basis may lead to instability, as convergence was not guaranteed for $s > 6$.

Hindmarsh et al. utilized a scaled (normalized) monomial basis to enhance convergence; however, this resulted in only marginal improvements [19]. Joubert et al. employed a scaled and transposed Chebyshev basis, which led to improved accuracy in traditional KSM [23]. In [3], Bai et al. achieved better convergence in traditional KSM by using Newton's basis. While correctly scaling the basis vectors reduces the condition number of the basis matrix, it may bring the convergence closer to standard KSM and reintroduce dependency calculations, which the correlation reduction algorithm aims to eliminate [3].

Hoemmen addressed this issue by introducing a new balance matrix and method as a preconditioning step to eliminate the need for scaled basis vectors [20]. In [21], variable preconditioning was implemented in CA method for solving electromagnetic problems. To tackle ill-conditioning in Krylov bases, Chebyshev polynomial bases were used instead of monomial bases in [10]. Furthermore, Chebyshev and Newton polynomials were integrated for the s-step CG method with dynamic basis updates [11]. The s-step BiCGSTAB method was introduced alongside new preconditioning methods to address the ill-conditioning of the problem's matrix [33].

In [17, 29], various communication reduction algorithms were derived. Other CA algorithms, including CGNR, CGNE, BiCG, CGS, and BiCGSTAB, were proposed in [9, 24, 39]. These methods are commonly employed iterative techniques for solving large (particularly sparse) systems on high-performance computing (HPC) platforms.

Adjusting both the parameter s and the regularization filter has been effectively implemented to enhance algorithm performance. This technique was applied to the BiCG, BiCGstab, CGS, CGNE, and CGNR algorithms [24, 39]. The semi-CG method was generalized and subsequently applied to signal reconstruction problems by incorporating the s-step technique along with regularization [26]. The Planner algorithm is utilized to solve an indefinite linear system, employing the s-step technique in its implementation [41]. A new variant of the CR algorithm was introduced, incorporating the s-step technique into its design [40]. In [42], the s-step technique was employed in algorithms designed to solve shifted linear systems. This technique was also applied to the algorithm for solving Sylvester matrix equations[25].

To solve the image (or signal) recovery problem, it is generally necessary to compute the inverse of an operator (or matrix). Instead of directly computing the inverse, iterative algorithms, such as KSMs, are employed to solve the linear system. In these scenarios, an operator is applied to the source signal, producing a different signal. To recover the original signal, the corresponding linear system should be solved. Therefore, this paper focuses on solving linear systems using Krylov subspace algorithms, particularly enhancing the CG algorithm to effectively tackle the linear systems arising from the signal recovery problem.

This paper introduces the s-step CG method for systems with both single and multiple right-hand sides

$$AX = B, \tag{2}$$

where $A \in R^{n\times n}$ is SPD matrix and $X, B \in R^{n\times m}$. This method by adjusting the number of steps while maintaining method convergence, communication is minimized. Furthermore, by utilizing a regularization technique involving shifted eigenvalues, we mitigate fluctuations in the approximation sequence, thereby introducing the regularized variable s-step CG method.
The remainder of the paper is organised as follows. In Section 2, we first present the CG and cooperative CG algorithms. Then based on these algorithms, the s-step algorithms for symmetric positive definite linear systems with single or multiple right-hand sides are introduced. Section 3 provides suggestions for selecting s in each iteration according to the function related to the parameters of the algorithm. Additionally, a regularization method involving the shifting of the eigenvalues of the problem is proposed, followed by the introduction of the Jacobi preconditioning method to derive the final version of the variable s-step CG algorithms. Section 4 conducts a brief analysis of the method's error and the effect of regularization. Finally, Section 5 presents numerical results for each algorithm through several numerical tests on different matrices, highlighting the degree of improvement in the approximations. Finally in Section 6, conclusions are presented.

2. Description of s-step CG methods

The CG method is one of the KSMs for solving symmetric positive definite linear systems. This algorithm is utilized in various branches of science to tackle different systems. In recent years, the algorithm of this method has been developed in various ways and applied to address a range of problems [2, 32, 43, 51]. Here, the initial version of the CG algorithm known as the Hestenes-Stiefel CG (HS-CG) method is used and presented in Algorithm 1.

Algorithm 1 CG algorithm for solving $Ax = b$

Choose x_0
$p_0 = r_0 = b - Ax_0$
For $i = 0$ Until Convergence Do
Compute and Store Ap_0
Compute $\langle p_i, Ap_i\rangle$
$\alpha_i = \frac{\langle r_i, r_i\rangle}{\langle p_i, Ap_i\rangle}$
$x_{i+1} = x_i + \alpha_i p_i$
$r_{i+1} = r_i - \alpha_i Ap_i$
Compute $\langle r_{i+1}, r_{i+1}\rangle$
$\beta_i = \frac{\langle r_{i+1}, r_{i+1}\rangle}{\langle r_i, r_i\rangle}$
$p_{i+1} = r_{i+1} + \beta_i p_i$
$i = i + 1$
End For.

In the s-step CG method, the new s directions are determined concurrently from the Krylov bases, which are derived from the residual vector r at each iteration. These s directions should be selected to ensure orthogonality with the preceding directions, and the approximation is obtained by minimizing the residual function in the s directions simultaneously. Consequently, each iteration of the s-step method corresponds to the sth iteration of the standard (one-step) CG method. This approach reduces computational tasks and memory usage, while significantly enhancing parallelization capabilities and data locality [30].

Algorithm 2 s-step CG algorithm for solving $Ax = b$

Choose x_0 and s_0
$r_0 = b - Ax_0$
$P = [r_0, Ar_0, A^2r_0, ...A^{s_0-1}r_0]$
For $i = 0$ Until Convergence Do
$\alpha_i = (P_i^tAP_i)^{-1}(P_i^tr_i)$
$x_{i+1} = x_i + P_i\alpha_i$
$r_{i+1} = r_i - AP_i\alpha_i$
Choose s_{i+1}
$B_{i+1} = [r_{i+1}, Ar_{i+1}, A^2r_{i+1}, ...A^{s_{i+1}-1}r_{i+1}]$
$\beta_i = (P_i^tAP_i)^{-1}(P_i^tAB_{i+1})$
$P_{i+1} = B_{i+1} + P_i\beta_i$
$i = i + 1$
End For.

In the cooperative CG method, instead of starting with a single initial guess, several initial guesses are utilized, allowing the system to be solved multiple times in parallel. This method is introduced in [7]. The cooperative CG algorithm is presented in Algorithm 3.

Algorithm 3 Cooperative CG algorithm for solving $AX = B$

Choose $X_0 \in R^{n\times t}$
$B = [b|b|...|b]_{n\times t}$
$R_0 = B - AX_0$
$P_0 = R_0$
For $i = 0$ Until Convergence Do
$\alpha_i = (P_i^tAP_i)^{-1}(R_i^tP_i)$
$X_{i+1} = X_i + P_i\alpha_i$
$R_{i+1} = R_i - AP_i\alpha_i$
$\beta_i = (P_i^tAP_i)^{-1}(P_i^tAR_{i+1})$
$P_{i+1} = R_{i+1} + P_i\beta_i$
$i = i + 1$
End For.

Based on the cooperative CG method, we formulate the s-step CG method for systems with multiple right-hand sides. The primary difference between the two approaches lies in the right-hand side of the problem, which can be easily adapted. The s-step CG algorithm for systems with multiple right-hand sides is presented in Algorithm 4.

Algorithm 4 s-step CG algorithm for solving $AX = B$

Choose $X_0 \in R^{n\times t}$
$R_0 = B - AX_0$
$P_0 = R_0$
For $i = 0$ Until Convergence Do
$\alpha_i = (P_i^t AP_i)^{-1}(R_i^t P_i)$
$X_{i+1} = X_i + P_i\alpha_i$
$R_{i+1} = R_i - AP_i\alpha_i$
Choose s_{i+1}
$W_{i+1} = [R_{i+1}, AR_{i+1}, A^2R_{i+1}, ..., A^{s_{i+1}-1}R_{i+1}]$
$\beta_i = (P_i^t AP_i)^{-1}(P_i^t AW_{i+1})$
$P_{i+1} = W_{i+1} + P_i\beta_i$
$i = i + 1$
End For.

3. Selection of variable s

In the literature review of s-step methods, it is observed while increasing s can reduce communication costs, a large s may result algorithmic instability. Determining an optimal value of s for different problems is challenging. An alternative approach is to dynamically adjust s at each step based on the decreasing residual norm. One method for determining the parameter s involves using numerical sequences within the GMRES algorithm [22], where the parameter s is derived from the vectors within the algorithm. In the CG algorithm, the parameter α_i, and in the s-step algorithm, the norm of α ($\|\alpha\|_2$), are closely related to the residual norm. By decreasing these parameters and reducing the residual norm, the parameter s can be increased by

$$s \propto \frac{1}{\|\alpha\|_2}. \tag{3}$$

The selection of the parameter s is inversely related to the residual norm and directly related to the selection of direction vectors. Correctly choosing the direction vector results in a more accurate α, which in turn affects the inner product of the residual in the direction vector. By decreasing α, the parameter s can be increased. For example, if F is a decreasing function with respect to α, then s can be defined as

$$s = F(\alpha).$$

Various functions can be employed for generating F. One simple example is

$$F(\alpha) = \frac{1}{\|\alpha\|_2}.$$

Taking the integer part of this value helps to prevent s from becoming zero. Therefore, in the s-step method, we consider the parameter s as

$$s = 1 + \left\lfloor \frac{1}{\|\alpha\|_2} \right\rfloor, \tag{4}$$

where the symbol $\left\lfloor \frac{1}{\|\alpha\|_2} \right\rfloor$ represents the greatest integer less than or equal to $\frac{1}{\|\alpha\|_2}$. While determining the parameter s as per (4) may suit the classical CG method for single right-hand side problems, it might not be as effective for problems with multiple right-hand sides. To solve this problem, we introduce the sequence

$$s_1 = 1, \quad s_n = \max\left\{ s_{n-1}, \left\lfloor \frac{1}{\|\alpha(:,i)\|_2} \right\rfloor \right\} \quad \forall i = 1, 2, \ldots, s_{n-1}, \tag{5}$$

that performs effectively across various scenarios. This sequence incrementally increases s by reducing the residual norm. In this method, the value of s progressively increases in iterations, maintaining stability for problems with multiple right-hand sides. By determining the parameter s in this way and incorporating it into the algorithms of the previous sections, the s-step method with a variable s can be efficiently utilized.

3.1. Regularization technique with shifting eigenvalues

The regularization method was first proposed by Tikhonov and used to reduce the amount of noise in ill-posed problems by making corrections to the problem matrix, thereby reducing its condition number. Among the regularization methods, we can mention the Tikhonov method [44], the truncated singular value decomposition method (TSVD), the mapped regularization method [38]. The Tikhonov function to regularize the linear system $Ax = b$ is defined as follows

$$\phi(\lambda) = \|Ax - b\|_2^2 + \lambda^2 \|Lx\|_2^2, \tag{6}$$

where ϕ is the filter function, λ is the regularization parameter, and L is an operator. The main problem in the regularization method is to determine the parameter λ in such a way that the Tikhonov function (6) is minimized. This parameter is determined using specialized techniques, and a very effective method is presented in [52]. The mentioned methods aim to move the eigenvalues of the matrix A away from zero, which results in reducing the condition number of this matrix. Here, we will attempt to achieve this by shifting the eigenvalues of the matrix A.
In the CG method, the sequence of approximations in each iteration is obtained from the approximation of the previous iteration as follows

$$x_{i+1} = x_i + \alpha p. \tag{7}$$

If we want to move forward in each s-step, this equation can be expressed as

$$x_0, x_0 + \alpha_1 p_1, x_0 + \alpha_1 p_1 + \alpha_2 p_2, \ldots, x_0 + \sum_{i=1}^{s} \alpha_i p_i = x_0 + P\alpha, \tag{8}$$

where $P \in \mathbb{R}^{n\times s}$ is the matrix of bases and $\alpha \in \mathbb{R}^{s\times 1}$ is the vector that includes the α_i coefficients. Without loss of generality, let us assume that the initial guess is zero, then we have

$$x_s = P\alpha. \tag{9}$$

By comparing (9) with the Fourier expansion of x_s on the basis vectors P

$$x_s = \sum_{i=1}^{s} \sigma_i^2 \langle x_s, p_i \rangle p_i, \tag{10}$$

we obtain

$$\alpha_i = \sigma_i^2 \langle x_s, p_i \rangle \tag{11}$$

On the other hand, in the CG algorithm we have

$$\alpha_i = \frac{\|r_i\|_2}{p_i^t A p_i} \tag{12}$$

From $\frac{1}{p_i^t A p_i}$ approximates the eigenvalues of the matrix A^{-1}, the above equation can be expressed as

$$\alpha_i = \frac{1}{\lambda_i} \|r_i\|_2, \tag{13}$$

where λ_i is the eigenvalue of the matrix A. To regularize the CG method, we use the fact that the α parameter contains the eigenvalues of the matrix A^{-1}. Now we can utilize this point to obtain α. Noting to Tikhonov regularization

$$\phi(\alpha) = \frac{\alpha}{\alpha + 1}, \tag{14}$$

we suggest shifting the eigenvalues of A^{-1} away from near zero by

$$\alpha_i = \frac{\langle r_i, r_i \rangle}{p_i^t (A + \mu_i I) p_i}, \tag{15}$$

The parameter μ_i should be determined in such a way that it changes with a greater rate corresponding to the sequence of eigenvalues of the matrix A, as this effectively moves the small eigenvalues of A^{-1} away from zero. Therefore we have

$$\alpha_i = \frac{\langle r_i, r_i \rangle}{p_i^t \left(A + \frac{\alpha_i^2}{\alpha_i + 1} I \right) p_i}, \tag{16}$$

and according to (3), for s-step algorithmthis we obtain

$$\alpha = \frac{\langle r, r \rangle}{P^t \left(A + \frac{1}{s(s+1)} I \right) P}. \tag{17}$$

In the above approximation, the parameter α of the problem matrix is given as a shift of $\mu = \frac{1}{s(s+1)}$. This ensures that the approximation of the eigenvalues of the matrix A^{-1} within α does not become excessively large.

3.2. Jacobi preconditioner

When solving linear systems, an ill-conditioned matrix makes the solution highly sensitive to errors. Specifically, a small error in the problem's input can lead to a significant error in the solution.
Preconditioning methods were used for KSMs and yielded favorable results[35, 46, 48]. In s-step KSMs, the basis matrix is typically constructed using the problem matrix. As a result, the condition number of the Krylov basis matrix is often proportional to the condition number of the problem matrix. Therefore, preconditioning is employed to reduce the condition number of the problem matrix, which in turn lowers the condition number of the Krylov basis matrix.
The preconditioner matrix is generally an approximation of the inverse of the problem matrix, obtained through various methods. Multiplying this matrix on both sides of the problem reduces the condition number of the system matrix compared to that of the original system matrix.
Various methods can be used to derive the preconditioner matrix, primarily through matrix decompositions of the problem matrix. Classical methods such as Jacobi, Gauss-Seidel, and Successive Over-Relaxation (SOR), as well as more recent methods like incomplete LU and incomplete Cholesky, are popular for constructing the preconditioner matrix.
By inverse of diagonal elements of the problem matrix, Jacobi preconditioner is obtained as

$$M = \operatorname{diag}\left(\frac{1}{a_{11}}, \frac{1}{a_{22}}, \ldots, \frac{1}{a_{nn}} \right). \tag{18}$$

The Jacobi preconditioner is computationally efficient due to its sparsity and ease of computation, but it is a weak approximation of inverse of the problem matrix [6].

4. Error analysis

In this section, we seek to present the error bound for the regularized variable s-step CG method. First the error of the iterations of the CG method is given.

Theorem 4.1. *[36] Suppose that x_N is the approximate value obtained from the CG method in N iterations and x^* is the exact solution. Then,*

$$\|x^* - x_N\|_A \leq 2\left(\frac{\sqrt{K_2(A)}-1}{\sqrt{K_2(A)}+1}\right)^N \|x^* - x_0\|_A, \tag{19}$$

where $K_2(A)$ is the condition number of the matrix A in 2-norm and x_0 is the initial guess.

The regularization error bound can be expressed as follows.

Theorem 4.2. *Suppose that x_i is the approximate value obtained from the CG method and x_N^{reg} is the regularized solution in N iterations. Then,*

$$\|x_N - x_N^{reg}\|_2 \leq \sum_{i=1}^{N} \frac{\mu_i}{\lambda_i(\lambda_i+\mu_i)}\|r_i\|_2, \tag{20}$$

where λ_i is the eigenvalue of the matrix A and r_i is the residual in the i-th iteration.

Proof. First, we can write

$$\|x_N - x_N^{reg}\|_2 \leq \left\|\sum_{i=1}^{N}\frac{\|r_i\|_2}{\lambda_i}p_i - \sum_{i=1}^{N}\frac{\|r_i\|_2}{\lambda_i+\mu_i}p_i\right\|_2 = \left\|\sum_{i=1}^{N}\left(\frac{1}{\lambda_i}-\frac{1}{\lambda_i+\mu_i}\right)\|r_i\|_2 p_i\right\|_2. \tag{21}$$

Since λ_i and μ_i are positive, we can derive:

$$\left\|\sum_{i=1}^{N}\left(\frac{1}{\lambda_i}-\frac{1}{\lambda_i+\mu_i}\right)\|r_i\|_2 p_i\right\|_2 \leq \sum_{i=1}^{N}\frac{\mu_i}{\lambda_i(\lambda_i+\mu_i)}\|r_i\|_2\|p_i\|_2. \tag{22}$$

□

Now, the general error bound of the regularized CG method can be obtained as the sum of the algorithm cutoff error bound and the regularization error bound.

Theorem 4.3. *Suppose that x^* is the exact solution and x_N^{reg} is the regularized solution in the N-th iteration. Then,*

$$\|x^* - x_N^{reg}\|_A \leq \|x^* - x_N\|_A + \|x_N - x_N^{reg}\|_2 \tag{23}$$

$$\leq 2\left(\frac{\sqrt{K_2(A)}-1}{\sqrt{K_2(A)}+1}\right)^N \|x^* - x_0\|_A + \sum_{i=1}^{N}\frac{\mu_i}{\lambda_i(\lambda_i+\mu_i)}\|r_i\|_2\|p_i\|_2, \tag{24}$$

where λ_i is the eigenvalue of the matrix A and r_i is the residual in the i-th iteration.

Proof. The proof of this theorem follows from Theorems 4.1 and 4.2. □

In the case that the s-step CG algorithm is regularized using (17), we choose μ_i to be small. Then the regularization error will also be small. However, it should be noted that as s_i increases, the algorithm may be interrupted due to $N = \sum s_i$. Therefore, the parameter μ plays a controlling role in the upper error bound.

The regularized s-step CG algorithms for the systems with single and multiple right-hand sides are presented in Algorithms 5 and 6. In these algorithms, Jacobi preconditioning is used as a general initial step.

Algorithm 5 Regularized variable s-step CG algorithm for $Ax = b$.

M=Jacobi perconditioner for A
$A = MA$ & $B = Mb$
Choose x_0
$r_0 = B - AX_0$
$P_0 = r_0$
$s_0 = 1$
For $i = 0$ Until Convergence Do
$\alpha_i = (P_i^t(A + \frac{1}{s_i(s_i+1)}I)P_i)^{-1}(P_i^t r_i)$
$s_{i+1} = 1 + \lfloor \frac{1}{\|\alpha_i\|_2} \rfloor$
$x_{i+1} = x_i + P_i\alpha_i$
$r_{i+1} = r_i - AP_i\alpha_i$
$B_{i+1} = [r_{i+1}, Ar_{i+1}, A^2r_{i+1}, ..., A^{s-1}r_{i+1}]$
$\beta_i = (P_i^t AP_i)^{-1}(P_i^t AB_{i+1})$
$P_{i+1} = B_{i+1} + P_i\beta_i$
$i = i + 1$
End For.

Algorithm 6 Regularized variable s-step CG algorithm for systems with multiple right-hand sides $AX = B$.

M=Jacobi perconditioner for A
$A = MA$ & $B = MB$
Choose $X_0 \in R^{n\times t}$
$R_0 = B - AX_0$
$P_0 = R_0$
$s_0 = 1$
For $i = 0$ Until Convergence Do
$\alpha_i = (P_i^t(A + \frac{1}{s_i(s_i+1)}I)P_i)^{-1}(R_i^t P_i)$
$s_{i+1} = \max\{s_{i-1}, \lfloor \frac{1}{\|\alpha_i(:,j)\|_2} \rfloor \quad \forall j = 1, 2, ..., s_{i-1}\}$
$X_{i+1} = X_i + P_i\alpha_i$
$R_{i+1} = R_i - AP_i\alpha_i$
$W_{i+1} = [R_{i+1}, AR_{i+1}, A^2R_{i+1}, ..., A^{s_i-1}R_{i+1}]$
$\beta_i = (P_i^t AP_i)^{-1}(P_i^t AW_{i+1})$
$P_{i+1} = W_{i+1} + P_i\beta_i$
$i = i + 1$
End For.

5. Numerical tests

In this section, we present several numerical examples. In each example, we compare the performance of the original CG algorithm (with $s = 1$) to our proposed algorithms. The condition numbers of the matrices used in these examples are listed in Table 1 [1]. In all cases, the initial guess is a zero vector (a vector with all elements equal to zero), and the right-hand side values consist of a one vector (a vector with all elements equal to one). Additionally, we employ the Jacobi preconditioner. In the first example, the parameter s for each iteration is determined using (4). As demonstrated, the variable s-step CG method achieves a nearly accurate solution in fewer iterations while using the same number of basis vectors. In subsequent examples, we determine the parameter s for each step using the sequence introduced in (5), which effectively reduces the number of iterations required by the algorithm. In the first example, the value

of s is determined using (4), while in the following examples, it is determined using (5). Procedures are implemented in Matlab and computations are performed on a computer with 2.16 GHz and 4G RAM.

Matrix	N	Norm-2	Condion number	SPD
1138-bus	1138	3.0149×10^{4}	1.2284×10^{7}	yes
662-bus	662	4.0081×10^{3}	8.2712×10^{5}	yes
bcsstk14	1806	1.1923×10^{10}	1.3100×10^{10}	yes
nos1	237	2.4567×10^{9}	2.5344×10^{7}	yes
nos5	468	5.8203×10^{5}	2.9191×10^{4}	yes

Table 1: Information of the matrices.

Example 5.1. In the first example, we compare the standard CG method with the s-step CG method. The value of s is determined using Equation (4). The performance of two mentioned methods for the matrices listed in Table 1 is illustrated in Figure 1. The CPU time for each method is presented in Table 2.

Matrix	CG	v s-step CG
1138-bus	2.4248	1.6112
662-bus	0.9528	0.5986
bcsstk14	5.8803	3.7132
nos1	0.1588	0.1532
nos5	0.5883	0.4424

Table 2: CPU time of CG and variable s-step CG for Example 5.1.

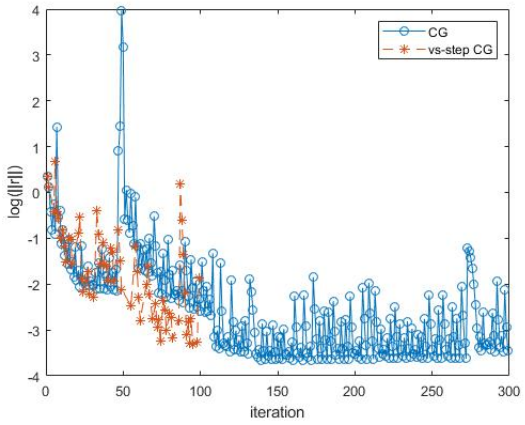
Example 5.2. In the second example, we study the standard cooperative CG method with the s-step cooperative CG method. The initial guesses are random vectors, and the right-hand side values are also random vectors. Additionally, the Jacobi preconditioner is employed with $t = 4$. The performance of the methods for the matrices listed in Table 1 is given in Figure 2, and their respective CPU time are given in Table 3.

Matrix	coop-CG	v s-step coop-CG
1138-bus	6.1140	5.8025
662-bus	2.7677	2.1462
bcsstk14	15.2836	13.5569
nos1	0.5502	0.5481
nos5	1.8980	1.1412

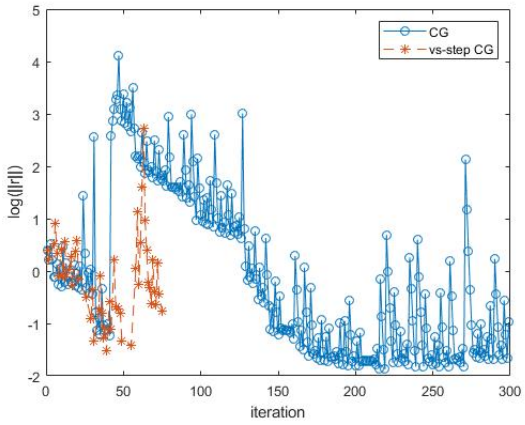
Table 3: CPU time of cooperative CG and variable s-step cooperative CG methods for Example 5.2.

Example 5.3. Here we compare the standard CG method with the s-step CG method for problems involving multiple right-hand sides $AX = B$. The initial guesses are random matrices, and the right-hand side values are also random matrices. The Jacobi preconditioner is also considered with $t = 4$. The results are depicted in Figure 3, and their respective CPU time are reported in Table 4.

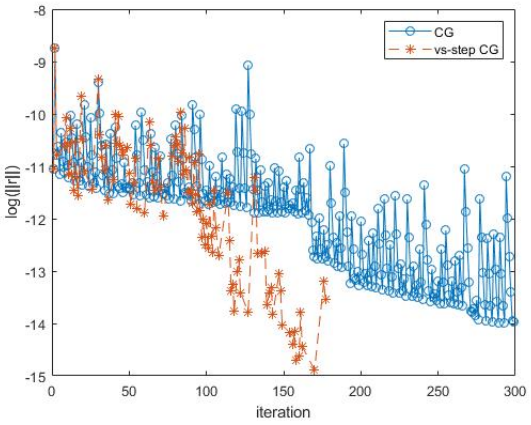
Example 5.4. In this example, the standard CG method is compared with the regularized s-step CG method. The initial guesses are random, and the right-hand side values are also random. Additionally, the Jacobi preconditioner is employed, with $t = 4$. The performance of these two methods for the matrices listed in Table 1 is presented in Figure 4 for the linear system $Ax = b$ and in Figure 5 for the linear system with multiple right-hand sides $AX = B$.



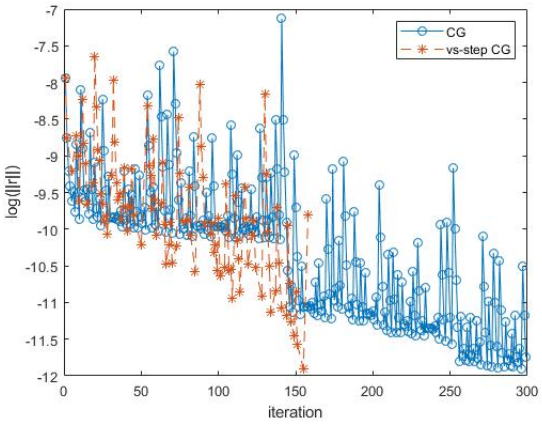
(a) 1138-bus matrix



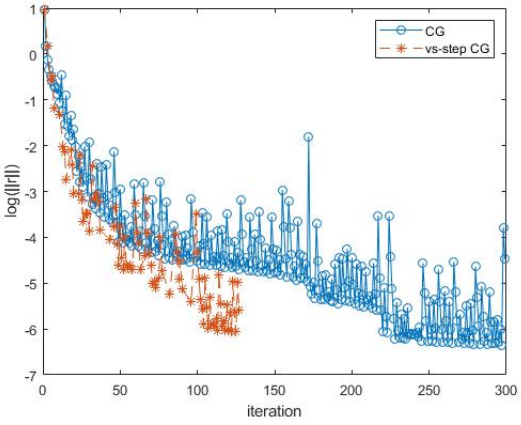
(b) 662-bus matrix



(c) bcsstk14 matrix

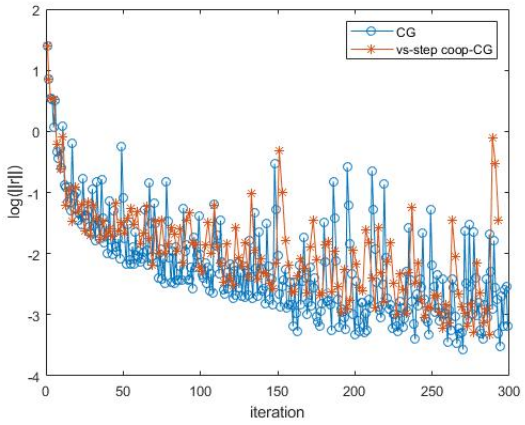


(d) nos1 matrix

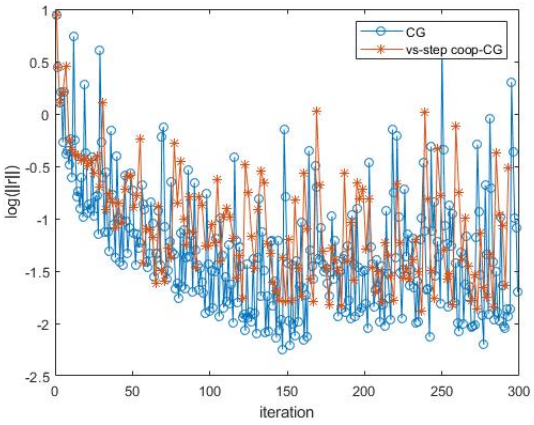


(e) nos5 matrix

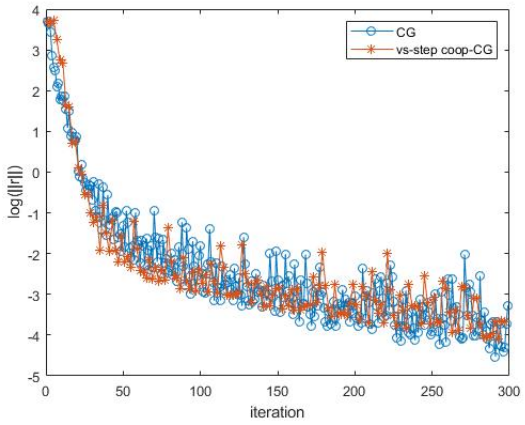
Figure 1: Numerical results of CG and variable s-step CG methods for Example 5.1.



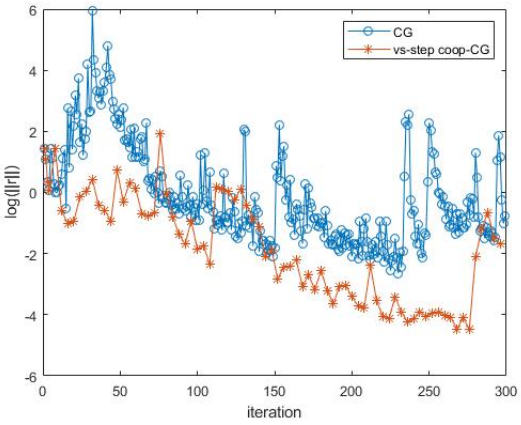
(a) 1138-bus matrix



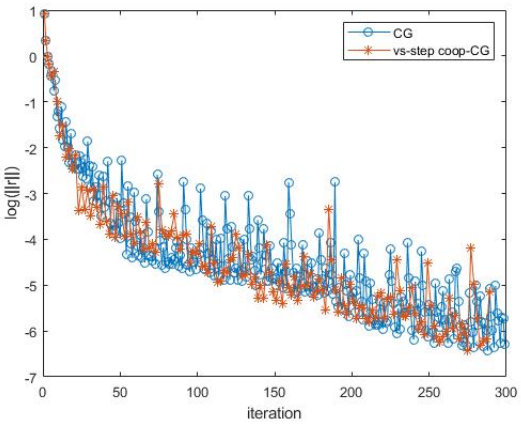
(b) 662-bus matrix



(c) bcsstk14 matrix

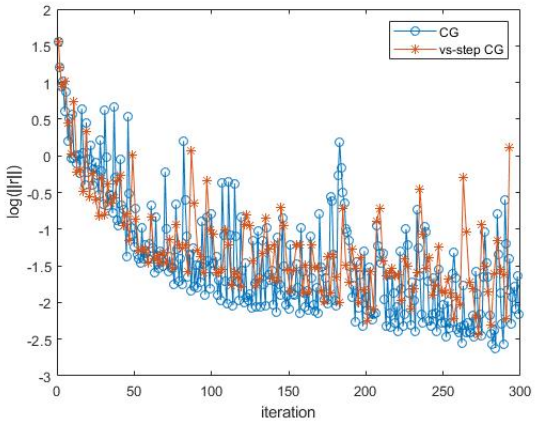


(d) nos1 matrix

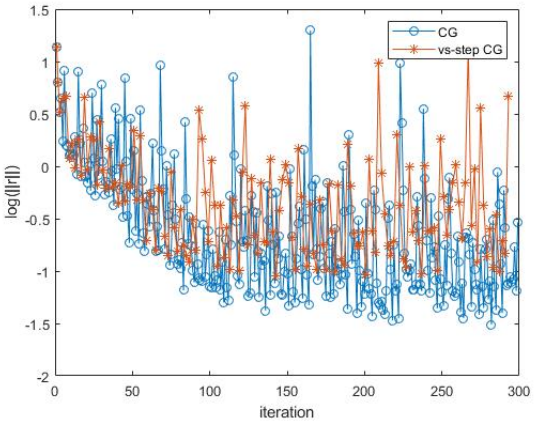


(e) nos5 matrix

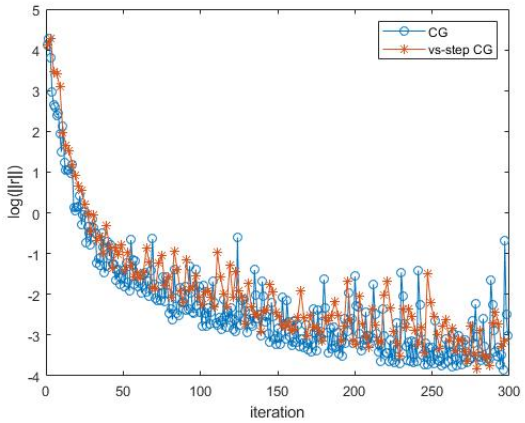
Figure 2: Numerical results of cooperative CG and variable s-step cooperative CG methods for Example 5.2.



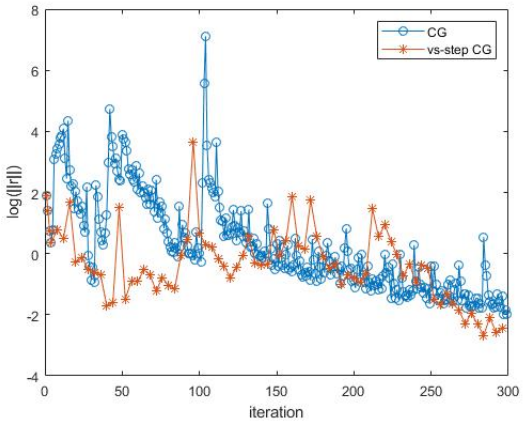
(a) 1138-bus matrix



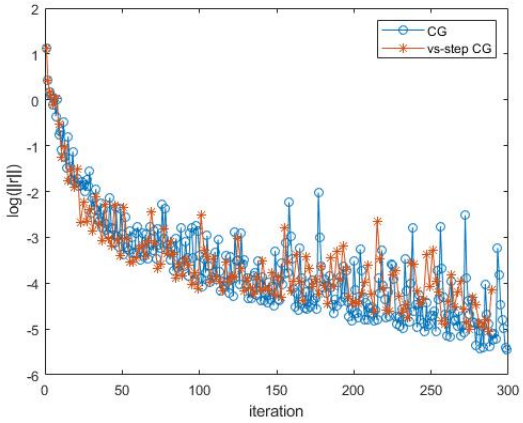
(b) 662-bus matrix



(c) bcsstk14 matrix

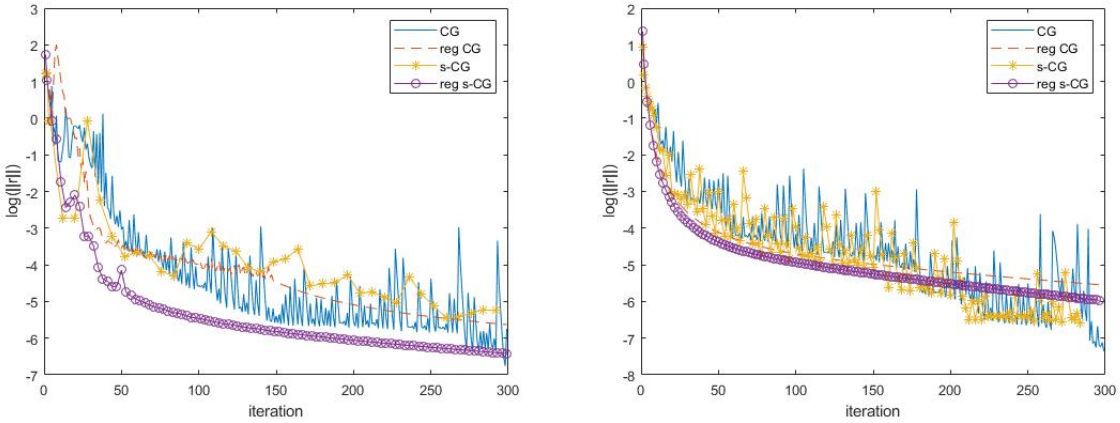


(d) nos1 matrix



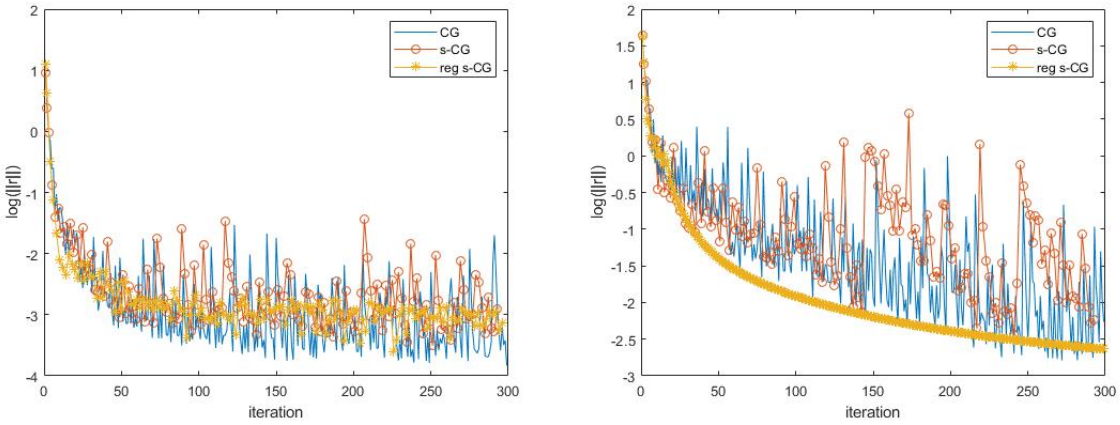
(e) nos5 matrix

Figure 3: Convergence of CG and variable s-step CG methods for Example 5.2.



(a) nos1 matrix (b) nos5 matrix

Figure 4: Convergence of CG and regularized variable s-step CG methods for $Ax = b$ to Example 5.4.



(a) 662-bus matrix (b) 1138-bus matrix

Figure 5: Convergence of CG and regularized variable s-step CG methods for $AX = B$ to Example 5.4.

Matrix	CG	v s-step CG
1138-bus	6.0901	5.6787
662-bus	2.3260	2.1408
bcsstk14	15.0201	13.5653
nos1	0.6120	0.3929
nos5	1.3404	1.1803

Table 4: CPU time of CG and variable s-step CG for multiple right hand sides problems for Example 5.2.

Example 5.5. In this example, multiple images are reconstructed using the algorithm presented for linear systems with multiple right-hand sides. Here the exact solution to the problem is an image that, after applying the operator (matrix) A to it, yields image B. With matrices B and A in hand, we aim to reconstruct the original image. Here, preconditioning was not used, and the parameters are considered as

$$A = gallery('toeppen', n, 2, -5, 20, -5, 2), \quad B = Ax_{Exact}, \tag{25}$$

$$s_1 = 1, \qquad s_i = 1 + \lfloor log(\sum_{j=1}^{i-1} s_j) \rfloor, \tag{26}$$

where X_{Exact} is presented in Table 5. The results are presented in Figures 6 and 7. Also the error of the algorithms is shown in Table 6.

Problem	n×n	SPD
Cameraman	256×256	Yes
Lena	190×190	Yes

Table 5: The Problems in Example 5.5.

Problem	$Error_{CG}$	$Error_{s-CG}$
Cameraman	2.2689e-11	2.0169e-11
Lena	3.6813e-14	3.3379e-14

Table 6: Errors of CG and variable s-step CG methods for Example 5.5.

Example 5.6. As the final example, the performance of the variable s-step technique is demonstrated by reconstructing a one-dimensional signal using the two specified algorithms for a linear system with a single right side. We consider the following parameters

$$A = gallery('toeppen', 100, 4, -10, 18, -10, 4), \quad b = Ax_{Exact}, \tag{27}$$

$$x_{Exact} = S_1 = i \sin(i\frac{\pi}{6}), \qquad x_{Exact} = S_2 = e^{\frac{-(i-\frac{n}{2})^2}{i}}, \tag{28}$$

$$s_1 = 1, \quad \forall i > 1, \quad s_i = 1 + \lfloor \sqrt{\sum_{j=1}^{i-1} s_j} \rfloor. \tag{29}$$

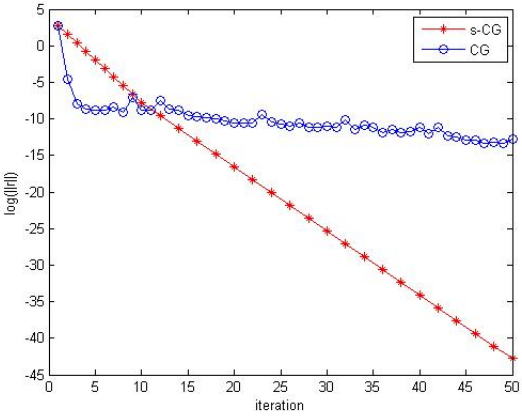
Here preconditioning was not used. Figures 8 and 9 show the reconstructed signal and the residue for each algorithm. Also Table 7 presents the CPU time of the algorithms.

(a) X_{Exact}

(b) B

(c) X_{CG}

(d) X_{s-CG}



(e) *Residuals*

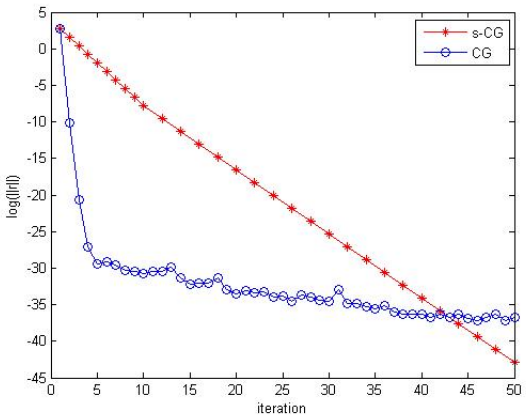
Figure 6: The results for Example 5.5.

(a) X_{Exact}

(b) B

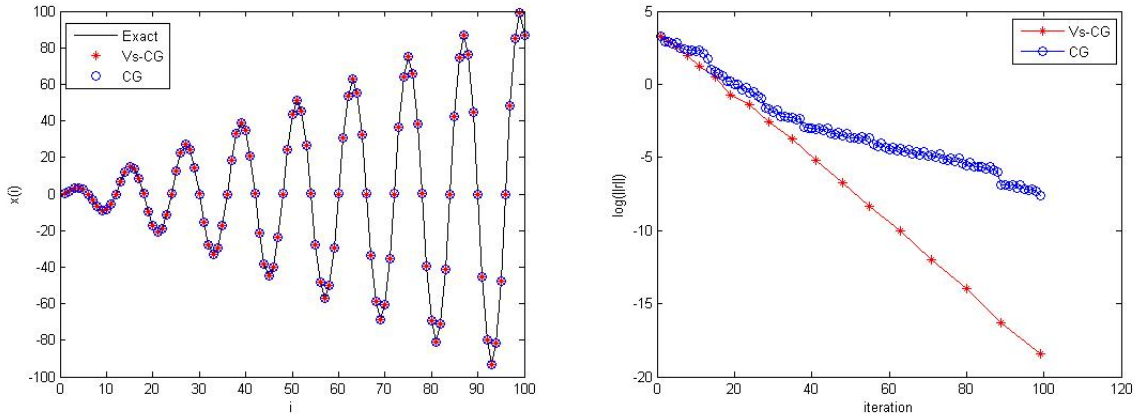
(c) X_{CG}

(d) X_{s-CG}



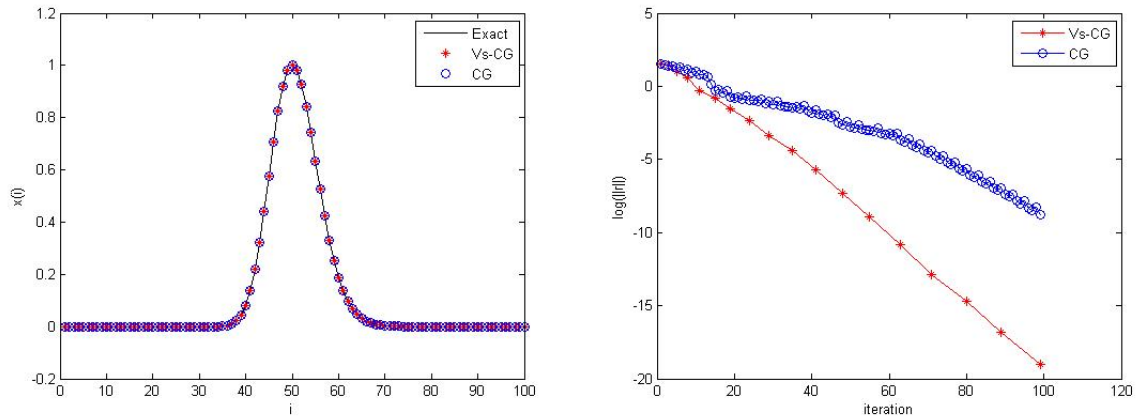
(e) *Residuals*

Figure 7: The results for Example 5.5.



(a) Signal S_1 (b) Residuals

Figure 8: The results for Example 5.6.



(a) Signal S_2 (b) Residuals

Figure 9: The results for Example 5.6.

Signal	Error of CG	Error of s-CG	CPU time of CG	CPU time of s-CG
S_1	1.6310e-08	1.8796e-13	0.020711	0.015870
S_2	3.1441e-10	5.1442e-15	0.020035	0.015378

Table 7: CPU time and errors for CG and variable s-step CG methods to Example 5.6.

6. Conclusions

Based on the findings from the preceding sections, it can be concluded that by selecting the variable parameter s, we not only improve parallelism but also maintain the convergence of the method. While it was previously believed that s-step algorithms would, at best, achieve convergence comparable to the original algorithm, in some cases, they can even improve it. With the appropriate selection of this parameter, convergence can be achieved using the same number of floating-point operations and less time (especially when parallelized). The variable s-step CG method proves to be an effective and efficient approach for various problems, including CG for systems with both single and multiple right-hand sides. This method helps reduce algorithmic communication while preserving convergence. Additionally, obtaining the s parameter for each step using algorithmic parameters simplifies the method's analysis and paves the way for future improvements. Although the proposed regularization method may increase the computational load and runtime per iteration, it can be particularly effective when the initial data contains noise. We recommend using the s-step CG method for well-posed problems and the regularized s-step CG method for ill-posed ones.

Acknowledgments

We would like to express our gratitude to the editor and referee for carefully reading the manuscript and for the constructive comments.

Conflicts of interest

The authors have no conflict of interest to declare.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

[1] National Institute of Standards and Technology: Matrix Market, https: http://math.nist.gov/MatrixMarket/.
[2] Auwal Bala Abubakar, Poom Kumam, Jinkui Liu, Hassan Mohammad, and Christiane Tammer. New three-term conjugate gradient algorithm for solving monotone nonlinear equations and signal recovery problems. *Int. J. Comput. Math.*, 100(10):1992–2013, 2023.
[3] Zhaojun Bai, Dan Hu, and Lothar Reichel. A Newton basis GMRES implementation. *IMA J. Numer. Anal.*, 14(4):563–581, 1994.
[4] Zhong-Zhi Bai. Motivations and realizations of Krylov subspace methods for large sparse linear systems. *J. Comput. Appl. Math.*, 283:71–78, 2015.
[5] Grey Ballard, James Demmel, Olga Holtz, and Oded Schwartz. Minimizing communication in numerical linear algebra. *SIAM J. Matrix Anal. Appl.*, 32(3):866–901, 2011.
[6] Michele Benzi. Preconditioning techniques for large linear systems: a survey. *Comput. Phys.*, 182(2):418–477, 2002.
[7] Amit Bhaya, Pierre-Alexandre Bliman, Guilherme Niedu, and Fernando Pazos. A cooperative conjugate gradient method for linear systems permitting multithread implementation of low complexity. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 638–643. IEEE, 2012.
[8] Xinyuan Cao, Mingsheng Chen, Qi Qi, Meng Kong, Jinhua Hu, Liang Zhang, and Xianliang Wu. Solving electromagnetic scattering problems by underdetermined equations and Krylov subspace. *IEEE Microwave and Wireless Components Letters*, 30(6):541–544, 2020.
[9] Erin Carson, Nicholas Knight, and James Demmel. Avoiding communication in two-sided Krylov subspace methods. Technical report, California Univ Berkeley Dept of Electrical Engineering And Computer Science, 2011.
[10] Erin C Carson. The adaptive s-step conjugate gradient method. *SIAM J. Matrix Anal. Appl.*, 39(3):1318–1338, 2018.

[11] Erin Claire Carson. An adaptive s-step conjugate gradient algorithm with dynamic basis updating. *Appl. Math.*, 65(2):123–151, 2020.

[12] Xu Chen, Xin-Xin Gong, Youfa Sun, and Siu-Long Lei. A Preconditioned Policy Krylov Subspace Method for Fractional Partial Integro-Differential HJB Equations in Finance. *Fractal and Fractional*, 8(6):316, 2024.

[13] Anthony T Chronopoulos and Charles William Gear. S-step iterative methods for symmetric linear systems. *J. Comput. Appl. Math.*, 25(2):153–168, 1989.

[14] Anthony T Chronopoulos and Andrey B Kucherov. Block s-step Krylov iterative methods. *Numer. Linear Algebra Appl.*, 17(1):3–15, 2010.

[15] Anthony T. Chronopoulos and Charles D. Swanson. Parallel iterative s-step methods for unsymmetric linear systems. *Parallel Comput.*, 22(5):623–641, 1996.

[16] James Demmel, Laura Grigori, Mark Hoemmen, and Julien Langou. Communication-optimal parallel and sequential QR and LU factorizations. *SIAM J. Sci. Comput.*, 34(1):A206–A239, 2012.

[17] James Demmel, Mark Frederick Hoemmen, Marghoob Mohiyuddin, Katherine A Yelick, Mark Hoemmen, and Katherine Yelick. Avoiding communication in computing Krylov subspaces. 2007.

[18] Laura Grigori and Sophie Moufawad. Communication avoiding ILU0 preconditioner. *SIAM J. Sci. Comput.*, 37(2):C217–C246, 2015.

[19] AC Hindmarsh and HF Walker. Note on a Householder implementation of the GMRES method. Technical report, Lawrence Livermore National Lab., CA (USA); Utah State Univ., Logan (USA . . . , 1986.

[20] Mark Hoemmen. *Communication-avoiding Krylov subspace methods*. University of California, Berkeley, 2010.

[21] Soichiro Ikuno, Gong Chen, Taku Itoh, Susumu Nakata, and Kuniyoshi Abe. Variable preconditioned Krylov subspace method with communication avoiding technique for electromagnetic analysis. *IEEE Transactions on Magnetics*, 53(6):1–4, 2017.

[22] David Imberti and Jocelyne Erhel. Varying the s in your s-step GMRES. *Electron. Trans. Numer. Anal.*, 47:206–230, 2017.

[23] Wayne D Joubert and Graham F Carey. Parallelizable restarted iterative methods for nonsymmetric linear systems. Part I: Theory. *Int. J. Comput. Math.*, 44(1-4):243–267, 1992.

[24] Hojjatollah Shokri Kaveh, Masoud Hajarian, and Anthony T Chronopoulos. Developing variable s-step CGNE and CGNR algorithms for non-symmetric linear systems. *J. Franklin Inst.*, page 107071, 2024.

[25] Hojjatollah Shokri Kaveh, Masoud Hajarian, and Anthony T Chronopoulos. Efficient image reconstruction via regularized variable s-step conjugate gradient method for Sylvester matrix equations. *J. Franklin Inst.*, 362(6):107634, 2025.

[26] Hojjatollah Shokri Kaveh, Masoud Hajarian, and Anthony T Chronopoulos. Variable s-step generalized semi-conjugate gradient algorithm for solving nonsymmetric linear systems arising in signal recovery. *J. Franklin Inst.*, page 107870, 2025.

[27] WeiJian Liu, WenChong Xie, RongFeng Li, ZeTao Wang, and YongLiang Wang. Adaptive detectors in the Krylov subspace. *Sci. China Inf. Sci.*, 57(10):1–11, 2014.

[28] Gérard Meurant and J Duintjer Tebbens. Krylov methods for nonsymmetric linear systems. *Cham: Springer*, 2020.

[29] Marghoob Mohiyuddin, Mark Hoemmen, James Demmel, and Katherine Yelick. Minimizing communication in sparse matrix solvers. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, pages 1–12. IEEE, 2009.

[30] Sophie Moufawad. *Enlarged Krylov Subspace Methods and Preconditioners for Avoiding Communication*. PhD thesis, Université Pierre et Marie Curie-Paris VI, 2014.

[31] Sophie Moufawad. Numerical Stability of s-step Enlarged Krylov Subspace Conjugate Gradient Methods. *arXiv preprint arXiv:1804.10629*, 2018.

[32] Ahmed Anwer Mustafa. New spectral LS conjugate gradient method for nonlinear unconstrained optimization. *Int. J. Comput. Math.*, 100(4):838–846, 2023.

[33] Maxim Naumov. S-step and communication-avoiding iterative methods. Technical report, Technical Report NVR-2016-003, NVIDIA, 2016.

[34] Julie Nutini, Issam Laradji, and Mark Schmidt. Let's Make Block Coordinate Descent Converge Faster: Faster Greedy Rules, Message-Passing, Active-Set Complexity, and Superlinear Convergence. *J. Mach. Learn. Res.*, 23(131):1–74, 2022.

[35] John W Pearson and Jennifer Pestana. Preconditioners for Krylov subspace methods: An overview. *GAMM-Mitt.*, 43(4):e202000015, 2020.

[36] Yousef Saad. *Iterative Methods For Sparse Linear Systems*. SIAM, 2003.

[37] Jiancheng Shi and Jianhua Liu. Study on performance of Krylov subspace methods at solving large-scale contact problems. *The International Journal of Advanced Manufacturing Technology*, 84(1):435–444, 2016.

[38] Hojjatollah Shokri Kaveh and Hojjatollah Adibi. Mapped regularization methods for the Cauchy problem of the Helmholtz and Laplace equations. *Iranian Journal of Science and Technology, Transactions A: Science*, 45(2):669–682, 2021.

[39] Hojjatollah Shokri Kaveh, Masoud Hajarian, Anthony Chronopoulos, et al. Finding solution of linear systems via new forms of BiCG, BiCGstab and CGS algorithms. *Radon Ser. Comput. Appl. Math.*, 43(6):1–24, 2024.

[40] Hojjatollah Shokri Kaveh, Masoud Hajarian, and Anthony T Chronopoulos. Variable s-step technique for new conjugate residual algorithms for Solving non-square linear systems arising in control problems. *Numerical Mathematics: Theory, Methods and Applications*, 18(2):1–21, 2025.

[41] Hojjatollah Shokri Kaveh, Masoud Hajarian, and Anthony T Chronopoulos. Variable s-step technique for Planar algorithms in solving indefinite linear systems. *J. Comput. Appl. Math.*, 44(6):306, 2025.

[42] Hojjatollah Shokri Kaveh, Masoud Hajarian, Anthony T Chronopoulos, Dimitar Slavchev, and Zahari Zlatev. Regularized variable s-step CG and CR algorithms to solve shifted linear systems. *Engineering Computations*, 42(6):1–21, 2025.

[43] Zhongbo Sun, Hongyang Li, Jing Wang, and Yantao Tian. Two modified spectral conjugate gradient methods and their global convergence for unconstrained optimization. *Int. J. Comput. Math.*, 95(10):2082–2099, 2018.

[44] Andreĭ Nikolaevich Tikhonov, Arsenin, et al. *Solutions Of Ill-posed Problems*. Winston, 1977.

[45] J. van Rosendale. Minimizing inner product data dependence in conjugate gradient iteration. *No. NASA-CR-172178*, 1983.

[46] C Vuik. Krylov subspace solvers and preconditioners. *ESAIM Proc. Surveys*, 63:1–43, 2018.
[47] Homer F Walker. Implementation of the GMRES method using Householder transformations. *SIAM Journal on Scientific and Statistical Computing*, 9(1):152–163, 1988.
[48] Andrew J Wathen. Preconditioning. *Acta Numer.*, 24:329–376, 2015.
[49] Lifeng Yang, Meng Fan, Xin Wang, Zhe Liu, Ning Li, Wei Yu, and Yu Wang. Efficiency Comparison of Direct and Krylov Subspace Iterative Parallel Solvers for Hydraulic Fracture Propagation Simulations in Shale Reservoirs. In *ARMA US Rock Mechanics/Geomechanics Symposium*, page D031S025R002. ARMA, 2025.
[50] Min Yang, Xinqiang Xu, Wanyin Wang, Dongming Zhao, and Wei Zhou. 3D gravity fast inversion based on Krylov subspace methods. *Journal of Geophysics and Engineering*, 21(1):29–46, 2024.
[51] Gonglin Yuan, Bopeng Wang, and Zhou Sheng. The Hager–Zhang conjugate gradient algorithm for large-scale nonlinear equations. *Int. J. Comput. Math.*, 96(8):1533–1547, 2019.
[52] Hossein Zare and Masoud Hajarian. Determination of regularization parameter via solving a multi-objective optimization problem. *Appl. Numer. Math.*, 156:542–554, 2020.