ISSN: 1392-6292/ eISSN: 1648-3510

VILNIUS TECH
Vilnius Gediminas Technical University

MATHEMATICAL MODELLING and ANALYSIS

2026
Volume 31
Issue 1
Pages 79–95
https://doi.org/10.3846/mma.2026.23695

Variable s-step CGNR method for solving the matrix equation $AXB = C$ arising in image processing

Hojjatollah Shokri Kaveh and Masoud Hajarian ✉

Department of Applied Mathematics, Faculty of Mathematical Sciences, Shahid Beheshti University, Tehran, Iran

Article History:
- received April 10, 2025
- revised August 2, 2025
- accepted September 23, 2025

Abstract. The matrix equation $AXB = C$ is widely utilized in signal and image processing. In this paper, we present a variable s-step algorithm based on the CGNR method for solving this matrix equation by employing normalization techniques. This algorithm is subsequently enhanced through the application of s-step and regularization methods. By varying the number of basic matrices involved (denoted as s), both the accuracy and speed of the algorithm are improved. The proposed algorithm effectively computes solutions to the matrix equation, demonstrating superior performance when the problem matrices are symmetric. Finally, we investigate the performance and efficacy of these techniques through several numerical examples.

Keywords: matrix equation; CGNR method; variable s-step algorithm; regularization method.

AMS Subject Classification: 65Fxx; 65F22.

✉ Corresponding author. E-mail: m_hajarian@sbu.ac.ir; masoudhajarian@gmail.com

1 Introduction

The matrix equation $AXB = C$ is an important equation in mathematics and engineering, with applications in various fields including linear algebra, control systems, system identification, and optimization. In this matrix equation

- A and B are constant matrices that typically belong to specific dimensions.
- X can be considered as the variable matrix that needs to be determined.
- C is the resulting matrix.

Applications of the matrix equation $AXB = C$:

- Control systems: In linear control, the matrix equation $AXB = C$ can be used for designing controllers and filters. Also the matrix equation is often employed to derive the matrices needed for the controllers [26].

- System identification: This matrix equation is utilized in system identification techniques, where we aim to estimate the parameters of a dynamic system based on its inputs and outputs.

- Optimization: The matrix equation $AXB = C$ is applied in optimization problems that require decision-making in a matrix format.

- Image and signal processing: In signal processing, this matrix equation can be used in the context of filtering images or signals [22]. In image processing, image X undergoes transformations A and B from both sides, resulting in the observed image C. The objective is to recover the original source image X. Multiplying these two operators from both sides may result in the image becoming blurred, disturbed, or both. Eliminating the effects of these two operators by finding their inverses can be very costly, so iterative algorithms are recommended. Additionally, techniques such as regularization may be necessary during the image reconstruction process to stabilize the algorithm. Iterative algorithms are employed to reconstruct the source image.

Given the volume of data in science and engineering, it is preferable to use algorithms with lower computational and communication costs to solve the matrix equations. It is more economical to solve matrix equations using iterative methods [8]. For example, using the acceleration technique, a method for solving the generalized matrix equations was introduced that incorporates the concept of momentum [18]. In [17], an iterative algorithm based on generalized displacement partitions was developed for solving coupled Sylvester matrix equations using a hierarchical identification approach. An Acceleration Over-Resolved (AOR) method for the complex or real generalized Lyapunov matrix equation is introduced in [6]. An advanced parameterless gradient descent method, utilizing the accelerated technique associated with momentum methods, was employed to solve the coupled matrix equations arising from dynamical systems [19].

Krylov subspace methods (KSMs) play a crucial role in the iterative solution of linear systems and matrix equations and are frequently employed for solving them [20]. Some notable techniques within this category include the conjugate gradient (CG), conjugate residual (CR), bi-conjugate gradient (BiCG), stabilized bi-conjugate gradient (BiCGStab), conjugate gradient square (CGS), bi-conjugate residual (BiCR), and generalized minimal residual (GMRES) methods [1, 13, 16].

To enhance the performance of KSMs, the s-step methods were introduced for symmetric and non-symmetric linear systems [14]. Utilizing the s-step approach can considerably reduce the number of iterations needed for an algorithm. While the original algorithm may require M iterations, the s-step variant can achieve results in only M/s iterations. Adjusting the parameter s allows for fine-tuning the total iteration count and minimizes data transfer, though excessively increasing s is impractical. A significant challenge in implementing s-step algorithms is selecting the most suitable value for s. A promising method involves dynamically modifying s based on the algorithm's

parameters, which can lead to even fewer iterations during execution [5, 15]. Recent innovations have focused on enhancing the numerical conditioning of the s-step methods by estimating Ritz values as the system matrix's eigenvalues and utilizing Newton and Chebyshev polynomials [3]. Improvements in precision and replacement of residuals have shown to stabilize the s-step CG approach in [25]. Moreover, employing Newton's polynomials to determine the optimal s in the s-step GMRES method has mitigated instability issues [24]. To address numerical instability arising from the s-step approach, an increase in Lanczos orthogonalization computations was employed in [2]. Additionally, s-step adaptations of Orthomin(k) and GMRES algorithms were proposed to better data locality in [4].

To minimize potential instability from the s-step strategy, it is advisable to employ regularization techniques analogous to those found in linear regularization frameworks. These methods aim to reshape the distribution of singular values to diminish the influence of inaccuracies in the matrix. Various approaches are available, including filtering singular values, truncating the singular value distribution, and applying iterative methods to achieve the desired distribution. The Tikhonov method was the first regularization technique proposed, effectively reducing noise in ill-posed problems by modifying the problem matrix to decrease its condition number. Beyond the Tikhonov method, other regularization techniques have been established, including truncated singular value decomposition (TSVD), the L-curve method, and mapped regularization techniques [20, 21]. A key aspect of these regularization methods is identifying the regularization parameter that minimizes the Tikhonov function [23].
Recently, several KSMs were enhanced with s-step and regularization techniques. In [10], new algorithms utilizing these techniques were presented for nonsymmetric linear systems. An s-step algorithm, based on a generalization of the semi-conjugate gradient algorithm, was presented for solving nonsymmetric linear systems [12]. Additionally, the CGNR algorithm was proposed for addressing the Sylvester matrix equations arising in image processing problems [11].

In terms of the innovation presented in this paper, it is important to highlight that the work begins with an expansion of the CGNR algorithm to effectively solve the matrix equation $AXB = C$. Following this initial step, we implement the variable s-step technique on the newly derived algorithm. This adjustment allows for a significant reduction in the number of iterations required to reach a solution by manipulating the s parameter. Additionally, the variable s-step algorithm contributes to a decrease in communication overhead within the process, which is particularly beneficial for improving efficiency. However, as the s parameter increases, there is a risk of the algorithm becoming unstable. To mitigate this risk, the authors incorporate a regularization technique. This ensures that the algorithm maintains a stable convergence rate, ideally matching or exceeding the speed of the original algorithm, all while achieving this with a reduced number of iterations. This comprehensive approach not only enhances the performance of the algorithm but also emphasizes its practical applicability in solving matrix equations effectively.

The outline is as follows. Section 2 introduces the variable s-step CGNR

algorithm for solving the matrix equation $AXB = C$. In Section 3, several necessary techniques for determining the parameters of the algorithm are discussed. In Section 4, we state and prove the convergence results. Finally, Section 5 tests the algorithm by solving various numerical examples.

2 Variable s-step CGNR algorithm for matrix equations

Here, our goal is to extend the CGNR algorithm [7] to solve the matrix equation $AXB = C$. In numerous image and signal processing applications, after applying transformations to the original image, a distorted version of the image is obtained. The objective in such cases is to reconstruct the original image from the distorted one. To achieve this, it is necessary to apply inverse transformations to the distorted image in order to recover the original content. However, computing the inverse of a transformation (if it exists) can be a complex and resource-intensive task, making iterative algorithms a preferred choice. Here, two transformations, denoted as A and B, are applied to an image X, resulting in a distorted image C. This scenario can be represented by the matrix equation $C = AXB$. In this paper, the process of reconstructing the original image involves finding a way to invert these transformations (A and B) effectively, typically through iterative methods that allow for continued refinement until the desired accuracy is achieved. We consider the matrix equation

$$AXB = C, \tag{2.1}$$

where $A \in \mathbb{R}^{m\times m}$, $B \in \mathbb{R}^{n\times n}$, and $C \in \mathbb{R}^{m\times n}$ are known, and $X \in \mathbb{R}^{m\times n}$ is unknown.

Algorithm 1. **CGNR algorithm for solving the linear system $\boldsymbol{Ax = b}$**
Choose x_0
$r_0 = b - Ax_0, \quad z_0 = A^T r_0, \quad p_0 = r_0$
For i=0 Until Convergence Do
$w_i = Ap_i, \quad \alpha_i = \frac{p_i^T r_i}{\|w_i\|_2^2}$
$x_{i+1} = x_i + \alpha_i p_i, \quad r_{i+1} = r_i - \alpha_i w_i$
$z_{i+1} = A^T r_{i+1}, \quad \beta_i = \frac{\|z_{i+1}\|_2^2}{\|z_i\|_2^2}$
$p_{i+1} = z_{i+1} + \beta_i p_i, \quad i = i + 1$
End For.

To obtain our algorithm for solving the matrix equation (2.1), we apply the same approach used in the CG algorithm. This is done following the initialization outlined below

$$P_0 = R_0 = C - AX_0B.$$

In the final projection into the approximation space, we utilize the concept employed in the CGNR algorithm. In i-th iteration, we set

$$Q_i = AP_{i-1}B.$$

To calculate the parameter α of the CGNR algorithm (Algorithm 1), which is presented in matrix form here, the following system should be solved

$$(Q_i^T Q_i)\alpha_i = (Q_i^T R_{i-1}).$$

In solving the above system, the norm of the solution may increase due to the ill-conditioning of the matrices, therefore, in this step, a suitable regularization filter $\gamma_i(1, \alpha_i)$ is applied to reduce it. Then the approximation and the residual are updated as follows

$$X_i = X_{i-1} + \gamma_i(1, \alpha_i)\alpha_i Q_i, \quad R_i = C - AX_iB,$$

where $\gamma_i(1, \alpha_i)$ is the regularization filter which will be discussed in Subsection 3.2. Similarly to the CG method, the parameter β and the new basis of P are calculated

$$\beta_i = \frac{\|R_i\|_2}{\|R_{i-1}\|_2}, \quad P_i = R_i + \beta_i P_{i-1}.$$

Our algorithm is an algorithm between CG and CGNR algorithms. In some cases it follows the CG algorithm and in some cases it is similar to the CGNR algorithm. The CGNR method for solving the matrix equation (2.1) is given in Algorithm 2.

Algorithm 2. **CGNR algorithm for the matrix equation (2.1)**
Choose X_0
$R_0 = C - AX_0B, \quad P_0 = R_0$
For i=1 Until Convergence Do
$Q_i = AP_{i-1}B$
Solve $(Q_i^T Q_i)\alpha_i = (Q_i^T R_{i-1})$
Calculate $\gamma_i(1, \alpha_i)$ from (3.2)–(3.6)
$X_i = X_{i-1} + \gamma_i(1, \alpha_i)Q_i\alpha_i, \quad R_i = C - AX_iB$
$\beta_i = \frac{\|R_i\|_2}{\|R_{i-1}\|_2}, \quad P_i = R_i + \beta_i P_{i-1}$
$i = i + 1$
End For.

To rewrite Algorithm 2 in s-step mode, noting that the number of projected bases (matrix Q_i) should be increased, the matrix $\tilde{Q}_i$ after choosing s_i is defined as follows $\tilde{Q}_i = [Q_i, WQ_iH, ..., W^{s_i-1}Q_iH^{s_i-1}]$, where $W = AA^T$ and $H = B^TB$. Since these matrices help to make the bases symmetric, this method can be considered the incomplete symmetrization of the bases.

On the other hand, because each time to calculate the matrices placed in $\tilde{Q}_i$, the transpose matrix of sides matrices are multiplied by them and a normalization occurs on the matrix equation, this method can be considered similar to CGNR, which normalizes and solves the linear system by multiplying by the transpose matrix. This algorithm is similar to Algorithm 2 and only the matrix of subspaces of $\tilde{Q}_i$ is used instead of the Q_i matrix, although it should be kept in mind that a different regularization filter should be chosen

for this algorithm and the previous algorithm. According to the change of s_i in this algorithm, the filters should be dependent on s_i in such a way that with the increase of s_i, the norm of the matrices will decrease even more, otherwise the error will accumulate and the algorithm will lose its stability and convergence. The variable s-step CGNR algorithm for finding the solution of the matrix equation (2.1) is given in Algorithm 3. Some parameters such as s_i and regularization introduced in the algorithms remain unset. In the next section, we will examine and specify these parameters.

Algorithm 3. **Variable s-step CGNR (VsCGNR) algorithm for the matrix equation (2.1)**
Choose X_0
$R_0 = C - AX_0B, \quad P_0 = R_0$
$W = AA^T, \quad H = B^TB$
For $i = 1$ Until Convergence Do
$Q_i = AP_{i-1}B$
$\tilde{Q}_i = [Q_i, WQ_iH, ..., W^{s_i-1}Q_iH^{s_i-1}]$
Solve $(\tilde{Q}_i^T\tilde{Q}_i)\alpha_i = (\tilde{Q}_i^TR_{i-1})$
Calculate $\gamma_i(s_i, \alpha_i)$ from (3.2)–(3.6)
$X_i = X_{i-1} + \gamma_i(s_i, \alpha_i)\tilde{Q}_i\alpha_i$
$R_i = C - AX_iB, \quad \beta_i = \frac{\|R_i\|_2}{\|R_{i-1}\|_2}$
$P_i = R_i + \beta_iP_{i-1}$
Choose s_i
$i = i + 1$
End For.

3 Techniques for balancing runtime and stability

To improve the efficiency of our algorithms, increasing the value of s can boost their performance. However, raising s too quickly may lead to instability in the algorithm. The next subsection outlines two approaches for selecting the parameter s and applying regularization to find a balanced trade-off between stability and performance speed-up.

3.1 Selection of s

Determining the parameter s is a crucial aspect when working with s-step algorithms. Selecting the right value for s can greatly influence the algorithm's efficiency, and an inappropriate choice may lead to divergence. Several factors, including system size, condition number, and the iterative method being used, affect the selection of this parameter. Currently, there is no single best value of s that applies to all linear systems.

Choosing a larger value of s can shorten processing time but may threaten the stability of the algorithm, increasing the risk of divergence. On the other hand, opting for a smaller s can improve stability but may prolong runtime. One possible solution to this dilemma is to adjust s dynamically during various iterations of the algorithm, allowing it to increase as the residual norm

decreases. The challenge here is to find an effective growth function that can set s appropriately throughout the iterations.

Like the choice of a fixed s, using a rapidly increasing growth function can reduce runtime further but might lead to instability. Conversely, a slowly growing function can help maintain stability but may not optimize the time efficiency as much. To address these issues, regularization techniques are applied. The next section introduces a function to determine s_i (the value of s in the i-th iteration) as follows:

$$s_1 = 1; \quad s_i = 1 + \log\left\lfloor \left(\sum_{j=1}^{i-1} s_j\right)^{0.5} \right\rfloor, \quad \forall i > 1. \tag{3.1}$$

Integrating this function with the mapped regularization technique discussed in the upcoming subsection can yield positive results [10].

3.2 Regularization filters

A technique employed to enhance the stability of algorithms is regularization, which entails fine-tuning the parameters to improve stability and decrease error sensitivity. Among the well-known regularization filters are the Tikhonov and TSVD filters [21]. A more advanced subset of these filters is known as mapped filters, which improve algorithm stability by projecting the parameter of interest into an appropriate space. Below are several filters specifically designed for the variable s-step algorithm based on the s parameter:

- Exponential filter:
$$\gamma(s_i, \alpha_i) = 1 - e^{-\frac{s_i}{\|\alpha_i\|_2}}. \tag{3.2}$$
- Hyperbolic filter:
$$\gamma(s_i, \alpha_i) = \tanh\left(\frac{s_i}{\|\alpha_i\|_2}\right). \tag{3.3}$$
- Tikhonov filter:
$$\gamma(s_i, \alpha_i) = \frac{s_i^2\|\alpha_i\|_2}{1 + s_i^2\|\alpha_i\|_2^2}. \tag{3.4}$$
- Logarithmic filter:
$$\gamma(s_i, \alpha_i) = \ln\left(\left(\sqrt{\frac{s_i^2}{\|\alpha_i\|_2^2} + 1} + \frac{s_i^2}{\|\alpha_i\|_2^2}\right)/\sinh(\frac{s_i^2}{\|\alpha_i\|_2})\right). \tag{3.5}$$
- Sign filter:
$$\gamma(s_i, \alpha_i) = \frac{s_i}{2\|\alpha_i\|_2}\left(1 + \operatorname{sign}\left(\|\alpha_i\|_2 - \frac{s_i}{\|\alpha_i\|_2}\right)\right). \tag{3.6}$$

The above filters are examples of capped filters. Furthermore, it is possible to create additional filters using various functions, many of which are based on the Tikhonov filter and produce similar results. The filters mentioned function similarly to vector consolidation methods in schemes for solving linear systems. Given that s-step algorithms are often represented in block form, the process of combining matrix columns can be computationally demanding. The mapped regularization technique provides an effective solution by facilitating algorithm stability through straightforward multiplication with a factor within the corresponding matrix. In the next section, we will explore features related to the orthogonalization of vectors and matrices in the framework of linear systems and their associated matrix equations.

4 Convergence results

In this section, some theorems related to the convergence and properties of the CGNR algorithm for the matrix equation (2.1) are presented. These theorems for the CGNR algorithm applied to linear systems have already been proven, and here we will simply generalize them to the specified matrix equation.

Theorem 1. *Let X^* be the exact solution of the matrix equation* (2.1)*, and $\{Q_i\}_{i=1}^{mn}$ is an orthogonal basis for $\mathbb{R}^{m\times n}$, then*

$$X^* = \sum_{i=1}^{mn} \alpha_i Q_i.$$

Proof. This theorem is one of the well-known theorems in the field of linear algebra and has been proven in [9]. □

Theorem 2. *Let X^* be the exact solution of the matrix equation* (2.1) *and X_j is the approximation obtained from Algorithm 2 in the j-th iteration, then*

$$\|X^* - X_{j+1}\|_2 \leq \|X^* - X_j\|_2.$$

Proof. According to Theorem 1, and the fact that $X_j = \sum_{i=1}^{j} \alpha_i Q_i$ the following relations are established

$$X^* - X_{j+1} = \sum_{i=j+2}^{mn} \alpha_i Q_i, \quad X^* - X_j = \sum_{i=j+1}^{mn} \alpha_i Q_i.$$

Due to the orthogonality of Q_i's, we have

$$\begin{aligned} \|X^* - X_j\|_2^2 = \| \sum_{i=j+1}^{mn} \alpha_i Q_i\|_2^2 = \|\alpha_{j+1} Q_{j+1}\|_2^2 + \| \sum_{i=j+2}^{mn} \alpha_i Q_i\|_2^2 \\ \geq \| \sum_{i=j+2}^{mn} \alpha_i Q_i\|_2^2 = \|X^* - X_{j+1}\|_2^2. \end{aligned}$$

□

In these theorems, It is clear that α_i can be transferred to the right side of Q_i as $\alpha_i I$, where I is the $n \times n$ identity matrix.

Theorem 3. *If the conditions of Theorem 2 are satisfied and R_j is the residual obtained from Algorithm 2 in the j-th iteration, then*

$$\|R_{j+1}\|_2 \leq \|R_j\|_2.$$

Proof. According to Theorems 1 and 2, we have

$$R_{j+1} = C - AX_{j+1}B = A(X^* - X_{j+1})B = A(\sum_{i=j+2}^{mn} \alpha_i Q_i)B,$$

$$R_j = C - AX_jB = A(X^* - X_j)B = A(\sum_{i=j+1}^{mn} \alpha_i Q_i)B.$$

Since the Q_i's are orthogonal, we can obtain

$$\|R_j\|_2^2 = \|A(X^* - X_j)\|_2^2 = \|A(\sum_{i=j+1}^{mn} \alpha_i Q_i)B\|_2^2$$

$$= \|A(\alpha_{j+1}Q_{j+1})B\|_2^2 + \|A(\sum_{i=j+2}^{mn} \alpha_i Q_i)B\|_2^2$$

$$\geq \|A(\sum_{i=j+2}^{mn} \alpha_i Q_i)B\|_2^2 = \|A(X^* - X_{j+1})B\|_2^2 = \|R_{j+1}\|_2^2.$$

□

Remark 1. Theorems 2 and 3 shows that the sequences of the norm of errors and residuals are monotonically decreasing. The properties emphasize that the CGNR algorithm possesses fast and converges smoothly.

Similar theorems regarding the variable s-step CGNR algorithm for the matrix equation (2.1) can be proven, in which only the number of iterations is reduced while consuming the same number of mn orthogonal matrices Q_i, ensuring that the algorithm still converges. The theorems in this section are stated for the case where there are no precision errors, and the numerical results may differ in this context. In the next section, we will numerically compare the algorithms described in the previous sections.

5 Numerical results

In this section, a random matrix is initialized as the starting point ($X_0 = \text{eye}(m, n)$), while the right-hand side vector is defined as

$$C_{m\times n} = A_{m\times m} \times 1_{m\times n} + 1_{m\times n} \times B_{n\times n},$$

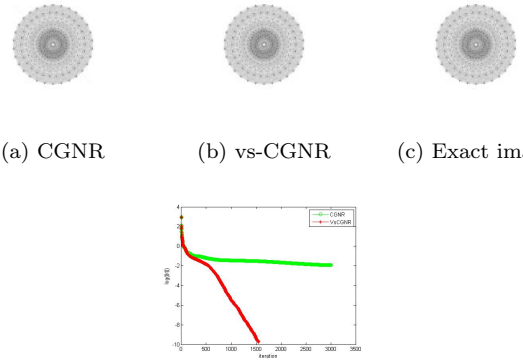
where $1_{m\times n}$ is a matrix with all its elements one. Also the value of the parameter s is determined by (3.1). In the illustrations provided, the algorithm is

set to execute for a total of 3000 iterations, and in the s-step mode, 3000 bases are also utilized. The termination criterion for the examples, unless specified otherwise, is defined by $\|Residual\| \leq 10^{-12}$. The matrices employed in this section are obtained from a variety of problems, all of which are accessible through MATLAB Help. The computational experiments are carried out on a computer equipped with an Intel(R) Celeron CPU N2830 @ 2.16 GHz and 4 GB of memory.

Example 1. In the first example, a symmetrical image is reconstructed using the proposed algorithm. The matrices

$$A = gallery('kms', 200, 0.5), \quad B = gallery('kms', 200, 0.6),$$

are utilized for the matrix equation (2.1). The number of iterations is $K = 3000$ and the Tikhonov filter was used for regularization. The exact solution to the problem is the 200×200 image in Figure 1. As shown in Figure 1 and Table 1, the application of regularization techniques results in improved numerical outcomes. Furthermore, Table 2 clearly demonstrates that the variable s-step positively affects the runtime of the algorithm.



(a) CGNR (b) vs-CGNR (c) Exact image

(d) residual norms

Figure 1. Reconstructed images and residual norms in Example 1. (a) shows the reconstructed figure using the original algorithm, (b) displays the reconstructed figure from the proposed algorithm, (c) presents the original figure, and (d) illustrates the residual norm obtained from the algorithms over different iterations.

Table 1. Residual and error norms for numerical Example 1. In this table, K represents the number of bases used in the algorithms, E denotes the error norm, and R refers to the residual norm obtained from the algorithm.

K	E(Reg-CG)	R(Reg-CG)	E(Reg-vsCG)	R(Reg-vsCG)
1000	0.3768	0.0351	0.1118	0.0096
3000	0.1351	0.0116	2.1096e-09	1.7612e-10

Table 2. Runtime for Example 1.

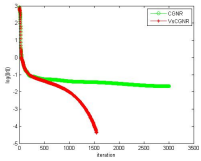
iterations	Runtime(Reg-CGNR)	Runtime(Reg-vsCGNR)
1000	19.665874	16.453234
3000	54.388632	44.651685

Example 2. In this example, we replicated Example 1 using an nonsymmetric image and presented the results in Figure 2, as well as in Tables 3 and 4.

(a) CGNR

(b) vs-CGNR

(c) Exact image



(d) residual norms

Figure 2. Reconstructed images and residual norms in Example 2. (a) shows the reconstructed figure using the original algorithm, (b) displays the reconstructed figure from the proposed algorithm, (c) presents the original figure, and (d) illustrates the residual norm obtained from the algorithms over different iterations.

Table 3. Residual and error norms for Example 2. In this table, K represents the number of bases used in the algorithms, E denotes the error norm, and R refers to the residual norm obtained from the algorithm.

K	E(Reg-CG)	R(Reg-CG)	E(Reg-vsCG)	R(Reg-vsCG)
1000	0.4308	0.0412	0.4036	0.0380
3000	0.2420	0.0211	5.0453e-04	4.2089e-05

As shown in the figures and tables of Examples 1 and 2, the presence or absence of symmetry does not affect the performance of the proposed algorithm. The residual norm consistently decreases more rapidly compared to the original algorithm, while the runtime decreases with a given number of bases.

Table 4. Runtime for Example 2.

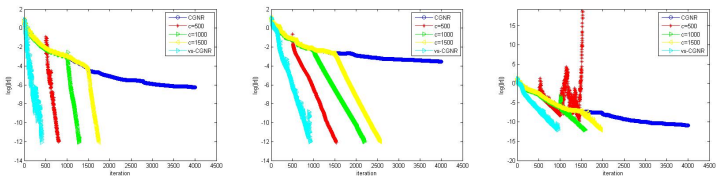
iterations	Runtime(Reg-CGNR)	Runtime(Reg-vsCGNR)
1000	103.910521	101.190861
3000	279.734534	243.524095

Example 3. In this example, we discuss the changes in the parameter s across different iterations. A general function is defined by c, and the parameter s changes as c varies where

$$s_0 = 1, \forall i \geq 1, s_i = 1 + \left\lfloor \sum_{j=1}^{i-1} s_j / c \right\rfloor, \tag{5.1}$$

$$A = gallery('tridiag', m, -1, 2, -1), \quad B = gallery('tridiag', n, 0.5, 2, 0.5).$$

The results are verified and presented in Figure 3 and Table 5.



(a) n=4, m=6 (b) n=5, m=10 (c) n=7, m=10

Figure 3. Residual norm in Example 3.

Table 5. Runtime for Example 3. In this table, the runtime of the original algorithm and the proposed algorithm is compared with three algorithms that utilize different s functions for various values of c in (5.1).

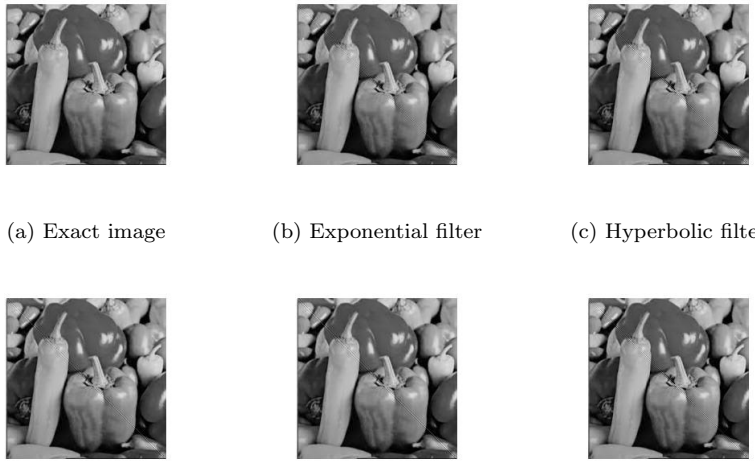
Dimentions	T(CGNR)	T(c=500)	T(c=1000)	T(c=1500)	T(vs-CGNR)
m=6, n=4	1.361646	0.464819	0.560035	0.708186	0.307103
m=10, n=5	1.556126	0.865437	1.000464	1.069885	0.824668
m=10, n=7	1.659863	1.364891	0.999417	1.100701	0.992463

The results of this example indicate that appropriate adjustments to the parameter s are particularly important throughout the algorithm. The proposed function for s exhibits better performance in terms of speed and accuracy compared to other functions. Figure 3 shows that changing the dimensions of the A, B does not affect the performance of the proposed algorithm.

Example 4. In this example, we evaluate and compare the proposed algorithm using various regularization filters. The matrices A and B are considered as follows

$$A = gallery('kms', m, 0.3), \quad B = gallery('kms', n, 0.2).$$

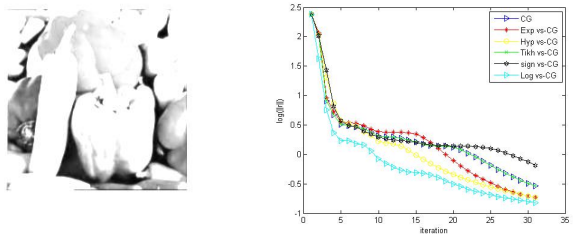
The exact solution to the problem is presented in Figure 4, while the matrix on the right side (C) is shown in Figure 5. The results for iteration number $K = 30$ are shown in Figures 4. The figures in this example indicate that the regularization filters exhibit similar performance.



(a) Exact image (b) Exponential filter (c) Hyperbolic filter

(d) Tikhonov filter (e) Sign filter (f) Logarithmic filter

Figure 4. Image reconstruction in Example 4.



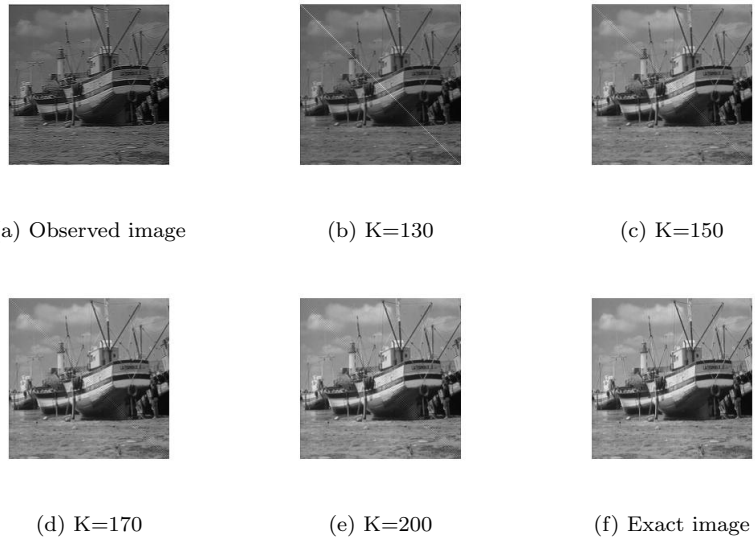
(a) Observed image (b) Residual norm

Figure 5. Residual norms and observed image in Example 4.

Example 5. The algorithms described in the previous sections reconstruct the non-diagonal elements of the solution at a faster speed. In this example, using the following two matrices

$$A = gallery('toeppen', m, 0.1, -0.4, 1, -0.4, 0.1),$$
$$B = gallery('toeppen', n, -0.1, 0.4, 1, 0.4, -0.1).$$

we apply $K = 130, 150, 170, 200$ basis matrices in the variable s-step algorithm to reconstruct the exact image from the observed image. The exact and observed images are shown in Figure 6. The results with the Tikhonov filter are presented in Figure 6.



(a) Observed image (b) K=130 (c) K=150

(d) K=170 (e) K=200 (f) Exact image

Figure 6. Image reconstruction in Example 5.

We can easily see that by increasing the number of iterations (i.e., the number of bases used), the image can be brought closer to the original image with appropriate accuracy, and the image resolution can also be enhanced.

6 Conclusions

This paper introduces a novel s-step algorithm for solving matrix equations, demonstrating several significant advantages as evidenced by our numerical results and previous discussion. The core contributions and findings can be summarized as follows:

1. The proposed algorithm significantly improves convergence rates and reduces runtime. The strategic application of the variable s-step technique positively impacts the computational efficiency, as corroborated by the

provided runtime tables. Furthermore, when coupled with appropriate regularization methods, this technique markedly enhances algorithm convergence. A crucial aspect of this improvement lies in selecting a suitable function for adjusting the parameter s at each iteration, which establishes a desirable balance between algorithmic stability and execution speed.

2. The algorithm exhibits remarkable insensitivity to symmetry. Our results clearly indicate that the presence or absence of symmetry in the images (as shown in Examples 1 and 2) does not adversely affect the algorithm's performance. Consistently, the residual norm decreases more rapidly compared to the original algorithm, irrespective of symmetry.

3. The efficacy of regularization filters is clearly demonstrated. The use of regularization filters, such as the Tikhonov filter, leads to improved numerical outcomes. Across various examples (e.g., Example 4), these filters consistently show similar performance.

4. The algorithm demonstrates a superior capability in reconstructing non-diagonal elements of the solution, achieving this at a faster pace (Example 5).

5. The flexibility in adjusting the parameter s is a key strength. The ability to define different functions for determining s (as illustrated in Example 3) provides the algorithm with adaptability, making it suitable for a diverse range of problems.

6. The proposed methodology shows broad applicability. Techniques traditionally employed for linear systems can be effectively extended to matrix equations after appropriate normalization.

7. While the algorithm offers substantial benefits, it also presents challenges. The determination of an optimal function for adjusting the s parameter at each iteration remains a critical and sensitive selection, requiring careful consideration.

In conclusion, the introduced algorithm stands as an efficient and robust method for solving matrix equations. By leveraging the variable s-step technique and appropriate regularization filters, it achieves faster convergence, reduced runtime, and stable performance across various conditions, including both symmetric and nonsymmetric scenarios.

Acknowledgements

We like to thank the editor and the anonymous referees for their comments, criticisms and suggestions which substantially improved the quality and presentation of this paper. Also we are deeply grateful to Professor Anthony T. Chronopoulos (The University Of Texas At San Antonio) for his valuable comments and assistance in obtaining the s-step algorithm.

References

[1] Z.-Z. Bai. Motivations and realizations of Krylov subspace methods for large sparse linear systems. *Journal of Computational and Applied Mathematics*, **283**:71–78, 2015. https://doi.org/10.1016/j.cam.2015.01.025.

[2] E. Carson, T. Gergelits and I. Yamazaki. Mixed precision s-step Lanczos and conjugate gradient algorithms. *Numerical Linear Algebra with Applications*, **29**(3):e2425, 2022. https://doi.org/10.1002/nla.2425.

[3] E.C. Carson. An adaptive s-step conjugate gradient algorithm with dynamic basis updating. *Applications of Mathematics*, **65**(2):123–151, 2020. https://doi.org/10.21136/AM.2020.0136-19.

[4] A.T. Chronopoulos and S.K. Kim. s-step orthomin and GMRES implemented on parallel computers. *arXiv*, **2001.04886**(v2):1–16, 2020. arXiv:2001.04886

[5] A.T. Chronopoulos and A.B. Kucherov. A parallel Krylov-type method for nonsymmetric linear systems. In *High Performance Computing — HiPC 2001*, volume 2228, pp. 104–114, Berlin, Heidelberg, 2001. Springer, Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-45307-5_10.

[6] M. Dehghan, G. Karamali and A. Shirilord. An iterative scheme for a class of generalized Sylvester matrix equations. *AUT Journal of Mathematics and Computing*, **5**(3):195–215, 2024. https://doi.org/10.22060/AJMC.2024.22444.1159.

[7] G.H. Golub and C.F. Van Loan. *Matrix computations*. Johns Hopkins University Press, Baltimore, MD, 4 edition, 2013. https://doi.org/10.56021/9781421407944.

[8] B. Hashemi and M. Dehghan. Results concerning interval linear systems with multiple right-hand sides and the interval matrix equation $ax = b$. *Journal of Computational and Applied Mathematics*, **235**(9):2969–2978, 2011. https://doi.org/10.1016/j.cam.2010.12.015.

[9] R.A. Horn and C.R. Johnson. *Matrix analysis*. Cambridge university press, Cambridge, 2012.

[10] H.S. Kaveh, M. Hajarian and A.T. Chronopoulos. Developing variable s-step CGNE and CGNR algorithms for non-symmetric linear systems. *Journal of the Franklin Institute*, **361**(14):107071, 2024. https://doi.org/10.1016/j.jfranklin.2024.107071.

[11] H.S. Kaveh, M. Hajarian and A.T. Chronopoulos. Efficient image reconstruction via regularized variable s-step conjugate gradient method for Sylvester matrix equations. *Journal of the Franklin Institute*, **362**(6):107634, 2025. https://doi.org/10.1016/j.jfranklin.2025.107634.

[12] H.S. Kaveh, M. Hajarian and A.T. Chronopoulos. Variable s-step generalized semi-conjugate gradient algorithm for solving nonsymmetric linear systems arising in signal recovery. *Journal of the Franklin Institute*, **362**(12):107870, 2025. https://doi.org/10.1016/j.jfranklin.2025.107870.

[13] G. Meurant and J.D. Tebbens. *Krylov methods for nonsymmetric linear systems*. Springer, Cham, 2020.

[14] S.M. Moufawad. S-step enlarged Krylov subspace conjugate gradient methods. *SIAM Journal on Scientific Computing*, **42**(1):A187–A219, 2020. https://doi.org/10.1137/18M1182528.

[15] M. Pernice and A.T. Chronopoulos. Vector preconditioned s-step methods on the IBM 3090/600S/6VF. In *Proceedings of the 5th SIAM Conference on Parallel Processing for Scientific Computing*, pp. 130–137, Houston, TX, 1991.

[16] Y. Saad. *Iterative Methods For Sparse Linear Systems*. SIAM, Philadelphia, 2003.

[17] A. Shirilord and M. Dehghan. Gradient-based iterative approach for solving constrained systems of linear matrix equations. *Computational and Applied Mathematics*, **43**(4):211, 2024. https://doi.org/10.1007/s40314-024-02687-6.

[18] A. Shirilord and M. Dehghan. Iterative algorithm for a generalized matrix equation with momentum acceleration approach and its convergence analysis. *Journal of the Franklin Institute*, **361**(12):107021, 2024. https://doi.org/10.1016/j.jfranklin.2024.107021.

[19] A. Shirilord and M. Dehghan. Gradient descent-based parameter-free methods for solving coupled matrix equations and studying an application in dynamical systems. *Applied Numerical Mathematics*, **212**:29–59, 2025. https://doi.org/10.1016/j.apnum.2025.01.011.

[20] S. Şimşek. A block quaternion GMRES method and its convergence analysis. *Calcolo*, **61**(2):33, 2024. https://doi.org/10.1007/s10092-024-00576-2.

[21] A.N. Tikhonov and V.Y. Arsenin. *Solutions of ill-posed problems*. V.H. Winston & Sons, Washington, 1977.

[22] Q. Wang and Z. He. A system of matrix equations and its applications. *Science China Mathematics*, **56**(9):1795–1820, 2013. https://doi.org/10.1007/s11425-013-4596-y.

[23] W. Wang, G. Qu, C. Song, Y. Ge and Y. Liu. Tikhonov regularization with conjugate gradient least squares method for large-scale discrete ill-posed problem in image restoration. *Applied Numerical Mathematics*, **204**:147–161, 2024. https://doi.org/10.1016/j.apnum.2024.06.010.

[24] Z. Xu, J.J. Alonso and E. Darve. A numerically stable communication-avoiding *s*-step GMRES algorithm. *arXiv*, **2303.08953**:1–36, 2023. arXiv:2303.08953

[25] I. Yamazaki, E. Carson and B. Kelley. Mixed precision *s*-step conjugate gradient with residual replacement on GPUs. In *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 886–896. IEEE, 2022. https://doi.org/10.1109/IPDPS53621.2022.00091.

[26] K. Zhou, J.C. Doyle and K. Glover. *Robust and optimal control*, volume 2. Prentice hall, 1996.