A note on the partial convergence management for the solution of of symmetric linear systems with multiple right-hand sides

Luc Giraud, Yanfei Xiang

A note on the partial convergence management for the solution of symmetric linear systems with multiple right-hand sides

Luc Giraud, Yanfei Xiang

RESEARCH REPORT
N° 9574
January 2025
Project-Team Concace

ISSN 0249-6399 ISRN INRIA/RR--9574--FR+ENG

A note on the partial convergence management for the solution of symmetric linear systems with multiple right-hand sides

Luc Giraud*, Yanfei Xiang*

Project-Team Concace

Research Report n° 9574 — January 2025 — 23 pages

Abstract: We consider the solution of large sparse symmetric linear systems with multiple right-hand sides available simultaneously. Based on the partial convergence detection and management, described in IB-BGMRES [Linear Algebra Appl., 419 (2006), pp. 265-285] and the breakdown-free idea discussed in [BIT Numer. Math., 57 (2017), pp. 379-403], the block conjugate residual and block conjugate gradient methods with partial convergence management are proposed. It enable to select the directions to use for extending the search space from one iteration to the next by choosing the directions that contribute the most to the residual norms. We illustrate the numerical and computational benefits of these two novel block conjugate direction variants on a set of simple academic examples enabling reproducible experiments.

Key-words: BGC, BCR, partial convergence, numerical robustness

* Inria, Inria centre at the University of Bordeaux

RESEARCH CENTRE
BORDEAUX – SUD-OUEST

200 avenue de la Vieille Tour
33405 Talence Cedex

Une note sur la gestion de la convergence partielle dans les solveurs linéaires pour matrices symétriques avec second-membres multiples

Résumé : Nous considérons la solution de grands systèmes linéaires symétriques clairsemés avec plusieurs côtés droits disponibles simultanément. Sur la base de la détection et de la gestion de la convergence partielle, décrites dans IB-BGMRES [Linear Algebra Appl., 419 (2006), pp. 265-285] et l'idée sans rupture discutée dans [BIT Numer. Math., 57 (2017), pp. 379-403], les méthodes de résidu conjugué par bloc et de gradient conjugué par bloc avec gestion de la convergence partielle sont proposées. Elle permet de sélectionner les directions à utiliser pour étendre l'espace de recherche d'une itération à l'autre en choisissant les directions qui contribuent le plus aux normes résiduelles. Nous illustrons les avantages numériques et informatiques de ces deux nouvelles variantes de directions du gradient conjugué par blocs sur un ensemble d'exemples académiques simples permettant des expériences reproductibles.

Mots-clés : BGC, BCR, convergence partielle, robustesse numérique

Contents

1 Introduction

For the solution of symmetric linear systems with multiple right-hand sides available simultaneously, we consider devising new block conjugate direction (residual/gradient) methods combine with a numerical component for managing the partial convergence of the multiple right-hand sides, originally referred to as the *inexact breakdown* mechanism in [14]. Partial convergence arises when one or several right-hand sides converge to the prescribe accuracy earlier than the others, or when a linear combination of some right-hand sides occurs. In such cases, the search block size can be reduced by retaining only a selected subset of directions typically used to extend the space. This adjustment ensures the robustness of the block conjugate direction methods while improving its computational efficiency.

In a matrix form the symmetric linear system with multiple right-hand sides writes

$$AX = B, \tag{1}$$

where $A \in \mathbb{R}^{n\times n}$ is a large sparse symmetric matrix, and $B = [b^{(1)}, b^{(2)}, \ldots, b^{(p)}] \in \mathbb{R}^{n\times p}$ ($p > 1$) are the multiple right-hand sides. When A is a symmetric positive definite matrix, the conjugate gradient method (CG) [7] is the standard choice. Alternatively, when A is symmetric and non-singular but not necessarily positive definite, the conjugate residual method (CR) [11] becomes applicable. Given the multiple right-hand sides shown in Equation (1) are available simultaneously, only the block conjugate gradient method (BCG) [12], a block version of CG, and corresponding straightforward block extension of CR (refers as BCR) are considered in this work. Under the context of block methods, the search space is defined as the sum of the Krylov subspaces associated with each individual right-hand side, so that at each iteration the search space is enlarged by p additional directions. Given the search space contains the individual Krylov subspaces, the convergence should be at least as fast as solving independently each right-hand side. Besides the block algorithm implementation enables the use of efficient BLAS-3 like computational kernels, the time to the solution is expected to be reduced. Unfortunately, these potential advantages come at the price of novel numerical difficulties. Such as the well-known *breakdown* [8, 12] issue caused by rank deficiency, which leads the columns of the block vectors become linear dependence. To be specific, when such breakdown occurs, at least one of the parameter matrices involved in the BCG or BCR is unavailable, leading these algorithms terminate early without finding a satisfactory approximation. Notably, Ji and Li developed a *breakdown-free* BCG method [8] with targeted formulations of the parameter matrices composed by an orthogonal search basis to avoid such breakdown issue. Another inherent challenge in block implementations is the *partial convergence* issue. Partial convergence arises when one or several right-hand sides converge to the prescribe accuracy earlier than the others, or when a linear combination of some right-hand sides occurs. Under the framework of the block version of the generalized minimum residual (GMRES) norm method [16] (refers as BGMRES), Robbé and Sadkane proposed the IB-BGMRES algorithm [14] with *inexact breakdown* (IB) mechanism for addressing the partial convergence issue. This IB mechanism ensures the robustness of IB-BGMRES and reduces the overall computational cost by focusing on the directions that contribute most to convergence. Additionally, it allows for the reintroduction of the abandoned information in next iteration when necessary, ensuring efficiency without sacrificing accuracy. Refer to IB-BGMRES-DR [1] and IB-BGCRO-DR [5] for some further applications of such IB mechanism to other block minimum residual norm methods with long-term recurrences. Although the numerical properties and advantages of the IB mechanism have been demonstrated in [1, 5, 14], to the best of our knowledge, related studies addressing partial convergence in the context of BCR and BCG methods with short-term recurrences have not yet been documented in the literature.

Based on the aforementioned discussions, this work introduces enhanced block conjugate direction methods (i.e., BCR and BCG) integrated with numerical mechanisms to address both the breakdown [8] and inexact breakdown [14] issues appeared when solving symmetric linear systems with multiple right-hand sides. As a result, two novel block conjugate direction methods with partial convergence detecting are proposed: IB-BCR and IB-BCG. These methods incorporate parameter matrices formulated using the breakdown-free strategy outlined in [8] while applying the inexact breakdown mechanism [14] to the residuals. This combination allows for adaptive adjustment of the search block size by retaining only a carefully selected subset of search directions, typically those contribute most to reduce the residual norm. This design enhances the robustness of the block conjugate direction methods while significantly improving their computational efficiency. Note that in the context of block methods with long-term recurrence, as discussed by Langou in [9], simply removing the converged parts may result in a loss of critical information, potentially slowing down convergence. To address this, the abandoned directions can be reintroduced in subsequent

iterations, as demonstrated in IB-BGMRES [14]. For block conjugate direction methods with short-term recurrence, however, the directions abandoned in a given iteration cannot be reintroduced in later iterations since there is no way to keep them without destroying the short-term recurrence structure.

The remainder of this manuscript is organized as follows. Section 2, the main focus of this work, details the developments aimed at devising novel block conjugate direction methods: IB-BCR and IB-BCG variants, equipped with breakdown handling [8, 12] and partial convergence detecting mechanism [14]. Section 3 demonstrates various numerical behaviors and features of the proposed block conjugate direction methods, with concluding remarks presented in Section 4.

The notations used in this work are illustrated as follows. The notation $\mathbb{R}$ refers to the real number field. The vectors are denoted by lowercase letters; matrices with multiple columns are described by uppercase letters. The symbol $||\cdot||$ denotes the Euclidean norm defaultly for both vectors and matrices. The superscript ^{T} denotes the transpose. For convenience of the algorithm illustration and presentation, some MATLAB notation and function are used. Without special note, a subscript $_j$ for a vector (in the single right-hand case) or a matrix (in the block case) is used to indicate that the vector or matrix is obtained at iteration j. The $\text{span}\{c_0, c_1, \ldots, c_j\}$ refers to a space spanned by the vectors $c_0, c_1, \ldots, c_j$. A matrix $C \in \mathbb{R}^{m\times\ell}$ consisting of m rows and ℓ columns sometimes is denoted as $C_{m\times\ell}$ explicitly. The identity and null matrices of dimension k are denoted respectively by I_k and 0_k or just I and 0 when the dimension is evident from the context.

2 Block conjugate residual and conjugate gradient variants

In this section, we first retrospect the relationship and key properties of the conjugate gradient (CG) [7] and conjugate residual (CR) [11] algorithms for a single right-hand side in Section 2.1. Then, in Section 2.2, we extend these methods to their block versions and briefly review the breakdown [8, 12] issue arising from rank deficiency in the BCG method, which also manifests in the BCR context. In Section 2.3, we present an alternative approach using the inexact breakdown (IB) [14] mechanism to address partial convergence or linear combination issues. This section explores how the IB mechanism can be integrated into the BCR and BCG algorithms, enhancing their robustness and computational efficiency. This leads to two novel block conjugate direction methods, IB-BCR and IB-BCG, equipped with breakdown handling and partial convergence detection. We also introduce a simpler remedy to handle partial or individual convergence.

2.1 Conjugate residual and conjugate gradient for single right-hand side

The CG method [7] is a well-established approach for solving linear systems with symmetric positive definite coefficient matrices. When the coefficient matrix is symmetric but not necessarily positive definite, the CR algorithm [11] serves as a viable alternative. Notably, the CR algorithm can be viewed as a generalized form of the CG method, as elaborated by Hestenes in his work [6].

For a single right-hand side solution, the CR method is based on the Lanczos process to generate the orthogonal basis for the Krylov subspace $\mathcal{K}_n(A, r_0) = \text{span}\{r_0, Ar_0, \cdots, A^{n-1}r_0\}$, in which r_0 denotes the residual vector associated with the initial guess. In the CR method, the residual vectors r_j are A-conjugate, meaning they satisfy: $r_i^T A r_j = 0$ for $i \neq j$, while the vectors Ap_j (where p_j are the search directions) are orthogonal to each other [15, Section 6.8]: $(Ap_i)^T Ap_j = 0$ for $i \neq j$. This property is in contrast to the CG method, where the search directions p_j are A-conjugate, and the residuals $r_j$are orthogonal to each other. Moreover when addressing symmetric systems, the CR method is mathematically equivalent to the minimal residual (MINRES) method [15] and the generalized minimum residual (GMRES) norm method [16], as all these methods minimize the 2-norm of the residual over the same subspaces. While the CG method minimizes the A-norm of the forward error. We refer the reader to [15, Algorithm 6.20] for the implementation of CR and to [17, Algorithm 3.1], [18, Algorithm 2] for its preconditioned variant depicted in Algorithm 1. We notice that in the preconditioned case the preconditioned residuals are A-conjugate and the Ap_j's are M-orthogonal (refer to [10, Algorithm 3] and [13, Section 6] for more discussions) assuming that the preconditioner is symmetric positive definite as for MINRES.

Algorithm 1 *Preconditioned conjugate residual method for* $Ax = b$

Require: $A \in \mathbb{R}^{n\times n}$, the left-hand side of the linear systems, and a preconditioner $M \in \mathbb{R}^{n\times n}$, an approximation of the inverse of A
Require: $b \in \mathbb{R}^n$, the right-hand side, and $x_0 \in \mathbb{R}^n$, the initial guess
Require: m maximum number of the iteration step
Compute $r_0 = b - Ax_0$, $z_0 = Mr_0$, $p_0 = z_0$
for $j = 0, 1, 2, \ldots, m$ **do**
  $\alpha_j = z_j^T Az_j/(MAp_j)^T Ap_j$
  $x_{j+1} = x_j + \alpha_j p_j$
  $r_{j+1} = r_j - \alpha_j(Ap_j)$
  $z_{j+1} = Mr_{j+1}$
  $\beta_j = z_{j+1}^T Az_{j+1}/z_j^T Az_j$
  $p_{j+1} = z_{j+1} + \beta_j p_j$
end for
Ensure: Return x_{j+1}, the computed solution

2.2 Block extensions and their possible breakdowns and alternative remedies

When addressing linear systems (1) with multiple right-hand sides, where $p > 1$, the block extensions of the conjugate direction methods, as discussed in Section 2.1, are generally considered. Because block methods define the search space as the sum of the Krylov subspaces associated with each individual right-hand side, so that at each iteration the search space is enlarged by p additional directions rather than a single direction. Starting from a block initial guess $X_0 = [x_0^{(1)}, x_0^{(2)}, \ldots, x_0^{(p)}] \in \mathbb{R}^{n\times p}$ and the associated block initial residual $R_0 = B - AX_0$, the block conjugate gradient (BCG) method was proposed by O'Leary in [12] to solve such systems with a symmetric positive definite coefficient matrix and multiple right-hand sides provided simultaneously. Its preconditioned version is outlined in Algorithm 2.

Algorithm 2 Block preconditioned conjugate gradient method [12] for $AX = B$

Require: $A \in \mathbb{R}^{n\times n}$ the left-hand side of the linear systems and a preconditioner $M \in \mathbb{R}^{n\times n}$ be an approximation of the inverse of A
Require: $B \in \mathbb{R}^{n\times p}$ the of right-hand-sides and $X_0 \in \mathbb{R}^{n\times p}$ the block initial guess
Require: m maximum number of the block iteration step
Compute $R_0 = B - AX_0$, $Z_0 = MR_0$, $P_0 = Z_0$
for $j = 0, 1, 2, \ldots, m$ **do**
  $\alpha_j = (P_j^T AP_j)^{-1}(Z_j^T R_j)$
  $X_{j+1} = X_j + P_j\alpha_j$
  $R_{j+1} = R_j - AP_j\alpha_j$
  $Z_{j+1} = MR_{j+1}$
  $\beta_j = (Z_j^T R_j)^{-1}(Z_{j+1}^T R_{j+1})$
  $P_{j+1} = Z_{j+1} + P_j\beta_j$
end for
Ensure: Return X_{j+1} computed solution

The block algorithm implementation enables the use of efficient BLAS-3 like computational kernels, thus the time to the solution is expected to be reduced. Unfortunately, this potential advantage comes at the price of novel numerical difficulties. As mentioned in [12], in the implementation of BCG, the block parameter matrices α_j and β_j involved in the jth iteration of BCG are respectively formulated as

$$\alpha_j = (P_j^T AP_j)^{-1}(R_j^T Z_j) \in \mathbb{R}^{p\times p},$$

and

$$\beta_j = (R_j^T Z_j)^{-1}(R_{j+1}^T Z_{j+1}) \in \mathbb{R}^{p\times p}, \text{ with } Z_{j+1} = MR_{j+1}.$$

Exploiting the key orthogonal properties of the CR method [11] and the original ideas of O'Leary for

the preconditioned BCG method [12], the core loop of preconditioned BCR algorithm essentially reads

$$\begin{aligned} X_{j+1} &= X_j + P_j\alpha_j, \\ R_{j+1} &= R_j - Q_j\alpha_j, \\ Z_{j+1} &= MR_{j+1}, \\ P_{j+1} &= Z_{j+1} + P_j\beta_j, \quad (2) \\ Q_{j+1} &= AP_{j+1}, \quad (3) \end{aligned}$$

where α_j and β_j are the parameter matrices to be determined such that $Z_{j+1}^T AZ_j = 0$ and $Q_{j+1}^T MQ_j = 0$. Specifically, these parameter matrices are defined by

$$\alpha_j = (Q_j^T MQ_j)^{-1}(Q_j^T Z_j) \in \mathbb{R}^{p\times p},$$

and

$$\beta_j = -(Q_j^T MQ_j)^{-1}((MQ_j)^T AZ_{j+1}) \in \mathbb{R}^{p\times p},$$

in which some matrix-multiplications related to a $p \times p$ non-singular matrix may be involved during the practical implementation to improve the stability [12]. Based on the orthogonality properties $Z_{j+1}^T Q_j = 0$ and $Q_{j+1}^T MQ_j = 0$, as well as the core loop of the preconditioned BCR algorithm, we can derive an additional orthogonality property $Z_{j+1}^T AZ_j = 0$ as follow:

$$\begin{aligned} Z_{j+1}^T AZ_j &= Z_{j+1}^T(Q_j - Q_{j-1}\beta_{j-1}) = -Z_{j+1}^T Q_{j-1}\beta_{j-1} = -R_{j+1}^T MQ_{j-1}\beta_{j-1} \\ &= (Q_j\alpha_j - R_j)^T MQ_{j-1}\beta_{j-1} = \alpha_j^T Q_j^T MQ_{j-1}\beta_{j-1} - Z_j^T Q_{j-1}\beta_{j-1} = 0. \end{aligned}$$

This leads to the fact that BCR generates the jth approximate solution X_j such that the Euclidean norm of the corresponding block residual R_j is minimized over the increasing subspaces $X_0 + \mathcal{K}_j(MA, MR_0)$ with $\mathcal{K}_j(MA, MR_0) = \oplus_{\ell=1}^p \mathcal{K}_j(MA, Mr_0^{(\ell)})$, i.e.,

$$\|R_j\| = \min_{X_j \in X_0 + \mathcal{K}_j(MA, MR_0)} \|B - AX_j\|,$$

which is the same as the case for the block minimum residual method (Block MINRES) [12, Section 3].

In BCG, if the matrices $(P_j^T AP_j)$ or $(R_j^T Z_j)$ are singular this leads to the so-called *breakdown* issue [12] because the algorithm terminates early without finding a satisfactory approximate solution. Such a situation occurs, when some rank deficiency appears simultaneously in R_j and P_j as shown in [8, Proposition 2.1]. The loss of rank in R_j indicates that either some solutions have converged or some linear combinations of the solutions have converged in the search space at the j^{th} iteration. The same situation might occur in BCR with the parameter matrix $(Q_j^T MQ_j)$ that can become singular if some rank deficiency appear simultaneously in Q_j and R_j as shown in Proposition 2.1.

Proposition 2.1. *For the preconditioned BCR algorithm, suppose A and M are non-singular symmetric matrices, R_j be an $n \times p$ residual matrix with rank p_j ($p_j \leq p$) at the j-th iteration, then, we have*

$$\textit{rank}(P_j) = \textit{rank}(Q_j) = \textit{rank}(Z_j) = \textit{rank}(R_j) = p_j, \quad (4)$$

where rank$(\cdot)$ denotes the rank of a matrix.

Proof. From the core loop of the preconditioned BCR algorithm, we have $Z_j = MR_j$ and $Q_j = AP_j$. Because A and M are nonsigular, we have $\text{rank}(Z_j) = \text{rank}(R_j) = p_j$ and $\text{rank}(P_j) = \text{rank}(Q_j)$.

In the sequel we will use twice the similar arguments that is, if $C \in \mathbb{R}^{n\times n}$ is a symmetric non-singular matrix and $E \in \mathbb{R}^{n\times p}$ then

$$\text{rank}(E^T CE) = \text{rank}(E). \quad (5)$$

Because C is a symmetric matrix, it is diagonalizable in an orthonormal basis, that is $C = WDW^T$ where the columns of the unitary matrix W are the unitary eigenvectors and D the diagonal of the eigenvalues. Consequently $E^T CE = E^T WDW^T E = \tilde{E}^T D\tilde{E}$ with $\tilde{E} = W^T E$ and $\text{rank}(\tilde{E}) = \text{rank}(E)$. If we note $\tilde{E} = U\Sigma V^T$ the SVD decomposition of $\tilde{E}$ with U and V unitary matrices we have

$$\text{rank}(E^T CE) = \text{rank}(V\Sigma^T U^T DU\Sigma V^T) = \text{rank}(\Sigma^T D\Sigma) = \text{rank}(\Sigma) = \text{rank}(\tilde{E}) = \text{rank}(E).$$

From Equation (2)-(3), the block vector Q_j is given by

$$Q_j = AZ_j + Q_{j-1}\beta_{j-1}. \tag{6}$$

Left multiplying (6) by $Q_j^T M$ on both sides, we get

$$Q_j^T MQ_j = Q_j^T MAZ_j + Q_j^T MQ_{j-1}\beta_{j-1} = Q_j^T MAZ_j \tag{7}$$

by definition of the parameter matrices $Q_j^T MQ_{j-1} = 0$. Because M is symmetric, then using Equation (7) and (5) as well as some basic properties of matrix rank, we have

$$\text{rank}(Q_j) = \text{rank}(Q_j^T MQ_j) = \text{rank}(Q_j^T MAZ_j) \leq \text{rank}(Z_j) = \text{rank}(R_j). \tag{8}$$

On the other hand, left multiplying (6) by Z_j^T on both sides, we get

$$Z_j^T Q_j = Z_j^T AZ_j + Z_j^T Q_{j-1}\beta_{j-1} = Z_j^T AZ_j \tag{9}$$

by the orthogonal property $Z_j^T Q_{j-1} = 0$.

Then, apply Equation (9) and (5) as well as some basic properties of matrix rank, we have

$$\text{rank}(Z_j) = \text{rank}(Z_j^T AZ_j) = \text{rank}(Z_j^T Q_j) \leq \text{rank}(Q_j) = \text{rank}(P_j). \tag{10}$$

Based on relation (8) and (10), we eventually have $\text{rank}(P_j) = \text{rank}(Q_j) = \text{rank}(Z_j) = \text{rank}(R_j)$. □

To overcome the singularity, Ji and Li proposed the breakdown-free BCG method [8], where the block search directions P_j are replaced by $\widetilde{P}_j$, an orthonormal basis of the space spanned by P_j, which can be computed by considering the left singular vectors of P_j associated with non-zero singular values. Although, this alternative makes sense in exact arithmetic, it can be somehow relaxed when a prescribed accuracy is required. In that situation, the partial convergence detection introduced in [14] can be adapted to the two block algorithms to monitor the size block according to a prescribed threshold for the individual convergence criterion based on backward error.

2.3 Partial convergence management

In this section, we investigate the application of the partial convergence management, referred to as *Inexact Breakdown* (IB) mechanism [14], to effectively tackle the challenges of variable convergence speed for the different right)hand sides. This IB mechanism [14] was initially devised to address partial convergence in the block minimum residual norm methods with long-term recurrences, and has been further extended in [1, 5] to the block versions of GMRES and the generalized conjugate residual with inner orthogonalization (GCRO) [19] algorithms. Since BCR is also a type of minimum residual norm method, the IB mechanism can be adapted to enhance its robustness and efficiency in addressing these numerical challenges. However, two main differences exist. First, from a numerical perspective, since BCR involves short-term recurrences, the directions abandoned at a given iteration cannot be reintroduced later, as is possible in methods with long-term recurrences [1, 5, 14] to avoid the loss of critical information [9]. This limitation may make its application to the BCR method be less efficient compared to methods discussed in [1, 5, 14], as there is no way to retain these directions without disrupting the short-term recurrence structure of BCR. Second, from a computational point of view, the algorithm does not compute a QR-factorization of the residual block, contrarily to block GMRES and GCRO where it is a byproduct of the least squares problem solution (refer to [5, Section 4] for more details). Consequently, the partial convergence policies will require the reduced singular value decomposition (SVD) calculation of a tall and skinny matrix, namely of the residual block R_j scaled by a certain diagonal matrix D_ε that depends on the selected stopping criterion and convergence threshold ε, which could be described as the following form:

$$R_j D_\varepsilon = \mathbb{U}_{1,L}\Sigma_1\mathbb{V}_{1,R}^T + \mathbb{U}_{2,L}\Sigma_2\mathbb{V}_{2,R}^T \text{ with } \sigma_{\min}(\Sigma_1) \geq \tau > \sigma_{\max}(\Sigma_2),$$

where $D_\varepsilon = \varepsilon^{-1}$ diag $(\|b^{(1)}\|^{-1}, \ldots, \|b^{(p)}\|^{-1}) \in \mathbb{R}^{p\times p}$, with $\tau = 1$ the prescribed IB-threshold. The vectors $(\mathbb{U}_{1,L}, \mathbb{U}_{2,L})$ and $(\mathbb{V}_{1,R}, \mathbb{V}_{2,R})$ are the left and right singular vectors of $R_j D_\varepsilon$, respectively. Once a partial convergence is detected, all the calculations but the update of the solution and residual blocks are

performed on blocks of lower column dimension, which lowers the number of matrix-vector products and preconditioning applications. Let us denote

$$[U_j^L, W_j] = SpaceExpansion(R_j, \varepsilon), \tag{11}$$

where $U_j^L \in \mathbb{R}^{n \times p_j}$ are the p_j left singular vectors computed by the selected partial convergence detection mechanism and $W_j = (U_j^L)^T R_j \in \mathbb{R}^{p_j \times p}$ the components of the residuals in the space spanned by U_j^L, that are the new directions to be added to the search space. The algorithm is depicted in Algorithm 3, where a bar notation is used to indicate that the dimensions of these matrices may reduce in case of partial convergence detection.

Algorithm 3 *Block preconditioned Conjugate Residual method with partial convergence detection (or Inexact-Breakdown) mechanism — IB-BCR*

Require: $A \in \mathbb{R}^{n \times n}$ the left-hand side of the linear systems and a preconditioner $M \in \mathbb{R}^{n \times n}$ be an approximation of the inverse of A
Require: $B \in \mathbb{R}^{n \times p}$ the block of right-hand sides and $X_0 \in \mathbb{R}^{n \times p}$ the block initial guess
Require: m maximum number of the block iteration step and the maximum number of matrix-vector products is set to be $maxMvps \in \mathbb{N}^+$
Require: $\varepsilon > 0$ a threshold for the selected backward error used in stopping criterion
$U_0^L, W_0 = SpaceExpansion(B - AX_0, \varepsilon)$ and $\overline{Z}_0 = MU_0^L$
$\overline{P}_0 = \overline{Z}_0$
for $j = 0, 1, 2, \ldots, m$ **do**
  $\overline{Q}_j = A\overline{P}_j$
  $\overline{\alpha}_j = (\overline{Q}_j^T M \overline{Q}_j)^{-1} (\overline{Q}_j^T \overline{Z}_j)$
  $X_{j+1} = X_j + \overline{P}_j \overline{\alpha}_j W_j$
  $R_{j+1} = R_j - \overline{Q}_j \overline{\alpha}_j W_j$
  if the stopping criterion related to ε or $maxMvps$ is met **then**
    return X_{j+1}
  else
    $[U_{j+1}^L, W_{j+1}] = SpaceExpansion(R_{j+1}, \varepsilon)$
    $\overline{Z}_{j+1} = MU_{j+1}^L$
    $\overline{\beta}_j = -(\overline{Q}_j^T M \overline{Q}_j)^{-1} ((M\overline{Q}_j)^T A \overline{Z}_{j+1})$
    $\overline{P}_{j+1} = \overline{Z}_{j+1} + \overline{P}_j \overline{\beta}_j$
  end if
end for
Ensure: Return X_{j+1}

We notice that Algorithm 3 can be accommodated to implement a crude search space expansion that simply discards the columns of the block that correspond to individual solutions that have converged. In that case, U_j^L consists of the columns of the residual that have not yet converged, and $W_j \in \mathbb{R}^{p \times p}$ is a diagonal matrix with entry equal to 0 when the corresponding right-hand side has converged. This somewhat naive alternative algorithm will be referred to as IC-BCR, for Individually Converged BCR.

Although the CG method [6] and its block version BCG [12] minimize the A-norm of the forward error, their most commonly used stopping criterion still relies on a backward error that is mostly based on the residual norm. This observation motivates us to adapt BCG with the previous developed partial convergence mechanism in the context of BCR. By exploiting the algorithmic resemblance between BCG and BCR, the development of corresponding IB-BCG counterpart is fairly straightforward and depicted in Algorithm 4. We conclude that the IB-BCG algorithm with short-term recurrences might lack efficiency and robustness for two main reasons. First, similar to the BCR case, it may be less efficient compared to algorithms with long-term recurrences, such as those discussed in [1, 5, 14], since the abandoned directions cannot be reintroduced in later iterations without disrupting the short-term recurrence structure. Second, it may lack robustness because BCG minimizes the A-norm of the errors, and therefore controlling the search space expansion through heuristics based on residuals does not fully align with this optimal numerical feature. Finally, the corresponding IC (Individually Converged) variant can also be considered; of does not

suffer from the two previously possible flaws of IB-BCG as it mostly consists in stopping iterating on the iterates that have converged.

Algorithm 4 Block preconditioned Conjugate Gradient method with partial convergence (or Inexact-Breakdown detection) mechanism — IB-BCG

Require: $A \in \mathbb{R}^{n\times n}$ the left-hand side of the linear systems and a preconditioner $M \in \mathbb{R}^{n\times n}$ that is an approximation of the inverse of A
Require: $B \in \mathbb{R}^{n\times p}$ the block of right-hand-sides and $X_0 \in \mathbb{R}^{n\times p}$ the block initial guess
Require: m maximum number of the block iteration step and the maximum number of matrix-vector products ($\#mvps$) is set to be $maxMvps \in N^+$
Require: $\varepsilon > 0$ a threshold for the selected backward error used in stopping criterion
$U_0^L, W_0 = SpaceExpansion(B - AX_0, \varepsilon)$ and $\overline{Z}_0 = MU_0^L$
$\overline{P}_0 = (\overline{Z}_0)$
for $j = 0, 1, 2, \ldots, m$ **do**
  $\overline{Q}_j = A\overline{P}_j$
  $\overline{\alpha}_j = (\overline{P}_j^T\overline{Q}_j)^{-1}(\overline{P}_j^T\overline{R}_j)$
  $X_{j+1} = X_j + \overline{P}_j\overline{\alpha}_j W_j$
  $R_{j+1} = R_j - \overline{Q}_j\overline{\alpha}_j W_j$
  if the stopping criterion related to ε or $maxMvps$ is met **then**
    return X_{j+1}
  else
    $[U_{j+1}^L, W_{j+1}] = SpaceExpansion(R_{j+1}, \varepsilon)$
    $\overline{Z}_{j+1} = MU_{j+1}^L$
    $\overline{\beta}_j = -(\overline{P}_j^T\overline{Q}_j)^{-1}(\overline{Q}_j^T\overline{Z}_{j+1})$
    $\overline{P}_{j+1} = (\overline{Z}_{j+1} + \overline{P}_j\overline{\beta}_j)$
  end if
end for
Ensure: Return X_{j+1} for approximation of the linear systems

We illustrate in the next sections about the complementary numerical performances of the proposed IB-BCR and IB-BCG variants with breakdown-free [8] and partial convergence detecting mechanism [14], as well as their naive IC-variants for individual convergence detection.

3 Numerical experiments

Numerical experiments are carried out on a set of symmetric positive definite (SPD) and symmetric but not positive definite matrices from the University of Florida Sparse Matrix Collection [4]. The main features of these SPD and symmetric matrices are respectively described in Table 1 and Table 2. In order to illustrate the numerical benefits of the proposed IB-BCR (shown in Algorithm 3) and IB-BCG (shown in Algorithm 4) methods, the performance of IB-BCR and IB-BCG are respectively evaluated in comparison with that of the classical BCR and classical BCG and their IC variants in terms of the consumed number of matrix-vector products (referred to as $\#mvps$) and the consumed block iteration steps (referred to as $\#iter$). Given the error related with true solution of linear system is generally unavailable in practice, we compare the convergence histories of the related methods (i.e., BCR and BCG as well as their IB and IC variants) in terms of the smallest and largest backward errors related to Euclidean norm of the residuals at each $\#mvps$ within the following sections, in which the stopping criterion is that the p individual normwise backward errors satisfy $\eta_{b^{(i)}}(x_j^{(i)}) = \frac{||b^{(i)} - Ax_j^{(i)}||}{||b^{(i)}||} < \varepsilon$ $(i = 1, \ldots, p)$ with respect to the approximate solution $x_j^{(i)}$, or $\#mvps$ exceeds the allowed maximum number of matrix-vector products (referred to as $maxMvps$).

In the default setting of the experiments, the block initial guess is set to be $0 \in \mathbb{R}^{n\times p}$, where p represents the number of the right-hand sides. The multiple right-hand sides $B = \texttt{randn}(n, p) = [b^{(1)}, b^{(2)}, \ldots, b^{(p)}] \in \mathbb{R}^{n\times p}$ are composed of p linearly independent vectors containing pseudo-random values drawn from the standard normal distribution (using the same seed when comparing these block methods). The search space expansion policy used in conjunction with the partial con-

vergence detecting is based on the backward error η_b. Without special notes, the $maxMvps$ is set to be $5000 \times p$ for each solver run, the convergence threshold is $\varepsilon = 10^{-8}$. For all the experiments involving SPD matrices, we consider the preconditioned BCR and BCG variants, where an incomplete Cholesky factorization is employed by default as the preconditioner. For the symmetric but not positive definite ones, no preconditioner is considered.

The experiments have been carried out in personal Linux (double precision floating point arithmetic) system by MATLAB (R2019a) with hardware setting as PC-Intel (R) Core (TM) i7-8665U CPU @ 1.90 GHz, 8 GB RAM. In order to evaluate the robustness and efficiency of the newly proposed IB-BCR and IB-BCG variants, we first investigate in Section 3.1 their numerical behavior when the set of right-hand sides is not full rank. Next we investigate their behavior when the convergence threshold ε varies in Section 3.2 and when the number p of right-hand sides varies in Section 3.3. In Section 3.4, we consider examples where all the solutions do not need to be computed using the same convergence threshold. Finally, we report on symmetric but not positive definite examples in Section 3.5.

Table 1: Main characteristics of the symmetric positive definite matrices

Name	n	Nonzero	Origin	Cond. number
apache1	80,800	542,184	Stru. Prob.	
bcsstk15	3,948	117,816	Stru. Prob.	6.53e+09
bcsstk16	4,884	290,378	Stru. Prob.	4.94e+09
bcsstk17	10,974	428,650	Stru. Prob.	1.29e+10
bcsstk18	11,948	149,090	Stru. Prob.	3.45e+11
bundle1	10,581	770,811	Comp. Grap./Vis. Prob.	1.00e+03
cbuckle	13,681	676,515	Stru. Prob.	3.29e+07
crankseg_1	52,804	10,614,210	Stru. Prob.	
crankseg_2	63,838	14,148,858	Stru. Prob.	
gridgena	48,962	512,084	Opti. Prob.	
gyro	17,361	1,021,159	Model Redu. Prob.	1.09e+09
Kuu	7,102	340,200	Stru. Prob.	1.57e+04
s1rmq4m1	5,489	262,411	Stru. Prob.	1.81e+06
s1rmt3m1	5,489	217,651	Stru. Prob.	2.54e+06
s2rmq4m1	5,489	263,351	Stru. Prob.	1.77e+08
s2rmt3m1	5,489	217,681	Stru. Prob.	2.49e+08
s3rmq4m1	5,489	262,943	Stru. Prob.	1.76e+10
s3rmt3m1	5,489	217,669	Stru. Prob.	2.48e+10

Table 2: Main characteristics of the symmetric but not positive definite matrices

Name	n	Nonzero	Origin	Cond. number
benzene	8,219	242,669	T/QC Prob.	1.45e+03
rail_5177	5,177	35,185	Model Redu. Prob.	5.33e+04
saylr4	3,564	22,316	CFD Prob.	6.86e+06

3.1 Partial convergence with full rank and rank deficient set of right-hand sides

In order to illustrate the benefit of using partial convergence detection, we consider the case where the right-hand sides are not full rank. For those experiments we select the matrix Kuu with full rank right-hand sides $B = \texttt{randn}(n, p)$ or rank deficient one $B = [B_{\text{pre}}, B_{\text{pre}}\ \texttt{randn}(p/2, p/2)]$ with $B_{\text{pre}} = \texttt{randn}(n, p/2)$ and $p = 20$.

The convergence histories of BCR variants for full rank right-hand sides are displayed in the left graph of Figure 1. Several observations can be made. The convergence histories of these three solvers overlap as long as no partial convergence is detected. After this first partial convergence, the convergence rate of IB-BCR becomes faster (in terms of $\#mvps$) than that of BCR and IC-BCR. And the largest and smallest backward errors of IB-BCR smoothly and simultaneously decreases to the target threshold. The more naive, but cost-free, variant IC-BCR exhibits a plateau for the largest backward error when the first right-hand side has converged, then followed by a super fast convergence. Finally, the breakdown-free BCR variant illustrates the drawback of a block solver without any partial or individual convergence detection, that is, many directions are introduced in the search space that enables some backward errors to go below the convergence threshold without special attention to the right-hand sides that converge the slowest. Results of the block size p_j along the iterations are displayed in the right graph of Figure 1, from which we noticed that the block size remains constant for BCR, while it reduces progressively for IB-BCR and more abruptly for IC-BCR as most of the right-hand sides converge at the same iterations for that example. Corresponding convergence histories of BCG variants are shown in Figure 2, similar comments can be made.

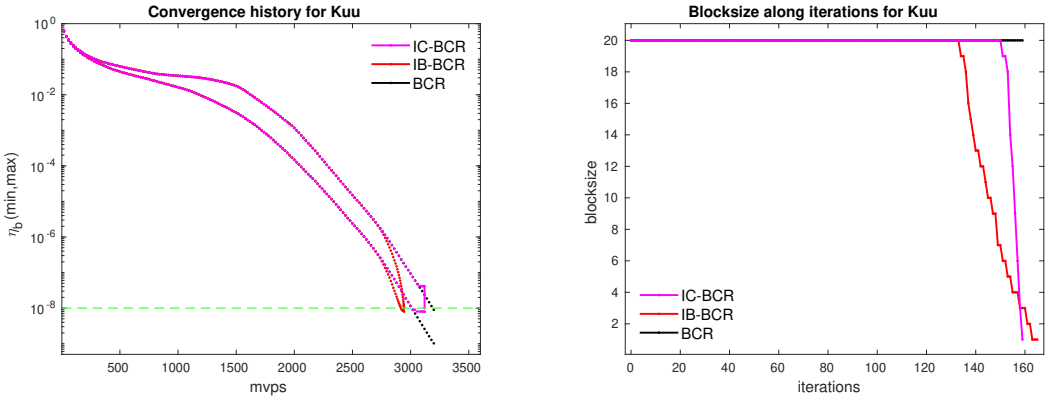


Figure 1: BCR variants for full rank right-hand sides with $p = 20$. Left: convergence histories of the largest/smallest backward errors $\eta_{b^{(i)}}$ as a function of the number of matrix-vector products $\#mvps$. Right: block size p_j along the iterations.

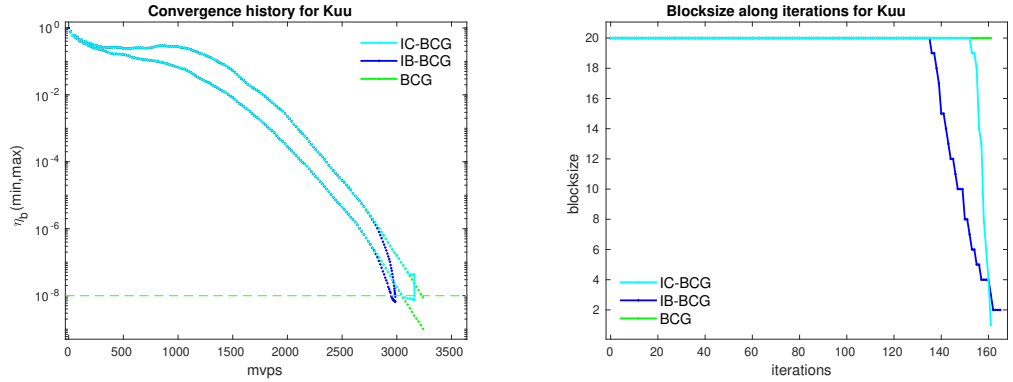


Figure 2: BCG variants for full rank right-hand sides with $p = 20$. Left: convergence histories of the largest/smallest backward errors $\eta_{b^{(i)}}$ as a function of the number of matrix-vector products $\#mvps$. Right: block size p_j along the iterations.

For the BCR variants, we illustrate in Figure 3 the robustness introduced by the partial convergence detection mechanism in a fake and somehow extreme case where the rank of the p right-hand sides is $p/2$. As it can be seen in the right graph of Figure 3, the rank deficiency is immediately detected by the IB variant that reduces to $p/2$ block size at the very first iteration. Although no real breakdown is encountered by the other two variants, their convergence is very slow due to fact that the block size remains equal to p despite the rank deficiency (except for the very last iterations in the IC variant), which also

reveals some lack of robustness. The displayed backward errors (left plot of Figure 3) are computed using the norm of the iterative residual, the true residual being computed only when the iterative one meets the convergence criterion. Corresponding numerical performance in terms of the number of matrix-vector products $(\#mvps)$ and block iterations $(\#iter)$ of these three block variants are summarized in Table 3. Corresponding numerical performance of BCG variants are summarized in Table 4 and Figure 4.

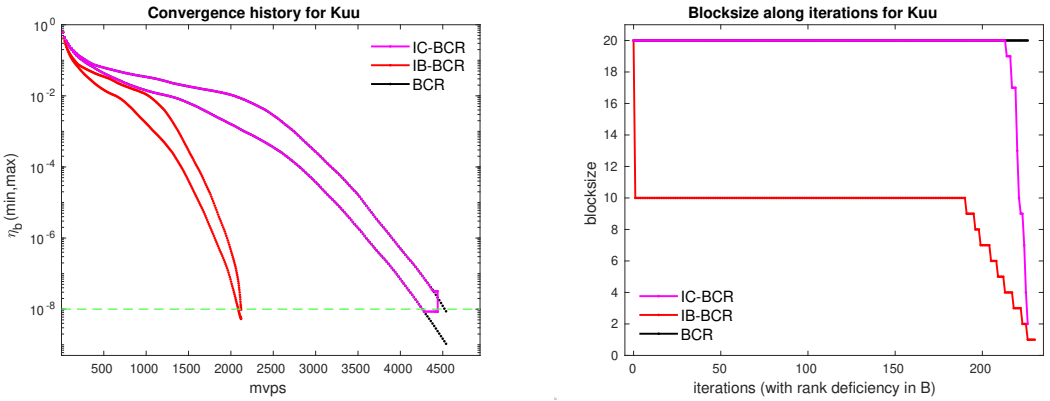


Figure 3: Same case as Figure 1 but the right-hand sides are linearly dependent as $B = [B_{\text{pre}},\ B_{\text{pre}}\ \texttt{randn}(p/2, p/2)]$ with $B_{\text{pre}} = \texttt{randn}(n, p/2)$.

Table 3: Numerical results of BCR variants in terms of $\#mvps$ and $\#iter$ for matrices Kuu with full rank and rank deficient set of right-hand sides for Section 3.1.

Columns in the right-hand sides B	Method	$\#mvps$	$\#iter$
linearly independent (Results for Figure 1)	BCR	3200	160
	IB-BCR	**2944**	166
	IC-BCR	3121	160
linearly dependent (Results for Figure 3)	BCR	4540	227
	IB-BCR	**2121**	231
	IC-BCR	4442	227

Table 4: Numerical results of BCG variants in terms of both $\#mvps$ and $\#iter$ for matrices Kuu with full rank and rank deficient set of right-hand sides for Section 3.1.

Columns in the right-hand sides B	Method	$\#mvps$	$\#iter$
linearly independent (Results for Figure 2)	BCG	3240	162
	IB-BCG	**2987**	166
	IC-BCG	3162	162
linearly dependent (Results for Figure 4)	BCG	2879	**231**
	IB-BCG	**2152**	232
	IC-BCG	2977	235

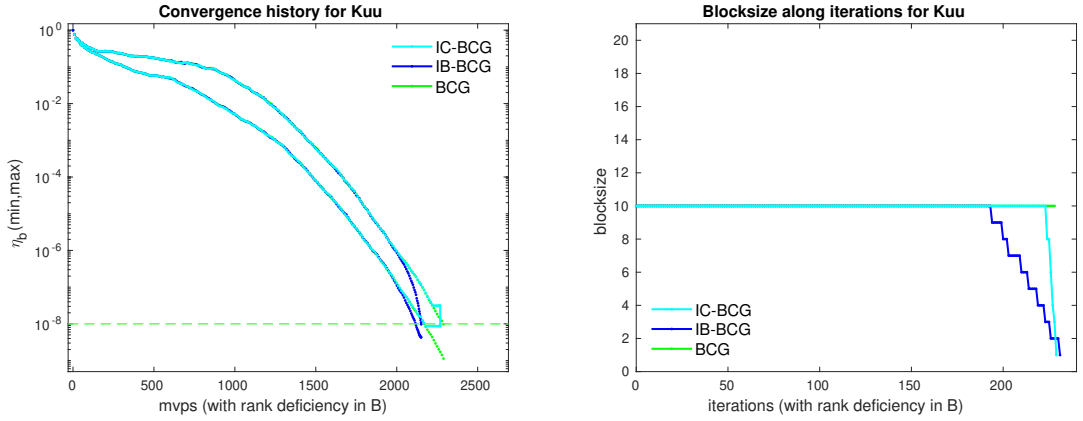


Figure 4: Same case as Figure 2 but the right-hand sides are linearly dependent as $B = [B_{\text{pre}},\ B_{\text{pre}}\ \texttt{randn}(p/2, p/2)]$ with $B_{\text{pre}} = \texttt{randn}(n, p/2)$.

In order to be more exhaustive, we respectively report in Table 5 and Table 6 for the numerical performances of the BCR and BCG variants in terms of $\#mvps$ and $\#iter$ for all matrices listed in Table 1. Due to potential gaps between the true residual and the iterative residual, some right-hand sides may fail to converge with respect to the backward error computed using the true residual norm, even when the stopping criterion is satisfied based on the iterative residual norm. Thus we introduce the notation "$*$" to indicate cases where convergence is achieved according to the iterative residual norm but not with respect to the true residual norm.

Table 5: Numerical results of BCR variants in terms of $\#mvps$ and $\#iter$ for all matrices listed in Table 1 with right-hand sides $B = \texttt{randn}(n, p)$ with $p = 20$ and $\varepsilon = 10^{-8}$ for Section 3.1.

Matrix	$\#mvps$ BCR / IB-BCR / IC-BCR	$\#iter$ BCR / IB-BCR / IC-BCR
apache1	15160 / **13938** / 14758	**758** / 870 / 772
bcsstk15	2700 / **2611** / 2668	135 / 155 / 135
bundle1	700 / **669** / 685	35 / 37 / 35
cbuckle	14460/ **13999** / 14248	**723** / 749 / 727
crankseg_1	6200 / **5535** / 5949	**310** / 333 / 314
crankseg_2	6980 / **6480** / 6821	**349** / 370 / 351
gridgena	9140 / **8798** / 9067	457 / 463 / 457
s1rmq4m1	2760 / **2445** / 2689	**138** / 157 / 139
s1rmt3m1	2800 / **2512** / 2739	**140** / 157 / 141
s1rmq4m1	2760 / **2445** / 2689	**138** / 157 / 139
s1rmt3m1	2800 / **2512** / 2739	**140** / 157 / 141

3.2 Influence of the value of the convergence threshold

In this section, we investigate how the value of convergence threshold affects the performance and robustness of the proposed IB-BCR and IB-BCG variants with partial convergence detecting. The convergence thresholds for η_b are set to be $\varepsilon = 10^{-1}, 10^{-2}, 10^{-5}, 10^{-12}$ and the number of right-hand sides is set to be $p = 20$. With these numerical setting, for illustration purpose, the convergence histories for the solution involving the matrix Kuu are depicted in Figure 5. The general trends are very similar for the various convergence thresholds. We observe that, for the IB variant, all the right-hand sides converge simultaneously to

Table 6: Numerical results of BCG variants in terms of $\#mvps$ and $\#iter$ for all matrices listed in Table 1 with right-hand sides $B = \texttt{randn}(n, p)$ with $p = 20$ and $\varepsilon = 10^{-8}$ for Section 3.1.

Matrix	$\#mvps$ BCG / IB-BCG / IC-BCG	$\#iter$ BCG / IB-BCG / IC-BCG
apache1	15420 / **13969** / 15017	**771** / 837 / 791
bcsstk15	2740 / **2614** / 2693	137 / 153 / 137
bcsstk16	1080 / **993** / 1047	54 / 55 / 54
bcsstk17	13540 / **12085** / 13284	**677** / 740 / 683
bcsstk18	6540 / **6303** / 6481	**327** / 343 / 329
bundle1	720 / **682** / 691	36 / 39 / 36
cbuckle	15080 / **14681** / 14920	**754** / 785 / 758
crankseg_1	6380 / **5714** / 6117	**319** / 337 / 324
crankseg_2	7200 / **6696** / 7055	**360** / 377 / 363
gridgena	9180 / **8846** / 9115	459 / 462 / 459
s1rmq4m1	2800 / **2462** / 2720	140 / 153 / 140
s1rmt3m1	2840 / **2522** / 2764	142 / 150 / 142
s2rmq4m1	4480 / **3964** / 4399	**224** / 240 / 225
s2rmt3m1	5760 / **5190** / 5671	**288** / 318 / 289

the required accuracy, unlike the IC variant. Although the IB variant is more effective in terms of reducing the number of $\#mvps$, both lead to the same solution quality for all the right-hand sides, while BCR and BCG compute solutions with much smaller backward error than required.

A more exhaustive set of numerical results are reported in Table 7-8 in terms of $\#mvps$ and $\#iter$ for other examples. This illustrates that the short-term recurrence induces some residual gaps that cause trouble for stringent convergence thresholds like 10^{-12}. The general trend is that the IB and IC variants minimize $\#mvps$ while the classical BCR and BCG ones minimize $\#iter$.

3.3 Influence of the number of right-hand sides

In this section, we illustrate how the number of the right-hand sides interplays with the performance of the BCR and BCG variants; we vary $p = 5, 10, 30, 40$. The numerical experiments are displayed in Table 9-10. No significant impact on the ranking of the variants can be observed. When solving for a large number of right-hand sides, it can be seen that it is preferable (when it is affordable from a memory view point) to solve all at once rather than dividing them in chunks of smaller number to be solved in sequence. For instance solving for $p = 40$ right-hand sides with the matrix apache1 by IB-BCR does not require 4 times more $\#mvps$ but 2.8 times compared with solving a sequence of 4 block systems with $p = 10$. Corresponding convergence histories are respectively described in the left and right graph of Figure 6, from which it is easy to notice that the larger p, the clearer the gap between the smallest and largest backward errors.

3.4 Experiments with individual convergence threshold

In this section, we illustrate that the partial convergence mechanism can also be adapted to cope with different individual convergence thresholds refer to as "variable accuracy" in the VA variant. Within the BCR or BCG context, it implies to change the $SpaceExpansion$ function described in Equation (11) (Step 11 of Algorithm 3 or Step 11 of Algorithm 4). We illustrate this feature in Figure 7 where we consider the solution for matrix crankseg_1 with $p = 20$ right-hand sides and set the convergence threshold to $\varepsilon = 10^{-4}$

Table 7: Numerical results of BCR variants in terms of $\#mvps$ and $\#iter$ for parts of matrices listed in Table 1 with the right-hand sides $B = \texttt{randn}(n, p)$, $p = 20$, $maxMvps = 5000 \times p$, and various convergence thresholds $\varepsilon = 10^{-1}, 10^{-2}, 10^{-5}, 10^{-12}$ for Section 3.2.

Matrix	ε	10^{-1}		10^{-2}		10^{-5}		10^{-12}	
	Method	$\#mvps$	$\#iter$	$\#mvps$	$\#iter$	$\#mvps$	$\#iter$	$\#mvps$	$\#iter$
apache1	BCR	740	37	7740	**387**	12080	**604**	18960*	**948**
	IB-BCR	**725**	38	**5619**	887	**10544**	740	**17630***	1021
	IC-BCR	726	37	6082	670	11530	624	18575*	959
bcsstk18	BCR	3160	158	4500	**225**	5780	**289**	7140*	357
	IB-BCR	**2025**	212	**3986**	317	**5562**	343	**6903***	367
	IC-BCR	2501	**156**	4183	235	5672	290	7078*	357
cbuckle	BCR	3580	**179**	6960	**348**	11640	**582**	17760*	**888**
	IB-BCR	**3167**	205	**6262**	384	**11036**	606	**17231***	905
	IC-BCR	3362	180	6655	349	11429	585	17608*	890
crankseg_2	BCR	2180	**109**	2880	**144**	4960	**248**	9480*	**474**
	IB-BCR	**1063**	520	**2197**	204	**4468**	270	**9004***	496
	IC-BCR	1143	131	2642	148	4816	251	9344*	476
gridgena	BCR	220	11	5880	**294**	8500	425	9940*	497
	IB-BCR	**193**	11	5821	366	8846	803	**9595***	502
	IC-BCR	195	11	**5582**	303	**8411**	425	9849*	497
Kuu	BCR	300	15	1640	**82**	2560	128	3840	192
	IB-BCR	279	17	**1225**	107	**2280**	136	**3631**	197
	IC-BCR	**278**	15	1425	85	2467	128	3785	192

Table 8: Numerical results of BCG variants in terms of both $\#mvps$ and $\#iter$ for a selection of the of testing matrices listed in Table 1 with the right-hand sides $B = \texttt{randn}(n, p)$, $p = 20$ and $maxMvps = 5000 \times p$ and various convergence thresholds $\varepsilon = 10^{-1}, 10^{-2}, 10^{-5}, 10^{-12}$ for Section 3.2.

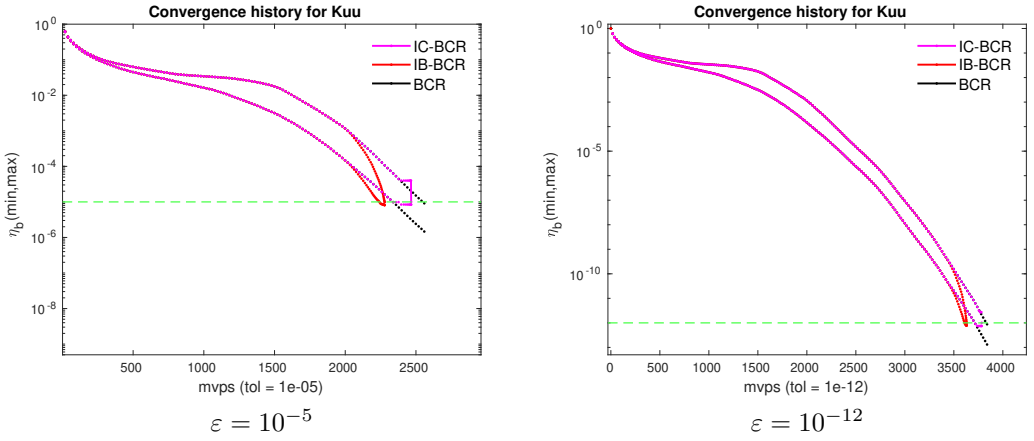
Matrix	ε	10^{-1}		10^{-2}		10^{-5}		10^{-12}	
	Method	$\#mvps$	$\#iter$	$\#mvps$	$\#iter$	$\#mvps$	$\#iter$	$\#mvps$	$\#iter$
apache1	BCG	7000	350	8520	**426**	12340	**617**	19040*	**952**
	IB-BCG	3072	685	**6670**	739	**10752**	718	**17997***	1036
	IC-BCG	**1742**	**123**	7727	454	11788	632	18706*	965
bcsstk18	BCG	4160	**208**	4860	**243**	5940	**297**	7660*	383
	IB-BCG	**3720**	304	**4396**	297	**5728**	329	**7029***	**372**
	IC-BCG	3962	219	4728	248	5850	299	7307*	378
cbuckle	BCG	7220	**361**	8860	**443**	12580	**629**	18380*	**919**
	IB-BCG	**6541**	372	**8253**	456	**12024**	646	**17857***	928
	IC-BCG	6994	363	8683	448	12325	634	18190*	921
crankseg_2	BCG	2040	**102**	3040	**152**	5220	**261**	9700*	**485**
	IB-BCG	**1257**	133	**2503**	180	**4707**	281	**9225***	499
	IC-BCG	1659	110	2830	156	5057	265	9567*	487
gridgena	BCG	6620	331	7740	387	8540	427	10100*	**505**
	IB-BCG	**5112**	354	7799	643	**8182**	433	**9690***	525
	IC-BCG	5761	**316**	**7535**	387	8469	427	9964*	506
Kuu	BCG	1360	**68**	1780	**89**	2620	**131**	3860	**193**
	IB-BCG	**846**	84	**1414**	102	**2345**	138	**3662**	197
	IC-BCG	1087	72	1639	90	2535	132	3810	194

Table 9: Numerical results of BCR variants with different number of right-hand sides ($p = 5, 10, 30, 40$) in terms of $\#mvps$ and $\#iter$ for parts of matrices listed in Table 1 with $B = \texttt{randn}(n, p)$, $maxMvps = 5000 \times p$ and $\varepsilon = 10^{-8}$ for Section 3.3.

Matrix	p	5		10		30		40	
	Method	$\#mvps$	$\#iter$	$\#mvps$	$\#iter$	$\#mvps$	$\#iter$	$\#mvps$	$\#iter$
apache1	BCR	7590	**1518**	10800	**1080**	19080	**636**	22200	**555**
	IB-BCR	**7082**	1641	**9920**	1191	**17022**	752	**19784**	717
	IC-BCR	7532	1520	10538	1091	18435	653	21416	569
bcsstk18	BCR	4200*	**840**	5370*	**537**	7140*	**238**	7760*	**194**
	IB-BCR	**3963***	871	**5213***	569	**6813***	251	**7344***	206
	IC-BCR	4070*	846	5297*	540	7053*	240	7647*	195
cbuckle	BCR	6055	**1211**	10010	**1001**	17640	**588**	20080	**502**
	IB-BCR	**5892**	1236	**9573**	1019	**16924**	618	**19255**	531
	IC-BCR	6000	1212	9808	1003	17491	590	19857	506
crankseg_2	BCR	2765	**553**	4160	**416**	9390	**313**	11760	**294**
	IB-BCR	**2581**	579	**3961**	431	**8737**	337	**10856**	323
	IC-BCR	2674	559	4060	417	9195	316	11538	297
gridgena	BCR	7130	**1426**	8150	815	10080	336	10880	272
	IB-BCR	7213	1548	**8023**	879	**9525**	342	**10203**	279
	IC-BCR	**7104**	1427	8118	815	9967	336	10752	272
Kuu	BCR	1600	320	2260	226	3900	**130**	4440	111
	IB-BCR	**1476**	329	**2097**	235	**3572**	136	**4075**	118
	IC-BCR	1582	320	2224	226	3822	131	4323	111
s2rmt3m1	BCR	3910*	**782**	4690*	**469**	6240*	**208**	6800*	**170**
	IB-BCR	**3785***	868	**4435***	554	**5620***	248	**5987***	209
	IC-BCR	3863*	784	4634*	471	6137*	209	6688*	172
ted_B_unscaled	BCR	135*	27	270*	27	870*	29	1120*	**28**
	IB-BCR	135*	27	270*	27	**851***	29	1144*	29
	IC-BCR	135*	28	270*	28	840*	29	1120*	29

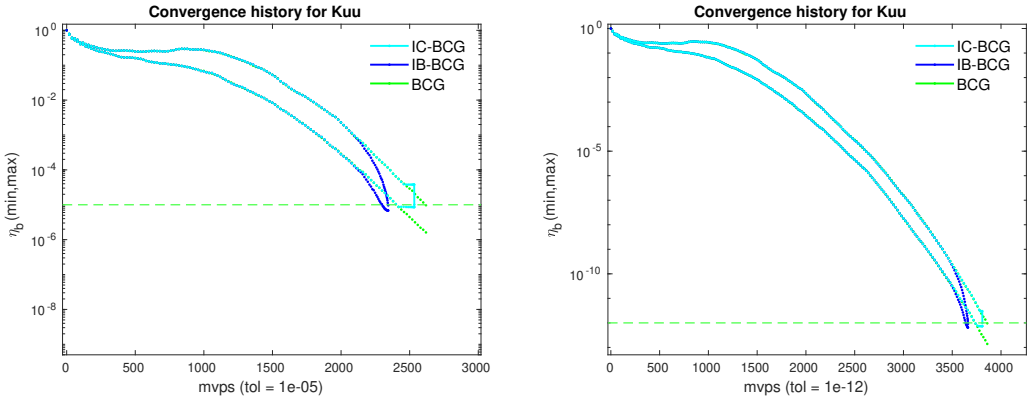
Table 10: Numerical results of BCG variants with different number of right-hand sides, $p = 5, 10, 30, 40$, in terms of $\#mvps$ and $\#iter$ for parts of matrices listed in Table 1 with $B = \texttt{randn}(n, p)$, $maxMvps = 5000 \times p$ and $\varepsilon = 10^{-8}$ for Section 3.3.

Matrix	p	5		10		30		40	
	Method	$\#mvps$	$\#iter$	$\#mvps$	$\#iter$	$\#mvps$	$\#iter$	$\#mvps$	$\#iter$
apache1	BCG	7730	**1546**	10900	**1090**	19440	**648**	22320	**558**
	IB-BCG	**7175**	1639	**10030**	1168	**17277**	756	**20001**	666
	IC-BCG	7667	1547	10667	1103	18694	667	21695	577
bcsstk18	BCG	4395	**879**	5510	**551**	7260	**242**	7880	**197**
	IB-BCG	**4130**	895	**5345**	582	**6959**	254	**7473**	209
	IC-BCG	4260	883	5449	553	7178	243	7773	199
cbuckle	BCG	6525	1305	10570	**1057**	18450	**615**	20920	**523**
	IB-BCG	**6376**	1312	**10209**	1073	**17755**	633	**20130**	547
	IC-BCG	6469	**1299**	10423	1060	18305	617	20683	525
crankseg_2	BCG	2845	**569**	4290	**429**	9780	**326**	12160	**304**
	IB-BCG	**2642**	586	**4088**	437	**9053**	341	**11242**	321
	IC-BCG	2755	576	4188	431	9521	328	11943	308
gridgena	BCG	7395	1479	8180	818	10110	**337**	10960	274
	IB-BCG	**7222**	1493	**7997**	820	**9593**	341	**10266**	278
	IC-BCG	7358	**1478**	8151	818	10023	338	10820	274
Kuu	BCG	1625	**325**	2290	**229**	3960	132	4480	**112**
	IB-BCG	**1517**	335	**2136**	234	**3622**	137	**4115**	117
	IC-BCG	1614	326	2262	230	3860	132	4363	113
s2rmt3m1	BCG	3955	**791**	4750	**475**	6360	**212**	6960	**174**
	IB-BCG	**3772**	821	**4434**	511	**5696**	241	**6092**	201
	IC-BCG	3909	793	4693	477	6259	213	6837	177
ted_B	BCG	135	27	270	27	870	29	1160	29
	IB-BCG	135	27	270	27	864	33	1134	30
	IC-BCG	135	27	270	27	**844**	29	**1128**	29



$\varepsilon = 10^{-5}$ $\varepsilon = 10^{-12}$

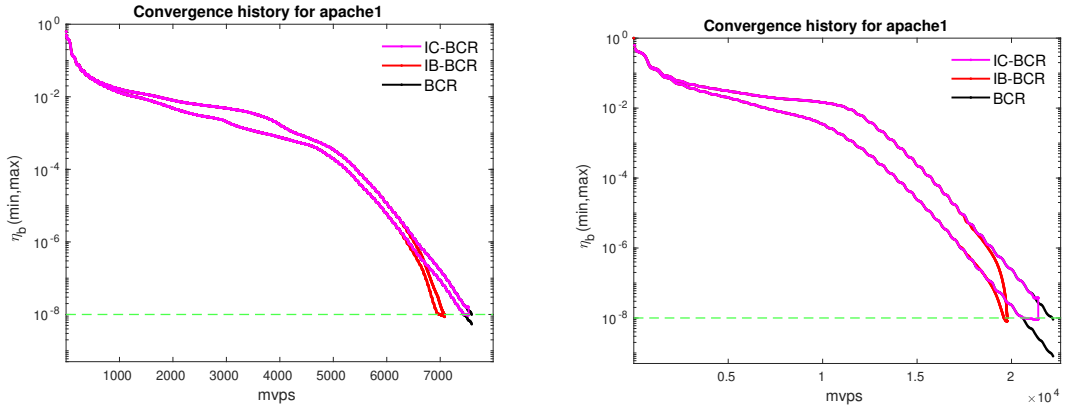
Results of BCR variants



$\varepsilon = 10^{-5}$ $\varepsilon = 10^{-12}$

Results of BCG variants

Figure 5: Convergence history for solving linear systems built by Kuu ($B = \texttt{randn}(n, p)$, $p = 20$ and $maxMvps = 5000 \times p$) with different values of the convergence threshold.



$p = 5$ $p = 40$

Figure 6: Comparison convergence history of the largest/smallest backward errors $\eta_{b^{(i)}}$ of the BCR variants with different number of right-hand sides p by solving matrix apache1 with $B = \text{randn}(n, p)$, $m_d = 5000 \times p$ and $\varepsilon = 10^{-8}$.

for the first $p/2$ right-hand sides and $\varepsilon = 10^{-8}$ for the last $p/2$ ones. It can be seen that numerically it works and enables some saving in terms of $\#mvps$. More numerical results in terms of $\#mvps$ and block $\#iter$

are summarized in Table 11-12, which exhibits a moderate positive benefit of this VA variant that was very effective in the block GCRO context presented in [5, Section 5.3].

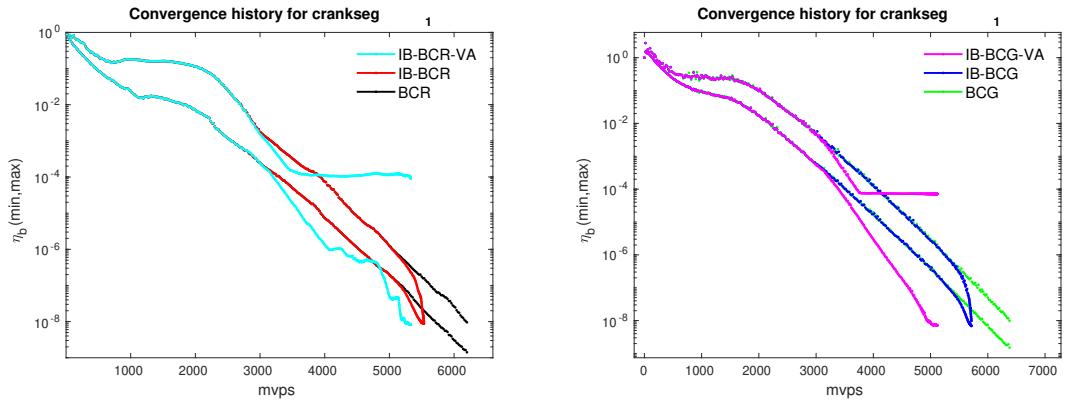


Figure 7: Histories of the largest/smallest backward errors $\eta_{b^{(i)}}$ at each $\#mvps$ for matrix crankseg_1 with convergence threshold equals to 10^{-4} for the first $p/2$ right-hand sides and 10^{-8} for the last $p/2$ ones (here $p = 20$).

Table 11: Numerical results of BCR variants for Section 3.4 in terms of $\#mvps$ and $\#iter$ with $B = \texttt{randn}(n, p)$, $p = 20$ and $maxMvps = 5000 \times p$.

Matrix	$\#mvps$ BCR / IB-BCR / IB-BCR-VA	$\#iter$ BCR / IB-BCR / IB-BCR-VA
cbuckle	14460 / **14018** / 14049	**723** / 749 / 1069
crankseg_1	6200 / 5554 / **5344**	**310** / 333 / 482
crankseg_2	6980 / **6499** / 6943	**349** / 369 / 845
Kuu	3200 / **2963** / 3609	**160** /166 / 423
shallow_water2	560 / 560 / **430**	28 / 28 / 28
ted_B_unscaled	540* / 540* / **415***	27 / 27 / 27

Table 12: Numerical results of BCG variants for Section 3.4 in terms of $\#mvps$ and $\#iter$ with $B = \texttt{randn}(n, p)$, $p = 20$ and $maxMvps = 5000 \times p$.

Matrix	$\#mvps$ BCG / IB-BCG / IB-BCG-VA	$\#iter$ BCG / IB-BCG / IB-BCG-VA
cbuckle	15080 / 14681 / **14145**	**754** / 785 / 980
crankseg_1	6380 / 5714 / **5124**	**319** / 337 / 482
crankseg_2	7200 / 6696 / **6183**	**360** / 377 / 548
Kuu	3240 / **2987** / 3100	**162** / 166 / 326
shallow_water2	560 / 560 / **420**	28 / 28 / 28
ted_B_unscaled	560 / 543 / **411**	28 / 28 / 28

3.5 Experiments with symmetric matrices

In this section, we consider testing three symmetric but not positive definite matrices described in Table 2, with linearly independent right-hand sides defined as $B = \texttt{randn}(n, p)$ with $p = 20$, $m_d = 5000 \times p$, $\varepsilon = 10^{-8}$, and no preconditioner is applied. The corresponding convergence history and numerical results

are respectively reported in Figure 8 and Table 13. The observations are very similar to what we have seen for symmetric positive definite matrices in the previous sections, that is IB-BCR is often the best in minimizing the number of $\#mvps$ at a possible extra cost of a few more block $\#iter$.

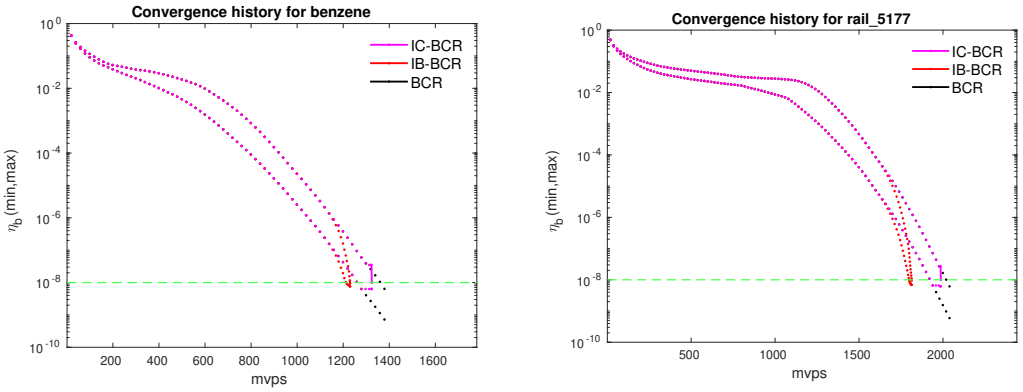


Figure 8: Comparison convergence history of the largest/smallest backward errors $\eta_{b^{(i)}}$ at each $\#mvps$ of the BCR variants by solving symmetry but not positive definite matrices listed in Table 2 with the right-hand sides $B = \texttt{randn}(n, p)$, $p = 20$, $maxMvps = 5000 \times p$ and $\varepsilon = 10^{-8}$.

Table 13: Numerical results of the BCR variants in terms of $\#mvps$ and $\#iter$ for matrices listed in Table 2 with $B = \texttt{randn}(n, p)$, $p = 20$ and $maxMvps = 5000 \times p$ for Section 3.5.

Matrix	Method	$\#mvps$	$\#iter$
benzene	BCR	1380	69
	IB-BCR	**1231**	70
	IC-BCR	1325	69
rail_5177	BCR	2040	102
	IB-BCR	**1815**	107
	IC-BCR	1988	102
saylr4	BCR	380	19
	IB-BCR	**325**	19
	IC-BCR	371	19

4 Concluding remarks

In this work, we propose new variants of the block conjugate residual (BCR) method, incorporating mechanisms to address the partial convergence issues. These variants are denoted IB-BCR, and a cost-free alternative, IC-BCR. The IB-BCR and IC-BCR methods ensure the robustness by incorporating mechanisms to detect and handle partial convergence effectively. These enhancements improve the computational efficiency and reliability of the BCR variants in handling symmetric linear systems with multiple right-hand sides. We also explore the possibility of applying partial convergence detection, originally developed in the minimum residual norm context, to the block conjugate gradient (BCG) algorithm, in order to develop a heuristic solver called IB-BCG for symmetric positive definite linear systems. This numerical mechanism is considered heuristic because the resulting policy for expanding the search space still relies on residual norm minimization, while ideally, it should be based on minimizing the A-norm error. However, A-norm error directions are not a natural by-product of BCG, unlike the residuals in the case of BCR. The general observed trend is that the IB-BCR or IB-BCG variants perform the best in terms of the number of matrix-vector products, while the standard BCR or BCG methods excel in terms of block iterations. The best variant in terms of time depends on the trade-off between the final matrix-vector products, block iterations, and the additional cost associated with implementing the breakdown and partial convergence detecting mechanisms.

For long-term recurrence algorithms based on the Arnoldi basis, as discussed in [5], a sufficiently accurate orthonormal basis can be generated, ensuring that the true residual norm closely aligns with the least squares residual norm. This alignment enables monitoring of the true convergence behavior using the iterated residual norm. In contrast, for short-term recurrence algorithms such as BCR and BCG, residual gaps can occur. These gaps sometimes hinder the convergence of the true backward error, based on the true residual norm, even when the iterated residual norm indicates convergence. Thus, the partial convergence detecting mechanism applied to the iterated residual may become less reliable in short-term recurrence cases, especially when residual gaps occur. A potential remedy to address this issue is extending the rounding error analysis from [20] to the block case. Such an extension could provide a framework for estimating the block residual gap. Furthermore, it might facilitate the development of corresponding replacement techniques, as explored in [3] and [2], to improve the robustness of these short-term recurrence algorithms. Those topics might be the core of future research.

References

[1] E. Agullo, L. Giraud, and Y.-F. Jing. “Block GMRES Method with Inexact Breakdowns and Deflated Restarting”. In: *SIAM J. Matrix Anal. Appl.* 35.4 (2014), pp. 1625–1651. DOI: 10.1137/140961912.

[2] E. Carson and J. Demmel. “A Residual Replacement Strategy for Improving the Maximum Attainable Accuracy of s-Step Krylov Subspace Methods”. In: *SIAM J. Matrix Anal. Appl.* 35.1 (2014), pp. 22–43. DOI: 10.1137/120893057.

[3] S. Cools, E. F. Yetkin, E. Agullo, L. Giraud, and W. Vanroose. “Analyzing the Effect of Local Rounding Error Propagation on the Maximal Attainable Accuracy of the Pipelined Conjugate Gradient Method”. In: *SIAM J. Matrix Anal. Appl.* 39.1 (2018), pp. 426–450. DOI: 10.1137/17M1117872.

[4] T. A. Davis and Y. Hu. “The University of Florida sparse matrix collection”. In: *ACM Trans. Math. Softw.* 38 (2011), p. 1.

[5] L. Giraud, Y.-F. Jing, and Y.-F. Xiang. “A Block Minimum Residual Norm Subspace Solver with Partial Convergence Management for Sequences of Linear Systems”. In: *SIAM J. Matrix Anal. Appl.* 43.2 (2022), pp. 710–739. DOI: 10.1137/21M1401127.

[6] M. Hestenes. “The conjugate gradient method for solving linear systems”. In: *Numerical Analysis*. Ed. by J. H. Curtiss. Providence Rhode Island: American Mathematical Society, 1956, pp. 83–102.

[7] M. Hestenes and E. Stiefel. “Methods of Conjugate Gradients for Solving Linear Systems”. In: *Journal of Research of the National Bureau of Standards* 49.6 (1952), pp. 409–436.

[8] H. Ji and Y.-H. Li. “A breakdown-free block conjugate gradient method”. In: *BIT* 57 (2017), pp. 379–403. DOI: 10.1007/s10543-016-0631-z.

[9] J. Langou. *Iterative Methods for Solving Linear Systems with Multiple Right-Hand Sides*. Ph.D. dissertation TH/PA/03/24. CERFACS, Toulouse, France, 2003.

[10] M. Levonyak, C. Pacher, and W. N. Gansterer. "Scalable Resilience Against Node Failures for Communication-Hiding Preconditioned Conjugate Gradient and Conjugate Residual Methods". In: *Proceedings of the 2020 SIAM Conference on Parallel Processing for Scientific Computing*. 2020, pp. 81–92. DOI: 10.1137/1.9781611976137.

[11] D. Luenberger. "The Conjugate Residual Method for Constrained Minimization Problems". In: *SIAM J. Numer. Anal.* 7.3 (1970), pp. 390–398. DOI: 10.1137/070703.

[12] D. P. O'Leary. "The block conjugate gradient algorithm and related methods". In: *Linear Algebra Appl.* 29 (1980), pp. 293–322. DOI: 10.1016/0024-3795(80)90247-5.

[13] L. F. Pavarino. "Preconditioned conjugate residual methods for mixed spectral discretizations of elasticity and Stokes problems". In: *Comput. Methods Appl. Mech. Engrg.* 146.1-2 (1997), pp. 19–30. DOI: 10.1016/S0045-7825(96)01224-8.

[14] M. Robbé and M. Sadkane. "Exact and inexact breakdowns in the block GMRES method". In: *Linear Algebra Appl.* 419.1 (2006), pp. 265–285. DOI: 10.1016/j.laa.2006.04.018.

[15] Y. Saad. *Iterative Methods for Sparse Linear Systems, 2nd ed.* Philadelphia: SIAM, 2003.

[16] Y. Saad and M. H. Schultz. "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems". In: *SIAM J. Sci. Stat. Comput.* 7.3 (1986), pp. 856–869. DOI: 10.1137/0907058.

[17] T. Sogabe. *Extensions of the Conjugate Residual Method*. Ph.D. dissertation. http://www.ist.aichi-pu.ac.jp/person/sogabe/thesis.pdf. University of Tokyo, Japan, 2003.

[18] T. Sogabe, M. Sugihara, and S. L. Zhang. "An extension of the conjugate residual method to nonsymmetric linear systems". In: *Journal of Computational and Applied Mathematics* 226.1 (2009), pp. 103–113. DOI: 10.1016/j.cam.2008.05.018.

[19] E. de Sturler. "Nested Krylov methods based on GCR". In: *J. Comput. Appl. Math.* 67.1 (1996), pp. 15–41. DOI: 10.1016/0377-0427(94)00123-5.

[20] H. A. van der Vorst and Q. Ye. "Residual Replacement Strategies for Krylov Subspace Iterative Methods for the Convergence of True Residuals". In: *SIAM J. Sci. Comput.* 22.3 (2000), pp. 835–852. DOI: 10.1137/S1064827599353865.

RESEARCH CENTRE
BORDEAUX – SUD-OUEST

200 avenue de la Vieille Tour
33405 Talence Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399