Numerical Algorithms
https://doi.org/10.1007/s11075-025-02038-4

RESEARCH

Block CG algorithms revisited

Petr Tichý[1] · Gérard Meurant[2] · Dorota Šimonová[1]

Received: 7 November 2024 / Accepted: 27 February 2025

Abstract
Our goal in this paper is to clarify the relationship between the block Lanczos and the block conjugate gradient (BCG) algorithms. Under the full rank assumption for the block vectors, we show the one-to-one correspondence between the algorithms. This allows, for example, the computation of the block Lanczos coefficients in BCG. The availability of block Jacobi matrices in BCG opens the door for further development, e.g., for error estimation in BCG based on (modified) block Gauss quadrature rules. Driven by the need to get a practical variant of the BCG algorithm well suited for computations in finite precision arithmetic, we also discuss some important variants of BCG due to Dubrulle. These variants avoid the troubles with a possible rank deficiency within the block vectors. We show how to incorporate preconditioning and computation of Lanczos coefficients into these variants. We hope to help clarify which variant of the block conjugate gradient algorithm should be used for computations in finite precision arithmetic. Numerical results illustrate the performance of different variants of BCG on some examples.

Keywords Conjugate gradients · Block methods · Multiple right-hand sides

Mathematics Subject Classification (2010) 65F10 · 65G50

This paper is dedicated to Michela Redivo-Zaglia on the occasion of her retirement and to Hassane Sadok on the occasion of his 65th birthday.

✉ Petr Tichý
petr.tichy@mff.cuni.cz

Gérard Meurant
gerard.meurant@gmail.com

Dorota Šimonová
simonova@karlin.mff.cuni.cz

[1] Faculty of Mathematics and Physics, Charles University, Sokolovská 83, Prague 18675, Czech Republic

[2] Paris, France

1 Introduction

Consider the problem of solving several systems of linear algebraic equations

$$Ax^{(i)} = b^{(i)}, \quad i = 1, \ldots, m,$$

where $A \in \mathbb{R}^{n \times n}$ is symmetric and positive definite, $m \geq 1$, and $b^{(i)} \in \mathbb{R}^n$, $i = 1, \ldots, m$. Denoting $b = [b^{(1)}, \ldots, b^{(m)}]$ and $x = [x^{(1)}, \ldots, x^{(m)}]$, we want to solve the system $Ax = b$ with a block right-hand side $b \in \mathbb{R}^{n \times m}$. If A is large and sparse, then it is natural to apply block Krylov subspace methods, in particular the (preconditioned) block conjugate gradient (BCG) method. In block methods, A is applied to a block of vectors rather than just a single vector at each iteration. This operation is used to construct a common subspace for all right-hand sides, producing a richer search subspace. From a computational perspective, block methods have the potential to benefit from the capabilities of modern computers. The block-level operations have been shown to be efficient in a high-performance computing environment. For a comprehensive overview and summary of block Krylov subspace methods, see [1].

Several versions of BCG (and related methods) were introduced by O'Leary in [2]. Note that the proposed algorithms also include auxiliary nonsingular block coefficients which can be freely chosen. They can be used as scaling factors. These scaling factors will play an important role later in our discussion. In [2] it was observed that the block algorithms can exhibit numerical instability when the vectors within a block become (nearly) linearly dependent. Rank deficiency can lead to inaccurate computation of (ill-conditioned) block coefficients, and consequently can slow down convergence and reduce the maximum attainable accuracy. In rank deficient cases, it was suggested to monitor the rank of the blocks and to reduce the block size if necessary. Nevertheless, the algorithmic details were not given in [2].

Several attempts have been made to improve BCG for solving practical problems. In general, approximate solutions for different right-hand sides cannot be expected to converge similarly. In [3] the authors suggest that the converged approximations can be removed, and one could continue the BCG algorithm only with unconverged ones. However, a technical realization of this approach is not present in the given paper. This improvement does not completely solve the rank deficiency problem. Nikishin and Yeremin [4] addressed the problem of monitoring near rank deficiency and presented a variable block CG method, which is used to speed-up the solution of single right-hand side systems.

An alternative approach to address the issue of singular or nearly singular matrices in BCG can be found in a paper by Dubrulle [5]. The approach suggests maintaining the block size and ensuring that the blocks are well conditioned (the vectors within a block remain sufficiently well linearly independent). This guarantees that the block coefficients are well-defined and can be computed in a numerically stable manner. The approach presented by Dubrulle has two main ingredients. First, the BCG algorithms of O'Leary can be reformulated by a convenient choice of the scaling factors. Second, the basis (related to a direction block or to a residual block) can be changed in order to implicitly eliminate the effects of rank deficiencies. For example, a QR factorization of a rank-deficient block $v \in \mathbb{R}^{n\times m}$ can be computed such that the Q-factor of size $n \times m$ has full column rank, and the upper triangular R-factor of size $m \times m$ is singular. Such a QR factorization can be obtained using the MATLAB command

$$[\mathtt{w}, \sigma] = \mathtt{qr}(\mathtt{v}, \text{"}\mathtt{econ}\text{"}).$$

Since this command relies on Householder reflections, we will refer to it as the Householder QR from now on. The resulting block $w \in \mathbb{R}^{n\times m}$ has always orthonormal columns. If v is rank deficient, then the upper triangular matrix $\sigma \in \mathbb{R}^{n\times m}$ is singular. However, the algorithm can be formulated such that it uses only the block w and not the inverse of the coefficient matrix σ. Then the inverses of the corresponding matrices $w^T w$ and $w^T Aw$, which are used to compute the block coefficients, are always well defined. One can observe that in the rank deficient case, some columns of w are artificially created. However, these spurious vectors do not cause any harm in the block CG process. In the following, we will refer to the process of creating a well conditioned block w as a regularization of the block v. Note that the span of columns of w contains the span of columns of v. In BCG, one can regularize either the direction blocks or the residual blocks. For the purpose of regularization, one may use a QR factorization or an LU factorization with pivoting, as discussed by Dubrulle. It is even possible to ensure A-orthogonality within the regularized direction block, without any additional multiplication with A. The two most promising Dubrulle variants are discussed in Section 5.

An idea analogous to that of Dubrulle was presented by Ji and Li in [6]. The algorithm proposed in [6] is referred to as a breakdown-free block CG. Similarly to [5], Ji and Li first reformulate BCG using a convenient choice of scaling factors and then focus on the change of basis for the direction block. In particular, an orthonormal basis for the direction block is computed. In case of a (near) rank deficiency, the corresponding vectors within the direction block are removed. In summary, while Dubrulle preserves the block sizes by incorporating spurious direction vectors, Ji and Li [6] eliminate the (nearly) linearly dependent vectors from the direction block. As will be demonstrated by our numerical experiments, the removal of vectors is not always an optimal approach. In finite precision computations, the algorithm proposed by Ji and Li typically exhibits inferior performance compared to the algorithms proposed by Dubrulle.

Birk and Frommer [7] present a deflated shifted block CG method that effectively integrates several techniques. The method is based on the Hermitian version of the non-symmetric block Lanczos algorithm proposed in [8], which includes deflation. Deflation here means the process of detecting and removing linearly dependent vectors in the block Krylov sequences. Due to the shifts, using the block Lanczos algorithm as the engine for BCG is a natural choice, as the basis of the underlying block Krylov subspace only needs to be generated once. Note that the resulting algorithm, which incorporates deflation techniques, can also be used to solve a single block system $Ax = b$. Thus, it may be regarded as a variant of BCG, which addresses the problem of rank deficiency.

This paper has two goals. The first goal is to clarify the relationship between blocks and coefficient matrices that appear in the block CG and block Lanczos algorithms. In particular, assuming exact arithmetic and full rank of the corresponding block Krylov subspaces, we will show the relation between the BCG residual blocks and the Lanczos blocks. We also show how to compute in BCG the block Jacobi matrices, defined by the block Lanczos coefficients. This clarification opens the door for further developments, e.g., for error estimation based on modifications of block Jacobi matrices; see our forthcoming paper [9] and the books [10] and [11] for an overview.

The second goal is to formulate a variant of preconditioned BCG that is suitable for practical computation in finite precision arithmetic. In particular, we would like to have a version that avoids problems with rank deficiency and that potentially allows to monitor the A-norm of the error for each system and to remove a system if an approximate solution is accurate enough. At the same time, we want to reconstruct in such a practical version of BCG the underlying block Jacobi matrices.

The paper is organized as follows. In Section 2 we recall the classical vector versions of the Lanczos and CG algorithms, and their relationship. Block versions of these algorithms are discussed in Section 3. Assuming exact arithmetic and full dimension of the block Krylov subspace, the connection between the block Lanczos and CG algorithms is investigated in Section 4. In Section 5 we formulate block CG variants which are suitable for practical computation and in Section 6 we discuss some practical issues like preconditioning and stopping criteria. Finally, in Section 7 we present numerical experiments justifying our considerations.

2 Lanczos and CG

Let us first discuss the case $m = 1$, i.e., the classical vector case. Consider a sequence of nested Krylov subspaces

$$\mathcal{K}_k(A, v) \equiv \operatorname{span}\{v, Av, \dots, A^{k-1}v\}, \qquad k = 1, 2, \dots, \tag{1}$$

where $v \in \mathbb{R}^n$ is a given starting vector and $A \in \mathbb{R}^{n\times n}$ is symmetric. The dimension of those Krylov subspaces is increasing up to an index d called the *grade of v with respect to A*, at which the maximal dimension is attained, and $\mathcal{K}_d(A, v)$ is invariant under multiplication with A.

Algorithm 1 Lanczos algorithm.

```
1: input $A, v$
2: $v_0 = 0$
3: $\beta_1 v_1 = v$
4: for $k = 1, \dots$ do
5:     $w = Av_k - \beta_k v_{k-1}$
6:     $\alpha_k = v_k^T w$
7:     $w = w - \alpha_k v_k$
8:     $\beta_{k+1} v_{k+1} = w$
9: end for
```

Assuming that $k < d$, the Lanczos algorithm (Algorithm 1) computes a sequence of *Lanczos vectors* $v_1, \dots, v_{k+1}$ which form an orthonormal basis of $\mathcal{K}_{k+1}(A, v)$. Note that the notation $\beta_{k+1}v_{k+1} = w$ used on lines 3 and 8 of Algorithm 1 means that v_{k+1}

is obtained by normalization of the given vector w with the positive normalization coefficient β_{k+1}. In other words, $\beta_{k+1} = \|w\|$ and $v_{k+1} = w/\beta_{k+1}$. The Lanczos vectors satisfy a three-term recurrence

$$\beta_{j+1}v_{j+1} = Av_j - \alpha_j v_j - \beta_j v_{j-1}, \qquad j = 1, \ldots, k, \tag{2}$$

which can be written in matrix form as

$$AV_k = V_k T_k + \beta_{k+1} v_{k+1} e_k^T,$$

where $V_k = [v_1, \ldots, v_k]$, the vector e_k denotes the kth column of the identity matrix of an appropriate size (here of size k), and

$$T_k = \begin{bmatrix} \alpha_1 & \beta_2 & & \\ \beta_2 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_k \\ & & \beta_k & \alpha_k \end{bmatrix} \tag{3}$$

is the k by k symmetric tridiagonal matrix of the Lanczos coefficients computed in Algorithm 1. The coefficients β_j being positive, T_k is a *Jacobi matrix*. Note that if A is positive definite, then $T_k = V_k^T AV_k$ is positive definite as well.

Algorithm 2 Conjugate gradients.

```
1: **input** $A, b, x_0$
2: $r_0 = b - Ax_0$
3: $p_0 = r_0$
4: **for** $k = 1, \ldots$ until convergence **do**
5:     $\gamma_{k-1} = \frac{r_{k-1}^T r_{k-1}}{p_{k-1}^T A p_{k-1}}$
6:     $x_k = x_{k-1} + \gamma_{k-1} p_{k-1}$
7:     $r_k = r_{k-1} - \gamma_{k-1} A p_{k-1}$
8:     $\delta_k = \frac{r_k^T r_k}{r_{k-1}^T r_{k-1}}$
9:     $p_k = r_k + \delta_k p_{k-1}$
10: **end for**
```

When solving a system of linear algebraic equations $Ax = b$ with symmetric and positive definite matrix A, the standard Hestenes and Stiefel version of the CG method (Algorithm 2) can be used. It is well known that in exact arithmetic, the CG iterates x_k minimize the A-norm of the error over the manifold $x_0 + \mathcal{K}_k(A, r_0)$,

$$\|x - x_k\|_A = \min_{y \in x_0 + \mathcal{K}_k(A, r_0)} \|x - y\|_A,$$

the residual vectors $r_j = b - Ax_j$, $j = 0, \ldots, k$, are mutually orthogonal, and the direction vectors p_j, $j = 0, \ldots, k$, are A-orthogonal (conjugate). Moreover,

$$\operatorname{span}\{r_0, \ldots, r_{k-1}\} = \operatorname{span}\{p_0, \ldots, p_{k-1}\} = \mathcal{K}_k(A, r_0).$$

In exact arithmetic, if $r_k = 0$ or $p_k = 0$, then the algorithm has found the solution and the maximum dimension d of the corresponding Krylov subspace has been reached. The coefficients γ_k and δ_{k+1} are strictly positive since A is positive definite.

If the Lanczos algorithm is applied to the symmetric and positive definite A and $v = r_0$, then the CG residual vectors $r_j = b - Ax_j$ are collinear with the Lanczos vectors v_{j+1},

$$v_{j+1} = (-1)^j \frac{r_j}{\|r_j\|}, \qquad j = 0, \ldots, k. \tag{4}$$

Thanks to this close relationship it can be shown (see, for instance [12]) that the recurrence coefficients of both algorithms are linked through

$$\beta_{k+1} = \frac{\sqrt{\delta_k}}{\gamma_{k-1}}, \quad \alpha_k = \frac{1}{\gamma_{k-1}} + \frac{\delta_{k-1}}{\gamma_{k-2}}, \quad \delta_0 = 0, \quad \gamma_{-1} = 1.$$

Writing these formulas in a matrix form, we get

$$T_k = L_k D_k L_k^T, \tag{5}$$

where L_k is unit lower bidiagonal and D_k is diagonal,

$$L_k = \begin{bmatrix} 1 & & & \\ \ell_1 & \ddots & & \\ & \ddots & \ddots & \\ & & \ell_{k-1} & 1 \end{bmatrix}, \quad D_k = \begin{bmatrix} d_1 & & & \\ & \ddots & & \\ & & \ddots & \\ & & & d_k \end{bmatrix}$$

with

$$\ell_j = \sqrt{\delta_j}, \quad j = 1, \ldots, k-1, \qquad d_j = \gamma_{j-1}^{-1}, \quad j = 1, \ldots, k.$$

In other words, CG implicitly computes the Cholesky factorization of the tridiagonal Jacobi matrix T_k from the Lanczos algorithm. Hence, the Lanczos-related quantities can be reconstructed in CG and vice versa. More precisely, the CG approximations x_k can be computed from the matrix V_k of the Lanczos vectors and the Lanczos coefficient matrix T_k via

$$x_k = x_0 + V_k y_k, \qquad T_k y_k = \beta_1 e_1. \tag{6}$$

Clearly, this above-mentioned version of CG is not very efficient since it requires storing the matrices V_k and T_k. However, a more efficient algorithm that implements (6) can be derived; see, e.g., [13, 14].

The relations discussed above show a one-to-one correspondence between the Lanczos and CG algorithms. In this paper we want to formulate analogous relations for the block versions of both algorithms.

3 Block Lanczos and block CG

In this section, we introduce analogues of the Lanczos and CG algorithms that work with blocks of vectors of size $n \times m$ instead of single vectors. Given a starting block

$v \in \mathbb{R}^{n\times m}$ and a matrix $A \in \mathbb{R}^{n\times n}$, we can define the corresponding generalization of the kth Krylov subspace (1) as

$$\mathcal{K}_k(A, v) \equiv \text{colspan}\{v, Av, \dots, A^{k-1}v\},$$

and we will call it the kth *block Krylov subspace* generated by A and v. Here colspan denotes the span of the columns of all blocks $v, Av, \dots, A^{k-1}v$. From the definition it is clear that

$$\mathcal{K}_k(A, v) = \mathcal{K}_k(A, v^{(1)}) + \mathcal{K}_k(A, v^{(2)}) + \cdots + \mathcal{K}_k(A, v^{(m)}),$$

where $v^{(i)}$ denotes the ith column of v, $i = 1, \dots, m$. In the following, we will assume for simplicity that $\mathcal{K}_k(A, v)$ has full dimension, i.e., that

$$\dim \mathcal{K}_k(A, v) = km \le n.$$

The case of rank deficiency will be discussed later in Section 5. To denote the blocks of vectors and the block coefficients, we will use the same notation as before, so at first glance the algorithms look almost the same. The only difference will be in the size of the objects we are working with. Instead of vectors of size $n \times 1$ we will work with blocks of size $n \times m$, and instead of scalar coefficients we will work with $m \times m$ coefficient matrices denoted by the same letters as before. The justification for using the same notation is that the algorithms that work with vectors can be seen as a special case of block algorithms with $m = 1$.

3.1 Block Lanczos algorithm

Suppose we are given a starting block $v \in \mathbb{R}^{n\times m}$ and a *symmetric* matrix $A \in \mathbb{R}^{n\times n}$. Then one can compute the orthonormal basis of the kth block Krylov subspace $\mathcal{K}_k(A, v)$ of dimension km using the block Lanczos algorithm (Algorithm 3) introduced in [15] and [16]. For practical efficiency, we use the modified Gram-Schmidt version of this algorithm.

Algorithm 3 Block Lanczos.

```
1: input $A, v$
2: $v_0 = 0$
3: $v_1\beta_1 = v$
4: for $k = 1, 2, \dots$ do
5:     $w = Av_k - v_{k-1}\beta_k^T$
6:     $\alpha_k = v_k^T w$
7:     $w = w - v_k\alpha_k$
8:     $v_{k+1}\beta_{k+1} = w$
9: end for
```

Algorithm 3 looks formally the same as Algorithm 1, but now v, v_k, and w are blocks of size $n \times m$, and α_k and β_k are coefficient matrices of size $m \times m$. For simplicity, we will call the coefficient matrices α_k and β_k coefficients again. We use the notation $v_{k+1}\beta_{k+1} = w$ with the meaning that β_{k+1} is a normalization coefficient such that $v_{k+1}^T v_{k+1} = I$, where I is the $m \times m$ identity matrix. In other words, β_{k+1} is chosen such that

$$\beta_{k+1}^{-T} w^T w \beta_{k+1}^{-1} = I, \quad \text{or, equivalently,} \quad w^T w = \beta_{k+1}^T \beta_{k+1}. \tag{7}$$

It can be shown, see, e.g., [16], that the blocks v_k produced by Algorithm 3 satisfy

$$v_i^T v_j = \delta_{i,j} I, \qquad i, j = 1, \ldots, k+1,$$

where $\delta_{i,j}$ denotes the Kronecker delta. The coefficients α_k are given by

$$\alpha_k = v_k^T w = v_k^T \left(A v_k - v_{k-1} \beta_k^T \right) = v_k^T A v_k.$$

Note that if we compute α_k using $\alpha_k = v_k^T A v_k$, we obtain the classical Gram-Schmidt version of the block Lanczos algorithm, while Algorithm 3 corresponds to the modified Gram-Schmidt version. Both versions are mathematically equivalent, but the modified Gram-Schmidt version better preserves orthogonality during finite precision computations. Assuming that no breakdown occurs due to rank deficiency, we obtain after k iterations

$$A V_k = V_k T_k + v_{k+1} \beta_{k+1} e_k^T, \tag{8}$$

where $V_k = [v_1, v_2, \ldots, v_k] \in \mathbb{R}^{n \times km}$, $e_k^T = [0, \ldots, 0, I] \in \mathbb{R}^{m \times km}$, and

$$T_k = \begin{bmatrix} \alpha_1 & \beta_2^T & & \\ \beta_2 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_k^T \\ & & \beta_k & \alpha_k \end{bmatrix} \tag{9}$$

is symmetric block tridiagonal with $m \times m$ blocks.

Concerning a particular choice of the normalization coefficients β_{k+1} satisfying (7), the standard way introduced by Golub and Underwood [16] is to define β_{k+1} as the R-factor of the QR factorization of w. If β_{k+1} is upper triangular, then the resulting block tridiagonal matrix T_k has semi-bandwidth m, so that the block Lanczos algorithm is equivalent to the band Lanczos algorithm; see [17]. Note that to define β_{k+1} satisfying (7), one could also use the more expensive singular value decomposition or the polar decomposition of w. Using the polar decomposition, the coefficient matrices β_{k+1} would be symmetric and positive definite.

In the following we will consider Algorithm 3 with the standard choice of β_{k+1}, that is, computed using a QR factorization. To define β_{k+1} uniquely and to be consistent with the case $m = 1$, we will consider β_{k+1} with positive diagonal entries. Then, β_{k+1}^T can also be seen as the L-factor of the Cholesky factorization of $w^T w$.

3.2 Block CG algorithm

Consider now a system of linear algebraic equations with multiple right-hand sides

$$Ax = b,$$

where $A \in \mathbb{R}^{n\times n}$ is *symmetric and positive definite*, and $b \in \mathbb{R}^{n\times m}$. Let $x_0 \in \mathbb{R}^{n\times m}$ be an initial approximate solution. Then one can compute approximate solutions using the block conjugate gradient (BCG) method that was introduced by O'Leary [2]; see Algorithm 4.

Algorithm 4 O'Leary Block CG (OL-BCG).

```
1: input $A, b, x_0$
2: $r_0 = b - Ax_0$
3: $p_0 = r_0\phi_0$
4: for $k = 1, 2, \dots$ do
5:     $\gamma_{k-1} = \left(p_{k-1}^T A p_{k-1}\right)^{-1} \phi_{k-1}^T r_{k-1}^T r_{k-1}$
6:     $x_k = x_{k-1} + p_{k-1}\gamma_{k-1}$
7:     $r_k = r_{k-1} - Ap_{k-1}\gamma_{k-1}$
8:     $\delta_k = \phi_{k-1}^{-1}\left(r_{k-1}^T r_{k-1}\right)^{-1} r_k^T r_k$
9:     $p_k = \left(r_k + p_{k-1}\delta_k\right)\phi_k$
10: end for
```

In this algorithm, the *nonsingular* coefficient matrices $\phi_k \in \mathbb{R}^{m\times m}$ are free to choose. They play the role of scaling coefficient matrices for the direction blocks p_k, see line 9 of Algorithm 4. Assuming again that the corresponding block Krylov subspace $\mathcal{K}_k(A, r_0)$ has full dimension km, there is no breakdown in block CG and it holds that

$$\mathcal{K}_k(A, r_0) = \text{colspan}\{r_0, r_1, \dots, r_{k-1}\} = \text{colspan}\{p_0, p_1, \dots, p_{k-1}\}.$$

Denoting by $x_k^{(i)}$, $r_k^{(i)}$, $p_k^{(i)}$, $i = 1, \dots, m$, the columns of x_k, r_k, p_k respectively, it holds that

$$x_k^{(i)} \in x_0^{(i)} + \mathcal{K}_k(A, r_0), \quad r_k^{(i)} \perp \mathcal{K}_k(A, r_0), \quad p_k^{(i)} \perp_A \mathcal{K}_k(A, r_0) = \mathcal{K}_k(A, p_0).$$

As a consequence, $x_k^{(i)}$ minimizes

$$\left(y - x^{(i)}\right)^T A \left(y - x^{(i)}\right) = \|y - x^{(i)}\|_A^2$$

over all vectors $y \in x_0^{(i)} + \mathcal{K}_k(A, r_0)$; see [2, p. 301, Theorem 2].

The classical choice $\phi_k = I$ leads to the Hestenes and Stiefel version of BCG (HS-BCG). However, one can also choose ϕ_k differently, e.g., as the inverse of the R-factor from the QR factorization of $r_k + p_{k-1}\delta_k$ so that p_k has orthonormal columns. The choice of ϕ_k can be used, e.g., to derive other BCG versions, see [2, 5] and Section 5, and some of these versions can avoid breakdown in the rank deficient case.

To discuss the relation between the Block CG and the Block Lanczos algorithms, we note that x_k and r_k are unique, but the columns $r_k^{(i)}$ of r_k are not mutually orthogonal in general. This observation indicates that the analog of relation (4) will not be straightforward in the block case.

It is well known that the BCG block residuals can be computed using a three-term recurrence; see, e.g., [2, 4]. Here we derive this recurrence because it is a key for formulating the relation between block Lanczos and block CG algorithms.

Lemma 1 *Consider the quantities determined by Algorithm 4. The BCG residual blocks r_j, $j = 1, \ldots, k$, satisfy*

$$AR_k = R_k \widehat{T}_k - r_k \gamma_{k-1}^{-1} \phi_{k-1}^{-1} e_k^T \tag{10}$$

with $R_k \in \mathbb{R}^{n \times km}$, $R_k \equiv \left[r_0, \ldots, r_{k-2}, r_{k-1}\right]$, and the $km \times km$ block tridiagonal coefficient matrix

$$\widehat{T}_k \equiv \begin{bmatrix} \left(\gamma_0^{-1}\phi_0^{-1}\right) & -\gamma_0^{-1}\delta_1 & & \\ -\gamma_0^{-1}\phi_0^{-1} & \left(\gamma_1^{-1}\phi_1^{-1} + \gamma_0^{-1}\delta_1\right) & \ddots & \\ & \ddots & \ddots & -\gamma_{k-2}^{-1}\delta_{k-1} \\ & & -\gamma_{k-2}^{-1}\phi_{k-2}^{-1} & \left(\gamma_{k-1}^{-1}\phi_{k-1}^{-1} + \gamma_{k-2}^{-1}\delta_{k-1}\right) \end{bmatrix}.$$

The matrix $\widehat{T}_k$ can be written in the factorized form

$$\begin{bmatrix} I & & & \\ -I & I & & \\ & \ddots & \ddots & \\ & & -I & I \end{bmatrix} \begin{bmatrix} \gamma_0^{-1}\phi_0^{-1} & & & \\ & \gamma_1^{-1}\phi_1^{-1} & & \\ & & \ddots & \\ & & & \gamma_{k-1}^{-1}\phi_{k-1}^{-1} \end{bmatrix} \begin{bmatrix} I & -\phi_0\delta_1 & & \\ & I & \ddots & \\ & & \ddots & -\phi_{k-2}\delta_{k-1} \\ & & & I \end{bmatrix}.$$

Proof For a residual block r_j, $j = 1, \ldots, k$, we obtain

$$r_j = r_{j-1} - Ap_{j-1}\gamma_{j-1} = r_{j-1} - A(r_{j-1} + p_{j-2}\delta_{j-1})\phi_{j-1}\gamma_{j-1},$$

and, therefore,

$$\begin{aligned} \left(r_{j-1} - r_j\right)\gamma_{j-1}^{-1}\phi_{j-1}^{-1} &= Ar_{j-1} + Ap_{j-2}\delta_{j-1} \\ &= Ar_{j-1} + \left(Ap_{j-2}\gamma_{j-2}\right)\gamma_{j-2}^{-1}\delta_{j-1} \\ &= Ar_{j-1} + \left(r_{j-2} - r_{j-1}\right)\gamma_{j-2}^{-1}\delta_{j-1}, \end{aligned}$$

with $r_{-1} \equiv 0, \delta_0 \equiv 0,\ \gamma_{-1} \equiv I$, where the zero and identity matrices are of appropriate sizes. This results in three-term recurrences

$$Ar_{j-1} = -r_{j-2}\gamma_{j-2}^{-1}\delta_{j-1} + r_{j-1}\left(\gamma_{j-1}^{-1}\phi_{j-1}^{-1} + \gamma_{j-2}^{-1}\delta_{j-1}\right) - r_j\gamma_{j-1}^{-1}\phi_{j-1}^{-1},$$

which can be written for $j = 1, \ldots, k$ in matrix form (10). □

Note $\widehat{T}_k$ from Lemma 1 is nonsymmetric in general, but the matrix

$$R_k^T A R_k = \mathrm{diag}(r_0^T r_0, \ldots, r_{k-1}^T r_{k-1}) \widehat{T}_k$$

is symmetric.

4 The connection

To show the connection between block CG and block Lanczos algorithms, we implicitly assume exact arithmetic and full dimension of the corresponding block Krylov subspace. We first introduce normalization coefficients ϱ_j of size $m \times m$ such that

$$\varrho_j^{-T} r_j^T r_j \varrho_j^{-1} = I, \quad \text{i.e.,} \quad r_j^T r_j = \varrho_j^T \varrho_j, \quad j = 0, \ldots, k. \tag{11}$$

As in the block Lanczos algorithm, we have some freedom in determining these coefficients. In particular, consider the R-factor of the QR factorization of r_j with diagonal positive entries and denote it by σ_j. This factor is uniquely determined and can also be seen as the transpose of the L-factor of the Cholesky factorization of $r_j^T r_j$. In Matlab notation

$$\sigma_j = \mathtt{chol}(r_j^T r_j).$$

Then, any ϱ_j satisfying (11) can be parameterized using

$$\varrho_j = \theta_j^T \sigma_j$$

where θ_j is a unitary $m \times m$ matrix to be chosen. In the following, we will use the freedom in the choice of unitary θ_j to show the connection between the two block algorithms, and also to compute the block Lanczos coefficients in the BCG algorithm.

4.1 Recurrences for the normalized residual blocks

Using the coefficients ϱ_k and ϱ_{k-1}, the block CG coefficients δ_k and γ_{k-1} can be written as

$$\delta_k = \phi_{k-1}^{-1} \left(\varrho_{k-1}^T \varrho_{k-1} \right)^{-1} \varrho_k^T \varrho_k, \qquad \gamma_{k-1}^{-1} = \left(\varrho_{k-1}^T \varrho_{k-1} \right)^{-1} \phi_{k-1}^{-T} \left(p_{k-1}^T A p_{k-1} \right).$$

Let us define normalized residual blocks $\widetilde{v}_j$ using the relation

$$\widetilde{v}_j = (-1)^{j-1} r_{j-1} \varrho_{j-1}^{-1}, \quad j = 1, \ldots, k+1, \tag{12}$$

and denote $\widetilde{V}_k \equiv [\widetilde{v}_1, \ldots, \widetilde{v}_k]$. Then the following lemma holds.

Lemma 2 *Consider the quantities determined by Algorithm 4 and let ϱ_{j-1}, $j = 1, \ldots, k+1$, be normalization coefficients which satisfy* (11). *Then the normalized BCG residual blocks $\widetilde{v}_j$ defined using* (12) *satisfy*

$$A\widetilde{V}_k = \widetilde{V}_k\widetilde{T}_k + \widetilde{v}_{k+1}\widetilde{\beta}_{k+1}e_k^T, \tag{13}$$

where

$$\widetilde{T}_k \equiv \begin{bmatrix} \widetilde{\alpha}_1 & \widetilde{\beta}_2^T & & \\ \widetilde{\beta}_2 & \ddots & \ddots & \\ & \ddots & \ddots & \widetilde{\beta}_k^T \\ & & \widetilde{\beta}_k & \widetilde{\alpha}_k \end{bmatrix}, \tag{14}$$

and the coefficients in $\widetilde{T}_k$ and the coefficient $\widetilde{\beta}_{k+1}$ are determined by

$$\widetilde{\alpha}_1 = \varrho_0\gamma_0^{-1}\phi_0^{-1}\varrho_0^{-1} = \widetilde{v}_1^T A\widetilde{v}_1, \tag{15}$$

and, for $j = 1, \ldots, k-1$, by

$$\begin{aligned} \widetilde{\alpha}_{j+1} &= \varrho_j\gamma_j^{-1}\phi_j^{-1}\varrho_j^{-1} + \varrho_j\gamma_{j-1}^{-1}\delta_j\varrho_j^{-1}, \\ \widetilde{\beta}_{j+1} &= \varrho_j\gamma_{j-1}^{-1}\phi_{j-1}^{-1}\varrho_{j-1}^{-1}, \\ \widetilde{\beta}_{j+1}^T &= \varrho_{j-1}\gamma_{j-1}^{-1}\delta_j\varrho_j^{-1}. \end{aligned} \tag{16}$$

The matrix $\widetilde{T}_k$ can be factorized as $\widetilde{T}_k = \widetilde{L}_k\widetilde{D}_k\widetilde{L}_k^T$ with

$$\widetilde{L}_k = \begin{bmatrix} I & & & \\ \widetilde{\ell}_1 & I & & \\ & \ddots & \ddots & \\ & & \widetilde{\ell}_{k-1} & I \end{bmatrix}, \quad \widetilde{D}_k = \begin{bmatrix} \widetilde{d}_1 & & & \\ & \ddots & & \\ & & \ddots & \\ & & & \widetilde{d}_k \end{bmatrix},$$

where $\widetilde{\ell}_j = \varrho_j\varrho_{j-1}^{-1}$, $j = 1, \ldots, k-1$, and $\widetilde{d}_j = \varrho_{j-1}\gamma_{j-1}^{-1}\phi_{j-1}^{-1}\varrho_{j-1}^{-1}$, $j = 1, \ldots, k$.

Proof Defining the block diagonal normalization matrix

$$N_k \equiv \operatorname{diag}\left(\varrho_0, -\varrho_1, \ldots, (-1)^{k-1}\varrho_{k-1}\right),$$

the relation (10) can be transformed to

$$A\left(R_kN_k^{-1}\right) = \left(R_kN_k^{-1}\right)(N_k\widehat{T}_kN_k^{-1}) - r_k\gamma_{k-1}^{-1}\phi_{k-1}^{-1}e_k^TN_k^{-1}. \tag{17}$$

Using (12), the last term in (17) can be written in the form

$$-r_k\gamma_{k-1}^{-1}\phi_{k-1}^{-1}e_k^TN_k^{-1} = \widetilde{v}_{k+1}\left(\varrho_k\gamma_{k-1}^{-1}\phi_{k-1}^{-1}\varrho_{k-1}^{-1}\right)e_k^T. \tag{18}$$

Therefore, $\widetilde{v}_j$ satisfy (13) with the block tridiagonal matrix $\widetilde{T}_k = N_k \widehat{T}_k N_k^{-1}$ and $\widetilde{\beta}_{k+1} = \varrho_k \gamma_{k-1}^{-1} \phi_{k-1}^{-1} \varrho_{k-1}^{-1}$. Moreover, since $\widetilde{V}_k^T \widetilde{v}_{k+1} = 0$, we get $\widetilde{V}_k^T A \widetilde{V}_k = \widetilde{T}_k$, and so $\widetilde{T}_k$ is symmetric.

Considering $\widetilde{T}_k$ in the form (14) and using Lemma 1, the relations (16) and (15) hold.

Finally, the matrix $\widetilde{T}_k = N_k \widehat{T}_k N_k^{-1}$ can be factorized as

$$\widetilde{T}_k = \left(N_k \widehat{L}_k N_k^{-1}\right) \left(N_k \widehat{D}_k N_k^{-1}\right) \left(N_k \widehat{U}_k N_k^{-1}\right) \equiv \widetilde{L}_k \widetilde{D}_k \widetilde{U}_k,$$

where $\widehat{L}_k$, $\widehat{D}_k$, and $\widehat{U}_k$ denote the individual matrices of the factorization of $\widehat{T}_k$ from Lemma 1. Using the relation

$$\delta_j \varrho_j^{-1} = \phi_{j-1}^{-1} \left(\varrho_{j-1}^T \varrho_{j-1}\right)^{-1} \varrho_j^T,$$

which follows from expressions for $\widetilde{\beta}_{j+1}$ in (16), we get $\widetilde{U}_k = \widetilde{L}_k^T$, and, therefore, $\widetilde{T}_k = \widetilde{L}_k \widetilde{D}_k \widetilde{L}_k^T$ with the block entries $\widetilde{\ell}_j = \varrho_j \varrho_{j-1}^{-1}$ and $\widetilde{d}_j = \varrho_{j-1} \gamma_{j-1}^{-1} \phi_{j-1}^{-1} \varrho_{j-1}^{-1}$. $\square$

4.2 The connection between T_k and $\widetilde{T}_k$

We are now ready to formulate the connection between the matrices T_k and $\widetilde{T}_k$ under the assumption that the block Lanczos and block CG algorithms are started with the same data.

Theorem 1 *Suppose that Algorithms 3 and 4 are started with the same symmetric and positive definite matrix A and the same vector $v = r_0$. Then*

$$T_k = U_k^T \widetilde{T}_k U_k,$$

where $U_k = \operatorname{diag}(\eta_1, \eta_2, \ldots, \eta_k)$, $\eta_1 = \theta_0^T$, and η_{j+1} for $j \ge 1$ are determined using

$$\left[\eta_{j+1}, \beta_{j+1}\right] = \mathtt{qr}(\widetilde{\beta}_{j+1} \eta_j), \quad j = 1, \ldots, k-1. \tag{19}$$

Proof Since the columns of $\widetilde{v}_j$ and v_j span the same space, there are unitary matrices η_j such that

$$v_j = \widetilde{v}_j \eta_j, \quad j = 1, \ldots, k+1. \tag{20}$$

Denoting $U_k = \operatorname{diag}(\eta_1, \eta_2, \ldots, \eta_k)$ and using (20), the relation (13) can be transformed to

$$A V_k = V_k (U_k^T \widetilde{T}_k U_k) + v_{k+1} (\eta_{k+1}^T \widetilde{\beta}_{k+1} \eta_k) e_k^T,$$

and it holds that

$$T_k = V_k^T A V_k = U_k^T \widetilde{T}_k U_k,$$

or, written in matrix form,

$$\begin{bmatrix} \alpha_1 & \beta_2^T & & \\ \beta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_k^T \\ & & \beta_k & \alpha_k \end{bmatrix} = \begin{bmatrix} \eta_1^T \widetilde{\alpha}_1 \eta_1 & \left(\eta_2^T \widetilde{\beta}_2 \eta_1\right)^T & & \\ \eta_2^T \widetilde{\beta}_2 \eta_1 & \eta_2^T \widetilde{\alpha}_2 \eta_2 & \ddots & \\ & \ddots & \ddots & \left(\eta_k^T \widetilde{\beta}_k \eta_{k-1}\right)^T \\ & & \eta_k^T \widetilde{\beta}_k \eta_{k-1} & \eta_k^T \widetilde{\alpha}_k \eta_k \end{bmatrix}.$$

Let us now determine η_j. At the beginning of the algorithms, the R-factors β_1 and σ_0 are equal so that

$$v_1\sigma_0 = r_0 = \widetilde{v}_1\varrho_0 = \widetilde{v}_1\theta_0^T\sigma_0.$$

Therefore, $v_1 = \widetilde{v}_1\theta_0^T$ and $\eta_1 = \theta_0^T$. To determine η_{j+1} for $j > 0$, we use $\beta_{j+1} = \eta_{j+1}^T\widetilde{\beta}_{j+1}\eta_j$, i.e.,

$$\eta_{j+1}\beta_{j+1} = \widetilde{\beta}_{j+1}\eta_j.$$

Obviously, $\eta_{j+1}\beta_{j+1}$ represents a QR factorization of $\widetilde{\beta}_{j+1}\eta_j$. □

In summary, BCG computes implicitly the block Cholesky factorization of the block tridiagonal matrix $\widetilde{T}_k$ that is unitarily similar to T_k. The unitary similarity transformation between T_k and $\widetilde{T}_k$ is realized via U_k with diagonal blocks satisfying (19).

4.3 Block Lanczos coefficients in BCG

We now use the freedom in the choice of ϱ_j satisfying (11) to compute the block Lanczos coefficients within the BCG algorithm. Using results from the previous section, our aim is to choose ϱ_j such that U_k discussed above is the identity matrix, so that $\widetilde{V}_k = V_k$ and $\widetilde{T}_k = T_k$; see (12) and (16). We summarize our results in the following theorem.

Theorem 2 *Consider the quantities determined by Algorithm 4 and let* $\sigma_0 = \mathtt{chol}(r_0^T r_0)$, $\beta_1 = \sigma_0$, $\theta_0 = I$, *and* $\ell_0 = 0$. *Then, for* $j = 1, \ldots, k$, *the Lanczos coefficients* α_j *and* β_{j+1} *forming* T_k, *see* (9), *and the coefficients* ℓ_j *and* d_j *from the factorization* $T_k = L_k D_k L_k^T$,

$$L_k = \begin{bmatrix} I & & & \\ \ell_1 & I & & \\ & \ddots & \ddots & \\ & & \ell_{k-1} & I \end{bmatrix}, \quad D_k = \begin{bmatrix} d_1 & & & \\ & \ddots & & \\ & & \ddots & \\ & & & d_k \end{bmatrix},$$

can be computed using the recurrences

$$\begin{aligned} \tau_j &= \gamma_{j-1}^{-1}\phi_{j-1}^{-1}\sigma_{j-1}^{-1}\theta_{j-1}, \\ d_j &= \theta_{j-1}^T\sigma_{j-1}\tau_j, \\ \alpha_j &= d_j + \ell_{j-1}\beta_j^T, \end{aligned}$$

$$\begin{aligned} \sigma_j &= \mathtt{chol}(r_j^T r_j), \\ [\theta_j, \beta_{j+1}] &= \mathtt{qr}(\sigma_j \tau_j), \\ \ell_j &= \theta_j^T \sigma_j \sigma_{j-1}^{-1} \theta_{j-1}. \end{aligned} \tag{21}$$

The Lanczos block v_{k+1} can be obtained from r_k as

$$v_{k+1} = (-1)^k \, r_k \sigma_k^{-1} \theta_k^{-T}. \tag{22}$$

Proof The unitary coefficient matrices η_j forming U_k are determined using the recurrence (19) started with $\eta_1 = \theta_0^T$. We first choose $\theta_0 = I$ so that $\eta_1 = I$. Using (19) and assuming that $\eta_j = I$, it holds that $\eta_{j+1} = I$ if and only if $\widetilde{\beta}_{j+1}$ is upper triangular with positive entries on the diagonal. Hence, our aim is to determine $\varrho_j = \theta_j^T \sigma_j$ such that

$$\widetilde{\beta}_{j+1} = \varrho_j \gamma_{j-1}^{-1} \phi_{j-1}^{-1} \varrho_{j-1}^{-1} = \theta_j^T \sigma_j \tau_j \tag{23}$$

has this property, where we defined

$$\tau_j \equiv \gamma_{j-1}^{-1} \phi_{j-1}^{-1} \varrho_{j-1}^{-1}.$$

In other words, in (23) we require that $\theta_j \widetilde{\beta}_{j+1}$ is the QR factorization of $\sigma_j \tau_j$. Hence, by computing the QR factorization of $\sigma_j \tau_j$ and assuming that $\eta_i = I$, $i = 1, \ldots, j$, we get

$$[\theta_j, \beta_{j+1}] = \mathtt{qr}(\sigma_j \tau_j) \tag{24}$$

and $\eta_{j+1} = I$. Note that with the choice (24), it holds that $v_{j+1} = (-1)^j r_j \varrho_j^{-1}$, $j = 1, \ldots, k$. Using (16) and the definition of τ_j, the coefficients α_j are given by

$$\alpha_j = \theta_{j-1}^T \sigma_{j-1} \tau_j + \varrho_{j-1} \varrho_{j-2}^{-1} \beta_j^T, \quad \alpha_1 = \sigma_0 \tau_1.$$

Using Lemma 2, one can compute the coefficients of the LDL^T factorization of T_k as

$$\ell_j = \varrho_j \varrho_{j-1}^{-1} = \theta_j^T \sigma_j \sigma_{j-1}^{-1} \theta_{j-1}, \quad \text{and} \quad d_j = \theta_{j-1}^T \sigma_{j-1} \tau_j$$

so that $\alpha_j = d_j + \ell_{j-1} \beta_j^T$. Note that since $\eta_j = I$, it holds that $v_{k+1} = (-1)^k \, r_k \sigma_k^{-1} \theta_k^{-T}$. □

To include the calculation of the Lanczos coefficients in BCG, the recurrences (21) with index k can be placed below line 9 in Algorithm 4. The recurrence (22) can also be added to reconstruct the Lanczos blocks v_{k+1}.

5 Practical block CG variants

From the practical point of view, one should consider the situation when the assumption of the full dimension of the block Krylov subspace is not satisfied. This question was discussed in detail for example in [4, 6, 7, 18, 19], and some algorithmic ways were suggested to deal with rank deficiency. Although rank deficiency can be handled algorithmically through deflation and variable block size, there are still quite complicated questions in the air, such as which matrix should be considered rank deficient when using finite precision arithmetic, or, if we remove some vectors from the process, how does this affect the convergence of the method in finite precision arithmetic. In general, it is not obvious whether deflation ideas derived assuming exact arithmetic are still applicable in finite precision arithmetic.

When using short recurrences in finite precision arithmetic, we will never have orthogonality and thus the dimension of the block Krylov subspace under control. Similarly as in CG, orthogonality among the blocks is usually lost after few iterations, and the blocks can become linearly dependent. Despite this fact, we usually observe that BCG converges, but the convergence can be significantly delayed. Behaviour of classical CG in finite precision arithmetic is quite well understood thanks to results by Paige, Greenbaum, and Strakoš; see [20–22]. The finite precision behaviour of BCG is not well understood and a generalization of the above mentioned results to the block case is an open problem.

From a pragmatic point of view, we should therefore be primarily interested in being able to perform the individual BCG iterations in a numerically stable way, and to preserve at least the local orthogonality (the orthogonality between consecutive blocks) well. The local orthogonality influences the delay of convergence and is also important for the estimation of the A-norm of the error; see [23, 24]. Looking at the BCG coefficients,

$$\gamma_{k-1} = \left(p_{k-1}^T A p_{k-1}\right)^{-1} \phi_{k-1}^T r_{k-1}^T r_{k-1} = \left(p_{k-1}^T A p_{k-1}\right)^{-1} p_{k-1}^T r_{k-1}, \tag{25}$$

$$\delta_k = \phi_{k-1}^{-1} \left(r_{k-1}^T r_{k-1}\right)^{-1} r_k^T r_k = -\left(p_{k-1}^T A p_{k-1}\right)^{-1} p_{k-1}^T A r_k, \tag{26}$$

for a practical realization of the BCG method it is sufficient to deal only with the (near) rank deficiency that may occur within the residual blocks r_{k-1} or the direction blocks p_{k-1}, and that can prevent us from calculating the CG coefficients in a numerically stable way. Inversions of almost singular matrices can destroy the quality of the computations.

What we think is the most promising approach to overcome the difficulties with rank deficiency within blocks was presented in the paper by Dubrulle [5]. His idea was to use changes of bases that supplement rank defects and enforce linear independence. Thanks to this approach, no deflation is necessary. Dubrulle [5] formulated several variants of BCG which ensure that the inverses of the matrices used in the algorithm are well defined during computations in finite precision arithmetic.Based on our numerical experiments with all variants of BCG described in [5], we present and discuss two of them that we consider to be the most interesting ones. The first one is based on the idea of computing the QR factorization of the residual block and then using the Q-factor in the upcoming computations. We call this variant the Dubrulle-R variant or DR-BCG. The second variant applies a similar idea to the direction blocks, and we call this variant the Dubrulle-P variant or DP-BCG.

5.1 The Dubrulle-R variant

We will now derive the Dubrulle-R variant (DR-BCG); see [5, Algorithm BCGrQ]. Let us consider a QR factorization of r_k in the form,

$$[w_k, \sigma_k] = \texttt{qr}(r_k),$$

implemented using Householder QR, with positive entries on the diagonal of σ_k (because of consistency with notation in Section 4). Using Householder QR ensures that w_k has orthonormal columns even if r_k has linearly dependent columns.

To formally derive DR-BCG, we assume σ_k to be nonsingular. Let us define

$$\phi_k \equiv \sigma_k^{-1}\sigma_{k-1}^{-1}$$

in Algorithm 4, and denote $s_k = p_k\sigma_{k-1}$ with $\sigma_{-1} = I$. Then

$$\begin{aligned}
\gamma_{k-1} &= \left(p_{k-1}^T A p_{k-1}\right)^{-1} \phi_{k-1}^T r_{k-1}^T r_{k-1} = \sigma_{k-2}\left(s_{k-1}^T A s_{k-1}\right)^{-1}\sigma_{k-1},\\
\delta_k &= \phi_{k-1}^{-1}\left(r_{k-1}^T r_{k-1}\right)^{-1} r_k^T r_k = \sigma_{k-2}\sigma_{k-1}^{-T}\sigma_k^T\sigma_k,
\end{aligned}$$

and x_k, w_k, and s_k satisfy the recurrences

$$\begin{aligned}
x_k &= x_{k-1} + s_{k-1}\left(s_{k-1}^T A s_{k-1}\right)^{-1}\sigma_{k-1},\\
w_k\sigma_k\sigma_{k-1}^{-1} &= w_{k-1} - As_{k-1}\left(s_{k-1}^T A s_{k-1}\right)^{-1},\\
s_k &= w_k + s_{k-1}\sigma_{k-1}^{-T}\sigma_k^T.
\end{aligned}$$

Denoting $\xi_{k-1} = \left(s_{k-1}^T A s_{k-1}\right)^{-1}$ and $\zeta_k = \sigma_k\sigma_{k-1}^{-1}$ we obtain Algorithm 5.

Algorithm 5 Dubrulle-R BCG (DR-BCG).

```
1: input $A, b, x_0$
2: $r_0 = b - Ax_0$
3: $[w_0, \sigma_0] = \texttt{qr}(r_0)$
4: $s_0 = w_0$
5: for $k = 1, 2, \ldots$ do
6:     $\xi_{k-1} = \left(s_{k-1}^T A s_{k-1}\right)^{-1}$
7:     $x_k = x_{k-1} + s_{k-1}\xi_{k-1}\sigma_{k-1}$
8:     $[w_k, \zeta_k] = \texttt{qr}(w_{k-1} - As_{k-1}\xi_{k-1})$
9:     $s_k = w_k + s_{k-1}\zeta_k^T$
10:    $\sigma_k = \zeta_k\sigma_{k-1}$
11: end for
```

We derived this algorithm assuming that σ_k is nonsingular, but, as one can observe, we do not need the nonsingularity of σ_k in Algorithm 5. The only assumption that we need is that $s_k^T As_k$ is nonsingular so that the coefficient ξ_k is well defined.

The residual blocks r_k are not available in Algorithm 5, but we can always reconstruct them. Using the recurrence for updating x_k, we obtain

$$r_k = r_{k-1} - As_{k-1}\xi_{k-1}\sigma_{k-1}.$$

Using the induction hypothesis $r_{k-1} = w_{k-1}\sigma_{k-1}$ we find out that

$$r_k = (w_{k-1} - As_{k-1}\xi_{k-1})\,\sigma_{k-1} = w_k\zeta_k\sigma_{k-1} = w_k\sigma_k.$$

In the following theorem we show how to compute the block Lanczos coefficients in DR-BCG. To be mathematically correct, we implicitly assume, as in Sections 3 and 4, exact arithmetic and full dimension of the corresponding block Krylov subspace.

Theorem 3 *Consider the quantities determined by Algorithm 5 and let $\beta_1 = \sigma_0$, $\theta_0 = I$, and $\ell_0 = 0$. Then, for $j = 1, \ldots, k$, the Lanczos coefficients α_j and β_{j+1} forming the matrix T_k and the coefficients ℓ_j and d_j used in $T_k = L_k D_k L_k^T$ can be computed using the recurrences*

$$\begin{aligned} \tilde{\tau}_k &= \xi_{k-1}^{-1}\theta_{k-1}, \\ d_k &= \theta_{k-1}^T\tilde{\tau}_k, \\ \alpha_k &= d_k + \ell_{k-1}\beta_k^T, \\ [\theta_k, \beta_{k+1}] &= \mathtt{qr}(\zeta_k\tilde{\tau}_k), \\ \ell_k &= \theta_k^T\zeta_k\theta_{k-1}. \end{aligned} \tag{27}$$

The Lanczos block v_{k+1} can be obtained from w_k as $v_{k+1} = (-1)^k\,w_k\theta_k^{-T}$.

Proof To derive formulas for computing the Lanczos coefficients in Algorithm 5, we can use the relations (21) from Section 4.3 with the choice $\phi_k = \sigma_k^{-1}\sigma_{k-1}^{-1}$. In DR-BCG, the σ_k's are already available and need not be computed. Using

$$\phi_k = \sigma_k^{-1}\sigma_{k-1}^{-1} \quad \text{and} \quad \gamma_{k-1} = \sigma_{k-2}\xi_{k-1}\sigma_{k-1}$$

we obtain $\sigma_k\tau_k = \zeta_k\xi_{k-1}^{-1}\theta_{k-1}$. Denoting $\tilde{\tau}_k \equiv \sigma_{k-1}\tau_k$ we obtain the relations (27). Finally, the relation for v_{k+1} follows from $v_{k+1} = (-1)^k\,r_k\sigma_k^{-1}\theta_k^{-T} = (-1)^k\,w_k\theta_k^{-T}$. □

Looking only at the final formulas (27), we do not need to assume that σ_k is nonsingular in order to compute the Lanczos coefficients and blocks. Thus, DR-BCG corresponds to a version of the block Lanczos algorithm that does not require deflation and allows the off-diagonal blocks of T_k to be singular. These variants deserve more attention and a deeper analysis, which is beyond the scope of this paper.

5.2 The Dubrulle-P variant

To derive the Dubrulle-P variant (DP-BCG), see [5, Algorithm BCGAdQ], we will consider an alternative way of computing coefficients γ_{k-1} and δ_k in BCG; see (25) and (26). For stable computations, we only need the inverse of the matrix $p_{k-1}^T A p_{k-1}$ to be well defined. The formulas (25) and (26) appear already in the Hestenes and Stiefel paper [25] for classical CG, and follow easily from the formulas in the paper by O'Leary [2]. They are also used to derive a version of BCG in [6] that will be discussed in the numerical experiments section. Using (25) and (26), the coefficient matrix ϕ_k which is free to choose appears only at line 9 of Algorithm 4. Let us consider a QR factorization of $r_k + p_{k-1}\delta_k$,

$$[p_k, \psi_k] = \texttt{qr}(r_k + p_{k-1}\delta_k).$$

Note that to implement the above QR factorization we again use Householder QR, which ensures that p_k has orthonormal columns even if $r_k + p_{k-1}\delta_k$ has linearly dependent columns. To formally derive the algorithm, we will assume that ψ_k is nonsingular. Let us define

$$\phi_k^{-1} \equiv \psi_k$$

in Algorithm 4, with coefficients γ_{k-1} and δ_k computed using the alternative formulas (25) and (26). Then we obtain Algorithm 6.

Algorithm 6 Dubrulle-P BCG (DP-BCG).

```
1: input $A, b, x_0$
2: $r_0 = b - Ax_0$
3: $[p_0, \psi_0] = \texttt{qr}(r_0)$
4: for $k = 1, 2, \ldots$ do
5:     $\gamma_{k-1} = \left(p_{k-1}^T A p_{k-1}\right)^{-1} p_{k-1}^T r_{k-1}$
6:     $x_k = x_{k-1} + p_{k-1}\gamma_{k-1}$
7:     $r_k = r_{k-1} - Ap_{k-1}\gamma_{k-1}$
8:     $\delta_k = -\left(p_{k-1}^T A p_{k-1}\right)^{-1} p_{k-1}^T A r_k$
9:     $[p_k, \psi_k] = \texttt{qr}(r_k + p_{k-1}\delta_k)$
10: end for
```

Using Theorem 2 and the particular choice of ϕ_k, the following theorem follows.

Theorem 4 *Consider the quantities determined by Algorithm 6 and let $\sigma_0 = \psi_0$, $\beta_1 = \sigma_0$, $\theta_0 = I$, and $\ell_0 = 0$. Then, for $j = 1, \ldots, k$, the Lanczos coefficients α_j and β_{j+1} forming the matrix T_k and the coefficients ℓ_j and d_j used in $T_k = L_k D_k L_k^T$ can be computed using the recurrences*

$$\begin{aligned}
\tau_k &= \gamma_{k-1}^{-1}\psi_{k-1}\sigma_{k-1}^{-1}\theta_{k-1} \\
d_k &= \theta_{k-1}^T \sigma_{k-1}\tau_k \\
\alpha_k &= d_k + \ell_{k-1}\beta_k^T \\
\sigma_k &= \texttt{chol}(\psi_k^T p_k^T r_k) \\
[\theta_k, \beta_{k+1}] &= \texttt{qr}(\sigma_k \tau_k) \\
\ell_k &= \theta_k^T \sigma_k \sigma_{k-1}^{-1}\theta_{k-1}.
\end{aligned} \tag{28}$$

The Lanczos block v_{k+1} can be reconstructed using $v_{k+1} = (-1)^k\, r_k \sigma_k^{-1}\theta_k^{-T}$.

Proof To compute the Lanczos coefficients in Algorithm 6, we can again use the relations (21) from Section 4.3, now with the choice $\phi_k \equiv \psi_k^{-1}$. Since $r_k^T r_k$ is not computed in Algorithm 6 any more, we can use the relation $\psi_k^T p_k^T r_k = r_k^T r_k$ to compute σ_k. Starting with $\sigma_0 = \psi_0$, we obtain the relations (28). □

If we want to compute the Lanczos coefficients within DP-BCG, we have to assume that the columns of r_k are linearly independent, so that the inverse of σ_k is well defined. In other words, we have to assume that the matrices ψ_k are nonsingular. This requirement is natural here because in DP-BCG, the columns of a residual block may be linearly dependent, in which case the corresponding Lanczos block does not exist.

6 Practical issues

In practical computations, it is convenient to make several modifications to those BCG variants. First, to speed up convergence, we need to introduce preconditioning. Second, one should think about stopping criteria for each system. Once we have sufficiently accurate approximate solutions for some of the systems, it would be beneficial to remove the corresponding systems from the algorithm (reduce the size of the blocks) to save computational resources. However, the subject of removing systems from the BCG process is beyond the scope of this paper.

6.1 Preconditioning

Suppose we are given a symmetric positive definite preconditioner M. Algorithms for preconditioned block CG can already be found in the original work of O'Leary [2]. They can be derived in the standard way by implicit application of block CG to the preconditioned system $L^{-1}AL^{-T}y = L^{-1}b$, where L is a nonsingular matrix such that $LL^T = M$. Then, the solutions of $Ax = b$ and the preconditioned system are related using $x = L^{-T}y$, and one can use this transformation to define the approximate solution $x_k = L^{-T}y_k$.

In the single-vector versions of CG, the splitting LL^T is needed to derive the algorithm, but then we just need to solve linear systems with M in the preconditioned algorithm. In the block versions of CG, the situation can be different, in particular for DR-BCG; see Algorithm 7.

Algorithm 7 Preconditioned DR-BCG.

```
1: input $A, b, x_0, M = LL^T$
2: $r_0 = b - Ax_0$
3: $[w_0, \sigma_0] = \mathtt{qr}(L^{-1}r_0)$
4: $s_0 = L^{-T}w_0$
5: for $k = 1, 2, \ldots$ do
6:     $\xi_{k-1} = \left(s_{k-1}^T A s_{k-1}\right)^{-1}$
7:     $x_k = x_{k-1} + s_{k-1}\xi_{k-1}\sigma_{k-1}$
8:     $[w_k, \zeta_k] = \mathtt{qr}(w_{k-1} - L^{-1}As_{k-1}\xi_{k-1})$
9:     $s_k = L^{-T}w_k + s_{k-1}\zeta_k^T$
10:    $\sigma_k = \zeta_k\sigma_{k-1}$
11: end for
```

To derive preconditioned DR-BCG, we need first to apply formally the DR-BCG algorithm to the preconditioned system. If we do so in the standard manner, we need to compute QR factorizations of the preconditioned residuals Lr_k. So the preconditioned residual should also be available in the preconditioned algorithm, and to compute it we need the matrix L. Therefore, to derive the standard version of preconditioned DR-BCG, we assume that a split preconditioner $M = LL^T$ is available. Note that it is possible to derive a preconditioned version of DR-BCG that does not require a split preconditioner. However, it would require computing the QR factorization of a block with Q having M^{-1}-orthogonal columns. There are several ways to do this. An algorithm that uses Householder-like transformations and appears to be stable has been proposed by Shao [26]. However, this algorithm uses multiple products of the matrix defining the inner product with some vectors. In our case, this corresponds to solving linear systems with the preconditioner M several times in one iteration, and this can be very expensive. Nevertheless, it seems to work well numerically, as predicted by our unpublished experiments.

Algorithm 8 Preconditioned DP-BCG.

```
1: input $A, b, x_0, M$
2: $r_0 = b - Ax_0$
3: $z_0 = M^{-1}r_0$
4: $[p_0, \psi_0] = \mathtt{qr}(z_0)$
5: for $k = 1, 2, \dots$ do
6:     $\gamma_{k-1} = \left(p_{k-1}^T A p_{k-1}\right)^{-1} p_{k-1}^T r_{k-1}$
7:     $x_k = x_{k-1} + p_{k-1}\gamma_{k-1}$
8:     $r_k = r_{k-1} - Ap_{k-1}\gamma_{k-1}$
9:     $z_k = M^{-1}r_k$
10:    $\delta_k = -\left(p_{k-1}^T A p_{k-1}\right)^{-1} p_{k-1}^T A z_k$
11:    $[p_k, \psi_k] = \mathtt{qr}(z_k + p_{k-1}\delta_k)$
12: end for
```

To precondition the DP-BCG algorithm, we can use standard techniques; see Algorithm 8. Here, the splitting $LL^T = M$ does not need to be available, it is only used for the derivation of the algorithm.

6.2 Stopping the algorithms

To stop the algorithms, one can use the individual squared residual norms $\|r_k^{(i)}\|^2$, i.e., the diagonal entries of $r_k^T r_k = \sigma_k^T \sigma_k$. However, as in classical CG, we can do better. In the block case, it is possible to estimate the diagonal entries of

$$(x - x_k)^T \, A \, (x - x_k) \, ,$$

i.e., the squared A-norm of the error $\|x^{(i)} - x_k^{(i)}\|_A^2$ of the individual systems. The details will be explained in a forthcoming paper, where we generalize the formulas for computing the lower and upper bounds, see, e.g, [11], also for the block CG algorithms. Note that to compute the upper bounds, we need to know an underestimate of the smallest eigenvalue of the preconditioned matrix.

7 Numerical experiments

The experiments are performed in MATLAB 9.14 (R2023a). We consider the matrices `bcsstk03` and `s3dkt3m2` from the SuiteSparse[1] matrix collection. In particular, we use the matrix `bcsstk03` of size $n = 112$ and $\kappa(A) \approx 9.5 \times 10^6$ to test the algorithms without preconditioning, and the matrix `s3dkt3m2` of size $n = 90449$ and $\kappa(A) \approx 3.6 \times 10^{11}$ to test the problems with preconditioning. The factor L in the preconditioner $M = LL^T$ is determined by the incomplete Cholesky (`ichol`) factorization with threshold dropping, `type = 'ict'`,`droptol = 1e-5`, and with the global diagonal shift `diagcomp = 1e-2`. We focus on numerical behaviour of the three considered variants of block CG with increasing block size m, in particular on HS-BCG (Algorithm 4 with $\phi_k = I$), DR-BCG (Algorithm 7), and DP-BCG (Algorithm 8). The block right-hand sides are generated randomly using the `rand(n,m)` command. We start the algorithms with the initial guess $x_0 = 0 \in \mathbb{R}^{n\times m}$ and plot the convergence characteristic

$$\omega_k \equiv \left(\frac{\text{trace}((x - x_k)^T\, A\, (x - x_k))}{\text{trace}(x^T A x)} \right)^{1/2},$$

which is an analogue of the relative A-norm of the error in the standard PCG case.

In Fig. 1 we observe that for the matrix `bcsstk03` and a single right-hand side, HS-BCG (red dotted) and DR-BCG (black solid) produce almost the same convergence curves while DP-BCG (blue dashed) is slightly delayed (top left part). For $m = 1$, all tested versions of block CG reach the same level of maximum attainable accuracy. As we increase the block size to $m = 2, 4, 6$ we can see that the picture changes significantly. We can see that DR-BCG is the winner both in terms of the speed of convergence as well as in terms of the maximum accuracy that can be achieved. The convergence of DP-BCG is delayed, but with increasing iterations it usually reaches the same level of maximum attainable accuracy as DR-BCG. The convergence of the HS-BCG version with no deflation is significantly delayed compared to the other two versions tested, and the level of maximum attainable accuracy is the worst. Note that during the MATLAB computations, a warning that a matrix is nearly singular or badly scaled was displayed several times during the HS-BCG computations, on lines related to the computation of the coefficients γ_{k-1} and δ_k.

In Fig. 2 we test the preconditioned block CG versions with the matrix `s3dkt3m2` and $m = 1, 4, 16$, and 64 randomly generated right-hand sides. This experiment also confirms the significantly better numerical behavior of the DR-BCG variant (black solid). The DP-BCG variant (blue dashed) does not lag behind DR-BCG as much in terms of convergence delay as in the previous experiment, but achieves a significantly worse level of maximum attainable accuracy. The HS-BCG variant (red dotted) is almost unusable in this experiment for $m > 1$. Inaccuracies in the computations of the block coefficients cause the HS-BCG variant not to converge.

Note that all the block versions have a similar computational cost per iteration. For the most efficient version, i.e. DR-BCG, the total number of matrix-vector products

[1] https://sparse.tamu.edu

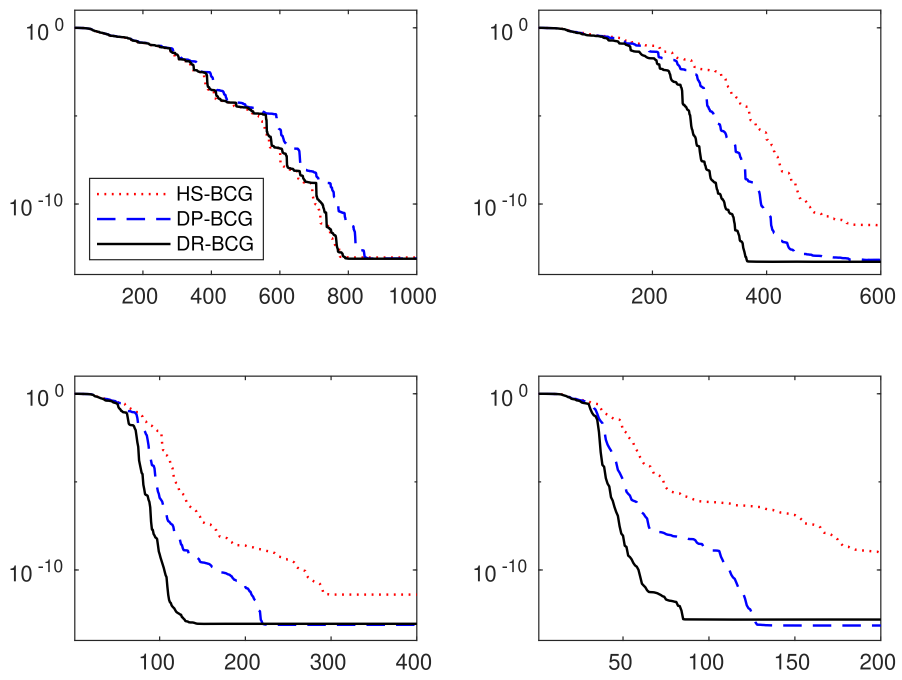

Fig. 1 The convergence characteristic ω_k in HS-BCG (red dotted), DP-BCG (blue dashed), and DR-BCG (black solid) for bcsstk03 and $b \in \mathbb{R}^{n\times m}$, $m = 1$ (top left), $m = 2$ (top right), $m = 4$ (bottom left), $m = 6$ (bottom right)

required to solve all systems is approximately 3000, 4000, 6400 and 9600 for $m = 1$, $m = 4$, $m = 16$ and $m = 64$ respectively. The totals show that the number of matrix-vector products needed to solve a single system decreases with increasing m. While for $m = 1$ we need almost 3000 matrix-vector products to reach the maximum level of accuracy, for $m = 4$, $m = 16$ and $m = 64$ we need only 1000, 400 and 150 matrix-vector products respectively to solve a single system. Computational time is influenced by many factors, so it cannot be taken as an objective measure of an algorithm's efficiency. Nevertheless, even on an old computer with Intel Core i5-4570 CPU, 8 GB RAM, running Linux, one can observe the significant advantages of block operations. The computation times for $m = 1$, $m = 4$, $m = 16$ and $m = 64$ were approximately 98, 52, 97 and 140 seconds respectively. So we were able to solve more systems in almost the same time. The reduction in time for $m = 4$ is probably due to better use of the computer cache.

In the second part of numerical experiments we discuss the breakdown-free BCG (BF-BCG) algorithm introduced in [6, Algorithm 2], and compare it with DP-BCG. We compare these two algorithms because they are almost identical, except for line 11 of Algorithm 8. The aim of this experiment is to demonstrate that algorithms with deflation, such as BF-BCG, do not generally perform better than their counterparts (in this case DP-BCG) which use Dubrulle's idea. For a comparison of DP-BCG and DR-BCG, see the previous experiment.

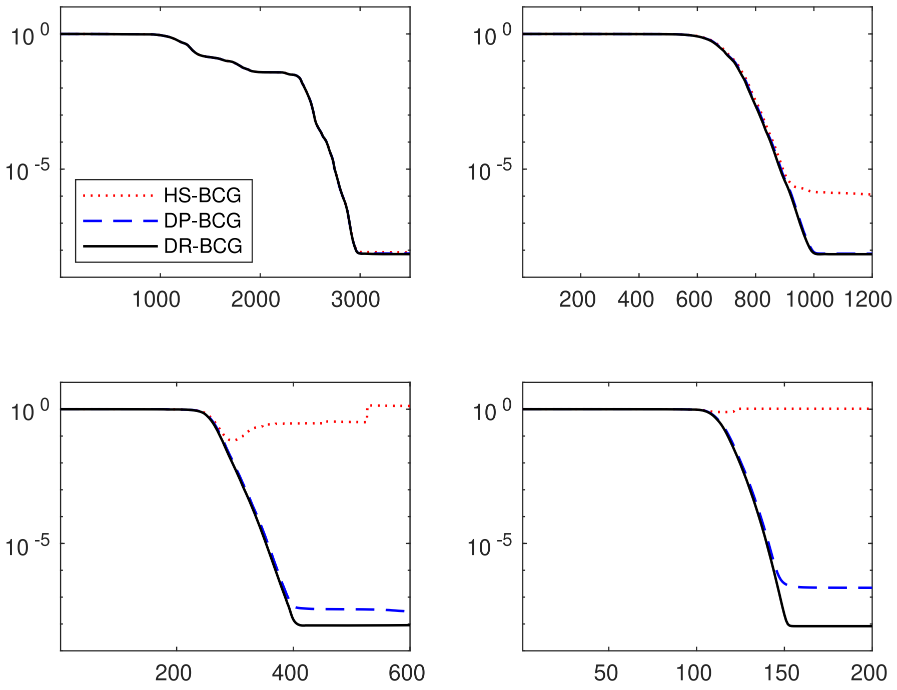

Fig. 2 The convergence characteristic ω_k in HS-BCG (red dotted), DP-BCG (blue dashed), and DR-BCG (black solid) for s3dkt3m2 and $b \in \mathbb{R}^{n\times m}$, $m = 1$ (top left), $m = 4$ (top right), $m = 16$ (bottom left), $m = 64$ (bottom right)

As mentioned above, the only difference between BF-BCG and DP-BCG is in line 11 of Algorithm 8. While in DP-BCG p_k always has orthonormal columns and is of size $n \times m$ even if the block $z_k + p_{k-1}\delta_k$ has linearly dependent columns, in BF-BCG p_k is determined as the orthonormal basis of the space spanned by the columns of $z_k + p_{k-1}\delta_k$. In exact arithmetic, the BF-BCG approach will avoid the breakdown due to linear dependence of columns of $z_k + p_{k-1}\delta_k$. Moreover, as shown in [6, Theorems 3.1 and 3.2], this strategy ensures that the orthogonality relations are preserved. In other words, convergence should not be affected by reducing the size of the blocks p_k in the rank deficient case. Note that while the size of p_k can be reduced in BF-BCG, the sizes of the blocks x_k, r_k, and z_k remain the same, and the coefficients γ_{k-1} and δ_k are rectangular matrices in general. The authors of [6] consider only the case of exact linear dependence of vectors in the block. A practical realization of their approach can be found in [27], where the rank revealing QR algorithm with the tolerance of the square root of the machine epsilon is used to determine the block p_k.

In Fig. 3 we compare DP-BCG (Algorithm 8) and the breakdown-free BCG algorithm (BF-BCG) implemented similarly as in [27, Algorithm 2.3]. For the experiment we again use the matrix `s3dkt3m2` and randomly generated right-hand sides with $m = 16$ (top part) and $m = 64$ (bottom part). Instead of the rank revealing QR which should be used in practical computations to determine the orthonormal basis of the space spanned by the columns of $z_k + p_{k-1}\delta_k$, we use the more expensive singular value decomposition. In more details, we determine the desired orthonormal basis as the left

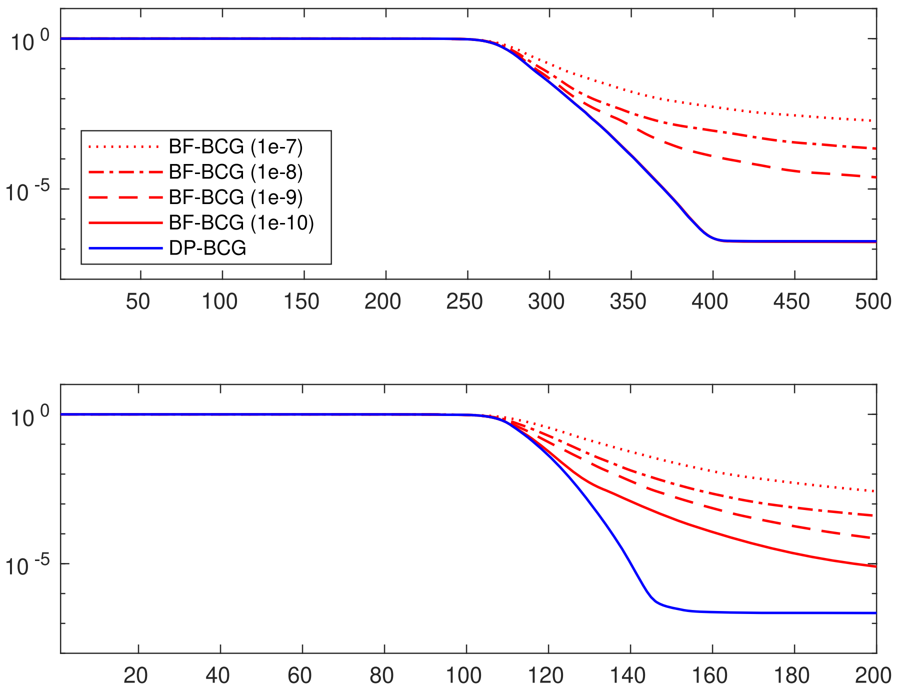

Fig. 3 The convergence characteristic ω_k in DP-BCG (blue) and BF-BCG (red) for s3dkt3m2 and random $b \in \mathbb{R}^{n\times m}$ with $m = 16$ (top part) and $m = 64$ (bottom part)

singular vectors corresponding to singular values whose relative size, with respect to the largest singular value, is greater than a given tolerance. In the experiment we used the tolerances 10^{-7} (red dotted), 10^{-8} (red dashed), 10^{-9} (red dashed dotted), and 10^{-10} (red solid). We can observe that in finite precision computations, DP-BCG (blue solid) clearly outperforms BF-BCG (red curves). We can see that reducing the size of the direction block in BF-BCG slows down the convergence. While the idea introduced in [6] is theoretically interesting, from a practical point of view the Dubrulle DP-BCG seems to work better than BF-BCG. The DP-BCG algorithm is not only simpler than BF-BCG, but also has a better numerical behavior.

8 Conclusions and future work

In this paper we discussed several variants of the block conjugate gradient (BCG) algorithms. We clarified the relationship between BCG algorithms and the block Lanczos algorithm. In particular, we showed how to easily compute the block Lanczos coefficients in BCG algorithms. We also discussed important variants of BCG due to Dubrulle, including their preconditioned variants. Numerical experiments show that Dubrulle's variants, in particular DR-BCG, are superior to the other BCG variants (HS-BCG or BF-BCG) and avoid the problems with rank deficiency of the computed block coefficients. In our numerical experiments, DR-BCG converged always faster than the other variants and reached the best level of maximum attainable accuracy.

The results presented in this paper open up several research topics for future work. First, we aim to generalize quadrature-based bounds on the A-norm of the error to the block case. Although the bounds are probably computable only from the block CG coefficients, see [28] and [11] for classical CG, for understanding and deriving the bounds we need the connection to block Jacobi matrices. Second, we believe that Dubrulle's variants deserve more attention and should be studied from both a theoretical and a practical point of view. Theoretically, we still do not fully understand which orthogonality relations are preserved in DR-BCG in the rank deficient case and whether ξ_{k-1} is always well defined. Practically, we should also explore the other variants due to Dubrulle, which, for example, use LU factorization with pivoting instead of a (more expensive) QR factorization to compute regularized blocks.

Acknowledgements The authors thank the anonymous referee for carefully reviewing the manuscript and providing helpful, constructive comments that have improved its presentation. Petr Tichý is a member of Nečas Center for Mathematical Modeling.

Author Contributions These authors contributed equally to this work.

Funding Open access publishing supported by the institutions participating in the CzechELib Transformative Agreement. This work was supported by the Cooperatio Program of Charles University, research area Mathematics.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Ethical Approval Not applicable.

Competing Interests The authors declare no competing interests.

References

1. Frommer, A., Lund, K., Szyld, D.B.: Block Krylov subspace methods for functions of matrices. Electron. Trans. Numer. Anal. **47**, 100–126 (2017)
2. O'Leary, D.P.: The block conjugate gradient algorithm and related methods. Linear Algebra Appl. **29**, 293–322 (1980)
3. Feng, Y.T., Owen, D.R.J., Perić, D.: A block conjugate gradient method applied to linear systems with multiple right-hand sides. Comput. Methods Appl. Mech. Engrg. **127**(1–4), 203–215 (1995)
4. Nikishin, A.A., Yeremin, A.Y.: Variable block CG algorithms for solving large sparse symmetric positive definite linear systems on parallel computers. I. General iterative scheme. SIAM J. Matrix Anal. Appl. **16**(4), 1135–1153 (1995)

5. Dubrulle, A.A.: Retooling the method of block conjugate gradients. Electron. Trans. Numer. Anal. **12**, 216–233 (2001)
6. Ji, H., Li, Y.: A breakdown-free block conjugate gradient method. BIT **57**(2), 379–403 (2017)
7. Birk, S., Frommer, A.: A deflated conjugate gradient method for multiple right hand sides and multiple shifts. Numer. Algorithms **67**(3), 507–529 (2014)
8. Aliaga, J.I., Boley, D.L., Freund, R.W., Hernández, V.: A Lanczos-type method for multiple starting vectors. Math. Comp. **69**(232), 1577–1601 (2000)
9. Meurant, G., Tichý, P.: Error norm estimates for the block conjugate gradient algorithm. arXiv:2502.14979 (2025)
10. Golub, G.H., Meurant, G.: Matrices. Moments and Quadrature with Applications. Princeton University Press, USA (2010)
11. Meurant, G., Tichý, P.: Error Norm Estimation in the Conjugate Gradient Algorithm. SIAM Spotlights, vol. 6. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, USA (2024)
12. Meurant, G.: The Lanczos and Conjugate Gradient Algorithms. Software, Environments, and Tools, vol. 19. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, USA (2006). From theory to finite precision computations
13. Saad, Y.: Iterative Methods for Sparse Linear Systems, 2nd edn. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA (2003)
14. Šimonová, D., Tichý, P.: When does the Lanczos algorithm compute exactly? Electron. Trans. Numer. Anal. **55**, 547–567 (2022)
15. Underwood, R.R.: An iterative block Lanczos method for the solution of large sparse symmetric eigenproblems. Phd thesis, Stanford University, USA, Stanford University, USA (1975)
16. Golub, G.H., Underwood, R.: The block Lanczos method for computing eigenvalues. In: Mathematical Software, III (Proc. Sympos., Math. Res. Center, Univ. Wisconsin, Madison, Wis., 1977), pp. 361–37739 (1977)
17. Ruhe, A.: Implementation aspects of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices. Math. Comp. **33**(146), 680–687 (1979)
18. Broyden, C.G.: A breakdown of the block CG method. Optim. Methods Softw. **7**(1), 41–55 (1996)
19. Freund, R.W., Malhotra, M.: A block QMR algorithm for non-Hermitian linear systems with multiple right-hand sides. Linear Algebra Appl. **254**, 119–157 (1997)
20. Paige, C.C.: Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem. Linear Algebra Appl. **34**, 235–258 (1980)
21. Greenbaum, A.: Behavior of slightly perturbed Lanczos and conjugate-gradient recurrences. Linear Algebra Appl. **113**, 7–63 (1989)
22. Greenbaum, A., Strakoš, Z.: Predicting the behavior of finite precision Lanczos and conjugate gradient computations. SIAM J. Matrix Anal. Appl. **13**(1), 121–137 (1992)
23. Strakoš, Z., Tichý, P.: On error estimation in the conjugate gradient method and why it works in finite precision computations. Electron. Trans. Numer. Anal. **13**, 56–80 (2002)
24. Strakoš, Z., Tichý, P.: Error estimation in preconditioned conjugate gradients. BIT **45**(4), 789–817 (2005)
25. Hestenes, M.R., Stiefel, E.: Methods of conjugate gradients for solving linear systems. J. Research Nat. Bur. Standards **49**, 409–4361953 (1952)
26. Shao, M.: Householder orthogonalization with a nonstandard inner product. SIAM J. Matrix Anal. Appl. **44**(2), 481–502 (2023)
27. Grigori, L., Tissot, O.: Scalable linear solvers based on enlarged Krylov subspaces with dynamic reduction of search directions. SIAM J. Sci. Comput. **41**(5), 522–547 (2019)
28. Meurant, G., Tichý, P.: On computing quadrature-based bounds for the A-norm of the error in conjugate gradients. Numer. Algorithms **62**(2), 163–191 (2013)