A REAL-VALUED BLOCK CONJUGATE GRADIENT TYPE METHOD FOR SOLVING COMPLEX SYMMETRIC LINEAR SYSTEMS WITH MULTIPLE RIGHT-HAND SIDES

Yasunori Futamura, Takahiro Yano, Akira Imakura,
Tetsuya Sakurai, Tsukuba

Received January 31, 2017. First published July 7, 2017.

Abstract. We consider solving complex symmetric linear systems with multiple right-hand sides. We assume that the coefficient matrix has indefinite real part and positive definite imaginary part. We propose a new block conjugate gradient type method based on the Schur complement of a certain 2-by-2 real block form. The algorithm of the proposed method consists of building blocks that involve only real arithmetic with real symmetric matrices of the original size. We also present the convergence property of the proposed method and an efficient algorithmic implementation. In numerical experiments, we compare our method to a complex-valued direct solver, and a preconditioned and nonpreconditioned block Krylov method that uses complex arithmetic.

Keywords: linear system with multiple right-hand sides; complex symmetric matrices; block Krylov subspace methods

MSC 2010: 65F10, 65F50

1. Introduction

In this paper we consider solving large sparse complex symmetric linear systems with multiple right-hand sides of the form

$$(A_{\mathrm{R}} + \mathrm{i}A_{\mathrm{I}})(X + \mathrm{i}Y) = B_{\mathrm{R}} + \mathrm{i}B_{\mathrm{I}}, \tag{1.1}$$

where $A_{\mathrm{R}}, A_{\mathrm{I}} \in \mathbb{R}^{n\times n}$ are real symmetric matrices, in addition, A_{R} is indefinite and A_{I} is positive definite. Here i is the imaginary unit and $B_{\mathrm{R}}, B_{\mathrm{I}} \in \mathbb{R}^{n\times s}$ are the real

This work was supported in part by JST/CREST, JST/ACT-I (Grant No. JPMJPR16U6), MEXT KAKENHI (Grant Nos. 25286097, 17K12690), and University of Tsukuba Basic Research Support Program Type A.

DOI: 10.21136/AM.2017.0023-17

and imaginary parts of the right-hand side. The matrices $X, Y \in \mathbb{R}^{n\times s}$ are the real part and imaginary part of the unknown. We assume that $s \ll n$, and $A_{\mathrm{R}} + \mathrm{i}A_{\mathrm{I}}$ is nonsingular.

This typical kind of linear systems arises in several application areas such as computation of Green's function in electronic structure calculations and spectral filtering in the algorithms of contour integral eigenvalue solvers such as the Sakurai-Sugiura method [26] and its variants [18], [19], [20], [27], and the FEAST algorithm [24]. It also needs to be solved in the algorithm of a stochastic estimation method for eigenvalue distribution [17].

There are many methods for solving (1.1). One of the most robust methods is a direct solver using complex symmetric $\mathrm{LDL}^{\mathrm{T}}$ (modified Cholesky) factorization.

One can utilize the preconditioned Krylov subspace methods which take advantage of the complex symmetry such as the conjugate orthogonal conjugate gradient (COCG) method [31], the conjugate orthogonal conjugate residual (COCR) method [28], and the quasi-minimal residual method for complex symmetric systems [16]. These methods iterate with short-term recurrences. Thus computational cost for every iteration is constant. However, they have no optimal characteristic.

If one can accept long-term recurrences, optimal Krylov subspace methods such as the generalized minimal residual method (GMRES) [25] and the generalized conjugate residual method (GCR) [14] can be used. While these methods have optimality, their computational complexities for one iteration are more expensive than the methods above that exploit the complex symmetry and employ short-term recurrences.

The (preconditioned) Hermitian and skew-Hermitian splitting iteration method (HSS) [6], [7] and its variants specialized for complex symmetric systems [3], [4], [5] were proposed. The variants for complex symmetric systems consider the case where both the real and imaginary parts of the coefficient matrix are positive semi-definite with at least one of them being positive definite, however, this is not the case for our study.

For the case where multiple right-hand sides are given simultaneously, we can utilize block Krylov subspace methods. The block Krylov subspace methods are extensions of Krylov subspace methods for systems of multiple right-hand sides. For complex symmetric systems, the block COCG method [29] was proposed. It is empirically shown that block Krylov subspace methods can be faster than a standard Krylov subspace method sequentially applied to each right-hand side.

There are several ways to convert the complex problem (1.1) to an equivalent 2-by-2 real block form. For example, one can convert (1.1) to

$$\begin{pmatrix} A_{\mathrm{R}} & -A_{\mathrm{I}} \\ A_{\mathrm{I}} & A_{\mathrm{R}} \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} B_{\mathrm{R}} \\ B_{\mathrm{I}} \end{pmatrix}. \tag{1.2}$$

If one needs a symmetric system,

$$\begin{pmatrix} A_{\mathrm{R}} & A_{\mathrm{I}} \\ A_{\mathrm{I}} & -A_{\mathrm{R}} \end{pmatrix} \begin{pmatrix} X \\ -Y \end{pmatrix} = \begin{pmatrix} B_{\mathrm{R}} \\ B_{\mathrm{I}} \end{pmatrix} \tag{1.3}$$

can be used. Spectral characteristics of (1.2), (1.3), and other formulations and analysis of preconditioners for the systems were studied in [10].

Axelsson et al. [1] proposed a real-valued method based on the Schur complement of the typical 2-by-2 real block form. The method involves only real arithmetic with matrices and vectors of order n. However, in [1] it is assumed that A_{R} is positive definite and A_{I} is positive semi-definite. This is not the case for our study. In [2], an extension and analysis of the method for the general nonsymmetric case were investigated, however, the study did not consider taking advantage of the symmetry and/or positive definiteness of the matrices. In this paper we propose a new method for solving (1.1), where A_{R} is indefinite and A_{I} is positive definite. Our method is derived by extending the work in [1] and combining it with the block conjugate gradient method for dealing with multiple right-hand sides simultaneously.

The paper is organized as follows. In Section 2, we describe real-valued formulation and the base algorithm for our method. Characteristics of the matrices that play an important role in this study are presented. In Section 4, we present a new block conjugate gradient type method that involves only real arithmetic. A bound on a certain error norm of the method is also described to analyze the convergence rate. In Section 5, an efficient and stable algorithmic implementation of our method is presented. In Section 6, we show numerical experiments to investigate the performance of our method by comparing with the complex $\mathrm{LDL}^{\mathrm{T}}$ factorization and the (preconditioned) block COCG method. In Section 7, the conclusion and future work are shown.

2. Real-valued formulation

In this section we describe the real-valued formulation and the base algorithm. We now recall the real-valued base algorithm [1] for solving the complex symmetric linear system. Here we consider a 2-by-2 real block form (1.3) which is equivalent to (1.1). The equation (1.3) can be converted to

$$\begin{pmatrix} C_\gamma & 0 \\ \sqrt{1+\gamma^2}A_{\mathrm{I}} & -(A_{\mathrm{R}}+\gamma A_{\mathrm{I}}) \end{pmatrix} \begin{pmatrix} X \\ \frac{1}{\sqrt{1+\gamma^2}}(\gamma X - Y) \end{pmatrix} = \begin{pmatrix} F \\ \frac{\gamma}{\sqrt{1+\gamma^2}}(B_{\mathrm{I}} - \gamma B_{\mathrm{R}}) \end{pmatrix},$$

where

$$C_\gamma := A_{\mathrm{R}} - \gamma A_{\mathrm{I}} + (1+\gamma^2) A_{\mathrm{I}} (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1} A_{\mathrm{I}} \tag{2.1}$$

and

$$F := B_{\mathrm{R}} - A_{\mathrm{I}}(A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}(B_{\mathrm{I}} - \gamma B_{\mathrm{R}}).$$

Here $\gamma \in \mathbb{R}$ is such that $\det(A_{\mathrm{R}} + \gamma A_{\mathrm{I}}) \neq 0$. See [1] for the details.

Using the above relations, we can form the base algorithm which is described in Algorithm 1.

Algorithm 1. Base algorithm of real-valued method.

1: Compute $F = B_{\mathrm{R}} - A_{\mathrm{I}}(A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}(B_{\mathrm{I}} - \gamma B_{\mathrm{R}})$
2: Solve $C_\gamma X = F$
3: Compute $Y = \gamma X - (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}(\gamma B_{\mathrm{R}} - B_{\mathrm{I}} + (1+\gamma^2)A_{\mathrm{I}}X)$

The linear system

$$C_\gamma X = F \tag{2.2}$$

in the second line is solved with some iterative method. In practice, C_γ is never formed explicitly. To compute the product of C_γ with an $n \times s$ matrix (which is necessary for block Krylov methods), the procedure shown in Algorithm 2 can be used.

Algorithm 2. Procedure for computing $W = C_\gamma V$, where $V, W \in \mathbb{R}^{n\times s}$.

1: Compute $T_1 = A_{\mathrm{I}}V$
2: Compute $T_2 = (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}T_1$
3: Compute $W = A_{\mathrm{R}}V - \gamma T_1 + (1+\gamma^2)A_{\mathrm{I}}T_2$

In Algorithm 1, the matrix inverse $(A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}$ appears in lines 1 and 3, and also in C_γ. Thus an inner solver needs to be employed for a linear system involving $(A_{\mathrm{R}} + \gamma A_{\mathrm{I}})$, where $(A_{\mathrm{R}} + \gamma A_{\mathrm{I}})$ is real symmetric and generally, indefinite. For the inner solver, a direct solver using the real symmetric $\mathrm{LDL}^{\mathrm{T}}$ factorization can be an option. Preconditioned Krylov subspace methods can also be used.

In [1], it is described that when A_{R} is real symmetric positive definite (s.p.d.) and A_{I} is real symmetric positive semi-definite (s.p.s.d.), $(A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}$ is a good preconditioner for the linear system (2.2) with a proper choice of γ. If the maximal eigenvalue of $A_{\mathrm{R}}^{-1}A_{\mathrm{I}}$ is known, then we can use optimal γ to obtain the preconditioned matrix with small spectral condition number κ. Even if the maximal eigenvalue is not known, $\gamma = 1$ can be a reasonable choice, since we have $\kappa = 2$ in that case.

However, in this study, we assume that A_{R} is indefinite and A_{I} is positive definite. Therefore, in our case, the promising theorems presented in [1] do not hold. Moreover, the (preconditioned) CG method (which is the best known method for s.p.d. matrices) cannot be used straightforwardly because in our case, neither C_γ

nor $(A_{\mathrm{I}} + \gamma A_{\mathrm{I}})$ are positive definite. Since the coefficient matrix C_γ is symmetric indefinite, one can consider using the MINRES method [23] which is optimal with respect to the residual norm and employs preferable short-term recurrences. However, for MINRES, the preconditioning matrix needs to be s.p.d. To find good s.p.d. preconditioner is not an easy task. Instead of an optimal method with short-term recurrences, one can consider utilizing optimal Krylov methods using long-term recurrences such as GMRES, or nonoptimal methods using short-term recurrence such as BiCGSTAB [30].

Here, to show that we can utilize optimal Krylov methods with short-term recurrences for solving the linear system (2.2), we present the following proposition.

Proposition 2.1. *Consider $G_\gamma := (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1} C_\gamma$. $A_{\mathrm{I}} G_\gamma$ is real symmetric positive definite and G_γ is self-adjoint w.r.t. the A_{I}-inner product.*

P r o o f. Since A_{R} is real symmetric and A_{I} is s.p.d., there exist nonsingular $U \in \mathbb{R}^{n\times n}$ and diagonal $\Lambda \in \mathbb{R}^{n\times n}$ that satisfy $A_{\mathrm{R}} U = A_{\mathrm{I}} U \Lambda$ and $U^{-1} = U^{\mathrm{T}} A_{\mathrm{I}}$. Using them, we have

$$\begin{aligned} G_\gamma &= (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}[A_{\mathrm{R}} - \gamma A_{\mathrm{I}} + (1+\gamma^2) A_{\mathrm{I}} (A_{\mathrm{R}} + \alpha A_{\mathrm{I}})^{-1} A_{\mathrm{I}}] \\ &= [A_{\mathrm{I}}(A_{\mathrm{I}}^{-1} A_{\mathrm{R}} + \gamma I)]^{-1} A_{\mathrm{I}} [A_{\mathrm{I}}^{-1} A_{\mathrm{R}} - \gamma I + (1+\gamma^2)(A_{\mathrm{I}}^{-1} A_{\mathrm{R}} + \gamma I)^{-1}] \\ &= (A_{\mathrm{I}}^{-1} A_{\mathrm{R}} + \gamma I)^{-1} [A_{\mathrm{I}}^{-1} A_{\mathrm{R}} - \gamma I + (1+\gamma^2)(A_{\mathrm{I}}^{-1} A_{\mathrm{R}} + \gamma I)^{-1}] \\ &= U(\Lambda + \gamma I)^{-1} [\Lambda - \gamma I + (1+\gamma^2)(\Lambda + \gamma I)^{-1}] U^{-1} \\ &= U \Theta U^{-1}, \end{aligned}$$

where $\Theta := (\Lambda + \gamma I)^{-1}[\Lambda - \gamma I + (1+\gamma^2)(\Lambda + \gamma I)^{-1}]$. Since Θ is a diagonal matrix, an eigenvalue θ of G_γ is written as

$$\theta = \frac{\lambda - \gamma + (1+\gamma^2)/(\lambda+\gamma)}{\lambda + \gamma} = \frac{1+\lambda^2}{(\lambda+\gamma)^2},$$

where λ is a diagonal element of Λ. Therefore, all eigenvalues of G_γ are real positive (but G_γ is not symmetric). Consider $A_{\mathrm{I}} G_\gamma$. Since $U^{-1} = U^{\mathrm{T}} A_{\mathrm{I}}$, we have $A_{\mathrm{I}} G_\gamma = A_{\mathrm{I}} U \Theta (A_{\mathrm{I}} U)^{\mathrm{T}}$. By virtue of this relation, we notice that $A_{\mathrm{I}} G_\gamma$ is s.p.d. and $(G_\gamma \boldsymbol{x}, \boldsymbol{y})_{A_{\mathrm{I}}} = (\boldsymbol{x}, G_\gamma \boldsymbol{y})_{A_{\mathrm{I}}}$, where $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ and $(\boldsymbol{x}, \boldsymbol{y})_{A_{\mathrm{I}}} = \boldsymbol{x}^{\mathrm{T}} A_{\mathrm{I}} \boldsymbol{y}$. Proposition 2.1 is proved. □

Consequently, if we choose $G_\gamma := (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1} C_\gamma$ as the coefficient matrix rather than C_γ when considering the linear system

$$G_\gamma X = \widehat{F} := (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1} F \tag{2.3}$$

which is equivalent to (2.2), and employ a matrix-weighted inner product, we can utilize optimal Krylov subspace methods such as CG and MINRES.

3. Real-valued block conjugate gradient method with a matrix-weighted inner product

In this section we propose a new real-valued block conjugate gradient method for solving complex symmetric systems (1.1).

With a nonzero matrix $V = [\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \ldots, \boldsymbol{v}^{(s)}]$, we define

$$\begin{aligned}\mathcal{K}_m(A; \boldsymbol{v}^{(i)}) &:= \operatorname{span}\{\boldsymbol{v}^{(i)}, A\boldsymbol{v}^{(i)}, \ldots, A^{m-1}\boldsymbol{v}^{(i)}\},\\ \mathcal{B}_m(A; V) &:= \left\{ \sum_{i=1}^{s} \sum_{j=0}^{m-1} \eta_{j,i} A^j \boldsymbol{v}^{(i)} \,;\ \eta_{j,i} \in \mathbb{C} \quad \forall\, j, i \right\}\\ &= \mathcal{K}_m(A; \boldsymbol{v}^{(1)}) + \mathcal{K}_m(A; \boldsymbol{v}^{(2)}) + \ldots + \mathcal{K}_m(A; \boldsymbol{v}^{(s)}).\end{aligned}$$

Block Krylov subspace methods are projection methods which find the solution vectors as

$$\boldsymbol{x}_k^{(i)} \in \boldsymbol{x}_0^{(i)} + \mathcal{B}_k(A; R_0),$$

where $X_0 = [\boldsymbol{x}_0^{(1)}, \boldsymbol{x}_0^{(2)}, \ldots, \boldsymbol{x}_0^{(s)}]$ is the initial guess and $R_0 = B - AX_0$ (B is here the right-hand side matrix).

The block CG method [22] is an extension of the CG method for linear systems with multiple right-hand sides. The method can be utilized for real-symmetric/complex-Hermitian positive definite systems.

The residual vector $\boldsymbol{r}_k^{(i)}$ of the ith right-hand side at the kth iteration of the block CG method satisfies the condition

$$\boldsymbol{r}_k^{(i)} \perp \mathcal{B}_k(A; R_0).$$

In the block CG method, the search space is extended by s bases per iteration. The block CG method often requires fewer iteration counts than the CG method.

The naive formulation of the block CG method is instable because $n \times s$ matrices in the algorithm tend to be nearly linearly dependent. There are several treatments for this problem [22], [21], [12].

It is known that, the approximate solution $\boldsymbol{x}_k^{(i)}$ of the kth iteration of the block CG method minimizes

$$\|\boldsymbol{x} - \boldsymbol{x}_i^*\|_A$$

over all $\boldsymbol{x}$ such that $\boldsymbol{x} \in \boldsymbol{x}_0^{(i)} + \mathcal{B}_k(A; R_0)$, where $\boldsymbol{x}_i^*$ is the true solution for the ith right-hand side (see Theorem 2 of [22]). In [22], O'Leary presented a bound on the error norm of the block CG method.

Now we consider the block CG method with the A_{I}-inner product for solving (2.3). Since G_γ is self-adjoint w.r.t. the A_{I}-inner product, we can form a block Lanczos basis $V_m \in \mathbb{C}^{n\times sm}$ such that

$$V_m^{\mathrm{T}} A_{\mathrm{I}} V_m = I$$

and

$$V_m^{\mathrm{T}} A_{\mathrm{I}} G_\gamma V_m = T_m.$$

Here $T_m \in \mathbb{C}^{sm\times sm}$ is a symmetric block tridiagonal matrix with block size s.

Therefore, the algorithm of the block CG method can be derived analogously by referencing [22] and replacing the standard inner product by the A_{I}-inner product. The pseudo-code of the method is shown in Algorithm 3. This is a naive implementation. We present an efficient algorithmic implementation of the method in Section 5.

In [22], O'Leary also proposed the block BiCG method for solving non-Hermitian systems with multiple right-hand sides. Du et al. [11] presented a stabilized formulation which is called the block BiCGrQ method using Dubrulle's idea [12] developed for the block CG method. The block BiCGrQ method can be specialized for solving complex symmetric linear systems. This specialization leads to the block COCGrQ method.

In the numerical experiments, we compare our method to the (preconditioned) block COCGrQ method. We show the pseudo-code of the preconditioned block COCGrQ method in the appendix.

Algorithm 3. Real-valued block CG algorithm for solving complex symmetric linear systems with multiple right-hand sides (naive version). The definition of C_γ is given in (2.1). C_γ is never formed explicitly. The products of C_γ by matrices are computed by the procedure in Algorithm 2. If the $\mathrm{LDL}^{\mathrm{T}}$ factorization of $(A_{\mathrm{R}}+\gamma A_{\mathrm{I}})$ is given, the multiplication by C_γ can be computed using only triangular solves and matrix-matrix products.

Input: $A_{\mathrm{R}} = A_{\mathrm{R}}^{\mathrm{T}} \in \mathbb{R}^{n\times n}$; $A_{\mathrm{I}} = A_{\mathrm{I}}^{\mathrm{T}} \in \mathbb{R}^{n\times n}$; $B_{\mathrm{R}}, B_{\mathrm{I}}, X_0 \in \mathbb{R}^{n\times s}$; $\gamma \in \mathbb{R}$

Output: $X, Y \in \mathbb{R}^{n\times s}$

$F = B_{\mathrm{R}} - A_{\mathrm{I}}(A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}(B_{\mathrm{I}} - \gamma B_{\mathrm{R}})$

$\widehat{R}_0 = (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}(F - C_\gamma X_0)$

$P_0 = \widehat{R}_0$; $Z_0 = A_{\mathrm{I}}\widehat{R}_0$

for $k = 0, 1, \ldots$ until solution converges **do**

  $\acute{S}_k = (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1} C_\gamma P_k$

  $\acute{W}_k = A_{\mathrm{I}}\acute{S}_k$

  $\alpha_k = (P_k^{\mathrm{T}}\acute{W}_k)^{-1}(\widehat{R}_k^{\mathrm{T}} Z_k)$

  $X_{k+1} = X_k + P_k\alpha_k$

9: $\widehat{R}_{k+1} = \widehat{R}_k - \acute{S}_k \alpha_k$
10: $Z_{k+1} = A_{\mathrm{I}} \widehat{R}_{k+1}$
11: $\beta_k = (\widehat{R}_k^{\mathrm{T}} Z_k)^{-1} (\widehat{R}_{k+1}^{\mathrm{T}} Z_{k+1})$
12: $P_{k+1} = \widehat{R}_{k+1} + P_k \beta_k$
13: **end for**
14: $X = X_k$
15: $Y = \gamma X_k - (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1} (\gamma B_{\mathrm{R}} - B_{\mathrm{I}} + (1+\gamma^2) A_{\mathrm{I}} X_k)$

4. Convergence of the block CG method in A_{I}-inner product

In this section we derive a theorem on convergence of the error norm of the block CG method with the matrix-weighted inner product.

Here we order the eigenvalues of G_γ as $0 < \theta_1 \leqslant \theta_2 \leqslant \ldots \leqslant \theta_n$. Let $\boldsymbol{u}_i$ be the eigenvector of G_γ corresponding to θ_i and $U := [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n]$. We extend O'Leary's results [22] to the block CG method in the general inner product with A_{I}.

Let $C_k(\theta) := \cos(k \arccos(\theta))$ for $-1 \leqslant \theta \leqslant 1$ be the kth Chebyshev polynomial of the first kind. We have the following identities.

Lemma 4.1 (Lemma 4 of [22]). *For $\theta > 1$,*

$$\frac{C_k(\theta)}{C_{k-1}(\theta)} \leqslant 2\theta.$$

Lemma 4.2 (Lemma 3 of [22]). *Let*

$$\widehat{C}_k(\theta) := C_k\Big(\frac{d_2 + d_1 - 2\theta}{d_2 - d_1}\Big) \Big/ C_k\Big(\frac{d_2 + d_1}{d_2 - d_1}\Big),$$
$$\widehat{C}'_k(\theta) := C_{k-1}\Big(\frac{d_2 + d_1 - 2\theta}{d_2 - d_1}\Big) \Big/ C_k\Big(\frac{d_2 + d_1}{d_2 - d_1}\Big),$$

where $0 < d_1 < d_2$. Then, for $d_1 \leqslant \theta \leqslant d_2$,

$$|\widehat{C}_k(\theta)| \leqslant 2\bigg(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\bigg)^k, \quad |\widehat{C}'_k(\theta)| \leqslant 2\bigg(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\bigg)^k,$$

where $\kappa = d_2/d_1$.

Using the Chebyshev polynomial, we analyze the convergence of the block CG method in A_{I} inner product for solving (2.3).

Lemma 4.3. *Let $E := [\boldsymbol{e}_0^{(1)}, \boldsymbol{e}_0^{(2)}, \dots, \boldsymbol{e}_0^{(s)}] := X_0 - X^*$ be an initial error matrix, where $X^* := G_\gamma^{-1}\widehat{F}$. Define $\widetilde{E} := [\boldsymbol{e}_0^{(1)}, \boldsymbol{e}_0^{(2)}, \dots, \boldsymbol{e}_0^{(s-1)}]$ to be E with the sth column omitted. Assume that the top $(s-1)\times(s-1)$ submatrix of $U^{-1}G_\gamma\widetilde{E}$ is nonsingular. Define*

$$\Phi = \begin{bmatrix} \Phi_1 \\ \Phi_2 \end{bmatrix} \in \mathbb{R}^{n\times(s-1)}, \quad \Phi_1\text{: nonsingular}$$

to be an $n\times(s-1)$ matrix such that the columns of $U\Phi$ form a basis of $\mathrm{span}\{G_\gamma\widetilde{E}\}$ and

$$\widetilde{R} = [\widetilde{\boldsymbol{r}}^{(1)}, \widetilde{\boldsymbol{r}}^{(2)}, \dots, \widetilde{\boldsymbol{r}}^{(s-1)}] = U\Phi\Phi_1^{-1} = U\begin{bmatrix} I \\ \Phi_2\Phi_1^{-1} \end{bmatrix}.$$

Now, define $\boldsymbol{g}_k^{(j)} \in \mathcal{B}_k(G_\gamma; G_\gamma\widetilde{E}) = \mathrm{span}\{G_\gamma\widetilde{E}, G_\gamma^2\widetilde{E}, \dots, G_\gamma^k\widetilde{E}\}$ for $j = 1, 2, \dots, s-1$ by

$$\boldsymbol{g}_k^{(j)} = \mathcal{P}_k^{(j)}(G_\gamma)\widetilde{\boldsymbol{r}}^{(j)}$$

with a $(k-1)$st degree polynomial

$$\mathcal{P}_k^{(j)}(\theta) = C_{k-1}\Big(\frac{\theta_n+\theta_s-2\theta}{\theta_n-\theta_s}\Big)\Big/C_{k-1}\Big(\frac{\theta_n+\theta_s-2\theta_j}{\theta_n-\theta_s}\Big).$$

Then $\boldsymbol{g}_k^{(j)}$ can be written as

$$\boldsymbol{g}_k^{(j)} = \boldsymbol{u}_j + \sum_{i=s}^{n}\sigma_{ik}^{(j)}\boldsymbol{u}_i \tag{4.1}$$

with

$$\sigma_{ik}^{(j)} = \widetilde{\boldsymbol{u}}_i^{\mathrm{T}}\widetilde{\boldsymbol{r}}^{(j)}C_{k-1}\Big(\frac{\theta_n+\theta_s-2\theta_i}{\theta_n-\theta_s}\Big)\Big/C_{k-1}\Big(\frac{\theta_n-\theta_s-2\theta_j}{\theta_n-\theta_s}\Big)$$

for $i = s, s+1, \dots, n$, where $U^{-1} = [\widetilde{\boldsymbol{u}}_1, \widetilde{\boldsymbol{u}}_2, \dots, \widetilde{\boldsymbol{u}}_n]^{\mathrm{T}}$.

P r o o f. From the definition of $\widetilde{\boldsymbol{r}}^{(j)}$ we have

$$\widetilde{\boldsymbol{r}}^{(j)} = \boldsymbol{u}_j + \sum_{i=s}^{n}\varphi_{ij}\boldsymbol{u}_i,$$

where φ_{ij} is the (i,j) element of $\Phi\Phi_1^{-1}$. Notice that $\varphi_{ij} = \widetilde{\boldsymbol{u}}_i^{\mathrm{T}}\widetilde{\boldsymbol{r}}^{(j)}$. Then, using $G_\gamma = U\Theta U^{-1}$ and $\mathcal{P}_k^{(j)}(\theta_j) = 1$,

$$\begin{aligned}\boldsymbol{g}_k^{(j)} &= \Big(\sum_{i=1}^{n}\boldsymbol{u}_i\mathcal{P}_k^{(j)}(\theta_i)\widetilde{\boldsymbol{u}}_i^{\mathrm{T}}\Big)\Big(\boldsymbol{u}_j + \sum_{i=s}^{n}(\widetilde{\boldsymbol{u}}_i^{\mathrm{T}}\widetilde{\boldsymbol{r}}^{(j)})\boldsymbol{u}_i\Big)\\ &= \boldsymbol{u}_j + \sum_{i=s}^{n}(\widetilde{\boldsymbol{u}}_i^{\mathrm{T}}\widetilde{\boldsymbol{r}}^{(j)})\mathcal{P}_k^{(j)}(\theta_i)\boldsymbol{u}_i,\end{aligned}$$

which proves Lemma 4.3. □

Theorem 4.1. *Suppose that* $\dim(\mathcal{B}_k(G_\gamma; G_\gamma\widetilde{E})) = k(s-1)$. *Define the* k*th error vector of the block CG method as*

$$\boldsymbol{e}_k^{(s)} = \boldsymbol{x}_k^{(s)} - \boldsymbol{x}_s^* = \boldsymbol{e}_0^{(s)} + \sum_{j=1}^{s} \mathcal{P}_{kj}^*(G_\gamma) G_\gamma \boldsymbol{e}_0^{(j)},$$
$$\boldsymbol{e}_0^{(s)} = \boldsymbol{x}_0^{(s)} - \boldsymbol{x}_s^* = U\boldsymbol{y},$$

where each $\mathcal{P}_{kj}^*(G_\gamma)$ *is a* $(k-1)$*-degree polynomial and* $\boldsymbol{y} = [y_1, y_2, \ldots, y_n]^{\mathrm{T}}$.

Then, for convergence of the block CG method, we have

$$\|\boldsymbol{e}_k^{(s)}\|_{A_{\mathrm{I}}G_\gamma}^2 = (\boldsymbol{e}_k^{(s)})^{\mathrm{T}} A_{\mathrm{I}} G_\gamma \boldsymbol{e}_k^{(s)} \leqslant \left(\frac{\sqrt{\kappa_s}-1}{\sqrt{\kappa_s}+1}\right)^{2k} c$$

with some constant c, *where* $\kappa_s = \theta_n/\theta_s$.

P r o o f. Since the block CG method constructs the optimal polynomials in terms of $\|\boldsymbol{e}_k^{(s)}\|_{A_{\mathrm{I}}G_\gamma}^2$, we can bound it with specific polynomials, i.e.,

$$\|\boldsymbol{e}_k^{(s)}\|_{A_{\mathrm{I}}G_\gamma}^2 \leqslant \|\widetilde{\boldsymbol{e}}_k^{(s)}\|_{A_{\mathrm{I}}G_\gamma}^2, \quad \widetilde{\boldsymbol{e}}_k^{(s)} = \boldsymbol{e}_0^{(s)} + \sum_{j=1}^{s} \mathcal{P}_{kj}(G_\gamma) G_\gamma \boldsymbol{e}_0^{(j)}.$$

Here, we define the polynomial $\mathcal{P}_{ks}(\theta)$ as

$$1 + \mathcal{P}_{ks}(\theta)\theta = \widehat{C}_k(\theta) := C_k\Big(\frac{\theta_n + \theta_s - 2\theta}{\theta_n - \theta_s}\Big) \Big/ C_k\Big(\frac{\theta_n + \theta_s}{\theta_n - \theta_s}\Big).$$

Because $\widehat{C}_k(\theta)$ is a kth degree polynomial with $\widehat{C}_k(0) = 1$, it can be written as $1 + \mathcal{P}_{ks}(\theta)\theta$ with the $(k-1)$st degree of the polynomial $\mathcal{P}_{ks}(\theta)$. We also define $\mathcal{P}_{kj}(\theta)$ $(j = 1, 2, \ldots, s-1)$ as

$$\sum_{j=1}^{s-1} \mathcal{P}_{kj}(G_\gamma) G_\gamma \boldsymbol{e}_0^{(j)} = -\sum_{j=1}^{s-1} \widehat{C}_k(\theta_j) y_j \boldsymbol{g}_k^{(j)}, \quad j = 1, 2, \ldots, s-1.$$

From the assumption, $k(s-1)$ basis vectors of $\mathcal{B}_k(G_\gamma; G_\gamma\widetilde{E})$ are independent. Therefore, the polynomials $\mathcal{P}_{kj}(\theta)$ $(j = 1, 2, \ldots, s-1)$ are determined uniquely.

Then we have

$$\widetilde{\boldsymbol{e}}_k^{(s)} = \widehat{C}_k(G_\gamma)\boldsymbol{e}_0^{(s)} - \sum_{j=1}^{s-1} \widehat{C}_k(\theta_j) y_j \boldsymbol{g}_k^{(j)} = \widehat{C}_k(G_\gamma) U\boldsymbol{y} - \sum_{j=1}^{s-1} \widehat{C}_k(\theta_j) y_j \boldsymbol{g}_k^{(j)}.$$

Using (4.1), we conclude that

$$\begin{aligned}\widetilde{\boldsymbol{e}}_k^{(s)} &= \sum_{i=1}^{n}\widehat{C}_k(\theta_i)y_i\boldsymbol{u}_i - \sum_{j=1}^{s-1}\widehat{C}_k(\theta_j)y_j\Big(\boldsymbol{u}_j+\sum_{i=s}^{n}\sigma_{ij}\boldsymbol{u}_i\Big)\\ &= \sum_{i=s}^{n}\widehat{C}_k(\theta_i)y_i\boldsymbol{u}_i - \sum_{j=1}^{s-1}\widehat{C}_k(\theta_j)y_j\sum_{i=s}^{n}\sigma_{ij}\boldsymbol{u}_i = \sum_{i=s}^{n}\Big(\widehat{C}_k(\theta_i)y_i - \sum_{j=1}^{s-1}\widehat{C}_k(\theta_j)y_j\sigma_{ij}\Big)\boldsymbol{u}_i.\end{aligned}$$

Therefore,

$$\|\widetilde{\boldsymbol{e}}_k^{(s)}\|_{A_{\mathrm{I}}G_\gamma}^2 = \Big(\sum_{i=s}^{n}\widehat{C}_k(\theta_i)y_i\boldsymbol{u}_i\Big)^{\mathrm{T}}A_{\mathrm{I}}G_\gamma\Big(\sum_{l=s}^{n}\widehat{C}_k(\theta_l)y_l\boldsymbol{u}_l\Big) \tag{4.2}$$

$$+\Big(\sum_{i=s}^{n}\sum_{j=1}^{s-1}\widehat{C}_k(\theta_j)y_j\sigma_{ij}\boldsymbol{u}_i\Big)^{\mathrm{T}}A_{\mathrm{I}}G_\gamma\Big(\sum_{l=s}^{n}\sum_{m=1}^{s-1}\widehat{C}_k(\theta_m)y_m\sigma_{lm}\boldsymbol{u}_l\Big) \tag{4.3}$$

$$-2\Big(\sum_{i=s}^{n}\widehat{C}_k(\theta_i)y_i\boldsymbol{u}_i\Big)^{\mathrm{T}}A_{\mathrm{I}}G_\gamma\Big(\sum_{l=s}^{n}\sum_{m=1}^{s-1}\widehat{C}_k(\theta_m)y_m\sigma_{lm}\boldsymbol{u}_l\Big). \tag{4.4}$$

Next, we consider the bound of each term using Lemmas 4.1 and 4.2 and

$$\max_{j=1,2,\ldots,s-1}\frac{\theta_n+\theta_s-2\theta_j}{\theta_n-\theta_s} = \frac{\theta_n+\theta_s-2\theta_1}{\theta_n-\theta_s}.$$

The 1st term (4.2) is bounded as

$$\begin{aligned}&\Big|\Big(\sum_{i=s}^{n}\widehat{C}_k(\theta_i)y_i\boldsymbol{u}_i\Big)^{\mathrm{T}}A_{\mathrm{I}}G_\gamma\Big(\sum_{l=s}^{n}\widehat{C}_k(\theta_l)y_l\boldsymbol{u}_l\Big)\Big|\\ &\quad= \Big|\sum_{i=s}^{n}(\widehat{C}_k(\theta_i))^2\theta_i y_i^2\Big| \leqslant 4\Big(\frac{\sqrt{\kappa_s}-1}{\sqrt{\kappa_s}+1}\Big)^{2k}\Big|\sum_{i=s}^{n}\theta_i y_i^2\Big| = \Big(\frac{\sqrt{\kappa_s}-1}{\sqrt{\kappa_s}+1}\Big)^{2k}c_1.\end{aligned}$$

The 2nd term (4.3) is bounded as

$$\begin{aligned}&\Big|\Big(\sum_{i=s}^{n}\sum_{j=1}^{s-1}\widehat{C}_k(\theta_j)y_j\sigma_{ij}\boldsymbol{u}_i\Big)^{\mathrm{T}}A_{\mathrm{I}}G_\gamma\Big(\sum_{l=s}^{n}\sum_{m=1}^{s-1}\widehat{C}_k(\theta_m)y_m\sigma_{lm}\boldsymbol{u}_l\Big)\Big|\\ &\quad= \Big|\sum_{i=s}^{n}\Big(\sum_{j=1}^{s-1}\frac{C_k\big(\frac{\theta_n+\theta_s-2\theta_j}{\theta_n-\theta_s}\big)}{C_k\big(\frac{\theta_n+\theta_s}{\theta_n-\theta_s}\big)}y_j\widetilde{\boldsymbol{u}}_i^{\mathrm{T}}\widetilde{\boldsymbol{r}}_j\frac{C_{k-1}\big(\frac{\theta_n+\theta_s-2\theta_i}{\theta_n-\theta_s}\big)}{C_{k-1}\big(\frac{\theta_n+\theta_s-2\theta_j}{\theta_n-\theta_s}\big)}\Big)^2\theta_i\Big|\\ &\quad\leqslant 4\Big(\frac{\theta_n+\theta_s-2\theta_1}{\theta_n-\theta_s}\Big)^2 4\Big(\frac{\sqrt{\kappa_s}-1}{\sqrt{\kappa_s}+1}\Big)^{2k}\Big|\sum_{i=s}^{n}\Big(\sum_{j=1}^{s-1}y_j\widetilde{\boldsymbol{u}}_i^{\mathrm{T}}\widetilde{\boldsymbol{r}}^{(j)}\Big)^2\theta_i\Big|\\ &\quad= 4\Big(\frac{\theta_n+\theta_s-2\theta_1}{\theta_n-\theta_s}\Big)^2\Big(\frac{\sqrt{\kappa_s}-1}{\sqrt{\kappa_s}+1}\Big)^{2k}c_2.\end{aligned}$$

The 3rd term (4.4) is bounded as

$$\begin{aligned}
&\Big|2\Big(\sum_{i=s}^{n}\widehat{C}_k(\theta_i)y_i\boldsymbol{u}_i\Big)^{\mathrm{T}}A_{\mathrm{I}}G_\gamma\Big(\sum_{l=s}^{n}\sum_{m=1}^{s-1}\widehat{C}_k(\theta_m)y_m\sigma_{lm}\boldsymbol{u}_l\Big)\Big| \\
&\quad=\Big|2\sum_{i=s}^{n}\widehat{C}_k(\theta_i)y_i\theta_i\Big(\sum_{j=1}^{s-1}\frac{C_k\big(\frac{\theta_n+\theta_s-2\theta_j}{\theta_n-\theta_s}\big)}{C_k\big(\frac{\theta_n+\theta_s}{\theta_n-\theta_s}\big)}y_j\widetilde{\boldsymbol{u}}_i^{\mathrm{T}}\widetilde{\boldsymbol{r}}_j\frac{C_{k-1}\big(\frac{\theta_n+\theta_s-2\theta_i}{\theta_n-\theta_s}\big)}{C_{k-1}\big(\frac{\theta_n+\theta_s-2\theta_j}{\theta_n-\theta_s}\big)}\Big)\Big| \\
&\quad\leqslant 4\Big(\frac{\theta_n+\theta_s-2\theta_1}{\theta_n-\theta_s}\Big)4\Big(\frac{\sqrt{\kappa_s}-1}{\sqrt{\kappa_s}+1}\Big)^{2k}\Big|\sum_{i=s}^{n}\Big(\sum_{j=1}^{s-1}y_j\widetilde{\boldsymbol{u}}_i^{\mathrm{T}}\widetilde{\boldsymbol{r}}^{(j)}\Big)\theta_i\Big| \\
&\quad=4\Big(\frac{\theta_n+\theta_s-2\theta_1}{\theta_n-\theta_s}\Big)\Big(\frac{\sqrt{\kappa_s}-1}{\sqrt{\kappa_s}+1}\Big)^{2k}c_3.
\end{aligned}$$

Therefore, we have

$$\begin{aligned}
\|\widetilde{\boldsymbol{e}}_s^{(k)}\|_{A_{\mathrm{I}}G_\gamma}&\leqslant\Big(\frac{\sqrt{\kappa_s}-1}{\sqrt{\kappa_s}+1}\Big)^{2k}\Big[c_1+4\Big(\frac{\theta_n+\theta_s-2\theta_1}{\theta_n-\theta_s}\Big)^2c_2+4\Big(\frac{\theta_n+\theta_s-2\theta_1}{\theta_n-\theta_s}\Big)c_3\Big] \\
&=\Big(\frac{\sqrt{\kappa_s}-1}{\sqrt{\kappa_s}+1}\Big)^{2k}c.
\end{aligned}$$

Here we note that the constant c does not depend on k. □

Notice that Lemma 4.3 and Theorem 4.1 are valid for any component p using the definition $E^{(p)}=[\boldsymbol{e}_0^{(1)},\dots,\boldsymbol{e}_0^{(p-1)},\boldsymbol{e}_0^{(p+1)},\dots,\boldsymbol{e}_0^{(s)}]$.

In Theorem 4.1, when $s=1$, we have $\kappa_s=\theta_n/\theta_1$, and the theorem is reduced to the theorem for the convergence of the error norm of the single-vector CG method (with A_{I}-inner product). Note here that θ_n/θ_1 is the spectral condition number of G_γ. Therefore, when compared with the single-vector CG method, we can see that for the block CG method with s right-hand sides, in terms of the convergence, the $s-1$ smallest eigenvalues can be ignored when computing the spectral condition number of G_γ. In other words, we can say that the block CG method deflates the $s-1$ smallest eigenvalues of G_γ.

5. Efficient algorithmic implementation

In this section we describe the efficient implementation of our method. In Algorithm 3, at a first glance, we can see that we need to multiply A_{I} to multi-vectors ($n\times s$ matrices) four times per iteration (twice in C_γ and twice for computations of $\acute{W}_k$ and $\widehat{Z}_{k+1}$). Let us now consider keeping intermediate multi-vectors used for computing

$$C_\gamma P_k=A_{\mathrm{R}}P_k-\gamma A_{\mathrm{I}}P_k+(1+\gamma^2)A_{\mathrm{I}}(A_{\mathrm{R}}+\gamma A_{\mathrm{I}})^{-1}A_{\mathrm{I}}P_k.$$

Here we define $S_k := A_{\mathrm{I}}P_k$, $T_k := (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}S_k$, and $W_k := C_\gamma P_k$. Using them, we can compute α_k without $\acute{W}_k = A_{\mathrm{I}}\acute{S}_k$, since

$$\begin{aligned} P_k^{\mathrm{T}}\acute{W}_k &= P_k^{\mathrm{T}}A_{\mathrm{I}}\acute{S}_k = S_k^{\mathrm{T}}\acute{S}_k = S_k^{\mathrm{T}}(A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}C_\gamma P_k \\ &= ((A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}S_k)^{\mathrm{T}}C_\gamma P_k = T_k^{\mathrm{T}}W_k. \end{aligned}$$

In addition, W_k can be computed as $W_k = A_{\mathrm{R}}P_k - \gamma S_k + (1+\gamma)^2 A_{\mathrm{I}}T_k$ and S_k can be updated by a recurrence $S_{k+1} = Z_{k+1} + S_k\beta_k$. Therefore, we can reduce the multiplication of A_{I} from four times to twice.

The linear system we originally solve is $C_\gamma X = F$ (see Algorithm 1). To compute the residual $R := F - C_\gamma X$ cheaply, we introduce a recurrence $R_{k+1} = R_k - W_k\alpha_k$.

Provided that the initial guesses X_0 and Y_0 satisfy

$$Y_0 = \gamma X_0 - (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}(\gamma B_{\mathrm{R}} - B_{\mathrm{I}} + (1+\gamma^2)A_{\mathrm{I}}X_0), \tag{5.1}$$

Y_k can be updated by the recurrences

$$\begin{aligned} Y_{k+1} &= Y_k + \gamma P_k\alpha_k - (1+\gamma^2)(A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}A_{\mathrm{I}}P_k\alpha_k \\ &= Y_k + (\gamma P_k - (1+\gamma^2)T_k)\alpha_k, \end{aligned} \tag{5.2}$$

since $X_{k+1} = X_k + P_k\alpha_k$, and the real part X^* and the imaginary part Y^* of the true solution of (1.1) satisfy

$$Y^* = \gamma X^* - (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}(\gamma B_{\mathrm{R}} - B_{\mathrm{I}} + (1+\gamma^2)A_{\mathrm{I}}X^*)$$

(see [1] for the derivation).

As seen in other Krylov methods, the residual R_k computed by the recurrence formula could be different from the true residual $F - C_\gamma X_k$ due to the rounding error. Thus the true residual has to be computed (at least) at the end of the algorithm to assess the accuracy of the approximate solution.

Fortunately, the true residual can be computed cheaply with $B_{\mathrm{R}} - A_{\mathrm{R}}X + A_{\mathrm{I}}Y$, since

$$\begin{aligned} R &= F - C_\gamma X \\ &= B_{\mathrm{R}} - A_{\mathrm{I}}(A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}(B_{\mathrm{I}} - \gamma B_{\mathrm{R}}) \\ &\quad - (A_{\mathrm{R}} - \gamma A_{\mathrm{I}} + (1+\gamma^2)A_{\mathrm{I}}(A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}A_{\mathrm{I}})X \\ &= B_{\mathrm{R}} - A_{\mathrm{R}}X + A_{\mathrm{I}}(\gamma X - (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}(\gamma B_{\mathrm{R}} - B_{\mathrm{I}} + (1+\gamma^2)A_{\mathrm{I}}X)) \\ &= B_{\mathrm{R}} - A_{\mathrm{R}}X + A_{\mathrm{I}}Y. \end{aligned} \tag{5.3}$$

We can use this equation to avoid the expensive multiplication of C_γ for computing the true residual. Using the relation, we can also compute the initial residual R_0 cheaply. This idea is not new (see [1]).

Utilizing the relations (5.3) and (5.2), we can reduce $(A_\mathrm{R} + \gamma A_\mathrm{I})^{-1}$ in the initialization and finalization phase (lines 1–2 and 15) from four times to twice.

Similar to other block Krylov subspace methods, our method shows instability since column vectors $n \times s$ matrices tend to be nearly linearly dependent. As presented in [12] for the standard block CG method, one can orthogonalize the columns of R_k by the (thin) QR factorization ($\mathrm{qr}(\cdot)$) for stabilizing the method.

In the stabilized formulation, we introduce a column orthonormal matrix $Q_k \in \mathbb{R}^{n\times s}$ such that $Q_k\Delta_k = R_k$, where $\Delta_k \in \mathbb{R}^{s\times s}$. The matrix Q_k is used for the iteration instead of R_k for numerical stability and R_k is not orthonormalized explicitly. The matrix Q_k and upper triangular $\varrho_k \in \mathbb{R}^{s\times s}$ are recurrently computed as $Q_k\varrho_k = \mathrm{qr}(Q_{k-1} - \widetilde{W}_{k-1}\widetilde{\alpha}_{k-1})$, where $\widetilde{\alpha}_k := \Delta_k\alpha_k\Delta_k^{-1}$, $\widetilde{W}_k := W_k\Delta_k^{-1}$, and Δ_k is updated as $\Delta_k = \varrho_k\Delta_{k-1}$. Note that the Frobenius norm of the residual can be computed via $\|\Delta_k\|_\mathrm{F}$, since $\|R_k\|_\mathrm{F} = \|Q_k\Delta_k\|_\mathrm{F} = \|\Delta_k\|_\mathrm{F}$.

In Algorithm 4, we show the resulting efficient algorithmic implementation with both the original formulation and the stabilized formulation with residual orthogonalization. If one chooses the stabilized formulation, the steps in bracket labeled with "rQ" need to be performed. Note that the symbols α_k, β_k, P_k, S_k, T_k, W_k, and Z_k in the "rQ" formulation are not mathematically equivalent to those of the original formulation. In this study, we refer to Algorithm 4 as the proposed method.

Note that it is possible to compute $\widehat{Q}_k$ such that $\widehat{Q}_k^\mathrm{T} Z_k = I_s$, where I_s is the identity matrix of order s. This enables one to avoid matrix inverses of small $s \times s$ matrices. It is also noted that ϱ_k do not have to be upper triangular. Thus any orthogonalization algorithms can be used.

Algorithm 4. Real-valued block CG algorithm for solving the complex symmetric linear system with multiple right-hand sides (Efficient version). $\mathrm{qr}(\cdot)$ indicates the thin QR decomposition.

Input: $A_\mathrm{R} = A_\mathrm{R}^\mathrm{T} \in \mathbb{R}^{n\times n}$; $A_\mathrm{I} = A_\mathrm{I}^\mathrm{T} \in \mathbb{R}^{n\times n}$; $B_\mathrm{R}, B_\mathrm{I}, X_0, Y_0 \in \mathbb{R}^{n\times s}$; $\gamma \in \mathbb{R}$
Output: $X, Y \in \mathbb{R}^{n\times s}$

1: **if** there is no initial guess **then**
2: $\quad X_0 = O$; $Y_0 = (A_\mathrm{R} + \gamma A_\mathrm{I})^{-1}(B_\mathrm{I} - \gamma B_\mathrm{R})$
3: **else**
4: $\quad$ ((5.1) must hold)
5: **end if**
6: $R_0 = B_\mathrm{R} - A_\mathrm{R}X_0 + A_\mathrm{I}Y_0$
7: {rQ: $Q_0\Delta_0 = \mathrm{qr}(R_0)$}

$\widehat{R}_0 = (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1} R_0$ $\{\mathrm{rQ}\text{: } \widehat{Q}_0 = (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1} Q_0\}$
$P_0 = \widehat{R}_0$; $S_0 = Z_0 = A_{\mathrm{I}} \widehat{R}_0$ $\{\mathrm{rQ}\text{: } P_0 = \widehat{Q}_0;\ S_0 = Z_0 = A_{\mathrm{I}} \widehat{Q}_0\}$
for $k = 0, 1, \ldots$ until solution converges **do**
  $T_k = (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1} S_k$
  $W_k = A_{\mathrm{R}} P_k - \gamma S_k + (1 + \gamma^2) A_{\mathrm{I}} T_k$
  $\alpha_k = (T_k^{\mathrm{T}} W_k)^{-1} (\widehat{R}_k^{\mathrm{T}} Z_k)$ $\{\mathrm{rQ}\text{: } \alpha_k = (T_k^{\mathrm{T}} W_k)^{-1} (\widehat{Q}_k^{\mathrm{T}} Z_k)\}$
  $X_{k+1} = X_k + P_k \alpha_k$ $\{\mathrm{rQ}\text{: } X_{k+1} = X_k + P_k \alpha_k \Delta_k\}$
  $Y_{k+1} = Y_k + (\gamma P_k - (1 + \gamma^2) T_k) \alpha_k$
  $\{\mathrm{rQ}\text{: } Y_{k+1} = Y_k + (\gamma P_k - (1 + \gamma^2) T_k) \alpha_k \Delta_k\}$
  $R_{k+1} = R_k - W_k \alpha_k$ $\{\mathrm{rQ}\text{: } Q_{k+1} \varrho_{k+1} = \mathrm{qr}(Q_k - W_k \alpha_k)\}$
  $\{\mathrm{rQ}\text{: } \Delta_{k+1} = \varrho_{k+1} \Delta_k\}$
  $\widehat{R}_{k+1} = (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1} R_{k+1}$ $\{\mathrm{rQ}\text{: } \widehat{Q}_{k+1} = (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1} Q_{k+1}\}$
  $Z_{k+1} = A_{\mathrm{I}} \widehat{R}_{k+1}$ $\{\mathrm{rQ}\text{: } Z_{k+1} = A_{\mathrm{I}} \widehat{Q}_{k+1}\}$
  $\beta_k = (\widehat{R}_k^{\mathrm{T}} Z_k)^{-1} (\widehat{R}_{k+1}^{\mathrm{T}} Z_{k+1})$ $\{\mathrm{rQ}\text{: } \beta_k = (\widehat{Q}_k^{\mathrm{T}} Z_k)^{-1} \varrho_{k+1}^{\mathrm{T}} (\widehat{Q}_{k+1}^{\mathrm{T}} Z_{k+1})\}$
  $P_{k+1} = \widehat{R}_{k+1} + P_k \beta_k$ $\{\mathrm{rQ}\text{: } P_{k+1} = \widehat{Q}_{k+1} + P_k \beta_k\}$
  $S_{k+1} = Z_{k+1} + S_k \beta_k$
end for
$X = X_k$; $Y = Y_k$

6. Numerical experiments

In this section we show numerical experiments to evaluate the performance of our method by using problems from real applications.

We test our method for the complex symmetric linear systems which need to be solved in the algorithm of a contour integral eigensolver for solving symmetric definite generalized eigenvalue problem $K\boldsymbol{t} = \mu M\boldsymbol{t}$. Here K is real symmetric indefinite, M is real s.p.d, and $(\mu, \boldsymbol{t})$ is an eigenpair. In the algorithm of a contour integral eigensolver, one needs to solve linear systems with a complex symmetric coefficient matrix $(z_j M - K)$ and with multiple right-hand sides, where z_j are complex scalars that are quadrature points of numerical contour integration. Since K is indefinite and M is positive definite, the linear system is of the type (1.1).

Table 1 shows the test matrices used for the experiments. The symbol $\mathrm{nnz}(\cdot)$ indicates the number of nonzero elements of the matrix. `BCSST25` is derived from a matrix pair (BCSSTK25 and BCSSTM25) obtained from the university of Florida sparse matrix collection [9]. For this problem, A_{I} is diagonal. `Ge87H76` is also obtained from [9]. For this matrix A_{I} is the identity matrix. `VCNT22500` is derived from a matrix pair with the same name which was obtained from ELSES matrix library [15]. `CQSZ20` is derived from a matrix pair obtained by the linear scaling

density functional theory code CONQUEST [8]. The symbol z indicates the complex scalar value used for computing $A_{\mathrm{R}} = \mathrm{real}(zM - K)$ and $A_{\mathrm{I}} = \mathrm{imag}(zM - K)$.

Name	Size	$\mathrm{nnz}(A_{\mathrm{R}})$	$\mathrm{nnz}(A_{\mathrm{I}})$	z
BCSST25	15 439	252 241	15 439	100+i
Ge87H76	112 985	7 892 195	112 985	0+440.01i
VCNT22500	22 500	8 737 290	8 737 290	−0.55+0.01i
CQSZ20	94 948	13 247 276	9 730 976	0.015+(4.9e−5)i

Table 1. Properties of the matrices used in the numerical examples and the complex scalar z used to make A_{R} and A_{I}.

We performed the experiments on a workstation which is equipped with two processors Intel Xeon E5-2667v3 (2.67GHz, 8 cores/16 thread, Haswells) with 512 GB main memory (DDR4 ECC REG 32GB×16). To implement the methods, we used the C++ language, C++ linear algebra library Eigen version 3.3.1 [13], and Intel Math Kernel Library (MKL). The real/complex symmetric $\mathrm{LDL}^{\mathrm{T}}$ factorization and their triangular solves were performed using PARDISO of MKL. In our method, we solved linear systems with $(A_{\mathrm{R}} + \gamma A_{\mathrm{I}})$ using the PARDISO direct solver. ILU0 preconditioner was implemented using Eigen's sparse matrix interface.

The real part B_{R} of the right-hand sides of linear systems were generated with random numbers. The imaginary part B_{I} was set as zero. Stopping criterion for iterative methods is $\max_i \|R_k(:,i)\|_2 / \|R_0(:,i)\|_2 < 1e-15$, where $R_k(:,i)$ is the ith column vector of R_k. The initial guess is set to zero for all examples. For our method, we let $\gamma = 0$ since A_{R} are nonsingular for all matrices.

6.1. Experiment I. In this example, we see how the number of iterations and the computation time are affected by the number of right-hand sides simultaneously solved by our real-valued block CG method (with residual orthogonalization). Tables 2, 3, 4, and 5 show the result for BCSST25, Ge87H76, VCNT22500, and CQSZ20, respectively. In the tables, # iter indicates the number of iteration and t_{fact}, t_{tsol}, t_{R}, t_{I}, and t_{qr} indicate computation time (in seconds) for the real $\mathrm{LDL}^{\mathrm{T}}$ factorization, the triangular solve using the factorization, the multiplication of A_{R}, the multiplication of A_{I}, and the QR factorization, respectively. The quantity t_{total} is the total computation time and `true_res` indicates the true residual norm.

The tables indicate that for all test matrices, as the number of right-hand sides s is increased, the number of iterations decreases. This indicates that the increase of the number of smallest eigenvalues of G_γ deflated may contribute to the reduction of the number of iterations needed to converge.

For all cases, the true relative residual norm is larger than 1e−15 although the stopping criterion is satisfied. This is a common situation in (block) Krylov subspace

methods and is caused by the rounding error. For all cases, we observe that as s increases, the true residual norm increases. Therefore we need to take care of the true residual norm.

s	1	2	4	8	16	32	64
# iter	23	18	14	12	10	8	7
t_{total}	1.0	0.8	1.0	1.4	2.1	3.3	5.8
t_{fact}	0.3	0.2	0.2	0.2	0.2	0.2	0.2
t_{tsol}	0.6	0.7	0.8	1.1	1.7	2.7	4.9
t_{R}	0.0	0.0	0.0	0.0	0.1	0.1	0.1
t_{I}	0.0	0.0	0.0	0.0	0.0	0.0	0.1
t_{qr}	0.0	0.0	0.0	0.0	0.1	0.2	0.3
`true_res`	1.4e−12	1.9e−12	1.9e−12	3.3e−12	3.4e−12	7.1e−12	9.0e−12

Table 2. Result of Experiment I for `BCSST25`.

s	1	2	4	8	16	32	64
# iter	37	28	22	16	13	10	8
t_{total}	443.4	467.6	524.9	560.2	639.8	789.5	1038.1
t_{fact}	276.4	282.6	276.3	282.3	276.5	282.1	276.4
t_{tsol}	166.5	184.2	247.3	275.9	359.6	499.8	750.0
t_{R}	0.3	0.5	0.8	1.1	1.8	2.8	4.4
t_{I}	0.0	0.1	0.1	0.1	0.2	0.3	0.5
t_{qr}	0.0	0.1	0.2	0.3	0.7	2.6	3.4
`true_res`	1.2e−14	2.0e−14	3.2e−13	1.1e−12	1.7e−12	3.1e−12	4.3e−12

Table 3. Result of Experiment I for `Ge87H76`.

s	1	2	4	8	16	32	64
# iter	116	76	48	32	22	17	14
t_{total}	22.7	24.3	26.1	28.8	34.3	50.0	78.9
t_{fact}	1.9	2.0	1.9	2.0	2.0	2.0	1.9
t_{tsol}	17.6	18.2	19.0	19.7	22.5	32.3	51.1
t_{R}	1.0	1.3	1.7	2.2	3.0	4.7	7.7
t_{I}	2.0	2.6	3.2	4.4	5.9	9.2	14.7
t_{qr}	0.0	0.0	0.1	0.1	0.2	0.6	0.8
`true_res`	7.0e−15	1.5e−14	1.9e−14	1.6e−13	9.2e−13	2.3e−12	4.8e−12

Table 4. Result of Experiment I for `VCNT22500`.

6.2. Experiment II. In this example we compare our method with complex symmetric LDL^{T} factorization, and ILU0-preconditiond and nonpreconditioned block

s	1	2	4	8	16	32	64
# iter	56	38	26	19	14	11	9
t_{total}	101.7	90.6	104.1	108.1	120.1	162.3	232.8
t_{fact}	24.3	23.9	23.9	24.3	23.9	24.3	23.9
t_{tsol}	75.3	63.7	76.2	77.7	86.8	121.3	182.0
t_{R}	0.8	1.0	1.4	2.1	3.1	4.9	7.8
t_{I}	1.2	1.5	2.1	3.0	4.4	6.9	11.1
t_{qr}	0.1	0.1	0.1	0.3	0.6	2.3	3.1
`true_res`	1.2e−13	1.5e−12	2.5e−12	4.8e−12	5.0e−11	6.4e−11	1.0e−10

Table 5. Result of Experiment I for `CQSZ20`.

COCGrQ method in terms of computation time. Tables 6, 7, 8, and 9 show the result for BCSST25, Ge87H76, VCNT22500, and CQSZ20, respectively.

In the tables, `Proposed`, `C-LDLT`, `BlockCOCG(I)`, and `BlockCOCG(ILU0)` indicate our method with residual orthogonalization, a direct solver using complex symmetric $\mathrm{LDL}^{\mathrm{T}}$ factorization, nonpreconditioned block COCGrQ, and ILU-preconditioned block COCGrQ, respectively.

For the block COCGrQ method, the computation time for mat-vec with complex symmetric matrix is shown in the row of t_{R}.

For iterative methods the maximum number of iterations is set to $\min(\lceil n/s \rceil, 3000)$. In this experiment, the number of right-hand sides s is fixed to 16.

For `BCSST25`, the proposed method is approximately three times slower than `C-LDLT` because the triangular solve is relatively expensive as compared to the factorization. For `BlockCOCG(I)` and `BlockCOCG(ILU0)`, the residual norm did not converge for this matrix.

For `Ge87H76`, the proposed method is faster than `C-LDLT` because, in contrast to the previous example, the triangular solve is relatively inexpensive as compared to the factorization. Both `BlockCOCG(I)` and `BlockCOCG(ILU0)` needed a lot of iterations, and are far slower than `Proposed` and `C-LDLT`. Though the ILU0 preconditioning successfully reduces the number of iterations, the total computational time is larger than in the nonpreconditioned case due to the computational cost of the (ILU0) triangular solve. Also we note that for the block COCG method, the relative true residual norm is quite larger (around 1e−11) than the stopping criterion 1e−15.

For `VCNT22500`, the proposed method is approximately eight times slower than `C-LDLT` because the triangular solve is relatively expensive as is also seen in the case of `BCSST25`. For `BlockCOCG(I)`, the norm of the residual computed by recurrences was smaller than tolerance. However, it is far slower than `Proposed` and `C-LDLT`. `BlockCOCG(ILU0)` failed because the (ILU0) triangular solve broke down.

For CQSZ20, the proposed method is slower than C-LDLT. While the computational cost of the triangular solve is considerably smaller than the $\mathrm{LDL}^{\mathrm{T}}$ factorization, the number of iterations is not small enough to make our method faster than C-LDLT. In BlockCOCG(I) and BlockCOCG(ILU0), the residual norm did not converge for this matrix.

In Figure 1, we show the histories of the residual norms of the proposed method (when $s = 16$) for different matrices. The figure indicates that for all cases, the residual norms decrease smoothly. In such cases, if one can accept the solution with low accuracy, the proposed method can be faster than C-LDLT. The case of CQSZ20 is an example for such situation.

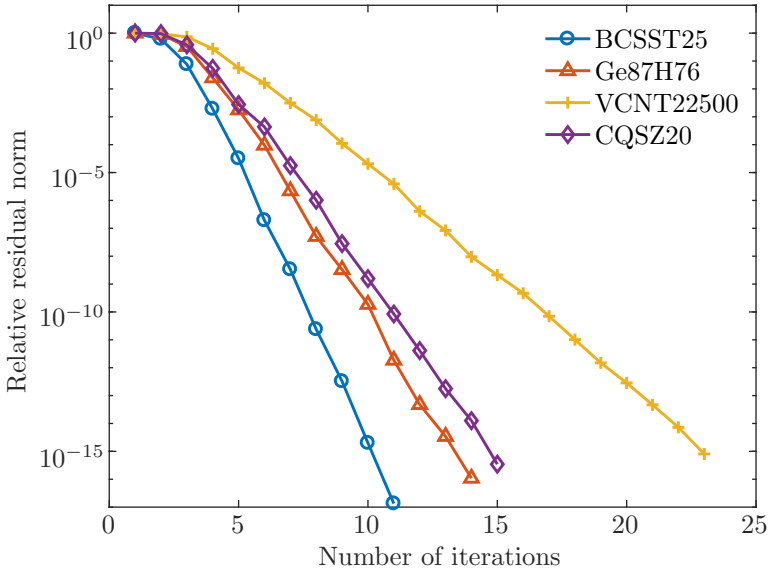

Figure 1. Residual norm histories of the proposed method for different matrices.

Method	Proposed	C-LDLT	BlockCOCG(I)	BlockCOCG(ILU0)
# iter	10	–	965(max)	965(max)
t_{total}	2.1	0.6	44.8	68.3
t_{fact}	0.2	0.3	–	0.0
t_{tsol}	1.7	0.2	–	23.9
t_{R}	0.1	–	19.8	20.0
t_{I}	0.0	–	–	–
t_{qr}	0.1	–	16.0	15.8
true_res	3.4e−12	7.9e−13	2.4e+03	8.0e−01

Table 6. Result of Experiment II for BCSST25.

Method	Proposed	C-LDLT	BlockCOCG(I)	BlockCOCG(ILU0)
# iter	13	–	2106	1469
t_{total}	639.8	819.1	1759.7	2383.1
t_{fact}	276.5	788.3	–	3.2
t_{tsol}	359.6	30.8	–	1177.5
t_{R}	1.8	–	1212.3	843.6
t_{I}	0.2	–	–	–
t_{qr}	0.7	–	343.0	238.8
true_res	1.7e−12	5.5e−15	4.1e−11	5.6e−11

Table 7. Result of Experiment II for Ge87H76.

Method	Proposed	C-LDLT	BlockCOCG(I)	BlockCOCG(ILU0)
# iter	22	–	1407	Fail
t_{total}	34.3	4.4	860.4	–
t_{fact}	2.0	3.3	–	–
t_{tsol}	22.5	1.1	–	–
t_{R}	3.0	–	807.4	–
t_{I}	5.9	–	–	–
t_{qr}	0.2	–	34.9	–
true_res	9.2e−13	1.4e−15	2.3e−11	–

Table 8. Result of Experiment II for VCNT22500.

Method	Proposed	C-LDLT	BlockCOCG(I)	BlockCOCG(ILU0)
# iter	14	–	3000(max)	3000(max)
t_{total}	120.1	75.4	3404.5	7490.4
t_{fact}	23.9	67.9	–	4.2
t_{tsol}	86.8	7.5	–	4096.4
t_{R}	3.1	–	2774.0	2752.4
t_{I}	4.4	–	–	–
t_{qr}	0.6	–	383.6	391.9
true_res	5.0e−11	2.1e−14	4.5e+01	1.8e+04

Table 9. Result of Experiment II for CQSZ20.

Finally, we note that the memory requirements of our method with real LDL^{T} factorization is a half of the direct solver using the complex symmetric LDL^{T} factorization. For large-scale problems, the memory requirement of the algorithm is an important characteristic as well as computational cost. Our method is useful when the shortage of memory is an issue.

7. Conclusions

In this study we proposed a new block Krylov type method for solving a specific type of complex symmetric system with multiple right-hand sides. We assume that the coefficient matrix has indefinite real part and positive definite imaginary part. We also investigate the convergence property of our method. It is shown that the $s-1$ smallest eigenvalues of the matrix are removed from the spectral condition number of the coefficient matrix. This indicates the possibility of having faster convergence rate than in the nonblocked version.

We also presented an efficient algorithmic implementation of our method. Namely, we show that expensive triangular solves and matrix-vector multiplications of the imaginary part are reduced by this efficient implementation.

In the numerical experiments, we observe that, for four matrices from different kinds of applications, the number of iterations decreases as the number of right-hand sides increases. We experimentally compare our method with the direct solver using complex symmetric $\mathrm{LDL}^{\mathrm{T}}$ factorization, the nonpreconditioned ILU0-preconditioned block COCG method. As a result, our method always turns out to be faster than the (ILU0-preconditioned) block COCG method. In some cases, our method is also faster than the direct solver. However, when the triangular solve is relatively expensive or number of iteration is not small enough, our method failed to outperform the direct solver.

The proposed method is a conjugate gradient type method that minimizes a norm of the error. Another optimal real-valued method which minimizes a norm of the residual can be derived analogously. Therefore, in our future work, we will investigate such minimal residual type method.

Appendix

In Algorithm 5 we show the pseudo-code of the preconditioned block COCGrQ method for solving the complex symmetric linear system with multiple right-hand sides

$$AX = B,$$

where $A = A^{\mathrm{T}} \neq A^{\mathrm{H}} \in \mathbb{C}^{n\times n}$ and $X, B \in \mathbb{C}^{n\times s}$. $M = M^{\mathrm{T}} \neq M^{\mathrm{H}} \in \mathbb{C}^{n\times n}$ in the pseudo-code is the preconditioning matrix.

Algorithm 5. Preconditioned block COCGrQ method. M is the preconditioning matrix and $\mathrm{qr}(\cdot)$ indicates the thin QR decomposition.

Input: $A = A^{\mathrm{T}} \in \mathbb{C}^{n\times n}$; $B, X_0 \in \mathbb{C}^{n\times s}$; $M \in \mathbb{C}^{n\times n}$

Output: $X \in \mathbb{C}^{n\times s}$

$R_0 = B - AX_0$; $Q_0\Delta_0 = R_0$
$P_0 = Q_0$; $Z_0 = MP_0$
for $k = 0, 1, \ldots$ until solution converges **do**
  $\alpha_k = (P_k^{\mathrm{T}} AP_k)^{-1}(Q_k^{\mathrm{T}} Z_k)$
  $X_{k+1} = X_k + P_k\alpha_k\Delta_k$
  $Q_{k+1}\varrho_{k+1} = \mathrm{qr}(Q_k - P_k\alpha_k)$
  $\Delta_{k+1} = \varrho_{k+1}\Delta_k$
  $Z_{k+1} = MQ_{k+1}$
  $\beta_k = (Q_k^{\mathrm{T}} Z_k)^{-1}\varrho_{k+1}^{\mathrm{T}}(Q_{k+1}^{\mathrm{T}} Z_{k+1})$
  $P_{k+1} = Z_{k+1} + P_k\beta_k$
end for
$X = X_k$

A c k n o w l e d g e m e n t. The authors would like to thank Dr. Tsuyoshi Miyazaki and Dr. Ayako Nakata for giving us a matrix from the linear scaling density functional theory code CONQUEST.

References

[1] *O. Axelsson, A. Kucherov*: Real valued iterative methods for solving complex symmetric linear systems. Numer. Linear Algebra Appl. *7* (2000), 197–218. zbl MR doi
[2] *O. Axelsson, M. Neytcheva, B. Ahmad*: A comparison of iterative methods to solve complex valued linear algebraic systems. Numer. Algorithms *66* (2014), 811–841. zbl MR doi
[3] *Z.-Z. Bai, M. Benzi, F. Chen*: Modified HSS iteration methods for a class of complex symmetric linear systems. Computing *87* (2010), 93–111. zbl MR doi
[4] *Z.-Z. Bai, M. Benzi, F. Chen*: On preconditioned MHSS iteration methods for complex symmetric linear systems. Numer. Algorithms *56* (2011), 297–317. zbl MR doi
[5] *Z.-Z. Bai, M. Benzi, F. Chen, Z.-Q. Wang*: Preconditioned MHSS iteration methods for a class of block two-by-two linear systems with applications to distributed control problems. IMA J. Numer. Anal. *33* (2013), 343–369. zbl MR doi
[6] *Z.-Z. Bai, G. H. Golub, M. K. Ng*: Hermitian and Skew-Hermitian splitting methods for non-Hermitian positive definite linear systems. SIAM J. Matrix Anal. Appl. *24* (2003), 603–626. zbl MR doi
[7] *Z.-Z. Bai, G. H. Golub, M. K. Ng*: On successive-overrelaxation acceleration of the Hermitian and skew-Hermitian splitting iterations. Numer. Linear Algebra Appl. *14* (2007), 319–335; erratum ibid. *19* (2012), 891. zbl MR doi
[8] CONQUEST: Linear Scaling DFT. `http://www.order-n.org/`.
[9] *T. A. Davis, Y. Hu*: The university of Florida sparse matrix collection. ACM Trans. Math. Softw. *38* (2011), Paper No. 1, 25 pages. zbl MR doi
[10] *D. Day, M. A. Heroux*: Solving complex-valued linear systems via equivalent real formulations. SIAM J. Sci. Comput. *23* (2001), 480–498. zbl MR doi
[11] *L. Du, Y. Futamura, T. Sakurai*: Block conjugate gradient type methods for the approximation of bilinear form $C^H A^{-1} B$. Comput. Math. Appl. *66* (2014), 2446–2455. MR doi

[12] *A. A. Dubrulle*: Retooling the method of block conjugate gradients. ETNA, Electron. Trans. Numer. Anal. *12* (2001), 216–233. zbl MR
[13] Eigen. `http://eigen.tuxfamily.org/`.
[14] *S. C. Eisenstat, H. C. Elman, M. H. Schultz*: Variational iterative methods for nonsymmetric systems of linear equations. SIAM J. Numer. Anal. *20* (1983), 345–357. zbl MR doi
[15] ELSES matrix library. `http://www.elses.jp/matrix/`.
[16] *R. W. Freund*: Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices. SIAM J. Sci. Stat. Comput. *13* (1992), 425–448. zbl MR doi
[17] *Y. Futamura, H. Tadano, T. Sakurai*: Parallel stochastic estimation method of eigenvalue distribution. JSIAM Lett. *2* (2010), 127–130. zbl MR doi
[18] *T. Ikegami, T. Sakurai*: Contour integral eigensolver for non-Hermitian systems: a Rayleigh-Ritz-type approach. Taiwanese J. Math. *14* (2010), 825–837. zbl MR
[19] *T. Ikegami, T. Sakurai, U. Nagashima*: A filter diagonalization for generalized eigenvalue problems based on the Sakurai-Sugiura projection method. J. Comput. Appl. Math. *233* (2010), 1927–1936. zbl MR doi
[20] *A. Imakura, L. Du, T. Sakurai*: A block Arnoldi-type contour integral spectral projection method for solving generalized eigenvalue problems. Appl. Math. Lett. *32* (2014), 22–27. zbl MR doi
[21] *A. A. Nikishin, A. Y. Yeremin*: 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* (1995), 1135–1153. zbl MR doi
[22] *D. P. O'Leary*: The block conjugate gradient algorithm and related methods. Linear Algebra Appl. *29* (1980), 293–322. zbl MR doi
[23] *C. C. Paige, M. A. Saunders*: Solutions of sparse indefinite systems of linear equations. SIAM J. Numer. Anal. *12* (1975), 617–629. zbl MR doi
[24] *E. Polizzi*: Density-matrix-based algorithm for solving eigenvalue problems. Phys. Rev. B *79* (2009), 115112. doi
[25] *Y. Saad, M. H. Schultz*: GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. *7* (1986), 856–869. zbl MR doi
[26] *T. Sakurai, H. Sugiura*: A projection method for generalized eigenvalue problems using numerical integration. J. Comput. Appl. Math. *159* (2003), 119–128. zbl MR doi
[27] *T. Sakurai, H. Tadano*: CIRR: a Rayleigh-Ritz type method with contour integral for generalized eigenvalue problems. Hokkaido Math. J. *36* (2007), 745–757. zbl MR doi
[28] *T. Sogabe, S.-L. Zhang*: A COCR method for solving complex symmetric linear systems. J. Comput. Appl. Math. *199* (2007), 297–303. zbl MR doi
[29] *H. Tadano, T. Sakurai*: A block Krylov subspace method for the contour integral method and its application to molecular orbital computations. IPSJ Trans. Adv. Comput. Syst. *2* (2009), 10–18. (In Japanese.)
[30] *H. A. van der Vorst*: Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the Solution of nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. *13* (1992), 631–644. zbl MR doi
[31] *H. A. van der Vorst, J. B. M. Melissen*: A Petrov-Galerkin type method for solving $Axk = b$, where A is symmetric complex. IEEE Transactions on Magnetics *26* (1990), 706–708. doi

Authors' address: *Yasunori Futamura*, *Takahiro Yano*, *Akira Imakura*, *Tetsuya Sakurai*, University of Tsukuba, 1-1-1 Tennodai, Tsukuba, Ibaraki 305-8577, Japan, e-mail: `futamura@cs.tsukuba.ac.jp`.