East Asian Journal on Applied Mathematics
doi: 10.4208/eajam.260410.240510a

Vol. **1**, No. 2, pp. 97-107
May 2011

An Extension of the COCR Method to Solving Shifted Linear Systems with Complex Symmetric Matrices

Tomohiro Sogabe[*,1] and Shao-Liang Zhang[2]

[1] *Graduate School of Information Science & Technology, Aichi Prefectural University, 1522-3 Ibaragabasama, Kumabari, Nagakute-cho, Aichi-gun, Aichi, 480-1198, Japan.*
[2] *Department of Computational Science and Engineering, Nagoya University, Furo-cho, Chikusa-ku, Nagoya 464-8603, Japan.*

Received 26 April 2010; Accepted (in revised version) 24 May 2010

Available online 7 April 2011

Abstract. The Conjugate Orthogonal Conjugate Residual (COCR) method [T. Sogabe and S.-L. Zhang, JCAM, 199 (2007), pp. 297-303.] has recently been proposed for solving complex symmetric linear systems. In the present paper, we develop a variant of the COCR method that allows the efficient solution of complex symmetric shifted linear systems. Some numerical examples arising from large-scale electronic structure calculations are presented to illustrate the performance of the variant.

Key words: Shifted linear systems, complex symmetric matrices, COCR, Krylov subspace methods, electronic structure calculation.

1. Introduction

We consider the solution of shifted linear systems of the form:

$$A(\sigma_k)\boldsymbol{x}^{(k)} = \boldsymbol{b} \quad \text{for all } k \in S := \{1, 2, \cdots, m\}, \tag{1.1}$$

where $A(\sigma_k) := A + \sigma_k I \in \mathbb{C}^{N\times N}$ is not Hermitian but a complex symmetric sparse matrix, i.e. $A(\sigma_k) = A(\sigma_k)^T \neq \bar{A}(\sigma_k)^T$, with a scalar shift $\sigma_k \in \mathbb{C}$, and $\boldsymbol{x}^{(k)}, \boldsymbol{b}$ are complex vectors of length N. The shifted linear systems arise in large-scale electronic structure calculations [15] and there is a strong need for a fast solution method.

For solving (1.1), Krylov subspace methods are very attractive, since the coefficient matrices are sparse. Moreover, in this case we can use the well-known *shift-invariance property of Krylov subspaces*:

$$K_n(A(\sigma_i), \boldsymbol{b}) = K_n(A(\sigma_j), \boldsymbol{b}), \quad 1 \leq i, j \leq m, \tag{1.2}$$

*Corresponding author. *Email addresses:* sogabe@ist.aichi-pu.ac.jp (T. Sogabe), zhang@na.cse.nagoya-u.ac.jp (S.-L. Zhang)

where $K_n(A, \boldsymbol{b}) := \mathrm{span}\{\boldsymbol{b}, A\boldsymbol{b}, \cdots, A^{n-1}\boldsymbol{b}\}$. The latter property means that only one Krylov subspace for solving m linear systems must be generated. So, when one finds approximate solutions over Krylov subspaces, one can save the costs of generating $m-1$ Krylov subspaces. This approach was shown to be highly effective: cf. [1, 6] for Hermitian positive definite case; [4, 5] for non-Hermitian case; and [3, 13, 15] for complex symmetric case (1.1).

For solving (1.1), the shifted COCG method [15] and the shifted QMR-type methods [3, 13] are powerful solvers. They are based on the shift invariance property (1.2) and basic solvers for complex symmetric linear systems, i.e. the COCG method [16] and the QMR method [3]. Although these methods are powerful, they fall into one group that uses the complex symmetric Lanczos process - e.g. see, [2, Algorithm 2.1]. This means that if the shifted QMR-type methods fail, the shifted COCG method may also fail, and vice versa. So, it is still worth finding an algorithm that is based on a different principle, in order to reduce the risk of facing such a situation in practice. The purpose of this paper is to find an algorithm using a different principle from the complex symmetric Lanczos process, as efficient as the shifted COCG method and the shifted QMR-type methods. In this paper, we consider the COCR method [12] that is based on A-conjugate orthogonalization process for solving complex symmetric linear systems, and then we develop the COCR method in order to solve the shifted linear systems (1.1).

The rest of this paper is organized as follows: in the next section, we review the algorithm of COCR and its main property. In Section 3, we develop a numerical method named Shifted COCR, in order to solve complex symmetric shifted linear systems, and describe the algorithm in a complete form. In Section 4, we report the results of some numerical examples. Finally, we make some concluding remarks in Section 5.

2. The COCR Method

In this Section, we briefly review the COCR method [12]. The COCR method is a Krylov subspace method for solving complex symmetric linear systems, and it is derived from a A-conjugate orthogonalization process (that is a special case of A-biorthogonalization process [11]) of Krylov subspace. Here we describe below the algorithm of COCR when applied to the system $A\boldsymbol{x} = \boldsymbol{b}$ with a complex symmetric matrix A.

Observing Algorithm 1, the nth residual vector can be written as

$$\boldsymbol{r}_n := \boldsymbol{b} - A\boldsymbol{x}_n = R_n(A)\boldsymbol{r}_0, \tag{2.1}$$

where the polynomial $R_n(\lambda)$ with a scalar λ is written by the following coupled two-term recurrence relation:

$$\begin{aligned} R_0(\lambda) &= 1, \quad P_0(\lambda) = 1, \\ R_n(\lambda) &= R_{n-1}(\lambda) - \alpha_{n-1}\lambda P_{n-1}(\lambda), \\ P_n(\lambda) &= R_n(\lambda) + \beta_{n-1}P_{n-1}(\lambda), \quad n = 1, 2, \cdots, \end{aligned}$$

Algorithm 1. COCR

Set $\boldsymbol{x}_0$ is an initial guess,
Set $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0,\ \boldsymbol{p}_{-1} = \boldsymbol{0},\ \beta_{-1} = 0,$
for $n = 0, 1, \cdots,$ until $\|\boldsymbol{r}_n\|_2 \leq \epsilon_1 \|\boldsymbol{b}\|_2$ **do:**
$\boldsymbol{p}_n = \boldsymbol{r}_n + \beta_{n-1}\boldsymbol{p}_{n-1},$
$(A\boldsymbol{p}_n = A\boldsymbol{r}_n + \beta_{n-1}A\boldsymbol{p}_{n-1},)$
$\alpha_n = \dfrac{(\bar{\boldsymbol{r}}_n, A\boldsymbol{r}_n)}{(\bar{A}\bar{\boldsymbol{p}}_n, A\boldsymbol{p}_n)},$
$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \alpha_n \boldsymbol{p}_n,$
$\boldsymbol{r}_{n+1} = \boldsymbol{r}_n - \alpha_n A\boldsymbol{p}_n,$
$\beta_n = \dfrac{(\bar{\boldsymbol{r}}_{n+1}, A\boldsymbol{r}_{n+1})}{(\bar{\boldsymbol{r}}_n, A\boldsymbol{r}_n)}.$
end

Eliminating $P_n(\lambda)$ in the above recurrences relation, we readily obtain the following three-term recurrences relation:

$$R_0(\lambda) = 1, \tag{2.2}$$

$$R_1(\lambda) = (1 - \alpha_0\lambda)R_0(\lambda), \tag{2.3}$$

$$R_n(\lambda) = \left(1 + \frac{\beta_{n-2}}{\alpha_{n-2}}\alpha_{n-1} - \alpha_{n-1}\lambda\right)R_{n-1}(\lambda) - \frac{\beta_{n-2}}{\alpha_{n-2}}\alpha_{n-1}R_{n-2}(\lambda), \quad n = 2, 3, \cdots \tag{2.4}$$

The above recurrences (2.2)-(2.4) which play an important role in deriving a variant of COCR for solving complex symmetric shifted linear systems in the next section. It is shown in [12] that if breakdown does not occur, then the nth residual vector of COCR satisfies

$$\boldsymbol{r}_n \perp \bar{A}K_n(\bar{A}, \bar{\boldsymbol{r}}_0),$$

which leads to A-conjugate orthogonality $(\bar{\boldsymbol{r}}_i, A\boldsymbol{r}_j) = 0$ for $i \neq j$. The COCR method corresponds to the Conjugate Residual (CR) method [14, 9 p. 194], when the coefficient matrix is real symmetric.

3. A Shifted COCR Method

In this Section, we describe a way to solve shifted linear systems using the information sent from the COCR method, when applied to a linear system $A\boldsymbol{x} = \boldsymbol{b}$ (so called *seed system*). For simplicity, we consider here solving two linear systems, i.e. the seed system $A\boldsymbol{x} = \boldsymbol{b}$ and the shifted system $(A + \sigma_1 I)\boldsymbol{x}^{(1)} = \boldsymbol{b}$. We can see from the residual vector

(2.1) that $\boldsymbol{r}_n$ with the initial guess $\boldsymbol{x}_0 = \boldsymbol{0}$ belongs to the subspace $K_{n+1}(A, \boldsymbol{b})$. Moreover, we observe that

$$\mathrm{span}\{\boldsymbol{r}_0, \cdots, \boldsymbol{r}_n\} = K_{n+1}(A, \boldsymbol{b}) = K_{n+1}(A + \sigma_1 I, \boldsymbol{b}).$$

Hence, we consider reusing the information of residual vectors $\boldsymbol{r}_0, \boldsymbol{r}_1, \cdots, \boldsymbol{r}_n$ to solve $(A + \sigma_1 I)\boldsymbol{x}^{(1)} = \boldsymbol{b}$. To be specific, for solving the shifted system $(A + \sigma_1 I)\boldsymbol{x}^{(1)} = \boldsymbol{b}$, we take the following collinear residual approach:

$$\boldsymbol{r}_n = \pi_n^{(1)} \boldsymbol{r}_n^{(1)}, \quad \pi_n^{(1)} \in \mathbb{C}. \tag{3.1}$$

Here we note that this approach is successfully used in the algorithm of restarted shifted GMRES [5]. Now we give a computational formula for updating $\boldsymbol{r}_{n+1}^{(1)}$ by using the information of $\boldsymbol{r}_{n+1}$. From (2.1) and the recurrence relation (2.4) it follows that

$$\boldsymbol{r}_{n+1} = \left(1 + \frac{\beta_{n-1}}{\alpha_{n-1}}\alpha_n - \alpha_n A\right)\boldsymbol{r}_n - \frac{\beta_{n-1}}{\alpha_{n-1}}\alpha_n \boldsymbol{r}_{n-1}. \tag{3.2}$$

Similarly, we consider updating the residual vector $\boldsymbol{r}_{n+1}^{(1)}$ with the following recurrence relation:

$$\boldsymbol{r}_{n+1}^{(1)} = \left\{1 + \frac{\beta_{n-1}^{(1)}}{\alpha_{n-1}^{(1)}}\alpha_n^{(1)} - \alpha_n^{(1)}(A + \sigma_1 I)\right\}\boldsymbol{r}_n^{(1)} - \frac{\beta_{n-1}^{(1)}}{\alpha_{n-1}^{(1)}}\alpha_n^{(1)} \boldsymbol{r}_{n-1}^{(1)}. \tag{3.3}$$

Substituting (3.1) into (3.3) we have

$$\boldsymbol{r}_{n+1} = \left\{1 + \frac{\beta_{n-1}^{(1)}}{\alpha_{n-1}^{(1)}}\alpha_n^{(1)} - \alpha_n^{(1)}(A + \sigma_1 I)\right\}\frac{\pi_{n+1}^{(1)}}{\pi_n^{(1)}}\boldsymbol{r}_n - \frac{\beta_{n-1}^{(1)}\alpha_n^{(1)}\pi_{n+1}^{(1)}}{\alpha_{n-1}^{(1)}\pi_{n-1}^{(1)}}\boldsymbol{r}_{n-1}. \tag{3.4}$$

To obtain the computational formula for $\boldsymbol{r}_{n+1}^{(1)}$, three parameters $\alpha_n^{(1)}$, $\beta_{n-1}^{(1)}$, and $\pi_{n+1}^{(1)}$ are essentially required. Hence, here we give computational formulas for the three parameters. First, comparing the coefficients of $A\boldsymbol{r}_n$ in (3.2) and (3.4) yields

$$\alpha_n^{(1)} = \left(\pi_n^{(1)}/\pi_{n+1}^{(1)}\right)\alpha_n. \tag{3.5}$$

Second, comparing the coefficients of $\boldsymbol{r}_{n-1}$ leads to $\frac{\beta_{n-1}}{\alpha_{n-1}}\alpha_n = \frac{\beta_{n-1}^{(1)}\alpha_n^{(1)}\pi_{n+1}^{(1)}}{\alpha_{n-1}^{(1)}\pi_{n-1}^{(1)}}$. Substituting the result of (3.5) into the previous equation yields

$$\beta_{n-1}^{(1)} = \left(\pi_{n-1}^{(1)}/\pi_n^{(1)}\right)^2 \beta_{n-1}. \tag{3.6}$$

Finally, comparing the coefficients of $\boldsymbol{r}_n$ leads to $(1 + \frac{\beta_{n-1}^{(1)}}{\alpha_{n-1}^{(1)}}\alpha_n^{(1)} - \alpha_n^{(1)}\sigma_1)\frac{\pi_{n+1}^{(1)}}{\pi_n^{(1)}} = 1 + \frac{\beta_{n-1}}{\alpha_{n-1}}\alpha_n$. Substituting (3.5) and (3.6) into the previous equation, we obtain

$$\pi_{n+1}^{(1)} = \left(1 + \frac{\beta_{n-1}}{\alpha_{n-1}}\alpha_n + \alpha_n\sigma_1\right)\pi_n^{(1)} - \frac{\beta_{n-1}}{\alpha_{n-1}}\alpha_n \pi_{n-1}^{(1)} = R_{n+1}(-\sigma_1). \tag{3.7}$$

Table 1: Summary of operations per iteration step, where AXPY: $a\boldsymbol{x}+\boldsymbol{y}$, IP: number of inner products, MV: number of matrix-vector multiplications, m: number of linear systems to be solved.

Solver	IP	AXPY	MV
COCR	$2m$	$4m$	m
Shifted COCR	2	$2(m+1)$	1

From (3.1), (3.3), (3.5), (3.6), and (3.7) we can update the residual vector $\boldsymbol{r}_{n+1}^{(1)}$. Next, we derive a computational formula for updating the approximate solution $\boldsymbol{x}_{n+1}^{(1)}$ from the recurrence relation (3.3). It follows from (3.3) and using $\boldsymbol{p}_{n-1}^{(1)} := (A+\sigma_1 I)^{-1}(\boldsymbol{r}_{n-1}^{(1)} - \boldsymbol{r}_n^{(1)})/\alpha_{n-1}^{(1)}$ that we readily have

$$\boldsymbol{p}_n^{(1)} = \boldsymbol{r}_n^{(1)} + \beta_{n-1}^{(1)}\boldsymbol{p}_{n-1}^{(1)}, \tag{3.8}$$

$$\boldsymbol{r}_{n+1}^{(1)} = \boldsymbol{r}_n^{(1)} - \alpha_n^{(1)}(A+\sigma_1 I)\boldsymbol{p}_n^{(1)}. \tag{3.9}$$

Hence, substituting $\boldsymbol{r}_n^{(1)} = \boldsymbol{b} - (A+\sigma_1 I)\boldsymbol{x}_n^{(1)}$ into (3.9) we obtain

$$\boldsymbol{x}_{n+1}^{(1)} = \boldsymbol{x}_n^{(1)} + \alpha_n^{(1)}\boldsymbol{p}_n^{(1)}. \tag{3.10}$$

Notice that from (3.1) the search directions are updated by

$$\boldsymbol{p}_n^{(1)} = \frac{1}{\pi_n^{(1)}}\boldsymbol{r}_n + \beta_{n-1}^{(1)}\boldsymbol{p}_{n-1}^{(1)}. \tag{3.11}$$

Then, approximate solutions are updated without using the recurrence relation (3.9).

From Algorithm 1, (3.1), and (3.5)-(3.7), (3.10), (3.11) we can obtain the approximate solutions for the system $(A+\sigma_1 I)\boldsymbol{x}^{(1)} = \boldsymbol{b}$ without matrix-vector multiplications. Additionally, for the evaluation of $\|\boldsymbol{r}_n^{(1)}\|_2$ we do not need to compute the inner product of $\boldsymbol{r}_n^{(1)}$ since we can implicitly obtain the residual 2-norm from the relation (3.1), i.e. $\|\boldsymbol{r}_n^{(1)}\|_2 = \|\boldsymbol{r}_n\|_2/|\pi_n^{(1)}|$.

The above derivation is based on the assumption that the seed and shifted systems are $A\boldsymbol{x} = \boldsymbol{b}$ and $(A+\sigma_1 I)\boldsymbol{x}^{(1)} = \boldsymbol{b}$ respectively. Similarly, it can be readily generalized to solve $m-1$ shifted linear systems $(A+\sigma_k I)\boldsymbol{x}^{(k)} = \boldsymbol{b}$ using the seed system $(A+\sigma_s I)\boldsymbol{x} = \boldsymbol{b}$. The resulting algorithm is given in Algorithm 2.

Here we note that Algorithm 2 chooses a seed system from given systems, but the choice of the seed systems is not restricted to a set of systems to be solved, i.e. we can choose another system - e.g., $A\boldsymbol{x} = \boldsymbol{b}$ as a seed system to solve $A(\sigma_k)\boldsymbol{x}^{(k)} = \boldsymbol{b}$ for all $k \in S$. In any case, the optimal choice of a seed system is an open problem.

The rest of this Section shows the summary of computational costs per iteration step for the COCR method and the shifted COCR method in Table 1, where the number of AXPY for the shifted COCR is $2(m+1)$ since it is a summation of 4 for a seed system and $2(m-1)$ for shifted systems.

Algorithm 2. Shifted COCR

Choose a seed system $s \in S = \{1, 2, \cdots, m\}$,
Set $\boldsymbol{r}_0^{(s)} = \boldsymbol{b},\ \beta_{-1}^{(s)} = 0$,
Set $\boldsymbol{x}_0^{(k)} = \boldsymbol{p}_{-1}^{(k)} = \boldsymbol{0},\ k = 1, 2, \cdots, m$,
Set $\pi_0^{(k)} = \pi_{-1}^{(k)} = 1,\ k = 1, 2, \cdots, m$,
for $n = 0, 1, \cdots,$ until $\|\boldsymbol{r}_n^{(s)}\|_2 \leq \epsilon_1 \|\boldsymbol{b}\|_2$ **do:**
$\boldsymbol{p}_n^{(s)} = \boldsymbol{r}_n^{(s)} + \beta_{n-1}^{(s)} \boldsymbol{p}_{n-1}^{(s)}$,
$(A(\sigma_s)\boldsymbol{p}_n^{(s)} = A(\sigma_s)\boldsymbol{r}_n^{(s)} + \beta_{n-1}^{(s)} A(\sigma_s)\boldsymbol{p}_{n-1}^{(s)},)$
$\alpha_n^{(s)} = \dfrac{(\bar{\boldsymbol{r}}_n^{(s)}, A(\sigma_s)\boldsymbol{r}_n^{(s)})}{(\bar{A}(\sigma_s)\bar{\boldsymbol{p}}_n^{(s)}, A(\sigma_s)\boldsymbol{p}_n^{(s)})}$,
$\boldsymbol{x}_{n+1}^{(s)} = \boldsymbol{x}_n^{(s)} + \alpha_n^{(s)} \boldsymbol{p}_n^{(s)}$,
{Begin shifted system}
for $k(\neq s) = 1, 2, \cdots, m$ **do:**
if $\|\boldsymbol{r}_n^{(k)}\|_2 (= \|\boldsymbol{r}_n\|_2 / |\pi_n^{(k)}|) > \epsilon_2 \|\boldsymbol{b}\|_2$ **then**
$\pi_{n+1}^{(k)} = R_{n+1}(\sigma_s - \sigma_k), \Leftarrow (3.7)$
$\beta_{n-1}^{(k)} = \left(\dfrac{\pi_{n-1}^{(k)}}{\pi_n^{(k)}}\right)^2 \beta_{n-1}$,
$\alpha_n^{(k)} = \dfrac{\pi_n^{(k)}}{\pi_{n+1}^{(k)}} \alpha_n$,
$\boldsymbol{p}_n^{(k)} = \dfrac{1}{\pi_n^{(k)}} \boldsymbol{r}_n + \beta_{n-1}^{(k)} \boldsymbol{p}_{n-1}^{(k)}$,
$\boldsymbol{x}_{n+1}^{(k)} = \boldsymbol{x}_n^{(k)} + \alpha_n^{(k)} \boldsymbol{p}_n^{(k)}$,
end if
end
{End shifted system}
$\boldsymbol{r}_{n+1}^{(s)} = \boldsymbol{r}_n^{(s)} - \alpha_n^{(s)} A(\sigma_s)\boldsymbol{p}_n^{(s)}$,
$\beta_n^{(s)} = \dfrac{(\bar{\boldsymbol{r}}_{n+1}^{(s)}, A(\sigma_s)\boldsymbol{r}_{n+1}^{(s)})}{(\bar{\boldsymbol{r}}_n^{(s)}, A(\sigma_s)\boldsymbol{r}_n^{(s)})}$.
end

From Table 1, we can say that the computational cost per iteration step for the shifted COCR method is much less than that for the COCR method when matrix-vector multipli-

cations are the most time-consuming part.

4. Numerical Examples

In this Section, we report the results of two numerical examples with the COCR method, the preconditioned COCR method, and the shifted COCR method. Here we note that the COCR method was chosen as representative, since it is known from [12] that the COCR method is competitive with other successful Krylov subspace methods such as the COCG method [16] and the QMR method [2]. In the numerical examples, we evaluate the methods in terms of the number of total matrix-vector multiplications and the computation time. We will also give the computation time of the shifted COCG method [15] for each example. All tests were performed on a workstation with a 2.6GHz AMD Opteron processor 252 using double precision arithmetic. Codes were written in Fortran 77 and compiled with g77 -*O*3. In all cases, the iteration was started with $\boldsymbol{x}_0 = \boldsymbol{0}$ for the COCR method and $\boldsymbol{x}_0^{(k)} = \boldsymbol{0}$, $k = 1, 2, \cdots, m$, for the shifted COCR method, and the stopping criterion was $\epsilon_1 = \epsilon_2 = 10^{-12}$. The preconditioners used in the experiments were the IC(0) preconditioner [7] and a complex symmetric version of the D-ILU preconditioner [8] (for simplicity we call it D-ILU hereafter). For the complex symmetric structure of A, IC(0) and D-ILU generate matrices of the form LDL^T. If the diagonal matrix D and the lower triangular matrix L are nonsingular, then the preconditioned matrix $D^{-1/2}L^{-1}AL^{-T}D^{-1/2}$ is also a nonsingular complex symmetric matrix. Thus in this case we can use LDL^T as a preconditioner.

We now make a brief explanation of the D-ILU preconditioner. The D-ILU uses $L = L_A, U = U_A$, where L_A is the strictly lower triangular part of A, and U_A strictly upper triangular part of A. Namely, the D-ILU only generates a diagonal matrix D. The cost of constructing the D-ILU preconditioning matrix is therefore much cheaper than that of the IC(0), while in our numerical examples the performance in terms of number of iterations tends to be worse than that of the IC(0). If the cost of constructing of the IC(0) becomes a bottleneck in terms of total CPU time as in Example 2, then the D-ILU will be an alternative.

The convergence plots show $\log_{10}$ of the relative residual 2-norm, $\log_{10} \|\boldsymbol{r}_n\|_2 / \|\boldsymbol{b}\|_2$, (on the vertical axis) versus the number of iteration steps (on the horizontal axis).

4.1. Example 1

The first problem comes from a Si(001) surface reconstruction simulation with 1024 atoms in [15] and is written as follows:

$$(\sigma_k I - H)\boldsymbol{x}^{(k)} = \boldsymbol{e}_1, \qquad k = 1, 2, \cdots, m,$$

where $\sigma_k = (k - 1 + i)/1000$, $H \in R^{2048\times 2048}$ is a symmetric matrix with 139264 entries, $\boldsymbol{e}_1 = (1, 0, \cdots, 0)^T$, and $m = 501$.

The numerical results are shown in Table 2. Here, "COCR" denotes that each system was solved by the COCR method, and "P1COCR" denotes that each system was solved

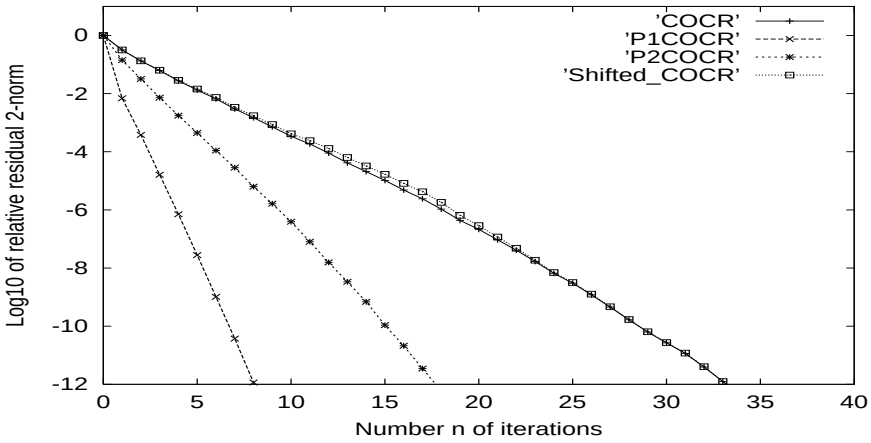

Figure 1: The convergence histories for Example 1 ($k = 401$).

Table 2: Numerical results for Example 1, where MV: total number of matrix-vector multiplications, Prec Sol: total number of solutions for systems with preconditioning matrices, Its Time: total time for iterations, Prec Time: total time for obtaining preconditioning matrices.

Solver	MV	Prec Sol	Its Time [s]	Prec Time [s]
COCR	13656	-	18.98	-
P1COCR	3604	3604	13.09	173.73
P2COCR	4997	4997	16.87	1.03
Shifted COCR	51	-	0.97	-

by the COCR method with the IC(0) preconditioner [7]. "P1COCR" needs to generate m preconditioning matrices, which may lead to large computational cost. Since in this example the step size is not so large, i.e. 0.001, we can consider reusing the preconditioning matrices. Based on this idea "P2COCR" denotes that the first 300 systems were solved by the COCR method with a preconditioning matrix for $k = 151$; the second 150 systems for $k = 351$; the last 51 systems for $k = 451$. "Shifted COCR"denotes that all systems were solved by the shifted COCR method with the seed system $s = 501$.

From Table 2, we can see that shifted COCR converged much faster than the other methods, which mainly comes from the fact that shifted COCR only required 51 matrix-vector multiplications.

The typical convergence histories for COCR, P1COCR, P2COCR and the shifted COCR are shown in Fig. 1, where the histories were obtained from the results for the system $k = 401$. From Fig. 1 we observe that, in terms of the number of steps, P1COCR and P2COCR converged faster than COCR and the shifted COCR. Since P2COCR does not use the preconditioning matrix obtained from the system $k = 401$, P2COCR required more iteration steps than P1COCR. The convergence history for COCR was very similar to that for the shifted COCR. We also used the shifted COCG method. The required CPU time was 0.99 sec., that was almost the same as that for the shifted COCR method.

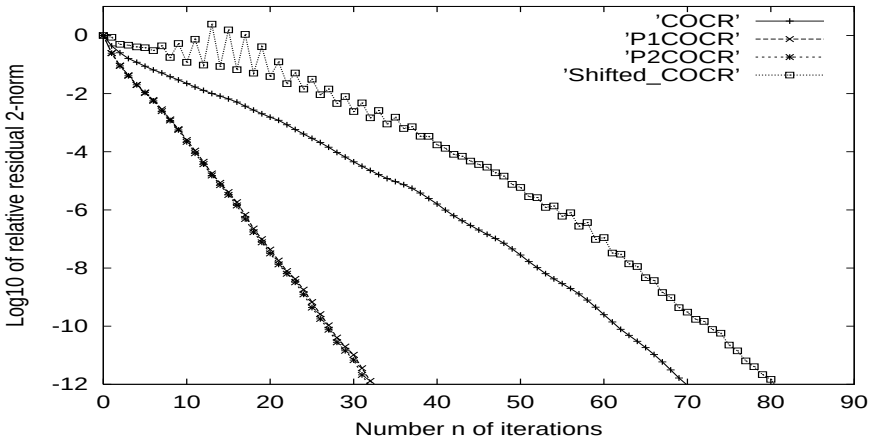

Figure 2: The convergence histories for Example 2 ($k = 51$).

Table 3: Numerical results for Example 2, where MV: total number of matrix-vector multiplications, Prec Sol: total number of solutions for systems with preconditioning matrices, Its Time: total time for iterations, Prec Time: total time for obtaining preconditioning matrices.

Solver	MV	Prec Sol	Its Time [s]	Prec Time [s]
COCR	7467	-	251.42	-
P1COCR	3848	3848	265.19	58.44
P2COCR	3877	3877	274.45	1.15
Shifted COCR	159	-	9.38	-

4.2. Example 2

The second problem comes from the electronic structure calculation of bulk fcc Cu with 1568 atoms in [15] and is written as follows:

$$(\sigma_k I - H)\boldsymbol{x}^{(k)} = \boldsymbol{e}_1, \quad k = 1, 2, \cdots, m,$$

where $\sigma_k = -0.5 + (k-1+i)/1000$, $H \in R^{14112\times 14112}$ is a symmetric matrix with 3924704 entries, $\boldsymbol{e}_1 = (1, 0, \cdots, 0)^T$, and $m = 101$.

The numerical results are shown in Table 3. Here, "P1COCR" denotes that each system was solved by the COCR method with the D-ILU preconditioner [8] instead of the IC(0), since in this example this preconditioner was more effective than the IC(0). "P2COCR" denotes that the first 50 systems were solved by the COCR method with a preconditioning matrix for $k = 1$; the last 51 systems for $k = 51$. For "Shifted COCR" we chose the seed system as $s = 101$.

Table 3 gives similar results to those shown in Table 2. We can see from Table 3 that shifted COCR was highly effective in solving shifted linear systems compared with other methods.

The typical convergence histories for COCR, P1COCR, P2COCR, and shifted COCR are shown in Fig. 2, where the histories were obtained from the results for the system $k = 51$.

From Fig. 2 we observe that shifted COCR showed jagged convergence behavior in the early stage of iteration steps, and then it converged with smooth behavior. We can see from Table 3 and Fig. 2 that shifted COCR tends to require more iteration steps to converge than COCR, although shifted COCR converged much faster than COCR. On the other hand, in terms of the number of iteration steps P1COCR and P2COCR converged at about half the number of iteration steps for COCR. We also used the shifted COCG method. The required CPU time was 8.73 sec., so it was slightly faster than the shifted COCR method.

5. Concluding Remarks

In the present paper, to solve complex symmetric shifted linear systems we developed a variant of the COCR method. From the numerical examples involving linear systems we found that the variant, namely the shifted COCR method, is much more efficient than the (preconditioned) COCR method. Furthermore, the shifted COCR method was found to be competitive with the shifted COCG method. Since the shifted COCR method has a different principle from that used in the shifted COCG method (and the shifted QMR-type methods), the shifted COCR method may be a method of choice for the case where the shifted COCG method (and the shifted QMR-type methods) fails.

Finally, we did not give convergence analysis of the shifted COCR method, since it is known (e.g. see [2], p. 428) that any complex matrix is similar to a complex symmetric matrix - and this implies that the convergence analysis is difficult in general, compared with some special matrices such as the real symmetric or Hermitian case. Even for other special matrices, e.g. complex symmetric circulant matrices or complex symmetric Toeplitz matrices, a meaningful analysis is an open problem.

Acknowledgments

The authors wish to thank Prof. Takeo FUJIWARA at the University of Tokyo and Prof. Takeo HOSHI at Tottori University for providing the authors with two test problems used in the numerical experiments. Finally, the authors thank two anonymous referees for useful comments that enhanced the quality of the paper.

This work was partially supported by MEXT. KAKENHI (Grant No. 21760058).

References

[1] J. van den Eshof, G.L.G. Sleijpen, Accurate conjugate gradient methods for families of shifted systems, Appl. Numer. Math., 49 (2004), pp. 17-37.

[2] R.W. Freund, Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 425-448.

[3] R.W. Freund, Solution of shifted linear systems by quasi-minimal residual iterations, in Numerical Linear Algebra, eds. L. Reichel, A. Ruttan and R.S. Varga, W. de Gruyter, Berlin, 1993, pp. 101-121.

[4] A. Frommer, Bi-CGSTAB(ℓ) for families of shifted linear systems, Computing, 70 (2003), pp. 87-109.
[5] A. Frommer, U. Grässner, Restarted GMRES for shifted linear systems, SIAM J. Sci. Comput., 19 (1998), pp. 15-26.
[6] A. Frommer, P. Maass, Fast CG-based methods for Tikhonov–Phillips regularization, SIAM J. Sci. Comput., 20 (1999), pp. 1831-1850.
[7] J.A. Meijerink, H.A. van der Vorst, An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix, Math. Comput., 31 (1977), pp. 148-162.
[8] C. Pommerell, Solution of large unsymmetric systems of linear equations, vol. 17 of Series in Micro-electronics, volume 17, Hartung-Gorre Verlag, Konstanz, 1992.
[9] Y. Saad, Iterative Methods for Sparse Linear Systems, 2nd ed., SIAM, Philadelphia, PA, 2003.
[10] V. Simoncini, D.B. Szyld, Recent computational developments in Krylov subspace methods for linear systems, Numer. Linear Algebra Appl., 14 (2007), pp. 1-59.
[11] T. Sogabe and S.-L. Zhang, An iterative method based on an A-biorthogonalization process for nonsymmetric linear systems, in: Proceedings of The 7th China-Japan Seminar on Numerical Mathematics, eds. Z.-C. Shi and H. Okamoto, Science Press, Beijing, 2006, pp. 120-130.
[12] T. Sogabe, S.-L. Zhang, A COCR method for solving complex symmetric linear systems, J. Comput. Appl. Math., 199 (2007), pp. 297-303.
[13] T. Sogabe, T. Hoshi, S.-L. Zhang, and T. Fujiwara, On a weighted quasi-residual minimization strategy of the QMR method for solving complex symmetric shifted linear systems, Electron. Trans. Numer. Anal., 31 (2008), pp. 126-140.
[14] E. Stiefel, Relaxationsmethoden bester Strategie zur Lösung linearer Gleichungssysteme, Comment. Math. Helv., 29 (1955), pp. 157-179.
[15] R. Takayama, T. Hoshi, T. Sogabe, S.-L. Zhang, T. Fujiwara, Linear algebraic calculation of Green's function for large-scale electronic structure theory, Phys. Rev. B, 73:165108 (2006), pp. 1-9.
[16] H.A. van der Vorst, J.B.M. Melissen, A Petrov-Galerkin type method for solving $A\boldsymbol{x} = \boldsymbol{b}$, where A is symmetric complex, IEEE Trans. Mag., 26:2 (1990), pp. 706-708.