NUMERICAL ANALYSIS AND MATHEMATICAL MODELLING
BANACH CENTER PUBLICATIONS, VOLUME 29
INSTITUTE OF MATHEMATICS
POLISH ACADEMY OF SCIENCES
WARSZAWA 1994

ON THE PRECONDITIONED BICONJUGATE GRADIENTS FOR SOLVING LINEAR COMPLEX EQUATIONS ARISING FROM FINITE ELEMENTS

MICHAL KŘÍŽEK and JAROSLAV MLÝNEK

Matematický ústav AV ČR
Žitná 25, 11567 Praha 1, Czech Republic
E-mail: KRIZEK@CSEARN.BITNET

The paper analyses the biconjugate gradient algorithm and its preconditioned version for solving large systems of linear algebraic equations with nonsingular sparse complex matrices. Special emphasis is laid on symmetric matrices arising from discretization of complex partial differential equations by the finite element method.

1. Introduction. Our concern in this paper is the solution of a large system of complex linear algebraic equations

$$Ax = b\,, \tag{1.1}$$

where A is a nonsingular $N \times N$ matrix. We shall assume that A is sparse, i.e., only $\mathcal{O}(N)$ entries of A are different from zero. The necessity of solving such systems arises when applying the finite element (or difference) method to approximate differential equations with complex coefficients and some boundary conditions. These problems come from electroengineering, acoustics, nuclear physics (Schrödinger equation), geophysics, plasma physics and other domains with time-periodic solutions (see e.g. [4, 8, 10]).

In solving factual three-dimensional problems the number of unknowns N can be so large that we are not able to solve (1.1) by direct methods based on the Gaussian elimination due to limitations of the internal fast computer memory. On the other hand, iterative methods enable us to store only nonzero entries of A and, of course, some information about their positions in A.

1991 *Mathematics Subject Classification*: 65N30, 65F10.
The paper is in final form and no version of it will be published elsewhere.

At the present time the most efficient iterative methods seem to be conjugate gradient type methods. The attractive feature of these methods is that they can usually be stopped when the number of iterations is much less than N. We shall deal with the biconjugate gradient method (BCG) which has, moreover, minimal requirements upon computer memory. This method has been firstly described by Fletcher [3, p. 80] for real nonsymmetric matrices. Its generalization to complex non-Hermitian matrices can be found in [6, 10]. We establish some other properties of the complex BCG method than in [6, 10] and derive a preconditioned biconjugate gradient method (PBCG). The efficiency of BCG and PBCG is demonstrated in solving a geophysical example.

2. Description and properties of the BCG method. Denote by α^C the conjugate complex number to a complex number α. Analogously, A^C will stand for the conjugate matrix to A. Further, let

$$A^H = (A^C)^T\,,$$

i.e., A^H is the conjugate transposed matrix to A. By $r^H = (r^C)^T$ we mean the row vector for a column vector r.

The *biconjugate gradient method* for the system (1.1) is defined in the following way. Let x_0 be an initial guess for the solution of (1.1) and let $\widetilde{r}_0$ be an arbitrary N-dimensional column vector such that $\widetilde{p}_0^H A p_0 \neq 0$ and $\widetilde{r}_0^H r_0 \neq 0$, where

$$p_0 = r_0 = b - Ax_0, \qquad \widetilde{p}_0 = \widetilde{r}_0\,. \tag{2.1}$$

Then we set

$$\alpha_k = \frac{\widetilde{r}_k^H r_k}{\widetilde{p}_k^H A p_k}\,, \tag{2.2}$$

$$x_{k+1} = x_k + \alpha_k p_k\,, \tag{2.3}$$

$$r_{k+1} = r_k - \alpha_k A p_k\,, \tag{2.4}$$

$$\widetilde{r}_{k+1} = \widetilde{r}_k - \alpha_k^C A^H \widetilde{p}_k\,, \tag{2.5}$$

$$\beta_k = \frac{\widetilde{r}_{k+1}^H r_{k+1}}{\widetilde{r}_k^H r_k}\,, \tag{2.6}$$

$$p_{k+1} = r_{k+1} + \beta_k p_k\,, \tag{2.7}$$

$$\widetilde{p}_{k+1} = \widetilde{r}_{k+1} + \beta_k^C \widetilde{p}_k, \qquad k = 0, 1, \ldots \tag{2.8}$$

Let

$$M = \inf\{k \in \{1, 2, \ldots\} \mid \widetilde{p}_k^H A p_k = 0 \text{ or } \widetilde{r}_k^H r_k = 0\}\,, \tag{2.9}$$

i.e., the BCG algorithm (2.1)–(2.8) *does not break down* within M iterations.

Before we prove that the BCG algorithm terminates in at most N iterations, we prove several auxiliary assertions.

LEMMA 2.1. *Let* $x_0, \dots, x_M$ *and* $r_0, \dots, r_M$ *be determined by* (2.1)–(2.8). *Then*

$$r_k = b - Ax_k, \qquad k = 0, \dots, M\,, \tag{2.10}$$

i.e., $r_0, \dots, r_M$ *represent a sequence of residual vectors.*

Proof. We prove (2.10) by induction. From (2.1) we see that r_0 is a residual vector. Suppose now that $r_k = b - Ax_k$ holds for some $k \in \{0, \dots, M-1\}$. Multiplying (2.3) by A, we have from (2.4)

$$\begin{aligned} r_{k+1} &= r_k - \alpha_k A p_k = b - Ax_k - \alpha_k A p_k \\ &= b - Ax_k - (Ax_{k+1} - Ax_k) = b - Ax_{k+1}. \ \blacksquare \end{aligned}$$

Remark 2.2. Let $\widetilde{x}_0$ and $\widetilde{b}$ be arbitrary N-dimensional column vectors, $\widetilde{r}_0 = \widetilde{b} - A^H \widetilde{x}_0$ and let

$$\widetilde{x}_{k+1} = \widetilde{x}_k + \alpha_k^C \widetilde{p}_k$$

be included in the BCG algorithm just after the relation (2.3). Then in a similar way to Lemma 2.1 we can prove that

$$\widetilde{r}_k = \widetilde{b} - A^H \widetilde{x}_k, \qquad k = 0, \dots, M\,, \tag{2.11}$$

i.e., $\widetilde{r}_0, \dots, \widetilde{r}_M$ are residual vectors associated with the problem

$$A^H \widetilde{x} = \widetilde{b}\,.$$

So, by the BCG method we can solve simultaneously the two systems $Ax = b$ and $A^H \widetilde{x} = \widetilde{b}$.

Remark 2.3. If A is a nonsingular Hermitian matrix, that is, $A = A^H$, and $\widetilde{r}_0 = r_0$, then the BCG method (2.1)–(2.8) leads to the standard *conjugate gradient method* with real coefficients α_k, β_k and vectors

$$\widetilde{p}_k = p_k, \quad \widetilde{r}_k = r_k, \qquad k = 0, \dots, M-1\,,$$

which are complex in general.

In the next theorem we show that the sequences $\{r_k\}$, $\{\widetilde{r}_k\}$ and $\{p_k\}$, $\{\widetilde{p}_k\}$ generated by the BCG method satisfy the *biorthogonality* and *biconjugacy condition*, respectively. This theorem can be found in [3, p. 80] for the real case.

THEOREM 2.4. *Let* (2.9) *hold. Then for all* $k, l \in \{0, \dots, M\}$, $k \neq l$, *we have*

$$\widetilde{r}_k^H r_l = 0 \tag{2.12}$$

and

$$\widetilde{p}_k^H A p_l = 0\,. \tag{2.13}$$

Proof. We prove the theorem by induction and only for $k < l$, as the case $k > l$ can be treated analogously. By (2.4), (2.1) and (2.2), we have

$$\widetilde{r}_0^H r_1 = \widetilde{r}_0^H r_0 - \alpha_0 \widetilde{p}_0^H A p_0 = 0\,, \tag{2.14}$$

and, by (2.7), (2.5), (2.2), (2.14) and (2.6),

$$\begin{aligned}\widetilde{p}_0^H A p_1 &= \widetilde{p}_0^H A r_1 + \beta_0 \widetilde{p}_0^H A p_0 \\ &= \frac{1}{\alpha_0}(\widetilde{r}_0^H - \widetilde{r}_1^H) r_1 + \frac{\beta_0}{\alpha_0} \widetilde{r}_0^H r_0 = \frac{1}{\alpha_0}(-\widetilde{r}_1^H r_1 + \beta_0 \widetilde{r}_0^H r_0) = 0\,.\end{aligned}$$

Now let l be a fixed positive integer less than M. Assuming that both (2.12) and (2.13) hold for all nonnegative $k < l$, we prove that they remain valid also if l is replaced by $l+1$. Using (2.4), (2.12), (2.8) and (2.13) for $k < l$, we obtain

$$\widetilde{r}_k^H r_{l+1} = \widetilde{r}_k^H r_l - \alpha_l \widetilde{r}_k^H A p_l = -\alpha_l (\widetilde{p}_k^H - \beta_{k-1} \widetilde{p}_{k-1}^H) A p_l = 0,$$

where $\beta_{-1} = 0$ and $\widetilde{p}_{-1} = 0$ if necessary, and from (2.4), (2.2), (2.8) and (2.13),

$$\widetilde{r}_l^H r_{l+1} = \widetilde{r}_l^H r_l - \alpha_l \widetilde{r}_l^H A p_l = \widetilde{r}_l^H r_l - \frac{\widetilde{r}_l^H r_l}{\widetilde{p}_l^H A p_l} (\widetilde{p}_l^H - \beta_{l-1} \widetilde{p}_{k-1}^H) A p_l = 0\,.$$

Combining (2.7), (2.5), (2.13) and (2.12) for $k < l$, we get

$$\widetilde{p}_k^H A p_{l+1} = \widetilde{p}_k^H A r_{l+1} + \beta_l \widetilde{p}_k^H A p_l = \frac{1}{\alpha_k}(\widetilde{r}_k^H - \widetilde{r}_{k+1}^H) r_{l+1} = 0\,,$$

and finally, from (2.7), (2.5), (2.2), (2.12) and (2.6),

$$\begin{aligned}\widetilde{p}_l^H A p_{l+1} &= \widetilde{p}_l^H A r_{l+1} + \beta_l \widetilde{p}_l^H A p_l \\ &= \frac{1}{\alpha_l}(\widetilde{r}_l^H - \widetilde{r}_{l+1}^H) r_{l+1} + \frac{\beta_l}{\alpha_l} \widetilde{r}_l^H r_l = \frac{1}{\alpha_l}(-\widetilde{r}_{l+1}^H r_{l+1} + \beta_l \widetilde{r}_l^H r_l) = 0. \ \blacksquare\end{aligned}$$

Corollary 2.5. *If* (2.9) *holds then*

$$\widetilde{r}_l^H p_k = \widetilde{p}_k^H r_l = 0 \qquad \text{for } 0 \le k < l \le M\,.$$

Proof. Using (2.7) repeatedly, we find that

$$p_k = r_k + \beta_{k-1} r_{k-1} + \beta_{k-1}\beta_{k-2} r_{k-2} + \ldots + \Big(\prod_{j=0}^{k-1} \beta_j\Big) r_0\,.$$

Thus (2.12) yields $\widetilde{r}_l^H p_k = 0$ for $k < l$. The second relation $\widetilde{p}_k^H r_l = 0$ follows similarly from (2.8) and (2.12). ∎

Lemma 2.6. *If* (2.9) *holds then the vectors* $\{r_l\}_{l=0}^{M-1}$ *are linearly independent* (*i.e.* $M \le N$ *necessarily*).

Proof. By (2.9),

$$\widetilde{r}_l^H r_l \neq 0 \qquad \forall l \in \{0, \ldots, M-l\}\,. \tag{2.15}$$

Suppose that $r_0, \ldots, r_{M-1}$ are linearly dependent. Then there exist complex numbers $c_0, \ldots, c_{M-1}$ and an integer $k \in \{0, \ldots, M-1\}$ so that

$$c_k \neq 0 \tag{2.16}$$

and

$$0 = \sum_{l=0}^{M-1} c_l r_l \,. \tag{2.17}$$

Now by (2.17) and (2.12),

$$0 = \sum_{l=0}^{M-1} c_l \widetilde{r}_k^H r_l = c_k \widetilde{r}_k^H r_k \,,$$

which contradicts (2.15) and (2.16). ∎

R e m a r k 2.7. Under the assumption (2.9) the BCG algorithm terminates in at most M iterations and from Lemma 2.6 we know that $M \leq N$. If $r_M = 0$ then, by Lemma 2.1, x_M is the true solution of the system (1.1). If $r_M \neq 0$ then, by (2.9), $\widetilde{p}_M^H A p_M = 0$ or $\widetilde{r}_M^H r_M = 0$, which means that the algorithm has broken down. However, the latter case happens very rarely in practical computations. In this case we have to restart the algorithm with other initial value (e.g. $x_0 = x_M$). The real difficulty may also come when the iteration is close to breakdown. For real symmetric and indefinite matrices the algorithm (2.1)–(2.8) can be modified so that it never breaks down — see [9, p. 1264] (cf. also [4, 5, 11]).

Let us introduce the so-called *Krylov space*

$$\mathcal{K}^k = \operatorname{span}(r_0, \ldots, r_{k-1}), \qquad k \in \{1, \ldots, M\} \,,$$

where span stands for the linear span. Sometimes $\mathcal{K}^k$ is called the right Krylov space (see [12, p. 485]) whereas $\operatorname{span}(\widetilde{r}_0, \ldots, \widetilde{r}_{k-1})$ is called the left Krylov space. The next lemma establishes several expressions of the right space. A similar lemma can be stated also for the left space.

Lemma 2.8. *Let* (2.9) *hold and let*

$$\begin{aligned}
\mathcal{K}_1^k &= \operatorname{span}(p_0, \ldots, p_{k-1}) \,, \\
\mathcal{K}_2^k &= \operatorname{span}(r_0, \ldots, A^{k-1} r_0), \\
\mathcal{K}_3^k &= \operatorname{span}(A(x^* - x_0), \ldots, A^k (x^* - x_0)) \,,
\end{aligned}$$

where x^* is the true solution of (1.1). *Then $\mathcal{K}^k = \mathcal{K}_1^k = \mathcal{K}_2^k = \mathcal{K}_3^k$.*

P r o o f. We prove the lemma by induction. The case $k = 1$ is evident due to (2.1). So let the assertion hold for some $k \in \{1, \ldots, M-1\}$. We divide the proof into the following four steps, showing successively that $\mathcal{K}^{k+1} \subset \mathcal{K}_1^{k+1} \subset \mathcal{K}_2^{k+1} \subset \mathcal{K}_3^{k+1} \subset \mathcal{K}^{k+1}$.

1. By (2.7), $r_k = p_k - \beta_{k-1} p_{k-1}$ and thus $\mathcal{K}^{k+1} \subset \mathcal{K}_1^{k+1}$.

2. We check whether $p_k \in \mathcal{K}_2^{k+1}$. By the induction assumption $r_{k-1} \in \mathcal{K}^k = \mathcal{K}_2^k \subset \mathcal{K}_2^{k+1}$, $p_{k-1} \in \mathcal{K}_1^k = K_2^k \subset \mathcal{K}_2^{k+1}$, and thus we obtain $A p_{k-1} \in \mathcal{K}_2^{k+1}$. Using now (2.7) and (2.4), we find that

$$p_k = r_k + \beta_{k-1} p_{k-1} = r_{k-1} - \alpha_{k-1} A p_{k-1} + \beta_{k-1} p_{k-1} \in \mathcal{K}_2^{k+1} \,.$$

Hence, $\mathcal{K}_1^{k+1} \subset \mathcal{K}_2^{k+1}$.

3. Since $r_0 = b - Ax_0 = A(x^* - x_0)$, we immediately see that $\mathcal{K}_2^{k+1} = \mathcal{K}_3^{k+1}$.

4. By Lemma 2.6, $\dim\mathcal{K}^{k+1} = k+1$. We already know that $\mathcal{K}^{k+1} \subset \mathcal{K}_3^{k+1}$. Hence, $\dim\mathcal{K}_3^{k+1} = k+1$ and thus necessarily $\mathcal{K}^{k+1} = \mathcal{K}_3^{k+1}$. ∎

3. The BCG algorithm for complex symmetric matrices. A finite element approximation of time-harmonic problems often leads to the system (1.1) whose matrix is symmetric — see e.g. [4, 8] or (5.2) below. Let us now derive the form of the BCG algorithm in this case.

THEOREM 3.1. *Let A be a nonsingular complex symmetric matrix (i.e. $A = A^T$), $\widetilde{r}_0 = r_0^C$ and let the sequences $\{r_k\}$, $\{\widetilde{r}_k\}$, $\{p_k\}$, $\{\widetilde{p}_k\}$ be generated by* (2.1)–(2.8). *Then*

$$\widetilde{r}_k = r_k^C \quad \text{for } k = 0, \dots, M-1\,, \tag{3.1}$$

and

$$\widetilde{p}_k = p_k^C \quad \text{for } k = 0, \dots, M-1\,. \tag{3.2}$$

Proof. By (2.1) and the assumption $\widetilde{r}_0 = r_0^C$ we see that the theorem is valid for $k = 0$. Further, suppose its validity for some $k \in \{0, \dots, M-2\}$; we prove it for $k+1$. By (2.5) and (2.4),

$$\widetilde{r}_{k+1} = \widetilde{r}_k - \alpha_k^C A^H \widetilde{p}_k = r_k^C - \alpha_k^C (A^T)^C p_k^C = r_k^C - \alpha_k^C A^C p_k^C = r_{k+1}^C\,.$$

To verify (3.2) we employ (2.8), (3.1) and (2.7):

$$\widetilde{p}_{k+1} = \widetilde{r}_{k+1} + \beta_k^C \widetilde{p}_k = r_{k+1}^C + \beta_k^C p_k^C = p_{k+1}^C\,. \;\blacksquare$$

COROLLARY 3.2. *For complex symmetric matrices the BCG algorithm* (2.1)–(2.8) *reduces to the form*

$$p_0 = r_0 = b - Ax_0\,, \tag{3.3}$$

$$\alpha_k = \frac{r_k^T r_k}{p_k^T A p_k}\,, \tag{3.4}$$

$$x_{k+1} = x_k + \alpha_k p_k\,, \tag{3.5}$$

$$x_{k+1} = r_k - \alpha_k A p_k\,, \tag{3.6}$$

$$\beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}\,, \tag{3.7}$$

$$p_{k+1} = r_{k+1} + \beta_k p_k, \quad k = 0, 1, \dots \tag{3.8}$$

Remark 3.3. Notice that only one matrix-vector multiplication is necessary in each iteration. Moreover, it is sufficient to store in computer memory only four vectors b, p, r, x (besides A). The vectors Ap_k, $k = 0, 1, \dots$, occurring in (3.4) and (3.6) can be currently stored in b. Contrary to Remark 2.3, the coefficients α_k and β_k are complex in general and $r_k^T r_k$ does not represent a scalar product in

the complex case. Note that the popular conjugate gradient type methods ORTHODIR, GMRES, ORTHOMIN and ORTHORES (see e.g. [14, p. 77]) require much more memory cells than the BCG method.

4. Preconditioned BCG method. Let A be a nonsingular complex $N \times N$ matrix and let Z be its "easily invertible" approximation (cf. Remark 4.2). Suppose that Z is of the form

$$Z = LU\,, \tag{4.1}$$

where L and U are lower and upper triangular matrices, respectively. Multiplying (1.1) by L^{-1} and setting

$$\widehat{x} = Ux\,, \tag{4.2}$$

$$\widehat{A} = L^{-1}AU^{-1}\,, \tag{4.3}$$

$$\widehat{b} = L^{-1}b\,, \tag{4.4}$$

we may write

$$\widehat{A}\widehat{x} = L^{-1}AU^{-1}Ux = L^{-1}Ax = L^{-1}b = \widehat{b}\,,$$

that is,

$$\widehat{A}\widehat{x} = \widehat{b}\,. \tag{4.5}$$

It is known that the condition number of $\widehat{A}$ has a considerable influence on the rate of convergence of all iterative methods. An appropriate choice of Z enables us to reduce essentially the condition number of $\widehat{A}$.

By the *preconditioned biconjugate gradient method* we shall mean the biconjugate gradient method (2.1)–(2.8) applied to the system (4.5). Thus formally we get:

$$\widehat{p}_0 = \widehat{r}_0 = \widehat{b} - \widehat{A}\widehat{x}_0, \quad \widehat{\widetilde{p}}_0 = \widehat{\widetilde{r}}_0\,, \tag{4.6}$$

$$\widehat{\alpha}_k = \frac{\widehat{\widetilde{r}}_k^H \widehat{r}_k}{\widehat{\widetilde{p}}_k^H \widehat{A}\widehat{p}_k}\,, \tag{4.7}$$

$$\widehat{x}_{k+1} = \widehat{x}_k + \widehat{\alpha}_k \widehat{p}_k\,, \tag{4.8}$$

$$\widehat{r}_{k+1} = \widehat{r}_k - \widehat{\alpha}_k \widehat{A}\widehat{p}_k\,, \tag{4.9}$$

$$\widehat{\widetilde{r}}_{k+1} = \widehat{\widetilde{r}}_k - \widehat{\alpha}_k^C \widehat{A}^H \widehat{\widetilde{p}}_k\,, \tag{4.10}$$

$$\widehat{\beta}_k = \frac{\widehat{\widetilde{r}}_{k+1}^H \widehat{r}_{k+1}}{\widehat{\widetilde{r}}_k^H \widehat{r}_k}\,, \tag{4.11}$$

$$\widehat{p}_{k+1} = \widehat{r}_{k+1} + \widehat{\beta}_k \widehat{p}_k\,, \tag{4.12}$$

$$\widehat{\widetilde{p}}_{k+1} = \widehat{\widetilde{r}}_{k+1} + \widehat{\beta}_k^C \widehat{\widetilde{p}}_k, \quad k = 0, 1, \ldots, \tag{4.13}$$

where $\widehat{x}_0$ is an initial guess for (4.5) and $\widehat{\widetilde{r}}_0$ is an arbitrary N-dimensional column vector. Throughout this section we again assume that the BCG algorithm (4.6)–

(4.13) does not break down. In this algorithm there occurs $\widehat{A}$ which is, however, not suitable for computer implementation. That is why, for $k = 0, 1, \ldots$, we set (cf. (4.2))

$$x_k = U^{-1}\widehat{x}_k \tag{4.14}$$

and introduce the following notation:

$$r_k = L\widehat{r}_k, \qquad \widetilde{r}_k = U^H\widehat{\widetilde{r}}_k\,, \tag{4.15}$$

$$s_k = U^{-1}\widehat{r}_k, \qquad \widetilde{s}_k = L^{-H}\widehat{\widetilde{r}}_k\,, \tag{4.16}$$

$$v_k = U^{-1}\widehat{p}_k, \qquad \widetilde{v}_k = L^{-H}\widehat{\widetilde{p}}_k\,, \tag{4.17}$$

$$\alpha_k = \widehat{\alpha}_k, \qquad \beta_k = \widehat{\beta}_k\,, \tag{4.18}$$

where $L^{-H} = (L^{-1})^H$. We will now try to eliminate from (4.6)–(4.13) all quantities with the symbol $\widehat{\ }$. Thus by (4.6), (4.15), (4.4), (4.3) and (4.14) we get

$$L^{-1}r_0 = L^{-1}b - L^{-1}AU^{-1}Ux_0 = L^{-1}(b - Ax_0)\,. \tag{4.19}$$

From (4.6), (4.16) and (4.17) we further have

$$Uv_0 = Us_0 = L^{-1}r_0\,, \tag{4.20}$$

$$L^H\widetilde{v}_0 = L^H\widetilde{s}_0 = U^{-H}\widetilde{r}_0\,. \tag{4.21}$$

The coefficient (4.7) can be, by (4.15)–(4.18) and (4.3), rewritten as follows:

$$\alpha_k = \frac{(L^H\widetilde{s}_k)^H L^{-1}r_k}{(L^H\widetilde{v}_k)^H L^{-1}AU^{-1}Uv_k} = \frac{\widetilde{s}_k^H r_k}{\widetilde{v}_k^H Av_k} \tag{4.22}$$

and the equation (4.8), by (4.14) and (4.17), as

$$Ux_{k+1} = Ux_k + \alpha_k Uv_k\,. \tag{4.23}$$

To rearrange (4.9) and (4.10), we employ (4.15), (4.17) and (4.3):

$$L^{-1}r_{k+1} = L^{-1}r_k - \alpha_k L^{-1}AU^{-1}Uv_k\,, \tag{4.24}$$

$$U^{-H}\widetilde{r}_{k+1} = U^{-H}\widetilde{r}_k - \alpha_k^C U^{-H}A^H L^{-H}L^H\widetilde{v}_k\,. \tag{4.25}$$

Finally, the coefficient (4.11) and the equations (4.12) and (4.13) can be transformed by means of (4.14)–(4.18) to the form

$$\beta_k = \frac{(L^H\widetilde{s}_{k+1})^H L^{-1}r_{k+1}}{(L^H\widetilde{s}_k)^H L^{-1}r_k} = \frac{\widetilde{s}_{k+1}^H r_{k+1}}{\widetilde{s}_k^H r_k}\,, \tag{4.26}$$

$$Uv_{k+1} = Us_{k+1} + \beta_k Uv_k\,, \tag{4.27}$$

$$L^H\widetilde{v}_{k+1} = L^H\widetilde{s}_{k+1} + \beta_k^C L^H\widetilde{v}_k\,. \tag{4.28}$$

From (4.1) and (4.15)–(4.28) we deduce the following corollary.

COROLLARY 4.1. *The preconditioned biconjugate gradient method (PBCG) for the system* (1.1) *may be written as follows*: *Let* x_0 *and* $\widetilde{r}_0$ *be arbitrary* N*-dimensional column vectors. Then we set*

$$r_0 = b - Ax_0\,, \tag{4.29}$$

$$v_0 = s_0 = Z^{-1} r_0 \,, \tag{4.30}$$

$$\widetilde{v}_0 = \widetilde{s}_0 = Z^{-H} \widetilde{r}_0 \,, \tag{4.31}$$

$$\alpha_k = \frac{\widetilde{s}_k^H r_k}{\widetilde{v}_k^H A v_k} \,, \tag{4.32}$$

$$x_{k+1} = x_k + \alpha_k v_k \,, \tag{4.33}$$

$$r_{k+1} = r_k - \alpha_k A v_k \,, \tag{4.34}$$

$$\widetilde{r}_{k+1} = \widetilde{r}_k - \alpha_k^C A^H \widetilde{v}_k \,, \tag{4.35}$$

$$s_{k+1} = Z^{-1} r_{k+1} \,, \tag{4.36}$$

$$\widetilde{s}_{k+1} = Z^{-H} \widetilde{r}_{k+1} \,, \tag{4.37}$$

$$\beta_k = \frac{\widetilde{s}_{k+1}^H r_{k+1}}{\widetilde{s}_k^H r_k} \,, \tag{4.38}$$

$$v_{k+1} = s_{k+1} + \beta_k v_k \,, \tag{4.39}$$

$$\widetilde{v}_{k+1} = \widetilde{s}_{k+1} + \beta_k^C \widetilde{v}_k, \qquad k = 0, 1, \ldots \tag{4.40}$$

R e m a r k 4.2. If A is sparse then the matrices L and U in (4.1) may be chosen so that they are sparse. The inversion of Z in (4.30) need not be performed, since the vector s_0 can be obtained more easily in the following two steps:

$$Ly = r_0 \,,$$
$$Us_0 = y \,,$$

which have not great pretensions to the number of arithmetic operations. Analogously we proceed in (4.31), (4.36) and (4.37).

R e m a r k 4.3. It follows from (4.29), (4.33) and (4.34) that

$$r_k = b - Ax_k, \qquad k = 0, 1, \ldots,$$

i.e., the sequence $\{r_k\}$ represents the residual vectors of the original equation (1.1). The proof is the same as in Lemma 2.1.

R e m a r k 4.4. If A is symmetric (or Hermitian) then Z can be chosen in the form $Z = LL^T$ (or $Z = LL^H$), where L is a lower triangular matrix. The algorithm (4.29)–(4.40) may then be simplified as in Corollary 3.2 (or Remark 2.3).

5. Numerical test. We compared the BCG method with classical iterative methods in solving a model geophysical problem which is thoroughly treated in [8, p. 163]. This problem is described by Helmholtz's partial differential equation

$$-\Delta u + \mathrm{i}\sigma u = f$$

in a rectangular domain $\Omega \subset \mathbb{R}^2$ with mixed boundary conditions. Here i stands for the imaginary unit, $f \in L^2(\Omega)$ and σ is a real piecewise constant function

with great jumps. Define a sesquilinear form

$$a(z,u) = \int_\Omega ((\operatorname{grad} z)^T \operatorname{grad} u^C + \mathrm{i}\sigma z u^C)\, d\Omega\,, \tag{5.1}$$

where z and u belong to the complex Sobolev space $H^1(\Omega)$. For the standard finite element piecewise bilinear and real basis functions $z^1, \dots, z^N$ the associated stiffness matrix

$$A = (a(z^i, z^j))_{i,j=1}^N \tag{5.2}$$

will be, due to (5.1), sparse complex and symmetric. Note that the matrix $A + A^H$ is moreover positive definite (cf. [15, p. 802]). Numerical tests show that the spectrum of A entirely lies inside the first quadrant near the real axis.

The next table contains the minimum number of iterations necessary to achieve the prescribed tolerance of the error and residual vector for $N = 740$. The true solution x^* of (1.1) has been obtained by the Gaussian elimination. The symbol $\|\cdot\|_\infty$ stands for the standard l^∞-norm (maximum norm). We have taken the initial guess $x_0 = 0$ in all the cases considered.

Method	$\|x^* - x_k\|_\infty < 5 \cdot 10^{-3}$	$\|r_k\|_\infty < 5 \cdot 10^{-4}$
Jacobi	∞	∞
Kaczmarz (orth. projections)	10892	9351
CG for system (5.3)	689	696
Gauss–Seidel	404	338
SOR with $\omega = 1.4 - 0.2i$	192	170
BCG (3.3)–(3.8)	93	108
PBCG by diagonal matrix	79	82
PBCG by incomplete factorization	39	43

The classical Jacobi method (see [13, p. 73]) has not converged. The Kaczmarz method of orthogonal projections (see [8, p. 89]) converges for any nonsingular matrix. Nevertheless, we see that its convergence can be extremely slow. The standard conjugate gradient method applied to the system (cf. [2, p. 68])

$$A^H A x = A^H b \tag{5.3}$$

(whose matrix is Hermitian but ill-conditioned) also converges very slowly. The number of iterations is almost equal to N. Moreover, it was necessary to use double precision in this case. The next two rows contain the results of the successive overrelaxation method [7] for $\omega = 1$ (Gauss–Seidel) and $\omega = 1.4 - 0.2\mathrm{i}$ (almost optimal). Note that the duration of one iteration of the proposed BCG algorithm (3.3)–(3.8) was only 1.2 times that for the Gauss–Seidel method. The last two rows illustrate the effect of preconditioning by the diagonal matrix $Z = \operatorname{diag}(A)$ (see [8, p. 222]) and by the incomplete Choleski factorization (cf. [1, p. 284]).

References

[1] O. Axelsson and V. A. Barker, *Finite Element Solution of Boundary Value Problems. Theory and Computation*, Academic Press, New York 1984.

[2] F. S. Beckman, *The solution of linear equations by the conjugate gradient method*, in: Mathematical Methods for Digital Computers, A. Ralston and H. S. Wilf (eds.), Wiley, New York 1960, 62–72.

[3] R. Fletcher, *Conjugate gradient methods for indefinite systems*, in: Proc. Dundee Conf. on Numerical Analysis, A. Dold and B.Eckmann (eds.), Lecture Notes in Math. 506, Springer, New York 1975, 73–89.

[4] R. W. Freund, *Conjugate gradient type methods for linear systems with complex symmetric coefficient matrices*, SIAM J. Sci. Statist. Comput. 13 (1992), 1–23.

[5] R. W. Freund and N. M. Nachtigal, *QMR: a quasi-minimal residual method for non-Hermitian linear systems*, RIACS Technical Report 90.51, NASA, Columbia 1990, 33 pp.

[6] D. A. H. Jacobs, *A generalization of the conjugate-gradient method to solve complex systems*, IMA J. Numer. Anal. 6 (1986), 447–452.

[7] A. Kiełbasiński, *Catalogue of linear algebra algorithms of the journal Numerische Mathematik*, Mat. Stos. 2 (1974), 5–13 (in Polish).

[8] M. Křížek and P. Neittaanmäki, *Finite Element Approximation of Variational Problems and Applications*, Longman, Harlow 1990.

[9] D. G. Luenberger, *Hyperbolic pairs in the conjugate gradients*, SIAM J. Appl. Math. 17 (1969), 1263–1267.

[10] Z. Mikić and E. C. Morse, *The use of a preconditioned bi-conjugate gradient method for hybrid plasma stability analysis*, J. Comput. Phys. 61 (1985), 154–185.

[11] B. N. Parlett, D. R. Taylor and Z. A. Liu, *A look-ahead Lanczos algorithm for symmetric matrices*, Math. Comp. 44 (1985), 105–124.

[12] Y. Saad, *The Lanczos biorthogonalization algorithm and other oblique projection methods for solving large unsymmetric systems*, SIAM J. Numer. Anal. 19 (1982), 485–506.

[13] R. S. Varga, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, New Jersey, 1962.

[14] H. A. van der Vorst and K. Dekker, *Conjugate gradient type methods and preconditioning*, J. Comput. Appl. Math. 24 (1988), 73–87.

[15] O. Widlund, *A Lanczos method for a class of nonsymmetric systems of linear equations*, SIAM J. Numer. Anal. 15 (1978), 801–812.