Journal of Computational and Applied Mathematics 226 (2009) 103–113

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

An extension of the conjugate residual method to nonsymmetric linear systems☆

T. Sogabe [a,*], M. Sugihara [b], S.-L. Zhang [a]

[a] Department of Computational Science and Engineering, Nagoya University, Furo-cho, Chikusa-ku, Nagoya, 464-8603, Japan

[b] Department of Mathematical Informatics, The University of Tokyo, Hongo, 7-3-1, Bunkyo-ku, Tokyo, 113-8656, Japan

ARTICLE INFO

Article history:
Received 10 April 2007
Received in revised form 23 December 2007

Keywords:
CG
CR
Bi-CG
Krylov subspace methods
Nonsymmetric linear systems
Lanczos algorithm
Coupled two-term recurrences

ABSTRACT

The Conjugate Gradient (CG) method and the Conjugate Residual (CR) method are Krylov subspace methods for solving symmetric (positive definite) linear systems. To solve nonsymmetric linear systems, the Bi-Conjugate Gradient (Bi-CG) method has been proposed as an extension of CG. Bi-CG has attractive short-term recurrences, and it is the basis for the successful variants such as Bi-CGSTAB. In this paper, we extend CR to nonsymmetric linear systems with the aim of finding an alternative basic solver. Numerical experiments show that the resulting algorithm with short-term recurrences often gives smoother convergence behavior than Bi-CG. Hence, it may take the place of Bi-CG for the successful variants.

1. Introduction

The Conjugate Gradient (CG) method [22] is one of the most well-known Krylov subspace methods for solving large and sparse linear systems $Ax = b$, where A is a symmetric positive definite matrix. If matrix A is symmetric indefinite, the Minimum Residual (MINRES) method [27] and the Conjugate Residual (CR) method described in [18,31] are applied. MINRES and CR are mathematically equivalent, and they enjoy a minimum norm residual property.

To solve nonsymmetric linear systems, generalized methods of CG, MINRES, and CR have been proposed such as Bi-CG [25,14], FOM [28], GMRES [32], and GCR [13]. Since Bi-CG is based on the two-sided Lanczos process [25], its approximate solutions can be computed with constant computational work and low memory requirements per iteration step; however, in many practical situations, Bi-CG shows irregular convergence behavior. On the other hand, GMRES and GCR enjoy the minimum norm residual property, and thus they show smooth convergence behavior. However, since GMRES and GCR are based on the Arnoldi process [1], the computational work and memory increase linearly with the number of iterations.

Many researchers have devoted much effort to improve the performance of Bi-CG. Sonneveld [35] proposed a variant of Bi-CG, referred to as CGS. van der Vorst [37] derived one of the most successful variants of Bi-CG, known as Bi-CGSTAB. Based on the idea of Bi-CGSTAB, various generalized methods have been proposed such as Bi-CGSTAB2 [20], Bi-CGSTAB(ℓ) [33], and GPBi-CG [39]. On the other hand, QMR [16] that is closely related to Bi-CG is an attractive solver in that QMR shows smoother convergence behavior than Bi-CG and it can remedy pivot breakdown and may avoid Lanczos breakdown by a

☆ This work was partially supported by KAKENHI (Nos 18760063 and 19560065).

* Corresponding author.
E-mail address: sogabe@na.cse.nagoya-u.ac.jp (T. Sogabe).

0377-0427/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2008.05.018

look-ahead strategy. Based on QMR, a transpose-free variant of QMR, called TFQMR [15], has been proposed. Using the idea of TFQMR and Bi-CGSTAB, QMRCGSTAB [10] and QMRCGSTAB(k) [36] have been proposed.

From the above, we see that Bi-CG plays a very important role in the successful variants such as Bi-CGSTAB, Bi-CGSTAB(ℓ), and GPBi-CG. Furthermore, Bi-CG is closely related to QMR in that approximate solutions and the residual 2-norms of QMR without look-ahead strategy can be obtained from approximate solutions and residual 2-norms of Bi-CG [17]. Hence, Bi-CG can be regarded as an important basic method for recent successful solvers. This motivated us to find another basic method with short-term recurrences. In this paper, we direct our attention to CR and extend it to nonsymmetric linear systems in order to develop another basic solver.

The paper is organized as follows: in the next section, we first describe a simple derivation of Bi-CG. Then, based on the derivation, we extend CR to nonsymmetric linear systems. In Section 3, we discuss some properties of the extended algorithm (referred to as Bi-CR), and then we describe a relation among Bi-CR and some Krylov subspace methods. In Section 4, we report some numerical experiments. Finally, we present conclusions and ideas for future work in Section 5.

Throughout this paper, A^{T} denotes the transpose of A, (x, y) denotes the dot product given by $x^{\mathrm{T}}y$, and we use the notation

$$K_n(A, r_0) := \mathrm{span}\{r_0, Ar_0, \ldots, A^{n-1}r_0\}$$

for the n-dimensional Krylov subspace generated by A and r_0.

2. An extension of CR without loss of short-term recurrence property

In this section, we describe one of the simplest derivations of Bi-CG, and then CR is extended to nonsymmetric linear systems by analogously using this derivation process.

2.1. H. A. van der Vorst's derivation of Bi-CG

The Bi-CG method is a Krylov subspace method for solving nonsymmetric linear systems of the form

$$Ax = b, \tag{1}$$

where A is an $N \times N$ real nonsymmetric matrix and b is an N-vector. The algorithm of Bi-CG is well known, and there are several ways to derive it. Recently, one of the simplest derivations has been given in [38, pp. 97–98], which was inspired by [23]. In this subsection, we give the details of his derivation.

First, using (1) and a dual linear system $A^{\mathrm{T}}x^* = b^*$, we consider the following $2N \times 2N$ symmetric linear system:

$$\begin{bmatrix} O & A \\ A^{\mathrm{T}} & O \end{bmatrix} \begin{bmatrix} x^* \\ x \end{bmatrix} = \begin{bmatrix} b \\ b^* \end{bmatrix}, \quad \text{or} \quad \tilde{A}\tilde{x} = \tilde{b}. \tag{2}$$

If we apply the algorithm of CG with the preconditioner

$$\tilde{M} = \begin{bmatrix} O & I \\ I & O \end{bmatrix}, \quad I\text{: Identity matrix}, \tag{3}$$

to (2), then the resulting algorithm at the nth iteration step can be written as

$$\begin{aligned}
&\tilde{p}_n^{\mathrm{CG}} = \tilde{M}^{-1}\tilde{r}_n^{\mathrm{CG}} + \beta_{n-1}\tilde{p}_{n-1}^{\mathrm{CG}}, \\
&\alpha_n = \frac{(\tilde{M}^{-1}\tilde{r}_n^{\mathrm{CG}}, \tilde{r}_n^{\mathrm{CG}})}{(\tilde{p}_n^{\mathrm{CG}}, \tilde{A}\tilde{p}_n^{\mathrm{CG}})}, \\
&\tilde{x}_{n+1}^{\mathrm{CG}} = \tilde{x}_n^{\mathrm{CG}} + \alpha_n\tilde{p}_n^{\mathrm{CG}}, \\
&\tilde{r}_{n+1}^{\mathrm{CG}} = \tilde{r}_n^{\mathrm{CG}} - \alpha_n\tilde{A}\tilde{p}_n^{\mathrm{CG}}, \\
&\beta_n = \frac{(\tilde{M}^{-1}\tilde{r}_{n+1}^{\mathrm{CG}}, \tilde{r}_{n+1}^{\mathrm{CG}})}{(\tilde{M}^{-1}\tilde{r}_n^{\mathrm{CG}}, \tilde{r}_n^{\mathrm{CG}})}.
\end{aligned}$$

Substituting $\tilde{M}^{-1}(= \tilde{M})$ of (3) and the vectors

$$\tilde{x}_n^{\mathrm{CG}} := \begin{bmatrix} x_n^{\mathrm{BCG}*} \\ x_n^{\mathrm{BCG}} \end{bmatrix}, \qquad \tilde{r}_n^{\mathrm{CG}} := \begin{bmatrix} r_n^{\mathrm{BCG}} \\ r_n^{\mathrm{BCG}*} \end{bmatrix}, \qquad \tilde{p}_n^{\mathrm{CG}} := \begin{bmatrix} p_n^{\mathrm{BCG}*} \\ p_n^{\mathrm{BCG}} \end{bmatrix}$$

into the previous recurrences, we readily obtain the following Bi-CG algorithm:

Algorithm 1 (*Bi-CG*).

x_0^{BCG} is an initial guess, $r_0^{\mathrm{BCG}} = b - Ax_0^{\mathrm{BCG}}$,
choose $r_0^{\mathrm{BCG}*}$ (for example, $r_0^{\mathrm{BCG}*} = r_0^{\mathrm{BCG}}$),
set $p_{-1}^{\mathrm{BCG}*} = p_{-1}^{\mathrm{BCG}} = 0,\ \beta_{-1} = 0$,
for $n = 0, 1, \ldots,$ until convergence, do:

$$p_n^{\mathrm{BCG}} = r_n^{\mathrm{BCG}} + \beta_{n-1} p_{n-1}^{\mathrm{BCG}},$$
$$p_n^{\mathrm{BCG}*} = r_n^{\mathrm{BCG}*} + \beta_{n-1} p_{n-1}^{\mathrm{BCG}*},$$
$$\alpha_n = \frac{(r_n^{\mathrm{BCG}*}, r_n^{\mathrm{BCG}})}{(p_n^{\mathrm{BCG}*}, Ap_n^{\mathrm{BCG}})},$$
$$x_{n+1}^{\mathrm{BCG}} = x_n^{\mathrm{BCG}} + \alpha_n p_n^{\mathrm{BCG}},$$
$$r_{n+1}^{\mathrm{BCG}} = r_n^{\mathrm{BCG}} - \alpha_n A p_n^{\mathrm{BCG}},$$
$$r_{n+1}^{\mathrm{BCG}*} = r_n^{\mathrm{BCG}*} - \alpha_n A^{\mathrm{T}} p_n^{\mathrm{BCG}*},$$
$$\beta_n = \frac{(r_{n+1}^{\mathrm{BCG}*}, r_{n+1}^{\mathrm{BCG}})}{(r_n^{\mathrm{BCG}*}, r_n^{\mathrm{BCG}})}.$$

Furthermore, van der Vorst mentioned in [38, p.98] that the preconditioned Bi-CG can be derived from the above framework with the preconditioner

$$\tilde{M} = \begin{bmatrix} O & K \\ K^{\mathrm{T}} & O \end{bmatrix}. \tag{4}$$

In the next subsection, we extend the CR method to nonsymmetric linear systems by the use of this framework.

2.2. An extension of CR

We see from Section 2.1 that the extended algorithm of CG is obtained by applying the preconditioned CG method to form (2), and that the resulting algorithm (Bi-CG) has attractive coupled two-term recurrences. Analogously, in this subsection, we consider applying the preconditioned CR method to form (2).

Based on the unpreconditioned CR algorithm described in [31, p.194], we can obtain the preconditioned algorithm given below.

Algorithm 2 (*Preconditioned CR*).

x_0^{CR} is an initial guess, $r_0^{\mathrm{CR}} = b - Ax_0^{\mathrm{CR}}$,
set $p_{-1} = 0,\ \beta_{-1} = 0$,
for $n = 0, 1, \ldots,$ until convergence, do:

$$p_n^{\mathrm{CR}} = M^{-1} r_n^{\mathrm{CR}} + \beta_{n-1} p_{n-1}^{\mathrm{CR}},$$
$$\alpha_n = \frac{(M^{-1} r_n^{\mathrm{CR}}, AM^{-1} r_n^{\mathrm{CR}})}{(M^{-1} Ap_n^{\mathrm{CR}}, Ap_n^{\mathrm{CR}})},$$
$$x_{n+1}^{\mathrm{CR}} = x_n^{\mathrm{CR}} + \alpha_n p_n^{\mathrm{CR}},$$
$$r_{n+1}^{\mathrm{CR}} = r_n^{\mathrm{CR}} - \alpha_n A p_n^{\mathrm{CR}},$$
$$\beta_n = \frac{(M^{-1} r_{n+1}^{\mathrm{CR}}, AM^{-1} r_{n+1}^{\mathrm{CR}})}{(M^{-1} r_n^{\mathrm{CR}}, AM^{-1} r_n^{\mathrm{CR}})}.$$

For the practical implementation, two recurrences, $Ap_n^{\mathrm{CR}} = AM^{-1} r_n^{\mathrm{CR}} + \beta_{n-1} Ap_{n-1}^{\mathrm{CR}}$ and $M^{-1} r_{n+1}^{\mathrm{CR}} = M^{-1} r_n^{\mathrm{CR}} - \alpha_n M^{-1} Ap_n^{\mathrm{CR}}$, are added to Algorithm 2 to reduce the number of matrix–vector multiplications and the preconditioner solve at each iteration step.

Next, we apply Algorithm 2 with the preconditioner (3) to form (2) so that we have

$$\tilde{p}_n^{\mathrm{CR}} = \tilde{M}^{-1} \tilde{r}_n^{\mathrm{CR}} + \beta_{n-1} \tilde{p}_{n-1}^{\mathrm{CR}},$$
$$\alpha_n = \frac{(\tilde{M}^{-1} \tilde{r}_n^{\mathrm{CR}}, \tilde{A}\tilde{M}^{-1} \tilde{r}_n^{\mathrm{CR}})}{(M^{-1} \tilde{A} \tilde{p}_n^{\mathrm{CR}}, \tilde{A} \tilde{p}_n^{\mathrm{CR}})},$$

$$\tilde{x}_{n+1}^{\mathrm{CR}} = \tilde{x}_n^{\mathrm{CR}} + \alpha_n \tilde{p}_n^{\mathrm{CR}},$$
$$\tilde{r}_{n+1}^{\mathrm{CR}} = \tilde{r}_n^{\mathrm{CR}} - \alpha_n \tilde{A}\tilde{p}_n^{\mathrm{CR}},$$
$$\beta_n = \frac{(\tilde{M}^{-1}\tilde{r}_{n+1}^{\mathrm{CR}}, \tilde{A}\tilde{M}^{-1}\tilde{r}_{n+1}^{\mathrm{CR}})}{(\tilde{M}^{-1}\tilde{r}_n^{\mathrm{CR}}, \tilde{A}\tilde{M}^{-1}\tilde{r}_n^{\mathrm{CR}})}.$$

Substituting $\tilde{M}^{-1}(=\tilde{M})$ of (3) and the vectors

$$\tilde{x}_n^{\mathrm{CR}} := \begin{bmatrix} x_n^* \\ x_n \end{bmatrix}, \qquad \tilde{r}_n^{\mathrm{CR}} := \begin{bmatrix} r_n \\ r_n^* \end{bmatrix}, \qquad \tilde{p}_n^{\mathrm{CR}} := \begin{bmatrix} p_n^* \\ p_n \end{bmatrix}$$

into the previous recurrences, we readily obtain the new algorithm. Since Bi-CG is obtained from the preconditioned CG and the algorithm is obtained from the preconditioned CR, we call it Bi-CR.

Algorithm 3 (*Bi-CR*).

x_0 is an initial guess, $r_0 = b - Ax_0$,
choose r_0^* (for example, $r_0^* = r_0$),
set $p_{-1}^* = p_{-1} = 0,\ \beta_{-1} = 0$,
for $n = 0, 1, \ldots,$ until convergence, do:

$$p_n = r_n + \beta_{n-1} p_{n-1}, \tag{5}$$
$$p_n^* = r_n^* + \beta_{n-1} p_{n-1}^*, \tag{6}$$
$$(Ap_n = Ar_n + \beta_{n-1} Ap_{n-1},)$$
$$\alpha_n = \frac{(r_n^*, Ar_n)}{(A^{\mathrm{T}} p_n^*, Ap_n)}, \tag{7}$$
$$x_{n+1} = x_n + \alpha_n p_n,$$
$$r_{n+1} = r_n - \alpha_n Ap_n,$$
$$r_{n+1}^* = r_n^* - \alpha_n A^{\mathrm{T}} p_n^*,$$
$$\beta_n = \frac{(r_{n+1}^*, Ar_{n+1})}{(r_n^*, Ar_n)}. \tag{8}$$

Here, $Ap_n = Ar_n + \beta_{n-1} Ap_{n-1}$ is newly added to reduce the number of matrix–vector multiplications per iteration step. We see from Algorithm 3 that the approximate solution x_n can be generated by coupled two-term recurrences, and if the coefficient matrix is symmetric, Bi-CR reduces to CR.

Furthermore, if we apply Algorithm 2 with the preconditioner (4) to (2), then we obtain the following preconditioned Bi-CR:

Algorithm 4 (*Preconditioned Bi-CR*).

x_0 is an initial guess, $r_0 = b - Ax_0$,
choose r_0^* (for example, $r_0^* = r_0$),
set $p_{-1}^* = p_{-1} = 0,\ \beta_{-1} = 0$,
for $n = 0, 1, \ldots,$ until convergence, do:

$$p_n = K^{-1} r_n + \beta_{n-1} p_{n-1},$$
$$p_n^* = K^{-T} r_n^* + \beta_{n-1} p_{n-1}^*,$$
$$(Ap_n = AK^{-1} r_n + \beta_{n-1} Ap_{n-1},)$$
$$\alpha_n = \frac{(K^{-T} r_n^*, AK^{-1} r_n)}{(K^{-T} A^{\mathrm{T}} p_n^*, Ap_n)},$$
$$x_{n+1} = x_n + \alpha_n p_n,$$
$$r_{n+1} = r_n - \alpha_n Ap_n,$$
$$r_{n+1}^* = r_n^* - \alpha_n A^{\mathrm{T}} p_n^*,$$
$$(K^{-T} r_{n+1}^* = K^{-T} r_n^* - \alpha_n K^{-T} A^{\mathrm{T}} p_n^*,)$$
$$\beta_n = \frac{(K^{-T} r_{n+1}^*, AK^{-1} r_{n+1})}{(K^{-T} r_n^*, AK^{-1} r_n)}.$$

Table 1
Summary of cost per iteration step

Method	Dot product	AXPY	Matrix–vector multiplication	Preconditioner solve
Bi-CG	2	5	$1+1$	$1+1$
Bi-CR	2	6 or 7	$1+1$	$1+1$

Here, let us remark on a name Bi-CR. In [9], a bi-conjugate residual (BCR) method is introduced with the same name as our method; however, the two algorithms are mathematically different since Algorithm 3 generates $x_n - x_0 \in K_n(A, r_0)$ while BCR generates $x_n^{\mathrm{BCR}} - x_0^{\mathrm{BCR}} \notin K_n(A, r_0^{\mathrm{BCR}})$ for nonsymmetric linear systems.

At the end of this section, we show the computational cost for Bi-CG and Bi-CR in Table 1. "AXPY" denotes addition of scaled vectors, "6 or 7" denotes "6" for the unpreconditioned Bi-CR and "7" for the preconditioned one, and "$1+1$" denotes 1 multiplication with the matrix and 1 with its transpose.

3. Some properties of Bi-CR and relation among Bi-CR and some Krylov subspace methods

In this section, we discuss some properties of Bi-CR, and then give another derivation using the results.

3.1. Some properties

Observing Algorithm 3, we see that the four iterates r_n, p_n, r_n^*, and p_n^* can be expressed as

$$r_n = R_n(A)r_0, \qquad p_n = P_n(A)r_0, \tag{9}$$

$$r_n^* = R_n(A^{\mathrm{T}})r_0^*, \qquad p_n^* = P_n(A^{\mathrm{T}})r_0^*, \tag{10}$$

where R_n and P_n are polynomials of degree n satisfying

$$R_0(\lambda) := 1, \qquad P_0(\lambda) := 1,$$
$$R_n(\lambda) := R_{n-1}(\lambda) - \alpha_{n-1}\lambda P_{n-1}(\lambda),$$
$$P_n(\lambda) := R_n(\lambda) + \beta_{n-1}P_{n-1}(\lambda), \quad \text{for } n = 1, 2, \ldots.$$

From (9) and (10), and Algorithm 3, the following results are obtained if breakdown does not occur:

Theorem 3.1. *For $i \neq j$, the following bi-orthogonality properties hold:*

$$(r_i^*, Ar_j) = 0, \tag{11}$$

$$(A^{\mathrm{T}}p_i^*, Ap_j) = 0. \tag{12}$$

Proof. It follows from (9) and (10) that $(r_i^*, Ar_j) = (R_i(A^{\mathrm{T}})r_0^*, AR_j(A)r_0) = (R_j(A^{\mathrm{T}})r_0^*, AR_i(A)r_0) = (r_j^*, Ar_i)$. Similarly, from (12) we obtain $(A^{\mathrm{T}}p_i^*, Ap_j) = (A^{\mathrm{T}}p_j^*, Ap_i)$. Hence, the statements of (11) and (12) are equivalent with

$$(r_i^*, Ar_j) = 0 \quad \text{and} \quad (A^{\mathrm{T}}p_i^*, Ap_j) = 0, \quad \text{for all } j < i. \tag{13}$$

Now, we give a proof of (13) by induction. Since the trivial case $i = 1$ is obvious from Algorithm 3, we assume that property (13) holds for $j < i \leq k$. Then, we show that

$$(r_{k+1}^*, Ar_j) = 0, \tag{14}$$

$$(A^{\mathrm{T}}p_{k+1}^*, Ap_j) = 0. \tag{15}$$

First, let us show (14). For the case $j < k$ it follows from the above assumption that

$$\begin{aligned}(r_{k+1}^*, Ar_j) &= (r_k^*, Ar_j) - \alpha_k(A^{\mathrm{T}}p_k^*, Ar_j)\\ &= -\alpha_k(A^{\mathrm{T}}p_k^*, Ar_j)\\ &= -\alpha_k(A^{\mathrm{T}}p_k^*, Ap_j) - \alpha_k\beta_{j-1}(A^{\mathrm{T}}p_k^*, Ap_{j-1})\\ &= 0.\end{aligned}$$

For the case $j = k$ we obtain

$$\begin{aligned}(r_{k+1}^*, Ar_k) &= (r_k^*, Ar_k) - \alpha_k(A^{\mathrm{T}}p_k^*, Ar_k)\\ &= (r_k^*, Ar_k) - \alpha_k(A^{\mathrm{T}}p_k^*, Ap_k) - \alpha_k\beta_{k-1}(A^{\mathrm{T}}p_k^*, Ap_{k-1})\\ &= (r_k^*, Ar_k) - \alpha_k(A^{\mathrm{T}}p_k^*, Ap_k)\\ &= 0\end{aligned}$$

from the computational formula of α_k of (7). Next, we show (15). For the case $j < k$ it follows from the first result of the proof that

$$(A^{\mathrm{T}}p^*_{k+1}, Ap_j) = (A^{\mathrm{T}}r^*_{k+1}, Ap_j) + \beta_k(A^{\mathrm{T}}p^*_k, Ap_j) = \frac{1}{\alpha_j}(A^{\mathrm{T}}r^*_{k+1}, r_j - r_{j+1}) = 0.$$

For the case $j = k$ we obtain

$$\begin{aligned}(A^{\mathrm{T}}p^*_{k+1}, Ap_k) &= (A^{\mathrm{T}}r^*_{k+1}, Ap_k) + \beta_k(A^{\mathrm{T}}p^*_k, Ap_k)\\ &= \frac{1}{\alpha_k}(A^{\mathrm{T}}r^*_{k+1}, r_k - r_{k+1}) + \beta_k(A^{\mathrm{T}}p^*_k, Ap_k)\\ &= -\frac{1}{\alpha_k}(A^{\mathrm{T}}r^*_{k+1}, r_{k+1}) + \beta_k(A^{\mathrm{T}}p^*_k, Ap_k)\\ &= 0\end{aligned}$$

from the computational formulas of α_k of (7) and β_k of (8). □

Corollary 3.2. *Some further properties of Bi-CR are*

$$(r^*_i, Ap_j) = 0 \quad \text{for } i > j, \tag{16}$$

$$(r^*_i, Ar_i) = (r^*_i, Ap_i), \tag{17}$$

$$(A^{\mathrm{T}}r^*_i, Ap_i) = (A^{\mathrm{T}}p^*_i, Ap_i). \tag{18}$$

Proof. First, we give a proof of (16). From the recurrence (5) it follows that $(r^*_i, Ap_j) = (r^*_i, Ar_j) + \beta_{j-1}(r^*_i, Ap_{j-1})$, and thus from property (11) we obtain $(r^*_i, Ap_j) = \beta_{j-1}(r^*_i, Ap_{j-1})$. Applying this process recursively, we finally obtain $(r^*_i, Ap_j) = \beta_{j-1}\beta_{j-2}\cdots\beta_0(r^*_i, Ap_0)$. Hence, from $p_0 = r_0$ and (11), property (16) is established.

Second, we give a proof of (17). From the recurrence (5) it follows that $(r^*_i, Ar_i) = (r^*_i, Ap_i) - \beta_{i-1}(r^*_i, Ap_{i-1})$. Since the second term is zero by (16), property (17) is established.

Finally, we give a proof of (18). From the recurrence (6) it follows that $(A^{\mathrm{T}}r^*_i, Ap_i) = (A^{\mathrm{T}}p^*_i, Ap_i) - \beta_{i-1}(A^{\mathrm{T}}p^*_{i-1}, Ap_i)$. Since the second term is zero from (12), property (18) is established. □

We see from the algorithm of Bi-CR that it can be also regarded as the algorithm of Bi-CG with formal inner product $(y^*, y)_A := (y^*)^{\mathrm{T}}Ay$. In this sense, M. Gutknecht has developed Bi-CG for $(y^*, y)_B$, where $BA = AB$. The theoretical results are given in [21].

3.2. Relation among Bi-CR and some Krylov subspace methods

In this section, we describe an algorithm of Bi-CR for non-Hermitian linear systems and show that Bi-CR reduces to CR when the coefficient matrix is Hermitian and that Bi-CR reduces to COCR [34] when the coefficient matrix is complex symmetric. Here we denote by $\langle x, y\rangle := x^H y = \bar{x}^{\mathrm{T}}y$ the standard dot product of two complex vectors.

From Algorithm 3, we readily obtain the algorithm of Bi-CR for non-Hermitian linear systems below.

Algorithm 5 (*Bi-CR for Non-Hermitian Linear Systems*).

x_0 is an initial guess, $r_0 = b - Ax_0$,
choose r^*_0 (for example, $r^*_0 = r_0$),
set $p^*_{-1} = p_{-1} = 0,\ \beta_{-1} = 0$,
for $n = 0, 1, \ldots,$ until convergence, do:

$$\begin{aligned}
&p_n = r_n + \beta_{n-1}p_{n-1},\\
&p^*_n = r^*_n + \bar{\beta}_{n-1}p^*_{n-1},\\
&(Ap_n = Ar_n + \beta_{n-1}Ap_{n-1},)\\
&\alpha_n = \frac{\langle r^*_n, Ar_n\rangle}{\langle A^H p^*_n, Ap_n\rangle},\\
&x_{n+1} = x_n + \alpha_n p_n,\\
&r_{n+1} = r_n - \alpha_n Ap_n,\\
&r^*_{n+1} = r^*_n - \bar{\alpha}_n A^H p^*_n,\\
&\beta_n = \frac{\langle r^*_{n+1}, Ar_{n+1}\rangle}{\langle r^*_n, Ar_n\rangle}.
\end{aligned}$$

We can see that the above algorithm with the choice $r_0^* = r_0$ reduces to the algorithm of CR when the coefficient matrix A is Hermitian since the choice $r_0^* = r_0$ leads to $r_n^* = r_n$, $p_n^* = p_n$, $\bar{\alpha}_n = \alpha_n$, and $\bar{\beta}_n = \beta_n$.

Algorithm 6 (*CR for Hermitian Linear Systems*).

x_0 is an initial guess, $r_0 = b - Ax_0$,
set $p_{-1} = 0,\ \beta_{-1} = 0$,
for $n = 0, 1, \ldots,$ until convergence, do:
$p_n = r_n + \beta_{n-1} p_{n-1}$,
$(Ap_n = Ar_n + \beta_{n-1} Ap_{n-1},)$
$\alpha_n = \dfrac{\langle r_n, Ar_n\rangle}{\langle Ap_n, Ap_n\rangle}$,
$x_{n+1} = x_n + \alpha_n p_n$,
$r_{n+1} = r_n - \alpha_n Ap_n$,
$\beta_n = \dfrac{\langle r_{n+1}, Ar_{n+1}\rangle}{\langle r_n, Ar_n\rangle}$.

Similarly, when the coefficient matrix A is not Hermitian but symmetric, i.e. $A = A^{\mathrm{T}} \neq A^H$, we can obtain COCR from Algorithm 5 with the choice $r_0^* = \bar{r}_0$ since this choice leads to $r_n^* = \bar{r}_n,\ p_n^* = \bar{p}_n$.

Algorithm 7 (*COCR for Complex Symmetric Linear Systems*).

x_0 is an initial guess, $r_0 = b - Ax_0$,
set $p_{-1} = 0,\ \beta_{-1} = 0$,
for $n = 0, 1, \ldots,$ until convergence, do:
$p_n = r_n + \beta_{n-1} p_{n-1}$,
$(Ap_n = Ar_n + \beta_{n-1} Ap_{n-1},)$
$\alpha_n = \dfrac{\langle \bar{r}_n, Ar_n\rangle}{\langle \overline{Ap}_n, Ap_n\rangle}$,
$x_{n+1} = x_n + \alpha_n p_n$,
$r_{n+1} = r_n - \alpha_n Ap_n$,
$\beta_n = \dfrac{\langle \bar{r}_{n+1}, Ar_{n+1}\rangle}{\langle \bar{r}_n, Ar_n\rangle}$.

It is known from [34] that COCR is equivalent with CR when the coefficient matrix A is real symmetric.

From the above, we see that Bi-CR includes CR and COCR. Hence, Bi-CR can be regarded as extensions of CR and COCR.

4. Numerical experiments

In this section, we report the results of numerical experiments on a range of Matrix Market problems from the Harwell-Boeing collection [12], the NEP collection [2], the SPARSKIT collection [29], and Tim Davis' collection [11]. The iterative solvers used in the experiments are Bi-CG and Bi-CR, and we evaluate the two methods with respect to the number of iterations (Its), computational time (Time), and $\log_{10}$ of the true relative residual 2-norm (TRR) defined as $\log_{10} \|b - Ax_n\|_2/\|b\|_2$. All experiments were performed on an ALPHA work station with a 750 MHz processor using double precision arithmetic. Codes were written in Fortran 77 and compiled with the optimization option *-O4*. In all cases the iteration was started with $x_0 = 0$ and $r_0^* = r_0$ in both methods, the right-hand side b was chosen as a vector with random entries from -1 to 1, and the stopping criterion was $\|r_n\|_2/\|b\|_2 \leq 10^{-12}$. The convergence histories show the number of iterations (on the horizontal axis) versus $\log_{10}$ of the relative residual 2-norm, $\log_{10} \|r_n\|_2/\|b\|_2$ (on the vertical axis).

Matrices in the experiments come from electronic circuit design (ADD20, ADD32, MEMPLUS), electrical engineering (BFW782A), finite element modeling (CAVITY05, CAVITY10, FIDAP036), fluid dynamics (CDDE1, E20R0000, E30R0000), oil reservoir simulation (ORSIRR1, ORSIRR2, ORSREG1, SHERMAN1, SHERMAN5), partial differential equations (PDE2961), aeroelasticity (TOLS4000), and petroleum engineering (WATT1, WATT2).

4.1. Comparison of Bi-CG and Bi-CR

In this subsection, we evaluate the performance of Bi-CG and Bi-CR with no preconditioning. The numerical results are shown in Table 2.

Table 2
Matrices, their sizes (N), and numerical results of Bi-CG and Bi-CR without preconditioning

Matrix	N	Its		Time (s)		TRR	
		Bi-CG	Bi-CR	Bi-CG	Bi-CR	Bi-CG	Bi-CR
ADD20	2 395	665	644	0.90	0.88	−11.98	−11.97
ADD32	4 960	130	129	0.39	0.39	−12.08	−12.12
BFW782A	782	435	390	0.18	0.17	−11.50	−11.52
CAVITY05	1 182	801	764	1.09	1.05	−12.04	−12.04
CAVITY10	2 597	1236	1155	3.70	3.53	−12.50	−12.03
CDDE1	961	173	173	0.09	0.09	−12.07	−12.05
E20R0000	4 241	1667	1597	9.01	8.74	−11.68	−11.69
E30R0000	9 661	2565	2526	36.83	36.74	−10.21	−10.25
FIDAP036	3 079	7263	6410	18.52	16.97	−11.48	−11.41
MEMPLUS	17 758	1913	1849	21.74	21.57	−11.73	−11.75
ORSIRR1	1 030	1646	1599	1.19	1.18	−11.87	−11.89
ORSIRR2	886	1192	1191	0.75	0.77	−12.11	−11.97
ORSREG1	2 205	630	582	1.12	1.05	−12.00	−11.95
PDE2961	2 961	335	319	0.60	0.58	−12.00	−12.55
SHERMAN1	1 000	709	704	0.31	0.31	−12.01	−12.01
SHERMAN5	3 312	2668	2664	3.83	3.88	−11.73	−11.02
TOLS4000	4 000	6248	5577	5.72	5.19	−9.00	−10.10
WATT1	1 856	544	492	0.80	0.74	−12.05	−12.01
WATT2	1 856	1489	689	2.14	1.01	−6.78	−7.74

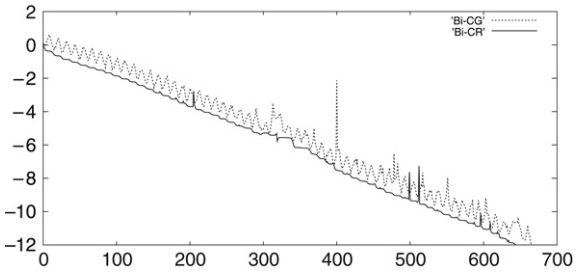

Fig. 1. Residual 2-norm histories of Bi-CG and Bi-CR for ADD20.

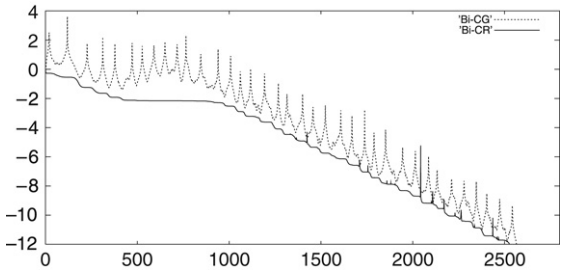

Fig. 2. Residual 2-norm histories of Bi-CG and Bi-CR for E30R0000.

Looking at Its and Time, Bi-CR required only about 90% of the iteration steps and computational time of Bi-CG in BFW782A, FIDAP036, TOLS4000, and WATT1. Notably in WATT2, Bi-CR performed much better than Bi-CG in that Bi-CR required only about 46% of the iteration steps and computational time of Bi-CG. In other problems, Bi-CG and Bi-CR required almost the same number of iteration steps and computational time.

In terms of TRR, accuracy of both approximate solutions was worse than the stopping criterion in E30R0000, TOLS4000, and WATT2; however, Bi-CR generated better approximate solutions than Bi-CG in TOLS4000 and in WATT2.

Residual 2-norm histories of Bi-CG and Bi-CR for ADD20, E30R0000, and WATT2 are shown in Figs. 1–3 respectively. We see from Figs. 1 and 2 that Bi-CG gives many peaks in the residual norm, whereas Bi-CR gives much smoother convergence behavior. In Fig. 3, Bi-CR gave much smoother convergence behavior and converged much faster than Bi-CG.

4.2. Comparison of Bi-CG and Bi-CR with ILU(0) preconditioning

In this subsection, we evaluate the performance of Bi-CG and Bi-CR with ILU(0) preconditioning [26]. The numerical results are shown in Table 3, where the result for TOLS4000 is not listed since ILU(0) caused breakdown.

Table 3
Matrices, their sizes (N), and numerical results of Bi-CG and Bi-CR with ILU(0) preconditioning

Matrix	N	Its		Time (s)		TRR	
		Bi-CG	Bi-CR	Bi-CG	Bi-CR	Bi-CG	Bi-CR
ADD20	2 395	274	274	0.72	0.73	−12.10	−12.14
ADD32	4 960	63	63	0.30	0.31	−12.05	−12.17
BFW782A	782	128	118	0.14	0.13	−11.23	−11.26
CAVITY05	1 182	168	168	0.48	0.49	−11.30	−11.36
CAVITY10	2 597	275	238	1.92	1.72	−12.29	−11.95
CDDE1	961	55	54	0.04	0.04	−12.14	−12.12
E20R0000	4 241	185	164	2.47	2.22	−12.16	−12.07
E30R0000	9 661	303	266	11.10	9.90	−11.03	−11.06
FIDAP036	3 079	177	177	1.07	1.09	−12.51	−12.00
MEMPLUS	17 758	492	488	12.07	12.18	−11.90	−11.89
ORSIRR1	1 030	76	72	0.10	0.10	−12.23	−11.99
ORSIRR2	886	77	73	0.09	0.09	−12.17	−12.05
ORSREG1	2 205	95	93	0.29	0.29	−12.03	−12.01
PDE2961	2 961	75	77	0.23	0.24	−12.00	−13.06
SHERMAN1	1 000	63	62	0.04	0.04	−12.43	−12.10
SHERMAN5	3 312	43	43	0.12	0.12	−12.17	−12.26
WATT1	1 856	60	59	0.14	0.14	−12.25	−12.06
WATT2	1 856	112	110	0.28	0.28	−8.09	−8.14

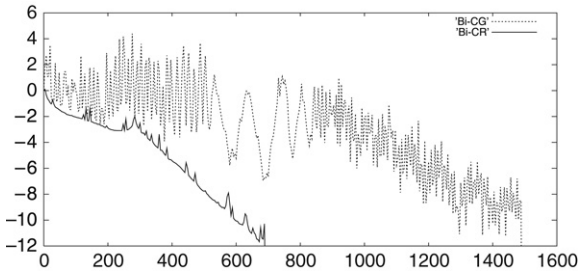

Fig. 3. Residual 2-norm histories of Bi-CG and Bi-CR for WATT2.

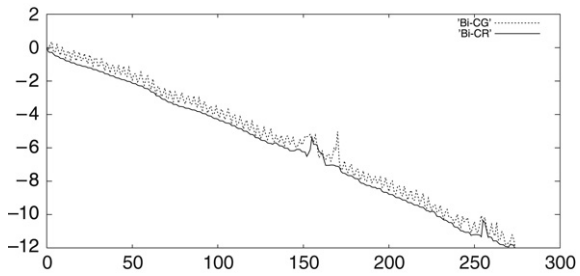

Fig. 4. Residual 2-norm histories of Bi-CG and Bi-CR with ILU(0) for ADD20.

With respect to Its and Time, Bi-CR required about 90% of the iteration steps and computational time of Bi-CG in CAVITY10, E20R0000, and E30R0000.

There was little difference in the performance of Bi-CG and Bi-CR in other problems except TOLS4000, since the preconditioner was quite effective in improving the convergence behavior. In terms of TRR, the accuracy of both approximate solutions was worse than the stopping criterion in E30R0000 and WATT2. In other problems, the two methods generated almost the same accuracy of the approximate solutions as the value of the given stopping criterion.

Residual 2-norm histories of Bi-CG and Bi-CR with ILU(0) preconditioning for ADD20, E30R0000, and WATT2 are shown in Figs. 4–6 respectively. As seen in Fig. 4, the convergence behavior with Bi-CG was jagged, whereas the behavior with Bi-CR was smoother. These histories were similar to those without preconditioning in Fig. 1. We see from Fig. 5 that Bi-CR showed fairly attractive convergence behavior in the last phase; however the smoothness as seen in Fig. 2 was not observed. In Fig. 6, the two methods showed similar convergence behavior.

5. Conclusions and future work

Based on H. A. van der Vorst's derivation of Bi-CG, in this paper we extended CR to nonsymmetric linear systems without the loss of its short-term recurrences. Since the resulting algorithm, named Bi-CR, reduces to CR if the coefficient matrix is

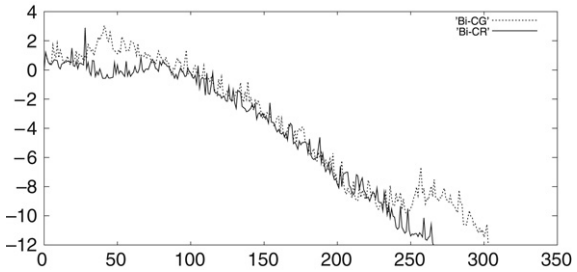

Fig. 5. Residual 2-norm histories of Bi-CG and Bi-CR with ILU(0) for E30R0000.

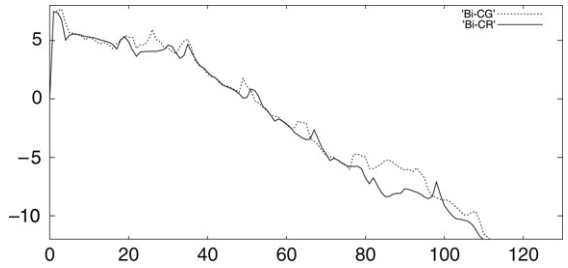

Fig. 6. Residual 2-norm histories of Bi-CG and Bi-CR with ILU(0) for WATT2.

symmetric, it can be regarded as an extension of CR. In addition, we have described the algorithm of Bi-CR for non-Hermitian linear systems and have mentioned the relation among CR, COCR, and Bi-CR.

From the numerical experiments we have learned that Bi-CR tends to show smoother convergence behavior and often converges faster than Bi-CG. Since Bi-CG is a basic solver for Bi-CGSTAB(ℓ) and GPBi-CG, it may take the place of Bi-CG for the attractive variants. In future work, we plan to give variants of Bi-CR, i.e., $r_n := H_n(A)r_n^{\mathrm{BCR}}$, where H_n is a suitable matrix polynomial of degree n, which is similar to the definition of Bi-CGSTAB(ℓ) or GPBi-CG.

Finally, we note that the preconditioned Bi-CR method requires the solutions of the types of the linear systems $Ky = z$ at each iteration step, where K is a preconditioning matrix, and thus it can be combined with many successful preconditioning strategies such as incomplete factorization preconditioners: e.g., ILU(p) [26], ILUT(p, τ) [30]; approximate inverse preconditioners: e.g., SPAI [19], AINV [8], FSAI [24]; structured preconditioners: e.g., MBGS-type, MBUGS-type [3]. Splitting iteration methods, see a state-of-the-art overview [4], that is, HSS [5], NSS [7], and PSS [6] can be also combined with the Bi-CR method as a preconditioner by applying splitting iteration methods to the form $A\tilde{y} = z$ since the solution of $Ky = z$ for the preconditioned Bi-CR method can be regarded as an approximate solution of $A\tilde{y} = z$.

Acknowledgment

The authors are grateful to Professor Martin H. Gutknecht for helpful comments and fruitful discussions.

References

[1] W.E. Arnoldi, The principle of minimized iteration in the solution of the matrix eigenvalue problem, Quart. Appl. Math. 9 (1951) 17–29.
[2] Z. Bai, D. Day, J. Demmel, J. Dongarra, A test matrix collection for non-Hermitian eigenvalue problems, Tech. Rep. CS-97-355, Department of Computer Science, University of Tennessee, Knoxville, TN, March, 1997.
[3] Z.-Z. Bai, Structured preconditioners for nonsingular matrices of block two-by-two structures, Math. Comp. 75 (2006) 791–815.
[4] Z.-Z. Bai, Splitting iteration methods for non-Hermitian positive definite systems of linear equations, Hokkaido Math. J. 36 (2007) 801–814.
[5] 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.
[6] Z.-Z. Bai, G.H. Golub, L.-Z. Lu, J.-F. Yin, Block triangular and skew-Hermitian splitting methods for positive-definite linear systems, SIAM J. Sci. Comput. 26 (2005) 844–863.
[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.
[8] M. Benzi, M. Tůma, A sparse approximate inverse preconditioner for nonsymmetric linear systems, SIAM J. Sci. Comput. 19 (1998) 968–994.
[9] C.G. Broyden, M.A. Boschetti, A comparison of three basic conjugate direction methods, Numer. Linear Algebra Appl. 3 (1996) 473–489.
[10] T.F. Chan, E. Gallopoulos, V. Simoncini, T. Szeto, C.H. Tong, A quasi-minimal residual variant of the Bi-CGSTAB algorithm for nonsymmetric systems, SIAM J. Sci. Comput. 15 (1994) 338–347.
[11] T. Davis, UF Sparse matrix collection. http://www.cise.ufl.edu/research/sparse/matrices.
[12] I.S. Duff, R.G. Grimes, J.G. Lewis, User's guide for the Harwell-Boeing sparse matrix collection, Tech. Rep. RAL-92-086, Rutherford Appleton Laboratory, Chilton, UK, 1992.
[13] S.C. Eisenstat, H.C. Elman, M.H. Schultz, Variational iterative methods for nonsymmeric systems of linear equations, SIAM J. Sci. Numer Anal. 20 (1983) 345–537.

[14] R. Fletcher, Conjugate gradient methods for indefinite systems, in: Lecture Notes in Math., vol. 506, Springer-Verlag, Berlin, Heidelberg, New York, 1976, pp. 73–89.
[15] R.W. Freund, A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems, SIAM J. Sci. Comput. 14 (1993) 470–482.
[16] R.W. Freund, N.M. Nachtigal, QMR: A quasi-minimal residual method for non-Hermitian linear systems, Numer. Math. 60 (1991) 315–339.
[17] R.W. Freund, T. Szeto, A quasi-minimal residual squared algorithm for non-Hermitian linear systems, in: Proceedings of the Second Copper Mountain Conference on Iterative Methods 1992.
[18] G.H. Golub, C.F. van Loan, Matrix Computations, 3rd ed., The Johns Hopkins University Press, Baltimore and London, 1996.
[19] M. Grote, T. Huckle, Parallel preconditioning with sparse approximate inverses, SIAM J. Sci. Comput. 18 (1997) 838–853.
[20] M.H. Gutknecht, Variants of BiCGSTAB for matrices with complex spectrum, SIAM J. Sci. Comput. 14 (1993) 1020–1033.
[21] M.H. Gutknecht, Lanczos-type solvers for nonsymmetric linear systems of equations, Acta Numer. 6 (1997) 271–397.
[22] M.R. Hestenes, E. Stiefel, Methods of conjugate gradients for solving linear systems, J. Res. Nat. Bur. Standards 49 (1952) 409–436.
[23] C.P. Jackson, P.C. Robinson, A numerical study of various algorithms related to the preconditioned conjugate gradient method, Internat. J. Numer. Methods Engrg. 21 (1985) 1315–1338.
[24] L.Yu. Kolotilina, A.Yu. Yeremin, Factorized sparse approximate inverse preconditionings I: Theory, SIAM J. Matrix Anal. Appl. 14 (1993) 45–58.
[25] C. Lanczos, Solution of systems of linear equations by minimized iterations, J. Res. Nat. Bur. Standards 49 (1952) 33–53.
[26] 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. Comp. 31 (1977) 148–162.
[27] C.C. Paige, M.A. Saunders, Solution of sparse indefinite systems of linear equations, SIAM J. Numer. Anal. 12 (1975) 617–629.
[28] Y. Saad, Krylov subspace methods for solving large unsymmetric linear systems, Math. Comp. 37 (1981) 105–126.
[29] Y. Saad, SPARSKIT: A basic tool kit for sparse matrix computation, Tech. Rep. CSRD TR 1029, CSRD, University of Illinois, Urbana, IL, 1990.
[30] Y. Saad, ILUT: a dual threshold incomplete *LU* factorization, Numer. Linear Algebra Appl. 4 (1994) 387–402.
[31] Y. Saad, Iterative Methods for Sparse Linear Systems, 2nd ed., SIAM, Philadelphia, PA, 2003.
[32] 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.
[33] G.L.G. Sleijpen, D.R. Fokkema, BICGSTAB(ℓ) for linear equations involving unsymmetric matrices with complex spectrum, Electron. Trans. Numer. Anal. 1 (1993) 11–32.
[34] T. Sogabe, S.-L. Zhang, A COCR method for solving complex symmetric linear systems, J. Comput. Appl. Math. 199 (2007) 297–303.
[35] P. Sonneveld, CGS, a fast Lanczos-type solver for nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 10 (1989) 36–52.
[36] C.H. Tong, A family of quasi-minimal residual methods for nonsymmetric linear systems, SIAM J. Sci. Comput. 15 (1994) 89–105.
[37] 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.
[38] H.A. van der Vorst, Iterative Krylov Methods for Large Linear Systems, Cambridge University Press, Cambridge, 2003.
[39] S.-L. Zhang, GPBi-CG: Generalized product-type methods based on Bi-CG for solving nonsymmetric linear systems, SIAM J. Sci. Comput. 18 (1997) 537–551.