

ML(k)BiCGSTAB: A BiCGSTAB VARIANT BASED ON MULTIPLE LANCZOS STARTING VECTORS*

MAN-CHUNG YEUNG[†] AND TONY F. CHAN[†]

Abstract. We present a variant of the popular BiCGSTAB method for solving nonsymmetric linear systems. The method, which we denote by ML(k)BiCGSTAB, is derived from a variant of the BiCG method based on a Lanczos process using multiple ($k > 1$) starting left Lanczos vectors. Compared with the original BiCGSTAB method, our new method produces a residual polynomial which is of lower degree after the same number of steps, but which also requires fewer matrix-vector products to generate, on average requiring only $1 + 1/k$ matvecs per step. Empirically, it also seems to be more stable and more quickly convergent. The new method can be implemented as a k -term recurrence and can be viewed as a bridge connecting the Arnoldi-based FOM/GMRES methods and the Lanczos-based BiCGSTAB methods.

Key words. BiCGSTAB, FOM, multiple Lanczos starting vectors, Krylov subspace, iterative methods, linear systems

AMS subject classifications. Primary, 65F10, 65F15; Secondary, 65F25, 65F30

PII. S1064827597321581

1. Introduction. BiCGSTAB [24] is a popular Krylov subspace method for the iterative solution of nonsymmetric linear systems. Its main features are that it is transpose-free, makes more efficient use of matrix-vector products when compared to BiCG [7, 12, 18], and is more stable than CGS [23]. In this paper, we introduce a new variant of BiCGSTAB which inherits all of these nice features. In addition, the key new ingredient of our method is the use of multiple starting left Lanczos vectors which has the desirable effects of lowering the cost per step and increasing the robustness.

BiCGSTAB is derived from BiCG which is a Lanczos-based Krylov subspace method. In BiCG, the residual vector r_l at the l th step lies in a Krylov subspace $K_{l+1}(r_0, A)$ and is chosen to be orthogonal to an auxiliary Krylov subspace $K_l(q_1, A^T)$. In our variant of the BiCG method, which we denote by ML(k)BiCG, r_l is still in $K_{l+1}(r_0, A)$; however, it is now chosen to be orthogonal to the union of k Krylov subspaces $K_{j+1}(q_s, A^T)$ and $K_j(q_{s'}, A^T)$, where $1 \leq s \leq i < s' \leq k$ and $jk + i = l$, generated from multiple ($k > 1$) linearly independent starting vectors q_u , $u = 1, 2, \dots, k$. Our motivation for using multiple left starting Lanczos vectors is to mitigate somewhat the ill conditioning of $K_l(q_1, A^T)$ for large l by replacing a high degree Krylov polynomial corresponding to one starting vector with a set of lower degree Krylov polynomials generated from different, independent starting vectors. We think this leads to better stability and robustness of the resulting iterative method. We derive an efficient implementation of this idea, requiring only memory of previous k iterates (i.e., a $k + 1$ -term recurrence).

We consider the major contribution of this paper, however, to be an extension of BiCGSTAB, which we denote by ML(k)BiCGSTAB, using multiple starting Lanczos vectors. The derivation is similar to, but rather more complicated than, that of

*Received by the editors May 16, 1997; accepted for publication (in revised form) March 10, 1998; published electronically December 15, 1999. The work of the authors was partially supported by the Army Research Office under contract DAAL-03-91-C-0047 (University of Tennessee subcontract ORA4466.04) and ONR under contract ONR-N00014-92-J-1890.

<http://www.siam.org/journals/sisc/21-4/32158.html>

[†]Department of Mathematics, University of California, Los Angeles, CA 90095-1555 (myeung@math.ucla.edu, chan@math.ucla.edu).

deriving BiCGSTAB from BiCG. ML(k)BiCGSTAB inherits from BiCGSTAB the advantages of being transpose-free and is more efficient in using matvec per step than ML(k)BiCG. Specifically, after $l = jk + i$ steps, where $i = 1, \dots, k$ and $j = 0, 1, \dots$, the residual vector \tilde{r}_l can be written as $\tilde{r}_l = \psi_{j+1}(A)\phi_l(A)r_0$, where ϕ_l is the degree l polynomial corresponding to the residual vector r_l of ML(k)BiCG, and ψ_{j+1} is a degree $j + 1$ smoothing polynomial. Thus, the “degree” of \tilde{r}_l is $l + j + 1$. To compute \tilde{r}_l , exactly $l + j + 1$ matvecs with A (and none with A^T) are required. Thus, the *cost per step on the average* is $1 + 1/k$ matvecs, but the *cost per degree* of the residual vector of ML(k)BiCGSTAB is the same as that of BiCGSTAB ($2l$ matvecs to obtain degree $2l$) and half as much as that of BiCG (l matvecs with A and l matvecs with A^T to obtain degree l). Both ML(k)BiCG and ML(k)BiCGSTAB can be implemented efficiently as k -term recurrences.

A way to view our method is through the conditioning of the collection of vectors that the residual vector is required to be orthogonal to, which we believe controls the stability of the method. For FOM/GMRES [15, 16, 18], these vectors are mutually orthogonal and thus perfectly conditioned. For BiCG, these vectors are the Lanczos vectors and they can be ill-conditioned. For our new ML(k)BiCG, these vectors consist of a union of k sets of Lanczos vectors, generated by initial vectors which are orthogonal, thus making them better conditioned than in the BiCG case. Our new ML(k)BiCGSTAB method, being a product method derived from ML(k)BiCG, inherits this increased stability. Thus, ML(k)BiCGSTAB can be viewed as an attempt to merge the advantages of FOM/GMRES (stability) and BiCGSTAB (short recurrence, transpose-free and efficient use of matvecs) while avoiding their respective disadvantages: long recurrence and imperfect stability [20].

Our ML(k)BiCGSTAB method has a close relationship to the *Lanczos-type method for multiple starting vectors*, proposed recently by Aliaga, Boley, Freund, and Hernández (ABFH) [1]. In fact, even though we have developed our methods independently of the ABFH framework, our BiCG extension ML(k)BiCG can be easily derived from the ABFH framework, using one right Lanczos vector and k left Lanczos vectors. Even so, ML(k)BiCG deserves some attention on its own right, because we believe it is the first attempt to use multiple left starting vector Lanczos-type methods for solving a *single* linear system.¹ The main application of nonequal left and right starting vectors cited in [1] is for computing transfer functions in multi-input/multioutput time invariant linear dynamical systems. A more significant difference between our present paper and [1] is that we have derived a BiCGSTAB variant based on multiple starting vectors, with the advantages stated above. We believe this is the first *product* Krylov method based on multiple starting vectors.

The origin of the ideas behind the new methods presented here can be traced to Ruhe’s vectorwise implementation of the block Lanczos method [14], in the same way that [1] can be considered an extension of Ruhe’s method to non-Hermitian matrices and nonequal left and right starting vectors plus look-ahead. As such, we believe our methods inherit the advantage of faster convergence of the underlying block Lanczos method.

Gutknecht [10] and Sleijpen and Fokkema [19] generalized BiCGSTAB to versions called BiCGSTAB2 and BiCGSTAB(k), respectively, in which they replaced the GMRES(1) part in BiCGSTAB with GMRES(k). The purpose of doing so was

¹Freund and Malhotra [8] considered a QMR-type method based on the ideas in [1] for linear systems with multiple right-hand-sides, but there the number of right and left Lanczos vectors are the same.

to increase the robustness and speed of convergence of BiCGSTAB [20, 21]. The robustness of GMRES(k)/FOM(k) for BiCGSTAB and BiCGSTAB(k) is exploited in [20, 21, 22]. From our experimental results, ML(k)BiCGSTAB may be a good alternative to achieve this goal of exploiting the robustness of GMRES(k)/FOM(k).

The outline of the paper is as follows. In section 2, we give our version of a Lanczos-type method with one right starting vector and k left starting vectors. In section 3, a BiCG-like method is derived from the Lanczos method in section 2. Section 4 contains the main derivation for the ML(k)BiCGSTAB method. Numerical results are given in section 5.

2. A Lanczos method for k linearly independent left starting vectors.

Let A be an $n \times n$ real matrix and let $k + 1$ real vectors v_0 and q_1, q_2, \dots, q_k be given. We define

$$(1) \quad p_{jk+i} = (A^T)^j q_i$$

for $i = 1, 2, \dots, k; j = 0, 1, 2, \dots$, and suppose the matrices

$$W_l = \begin{bmatrix} p_1^T v_0 & p_1^T A v_0 & \cdots & p_1^T A^{l-1} v_0 \\ p_2^T v_0 & p_2^T A v_0 & \cdots & p_2^T A^{l-1} v_0 \\ \vdots & \vdots & \ddots & \vdots \\ p_l^T v_0 & p_l^T A v_0 & \cdots & p_l^T A^{l-1} v_0 \end{bmatrix}, \quad l = 1, 2, \dots, \nu$$

are nonsingular, where ν is the grade of v_0 with respect to A , i.e., the degree of the minimal polynomial of v_0 with respect to A .

We now consider a sequence $\{v_l\}_{l=0,1,\dots,\nu}$ of vectors from the Krylov space

$$K(v_0, A) = \text{span}\{v_0, Av_0, A^2 v_0, \dots\}$$

with the properties

$$(2) \quad v_l \in A^l v_0 + K_l(v_0, A) \equiv A^l v_0 + \text{span}\{v_0, Av_0, \dots, A^{l-1} v_0\} \subset K_{l+1}(v_0, A)$$

and

$$(3) \quad v_l \perp \text{span}\{p_1, p_2, \dots, p_l\}.$$

The existence and uniqueness of such a sequence is guaranteed by the nonsingularity of the W_l 's. In fact, if we express v_l as

$$(4) \quad v_l = A^l v_0 + \gamma_0^{(l)} v_0 + \gamma_1^{(l)} A v_0 + \cdots + \gamma_{l-1}^{(l)} A^{l-1} v_0,$$

then (3) is equivalent to $W_l \gamma^{(l)} = -f$, where $\gamma^{(l)} = [\gamma_0^{(l)}, \gamma_1^{(l)}, \dots, \gamma_{l-1}^{(l)}]^T$ and $f = [p_1^T A^l v_0, \dots, p_l^T A^l v_0]^T$.

Two simple facts can be derived for the sequence $\{v_l\}_{l=0,1,\dots,\nu}$: (a) $v_\nu = 0$ and (b) $v_l \not\perp p_{l+1}$ whenever $l < \nu$. To see (a), we note that ν is the grade of v_0 and hence $A^\nu v_0 \in K_\nu(v_0, A)$. Property (2) is now reduced to $v_\nu \in K_\nu(v_0, A)$ and thus (a) follows by the uniqueness of v_ν . To prove (b), we assume that $v_l \perp p_{l+1}$. Then, combining with property (3), we have $v_l \perp \text{span}\{p_1, \dots, p_{l+1}\}$, and hence $W_{l+1} \tilde{\gamma}^{(l)} = 0$, where $\tilde{\gamma}^{(l)} = [\gamma_0^{(l)}, \dots, \gamma_{l-1}^{(l)}, 1]^T$ and $\gamma_i^{(l)}$ are defined in (4), in contradiction with the nonsingularity of W_{l+1} .

Applying (4) to itself recursively, we can represent v_l in terms of its previous v_0, \dots, v_{l-1} as follows:

$$v_l = Av_{l-1} + h_{l-1}^{(l-1)}v_{l-1} + h_{l-2}^{(l-1)}v_{l-2} + \dots + h_0^{(l-1)}v_0.$$

Noting that $v_i \perp \text{span}\{p_1, \dots, p_i\}$, $v_i \not\perp p_{i+1}$ and $A^T p_i = p_{k+i}$, and examining in turn

$$p_i^T v_l = p_i^T Av_{l-1} + h_{l-1}^{(l-1)}p_i^T v_{l-1} + h_{l-2}^{(l-1)}p_i^T v_{l-2} + \dots + h_0^{(l-1)}p_i^T v_0$$

for $i = 1, 2, \dots, l-k-1$, we find all the coefficients zero except $h_{l-1}^{(l-1)}, h_{l-2}^{(l-1)}, \dots, h_{m_l}^{(l-1)}$, where $m_l = \max(l-k-1, 0)$. Thus we obtain a $k+2$ term recursion relationship for $\{v_l\}_{l=1, \dots, \nu}$ of the form

$$(5) \quad v_l = Av_{l-1} + h_{l-1}^{(l-1)}v_{l-1} + h_{l-2}^{(l-1)}v_{l-2} + \dots + h_{m_l}^{(l-1)}v_{m_l}.$$

If we set $V_\nu = [v_0, v_1, \dots, v_{\nu-1}]$ and $H_\nu = (h_{ij})_{i,j=1, \dots, \nu}$, the $\nu \times \nu$ Hessenberg matrix with $h_{j+1,j} = 1; h_{ij} = -h_{i-1}^{(j-1)}$ for $m_j + 1 \leq i \leq j; h_{ij} = 0$ otherwise, and if we recall that $v_\nu = 0$, then the recurrence relations (5) can be written in matrix form as

$$(6) \quad AV_\nu = V_\nu H_\nu.$$

Moreover, set $P_\nu = [p_1, p_2, \dots, p_\nu]$ and apply P_ν^T to (6) from the left,

$$(7) \quad P_\nu^T AV_\nu = P_\nu^T V_\nu H_\nu.$$

Because of (3) and the fact that $v_l \not\perp p_{l+1}$, $P_\nu^T V_\nu$ is a lower triangular matrix with all the entries nonzero in its diagonal. Comparing the corresponding principal blocks of both sides of (7), we obtain a condition which guarantees the nonsingularity of H_ν : if the matrices

$$S_l = \begin{bmatrix} p_1^T Av_0 & p_1^T A^2 v_0 & \dots & p_1^T A^l v_0 \\ p_2^T Av_0 & p_2^T A^2 v_0 & \dots & p_2^T A^l v_0 \\ \vdots & \vdots & & \vdots \\ p_l^T Av_0 & p_l^T A^2 v_0 & \dots & p_l^T A^l v_0 \end{bmatrix}, \quad l = 1, 2, \dots, \nu$$

are nonsingular, so are the principal blocks of H_ν .

Finally, we can see from (2) that

$$(8) \quad K_{l+1}(v_0, A) = \text{span}\{v_0, v_1, \dots, v_l\}, \quad l = 0, 1, \dots, \nu - 1.$$

Since the dimension of $K_\nu(v_0, A)$ is ν , the vectors $\{v_l\}_{l=0,1, \dots, \nu-1}$ are linearly independent.

Summing up the above discussions, we conclude the following.

THEOREM 2.1. *Let be given vectors $v_0, q_1, q_2, \dots, q_k, p_1, p_2, \dots, p_\nu$, and matrices W_ν and S_ν as defined above. If the principal submatrices of W_ν are nonsingular, there exist an $n \times \nu$ matrix $V_\nu = [v_0, v_1, \dots, v_{\nu-1}]$ of rank ν , whose first column is v_0 , and a $\nu \times \nu$ Hessenberg matrix H_ν , which has upper bandwidth k and all the entries $h_{j+1,j} = 1$ in its lower subdiagonal, such that*

$$v_l \perp \text{span}\{p_1, p_2, \dots, p_l\}, \quad v_l \not\perp p_{l+1}, \quad l = 0, 1, \dots, \nu - 1,$$

and

$$(9) \quad AV_\nu = V_\nu H_\nu,$$

such that V_ν and H_ν are unique. Furthermore, if the principal submatrices of S_ν are nonsingular, so are the principal submatrices of H_ν .

Our procedure for generating the vectors v_i 's and p_i 's are closely related to the multiple starting vector Lanczos method in [1]. In fact, the columns of V_ν are exactly the same as the right Lanczos vectors in [1]. However, the p_i 's are different from the left Lanczos vectors in [1]; in fact, they are not orthogonal to the v_i 's. It turns out we do not need the full bi-orthogonality property in deriving our extensions of BiCG and BiCGSTAB.

3. ML(k)BiCG: A BiCG variant based on multiple starting vector Lanczos. We now turn to the linear system²

$$(10) \quad Ax = b$$

and we shall derive, based on Theorem 2.1, an oblique projection method by borrowing the techniques used in the derivation of the biconjugate gradient algorithm from the nonsymmetric Lanczos procedure [12, p. 211]. Even though this is a well-known procedure, in our case there is some difference from the standard case and therefore we include the derivation here for completeness and clarity.

Suppose an initial guess x_0 to (10) is given. We set in Theorem 2.1 $v_0 = b - Ax_0$. At the l th step of our projection method, we seek an approximate solution x_l with

$$(11) \quad x_l \in x_0 + span\{v_0, v_1, \dots, v_{l-1}\}$$

and

$$(12) \quad r_l \equiv b - Ax_l \perp span\{p_1, p_2, \dots, p_l\},$$

where the v_i 's and p_i 's are as defined in Theorem 2.1.

Since $P_l^T V_l$, a lower triangular matrix with all diagonal entries nonzero, is nonsingular, it follows easily that x_l is uniquely determined by conditions (11) and (12) and has the following expression:

$$(13) \quad x_l = x_0 + V_l H_l^{-1} e_1.$$

The corresponding residual r_l has the same direction as v_l . Recall that $P_l \equiv [p_1, \dots, p_l]$, $V_l \equiv [v_0, \dots, v_{l-1}]$, H_l is the $l \times l$ principal submatrix of H_ν , and e_1 is the first column of the $l \times l$ identity matrix. From (11) and (8), we have $r_l \in v_0 - span\{Av_0, A^2v_0, \dots, A^l v_0\}$ and hence $r_l \neq 0$ if $l < \nu$ since ν is the grade of v_0 . Moreover, $r_\nu = 0$ from (13) and (9).

Letting $r_i = \xi_i v_i$ for some scalar ξ_i which is not zero whenever $i < \nu$ and setting $\Lambda_l = \text{diag}\{\xi_0, \xi_1, \dots, \xi_{l-1}\}$, (13) can be rewritten as

$$x_l = x_0 + R_l \Lambda_l^{-1} H_l^{-1} e_1,$$

²Throughout the paper we do not assume the matrix A is nonsingular except where specified. We just require that the assumptions in Theorem 2.1 hold. In practice, A is usually large and sparse in most problems [5].

where $R_l \equiv [r_0, r_1, \dots, r_{l-1}]$. Write the LDU decomposition of $H_l \Lambda_l$, which exists and is unique due to the nonsingularities of the principal submatrices of $H_l \Lambda_l$, as

$$H_l \Lambda_l = L_l D_l U_l,$$

and define

$$G_l \equiv [g_0, g_1, \dots, g_{l-1}] = R_l U_l^{-1}, \quad z_l = D_l^{-1} L_l^{-1} e_1.$$

Because of the lower triangular structure of $L_l D_l$, we have

$$z_l = \begin{bmatrix} z_{l-1} \\ \alpha_l \end{bmatrix}$$

for some α_l . As a result, x_l can be updated as

$$(14) \quad x_l = x_0 + G_l z_l = x_0 + G_{l-1} z_{l-1} + \alpha_l g_{l-1} = x_{l-1} + \alpha_l g_{l-1}$$

and hence

$$(15) \quad r_l = r_{l-1} - \alpha_l A g_{l-1}.$$

On the other hand, since $G_{l+1} = R_{l+1} U_{l+1}^{-1}$, g_l can be computed from the previous g_i 's and r_l by the update

$$(16) \quad g_l = r_l + \beta_{l-1}^{(l)} g_{l-1} + \beta_{l-2}^{(l)} g_{l-2} + \dots + \beta_{\tilde{m}_l}^{(l)} g_{\tilde{m}_l},$$

where $\tilde{m}_l = \max(l - k, 0)$ and where $-\beta_i^{(l)}$, $i = \tilde{m}_l, \dots, l - 1$ and $-\beta_l^{(l)} = 1$ are the nonzero entries of the last column of U_{l+1} .

To compute the coefficients α_l and $\beta_i^{(l)}$ in (14), (15), and (16), we need the A -orthogonality of the vectors g_i and p_i and the orthogonality of the vectors p_l and r_l . Since

$$\begin{aligned} P_l^T A G_l &= P_l^T A R_l U_l^{-1} = P_l^T A V_l \Lambda_l U_l^{-1} = P_l^T (V_l H_l + v_l e_l^T) \Lambda_l U_l^{-1} \\ &= P_l^T V_l H_l \Lambda_l U_l^{-1} = P_l^T V_l L_l D_l U_l U_l^{-1} = P_l^T V_l L_l D_l, \end{aligned}$$

where e_l denotes the last column of the $l \times l$ identity matrix, and since $P_l^T V_l$ is nonsingular and lower triangular, we have

$$(17) \quad p_i^T A g_j = 0, \quad i \leq j,$$

and

$$p_i^T A g_{i-1} \neq 0.$$

Thus, utilizing this information, we examine the following equations derived from (15):

$$p_l^T r_l = p_l^T r_{l-1} - \alpha_l p_l^T A g_{l-1}$$

and

$$p_i^T A g_l = p_i^T A r_l + \beta_{l-1}^{(l)} p_i^T A g_{l-1} + \beta_{l-2}^{(l)} p_i^T A g_{l-2} + \dots + \beta_{\tilde{m}_l}^{(l)} p_i^T A g_{\tilde{m}_l}$$

for i going from $\tilde{m}_l + 1$ to l . Then we get

$$(18) \quad \alpha_l = \frac{p_l^T r_{l-1}}{p_l^T A g_{l-1}}$$

and

$$(19) \quad \beta_{i-1}^{(l)} = -\frac{p_i^T A r_l + p_i^T \sum_{j=\tilde{m}_l}^{i-2} \beta_j^{(l)} A g_j}{p_i^T A g_{i-1}}, \quad i = \tilde{m}_l + 1, \dots, l.$$

Now, putting the relations (14), (15), (16), (18), and (19) together, we have the following algorithm.

ALGORITHM 1. ML(k)BiCG.

1. Choose an initial guess x_0 and k vectors q_1, q_2, \dots, q_k .
2. Compute $r_0 = b - Ax_0$ and set $p_1 = q_1, g_0 = r_0$.
3. For $l = 1, 2, \dots$, until convergence:
4. $\alpha_l = p_l^T r_{l-1} / p_l^T A g_{l-1}$;
5. $x_l = x_{l-1} + \alpha_l g_{l-1}$;
6. $r_l = r_{l-1} - \alpha_l A g_{l-1}$;
7. For $s = \max(l - k, 0), \dots, l - 1$
8. $\beta_s^{(l)} = -p_{s+1}^T A \left(r_l + \sum_{t=\max(l-k, 0)}^{s-1} \beta_t^{(l)} g_t \right) / p_{s+1}^T A g_s$;
9. End
10. $g_l = r_l + \sum_{s=\max(l-k, 0)}^{l-1} \beta_s^{(l)} g_s$;
11. Compute p_{l+1} according to (1)
12. End

It is worthwhile to remark on two special cases where $k = 1$ and $k \geq \nu$. If $k = 1$, then $p_l = (A^T)^{l-1} q_1$ and conditions (11) and (12) become

$$x_l \in x_0 + \text{span}\{v_0, v_1, \dots, v_{l-1}\}, \quad r_l \perp K_l(q_1, A^T),$$

which are exactly what the BiCG approximate solution x_l^{BiCG} needs to satisfy. As a result, Algorithm 1 is equivalent to the BiCG algorithm mathematically. On the other hand, when $k \geq \nu$, (11) and (12) reduce to

$$x_l \in x_0 + \text{span}\{v_0, v_1, \dots, v_{l-1}\}, \quad r_l \perp \text{span}\{q_1, q_2, \dots, q_l\}$$

for $1 \leq l \leq \nu$. If, at the l th step of the computations, we choose $(p_{l+1} =) q_{l+1} = r_l$ while setting $(p_1 =) q_1 = r_0$ beforehand, then Algorithm 1 is mathematically equivalent to the FOM algorithm.

From its derivation, we can state the following result about Algorithm 1.

THEOREM 3.1. Under the assumptions of Theorem 2.1, ML(k)BiCG does not break down by zero division before step ν , and the approximate solution x_ν at step ν is exact to the system (10).

4. ML(k)BiCGSTAB: A BiCGSTAB variant based on multiple starting vector Lanczos. The implementation of Algorithm 1 requires the use of A^T to compute p_{l+1} in line 11. In practice, however, the transpose of A is not always available; for instance, if the matrix is not formed explicitly and the matvec product is given only as an operator. But this difficulty can be overcome by adopting the

techniques in the derivations of CGS and BiCGSTAB. In this section, we give a transpose-free version of Algorithm 1 which we call ML(k)BiCGSTAB.³

We first rearrange the outer *for* loop of Algorithm 1 into a form more convenient for our development. Let $l = jk + i$ and let the index i vary from 1 to k and j starting with 0. Then we convert the loop in l into doubly nested loops in j and i , respectively. By moving the case where $i = 1$ outside the i -loop, we rewrite the l -loop (omitting lines 5 and 11) of Algorithm 1 as

1. For $j = 0, 1, 2, \dots$
2. $\alpha_{jk+1} = p_{jk+1}^T r_{(j-1)k+k} / p_{jk+1}^T Ag_{(j-1)k+k};$
3. $r_{jk+1} = r_{(j-1)k+k} - \alpha_{jk+1} Ag_{(j-1)k+k};$
4. For $i = 1, 2, \dots, k$
5. For $s = \max((j-1)k + i, 0), \dots, jk + i - 1$
6. $\beta_s^{(jk+i)} = -p_{s+1}^T A \left(r_{jk+i} + \sum_{t=\max((j-1)k+i, 0)}^{s-1} \beta_t^{(jk+i)} g_t \right) / p_{s+1}^T Ag_s;$
7. End
8. $g_{jk+i} = r_{jk+i} + \sum_{s=\max((j-1)k+i, 0)}^{jk+i-1} \beta_s^{(jk+i)} g_s;$
9. If $i < k$
10. $\alpha_{jk+i+1} = p_{jk+i+1}^T r_{jk+i} / p_{jk+i+1}^T Ag_{jk+i};$
11. $r_{jk+i+1} = r_{jk+i} - \alpha_{jk+i+1} Ag_{jk+i};$
12. End
13. End
14. End

in which Lines 5–8 can be again expanded into

1. For $s = \max((j-1)k + i, 0), \dots, (j-1)k + k - 1$
2. $\beta_s^{(jk+i)} = -p_{s+1}^T A \left(r_{jk+i} + \sum_{t=\max((j-1)k+i, 0)}^{s-1} \beta_t^{(jk+i)} g_t \right) / p_{s+1}^T Ag_s;$
3. End
4. $\beta_{(j-1)k+k}^{(jk+i)} = -p_{jk+1}^T A \left(r_{jk+i} + \sum_{t=\max((j-1)k+i, 0)}^{(j-1)k+k-1} \beta_t^{(jk+i)} g_t \right) / p_{jk+1}^T Ag_{(j-1)k+k};$
5. For $s = jk + 1, \dots, jk + i - 1$
6. $\beta_s^{(jk+i)} = -p_{s+1}^T A \left(r_{jk+i} + \sum_{t=\max((j-1)k+i, 0)}^{(j-1)k+k} \beta_t^{(jk+i)} g_t \right. \\ \left. + \sum_{t=jk+1}^{s-1} \beta_t^{(jk+i)} g_t \right) / p_{s+1}^T Ag_s;$
7. End
8. $g_{jk+i} = r_{jk+i} + \sum_{s=\max((j-1)k+i, 0)}^{(j-1)k+k} \beta_s^{(jk+i)} g_s + \sum_{s=jk+1}^{jk+i-1} \beta_s^{(jk+i)} g_s;$

and further into

1. If $j = 0$
2. $\beta_0^{(i)} = -p_1^T Ar_i / p_1^T Ag_0;$
3. For $s = 1, \dots, i - 1$
4. $\beta_s^{(i)} = -p_{s+1}^T A \left(r_i + \beta_0^{(i)} g_0 + \sum_{t=1}^{s-1} \beta_t^{(i)} g_t \right) / p_{s+1}^T Ag_s;$
5. End
6. $g_i = r_i + \sum_{s=0}^{i-1} \beta_s^{(i)} g_s;$
7. Else
8. For $s = i, \dots, k - 1$
9. $\beta_{(j-1)k+s}^{(jk+i)} = -p_{(j-1)k+s+1}^T A \left(r_{jk+i} \right.$

³The derivation of ML(k)BiCGSTAB here may be simplified by adapting the approach to Lanczos-type product methods used in [11], which includes the usage of the w - and \hat{w} -tables.

- $+ \sum_{t=i}^{s-1} \beta_{(j-1)k+t}^{(jk+i)} g_{(j-1)k+t} \Big) / p_{(j-1)k+s+1}^T Ag_{(j-1)k+s};$
- 10. End
- 11. $\beta_{(j-1)k+k}^{(jk+i)} = -p_{jk+1}^T A \left(r_{jk+i} + \sum_{t=i}^{k-1} \beta_{(j-1)k+t}^{(jk+i)} g_{(j-1)k+t} \right) / p_{jk+1}^T Ag_{(j-1)k+k};$
- 12. For $s = 1, \dots, i - 1$
- 13. $\beta_{jk+s}^{(jk+i)} = -p_{jk+s+1}^T A \left(r_{jk+i} + \sum_{t=i}^k \beta_{(j-1)k+t}^{(jk+i)} g_{(j-1)k+t} \right. \\ \left. + \sum_{t=1}^{s-1} \beta_{jk+t}^{(jk+i)} g_{jk+t} \right) / p_{jk+s+1}^T Ag_{jk+s};$
- 14. End
- 15. $g_{jk+i} = r_{jk+i} + \sum_{s=i}^k \beta_{(j-1)k+s}^{(jk+i)} g_{(j-1)k+s} + \sum_{s=1}^{i-1} \beta_{jk+s}^{(jk+i)} g_{jk+s};$
- 16. End

Thus, the l -loop of Algorithm 1 is equivalent to the following triple nested loops.

- 1. For $j = 0, 1, 2, \dots$
- 2. $\alpha_{jk+1} = p_{jk+1}^T r_{(j-1)k+k} / p_{jk+1}^T Ag_{(j-1)k+k};$
- 3. $r_{jk+1} = r_{(j-1)k+k} - \alpha_{jk+1} Ag_{(j-1)k+k};$
- 4. For $i = 1, 2, \dots, k$
- 5. If $j = 0$
- 6. $\beta_0^{(i)} = -p_1^T Ar_i / p_1^T Ag_0;$
- 7. For $s = 1, \dots, i - 1$
- 8. $\beta_s^{(i)} = -p_{s+1}^T A \left(r_i + \beta_0^{(i)} g_0 + \sum_{t=1}^{s-1} \beta_t^{(i)} g_t \right) / p_{s+1}^T Ag_s;$
- 9. End
- 10. $g_i = r_i + \sum_{s=0}^{i-1} \beta_s^{(i)} g_s;$
- 11. Else
- 12. For $s = i, \dots, k - 1$
- 13. $\beta_{(j-1)k+s}^{(jk+i)} = -p_{(j-1)k+s+1}^T Ar_{jk+i} \\ + \sum_{t=i}^{s-1} \beta_{(j-1)k+t}^{(jk+i)} g_{(j-1)k+t} / p_{(j-1)k+s+1}^T Ag_{(j-1)k+s};$
- 14. End
- 15. $\beta_{(j-1)k+k}^{(jk+i)} = -p_{jk+1}^T A \left(r_{jk+i} \right. \\ \left. + \sum_{t=i}^{k-1} \beta_{(j-1)k+t}^{(jk+i)} g_{(j-1)k+t} \right) / p_{jk+1}^T Ag_{(j-1)k+k};$
- 16. For $s = 1, \dots, i - 1$
- 17. $\beta_{jk+s}^{(jk+i)} = -p_{jk+s+1}^T A \left(r_{jk+i} + \sum_{t=i}^k \beta_{(j-1)k+t}^{(jk+i)} g_{(j-1)k+t} \right. \\ \left. + \sum_{t=1}^{s-1} \beta_{jk+t}^{(jk+i)} g_{jk+t} \right) / p_{jk+s+1}^T Ag_{jk+s};$
- 18. End
- 19. $g_{jk+i} = r_{jk+i} + \sum_{s=i}^k \beta_{(j-1)k+s}^{(jk+i)} g_{(j-1)k+s} + \sum_{s=1}^{i-1} \beta_{jk+s}^{(jk+i)} g_{jk+s};$
- 20. End
- 21. If $i < k$
- 22. $\alpha_{jk+i+1} = p_{jk+i+1}^T r_{jk+i} / p_{jk+i+1}^T Ag_{jk+i};$
- 23. $r_{jk+i+1} = r_{jk+i} - \alpha_{jk+i+1} Ag_{jk+i};$
- 24. End
- 25. End
- 26. End

We now introduce an auxiliary polynomial $\psi_j(\lambda)$ defined by the recurrence relation,

$$\psi_0(\lambda) = 1,$$

$$\psi_j(\lambda) = (\rho_j \lambda + 1) \psi_{j-1}(\lambda), \quad j = 1, 2, \dots,$$

where ρ_j is a free parameter. This polynomial $\psi_j(\lambda)$ was first used by van der Vorst in the derivation of BiCGSTAB [24]. If we express $\psi_j(\lambda)$ in terms of the power basis,

$$\psi_j(\lambda) = \eta_j^{(j)} \lambda^j + \dots + \eta_1^{(j)} \lambda + \eta_0^{(j)},$$

then it is clear that $\eta_j^{(j)} = \rho_1 \rho_2 \dots \rho_j$ and $\eta_0^{(j)} = 1$.

Next, we define the following vectors, analogous to those defined in BiCGSTAB,

$$\begin{aligned} \tilde{\pi}_{jk+i} &= \psi_j(A)r_{jk+i}, & \pi_{jk+i} &= \psi_{j+1}(A)r_{jk+i}, \\ \tilde{\omega}_{jk+i} &= \psi_j(A)g_{jk+i}, & \omega_{jk+i} &= \psi_{j+1}(A)g_{jk+i}, \end{aligned}$$

for $i = 1, 2, \dots, k; j = 0, 1, \dots$, and set $\pi_0 = \omega_0 = r_0 (= g_0)$. Our goal is to define π_{jk+i} to be the residual of our new method. The other three vectors are needed in deriving a recurrence for π_{jk+i} . By recalling (1), (12), and (17), we find that the scalars α_l 's and β_l 's in Algorithm 1 can be computed via these new vectors. The derivations are quite complicated and therefore we give the details in the Appendix while summarizing only the results here. In fact, we have

$$\alpha_{jk+1} = \frac{q_1^T \pi_{(j-1)k+k}}{q_1^T A \omega_{(j-1)k+k}}$$

for $0 \leq j$;

$$\alpha_{jk+i+1} = \frac{\rho_{j+1} q_{i+1}^T \tilde{\pi}_{jk+i}}{q_{i+1}^T (\omega_{jk+i} - \tilde{\omega}_{jk+i})}$$

for $0 \leq j, 1 \leq i < k$;

$$\beta_0^{(i)} = -\frac{q_1^T \pi_i}{\rho_1 q_1^T A \omega_0}$$

for $1 \leq i \leq k$;

$$\beta_s^{(i)} = -\frac{q_{s+1}^T \left(\pi_i + \beta_0^{(i)} \rho_1 A \omega_0 + \sum_{t=1}^{s-1} \beta_t^{(i)} (\omega_t - \tilde{\omega}_t) \right)}{q_{s+1}^T (\omega_s - \tilde{\omega}_s)}$$

for $1 \leq s < i \leq k$;

$$\beta_{(j-1)k+s}^{(jk+i)} = -\frac{q_{s+1}^T \left(\tilde{\pi}_{jk+i} + \sum_{t=i}^{s-1} \beta_{(j-1)k+t}^{(jk+i)} (\omega_{(j-1)k+t} - \tilde{\omega}_{(j-1)k+t}) \right)}{q_{s+1}^T (\omega_{(j-1)k+s} - \tilde{\omega}_{(j-1)k+s})}$$

for $1 \leq j, 1 \leq i \leq s \leq k-1$;

$$\beta_{(j-1)k+k}^{(jk+i)} = -\frac{q_1^T \left(\pi_{jk+i} + \rho_{j+1} \sum_{t=i}^{k-1} \beta_{(j-1)k+t}^{(jk+i)} A \omega_{(j-1)k+t} \right)}{\rho_{j+1} q_1^T A \omega_{(j-1)k+k}}$$

for $1 \leq j, 1 \leq i \leq k$;

$$\beta_{jk+s}^{(jk+i)} = -\frac{q_{s+1}^T \left(\pi_{jk+i} + \rho_{j+1} \sum_{t=i}^k \beta_{(j-1)k+t}^{(jk+i)} A \omega_{(j-1)k+t} + \sum_{t=1}^{s-1} \beta_{jk+t}^{(jk+i)} (\omega_{jk+t} - \tilde{\omega}_{jk+t}) \right)}{q_{s+1}^T (\omega_{jk+s} - \tilde{\omega}_{jk+s})}$$

for $1 \leq j, 1 \leq s < i \leq k$.

Moreover, the vectors $\tilde{\pi}_{jk+i}$, π_{jk+i} , $\tilde{\omega}_{jk+i}$ and ω_{jk+i} themselves can be updated as follows:

$$\tilde{\pi}_{jk+1} = \pi_{(j-1)k+k} - \alpha_{jk+1}A\omega_{(j-1)k+k}$$

for $0 \leq j$;

$$\pi_{jk+1} = \rho_{j+1}A\tilde{\pi}_{jk+1} + \tilde{\pi}_{jk+1}$$

for $0 \leq j$;

$$\tilde{\pi}_{jk+i+1} = \tilde{\pi}_{jk+i} - \frac{\alpha_{jk+i+1}}{\rho_{j+1}}(\omega_{jk+i} - \tilde{\omega}_{jk+i})$$

for $1 \leq i < k, 0 \leq j$;

$$\pi_{jk+i+1} = \pi_{jk+i} - \alpha_{jk+i+1}A\omega_{jk+i}$$

for $1 \leq i < k, 0 \leq j$;

$$\tilde{\omega}_i = \tilde{\pi}_i + \beta_0^{(i)}\omega_0 + \sum_{s=1}^{i-1} \beta_s^{(i)}\tilde{\omega}_s$$

for $1 \leq i \leq k$;

$$\omega_i = \pi_i + \beta_0^{(i)}(\rho_1A\omega_0 + \omega_0) + \sum_{s=1}^{i-1} \beta_s^{(i)}\omega_s$$

for $1 \leq i \leq k$;

$$\tilde{\omega}_{jk+i} = \tilde{\pi}_{jk+i} + \sum_{s=i}^k \beta_{(j-1)k+s}^{(jk+i)}\omega_{(j-1)k+s} + \sum_{s=1}^{i-1} \beta_{jk+s}^{(jk+i)}\tilde{\omega}_{jk+s}$$

for $1 \leq j, 1 \leq i \leq k$;

$$\omega_{jk+i} = \pi_{jk+i} + \sum_{s=i}^k \beta_{(j-1)k+s}^{(jk+i)}(\rho_{j+1}A\omega_{(j-1)k+s} + \omega_{(j-1)k+s}) + \sum_{s=1}^{i-1} \beta_{jk+s}^{(jk+i)}\omega_{jk+s}$$

for $1 \leq j, 1 \leq i \leq k$. The details of the derivation of these equations can be found in the appendix.

The formulas just derived constitute the main operations of the ML(k)BiCGSTAB algorithm, which we summarize as follows:

1. Set $\pi_0 = \omega_0 = r_0$.
2. For $j = 0, 1, 2, \dots$
3. $\alpha_{jk+1} = \frac{q_1^T \pi_{(j-1)k+k}}{q_1^T A\omega_{(j-1)k+k}}$;
4. $\tilde{\pi}_{jk+1} = \pi_{(j-1)k+k} - \alpha_{jk+1}A\omega_{(j-1)k+k}$;
5. Choose $\rho_{j+1} \neq 0$;
6. $\pi_{jk+1} = \rho_{j+1}A\tilde{\pi}_{jk+1} + \tilde{\pi}_{jk+1}$;
7. For $i = 1, 2, \dots, k$

8. If $j = 0$
9. $\beta_0^{(i)} = -\frac{q_1^T \pi_i}{\rho_1 q_1^T A \omega_0};$
10. For $s = 1, \dots, i-1$
11. $\beta_s^{(i)} = -\frac{q_{s+1}^T \left(\pi_i + \beta_0^{(i)} \rho_1 A \omega_0 + \sum_{t=1}^{s-1} \beta_t^{(i)} (\omega_t - \tilde{\omega}_t) \right)}{q_{s+1}^T (\omega_s - \tilde{\omega}_s)};$
12. End
13. $\tilde{\omega}_i = \tilde{\pi}_i + \beta_0^{(i)} \omega_0 + \sum_{s=1}^{i-1} \beta_s^{(i)} \tilde{\omega}_s;$
14. $\omega_i = \pi_i + \beta_0^{(i)} (\rho_1 A \omega_0 + \omega_0) + \sum_{s=1}^{i-1} \beta_s^{(i)} \omega_s;$
15. Else
16. For $s = i, \dots, k-1$
17. $\beta_{(j-1)k+s}^{(jk+i)} = -\frac{q_{s+1}^T \left(\tilde{\pi}_{jk+i} + \sum_{t=i}^{s-1} \beta_{(j-1)k+t}^{(jk+i)} (\omega_{(j-1)k+t} - \tilde{\omega}_{(j-1)k+t}) \right)}{q_{s+1}^T (\omega_{(j-1)k+s} - \tilde{\omega}_{(j-1)k+s})};$
18. End
19. $\beta_{(j-1)k+k}^{(jk+i)} = -\frac{q_1^T \left(\pi_{jk+i} + \rho_{j+1} \sum_{t=i}^{k-1} \beta_{(j-1)k+t}^{(jk+i)} A \omega_{(j-1)k+t} \right)}{\rho_{j+1} q_1^T A \omega_{(j-1)k+k}};$
20. For $s = 1, \dots, i = 1$
21. $\beta_{jk+s}^{(jk+i)} = -q_{s+1}^T \left(\pi_{jk+i} + \rho_{j+1} \sum_{t=i}^k \beta_{(j-1)k+t}^{(jk+i)} A \omega_{(j-1)k+t} \right. \\ \left. + \sum_{t=1}^{s-1} \beta_{jk+t}^{(jk+i)} (\omega_{jk+t} - \tilde{\omega}_{jk+t}) \right) / q_{s+1}^T (\omega_{jk+s} - \tilde{\omega}_{jk+s});$
22. End
23. $\tilde{\omega}_{jk+i} = \tilde{\pi}_{jk+i} + \sum_{s=i}^k \beta_{(j-1)k+s}^{(jk+i)} \omega_{(j-1)k+s} + \sum_{s=1}^{i-1} \beta_{jk+s}^{(jk+i)} \tilde{\omega}_{jk+s};$
24. $\omega_{jk+i} = \pi_{jk+i} + \sum_{s=i}^k \beta_{(j-1)k+s}^{(jk+i)} (\rho_{j+1} A \omega_{(j-1)k+s} + \omega_{(j-1)k+s}) \\ + \sum_{s=1}^{i-1} \beta_{jk+s}^{(jk+i)} \omega_{jk+s};$
25. End
26. If $i < k$
27. $\alpha_{jk+i+1} = \frac{\rho_{j+1} q_{i+1}^T \tilde{\pi}_{jk+i}}{q_{i+1}^T (\omega_{jk+i} - \tilde{\omega}_{jk+i})};$
28. $\tilde{\pi}_{jk+i+1} = \tilde{\pi}_{jk+i} - \frac{\alpha_{jk+i+1}}{\rho_{j+1}} (\omega_{jk+i} - \tilde{\omega}_{jk+i});$
29. $\pi_{jk+i+1} = \pi_{jk+i} - \alpha_{jk+i+1} A \omega_{jk+i};$
30. End
31. End
32. End

Some simplifications can be made to these operations, for instance, (a) resetting the scalar α_{jk+i+1} in line 27 to be $q_{i+1}^T \tilde{\pi}_{jk+i} / q_{i+1}^T (\omega_{jk+i} - \tilde{\omega}_{jk+i})$ since α_{jk+i+1} is used only in lines 28 and 29 and the factor ρ_{j+1} will be cancelled in line 28; (b) merging lines 9–14 and lines 19–24 by adding a conditional control “if $j \geq 1$ ” in line 16 and treating $\beta_l^{(i)}$ with $l < 0$ as zero; such $\beta_l^{(i)}$ ’s will appear in lines 19–24 when $j = 0$; (c) introducing the auxiliary vector

$$d_{jk+i} \equiv \omega_{jk+i} - \tilde{\omega}_{jk+i},$$

which can be updated by using lines 13, 14, 23, and 24 as

$$d_{jk+i} = \pi_{jk+i} - \tilde{\pi}_{jk+i} + \rho_{j+1} \sum_{s=i}^k \beta_{(j-1)k+s}^{(jk+i)} A \omega_{(j-1)k+s} + \sum_{s=1}^{i-1} \beta_{jk+s}^{(jk+i)} d_{jk+s}.$$

With these changes, we rewrite lines 8–30 as

1. For $s = i, \dots, k - 1$ and $j \geq 1$
2.
$$\beta_{(j-1)k+s}^{(jk+i)} = -\frac{q_{s+1}^T \left(\tilde{\pi}_{jk+i} + \sum_{t=i}^{s-1} \beta_{(j-1)k+t}^{(jk+i)} d_{(j-1)k+t} \right)}{q_{s+1}^T d_{(j-1)k+s}};$$
3. End
4.
$$\beta_{(j-1)k+k}^{(jk+i)} = -\frac{q_1^T \left(\pi_{jk+i} + \rho_{j+1} \sum_{t=i}^{k-1} \beta_{(j-1)k+t}^{(jk+i)} A\omega_{(j-1)k+t} \right)}{\rho_{j+1} q_1^T A\omega_{(j-1)k+k}};$$
5. For $s = 1, \dots, i - 1$
6.
$$\beta_{jk+s}^{(jk+i)} = -\frac{q_{s+1}^T \left(\pi_{jk+i} + \rho_{j+1} \sum_{t=i}^k \beta_{(j-1)k+t}^{(jk+i)} A\omega_{(j-1)k+t} + \sum_{t=1}^{s-1} \beta_{jk+t}^{(jk+i)} d_{jk+t} \right)}{q_{s+1}^T d_{jk+s}};$$
7. End
8. $d_{jk+i} = \pi_{jk+i} - \tilde{\pi}_{jk+i} + \rho_{j+1} \sum_{s=i}^k \beta_{(j-1)k+s}^{(jk+i)} A\omega_{(j-1)k+s} + \sum_{s=1}^{i-1} \beta_{jk+s}^{(jk+i)} d_{jk+s};$
9. $\omega_{jk+i} = \pi_{jk+i} + \sum_{s=i}^k \beta_{(j-1)k+s}^{(jk+i)} (\rho_{j+1} A\omega_{(j-1)k+s} + \omega_{(j-1)k+s}) + \sum_{s=1}^{i-1} \beta_{jk+s}^{(jk+i)} \omega_{jk+s};$
10. If $i < k$
11.
$$\alpha_{jk+i+1} = \frac{q_{i+1}^T \tilde{\pi}_{jk+i}}{q_{i+1}^T d_{jk+i}};$$
12. $\tilde{\pi}_{jk+i+1} = \tilde{\pi}_{jk+i} - \alpha_{jk+i+1} d_{jk+i};$
13. $\pi_{jk+i+1} = \pi_{jk+i} - \rho_{j+1} \alpha_{jk+i+1} A\omega_{jk+i};$
14. End

in which lines 1–9 can be further rewritten as

1. $z_d = \tilde{\pi}_{jk+i}; z_\omega = \pi_{jk+i}; z_{A\omega} = 0;$
2. For $s = i, \dots, k - 1$ and $j \geq 1$
3.
$$\beta_{(j-1)k+s}^{(jk+i)} = -\frac{q_{s+1}^T z_d}{q_{s+1}^T d_{(j-1)k+s}};$$
4. $z_d = z_d + \beta_{(j-1)k+s}^{(jk+i)} d_{(j-1)k+s};$
5. $z_\omega = z_\omega + \beta_{(j-1)k+s}^{(jk+i)} \omega_{(j-1)k+s};$
6. $z_{A\omega} = z_{A\omega} + \beta_{(j-1)k+s}^{(jk+i)} A\omega_{(j-1)k+s};$
7. End
8.
$$\beta_{(j-1)k+k}^{(jk+i)} = -\frac{q_1^T (\pi_{jk+i} + \rho_{j+1} z_{A\omega})}{\rho_{j+1} q_1^T A\omega_{(j-1)k+k}};$$
9. $z_\omega = z_\omega + \beta_{(j-1)k+k}^{(jk+i)} \omega_{(j-1)k+k};$
10. $z_{A\omega} = \rho_{j+1} \left(z_{A\omega} + \beta_{(j-1)k+k}^{(jk+i)} A\omega_{(j-1)k+k} \right);$
11. $z_d = \pi_{jk+i} + z_{A\omega};$
12. For $s = 1, \dots, i - 1$
13.
$$\beta_{jk+s}^{(jk+i)} = -\frac{q_{s+1}^T z_d}{q_{s+1}^T d_{jk+s}};$$
14. $z_d = z_d + \beta_{jk+s}^{(jk+i)} d_{jk+s};$
15. $z_\omega = z_\omega + \beta_{jk+s}^{(jk+i)} \omega_{jk+s};$
16. End
17. $d_{jk+i} = z_d - \tilde{\pi}_{jk+i};$
18. $\omega_{jk+i} = z_\omega + z_{A\omega};$

As the approximate solution x_l at step $l (= jk+i)$ of the ML(k)BiCGSTAB algorithm,

we define

$$(20) \quad x_{jk+1} = x_{(j-1)k+k} - \rho_{j+1}\tilde{\pi}_{jk+1} + \alpha_{jk+1}\omega_{(j-1)k+k}$$

for $j \geq 0$ and

$$(21) \quad x_{jk+i+1} = x_{jk+i} + \rho_{j+1}\alpha_{jk+i+1}\omega_{jk+i}$$

for $j \geq 0, k > i \geq 1$. One can readily verify by induction that π_l is the residual vector of x_l , that is, $\pi_l = b - Ax_l$.

We are now ready to state the ML(k)BiCGSTAB algorithm formally. As part of the algorithm, we choose the parameter ρ_{j+1} to minimize the 2-norm of the vector $\pi_{jk+1} = \rho_{j+1}A\tilde{\pi}_{jk+1} + \tilde{\pi}_{jk+1}$, i.e., $\rho_{j+1} = -\tilde{\pi}_{jk+1}^T A\tilde{\pi}_{jk+1} / \|A\tilde{\pi}_{jk+1}\|^2$. Noting that the data $q_1^T A\omega_{(j-1)k+k}$, $q_{s+1}^T d_{(j-1)k+s}$ and $q_{s+1}^T d_{jk+s}$ are repeatedly used inside each loop of the control variable j , to save the computational cost we introduce a variable $c_{j'k+i'}$, such that $c_{j'k+i'} = q_{i'+1}^T d_{j'k+i'}$ if $1 \leq i' \leq k-1$ and $c_{j'k+k} = q_1^T A\omega_{j'k+k}$ if $i' = k$. Moreover, since π_l is the residual of x_l , we relabel π_l as r_l . We also relabel ω_l and $\tilde{\pi}_l$ as g_l and u_l , respectively.

ALGORITHM 2. ML(k)BiCG.

1. Choose an initial guess x_0 and k vectors q_1, q_2, \dots, q_k .
2. Compute $r_0 = b - Ax_0$ and set $g_0 = r_0$.
3. For $j = 0, 1, 2, \dots$
4. $w_{(j-1)k+k} = Ag_{(j-1)k+k}$;
5. $c_{(j-1)k+k} = q_1^T w_{(j-1)k+k}$;
6. $\alpha_{jk+1} = q_1^T r_{(j-1)k+k} / c_{(j-1)k+k}$;
7. $u_{jk+1} = r_{(j-1)k+k} - \alpha_{jk+1}w_{(j-1)k+k}$;
8. $\rho_{j+1} = -u_{jk+1}^T Au_{jk+1} / \|Au_{jk+1}\|^2$;
9. $x_{jk+1} = x_{(j-1)k+k} - \rho_{j+1}u_{jk+1} + \alpha_{jk+1}g_{(j-1)k+k}$;
10. $r_{jk+1} = \rho_{j+1}Au_{jk+1} + u_{jk+1}$;
11. For $i = 1, 2, \dots, k$
12. $z_d = u_{jk+i}$; $z_g = r_{jk+i}$; $z_w = 0$;
13. For $s = i, \dots, k-1$ and $j \geq 1$
14. $\beta_{(j-1)k+s}^{(jk+i)} = -q_{s+1}^T z_d / c_{(j-1)k+s}$;
15. $z_d = z_d + \beta_{(j-1)k+s}^{(jk+i)} d_{(j-1)k+s}$;
16. $z_g = z_g + \beta_{(j-1)k+s}^{(jk+i)} g_{(j-1)k+s}$;
17. $z_w = z_w + \beta_{(j-1)k+s}^{(jk+i)} w_{(j-1)k+s}$;
18. End
19. $\beta_{(j-1)k+k}^{(jk+i)} = -\frac{q_1^T (r_{jk+i} + \rho_{j+1}z_w)}{\rho_{j+1}c_{(j-1)k+k}}$;
20. $z_g = z_g + \beta_{(j-1)k+k}^{(jk+i)} g_{(j-1)k+k}$;
21. $z_w = \rho_{j+1} (z_w + \beta_{(j-1)k+k}^{(jk+i)} w_{(j-1)k+k})$;
22. $z_d = r_{jk+i} + z_w$;
23. For $s = 1, \dots, i-1$
24. $\beta_{jk+s}^{(jk+i)} = -q_{s+1}^T z_d / c_{jk+s}$;
25. $z_d = z_d + \beta_{jk+s}^{(jk+i)} d_{jk+s}$;
26. $z_g = z_g + \beta_{jk+s}^{(jk+i)} g_{jk+s}$;
27. End
28. $d_{jk+i} = z_d - u_{jk+i}$;

- 29. $g_{jk+i} = z_g + z_w;$
- 30. $If\ i < k$
- 31. $c_{jk+i} = q_{i+1}^T d_{jk+i};$
- 32. $\alpha_{jk+i+1} = q_{i+1}^T u_{jk+i} / c_{jk+i};$
- 33. $u_{jk+i+1} = u_{jk+i} - \alpha_{jk+i+1} d_{jk+i};$
- 34. $x_{jk+i+1} = x_{jk+i} + \rho_{j+1} \alpha_{jk+i+1} g_{jk+i};$
- 35. $w_{jk+i} = A g_{jk+i};$
- 36. $r_{jk+i+1} = r_{jk+i} - \rho_{j+1} \alpha_{jk+i+1} w_{jk+i};$
- 37. *End*
- 38. *End*
- 39. *End*

The denominators in the ML(k)BiCGSTAB algorithm appear only in the evaluations of α 's and β 's and from the process of their derivations. These denominators differ from their counterparts in Algorithm 1 but cannot be zero if the denominators in Algorithm 1 are not zero, assuming the ρ_j 's defined in Algorithm 2 are nonzero. Hence, under the assumption that $\rho_{j+1} \neq 0$ for all j , we can see that if Algorithm 1 does not break down by zero division at some step, then neither does ML(k)BiCGSTAB at the same step. Moreover, since the residual vector r_{jk+i}^2 is by definition the vector $\psi_{j+1}(A)r_{jk+i}^1$, where r_{jk+i}^1 and r_{jk+i}^2 denote the corresponding residual vectors in Algorithms 1 and 2, respectively, and since $r_{jk+i}^1 \in K_{jk+i+1}(v_0, A)$, we have $r_{jk+i}^2 \in K_{jk+i+j+2}(v_0, A)$. Thus it is possible that r_{jk+i}^2 vanishes when $jk+i+j+1 \geq \nu$, or $jk+i \geq \nu-j-1$. On the other hand, r_ν^2 must be zero because $r_\nu^1 = 0$.

THEOREM 4.1. *Under the assumptions of Theorem 2.1 and if $\rho_{j+1} \neq 0$ for all j , the ML(k)BiCGSTAB algorithm does not break down by zero division before step ν and an exact solution⁴ to (10) is obtained at or before step ν .*

A similar remark to the one at the end of section 3 can also be made here. Mathematically, ML(1)BiCGSTAB is equivalent to BiCGSTAB since it was established based on BiCG by using exactly the same techniques used in deriving BiCGSTAB. In the case where $k \geq \nu$, we can obtain an exact solution in the first loop of j , i.e., $j = 0$, and the algorithm now can be regarded as a FOM algorithm (with the q_i 's appropriately chosen), for the reasons stated in the following. Since

$$r_i^1 \in v_0 - span\{Av_0, A^2v_0, \dots, A^i v_0\}$$

from section 3, we have

$$r_i^2 = \psi_1(A)r_i^1 \in r_0^F - span\{Ar_0^F, A^2r_0^F, \dots, A^i r_0^F\},$$

where $v_0 = b - Ax_0$, $r_0^F \equiv \psi_1(A)v_0$ and r_i^1 and r_i^2 are the residual vectors of Algorithm 1 and ML(k)BiCGSTAB, respectively, and $1 \leq i \leq \nu$. Thus if A is nonsingular and since $\psi_1(\lambda) = \rho_1\lambda + 1$,

$$x_i^2 \in x_0^F + span\{r_0^F, Ar_0^F, \dots, A^{i-1}r_0^F\},$$

where x_i^2 denotes the approximate solution of ML(k)BiCGSTAB, defined by (20) and (21), with residual r_i^2 and where $x_0^F = x_0 - \rho_1 v_0$ and $r_0^F = b - Ax_0^F$. If at step i of the ML(k)BiCGSTAB algorithm, we choose $q_1 (= A^0 q_1 = p_1) = \psi_1(A^T)\psi_1(A)v_0$ ⁵

⁴If the coefficient matrix A is singular, the system (10) may have more than one solution.

⁵Note that ρ_1 , the leading coefficient of $\psi_1(\lambda)$, is a function of q_1 according to Steps 4–8 of Algorithm 2, and here we suppose the equation $q_1 = \psi_1(A^T)\psi_1(A)v_0$ has solutions for q_1 .

and $q_{i'} (= A^0 q_{i'} = p_{i'}) = \psi_1(A^T) r_{i'-1}^2$ for $2 \leq i' \leq i$, then

$$(r_0^F)^T r_i^2 = (\psi_1(A^T) \psi_1(A) v_0)^T r_i^1 = p_1^T r_i^1 = 0, \quad 1 \leq i,$$

and

$$(r_{i'}^2)^T r_i^2 = (r_{i'}^2)^T \psi_1(A) r_i^1 = p_{i'+1}^T r_i^1 = 0, \quad 1 \leq i' \leq i-1$$

by (12). In other words,

$$r_i^2 \perp \text{span}\{r_0^F, r_1^2, \dots, r_{i-1}^2\}, \quad 1 \leq i \leq \nu.$$

As a result, the $\text{ML}(k)\text{BiCGSTAB}$ ($k \geq \nu$) algorithm with the special choices for q_i 's described above is mathematically equivalent to the FOM algorithm defined by

$$x_i^F \in x_0^F + \text{span}\{r_0^F, Ar_0^F, \dots, A^{i-1}r_0^F\}$$

and

$$r_i^F \perp \{r_0^F, r_1^F, \dots, r_{i-1}^F\},$$

where the initial guess x_0^F and residual r_0^F are defined as above.

It is quite straightforward to give a preconditioned version of $\text{ML}(k)\text{BiCGSTAB}$. Suppose we are solving the right-preconditioned system,

$$AM^{-1}y = b, \quad y = Mx.$$

Directly applying the $\text{ML}(k)\text{BiCGSTAB}$ algorithm to the system $AM^{-1}y = b$ for the y -variable and then recovering the x -approximation from the y -approximation with the relation $y_l = Mx_l$ yields the following algorithm.

ALGORITHM 3. $\text{ML}(k)\text{BiCGSTAB}$ WITH PRECONDITIONING.

1. Choose an initial guess x_0 and k vectors q_1, q_2, \dots, q_k .
2. Compute $r_0 = b - Ax_0$ and set $g_0 = r_0$.
3. For $j = 0, 1, 2, \dots$
4. $\tilde{g}_{(j-1)k+k} = M^{-1}g_{(j-1)k+k}$;
5. $w_{(j-1)k+k} = A\tilde{g}_{(j-1)k+k}$;
6. $c_{(j-1)k+k} = q_1^T w_{(j-1)k+k}$;
7. $\alpha_{jk+1} = q_1^T r_{(j-1)k+k} / c_{(j-1)k+k}$;
8. $u_{jk+1} = r_{(j-1)k+k} - \alpha_{jk+1} w_{(j-1)k+k}$;
9. $\tilde{u}_{jk+1} = M^{-1}u_{jk+1}$;
10. $\rho_{j+1} = -u_{jk+1}^T A\tilde{u}_{jk+1} / \|A\tilde{u}_{jk+1}\|^2$;
11. $x_{jk+1} = x_{(j-1)k+k} - \rho_{j+1}\tilde{u}_{jk+1} + \alpha_{jk+1}\tilde{g}_{(j-1)k+k}$;
12. $r_{jk+1} = \rho_{j+1}A\tilde{u}_{jk+1} + u_{jk+1}$;
13. For $i = 1, 2, \dots, k$
14. $z_d = u_{jk+i}$; $z_g = r_{jk+i}$; $z_w = 0$;
15. For $s = i, \dots, k-1$ and $j \geq 1$
16. $\beta_{(j-1)k+s}^{(jk+i)} = -q_{s+1}^T z_d / c_{(j-1)k+s}$;
17. $z_d = z_d + \beta_{(j-1)k+s}^{(jk+i)} d_{(j-1)k+s}$;
18. $z_g = z_g + \beta_{(j-1)k+s}^{(jk+i)} g_{(j-1)k+s}$;
19. $z_w = z_w + \beta_{(j-1)k+s}^{(jk+i)} w_{(j-1)k+s}$;
20. End

TABLE 1

Average cost per step of the preconditioned ML(k)BiCGSTAB and its storage requirement.

| Preconditioning ($M^{-1}v$) | $1 + 1/k$ | Vector addition | 3 |
|-------------------------------|-----------|------------------|-----------------------------|
| Matvec (Av) | $1 + 1/k$ | Saxpy | $2.5k + 3.5 + 1/k$ |
| dot product | $k + 2$ | Scalar operation | $k + 3 - 1/k$ |
| Scalar-vector | 1 | Storage | $A + M + 4kn + O(k) + O(n)$ |

21.
$$\beta_{(j-1)k+k}^{(jk+i)} = -\frac{q_1^T (r_{jk+i} + \rho_{j+1}z_w)}{\rho_{j+1}C_{(j-1)k+k}};$$
22.
$$z_g = z_g + \beta_{(j-1)k+k}^{(jk+i)}g_{(j-1)k+k};$$
23.
$$z_w = \rho_{j+1} \left(z_w + \beta_{(j-1)k+k}^{(jk+i)}w_{(j-1)k+k} \right);$$
24.
$$z_d = r_{jk+i} + z_w;$$
25. For $s = 1, \dots, i - 1$
26.
$$\beta_{jk+s}^{(jk+i)} = -q_{s+1}^T z_d / c_{jk+s};$$
27.
$$z_d = z_d + \beta_{jk+s}^{(jk+i)}d_{jk+s};$$
28.
$$z_g = z_g + \beta_{jk+s}^{(jk+i)}g_{jk+s};$$
29. End
30.
$$d_{jk+i} = z_d - u_{jk+i};$$
31.
$$g_{jk+i} = z_g + z_w;$$
32. If $i < k$
33.
$$c_{jk+i} = q_{i+1}^T d_{jk+i};$$
34.
$$\alpha_{jk+i+1} = q_{i+1}^T u_{jk+i} / c_{jk+i};$$
35.
$$u_{jk+i+1} = u_{jk+i} - \alpha_{jk+i+1}d_{jk+i};$$
36.
$$\tilde{g}_{jk+i} = M^{-1}g_{jk+i};$$
37.
$$x_{jk+i+1} = x_{jk+i} + \rho_{j+1}\alpha_{jk+i+1}\tilde{g}_{jk+i};$$
38.
$$w_{jk+i} = A\tilde{g}_{jk+i};$$
39.
$$r_{jk+i+1} = r_{jk+i} - \rho_{j+1}\alpha_{jk+i+1}w_{jk+i};$$
40. End
41. End
42. End

With suitable changes of variables, it may be shown that both the left and split preconditioning versions of ML(k)BiCGSTAB also lead to Algorithm 3 provided that q_1, q_2, \dots, q_k are appropriately chosen. For the concepts of left, right, and split preconditioning, one is referred to [18].

Each loop of the control variable j in Algorithm 3 involves solving $k + 1$ systems with coefficient matrix M , $k + 1$ matrix-vector multiplications with A , $k^2 + 2k$ dot products, $2.5k^2 + 3.5k + 1$ saxpy's, $3k$ vector additions, k scalar-vector multiplications and $k^2 + 3k - 1$ scalar operations. Since there are k steps in each loop of j , the average cost per step can be calculated and is listed in Table 1. Regarding the storage, the data $\{q_1, \dots, q_k\}$, $\{d_{(j-1)k+i}, \dots, d_{jk+i-1}\}$, $\{g_{(j-1)k+i}, \dots, g_{jk+i-1}\}$ and $\{w_{(j-1)k+i}, \dots, w_{jk+i-1}\}$ are used in the process at step $jk+i$ and hence they must be stored. Since they dominate the memory when k is large, the storage of the algorithm is about $4kn$. We note that when $k = 1$ the cost is the same as BiCGSTAB's and for large k , the cost tends to that of FOM.

5. Numerical Experiments. In this section, we shall illustrate the numerical convergence behavior of ML(k)BiCGSTAB. We shall compare ML(k)BiCGSTAB to

TABLE 2

Comparison of methods on a representative group of matrices from the Harwell–Boeing collection. $ML(25)$, $ML(50)$, and $ML(100)$ stand for $ML(25)BiCGSTAB$, $ML(50)BiCGSTAB$, and $ML(100)BiCGSTAB$, respectively. The vector \tilde{r}_0 in the $BiCG$ and $BiCGSTAB$ codes was set to be r_0 . The numbers in the table are number of matvecs. “–” means no convergence within 20n matvecs for $BiCG$ and 10n for the other methods, “b” denotes breakdown, “o” denotes overflow.

| Matrix | Order | $BiCG$ | $BiCGSTAB$ | $GMRES(100)$ | $ML(25)$ | $ML(50)$ | $ML(100)$ |
|----------|-------|--------|------------|--------------|----------|----------|-----------|
| 1138bus | 1138 | 4748 | 5872 | – | 1966 | 1384 | 1395 |
| bcsprw06 | 1454 | 5810 | 14246 | – | 3167 | 2720 | 1899 |
| bcsstk08 | 1074 | 15844 | – | – | 3859 | 1902 | 1242 |
| bcsstk14 | 1806 | 29294 | – | – | – | 13315 | 6336 |
| bcsstk19 | 817 | – | – | – | – | – | – |
| bcsstm27 | 1224 | – | – | – | – | – | – |
| can1054 | 1054 | 9908 | – | – | 4058 | 3126 | 2606 |
| dwt1005 | 1005 | 1178 | 2934 | – | 673 | 625 | 645 |
| eris1176 | 1176 | 1426 | 1530 | 1197 | 698 | 532 | 499 |
| fs5414 | 541 | 2738 | 2640 | – | 728 | 469 | 403 |
| gr3030 | 900 | 76 | 52 | 38 | 40 | 40 | 40 |
| gre1107 | 1107 | – | b | – | – | 8676 | 3262 |
| hor131 | 434 | – | – | – | 1945 | 1268 | 1048 |
| impcold | 425 | – | b | – | 1619 | 916 | 597 |
| jagmesh2 | 1009 | 1726 | 2958 | – | 1152 | 995 | 1129 |
| jpwh991 | 991 | 100 | 58 | 49 | 55 | 53 | 55 |
| lns511 | 511 | – | – | – | – | – | – |
| lock1074 | 1068 | o | – | – | – | – | – |
| lshp1270 | 1270 | 2492 | 4458 | – | 1628 | 1591 | 1445 |
| mahindas | 1258 | – | b | – | – | – | – |
| mcfe | 765 | – | – | – | – | – | – |
| nnc1374 | 1374 | – | b | – | – | – | – |
| nos3 | 960 | 494 | 384 | 1968 | 251 | 249 | 246 |
| orsirr1 | 1030 | 2068 | 3318 | 1270 | 838 | 781 | 772 |
| plat1919 | 1919 | – | – | – | – | – | – |
| pores2 | 1224 | – | – | – | – | – | – |
| saylr3 | 1000 | o | – | – | o | o | o |
| sherman2 | 1080 | – | b | – | – | – | – |
| watt2 | 1856 | 19406 | – | 1131 | – | – | – |
| west0989 | 989 | – | – | – | – | – | – |

TABLE 3

A test run of $BiCG$ and $BiCGSTAB$ on a representative group of matrices from the Harwell–Boeing collection. The vector \tilde{r}_0 was set to be a random vector with iid entries from $N(0, 1)$. See Table 2 for the meaning of the notations.

| Matrix | Order | $BiCG$ | $BiCGSTAB$ | Matrix | Order | $BiCG$ | $BiCGSTAB$ |
|----------|-------|--------|------------|----------|-------|--------|------------|
| 1138bus | 1138 | 10504 | 8164 | jpwh991 | 991 | 108 | 62 |
| bcsprw06 | 1454 | – | 13258 | lns511 | 511 | – | – |
| bcsstk08 | 1074 | – | – | lock1074 | 1068 | o | – |
| bcsstk14 | 1806 | 35184 | – | lshp1270 | 1270 | – | 4662 |
| bcsstk19 | 817 | – | – | mahindas | 1258 | – | – |
| bcsstm27 | 1224 | – | – | mcfe | 765 | – | – |
| can1054 | 1054 | 10844 | – | nnc1374 | 1374 | – | – |
| dwt1005 | 1005 | – | 2744 | nos3 | 960 | 512 | 388 |
| eris1176 | 1176 | – | 1648 | orsirr1 | 1030 | 2214 | 3676 |
| fs5414 | 541 | 2668 | 4142 | plat1919 | 1919 | – | – |
| gr3030 | 900 | 82 | 55 | pores2 | 1224 | – | – |
| gre1107 | 1107 | – | – | saylr3 | 1000 | – | – |
| hor131 | 434 | – | – | sherman2 | 1080 | – | b |
| impcold | 425 | – | b | watt2 | 1856 | – | – |
| jagmesh2 | 1009 | 2894 | 3300 | west0989 | 989 | – | – |

BiCG, BiCGSTAB, and GMRES(m)⁶ [16] on a test suite of matrices from the Harwell-Boeing collection [4]. For the implementation of these latter three methods, we used the versions described in [2]. All the experiments were run in MATLAB 4.2c on a SUN SparcStation with machine precision about 10^{-16} . As for the initial guesses and right-hand sides, we always chose $x_0 = 0$ and $b = [1, 1, \dots, 1]^T$. For the initial vectors q_1, q_2, \dots, q_k in the ML(k)BiCGSTAB algorithm, we first chose k random vectors with independent and identically distributed (iid) entries from a normal distribution with mean 0 and variance 1 ($N(0, 1)$) and then made them orthogonal to each other by using the modified Gram-Schmidt algorithm [9]. The iteration was stopped as soon as the true relative error $\|b - Ax_l\|_2 / \|b\|_2$ was less than 10^{-7} . Finally, all the figures plot the true relative residual versus the number of matrix-vector multiplies taken.

We ran all four methods, on a representative group of matrices from the Harwell-Boeing collection. The results are summarized in Tables 2 and 3. In Table 2, we used $\tilde{r}_0 = r_0$ in the BiCG and BiCGSTAB codes and in Table 3, \tilde{r}_0 was a random vector with iid entries from $N(0, 1)$. We observe that, in terms of number of matvecs, ML(50)BiCGSTAB and ML(100)BiCGSTAB are always better than the other four methods, at least for this collection of matrices. The only exception is the matrix *watt2* where only BiCG and GMRES converged. We can also see that ML(k)BiCGSTAB for $k = 25$ is almost as good as for $k = 50$, whereas $k = 100$ does not give much improvement over $k = 50$ in most cases. We believe that the improvement of ML(k)BiCGSTAB over BiCGSTAB can be attributed to the use of multiple starting vectors. In principle, ML(k)BiCGSTAB can never be better than full GMRES, but as we can see from the table, it can be much better than *restarted* GMRES. We can also see from the table that ML(k)BiCGSTAB and BiCG tend to converge and diverge more or less on the same subset of matrices, but ML(k)BiCGSTAB typically requires many fewer matvecs when they all converge.

Next, we present the convergence history for three matrices from Table 2. These matrices are described below. We have used ML(30)BiCGSTAB in these examples.

Example 1. This example is the first matrix named IMPCOLD from the CHEMIMP group of the Harwell-Boeing collection. The order of the matrix is 425 and it has 1339 nonzero entries. In this example, no preconditioner was used and the convergence curves are plotted in Figure 1(a). BiCGSTAB encounter a breakdown after 450 matvecs.

Example 2. The matrix is the second one named ORSIRR1 from the OILGEN group. The order of the matrix is 1030 and the number of nonzero entries is 6858. We first run the algorithms without preconditioning and then with ILU(0) preconditioning. The results are shown in Figures 1(b) and 2(a), respectively.

Example 3. This is the HOR131 matrix from the NNCENG group. The order is 434 and it has 4710 nonzero entries. The ILU(0) preconditioner was used and the result is plotted in Figure 2(b).

We observe that when all four methods converge (as in Examples 2 and 3), ML(30)-BiCGSTAB requires approximately the same or fewer matvecs than the other three methods. In fact, it can be significantly faster than the other three methods, as in Example 2. Moreover, ML(30)BiCGSTAB manages to converge when the other three methods fail, as in Example 1.

⁶We note that we have compared with only the basic versions of BiCGSTAB and GMRES. There exist now many new variants of these methods which may perform better, e.g., BiCGSTAB2, BiCGSTAB(k), Deflated GMRES [6, 13], FGMRES [17], GMRESR [25], Mixed-BiCGSTAB-CGS [3], etc.

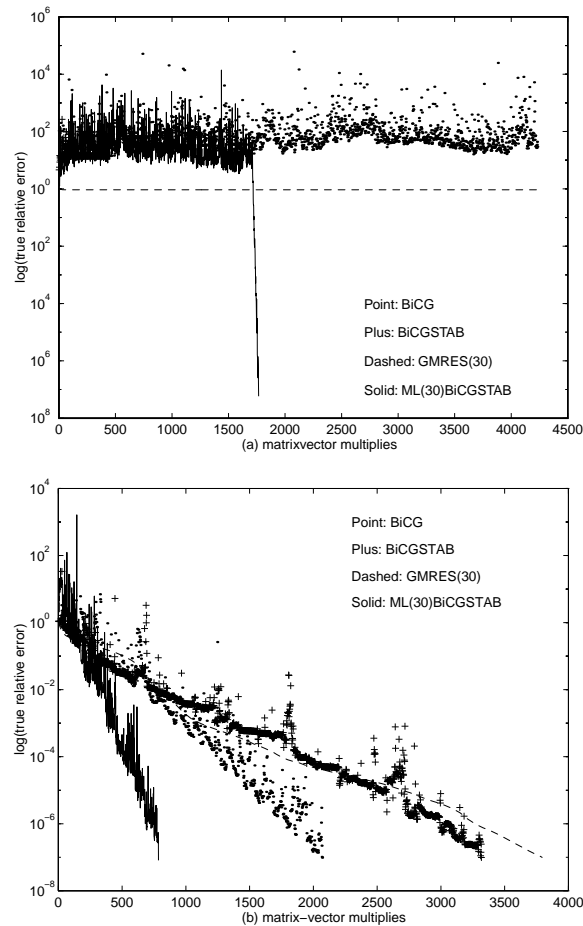


FIG. 1. (a) Example 1: with no preconditioning; (b) Example 2: with no preconditioning.

Finally, we present some numerical results to demonstrate the dependence of the performance of $ML(k)BiCGSTAB$ on the value of k . In Figure 3, we plot the number of matvecs (scaled by $1/10n$) versus k for the two matrices ORSIRR1 and HOR131. In order to illustrate the improvement of $ML(k)BiCGSTAB$ over $BiCGSTAB$ for $k > 1$, we plot for $k = 1$ the number of matvecs for $BiCGSTAB$ instead of for the mathematically equivalent $ML(1)BiCGSTAB$. We observe that for both matrices there is a dramatic improvement in performance as k increases from 1. This behavior is typical for the matrices that we have tested and this can be partially observed from Table 2. Thus the advantage of $ML(k)BiCGSTAB$ can be realized even for small values of k . On the other hand, we can also see that for large enough values of k (e.g., $k > 10$ for ORSIRR1 and $k > 30$ for HOR131), the performance is not sensitive to the value of k . Thus, it is not crucial to choose an optimal value of k as long as k is large enough. We have also found that the performance is not sensitive to the specific choice of the random starting vectors q_i 's, provided that k is large enough. However, we should caution that the performance could be sensitive to the choice of q_i 's for *small* values of k .

More testings are of course needed to better understand and assess the perfor-

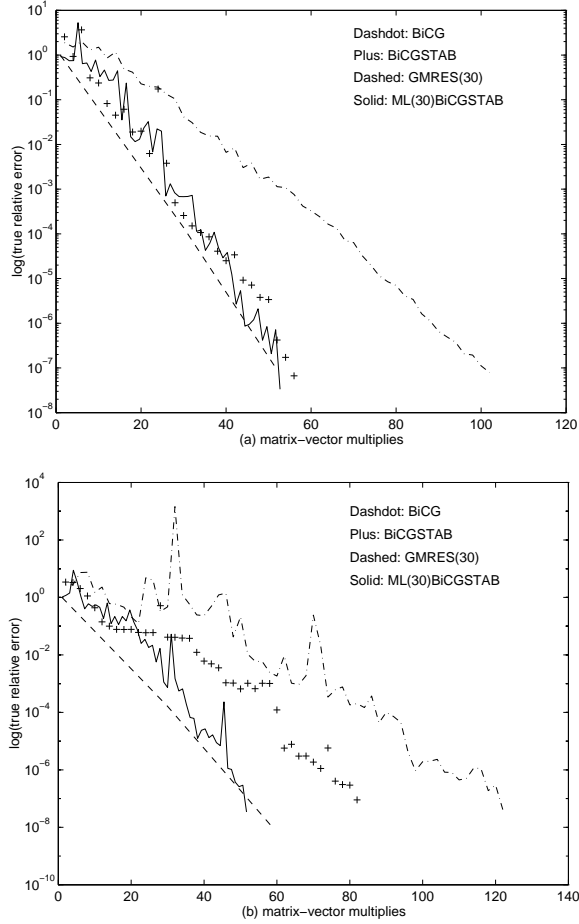


FIG. 2. (a) Example 2: with $ILU(0)$ preconditioning; (b) Example 3: with $ILU(0)$ preconditioning.

mance of $ML(k)BiCGSTAB$, but we hope we have at least demonstrated the potential advantages of this new method.

6. Appendix. Here we give the detailed derivation of the coefficients α_l and β_l of the $ML(k)BiCGSTAB$ algorithm:

$$\begin{aligned} \alpha_{j,k+1} &= \frac{\eta_j^{(j)} p_{jk+1}^T r_{(j-1)k+k}}{\eta_j^{(j)} p_{jk+1}^T A g_{(j-1)k+k}} = \frac{\sum_{s=0}^j \eta_s^{(j)} p_{sk+1}^T r_{(j-1)k+k}}{\sum_{s=0}^j \eta_s^{(j)} p_{sk+1}^T A g_{(j-1)k+k}} \\ &= \frac{\sum_{s=0}^j \eta_s^{(j)} q_1^T A^s r_{(j-1)k+k}}{\sum_{s=0}^j \eta_s^{(j)} q_1^T A^{s+1} g_{(j-1)k+k}} = \frac{q_1^T \psi_j(A) r_{(j-1)k+k}}{q_1^T A \psi_j(A) g_{(j-1)k+k}} \\ &= \frac{q_1^T \pi_{(j-1)k+k}}{q_1^T A \omega_{(j-1)k+k}} \end{aligned}$$

for $0 \leq j$;

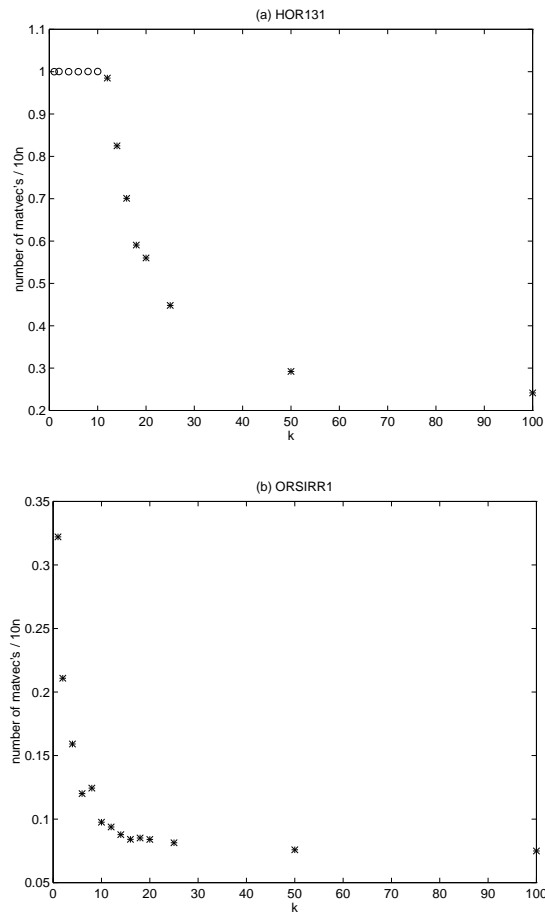


FIG. 3. Number of matvecs/10n vs. k for the matrices (a) HOR131 and (b) ORSIRR1. “o” denotes no convergence within 10n matvecs. For k = 1, we plotted the number of matvecs for BiCGSTAB. Note that there is a dramatic improvement in performance as k increases from 1, but that for k ≥ 30, the performance is not sensitive to k.

$$\begin{aligned}
 \alpha_{jk+i+1} &= \frac{\eta_j^{(j)} p_{jk+i+1}^T r_{jk+i}}{\eta_j^{(j)} p_{jk+i+1}^T A g_{jk+i}} = \frac{\sum_{s=0}^j \eta_s^{(j)} p_{sk+i+1}^T r_{jk+i}}{\sum_{s=0}^j \eta_s^{(j)} p_{sk+i+1}^T A g_{jk+i}} \\
 &= \frac{\sum_{s=0}^j \eta_s^{(j)} q_{i+1}^T A^s r_{jk+i}}{\sum_{s=0}^j \eta_s^{(j)} q_{i+1}^T A^{s+1} g_{jk+i}} = \frac{q_{i+1}^T \psi_j(A) r_{jk+i}}{q_{i+1}^T A \psi_j(A) g_{jk+i}} \\
 &= \frac{\rho_{j+1} q_{i+1}^T \psi_j(A) r_{jk+i}}{q_{i+1}^T (\psi_{j+1}(A) - \psi_j(A)) g_{jk+i}} = \frac{\rho_{j+1} q_{i+1}^T \tilde{\pi}_{jk+i}}{q_{i+1}^T (\omega_{jk+i} - \tilde{\omega}_{jk+i})}
 \end{aligned}$$

for $0 \leq j, 1 \leq i < k$;

$$\begin{aligned} \beta_0^{(i)} &= -\frac{p_1^T (\eta_1^{(1)} Ar_i + \eta_0^{(1)} r_i)}{\eta_1^{(1)} p_1^T Ag_0} = -\frac{q_1^T \psi_1(A) r_i}{\rho_1 q_1^T Ag_0} \\ &= -\frac{q_1^T \pi_i}{\rho_1 q_1^T A\omega_0} \end{aligned}$$

for $1 \leq i \leq k$;

$$\begin{aligned} \beta_s^{(i)} &= -\frac{p_{s+1}^T \left(\eta_1^{(1)} Ar_i + \beta_0^{(i)} \eta_1^{(1)} Ag_0 + \sum_{t=1}^{s-1} \beta_t^{(i)} \eta_1^{(1)} Ag_t \right)}{\eta_1^{(1)} p_{s+1}^T Ag_s} \\ &= -\frac{p_{s+1}^T \left(\psi_1(A) r_i + \beta_0^{(i)} \eta_1^{(1)} Ag_0 + \sum_{t=1}^{s-1} \beta_t^{(i)} (\psi_1(A) - \psi_0(A)) g_t \right)}{p_{s+1}^T (\psi_1(A) - \psi_0(A)) g_s} \\ &= -\frac{q_{s+1}^T \left(\pi_i + \beta_0^{(i)} \rho_1 A\omega_0 + \sum_{t=1}^{s-1} \beta_t^{(i)} (\omega_t - \tilde{\omega}_t) \right)}{q_{s+1}^T (\omega_s - \tilde{\omega}_s)} \end{aligned}$$

for $1 \leq s < i \leq k$;

$$\begin{aligned} \beta_{(j-1)k+s}^{(jk+i)} &= -\frac{\eta_j^{(j)} p_{(j-1)k+s+1}^T Ar_{jk+i} + \eta_j^{(j)} \sum_{t=i}^{s-1} \beta_{(j-1)k+t}^{(jk+i)} p_{(j-1)k+s+1}^T Ag_{(j-1)k+t}}{\eta_j^{(j)} p_{(j-1)k+s+1}^T Ag_{(j-1)k+s}} \\ &= -\frac{\eta_j^{(j)} p_{jk+s+1}^T r_{jk+i} + \rho_j \sum_{t=i}^{s-1} \beta_{(j-1)k+t}^{(jk+i)} \eta_{j-1}^{(j-1)} p_{(j-1)k+s+1}^T Ag_{(j-1)k+t}}{\rho_j \eta_{j-1}^{(j-1)} p_{(j-1)k+s+1}^T Ag_{(j-1)k+s}} \\ &= -\frac{\sum_{u=0}^j \eta_u^{(j)} p_{uk+s+1}^T r_{jk+i} + \rho_j \sum_{t=i}^{s-1} \beta_{(j-1)k+t}^{(jk+i)} \sum_{u=0}^{j-1} \eta_u^{(j-1)} p_{uk+s+1}^T Ag_{(j-1)k+t}}{\rho_j \sum_{u=0}^{j-1} \eta_u^{(j-1)} p_{uk+s+1}^T Ag_{(j-1)k+s}} \\ &= -\frac{\sum_{u=0}^j \eta_u^{(j)} q_{s+1}^T A^u r_{jk+i} + \rho_j \sum_{t=i}^{s-1} \beta_{(j-1)k+t}^{(jk+i)} \sum_{u=0}^{j-1} \eta_u^{(j-1)} q_{s+1}^T A^{u+1} g_{(j-1)k+t}}{\rho_j \sum_{u=0}^{j-1} \eta_u^{(j-1)} q_{s+1}^T A^{u+1} g_{(j-1)k+s}} \\ &= -\frac{q_{s+1}^T \psi_j(A) r_{jk+i} + \rho_j \sum_{t=i}^{s-1} \beta_{(j-1)k+t}^{(jk+i)} q_{s+1}^T A \psi_{j-1}(A) g_{(j-1)k+t}}{\rho_j q_{s+1}^T A \psi_{j-1}(A) g_{(j-1)k+s}} \\ &= -\frac{q_{s+1}^T \psi_j(A) r_{jk+i} + \sum_{t=i}^{s-1} \beta_{(j-1)k+t}^{(jk+i)} q_{s+1}^T (\psi_j(A) - \psi_{j-1}(A)) g_{(j-1)k+t}}{q_{s+1}^T (\psi_j(A) - \psi_{j-1}(A)) g_{(j-1)k+s}} \\ &= -\frac{q_{s+1}^T \left(\tilde{\pi}_{jk+i} + \sum_{t=i}^{s-1} \beta_{(j-1)k+t}^{(jk+i)} (\omega_{(j-1)k+t} - \tilde{\omega}_{(j-1)k+t}) \right)}{q_{s+1}^T (\omega_{(j-1)k+s} - \tilde{\omega}_{(j-1)k+s})} \end{aligned}$$

for $1 \leq j, 1 \leq i \leq s \leq k - 1$;

$$\begin{aligned}
 \beta_{(j-1)k+k}^{(jk+i)} &= \frac{\eta_{j+1}^{(j+1)} p_{jk+1}^T A r_{jk+i} + \sum_{t=i}^{k-1} \beta_{(j-1)k+t}^{(jk+i)} \eta_{j+1}^{(j+1)} p_{jk+1}^T A g_{(j-1)k+t}}{\eta_{j+1}^{(j+1)} p_{jk+1}^T A g_{(j-1)k+k}} \\
 &= \frac{\eta_{j+1}^{(j+1)} p_{(j+1)k+1}^T r_{jk+i} + \rho_{j+1} \sum_{t=i}^{k-1} \beta_{(j-1)k+t}^{(jk+i)} \eta_j^{(j)} p_{jk+1}^T A g_{(j-1)k+t}}{\rho_{j+1} \eta_j^{(j)} p_{jk+1}^T A g_{(j-1)k+k}} \\
 &= \frac{\sum_{u=0}^{j+1} \eta_u^{(j+1)} p_{uk+1}^T r_{jk+i} + \rho_{j+1} \sum_{t=i}^{k-1} \beta_{(j-1)k+t}^{(jk+i)} \sum_{u=0}^j \eta_u^{(j)} p_{uk+1}^T A g_{(j-1)k+t}}{\rho_{j+1} \sum_{u=0}^j \eta_u^{(j)} p_{uk+1}^T A g_{(j-1)k+k}} \\
 &= \frac{\sum_{u=0}^{j+1} \eta_u^{(j+1)} q_1^T A^u r_{jk+i} + \rho_{j+1} \sum_{t=i}^{k-1} \beta_{(j-1)k+t}^{(jk+i)} \sum_{u=0}^j \eta_u^{(j)} q_1^T A^{u+1} g_{(j-1)k+t}}{\rho_{j+1} \sum_{u=0}^j \eta_u^{(j)} q_1^T A^{u+1} g_{(j-1)k+k}} \\
 &= \frac{q_1^T \psi_{j+1}(A) r_{jk+i} + \rho_{j+1} \sum_{t=i}^{k-1} \beta_{(j-1)k+t}^{(jk+i)} q_1^T A \psi_j(A) g_{(j-1)k+t}}{\rho_{j+1} q_1^T A \psi_j(A) g_{(j-1)k+k}} \\
 &= \frac{q_1^T \left(\pi_{jk+i} + \rho_{j+1} \sum_{t=i}^{k-1} \beta_{(j-1)k+t}^{(jk+i)} A \omega_{(j-1)k+t} \right)}{\rho_{j+1} q_1^T A \omega_{(j-1)k+k}}
 \end{aligned}$$

for $1 \leq j, 1 \leq i \leq k$;

$$\begin{aligned}
 \beta_{jk+s}^{(jk+i)} &= - \left(\eta_{j+1}^{(j+1)} p_{jk+s+1}^T A r_{jk+i} + \eta_{j+1}^{(j+1)} \sum_{t=i}^k \beta_{(j-1)k+t}^{(jk+i)} p_{jk+s+1}^T A g_{(j-1)k+t} \right. \\
 &\quad \left. + \eta_{j+1}^{(j+1)} \sum_{t=1}^{s-1} \beta_{jk+t}^{(jk+i)} p_{jk+s+1}^T A g_{jk+t} \right) / \eta_{j+1}^{(j+1)} p_{jk+s+1}^T A g_{jk+s} \\
 &= - \left(\eta_{j+1}^{(j+1)} p_{(j+1)k+s+1}^T r_{jk+i} + \rho_{j+1} \sum_{t=i}^k \beta_{(j-1)k+t}^{(jk+i)} \eta_j^{(j)} p_{jk+s+1}^T A g_{(j-1)k+t} \right. \\
 &\quad \left. + \rho_{j+1} \sum_{t=1}^{s-1} \beta_{jk+t}^{(jk+i)} \eta_j^{(j)} p_{jk+s+1}^T A g_{jk+t} \right) / \rho_{j+1} \eta_j^{(j)} p_{jk+s+1}^T A g_{jk+s} \\
 &= - \left(\sum_{u=0}^{j+1} \eta_u^{(j+1)} p_{uk+s+1}^T r_{jk+i} + \rho_{j+1} \sum_{t=i}^k \beta_{(j-1)k+t}^{(jk+i)} \sum_{u=0}^j \eta_u^{(j)} p_{uk+s+1}^T A g_{(j-1)k+t} \right. \\
 &\quad \left. + \rho_{j+1} \sum_{t=1}^{s-1} \beta_{jk+t}^{(jk+i)} \sum_{u=0}^j \eta_u^{(j)} p_{uk+s+1}^T A g_{jk+t} \right) / \rho_{j+1} \sum_{u=0}^j \eta_u^{(j)} p_{uk+s+1}^T A g_{jk+s}
 \end{aligned}$$

$$\begin{aligned}
 &= - \left(\sum_{u=0}^{j+1} \eta_u^{(j+1)} q_{s+1}^T A^u r_{jk+i} + \rho_{j+1} \sum_{t=i}^k \beta_{(j-1)k+t}^{(jk+i)} \sum_{u=0}^j \eta_u^{(j)} q_{s+1}^T A^{u+1} g_{(j-1)k+t} \right. \\
 &\quad \left. + \rho_{j+1} \sum_{t=1}^{s-1} \beta_{jk+t}^{(jk+i)} \sum_{u=0}^j \eta_u^{(j)} q_{s+1}^T A^{u+1} g_{jk+t} \right) / \rho_{j+1} \sum_{u=0}^j \eta_u^{(j)} q_{s+1}^T A^{u+1} g_{jk+s} \\
 &= - \left(q_{s+1}^T \psi_{j+1}(A) r_{jk+i} + \rho_{j+1} \sum_{t=i}^k \beta_{(j-1)k+t}^{(jk+i)} q_{s+1}^T A \psi_j(A) g_{(j-1)k+t} \right. \\
 &\quad \left. + \rho_{j+1} \sum_{t=1}^{s-1} \beta_{jk+t}^{(jk+i)} q_{s+1}^T A \psi_j(A) g_{jk+t} \right) / \rho_{j+1} q_{s+1}^T A \psi_j(A) g_{jk+s} \\
 &= - \left(q_{s+1}^T \psi_{j+1}(A) r_{jk+i} + \rho_{j+1} \sum_{t=i}^k \beta_{(j-1)k+t}^{(jk+i)} q_{s+1}^T A \psi_j(A) g_{(j-1)k+t} \right. \\
 &\quad \left. + \sum_{t=1}^{s-1} \beta_{jk+t}^{(jk+i)} q_{s+1}^T (\psi_{j+1}(A) - \psi_j(A)) g_{jk+t} \right) / q_{s+1}^T (\psi_{j+1}(A) - \psi_j(A)) g_{jk+s} \\
 &= - \frac{q_{s+1}^T \left(\pi_{jk+i} + \rho_{j+1} \sum_{t=i}^k \beta_{(j-1)k+t}^{(jk+i)} A \omega_{(j-1)k+t} + \sum_{t=1}^{s-1} \beta_{jk+t}^{(jk+i)} (\omega_{jk+t} - \tilde{\omega}_{jk+t}) \right)}{q_{s+1}^T (\omega_{jk+s} - \tilde{\omega}_{jk+s})}
 \end{aligned}$$

for $1 \leq j, 1 \leq s < i \leq k$.

The detailed derivation of the formulas for updating $\tilde{\pi}_{jk+i}, \pi_{jk+i}, \tilde{\omega}_{jk+i}$ and ω_{jk+i} are given below. We have used the formulas in Algorithm 1 in our derivations.

$$\begin{aligned}
 \tilde{\pi}_{jk+1} &= \psi_j(A) r_{jk+1} \\
 &= \psi_j(A) (r_{(j-1)k+k} - \alpha_{jk+1} A g_{(j-1)k+k}) \\
 &= \pi_{(j-1)k+k} - \alpha_{jk+1} A \omega_{(j-1)k+k}
 \end{aligned}$$

for $0 \leq j$;

$$\begin{aligned}
 \pi_{jk+1} &= \psi_{j+1}(A) r_{jk+1} \\
 &= (\rho_{j+1} A \psi_j(A) + \psi_j(A)) r_{jk+1} \\
 &= \rho_{j+1} A \tilde{\pi}_{jk+1} + \tilde{\pi}_{jk+1}
 \end{aligned}$$

for $0 \leq j$;

$$\begin{aligned}
 \tilde{\pi}_{jk+i+1} &= \psi_j(A)r_{jk+i+1} \\
 &= \psi_j(A)(r_{jk+i} - \alpha_{jk+i+1}Ag_{jk+i}) \\
 &= \psi_j(A)r_{jk+i} - \frac{\alpha_{jk+i+1}}{\rho_{j+1}}(\psi_{j+1}(A) - \psi_j(A))g_{jk+i} \\
 &= \tilde{\pi}_{jk+i} - \frac{\alpha_{jk+i+1}}{\rho_{j+1}}(\omega_{jk+i} - \tilde{\omega}_{jk+i})
 \end{aligned}$$

for $1 \leq i < k, 0 \leq j$;

$$\begin{aligned}
 \pi_{jk+i+1} &= \psi_{j+1}(A)r_{jk+i+1} \\
 &= \psi_{j+1}(A)(r_{jk+i} - \alpha_{jk+i+1}Ag_{jk+i}) \\
 &= \pi_{jk+i} - \alpha_{jk+i+1}A\omega_{jk+i}
 \end{aligned}$$

for $1 \leq i < k, 0 \leq j$;

$$\begin{aligned}
 \tilde{\omega}_i &= \psi_0(A)g_i \\
 &= \psi_0(A)\left(r_i + \sum_{s=0}^{i-1} \beta_s^{(i)}g_s\right) \\
 &= \tilde{\pi}_i + \beta_0^{(i)}\omega_0 + \sum_{s=1}^{i-1} \beta_s^{(i)}\tilde{\omega}_s
 \end{aligned}$$

for $1 \leq i \leq k$;

$$\begin{aligned}
 \omega_i &= \psi_1(A)g_i \\
 &= \psi_1(A)\left(r_i + \sum_{s=0}^{i-1} \beta_s^{(i)}g_s\right) \\
 &= \psi_1(A)r_i + \beta_0^{(i)}\psi_1(A)g_0 + \sum_{s=1}^{i-1} \beta_s^{(i)}\psi_1(A)g_s \\
 &= \pi_i + \beta_0^{(i)}(\rho_1A\omega_0 + \omega_0) + \sum_{s=1}^{i-1} \beta_s^{(i)}\omega_s
 \end{aligned}$$

for $1 \leq i \leq k$;

$$\begin{aligned}
 \tilde{\omega}_{jk+i} &= \psi_j(A)g_{jk+i} \\
 &= \psi_j(A)\left(r_{jk+i} + \sum_{s=i}^k \beta_{(j-1)k+s}^{(jk+i)}g_{(j-1)k+s} + \sum_{s=1}^{i-1} \beta_{jk+s}^{(jk+i)}g_{jk+s}\right) \\
 &= \tilde{\pi}_{jk+i} + \sum_{s=i}^k \beta_{(j-1)k+s}^{(jk+i)}\omega_{(j-1)k+s} + \sum_{s=1}^{i-1} \beta_{jk+s}^{(jk+i)}\tilde{\omega}_{jk+s}
 \end{aligned}$$

for $1 \leq j, 1 \leq i \leq k$;

$$\begin{aligned}
\omega_{jk+i} &= \psi_{j+1}(A)g_{jk+i} \\
&= \psi_{j+1}(A) \left(r_{jk+i} + \sum_{s=i}^k \beta_{(j-1)k+s}^{(jk+i)} g_{(j-1)k+s} + \sum_{s=1}^{i-1} \beta_{jk+s}^{(jk+i)} g_{jk+s} \right) \\
&= \psi_{j+1}(A)r_{jk+i} + \sum_{s=i}^k \beta_{(j-1)k+s}^{(jk+i)} (\rho_{j+1}A + I)\psi_j(A)g_{(j-1)k+s} \\
&\quad + \sum_{s=1}^{i-1} \beta_{jk+s}^{(jk+i)} \psi_{j+1}(A)g_{jk+s} \\
&= \pi_{jk+i} + \sum_{s=i}^k \beta_{(j-1)k+s}^{(jk+i)} (\rho_{j+1}A\omega_{(j-1)k+s} + \omega_{(j-1)k+s}) + \sum_{s=1}^{i-1} \beta_{jk+s}^{(jk+i)} \omega_{jk+s}
\end{aligned}$$

for $1 \leq j, 1 \leq i \leq k$, where I is the $n \times n$ identity matrix.

Acknowledgements. The first author appreciates the valuable discussions with the research groups of Professors Youcef Saad and Alan Edelman during his visit to the University of Minnesota and Massachusetts Institute of Technology in summer 1996. In addition, sincere thanks go to Dr. Edmond Chow, Professors Henk A. van der Vorst and Qiang Ye, and the referees for their valuable comments on an earlier draft. Professor Saad's book [18] has also been invaluable.

REFERENCES

- [1] J. ALIAGA, D. BOLEY, R. FREUND, AND V. HERNÁNDEZ, *A Lanczos-type method for multiple starting vectors*, Numerical Analysis Manuscript 96-18, Bell Laboratories, Murray Hill, NJ, 1996; also available online at <http://cm.bell-labs.com/cs/doc/96>.
- [2] R. BARRETT, M. BERRY, T. F. CHAN, J. DEMMEL, J. DONATO, J. DONGARRA, V. ELJKHOUT, R. POZO, C. ROMINE, AND H. VAN DER VORST, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, PA, 1994.
- [3] T. CHAN AND Q. YE, *A mixed product Krylov subspace method for solving nonsymmetric linear systems*, University of California at Los Angeles Cam. Report 97-53, 1997; also available online at <http://www.math.ucla.edu/applied/cam/index.html>.
- [4] I. S. DUFF, R. G. GRIMES, AND J. G. LEWIS, *Sparse matrix test problems*, ACM Trans. Math. Software, 15 (1989), pp. 1–14.
- [5] A. EDELMAN, *The first annual large dense linear system survey*, The SIGNAL Newsletter, 26 (1991), pp. 6–12.
- [6] J. ERHEL, K. BURRAGE, AND B. POHL, *Restarted GMRES preconditioned by deflation*, J. Comput. Appl. Math., 69 (1996), pp. 303–318.
- [7] R. FLETCHER, *Conjugate gradient methods for indefinite systems*, in Proceedings of the Dundee Biennial Conference on Numerical Analysis 1974, G. A. Watson, ed., Springer-Verlag, New York, 1975, pp. 73–89.
- [8] R. FREUND AND M. MALHOTRA, *A Block-QMR Algorithm for Non-Hermitian Linear Systems with Multiple Right-hand Sides*, Numerical Analysis Manuscript 95-09, AT&T Bell Laboratories, Murray Hill, NJ, 1995; also available online at <http://cm.bell-labs.com/cs/doc/95>.
- [9] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, 2nd ed., The Johns Hopkins University Press, Baltimore, MD, 1989.
- [10] M. H. GUTKNECHT, *Variants of BICGStab for matrices with complex spectrum*, SIAM J. Sci. Comput., 14 (1993), pp. 1020–1033.
- [11] M. H. GUTKNECHT, *Lanczos-type solvers for nonsymmetric linear systems of equations*, Acta Numerica, 1997, pp. 271–397.
- [12] C. LANCZOS, *Solution of systems of linear equations by minimized iterations*, J. Research Nat. Bureau of Standards, 49 (1952), pp. 33–53.
- [13] R. B. MORGAN, *A restarted GMRES method augmented with eigenvectors*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 1154–1171.
- [14] A. RUHE, *Implementation aspects of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices*, Math. Comp., 33 (1979), pp. 680–687.

- [15] Y. SAAD, *Practical use of some Krylov subspace methods for solving indefinite and nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 203–228.
- [16] Y. SAAD, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.
- [17] Y. SAAD, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Statist. Comput., 14 (1993), pp. 461–469.
- [18] Y. SAAD, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, Boston, 1996.
- [19] G. L. G. SLEIJPEN AND D. R. FOKKEMA, *BiCGSTAB(k) for linear equations involving nonsymmetric matrices with complex spectrum*, Electron. Trans. Numer. Anal., 1 (1993), pp. 11–32, 1993.
- [20] G. L. G. SLEIJPEN AND H. A. VAN DER VORST, *Maintaining convergence properties of BiCGSTAB methods in finite precision arithmetic*, Numer. Algorithms, 10 (1995), pp. 203–223.
- [21] G. L. G. SLEIJPEN AND H. A. VAN DER VORST, *An overview of approaches for the stable computation of hybrid BiCG methods*, Appl. Numer. Math., 19 (1995), pp. 235–254.
- [22] G. L. G. SLEIJPEN AND H. A. VAN DER VORST, *Reliable updated residuals in hybrid Bi-CG methods*, Computing, 56 (1996), pp. 141–163.
- [23] P. SONNEVELD, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 36–52.
- [24] 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. Statist. Comput., 12 (1992), pp. 631–644.
- [25] H. A. VAN DER VORST AND C. VUIK, *GMRESR: A family of nested GMRES methods*, Numer. Linear Algebra Appl., 1 (1994), pp. 369–386.