

# ML( $N$ )BICGSTAB: REFORMULATION, ANALYSIS AND IMPLEMENTATION

MAN-CHUNG YEUNG\*

**Abstract.** With the help of index functions, we re-derive the ML( $n$ )BiCGStab algorithm in [35] in a more systematic way. There are  $n$  ways to define the ML( $n$ )BiCGStab residual vector. Each different definition will lead to a different ML( $n$ )BiCGStab algorithm. We demonstrate this by deriving a second algorithm which requires less storage. We also analyze the breakdown situations and summarize some useful properties about ML( $n$ )BiCGStab. Implementation issues are also addressed. In particular, we discuss in details on the choices of the parameters in ML( $n$ )BiCGStab.

**Key words.** CGS, BiCGStab, ML( $n$ )BiCGStab, multiple starting Lanczos, Krylov subspace, iterative methods, linear systems

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

**1. Introduction.** If we express the BiCG[5, 14] residual as  $\mathbf{r}_k^{BiCG} = p_k(\mathbf{A})\mathbf{r}_0$  in terms of a polynomial, the residual vector  $\mathbf{r}_k$  of a Lanczos-type product method<sup>1</sup> based on BiCG is defined to be  $\mathbf{r}_k = \phi_k(\mathbf{A})p_k(\mathbf{A})\mathbf{r}_0$  where  $\phi_k(\lambda)$  is some polynomial of degree  $k$  with  $\phi_k(0) = 1$ . In CGS[28],  $\phi_k = p_k$ . Since, in every iteration, CGS searches for an approximate solution in a larger Krylov subspace, it often converges much faster than BiCG. However, CGS usually behaves irregularly due to a lack of a smoothing mechanism. In BiCGStab[30], the  $\phi_k$  is

$$(1.1) \quad \phi_k(\lambda) = \begin{cases} 1 & \text{if } k = 0 \\ (\rho_k \lambda + 1)\phi_{k-1}(\lambda) & \text{if } k > 0. \end{cases}$$

The  $\rho_k$  is a free parameter and is selected to minimize the 2-norm of  $\mathbf{r}_k^{BiCGStab}$  in the  $k$ -iteration, that is,  $\rho_k \lambda + 1$  is the GMRES(1)[19] polynomial. As a result, BiCGStab is generally more stable and robust than CGS. BiCGStab has been extended to BiCGStab2[9] and BiCGStab( $l$ )[22, 27] through the use of higher degree minimizing polynomials. In BiCGStab2, the  $\phi_k$  is defined by

$$\phi_k(\lambda) = \begin{cases} 1 & \text{if } k = 0 \\ (\rho_k \lambda + 1)\phi_{k-1}(\lambda) & \text{if } k \text{ is odd} \\ ((\alpha_k \lambda + \beta_k)(\rho_{k-1} \lambda + 1) + 1 - \beta_k)\phi_{k-2}(\lambda) & \text{if } k \text{ is even.} \end{cases}$$

The first and the second degree factors in the expression of  $\phi_k(\lambda)$  are the GMRES(1) and GMRES(2) polynomials respectively. On the other hand, BiCGStab( $l$ ) defines its  $\phi_k$  as

$$\phi_k(\lambda) = \begin{cases} 1 & \text{if } k = 0 \\ (1 + \sum_{j=1}^l \alpha_j \lambda^j)\phi_{k-l}(\lambda) & \text{if } k \text{ is a multiple of } l \end{cases}$$

where the factor  $1 + \sum_{j=1}^l \alpha_j \lambda^j$  is the GMRES( $l$ ) polynomial. BiCGStab2 and BiCGStab( $l$ ) usually converge faster than BiCGStab because of smaller residuals in magnitude while avoiding near-breakdown caused by a possibly too small  $\rho$ . CGS,

\*Dept. 3036, 1000 East University Avenue, Laramie, WY 82071. E-mail: myeung@uwyo.edu. This research was supported by Flittie Sabbatical Augmentation Award, University of Wyoming.

<sup>1</sup>For this type of Krylov subspace methods, one can consult [10]. They are called hybrid BiCG methods in [25].

BiCGStab and BiCGStab2 have been summarized into GPBi-CG[36]. The  $\phi_k$  of GPBi-CG is

$$\phi_k(\lambda) = \begin{cases} 1 & \text{if } k = 0 \\ \rho_1\lambda + 1 & \text{if } k = 1 \\ (\alpha_k\lambda + 1 + \beta_k)\phi_{k-1}(\lambda) - \beta_k\phi_{k-2}(\lambda) & \text{if } k > 1. \end{cases}$$

GPBi-CG will become CGS, BiCGStab or BiCGStab2 when the  $\alpha, \beta, \rho$  are appropriately chosen. For detailed descriptions of these and other product-type methods, one is referred to [18, 20, 21, 31] and the references therein. Moreover, a complete development history of product-type methods can be found in [11]. According to [11], the history dates back as early as to the IDR[33] method three decades ago.

Generalizations of BiCGStab from the standard Lanczos-based BiCG to versions from BiCG's of other types have been made. For example, BL-BiCGStab[8] is a BiCGStab variant built on the BL-BiCG[16] for the solution of systems with multiple right-hand sides. Also, ML( $n$ )BiCGStab[35] is another BiCGStab variant built on ML( $n$ )BiCG, a BiCG-like method derived from a variant of the Lanczos-type process described in [1] with  $n$  left-starting vectors and a single right-starting vector.

The derivation of the ML( $n$ )BiCGStab algorithm in [35] was complicated. In this paper, we shall exploit the concept of index functions to re-derive the algorithm in a more systematic way. Index functions were introduced in [34] by Boley for the purpose of simplifying the development of the transpose-free multiple starting Lanczos process there and they proved to be very helpful.

It turns out that the definition of the ML( $n$ )BiCGStab residual vector  $\mathbf{r}_k$  in [35] is not unique. There are at least  $n$  different ways to define  $\mathbf{r}_k$ . Let  $\widehat{\mathbf{r}}_k$  be the residual in ML( $n$ )BiCG and  $\phi_k(\lambda)$  as in (1.1). Then, the ML( $n$ )BiCGStab residual  $\mathbf{r}_k$  introduced in [35] is

$$(1.2) \quad \mathbf{r}_{jn+i} = \phi_j(\mathbf{A})\widehat{\mathbf{r}}_{jn+i}, \quad 1 \leq i \leq n, j = 0, 1, \dots$$

Starting from  $k = 1$ , let us call every  $n$  consecutive iterations an iteration cycle. For example, iterations  $k = 1, 2, \dots, n$  form the first cycle, iterations  $k = n + 1, n + 2, \dots, 2n$  the second cycle and so on. Then definition (1.2) increases the degree of  $\phi$  by 1 at the beginning of a cycle. We believe that one can define  $\mathbf{r}_k$  by increasing the degree of  $\phi$  by 1 anywhere within a cycle and this will lead to a different ML( $n$ )BiCGStab algorithm. As an illustration, we derive and present a second ML( $n$ )BiCGStab algorithm associated with the definition

$$(1.3) \quad \mathbf{r}_{jn+i} = \begin{cases} \phi_j(\mathbf{A})\widehat{\mathbf{r}}_{jn+i} & \text{if } 1 \leq i \leq n-1 \\ \phi_{j+1}(\mathbf{A})\widehat{\mathbf{r}}_{jn+i} & \text{if } i = n. \end{cases}$$

(1.3) increases the degree of  $\phi$  by 1 at the end of a cycle. The resulting algorithm requires about 25% less storage (besides storing the coefficient matrix and the preconditioner) than the algorithm associated with definition (1.2). However, one drawback with this storage-saving algorithm is that its computed residual  $\mathbf{r}_k$  easily diverges from its corresponding exact residual when  $n$  is moderately large.

A ML( $n$ )BiCGStab algorithm mainly involves three types of operations: matrix-vector multiplication and preconditioner system solving ( $\mathbf{A}\mathbf{M}^{-1}\mathbf{v}$ ), dot product ( $\mathbf{u}^H\mathbf{v}$ ) and saxpy ( $\mathbf{u} + \alpha\mathbf{v}$ ). Usually, computing  $\mathbf{A}\mathbf{M}^{-1}\mathbf{v}$  takes the longest. In one iteration, ML( $n$ )BiCGStab requires  $1 + 1/n$  of  $\mathbf{A}\mathbf{M}^{-1}\mathbf{v}$ ,  $n$  of  $\mathbf{u}^H\mathbf{v}$  and about  $2n$  of  $\mathbf{u} + \alpha\mathbf{v}$  on the average. These observations suggest that we can employ the parameter  $n$  to

$k$	0	1 2 3	4 5 6	7 8 9	10 11 12	...
$g_n(k)$	-1	0 0 0	1 1 1	2 2 2	3 3 3	...
$r_n(k)$	3	1 2 3	1 2 3	1 2 3	1 2 3	...

FIG. 2.1. Simple illustration of the index functions for  $n = 3$ .

take a balance among these operations, that is, we want the computational time per iteration is minimized. It should be ideal if the total number of iterations to reach convergence were minimized too, but doing so is difficult since it is hard to predict the overall performance on a problem. The idea of minimizing the time per iteration has led to a Matlab function *find.n.m* which automatically finds a “good”  $n$ . Numerical experiments show that *find.n.m* plus a ML( $n$ )BiCGStab algorithm can be faster than BiCGStab in terms of time.

Just like BiCGStab, ML( $n$ )BiCGStab can suffer from three types of breakdown, caused respectively by the failure of the underlying Lanczos process, the non-existence of the  $LU$  factorization in the construction of ML( $n$ )BiCG and the parameter  $\rho$ . We will address these issues. Also, we will summarize some properties of ML( $n$ )BiCGStab.

The outline of the paper is as follows. In §2, we introduce index functions. In §3, we present the ML( $n$ )BiCG algorithm obtained in [35], based on which a ML( $n$ )BiCGStab algorithm is derived. In §4, we re-derive the ML( $n$ )BiCGStab algorithm in [35] using index functions. In §5, we derive a storage-saving ML( $n$ )BiCGStab algorithm from a different definition of the residual vector. In §6, we discuss the relationships of ML( $n$ )BiCGStab with some existing methods. In §7, implementation issues are addressed. We end the paper with conclusions and remarks in §8.

**2. Index Functions.** Let be given a positive integer  $n$ . Define

$$g_n(k) = \lfloor (k-1)/n \rfloor \quad \text{and} \quad r_n(k) = k - ng_n(k)$$

where  $k \in \mathcal{Z}$ , the set of all integers, and  $\lfloor \cdot \rfloor$  rounds its argument to the nearest integer towards minus infinity. We call  $g_n$  and  $r_n$  index functions and they are defined on  $\mathcal{Z}$  with ranges  $\mathcal{Z}$  and  $\{1, 2, \dots, n\}$  respectively.

If we write

$$(2.1) \quad k = jn + i$$

with  $1 \leq i \leq n$  and  $j \in \mathcal{Z}$ , then

$$(2.2) \quad g_n(jn + i) = j \quad \text{and} \quad r_n(jn + i) = i.$$

Fig. 2.1 illustrates the behaviors of  $g_n$  and  $r_n$  with  $n = 3$ .

About the index functions, they have the following properties which can be easily verified by using (2.2).

**PROPOSITION 2.1.** Let  $k \in \mathcal{N}$ , the set of all positive integers, and  $s \in \mathcal{N}_0 := \mathcal{N} \cup \{0\}$ .

- (a)  $g_n(k+n) = g_n(k) + 1$  and  $r_n(k+n) = r_n(k)$ .
- (b)  $g_n(s+1) + 1 = g_n(k+1)$  if  $\max(k-n, 0) \leq s \leq g_n(k)n - 1$ .
- (c)  $g_n(s+1) = g_n(g_n(k)n + 1) = g_n(k)$  if  $g_n(k)n \leq s \leq k - 1$ .
- (d)  $g_n(k+1) = g_n(k) + 1$  if  $r_n(k) = n$ .  
 $g_n(k+1) = g_n(k)$  if  $r_n(k) < n$ .
- (e)  $\max(k-n, 0) > g_n(k)n - 1$  if  $r_n(k) = n$  or  $g_n(k) = 0$ .

**3. A ML( $n$ )BiCG Algorithm.** Parallel to the derivation of BiCGStab from BiCG, ML( $n$ )BiCGStab was derived from a BiCG-like method named ML( $n$ )BiCG in [35]. ML( $n$ )BiCG was built upon a Lanczos-type process with  $n$  left strating vectors and a single right starting vector, a special case of the Lanczos-type process described in [1]. In this section, we present the algorithm of ML( $n$ )BiCG from [35] and summarize some properties related to it.

**3.1. The Algorithm.** Consider the solution of the linear system<sup>2</sup>

$$(3.1) \quad \mathbf{A}\mathbf{x} = \mathbf{b}$$

where  $\mathbf{A} \in \mathcal{C}^{N \times N}$  and  $\mathbf{b} \in \mathcal{C}^N$ . Let be given  $n$  vectors  $\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n \in \mathcal{C}^N$ . Set

$$(3.2) \quad \mathbf{p}_k = (\mathbf{A}^H)^{g_n(k)} \mathbf{q}_{r_n(k)}$$

for  $k = 1, 2, \dots$ . The following algorithm for the solution of (3.1) is from [35].

ALGORITHM 3.1. ML( $n$ )BiCG<sup>3</sup>

1. Choose an initial guess  $\widehat{\mathbf{x}}_0$  and  $n$  vectors  $\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n$ .
2. Compute  $\widehat{\mathbf{r}}_0 = \mathbf{b} - \mathbf{A}\widehat{\mathbf{x}}_0$  and set  $\mathbf{p}_1 = \mathbf{q}_1$ ,  $\widehat{\mathbf{g}}_0 = \widehat{\mathbf{r}}_0$ .
3. For  $k = 1, 2, \dots$ , until convergence:
4.  $\alpha_k = \mathbf{p}_k^H \widehat{\mathbf{r}}_{k-1} / \mathbf{p}_k^H \mathbf{A} \widehat{\mathbf{g}}_{k-1}$ ;
5.  $\widehat{\mathbf{x}}_k = \widehat{\mathbf{x}}_{k-1} + \alpha_k \widehat{\mathbf{g}}_{k-1}$ ;
6.  $\widehat{\mathbf{r}}_k = \widehat{\mathbf{r}}_{k-1} - \alpha_k \mathbf{A} \widehat{\mathbf{g}}_{k-1}$ ;
7. For  $s = \max(k - n, 0), \dots, k - 1$
8.  $\beta_s^{(k)} = -\mathbf{p}_{s+1}^H \mathbf{A} \left( \widehat{\mathbf{r}}_k + \sum_{t=\max(k-n,0)}^{s-1} \beta_t^{(k)} \widehat{\mathbf{g}}_t \right) / \mathbf{p}_{s+1}^H \mathbf{A} \widehat{\mathbf{g}}_s$ ;
9. End
10.  $\widehat{\mathbf{g}}_k = \widehat{\mathbf{r}}_k + \sum_{s=\max(k-n,0)}^{k-1} \beta_s^{(k)} \widehat{\mathbf{g}}_s$ ;
11. Compute  $\mathbf{p}_{k+1}$  according to (3.2)
12. End

**3.2. Properties.** Let  $\nu$  be the degree of the minimal polynomial  $p_{min}(\lambda, \mathbf{A}, \widehat{\mathbf{r}}_0)$  of  $\widehat{\mathbf{r}}_0$  with respect to  $\mathbf{A}$  (that is, the unique monic polynomial  $p(\lambda)$  of minimum degree such that  $p(\mathbf{A})\widehat{\mathbf{r}}_0 = \mathbf{0}$ ) and let

$$\widehat{\mathbf{S}}_\nu = [\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_\nu]^H [\mathbf{A}\widehat{\mathbf{r}}_0, \mathbf{A}^2\widehat{\mathbf{r}}_0, \dots, \mathbf{A}^\nu\widehat{\mathbf{r}}_0]$$

and

$$\widehat{\mathbf{W}}_\nu = [\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_\nu]^H [\widehat{\mathbf{r}}_0, \mathbf{A}\widehat{\mathbf{r}}_0, \dots, \mathbf{A}^{\nu-1}\widehat{\mathbf{r}}_0].$$

Denote by  $\widehat{\mathbf{S}}_l$  and  $\widehat{\mathbf{W}}_l$  the  $l \times l$  leading principal submatrices of  $\widehat{\mathbf{S}}_\nu$  and  $\widehat{\mathbf{W}}_\nu$  respectively. We now summarize some useful facts about Algorithm 3.1. These facts can be seen from the construction procedure of the algorithm.

<sup>2</sup>Throughout the paper we do not assume the matrix  $\mathbf{A}$  is nonsingular except where specified.

<sup>3</sup>Even though we have not tested the algorithm, we believe it is numerically instable because of Line 11 in which  $\mathbf{A}^H$  is successively multiplied to a same vector, a type of operation which is highly sensitive to round-off errors. The algorithm is established only for the purpose of developing ML( $n$ )BiCGStab algorithms.

PROPOSITION 3.2. *In infinite precision arithmetic, if  $\prod_{l=1}^{\nu} \det(\widehat{\mathbf{S}}_l) \det(\widehat{\mathbf{W}}_l) \neq 0$ , then Algorithm 3.1 does not break down by zero division for  $k = 1, 2, \dots, \nu$ , and the approximate solution  $x_{\nu}$  at step  $\nu$  is exact to the system (3.1). Moreover, the computed quantities satisfy*

- (a)  $\widehat{\mathbf{x}}_k \in \widehat{\mathbf{x}}_0 + \text{span}\{\widehat{\mathbf{r}}_0, \mathbf{A}\widehat{\mathbf{r}}_0, \dots, \mathbf{A}^{k-1}\widehat{\mathbf{r}}_0\}$  and  $\widehat{\mathbf{r}}_k = \mathbf{b} - \mathbf{A}\widehat{\mathbf{x}}_k \in \widehat{\mathbf{r}}_0 + \text{span}\{\mathbf{A}\widehat{\mathbf{r}}_0, \mathbf{A}^2\widehat{\mathbf{r}}_0, \dots, \mathbf{A}^k\widehat{\mathbf{r}}_0\}$  for  $1 \leq k \leq \nu$ .
- (b)  $\text{span}\{\widehat{\mathbf{r}}_0, \widehat{\mathbf{r}}_1, \dots, \widehat{\mathbf{r}}_{k-1}\} = \text{span}\{\widehat{\mathbf{r}}_0, \mathbf{A}\widehat{\mathbf{r}}_0, \dots, \mathbf{A}^{k-1}\widehat{\mathbf{r}}_0\}$  for  $1 \leq k \leq \nu$ .
- (c)  $\text{span}\{\mathbf{A}\widehat{\mathbf{r}}_0, \mathbf{A}\widehat{\mathbf{r}}_1, \dots, \mathbf{A}\widehat{\mathbf{r}}_{\nu-1}\} = \text{span}\{\widehat{\mathbf{r}}_0, \mathbf{A}\widehat{\mathbf{r}}_0, \dots, \mathbf{A}^{\nu-1}\widehat{\mathbf{r}}_0\}$ .
- (d)  $\widehat{\mathbf{r}}_k \perp \text{span}\{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_k\}$  and  $\widehat{\mathbf{r}}_k \not\perp \mathbf{p}_{k+1}$  for  $0 \leq k \leq \nu - 1$ .<sup>4</sup>
- (e)  $\text{span}\{\widehat{\mathbf{g}}_0, \widehat{\mathbf{g}}_1, \dots, \widehat{\mathbf{g}}_{k-1}\} = \text{span}\{\widehat{\mathbf{r}}_0, \mathbf{A}\widehat{\mathbf{r}}_0, \dots, \mathbf{A}^{k-1}\widehat{\mathbf{r}}_0\}$  for  $1 \leq k \leq \nu$ .
- (f)  $\text{span}\{\mathbf{A}\widehat{\mathbf{g}}_0, \mathbf{A}\widehat{\mathbf{g}}_1, \dots, \mathbf{A}\widehat{\mathbf{g}}_{\nu-1}\} = \text{span}\{\widehat{\mathbf{r}}_0, \mathbf{A}\widehat{\mathbf{r}}_0, \dots, \mathbf{A}^{\nu-1}\widehat{\mathbf{r}}_0\}$ .
- (g)  $\mathbf{A}\widehat{\mathbf{g}}_k \perp \text{span}\{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_k\}$  and  $\mathbf{A}\widehat{\mathbf{g}}_k \not\perp \mathbf{p}_{k+1}$  for  $0 \leq k \leq \nu - 1$ .

Because of Proposition 3.2(a) and (d), ML( $n$ )BiCG is an oblique projection method according to [18].

Remarks: (i) Just like BiCG, ML( $n$ )BiCG also has two types of breakdown caused respectively by the failure of the underlying Lanczos process and the nonexistence of the  $LU$  factorization in its construction. The condition  $\prod_{l=1}^{\nu} \det(\widehat{\mathbf{W}}_l) \neq 0$  guarantees that the underlying Lanczos process works without break-down, and the condition  $\prod_{l=1}^{\nu} \det(\widehat{\mathbf{S}}_l) \neq 0$  makes sure the  $LU$  factorizations exit; (ii)  $\det(\widehat{\mathbf{S}}_{\nu}) \neq 0$  implies that  $p_{\min}(0, \mathbf{A}, \widehat{\mathbf{r}}_0) \neq 0$  which, in turn, implies that (3.1) is consistent and there is a solution lying in  $\widehat{\mathbf{x}}_0 + \text{span}\{\widehat{\mathbf{r}}_0, \mathbf{A}\widehat{\mathbf{r}}_0, \dots, \mathbf{A}^{\nu-1}\widehat{\mathbf{r}}_0\}$ .

The derivation of ML( $n$ )BiCGStab needs the following result.

COROLLARY 3.3. *Let  $s \in \mathcal{N}$  and*

$$\psi_{g_n(s)}(\lambda) = c_{g_n(s)} \lambda^{g_n(s)} + c_{g_n(s)-1} \lambda^{g_n(s)-1} + \dots + c_0$$

be any polynomial of degree  $g_n(s)$ . Then, under the assumptions of Proposition 3.2,

$$\mathbf{p}_s^H \widehat{\mathbf{r}}_k = \frac{1}{c_{g_n(s)}} \mathbf{q}_{r_n(s)}^H \psi_{g_n(s)}(\mathbf{A}) \widehat{\mathbf{r}}_k \quad \text{and} \quad \mathbf{p}_s^H \mathbf{A} \widehat{\mathbf{g}}_k = \frac{1}{c_{g_n(s)}} \mathbf{q}_{r_n(s)}^H \mathbf{A} \psi_{g_n(s)}(\mathbf{A}) \widehat{\mathbf{g}}_k$$

if  $0 \leq k \leq \nu - 1$  and  $s \leq k + n$ .

*Proof.* We only verify the equation about  $\widehat{\mathbf{r}}_k$  here. If  $g_n(s) = 0$ , then  $\mathbf{p}_s = \mathbf{q}_{r_n(s)}$  by (3.2) and hence the equation follows. For  $g_n(s) > 0$  and  $1 \leq l \leq g_n(s)$ ,

$$\mathbf{q}_{r_n(s)}^H \mathbf{A}^{g_n(s)-l} \widehat{\mathbf{r}}_k = \mathbf{q}_{r_n(s-ln)}^H \mathbf{A}^{g_n(s-ln)} \widehat{\mathbf{r}}_k = \mathbf{p}_{s-ln}^H \widehat{\mathbf{r}}_k = 0$$

by Proposition 2.1(a), (3.2) and Proposition 3.2(d). Consequently,

$$\mathbf{q}_{r_n(s)}^H \psi_{g_n(s)}(\mathbf{A}) \widehat{\mathbf{r}}_k = c_{g_n(s)} \mathbf{q}_{r_n(s)}^H \mathbf{A}^{g_n(s)} \widehat{\mathbf{r}}_k = c_{g_n(s)} \mathbf{p}_s^H \widehat{\mathbf{r}}_k. \quad \blacksquare$$

There exist examples where the condition  $\prod_{l=1}^{\nu} \det(\widehat{\mathbf{W}}_l) \det(\widehat{\mathbf{S}}_l) \neq 0$  in Proposition 3.2 holds, as shown below.

LEMMA 3.4. *Consider the case where  $n = 1$ ,  $\widehat{\mathbf{r}}_0 \in \mathcal{R}^N$ ,  $\widehat{\mathbf{r}}_0 \neq \mathbf{0}$  and  $\mathbf{A} \in \mathcal{R}^{N \times N}$  is nonsingular. If  $\mathbf{q}_1 \in \mathcal{R}^N$  is a random vector with independent and identically distributed elements from  $N(0, 1)$ , the normal distribution with mean 0 and variance 1, then  $\text{Prob}\left(\prod_{l=1}^{\nu} \det(\widehat{\mathbf{W}}_l) \det(\widehat{\mathbf{S}}_l) = 0\right) = 0$ .*

<sup>4</sup>We say that  $\mathbf{u} \perp \mathbf{v}$  if  $\mathbf{u}^H \mathbf{v} = 0$ .

*Proof.* Since  $\mathbf{p}_k = \mathbf{A}^{g_n(k)} \mathbf{q}_{r_n(k)} = \mathbf{A}^{k-1} \mathbf{q}_1$  when  $n = 1$ , both  $\widehat{\mathbf{S}}_\nu$  and  $\widehat{\mathbf{W}}_\nu$  are Hankel matrices

$$\widehat{\mathbf{S}}_\nu = \begin{bmatrix} \widehat{s}_1 & \widehat{s}_2 & \cdots & \widehat{s}_\nu \\ \widehat{s}_2 & \widehat{s}_3 & \cdots & \widehat{s}_{\nu+1} \\ \cdots & \cdots & \cdots & \cdots \\ \widehat{s}_\nu & \widehat{s}_{\nu+1} & \cdots & \widehat{s}_{2\nu-1} \end{bmatrix}, \quad \widehat{\mathbf{W}}_\nu = \begin{bmatrix} \widehat{w}_1 & \widehat{w}_2 & \cdots & \widehat{w}_\nu \\ \widehat{w}_2 & \widehat{w}_3 & \cdots & \widehat{w}_{\nu+1} \\ \cdots & \cdots & \cdots & \cdots \\ \widehat{w}_\nu & \widehat{w}_{\nu+1} & \cdots & \widehat{w}_{2\nu-1} \end{bmatrix}$$

where  $\widehat{s}_t = \mathbf{q}_1^T \mathbf{A}^t \widehat{\mathbf{r}}_0$  and  $\widehat{w}_t = \mathbf{q}_1^T \mathbf{A}^{t-1} \widehat{\mathbf{r}}_0$  for  $t = 1, 2, \dots, 2\nu - 1$ .

We first prove

$$(3.3) \quad \text{Prob} \left( \det(\widehat{\mathbf{W}}_l) = 0 \right) = 0$$

for any fixed  $l$  with  $1 \leq l \leq \nu$ . It is trivial that (3.3) holds when  $l = 1$  and we therefore assume  $l \geq 2$  in the following discussion.

By assumption,  $\nu$  is the degree of the minimal polynomial of  $\widehat{\mathbf{r}}_0$  with respect to  $\mathbf{A}$ . This implies that  $\mathcal{K} = \text{span}\{\mathbf{A}^t \widehat{\mathbf{r}}_0 \mid t \in \mathcal{N}_0\}$  is a  $\nu$ -dimensional space with  $\{\widehat{\mathbf{r}}_0, \mathbf{A}\widehat{\mathbf{r}}_0, \dots, \mathbf{A}^{\nu-1}\widehat{\mathbf{r}}_0\}$  as a basis. Since  $\mathbf{A}$  is nonsingular,  $\{\mathbf{A}^{l-1}\widehat{\mathbf{r}}_0, \mathbf{A}^l\widehat{\mathbf{r}}_0, \dots, \mathbf{A}^{l+\nu-2}\widehat{\mathbf{r}}_0\}$  is another basis of  $\mathcal{K}$ .

Perform an orthogonal factorization of the  $N \times \nu$  matrix

$$[\mathbf{A}^{l-1}\widehat{\mathbf{r}}_0, \mathbf{A}^l\widehat{\mathbf{r}}_0, \dots, \mathbf{A}^{l+\nu-2}\widehat{\mathbf{r}}_0] = \mathbf{Q}\mathbf{R}$$

where  $\mathbf{Q} \in \mathcal{R}^{N \times N}$  is orthogonal and  $\mathbf{R} \in \mathcal{R}^{N \times \nu}$  is upper triangular with positive main diagonal elements  $r_{11}, r_{22}, \dots, r_{\nu\nu}$ . Clearly, the first  $\nu$  columns of  $\mathbf{Q}$  form a basis of  $\mathcal{K}$  and the last  $N - \nu$  columns belong to  $\mathcal{K}^\perp$ .

Write

$$\begin{aligned} \mathbf{A}^{l-2}\widehat{\mathbf{r}}_0 &= \xi_1 \mathbf{A}^{l-1}\widehat{\mathbf{r}}_0 + \xi_2 \mathbf{A}^l\widehat{\mathbf{r}}_0 + \cdots + \xi_\nu \mathbf{A}^{l+\nu-2}\widehat{\mathbf{r}}_0 \\ &= [\mathbf{A}^{l-1}\widehat{\mathbf{r}}_0, \mathbf{A}^l\widehat{\mathbf{r}}_0, \dots, \mathbf{A}^{l+\nu-2}\widehat{\mathbf{r}}_0] \xi \\ &= \mathbf{Q}\mathbf{R}\xi \equiv \mathbf{Q}\eta \end{aligned}$$

for some scalars  $\xi_1, \xi_2, \dots, \xi_\nu \in \mathcal{R}$ , where  $\xi = [\xi_1, \xi_2, \dots, \xi_\nu]^T \in \mathcal{R}^\nu$  and  $\eta = [\eta_1, \eta_2, \dots, \eta_N]^T = \mathbf{R}\xi \in \mathcal{R}^N$ . Since  $\mathbf{A}$  is nonsingular and  $\{\widehat{\mathbf{r}}_0, \mathbf{A}\widehat{\mathbf{r}}_0, \dots, \mathbf{A}^{\nu-1}\widehat{\mathbf{r}}_0\}$  linearly independent, we have  $\xi_\nu \neq 0$  and hence  $\eta_\nu = r_{\nu\nu}\xi_\nu \neq 0$ . Let  $\theta = [\theta_1, \theta_2, \dots, \theta_N]^T = \mathbf{Q}^T \mathbf{q}_1$ . Then  $\theta$  is a random vector with iid elements from  $N(0, 1)$ [4]. We now express  $\det(\widehat{\mathbf{W}}_l)$  in terms of the elements of  $\theta$ . Let us write

$$\begin{aligned} &[\widehat{w}_1, \widehat{w}_2, \dots, \widehat{w}_{l-2}, \widehat{w}_{l-1}, \widehat{w}_l, \dots, \widehat{w}_{2l-1}] \\ &= \mathbf{q}_1^T [\widehat{\mathbf{r}}_0, \mathbf{A}\widehat{\mathbf{r}}_0, \dots, \mathbf{A}^{l-3}\widehat{\mathbf{r}}_0, \mathbf{A}^{l-2}\widehat{\mathbf{r}}_0, \mathbf{A}^{l-1}\widehat{\mathbf{r}}_0, \dots, \mathbf{A}^{2l-2}\widehat{\mathbf{r}}_0] \\ &= \mathbf{q}_1^T [\widehat{\mathbf{r}}_0, \mathbf{A}\widehat{\mathbf{r}}_0, \dots, \mathbf{A}^{l-3}\widehat{\mathbf{r}}_0, \mathbf{Q}\eta, \mathbf{Q}\mathbf{R}^{(l)}] \\ &= \mathbf{q}_1^T \mathbf{Q} [\mathbf{Q}^T [\widehat{\mathbf{r}}_0, \mathbf{A}\widehat{\mathbf{r}}_0, \dots, \mathbf{A}^{l-3}\widehat{\mathbf{r}}_0], \eta, \mathbf{R}^{(l)}] \\ &= \theta^T [\mathbf{Q}^T [\widehat{\mathbf{r}}_0, \mathbf{A}\widehat{\mathbf{r}}_0, \dots, \mathbf{A}^{l-3}\widehat{\mathbf{r}}_0], \eta, \mathbf{R}^{(l)}] \end{aligned}$$

where  $\mathbf{R}^{(l)}$  denotes the matrix consisting of the first  $l$  columns of  $\mathbf{R}$ . Since the last  $N - \nu$  columns of  $\mathbf{Q}$  belong to  $\mathcal{K}^\perp$ , the last  $N - \nu$  rows of the matrix  $\mathbf{Q}^T [\widehat{\mathbf{r}}_0, \mathbf{A}\widehat{\mathbf{r}}_0, \dots, \mathbf{A}^{l-3}\widehat{\mathbf{r}}_0]$  are zeros. Similarly, the last  $N - \nu$  elements of  $\eta = \mathbf{R}\xi$  are zeros because the last  $N - \nu$  rows of  $\mathbf{R}$  are zeros. We therefore have

$$\widehat{w}_t = \begin{cases} \text{a linear combination of } \theta_1, \theta_2, \dots, \theta_\nu & \text{if } 1 \leq t \leq l-2, \\ \eta_1 \theta_1 + \eta_2 \theta_2 + \cdots + \eta_\nu \theta_\nu & \text{with } \eta_\nu \neq 0 \text{ if } t = l-1, \\ r_{1,t-l+1} \theta_1 + r_{2,t-l+1} \theta_2 + \cdots + r_{t-l+1,t-l+1} \theta_{t-l+1} \\ & \text{with } r_{t-l+1,t-l+1} \neq 0 \text{ if } l \leq t \leq 2l-1. \end{cases}$$

This shows that none of the random variables  $\theta_{\nu+1}, \theta_{\nu+2}, \dots, \theta_N$  is involved in any of the  $\widehat{w}$ 's. In more details, when  $l < \nu$ ,  $\widehat{w}_t = \widehat{w}_t(\theta_1, \theta_2, \dots, \theta_\nu)$  if  $1 \leq t \leq l-1$  and  $\widehat{w}_t = \widehat{w}_t(\theta_1, \theta_2, \dots, \theta_{\nu-1})$  if  $l \leq t \leq 2l-1$ ; when  $l = \nu$ ,  $\widehat{w}_t = \widehat{w}_t(\theta_1, \theta_2, \dots, \theta_\nu)$  if  $1 \leq t \leq \nu-1$  or  $t = 2\nu-1$ , and  $\widehat{w}_t = \widehat{w}_t(\theta_1, \theta_2, \dots, \theta_{\nu-1})$  if  $\nu \leq t < 2\nu-1$ .

We now expand  $\det(\widehat{\mathbf{W}}_l)$  by minors down its last column and write it into a polynomial of  $\theta_\nu$ . This yields

$$\begin{aligned} & (-1)^{\frac{1}{2}l(l+1)+1} \det(\widehat{\mathbf{W}}_l) \\ &= \widehat{w}_{2l-1} \widehat{w}_{l-1}^{l-1} + \dots \\ &= \begin{cases} (\sum_{s=1}^l r_{sl} \theta_s) \eta_\nu^{l-1} \theta_\nu^{l-1} + c_{l-2} \theta_\nu^{l-2} + \dots + c_1 \theta_\nu + c_0 & \text{if } 2 \leq l < \nu, \\ r_{\nu\nu} \eta_\nu^{\nu-1} \theta_\nu^\nu + d_{\nu-1} \theta_\nu^{\nu-1} + \dots + d_1 \theta_\nu + d_0 & \text{if } l = \nu \end{cases} \end{aligned}$$

where the coefficients  $c_0, \dots, c_{l-2}$  and  $d_0, \dots, d_{\nu-1}$  are polynomials in  $\theta_1, \theta_2, \dots, \theta_{\nu-1}$ . Now (3.3) follows from the facts that  $r_{ll} \neq 0, r_{\nu\nu} \neq 0, \eta_\nu \neq 0$  and  $\theta_1, \theta_2, \dots, \theta_\nu$  are independent random variables.

Note that  $\nu$  is also the degree of the minimal polynomial of  $\mathbf{A}\widehat{\mathbf{r}}_0$  with respect to  $\mathbf{A}$  when  $\mathbf{A}$  is nonsingular. With  $\widehat{\mathbf{r}}_0$  replaced by  $\mathbf{A}\widehat{\mathbf{r}}_0$  in (3.3), we then have

$$(3.4) \quad \text{Prob}(\det(\widehat{\mathbf{S}}_l) = 0) = 0$$

for any  $l$  with  $1 \leq l \leq \nu$ .

Now, (3.3) and (3.4) together imply that

$$\begin{aligned} & \text{Prob}\left(\prod_{l=1}^\nu \det(\widehat{\mathbf{W}}_l) \det(\widehat{\mathbf{S}}_l) = 0\right) \\ & \leq \sum_{l=1}^\nu \text{Prob}(\det(\widehat{\mathbf{W}}_l) = 0) + \sum_{l=1}^\nu \text{Prob}(\det(\widehat{\mathbf{S}}_l) = 0) = 0 \end{aligned}$$

and the lemma is proved.  $\blacksquare$

**THEOREM 3.5.** *Consider the case where  $n = 1, \widehat{\mathbf{r}}_0 \in \mathcal{R}^N, \widehat{\mathbf{r}}_0 \neq \mathbf{0}$  and  $\mathbf{A} \in \mathcal{R}^{N \times N}$ . If  $\mathbf{q}_1 \in \mathcal{R}^N$  is a random vector with independent and identically distributed elements from  $N(0, 1)$ , then  $\text{Prob}\left(\prod_{l=1}^\nu \det(\widehat{\mathbf{W}}_l) \det(\widehat{\mathbf{S}}_l) = 0\right) = 0$  if and only if  $p_{\min}(0, \mathbf{A}, \widehat{\mathbf{r}}_0) \neq 0$ .*

*Proof.* If  $p_{\min}(0, \mathbf{A}, \widehat{\mathbf{r}}_0) = 0$ , then  $\mathbf{A}^\nu \widehat{\mathbf{r}}_0$  is a linear combination of  $\mathbf{A}\widehat{\mathbf{r}}_0, \dots, \mathbf{A}^{\nu-1} \widehat{\mathbf{r}}_0$  or  $\mathbf{A}^\nu \widehat{\mathbf{r}}_0 = \mathbf{0}$  in the case when  $\nu = 1$ . Hence  $\det(\widehat{\mathbf{S}}_\nu) = 0$  no matter what  $\mathbf{q}_1$  is and therefore the probability of  $\prod_{l=1}^\nu \det(\widehat{\mathbf{W}}_l) \det(\widehat{\mathbf{S}}_l) = 0$  is 1.

We now suppose  $p_{\min}(0, \mathbf{A}, \widehat{\mathbf{r}}_0) \neq 0$ . By the real version of the Schur's unitary triangularization theorem (see, for instance, [13]),  $\mathbf{A}$  can be decomposed as

$$\mathbf{A} = \mathbf{Q}^T \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{0} & \mathbf{B}_{22} \end{bmatrix} \mathbf{Q} \equiv \mathbf{Q}^T \mathbf{B} \mathbf{Q}$$

where  $\mathbf{Q} \in \mathcal{R}^{N \times N}$  orthogonal,  $\mathbf{B}_{11} \in \mathcal{R}^{N_1 \times N_1}$  nonsingular and  $\mathbf{B}_{22} \in \mathcal{R}^{N_2 \times N_2}$  strictly upper triangular (namely, an upper triangular matrix with its main diagonal elements zero). Let  $\widetilde{\mathbf{r}}_0 = \mathbf{Q} \widehat{\mathbf{r}}_0 \equiv [\widetilde{\mathbf{r}}_{01}^T, \widetilde{\mathbf{r}}_{02}^T]^T$  where  $\widetilde{\mathbf{r}}_{01} \in \mathcal{R}^{N_1}$  and  $\widetilde{\mathbf{r}}_{02} \in \mathcal{R}^{N_2}$ . Then  $p_{\min}(\mathbf{B}, \mathbf{A}, \widehat{\mathbf{r}}_0) \widetilde{\mathbf{r}}_0 = \mathbf{Q} p_{\min}(\mathbf{A}, \mathbf{A}, \widehat{\mathbf{r}}_0) \widehat{\mathbf{r}}_0 = \mathbf{0}$ . Note that

$$(3.5) \quad \mathbf{B}^k = \begin{bmatrix} \mathbf{B}_{11}^k & * \\ \mathbf{0} & \mathbf{B}_{22}^k \end{bmatrix}$$

for  $k \in \mathcal{N}$ , we have

$$p_{\min}(\mathbf{B}, \mathbf{A}, \hat{\mathbf{r}}_0) = \begin{bmatrix} p_{\min}(\mathbf{B}_{11}, \mathbf{A}, \hat{\mathbf{r}}_0) & * \\ \mathbf{0} & p_{\min}(\mathbf{B}_{22}, \mathbf{A}, \hat{\mathbf{r}}_0) \end{bmatrix}.$$

Thus,  $p_{\min}(\mathbf{B}, \mathbf{A}, \hat{\mathbf{r}}_0)\tilde{\mathbf{r}}_0 = \mathbf{0}$  implies that  $p_{\min}(\mathbf{B}_{22}, \mathbf{A}, \hat{\mathbf{r}}_0)\tilde{\mathbf{r}}_{02} = \mathbf{0}$ . If we write  $p_{\min}(\lambda, \mathbf{A}, \hat{\mathbf{r}}_0) = \sum_{i=0}^{\nu} c_i \lambda^i$  with  $c_0 \neq 0$ , then  $\sum_{i=1}^{\nu} c_i \mathbf{B}_{22}^i$  is a strictly upper triangular matrix. Thus,  $p_{\min}(\mathbf{B}_{22}, \mathbf{A}, \hat{\mathbf{r}}_0) = (\sum_{i=1}^{\nu} c_i \mathbf{B}_{22}^i) + c_0 \mathbf{I}$  is an upper triangular matrix whose main diagonal elements are  $c_0$ . So,  $p_{\min}(\mathbf{B}_{22}, \mathbf{A}, \hat{\mathbf{r}}_0)$  is nonsingular and therefore  $p_{\min}(\mathbf{B}_{22}, \mathbf{A}, \hat{\mathbf{r}}_0)\tilde{\mathbf{r}}_{02} = \mathbf{0}$  yields  $\tilde{\mathbf{r}}_{02} = \mathbf{0}$ . Since  $\tilde{\mathbf{r}}_0 \neq \mathbf{0}$  due to  $\hat{\mathbf{r}}_0 \neq \mathbf{0}$  by assumption,  $\tilde{\mathbf{r}}_{02} \neq \tilde{\mathbf{r}}_0$ . In other words,  $N_2 < N$  or  $\mathbf{B}_{11}$  is not a null matrix.

Now that  $\tilde{\mathbf{r}}_{02} = \mathbf{0}$ , (3.5) implies that

$$(3.6) \quad \mathbf{B}^k \tilde{\mathbf{r}}_0 = \begin{bmatrix} \mathbf{B}_{11}^k \tilde{\mathbf{r}}_{01} \\ \mathbf{0} \end{bmatrix}$$

for  $k \in \mathcal{N}$ . Therefore,  $p(\mathbf{B})\tilde{\mathbf{r}}_0 = [(p(\mathbf{B}_{11})\tilde{\mathbf{r}}_{01})^T, \mathbf{0}^T]^T$  for any polynomial  $p(\lambda)$ . Thus, the minimal polynomial of  $\tilde{\mathbf{r}}_0$  with respect to  $\mathbf{B}$  is equal to the minimal polynomial of  $\tilde{\mathbf{r}}_{01}$  with respect to  $\mathbf{B}_{11}$ . This implies that,  $\nu$ , the degree of the minimal polynomial of  $\hat{\mathbf{r}}_0$  with respect to  $\mathbf{A}$ , is also the degree of the minimal polynomial of  $\tilde{\mathbf{r}}_{01}$  with respect to  $\mathbf{B}_{11}$ .

We now set  $\theta = \mathbf{Q} \mathbf{q}_1 \equiv [\theta_1^T, \theta_2^T]^T$  where  $\theta_1 \in \mathcal{R}^{N_1}$  and  $\theta_2 \in \mathcal{R}^{N_2}$ . Since  $\mathbf{q}_1$  is random with iid elements from  $N(0, 1)$ , so is  $\theta$ . By (3.6),

$$\mathbf{q}_1^T \mathbf{A}^k \hat{\mathbf{r}}_0 = \theta^T \mathbf{B}^k \tilde{\mathbf{r}}_0 = \theta_1^T \mathbf{B}_{11}^k \tilde{\mathbf{r}}_{01} \quad \text{and} \quad \mathbf{q}_1^T \hat{\mathbf{r}}_0 = \theta_1^T \tilde{\mathbf{r}}_{01}$$

where  $k \in \mathcal{N}$ . Thus

$$\hat{\mathbf{S}}_{\nu}(\mathbf{A}, \hat{\mathbf{r}}_0, \mathbf{q}_1) = \hat{\mathbf{S}}_{\nu}(\mathbf{B}_{11}, \tilde{\mathbf{r}}_{01}, \theta_1) \quad \text{and} \quad \hat{\mathbf{W}}_{\nu}(\mathbf{A}, \hat{\mathbf{r}}_0, \mathbf{q}_1) = \hat{\mathbf{W}}_{\nu}(\mathbf{B}_{11}, \tilde{\mathbf{r}}_{01}, \theta_1).$$

Now, the desired probability follows from Lemma 3.4 because  $\mathbf{B}_{11}$  is nonsingular,  $\theta_1$  is iid  $N(0, 1)$  random and  $\nu$  is the degree of the minimal polynomial of  $\tilde{\mathbf{r}}_{01}$  with respect to  $\mathbf{B}_{11}$ .  $\blacksquare$

Extension of the theorem to the general case should be possible, namely,  $n \geq 1$ ,  $\mathbf{A} \in \mathcal{C}^{N \times N}$ ,  $\hat{\mathbf{r}}_0 \in \mathcal{C}^{N \times 1}$  and  $[\mathbf{q}_1, \dots, \mathbf{q}_n]$  is a Gaussian matrix.

Even though the probability of a breakdown is zero in theory, it is still possible that ML( $n$ )BiCG encounters a breakdown or a near-breakdown in practice, especially when the  $\mathbf{q}$ 's are deterministic. Techniques for curing breakdowns in the contexts of the standard Lanczos process and the multiple starting Lanczos processes can be found in [2, 7, 10, 17, 20, 21] and [1, 6] respectively and the references therein.

**4. A ML( $n$ )BiCGStab Algorithm.** An algorithm of the ML( $n$ )BiCGStab method has been derived in [35] (Algorithm 2 without preconditioning and Algorithm 3 with preconditioning in [35]), but the derivation there is complicated and less inspiring. In this section, we re-derive the algorithm in a more systematic fashion with the help of index functions.

**4.1. Notation and Definitions.** Let  $\phi_k(\lambda)$  be the polynomial of degree  $k$  defined by the recurrence relation

$$(4.1) \quad \begin{aligned} \phi_0(\lambda) &= 1, \\ \phi_k(\lambda) &= (\rho_k \lambda + 1) \phi_{k-1}(\lambda), \quad k = 1, 2, \dots, \end{aligned}$$

where  $\rho_k$  is a free parameter. These polynomials are used by BiCGStab as smoothing polynomials. If expressed in terms of the power basis

$$\phi_k(\lambda) = c_k^{(k)} \lambda^k + \dots + c_1^{(k)} \lambda + c_0^{(k)},$$

it is clear that  $c_k^{(k)} = \rho_1 \rho_2 \dots \rho_k$  and  $c_0^{(k)} = 1$ . Thus,

$$(4.2) \quad c_k^{(k)} = \rho_k c_{k-1}^{(k-1)}.$$

For  $k \in \mathcal{N}$ , define

$$(4.3) \quad \begin{aligned} \mathbf{r}_k &= \phi_{g_n(k)+1}(\mathbf{A}) \widehat{\mathbf{r}}_k, & \mathbf{u}_k &= \phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{r}}_k, \\ \mathbf{g}_k &= \phi_{g_n(k)+1}(\mathbf{A}) \widehat{\mathbf{g}}_k, & \mathbf{d}_k &= \rho_{g_n(k)+1} \mathbf{A} \phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{g}}_k, \\ \mathbf{w}_k &= \mathbf{A} \mathbf{g}_k \end{aligned}$$

and for  $k = 0$ , set

$$(4.4) \quad \mathbf{r}_0 = \widehat{\mathbf{r}}_0 \quad \text{and} \quad \mathbf{g}_0 = \widehat{\mathbf{g}}_0.$$

The vectors  $\mathbf{r}_k$  will be the residual vectors of the approximate solutions  $\mathbf{x}_k$  computed in the following ML( $n$ )BiCGStab algorithm.

**4.2. Algorithm Derivation.** The derivation of a ML( $n$ )BiCGStab algorithm depends on Proposition 2.1 and Corollary 3.3. In order to exploit Proposition 2.1, we divide the operations in Algorithm 3.1 (forgetting Lines 1, 2, 5 and 11) into cases, basically corresponding to those described in the proposition:

**Derivation Version (DV) #1.**

1. For  $k = 1, 2, \dots$ , until convergence:
  2. If  $r_n(k) = 1$
  3.  $\alpha_k = \mathbf{p}_k^H \widehat{\mathbf{r}}_{k-1} / \mathbf{p}_k^H \mathbf{A} \widehat{\mathbf{g}}_{k-1}$ ;
  4.  $\widehat{\mathbf{r}}_k = \widehat{\mathbf{r}}_{k-1} - \alpha_k \mathbf{A} \widehat{\mathbf{g}}_{k-1}$ ;
  5. Else
  6.  $\alpha_k = \mathbf{p}_k^H \widehat{\mathbf{r}}_{k-1} / \mathbf{p}_k^H \mathbf{A} \widehat{\mathbf{g}}_{k-1}$ ;
  7.  $\widehat{\mathbf{r}}_k = \widehat{\mathbf{r}}_{k-1} - \alpha_k \mathbf{A} \widehat{\mathbf{g}}_{k-1}$ ;
  8. End
  9. If  $r_n(k) < n$
  10. For  $s = \max(k - n, 0), \dots, g_n(k)n - 1$
  11.  $\beta_s^{(k)} = -\mathbf{p}_{s+1}^H \mathbf{A} \left( \widehat{\mathbf{r}}_k + \sum_{t=\max(k-n,0)}^{s-1} \beta_t^{(k)} \widehat{\mathbf{g}}_t \right) / \mathbf{p}_{s+1}^H \mathbf{A} \widehat{\mathbf{g}}_s$ ;
  12. End
  13.  $\beta_{g_n(k)n}^{(k)} = -\mathbf{p}_{g_n(k)n+1}^H \mathbf{A} \left( \widehat{\mathbf{r}}_k + \sum_{t=\max(k-n,0)}^{g_n(k)n-1} \beta_t^{(k)} \widehat{\mathbf{g}}_t \right) / \mathbf{p}_{g_n(k)n+1}^H \mathbf{A} \widehat{\mathbf{g}}_{g_n(k)n}$ ;
  14. For  $s = g_n(k)n + 1, \dots, k - 1$
  15.  $\beta_s^{(k)} = -\mathbf{p}_{s+1}^H \mathbf{A} \left( \widehat{\mathbf{r}}_k + \sum_{t=\max(k-n,0)}^{g_n(k)n} \beta_t^{(k)} \widehat{\mathbf{g}}_t + \sum_{t=g_n(k)n+1}^{s-1} \beta_t^{(k)} \widehat{\mathbf{g}}_t \right) / \mathbf{p}_{s+1}^H \mathbf{A} \widehat{\mathbf{g}}_s$ ;
  16. End
  17.  $\widehat{\mathbf{g}}_k = \widehat{\mathbf{r}}_k + \sum_{s=\max(k-n,0)}^{g_n(k)n} \beta_s^{(k)} \widehat{\mathbf{g}}_s + \sum_{s=g_n(k)n+1}^{k-1} \beta_s^{(k)} \widehat{\mathbf{g}}_s$ ;
  18. Else
  19.  $\beta_{g_n(k)n}^{(k)} = -\mathbf{p}_{g_n(k)n+1}^H \mathbf{A} \widehat{\mathbf{r}}_k / \mathbf{p}_{g_n(k)n+1}^H \mathbf{A} \widehat{\mathbf{g}}_{g_n(k)n}$ ;
  20. For  $s = g_n(k)n + 1, \dots, k - 1$
  21.  $\beta_s^{(k)} = -\mathbf{p}_{s+1}^H \mathbf{A} \left( \widehat{\mathbf{r}}_k + \beta_{g_n(k)n}^{(k)} \widehat{\mathbf{g}}_{g_n(k)n} + \sum_{t=g_n(k)n+1}^{s-1} \beta_t^{(k)} \widehat{\mathbf{g}}_t \right) / \mathbf{p}_{s+1}^H \mathbf{A} \widehat{\mathbf{g}}_s$ ;

22. End  
 23.  $\widehat{\mathbf{g}}_k = \widehat{\mathbf{r}}_k + \beta_{g_n(k)n}^{(k)} \widehat{\mathbf{g}}_{g_n(k)n} + \sum_{s=g_n(k)n+1}^{k-1} \beta_s^{(k)} \widehat{\mathbf{g}}_s;$   
 24. End  
 25. End

We have adopted the following conventions in DV#1:

- (i) About a *for* loop of the form:  
 For  $s = a, \dots, b$ , Statements; End  
 if  $b < a$ , the statements within the loop are not implemented.  
 (ii) About a *sum* of the form  $\sum_{s=a}^b term_s$ , if  $b < a$ , this sum is considered to be 0.

These conventions will also be applied to the DVs and algorithms in the sequel.

Now, by Corollary 3.3, (4.2), (3.2) and Proposition 2.1(a), DV#1 can be transformed into the version below.

### Derivation Version #2.

1. For  $k = 1, 2, \dots$ , until convergence:
2. If  $r_n(k) = 1$
3.  $\alpha_k = \mathbf{q}_{r_n(k)}^H \phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{r}}_{k-1} / \mathbf{q}_{r_n(k)}^H \mathbf{A} \phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{g}}_{k-1};$
4.  $\phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{r}}_k = \phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{r}}_{k-1} - \alpha_k \mathbf{A} \phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{g}}_{k-1};$
5.  $\phi_{g_n(k)+1}(\mathbf{A}) \widehat{\mathbf{r}}_k = (\rho_{g_n(k)+1} \mathbf{A} + \mathbf{I}) \phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{r}}_k;$
6. Else
7.  $\alpha_k = \mathbf{q}_{r_n(k)}^H \phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{r}}_{k-1} / \mathbf{q}_{r_n(k)}^H \mathbf{A} \phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{g}}_{k-1};$
8.  $\phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{r}}_k = \phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{r}}_{k-1} - \alpha_k \mathbf{A} \phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{g}}_{k-1};$
9.  $\phi_{g_n(k)+1}(\mathbf{A}) \widehat{\mathbf{r}}_k = \phi_{g_n(k)+1}(\mathbf{A}) \widehat{\mathbf{r}}_{k-1} - \alpha_k \mathbf{A} \phi_{g_n(k)+1}(\mathbf{A}) \widehat{\mathbf{g}}_{k-1};$
10. End
11. If  $r_n(k) < n$
12. For  $s = \max(k-n, 0), \dots, g_n(k)n-1$
13.  $\beta_s^{(k)} = -\mathbf{q}_{r_n(s+1)}^H \left( \phi_{g_n(s+1)+1}(\mathbf{A}) \widehat{\mathbf{r}}_k + \sum_{t=\max(k-n, 0)}^{s-1} \beta_t^{(k)} \rho_{g_n(s+1)+1} \mathbf{A} \phi_{g_n(s+1)}(\mathbf{A}) \widehat{\mathbf{g}}_t \right) / \rho_{g_n(s+1)+1} \mathbf{q}_{r_n(s+1)}^H \mathbf{A} \phi_{g_n(s+1)}(\mathbf{A}) \widehat{\mathbf{g}}_s;$
14. End
15.  $\beta_{g_n(k)n}^{(k)} = -\mathbf{q}_1^H \left( \phi_{g_n(k)+1}(\mathbf{A}) \widehat{\mathbf{r}}_k + \sum_{t=\max(k-n, 0)}^{g_n(k)n-1} \beta_t^{(k)} \rho_{g_n(k)+1} \mathbf{A} \phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{g}}_t \right) / \rho_{g_n(k)+1} \mathbf{q}_1^H \mathbf{A} \phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{g}}_{g_n(k)n};$
16. For  $s = g_n(k)n+1, \dots, k-1$
17.  $\beta_s^{(k)} = -\mathbf{q}_{r_n(s+1)}^H \left( \phi_{g_n(s+1)+1}(\mathbf{A}) \widehat{\mathbf{r}}_k + \sum_{t=\max(k-n, 0)}^{g_n(k)n} \beta_t^{(k)} \rho_{g_n(s+1)+1} \mathbf{A} \phi_{g_n(s+1)}(\mathbf{A}) \widehat{\mathbf{g}}_t + \sum_{t=g_n(k)n+1}^{s-1} \beta_t^{(k)} \rho_{g_n(s+1)+1} \mathbf{A} \phi_{g_n(s+1)}(\mathbf{A}) \widehat{\mathbf{g}}_t \right) / \rho_{g_n(s+1)+1} \mathbf{q}_{r_n(s+1)}^H \mathbf{A} \phi_{g_n(s+1)}(\mathbf{A}) \widehat{\mathbf{g}}_s;$
18. End
19.  $\rho_{g_n(k)+1} \mathbf{A} \phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{g}}_k = \rho_{g_n(k)+1} \mathbf{A} \phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{r}}_k + \sum_{s=\max(k-n, 0)}^{g_n(k)n} \beta_s^{(k)} \rho_{g_n(k)+1} \mathbf{A} \phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{g}}_s + \sum_{s=g_n(k)n+1}^{k-1} \beta_s^{(k)} \rho_{g_n(k)+1} \mathbf{A} \phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{g}}_s;$
20.  $\phi_{g_n(k)+1}(\mathbf{A}) \widehat{\mathbf{g}}_k = \phi_{g_n(k)+1}(\mathbf{A}) \widehat{\mathbf{r}}_k + \sum_{s=\max(k-n, 0)}^{g_n(k)n} \beta_s^{(k)} \phi_{g_n(k)+1}(\mathbf{A}) \widehat{\mathbf{g}}_s + \sum_{s=g_n(k)n+1}^{k-1} \beta_s^{(k)} \phi_{g_n(k)+1}(\mathbf{A}) \widehat{\mathbf{g}}_s;$
21. Else
22.  $\beta_{g_n(k)n}^{(k)} = -\mathbf{q}_1^H \phi_{g_n(k)+1}(\mathbf{A}) \widehat{\mathbf{r}}_k / \rho_{g_n(k)+1} \mathbf{q}_1^H \mathbf{A} \phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{g}}_{g_n(k)n};$
23. For  $s = g_n(k)n+1, \dots, k-1$

24. 
$$\beta_s^{(k)} = -\mathbf{q}_{r_n(s+1)}^H \left( \phi_{g_n(s+1)+1}(\mathbf{A})\widehat{\mathbf{r}}_k + \beta_{g_n(k)n}^{(k)} \rho_{g_n(s+1)+1} \mathbf{A} \phi_{g_n(s+1)}(\mathbf{A})\widehat{\mathbf{g}}_{g_n(k)n} + \sum_{t=g_n(k)n+1}^{s-1} \beta_t^{(k)} \rho_{g_n(s+1)+1} \mathbf{A} \phi_{g_n(s+1)}(\mathbf{A})\widehat{\mathbf{g}}_t \right) / \rho_{g_n(s+1)+1} \mathbf{q}_{r_n(s+1)}^H \mathbf{A} \phi_{g_n(s+1)}(\mathbf{A})\widehat{\mathbf{g}}_s;$$
25. End
26. 
$$\phi_{g_n(k)+1}(\mathbf{A})\widehat{\mathbf{g}}_k = \phi_{g_n(k)+1}(\mathbf{A})\widehat{\mathbf{r}}_k + \beta_{g_n(k)n}^{(k)} \phi_{g_n(k)+1}(\mathbf{A})\widehat{\mathbf{g}}_{g_n(k)n} + \sum_{s=g_n(k)n+1}^{k-1} \beta_s^{(k)} \phi_{g_n(k)+1}(\mathbf{A})\widehat{\mathbf{g}}_s;$$
27. End
28. End

Lines 4, 8, 9, 19, 20 and 26, DV#2, were obtained from Lines 4, 7, 17 and 23, DV#1, by multiplying them with  $\phi_{g_n(k)}(\mathbf{A})$ ,  $\phi_{g_n(k)+1}(\mathbf{A})$  and  $\rho_{g_n(k)+1} \mathbf{A} \phi_{g_n(k)}(\mathbf{A})$  respectively. Line 5, DV#2, is a direct result of the definition (4.1) of  $\phi$ .

To help better understand how DV#1 arrived at DV#2, let us demonstrate the transformation of the term  $\mathbf{p}_{g_n(k)n+1}^H \mathbf{A} \widehat{\mathbf{r}}_k$  on Line 13, DV#1, into the term  $\mathbf{q}_1^H \phi_{g_n(k)+1}(\mathbf{A})\widehat{\mathbf{r}}_k$  on Line 15, DV#2, as follows.

By (3.2) and Proposition 2.1(a), we have

$$\begin{aligned} \mathbf{p}_{(g_n(k)+1)n+1} &= (\mathbf{A}^H)^{g_n((g_n(k)+1)n+1)} \mathbf{q}_{r_n((g_n(k)+1)n+1)} \\ &= (\mathbf{A}^H)^{g_n(g_n(k)n+1)+1} \mathbf{q}_{r_n(g_n(k)n+1)} \\ &= \mathbf{A}^H \mathbf{p}_{g_n(k)n+1}. \end{aligned}$$

Hence  $\mathbf{p}_{g_n(k)n+1}^H \mathbf{A} \widehat{\mathbf{r}}_k = \mathbf{p}_{(g_n(k)+1)n+1}^H \widehat{\mathbf{r}}_k$ . Since  $(g_n(k)+1)n+1 \leq k+n$ , an application of Corollary 3.3 to  $\mathbf{p}_{(g_n(k)+1)n+1}^H \widehat{\mathbf{r}}_k$  with  $\psi = \phi$  ( $\phi$  is defined by (4.1)) then yields

$$\begin{aligned} \mathbf{p}_{g_n(k)n+1}^H \mathbf{A} \widehat{\mathbf{r}}_k &= \frac{1}{c_{g_n((g_n(k)+1)n+1)}^{(g_n(k)+1)n+1}} \mathbf{q}_{r_n((g_n(k)+1)n+1)}^H \phi_{g_n((g_n(k)+1)n+1)}(\mathbf{A})\widehat{\mathbf{r}}_k \\ &= \frac{1}{c_{g_n(k)+1}^{(g_n(k)+1)}} \mathbf{q}_1^H \phi_{g_n(k)+1}(\mathbf{A})\widehat{\mathbf{r}}_k. \end{aligned}$$

The second equation above follows from (2.2). The coefficient  $1/c_{g_n(k)+1}^{(g_n(k)+1)}$  is missed from Line 15, DV#2, because it was canceled out by the coefficient from the denominator.

Our goal is to establish updating relations for the quantities introduced in (4.3). To achieve the goal, we further transform DV#2 into the following version. This time, we work on the index function  $g_n$  with the help of Proposition 2.1 so that the definitions in (4.3) can be applied.

### Derivation Version #3.

1. For  $k = 1, 2, \dots$ , until convergence:
  2. If  $r_n(k) = 1$ 
    3.  $\alpha_k = \mathbf{q}_{r_n(k)}^H \phi_{g_n(k-1)+1}(\mathbf{A})\widehat{\mathbf{r}}_{k-1} / \mathbf{q}_{r_n(k)}^H \mathbf{A} \phi_{g_n(k-1)+1}(\mathbf{A})\widehat{\mathbf{g}}_{k-1};$
    4.  $\phi_{g_n(k)}(\mathbf{A})\widehat{\mathbf{r}}_k = \phi_{g_n(k-1)+1}(\mathbf{A})\widehat{\mathbf{r}}_{k-1} - \alpha_k \mathbf{A} \phi_{g_n(k-1)+1}(\mathbf{A})\widehat{\mathbf{g}}_{k-1};$
    5.  $\phi_{g_n(k)+1}(\mathbf{A})\widehat{\mathbf{r}}_k = (\rho_{g_n(k)+1} \mathbf{A} + \mathbf{I}) \phi_{g_n(k)}(\mathbf{A})\widehat{\mathbf{r}}_k;$
  6. Else
    7.  $\alpha_k = \mathbf{q}_{r_n(k)}^H \phi_{g_n(k-1)}(\mathbf{A})\widehat{\mathbf{r}}_{k-1} / \mathbf{q}_{r_n(k)}^H \mathbf{A} \phi_{g_n(k-1)}(\mathbf{A})\widehat{\mathbf{g}}_{k-1};$
    8.  $\phi_{g_n(k)}(\mathbf{A})\widehat{\mathbf{r}}_k = \phi_{g_n(k-1)}(\mathbf{A})\widehat{\mathbf{r}}_{k-1} - \alpha_k \mathbf{A} \phi_{g_n(k-1)}(\mathbf{A})\widehat{\mathbf{g}}_{k-1};$
    9.  $\phi_{g_n(k)+1}(\mathbf{A})\widehat{\mathbf{r}}_k = \phi_{g_n(k-1)+1}(\mathbf{A})\widehat{\mathbf{r}}_{k-1} - \alpha_k \mathbf{A} \phi_{g_n(k-1)+1}(\mathbf{A})\widehat{\mathbf{g}}_{k-1};$
  10. End
  11. If  $r_n(k) < n$

12. For  $s = \max(k - n, 0), \dots, g_n(k)n - 1$
13. 
$$\beta_s^{(k)} = -\mathbf{q}_{r_n(s+1)}^H \left( \phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{r}}_k + \sum_{t=\max(k-n,0)}^{s-1} \beta_t^{(k)} \rho_{g_n(t)+1} \mathbf{A} \phi_{g_n(t)}(\mathbf{A}) \widehat{\mathbf{g}}_t \right) / \rho_{g_n(s)+1} \mathbf{q}_{r_n(s+1)}^H \mathbf{A} \phi_{g_n(s)}(\mathbf{A}) \widehat{\mathbf{g}}_s;$$
14. End
15. 
$$\beta_{g_n(k)n}^{(k)} = -\mathbf{q}_1^H \left( \phi_{g_n(k)+1}(\mathbf{A}) \widehat{\mathbf{r}}_k + \sum_{t=\max(k-n,0)}^{g_n(k)n-1} \beta_t^{(k)} \rho_{g_n(k)+1} \mathbf{A} \phi_{g_n(t)+1}(\mathbf{A}) \widehat{\mathbf{g}}_t \right) / \rho_{g_n(k)+1} \mathbf{q}_1^H \mathbf{A} \phi_{g_n(g_n(k)n)+1}(\mathbf{A}) \widehat{\mathbf{g}}_{g_n(k)n};$$
16. For  $s = g_n(k)n + 1, \dots, k - 1$
17. 
$$\beta_s^{(k)} = -\mathbf{q}_{r_n(s+1)}^H \left( \phi_{g_n(k)+1}(\mathbf{A}) \widehat{\mathbf{r}}_k + \sum_{t=\max(k-n,0)}^{g_n(k)n} \beta_t^{(k)} \rho_{g_n(k)+1} \mathbf{A} \phi_{g_n(t)+1}(\mathbf{A}) \widehat{\mathbf{g}}_t + \sum_{t=g_n(k)n+1}^{s-1} \beta_t^{(k)} \rho_{g_n(t)+1} \mathbf{A} \phi_{g_n(t)}(\mathbf{A}) \widehat{\mathbf{g}}_t \right) / \rho_{g_n(s)+1} \mathbf{q}_{r_n(s+1)}^H \mathbf{A} \phi_{g_n(s)}(\mathbf{A}) \widehat{\mathbf{g}}_s;$$
18. End
19. 
$$\rho_{g_n(k)+1} \mathbf{A} \phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{g}}_k = \rho_{g_n(k)+1} \mathbf{A} \phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{r}}_k + \sum_{s=\max(k-n,0)}^{g_n(k)n} \beta_s^{(k)} \rho_{g_n(k)+1} \mathbf{A} \phi_{g_n(s)+1}(\mathbf{A}) \widehat{\mathbf{g}}_s + \sum_{s=g_n(k)n+1}^{k-1} \beta_s^{(k)} \rho_{g_n(s)+1} \mathbf{A} \phi_{g_n(s)}(\mathbf{A}) \widehat{\mathbf{g}}_s;$$
20. 
$$\phi_{g_n(k)+1}(\mathbf{A}) \widehat{\mathbf{g}}_k = \phi_{g_n(k)+1}(\mathbf{A}) \widehat{\mathbf{r}}_k + \sum_{s=\max(k-n,0)}^{g_n(k)n} \beta_s^{(k)} (\rho_{g_n(k)+1} \mathbf{A} + \mathbf{I}) \phi_{g_n(s)+1}(\mathbf{A}) \widehat{\mathbf{g}}_s + \sum_{s=g_n(k)n+1}^{k-1} \beta_s^{(k)} \phi_{g_n(s)+1}(\mathbf{A}) \widehat{\mathbf{g}}_s;$$
21. Else
22. 
$$\beta_{g_n(k)n}^{(k)} = -\mathbf{q}_1^H \phi_{g_n(k)+1}(\mathbf{A}) \widehat{\mathbf{r}}_k / \rho_{g_n(k)+1} \mathbf{q}_1^H \mathbf{A} \phi_{g_n(g_n(k)n)+1}(\mathbf{A}) \widehat{\mathbf{g}}_{g_n(k)n};$$
23. For  $s = g_n(k)n + 1, \dots, k - 1$
24. 
$$\beta_s^{(k)} = -\mathbf{q}_{r_n(s+1)}^H \left( \phi_{g_n(k)+1}(\mathbf{A}) \widehat{\mathbf{r}}_k + \beta_{g_n(k)n}^{(k)} \rho_{g_n(k)+1} \mathbf{A} \phi_{g_n(g_n(k)n)+1}(\mathbf{A}) \widehat{\mathbf{g}}_{g_n(k)n} + \sum_{t=g_n(k)n+1}^{s-1} \beta_t^{(k)} \rho_{g_n(t)+1} \mathbf{A} \phi_{g_n(t)}(\mathbf{A}) \widehat{\mathbf{g}}_t \right) / \rho_{g_n(s)+1} \mathbf{q}_{r_n(s+1)}^H \mathbf{A} \phi_{g_n(s)}(\mathbf{A}) \widehat{\mathbf{g}}_s;$$
25. End
26. 
$$\phi_{g_n(k)+1}(\mathbf{A}) \widehat{\mathbf{g}}_k = \phi_{g_n(k)+1}(\mathbf{A}) \widehat{\mathbf{r}}_k + \beta_{g_n(k)n}^{(k)} (\rho_{g_n(k)+1} \mathbf{A} + \mathbf{I}) \phi_{g_n(g_n(k)n)+1}(\mathbf{A}) \widehat{\mathbf{g}}_{g_n(k)n} + \sum_{s=g_n(k)n+1}^{k-1} \beta_s^{(k)} \phi_{g_n(s)+1}(\mathbf{A}) \widehat{\mathbf{g}}_s;$$
27. End
28. End

As an example, let us show how the  $g_n(s + 1)$  inside the sum  $\sum_{t=\max(k-n,0)}^{s-1} \dots$  on Line 13, DV#2, was written as the  $g_n(t)$  on Line 13, DV#3.

If  $g_n(k) = 0$ , Line 13 of DV#2 is not implemented because of Convention (i) immediately following DV#1. So, we assume that  $g_n(k) > 0$ . Since

$$\max(k - n, 0) \leq s, t \leq g_n(k)n - 1,$$

we have

$$g_n(s + 1) = g_n(k + 1) - 1 = g_n(t + 1)$$

by Proposition 2.1(b). Now that  $g_n(k) > 0$ ,  $\max(k - n, 0) = k - n$  and hence

$$(4.5) \quad k - n \leq t \leq g_n(k)n - 1.$$

Let  $k = jn + i$  as in (2.1). Then (4.5) is

$$(j - 1)n + i \leq t \leq (j - 1)n + n - 1$$

which implies that  $r_n(t) < n$ . Now, Proposition 2.1(d) yields  $g_n(t + 1) = g_n(t)$  and therefore we have  $g_n(s + 1) = g_n(t)$ .

Substituting the vectors defined in (4.3) then leads to a set of updating relations about them.

**Derivation Version #4.**

1. For  $k = 1, 2, \dots$ , until convergence:
  2. If  $r_n(k) = 1$ 
    3.  $\alpha_k = \mathbf{q}_{r_n(k)}^H \mathbf{r}_{k-1} / \mathbf{q}_{r_n(k)}^H \mathbf{A} \mathbf{g}_{k-1}$ ;
    4.  $\mathbf{u}_k = \mathbf{r}_{k-1} - \alpha_k \mathbf{A} \mathbf{g}_{k-1}$ ;
    5.  $\mathbf{r}_k = \rho_{g_n(k)+1} \mathbf{A} \mathbf{u}_k + \mathbf{u}_k$ ;
  6. Else
    7.  $\alpha_k = \rho_{g_n(k-1)+1} \mathbf{q}_{r_n(k)}^H \mathbf{u}_{k-1} / \mathbf{q}_{r_n(k)}^H \mathbf{d}_{k-1}$ ;
    8.  $\mathbf{u}_k = \mathbf{u}_{k-1} - (\alpha_k / \rho_{g_n(k-1)+1}) \mathbf{d}_{k-1}$ ;
    9.  $\mathbf{r}_k = \mathbf{r}_{k-1} - \alpha_k \mathbf{A} \mathbf{g}_{k-1}$ ;
  10. End
    11. If  $r_n(k) < n$ 
      12. For  $s = \max(k-n, 0), \dots, g_n(k)n-1$ 
        13.  $\beta_s^{(k)} = -\mathbf{q}_{r_n(s+1)}^H \left( \mathbf{u}_k + \sum_{t=\max(k-n, 0)}^{s-1} \beta_t^{(k)} \mathbf{d}_t \right) / \mathbf{q}_{r_n(s+1)}^H \mathbf{d}_s$ ;
        14. End
          15.  $\beta_{g_n(k)n}^{(k)} = -\mathbf{q}_1^H \left( \mathbf{r}_k + \rho_{g_n(k)+1} \sum_{t=\max(k-n, 0)}^{g_n(k)n-1} \beta_t^{(k)} \mathbf{A} \mathbf{g}_t \right) / \rho_{g_n(k)+1} \mathbf{q}_1^H \mathbf{A} \mathbf{g}_{g_n(k)n}$ ;
          16. For  $s = g_n(k)n+1, \dots, k-1$ 
            17.  $\beta_s^{(k)} = -\mathbf{q}_{r_n(s+1)}^H \left( \mathbf{r}_k + \rho_{g_n(k)+1} \sum_{t=\max(k-n, 0)}^{g_n(k)n} \beta_t^{(k)} \mathbf{A} \mathbf{g}_t + \sum_{t=g_n(k)n+1}^{s-1} \beta_t^{(k)} \mathbf{d}_t \right) / \mathbf{q}_{r_n(s+1)}^H \mathbf{d}_s$ ;
          18. End
            19.  $\mathbf{d}_k = \mathbf{r}_k - \mathbf{u}_k + \rho_{g_n(k)+1} \sum_{s=\max(k-n, 0)}^{g_n(k)n} \beta_s^{(k)} \mathbf{A} \mathbf{g}_s + \sum_{s=g_n(k)n+1}^{k-1} \beta_s^{(k)} \mathbf{d}_s$ ;
            20.  $\mathbf{g}_k = \mathbf{r}_k + \sum_{s=\max(k-n, 0)}^{g_n(k)n} \beta_s^{(k)} (\rho_{g_n(k)+1} \mathbf{A} + \mathbf{I}) \mathbf{g}_s + \sum_{s=g_n(k)n+1}^{k-1} \beta_s^{(k)} \mathbf{g}_s$ ;
          21. Else
            22.  $\beta_{g_n(k)n}^{(k)} = -\mathbf{q}_1^H \mathbf{r}_k / \rho_{g_n(k)+1} \mathbf{q}_1^H \mathbf{A} \mathbf{g}_{g_n(k)n}$ ;
            23. For  $s = g_n(k)n+1, \dots, k-1$ 
              24.  $\beta_s^{(k)} = -\mathbf{q}_{r_n(s+1)}^H \left( \mathbf{r}_k + \rho_{g_n(k)+1} \beta_{g_n(k)n}^{(k)} \mathbf{A} \mathbf{g}_{g_n(k)n} + \sum_{t=g_n(k)n+1}^{s-1} \beta_t^{(k)} \mathbf{d}_t \right) / \mathbf{q}_{r_n(s+1)}^H \mathbf{d}_s$ ;
              25. End
                26.  $\mathbf{g}_k = \mathbf{r}_k + \beta_{g_n(k)n}^{(k)} (\rho_{g_n(k)+1} \mathbf{A} + \mathbf{I}) \mathbf{g}_{g_n(k)n} + \sum_{s=g_n(k)n+1}^{k-1} \beta_s^{(k)} \mathbf{g}_s$ ;
              27. End
                28. End

We consider  $\mathbf{r}_k$  to be the residual of the  $k$ th approximate solution  $\mathbf{x}_k$ . Updating relations about  $\mathbf{x}_k$  can be obtained from Lines 4, 5 and 9 respectively:

$$(4.6) \quad \mathbf{x}_k = \begin{cases} \mathbf{x}_{k-1} - \rho_{g_n(k)+1} \mathbf{u}_k + \alpha_k \mathbf{g}_{k-1}, & \text{if } r_n(k) = 1 \\ \mathbf{x}_{k-1} + \alpha_k \mathbf{g}_{k-1}, & \text{if } r_n(k) > 1. \end{cases}$$

After adding (4.6) to DV#4 and simplifying its operations appropriately, we arrive at the following ML( $n$ )BiCGStab algorithm. Just like BiCGStab, the free parameter  $\rho_{g_n(k)+1}$  is chosen to minimize the 2-norm of  $\mathbf{r}_k$ .

**ALGORITHM 4.1. ML( $n$ )BiCGStab without preconditioning associated with definition (4.3)**

1. Choose an initial guess  $\mathbf{x}_0$  and  $n$  vectors  $\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n$ .
2. Compute  $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$  and set  $\mathbf{g}_0 = \mathbf{r}_0$ . Compute  $\mathbf{w}_0 = \mathbf{A}\mathbf{g}_0$ ,  $c_0 = \mathbf{q}_1^H \mathbf{w}_0$ .
3. For  $k = 1, 2, \dots$ , until convergence:
  4. If  $r_n(k) = 1$ 
    5.  $\alpha_k = \mathbf{q}_{r_n(k)}^H \mathbf{r}_{k-1} / c_{k-1}$ ;
    6.  $\mathbf{u}_k = \mathbf{r}_{k-1} - \alpha_k \mathbf{w}_{k-1}$ ;
    7.  $\mathbf{x}_k = \mathbf{x}_{k-1} + \alpha_k \mathbf{g}_{k-1}$ ;
    8.  $\rho_{g_n(k)+1} = -(\mathbf{A}\mathbf{u}_k)^H \mathbf{u}_k / \|\mathbf{A}\mathbf{u}_k\|_2^2$ ;
    9.  $\mathbf{x}_k = \mathbf{x}_k - \rho_{g_n(k)+1} \mathbf{u}_k$ ;
    10.  $\mathbf{r}_k = \rho_{g_n(k)+1} \mathbf{A}\mathbf{u}_k + \mathbf{u}_k$ ;
  11. Else
    12.  $\tilde{\alpha}_k = \mathbf{q}_{r_n(k)}^H \mathbf{u}_{k-1} / c_{k-1}$ ;      %  $\tilde{\alpha}_k = \alpha_k / \rho_{g_n(k-1)+1}$
    13. If  $r_n(k) < n$ 
      14.  $\mathbf{u}_k = \mathbf{u}_{k-1} - \tilde{\alpha}_k \mathbf{d}_{k-1}$ ;
    15. End
    16.  $\mathbf{x}_k = \mathbf{x}_{k-1} + \rho_{g_n(k-1)+1} \tilde{\alpha}_k \mathbf{g}_{k-1}$ ;
    17.  $\mathbf{r}_k = \mathbf{r}_{k-1} - \rho_{g_n(k-1)+1} \tilde{\alpha}_k \mathbf{w}_{k-1}$ ;
  18. End
  19. If  $r_n(k) < n$ 
    20.  $\mathbf{z}_d = \mathbf{u}_k$ ,  $\mathbf{g}_k = \mathbf{0}$ ,  $\mathbf{z}_w = \mathbf{0}$ ;
    21. For  $s = k - n, \dots, g_n(k)n - 1$  and  $g_n(k) \geq 1$ 
      22.  $\beta_s^{(k)} = -\mathbf{q}_{r_n(s+1)}^H \mathbf{z}_d / c_s$ ;
      23.  $\mathbf{z}_d = \mathbf{z}_d + \beta_s^{(k)} \mathbf{d}_s$ ;
      24.  $\mathbf{g}_k = \mathbf{g}_k + \beta_s^{(k)} \mathbf{g}_s$ ;
      25.  $\mathbf{z}_w = \mathbf{z}_w + \beta_s^{(k)} \mathbf{w}_s$ ;
    26. End
    27.  $\mathbf{z}_w = \mathbf{r}_k + \rho_{g_n(k)+1} \mathbf{z}_w$ ;
    28.  $\beta_{g_n(k)n}^{(k)} = -\mathbf{q}_1^H \mathbf{z}_w / (\rho_{g_n(k)+1} c_{g_n(k)n})$ ;
    29.  $\mathbf{z}_w = \mathbf{z}_w + \rho_{g_n(k)+1} \beta_{g_n(k)n}^{(k)} \mathbf{w}_{g_n(k)n}$ ;
    30.  $\mathbf{g}_k = \mathbf{g}_k + \mathbf{z}_w + \beta_{g_n(k)n}^{(k)} \mathbf{g}_{g_n(k)n}$ ;
    31. For  $s = g_n(k)n + 1, \dots, k - 1$ 
      32.  $\beta_s^{(k)} = -\mathbf{q}_{r_n(s+1)}^H \mathbf{z}_w / c_s$ ;
      33.  $\mathbf{g}_k = \mathbf{g}_k + \beta_s^{(k)} \mathbf{g}_s$ ;
      34.  $\mathbf{z}_w = \mathbf{z}_w + \beta_s^{(k)} \mathbf{d}_s$ ;
    35. End
    36.  $\mathbf{d}_k = \mathbf{z}_w - \mathbf{u}_k$ ;
    37.  $c_k = \mathbf{q}_{r_n(k+1)}^H \mathbf{d}_k$ ;
    38.  $\mathbf{w}_k = \mathbf{A}\mathbf{g}_k$ ;
  39. Else
    40.  $\beta_{g_n(k)n}^{(k)} = -\mathbf{q}_1^H \mathbf{r}_k / (\rho_{g_n(k)+1} c_{g_n(k)n})$ ;
    41.  $\mathbf{z}_w = \mathbf{r}_k + \rho_{g_n(k)+1} \beta_{g_n(k)n}^{(k)} \mathbf{w}_{g_n(k)n}$ ;
    42.  $\mathbf{g}_k = \mathbf{z}_w + \beta_{g_n(k)n}^{(k)} \mathbf{g}_{g_n(k)n}$ ;
    43. For  $s = g_n(k)n + 1, \dots, k - 1$ 
      44.  $\beta_s^{(k)} = -\mathbf{q}_{r_n(s+1)}^H \mathbf{z}_w / c_s$ ;
      45.  $\mathbf{g}_k = \mathbf{g}_k + \beta_s^{(k)} \mathbf{g}_s$ ;

TABLE 4.1  
Average cost per ( $k$ -)iteration of Algorithm 9.1 and its storage requirement.

Preconditioning ( $\mathbf{M}^{-1}\mathbf{v}$ )	$1 + \frac{1}{n}$	Vector addition ( $\mathbf{u} \pm \mathbf{v}$ )	$2 - \frac{2}{n}$
Matvec ( $\mathbf{A}\mathbf{v}$ )	$1 + \frac{1}{n}$	Saxpy ( $\mathbf{u} + \alpha\mathbf{v}$ )	$\max(2.5n + 2.5 - \frac{2}{n}, 6)$
dot product ( $\mathbf{u}^H\mathbf{v}$ )	$n + 1 + \frac{2}{n}$	Storage	$\mathbf{A} + \mathbf{M} + (4n + 5)N + O(n)$

$$46. \quad \mathbf{z}_w = \mathbf{z}_w + \beta_s^{(k)} \mathbf{d}_s;$$

47. End

$$48. \quad \mathbf{w}_k = \mathbf{A}\mathbf{g}_k;$$

$$49. \quad c_k = \mathbf{q}_{r_n(k)+1}^H \mathbf{w}_k;$$

50. End

51. End

We remark that (i) the algorithm does not compute the quantities  $\mathbf{u}_k$  and  $\mathbf{d}_k$  when  $r_n(k) = n$  (see Lines 13-15 and Lines 39-50); (ii) if the  $\mathbf{u}_k$  on Line 6 happens to be zero, then the  $\rho_{g_n(k)+1}$  on Line 8 and therefore the  $\mathbf{x}_k$  and  $\mathbf{r}_k$  on Lines 9 and 10 will not be computable. In this case, however, the  $\mathbf{x}_k$  on Line 7 will be the exact solution to system (3.1) and the algorithm stops there.

We now compare Algorithm 4.1 to the ML( $n$ )BiCGStab algorithm in [35]. First, the definitions of  $\mathbf{r}_k$ ,  $\mathbf{u}_k$  and  $\mathbf{g}_k$  are the same in both algorithms, but  $\mathbf{d}_k$  is defined differently. In [35],  $\mathbf{d}_k = \phi_{g_n(k)}(\mathbf{A})\widehat{\mathbf{g}}_k$ . In exact arithmetic, however, both algorithms compute the same  $\rho_{g_n(k)+1}$ ,  $\mathbf{r}_k$  and  $\mathbf{x}_k$ . Second, the derivation of Algorithm 4.1 has been made simpler by using index functions. As a result, some redundant operations in Algorithm 2 of [35] can be seen and removed and some arithmetics are simplified. For example, the vectors  $\mathbf{d}_k$ ,  $\mathbf{u}_k$  are computed in every iteration in Algorithm 2 of [35]. They are now computed only when  $r_n(k) < n$ . Also, the expression of  $\beta_{g_n(k)n}^{(k)}$  on Line 39 of Algorithm 4.1 is simpler. Some other minor changes were also made so that the algorithm becomes more efficient.

Computational cost and storage requirement of Algorithm 4.1, obtained based on its preconditioned version, Algorithm 9.1 in §9, are summarized in Table 4.1. Since the vectors  $\{\mathbf{q}_1, \dots, \mathbf{q}_n\}$ ,  $\{\mathbf{d}_{k-n}, \dots, \mathbf{d}_{g_n(k)n-1}, \mathbf{d}_{g_n(k)n+1}, \dots, \mathbf{d}_{k-1}\}$ ,  $\{\mathbf{g}_{k-n}, \dots, \mathbf{g}_{k-1}\}$  and  $\{\mathbf{w}_{k-n}, \dots, \mathbf{w}_{g_n(k)n}, \mathbf{w}_{k-1}\}$  are required in iteration  $k$ , they must be stored. When  $n$  is large, this storage is dominant. So, the storage requirement of the algorithm is about  $4nN$ .

**4.3. Properties.** We summarize the properties about Algorithm 4.1 in the following proposition. Since  $\mathbf{r}_0 = \widehat{\mathbf{r}}_0$  by (4.4),  $\nu$  is also the degree of the minimal polynomial of  $\mathbf{r}_0$  with respect to  $\mathbf{A}$ .

PROPOSITION 4.2. *Under the assumptions of Proposition 3.2, if  $\rho_{g_n(k)+1} \neq 0$  and  $-1/\rho_{g_n(k)+1} \notin \sigma(\mathbf{A})$  for  $1 \leq k \leq \nu - 1$ , where  $\sigma(\mathbf{A})$  is the spectrum of  $\mathbf{A}$ , then Algorithm 4.1 does not break down by zero division for  $k = 1, 2, \dots, \nu$ , and the approximate solution  $\mathbf{x}_\nu$  at step  $k = \nu$  is exact to the system (3.1). Moreover, the computed quantities satisfy*

$$(a) \quad \mathbf{x}_k \in \mathbf{x}_0 + \text{span}\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \dots, \mathbf{A}^{g_n(k)+k}\mathbf{r}_0\} \text{ and } \mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k \in \mathbf{r}_0 + \text{span}\{\mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \dots, \mathbf{A}^{g_n(k)+k+1}\mathbf{r}_0\} \text{ for } 1 \leq k \leq \nu - 1.$$

- (b)  $\mathbf{r}_k \neq \mathbf{0}$  for  $1 \leq k \leq \nu - 1$  and  $\mathbf{r}_\nu = \mathbf{0}$ .
- (c)  $\mathbf{r}_k \not\perp \mathbf{q}_1$  for  $1 \leq k \leq \nu - 1$  with  $r_n(k) = n$ .
- (d)  $\mathbf{u}_k \perp \text{span}\{\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_{r_n(k)}\}$  and  $\mathbf{u}_k \not\perp \mathbf{q}_{r_n(k)+1}$  for  $1 \leq k \leq \nu - 1$  with  $r_n(k) < n$ .
- (e)  $\mathbf{A}(\mathbf{g}_k - \mathbf{d}_k) \perp \text{span}\{\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_{r_n(k)}\}$  and  $\mathbf{A}(\mathbf{g}_k - \mathbf{d}_k) \not\perp \mathbf{q}_{r_n(k)+1}$  for  $1 \leq k \leq \nu - 1$  with  $r_n(k) < n$ .

*Proof.* The divisors in Algorithm 4.1 are  $c_k, \|\mathbf{A}\mathbf{u}_k\|_2^2$  and  $\rho_{g_n(k)+1}$  respectively, where the  $\rho$ 's are assumed to be nonzero. By Proposition 3.2(c), we have  $\mathbf{A}\widehat{\mathbf{r}}_k \neq \mathbf{0}$  for  $1 \leq k \leq \nu - 1$ . Since  $\mathbf{A}\mathbf{u}_k = \phi_{g_n(k)}(\mathbf{A})\widehat{\mathbf{r}}_k = (\rho_{g_n(k)}\mathbf{A} + \mathbf{I})(\rho_{g_n(k)-1}\mathbf{A} + \mathbf{I}) \cdots (\rho_1\mathbf{A} + \mathbf{I})\widehat{\mathbf{r}}_k$  by (4.3) and (4.1) and since  $-1/\rho \notin \sigma(\mathbf{A})$ ,  $\widehat{\mathbf{r}}_k \neq \mathbf{0}$  implies  $\mathbf{A}\mathbf{u}_k \neq \mathbf{0}$ . Hence  $\|\mathbf{A}\mathbf{u}_k\|_2 \neq 0$  for  $1 \leq k \leq \nu - 1$ .

The  $c$ 's are defined on Lines 37 and 49 in the algorithm. If  $c_k = \mathbf{q}_{r_n(k)+1}^H \mathbf{d}_k$  with  $r_n(k) < n$ , then  $c_k = \rho_{g_n(k)+1} \mathbf{q}_{r_n(k)+1}^H \mathbf{A} \phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{g}}_k = \rho_{g_n(k)+1} \mathbf{q}_{r_n(k)+1}^H \mathbf{A} \phi_{g_n(k)+1}(\mathbf{A}) \widehat{\mathbf{g}}_k = \rho_{g_n(k)+1} \rho_{g_n(k)} \cdots \rho_1 \mathbf{p}_{k+1}^H \mathbf{A} \widehat{\mathbf{g}}_k$  by (4.3), Proposition 2.1(d) and Corollary 3.3 respectively. Since the  $\rho$ 's are nonzero by assumption and  $\mathbf{p}_{k+1}^H \mathbf{A} \widehat{\mathbf{g}}_k \neq 0$  by Proposition 3.2(g), we have  $c_k \neq 0$  for  $1 \leq k \leq \nu - 1$ . On the other hand, if  $c_k = \mathbf{q}_{r_n(k)+1}^H \mathbf{w}_k = \mathbf{q}_{r_n(k)+1}^H \mathbf{A} \mathbf{g}_k$  with  $r_n(k) = n$ , then  $c_k = \mathbf{q}_{r_n(k)+1}^H \mathbf{A} \phi_{g_n(k)+1}(\mathbf{A}) \widehat{\mathbf{g}}_k = \mathbf{q}_{r_n(k)+1}^H \mathbf{A} \phi_{g_n(k)+1}(\mathbf{A}) \widehat{\mathbf{g}}_k = \rho_{g_n(k)+1} \rho_{g_n(k)} \cdots \rho_1 \mathbf{p}_{k+1}^H \mathbf{A} \widehat{\mathbf{g}}_k \neq 0$  by (4.3), Proposition 2.1(d), Corollary 3.3 and Proposition 3.2(g). Therefore, in either case, we always have  $c_k \neq 0$  for  $1 \leq k \leq \nu - 1$ . Moreover,  $c_0 = \mathbf{q}_1^H \mathbf{w}_0 = \mathbf{q}_1^H \mathbf{A} \mathbf{g}_0$  according to Line 2 of the algorithm. Since  $\mathbf{p}_1 = \mathbf{q}_1$  by (3.2) and  $\mathbf{g}_0 = \widehat{\mathbf{g}}_0$  by (4.4),  $c_0 \neq 0$  by Proposition 3.2(g).

Now that  $\|\mathbf{A}\mathbf{u}_k\|_2 \neq 0$  and  $\rho_{g_n(k)+1} \neq 0$  for  $1 \leq k \leq \nu - 1$  and  $c_k \neq 0$  for  $0 \leq k \leq \nu - 1$ , Algorithm 4.1 does not break down by zero division in the first  $\nu - 1$  iterations. When  $k = \nu$ ,  $\mathbf{u}_k = \mathbf{u}_\nu = \phi_{g_n(\nu)}(\mathbf{A})\widehat{\mathbf{r}}_\nu = \mathbf{0}$  and  $\mathbf{r}_k = \mathbf{r}_\nu = \phi_{g_n(\nu)+1}(\mathbf{A})\widehat{\mathbf{r}}_\nu = \mathbf{0}$  since  $\widehat{\mathbf{r}}_\nu = \mathbf{0}$  by Proposition 3.2. If it happens that  $r_n(\nu) = 1$ , then the  $\mathbf{x}_k (= \mathbf{x}_\nu)$  on Line 7 is the exact solution to system (3.1) since its residual  $\mathbf{u}_\nu$  is zero. So, the algorithm stops there. Otherwise, the  $\mathbf{x}_k (= \mathbf{x}_\nu)$  on Line 16 will be exact with residual  $\mathbf{r}_\nu = \mathbf{0}$  and where the algorithm stops.

Part (a) follows from the definition of  $\mathbf{r}_k$  in (4.3) and Proposition 3.2(a).

Since  $\widehat{\mathbf{r}}_k \neq \mathbf{0}$  for  $1 \leq k \leq \nu - 1$  by Proposition 3.2(b) and since  $-1/\rho \notin \sigma(\mathbf{A})$ , we have  $\mathbf{r}_k = \phi_{g_n(k)+1}(\mathbf{A})\widehat{\mathbf{r}}_k = (\rho_{g_n(k)+1}\mathbf{A} + \mathbf{I})(\rho_{g_n(k)}\mathbf{A} + \mathbf{I}) \cdots (\rho_1\mathbf{A} + \mathbf{I})\widehat{\mathbf{r}}_k \neq \mathbf{0}$ . Therefore, Part (b) holds.

For Part (c), write  $k = jn + n$  with  $0 \leq j$ . By (4.3) and Corollary 3.3, we have  $\mathbf{q}_1^H \mathbf{r}_k = \mathbf{q}_1^H \phi_{g_n(k)+1}(\mathbf{A})\widehat{\mathbf{r}}_k = \mathbf{q}_{r_n((j+1)n+1)}^H \phi_{g_n((j+1)n+1)}(\mathbf{A})\widehat{\mathbf{r}}_k = \mathbf{q}_{r_n(k)+1}^H \phi_{g_n(k)+1}(\mathbf{A})\widehat{\mathbf{r}}_k = \rho_{g_n(k)+1} \rho_{g_n(k)-1} \cdots \rho_1 \mathbf{p}_{k+1}^H \widehat{\mathbf{r}}_k = \rho_{g_n(k)+1} \rho_{g_n(k)} \cdots \rho_1 \mathbf{p}_{k+1}^H \widehat{\mathbf{r}}_k$ . Now Part (c) follows from Proposition 3.2(d) and the assumption that the  $\rho$ 's are nonzero.

For the proof of Part (d), we first note that Algorithm 4.1 does not compute  $\mathbf{u}_k$  when  $r_n(k) = n$  (see Lines 13 - 15). Write  $k = jn + i$  as in (2.1) and let  $1 \leq t \leq i < n$ . Then  $r_n(k) = i$ ,  $g_n(k) = j = g_n(jn + t)$  and  $r_n(jn + t) = t$ . Now, by (4.3) and Corollary 3.3, we have  $\mathbf{q}_t^H \mathbf{u}_k = \mathbf{q}_t^H \phi_{g_n(k)}(\mathbf{A})\widehat{\mathbf{r}}_k = \mathbf{q}_{r_n(jn+t)}^H \phi_{g_n(jn+t)}(\mathbf{A})\widehat{\mathbf{r}}_k = \rho_{g_n(jn+t)} \rho_{g_n(jn+t)-1} \cdots \rho_1 \mathbf{p}_{jn+t}^H \widehat{\mathbf{r}}_k = \rho_{g_n(k)} \rho_{g_n(k)-1} \cdots \rho_1 \mathbf{p}_{jn+t}^H \widehat{\mathbf{r}}_k$ . Since  $\mathbf{p}_{jn+t}^H \widehat{\mathbf{r}}_k = 0$  by Proposition 3.2(d),  $\mathbf{q}_t^H \mathbf{u}_k = 0$  for  $1 \leq t \leq i = r_n(k)$ . Similarly,  $\mathbf{q}_{i+1}^H \mathbf{u}_k = \rho_{g_n(k)} \rho_{g_n(k)-1} \cdots \rho_1 \mathbf{p}_{jn+i+1}^H \widehat{\mathbf{r}}_k = \rho_{g_n(k)} \rho_{g_n(k)-1} \cdots \rho_1 \mathbf{p}_{k+1}^H \widehat{\mathbf{r}}_k$  (the validity of this equation requires  $i < n$ ). Because of Proposition 3.2(d) and  $\rho \neq 0$ ,  $\mathbf{q}_{i+1}^H \mathbf{u}_k \neq 0$ .

Similar to the quantity  $\mathbf{u}_k$ , Algorithm 4.1 does not compute  $\mathbf{d}_k$  when  $r_n(k) = n$  (see Lines 40 - 49). By (4.3),  $\mathbf{g}_k - \mathbf{d}_k = \phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{g}}_k$ . Now, the proof of Part (e) is

parallel to that of Part (d).  $\blacksquare$

The conditions of  $\rho_{g_n(k)+1} \neq 0$  and  $-1/\rho_{g_n(k)+1} \notin \sigma(\mathbf{A})$  can be easily made satisfied. For example, one can add some small random noise to it after it is computed.

This proposition indicates that exact solution can only be found at iteration  $k = \nu$ . It is possible, however,  $\|\mathbf{r}_k\|_2$  can become very small for some  $k < \nu$ . In practice, we terminate the algorithm when  $\|\mathbf{r}_k\|_2$  falls within a given tolerance.

As in the case of ML( $n$ )BiCG, ML( $n$ )BiCGStab can also encounter an exact or a near-breakdown in practice. Besides the two types of breakdown described about ML( $n$ )BiCG, ML( $n$ )BiCGStab has one more type of breakdown caused by  $\rho$ . A breakdown-free ML( $n$ )BiCGStab algorithm has been developed in [15]. Also, see [3, 10, 12, 20, 21, 25] and the references therein for look-ahead techniques to cure breakdowns in Lanczos-type product methods.

**5. A Second ML( $n$ )BiCGStab Algorithm.** If we write  $k = jn + i$  as in (2.1), the  $\mathbf{r}_k$  defined by (4.4) then becomes

$$(5.1) \quad \mathbf{r}_{jn+i} = \phi_{j+1}(\mathbf{A}) \widehat{\mathbf{r}}_{jn+i}$$

where  $i = 1, 2, \dots, n$  and  $j = 0, 1, 2, \dots$ .

Starting with  $k = 1$ , let us call every  $n$  consecutive  $k$ -iterations a ‘‘cycle’’, namely, iterations  $k = 1, 2, \dots, n$  form the first cycle, iterations  $k = n + 1, n + 2, \dots, n + n$  the second cycle and so on. Then (5.1) increases the degree of the polynomial  $\phi$  by 1 at the beginning of every cycle. For example, consider  $n = 3$ . Then (5.1) implies that

$$\begin{array}{lll} \mathbf{r}_1 = \phi_1(\mathbf{A}) \widehat{\mathbf{r}}_1, & \mathbf{r}_4 = \phi_2(\mathbf{A}) \widehat{\mathbf{r}}_4, & \mathbf{r}_7 = \phi_3(\mathbf{A}) \widehat{\mathbf{r}}_7, \\ \mathbf{r}_2 = \phi_1(\mathbf{A}) \widehat{\mathbf{r}}_2, & \mathbf{r}_5 = \phi_2(\mathbf{A}) \widehat{\mathbf{r}}_5, & \mathbf{r}_8 = \phi_3(\mathbf{A}) \widehat{\mathbf{r}}_8, \\ \mathbf{r}_3 = \phi_1(\mathbf{A}) \widehat{\mathbf{r}}_3, & \mathbf{r}_6 = \phi_2(\mathbf{A}) \widehat{\mathbf{r}}_6, & \mathbf{r}_9 = \phi_3(\mathbf{A}) \widehat{\mathbf{r}}_9. \end{array}$$

Iteration  $k = 4$  is the first iteration of the second cycle and the degree of  $\phi$  is increased from 1 to 2 there.

One can define  $\mathbf{r}_k$  by increasing the degree of  $\phi$  by one anywhere within a cycle. Correspondingly, (we believe) the definition will lead to a different algorithm of ML( $n$ )BiCGStab. As an illustration, let us increase the degree of  $\phi$  at the end of every cycle and derive the algorithm associated with it.

**5.1. Notation and Definitions.** Let  $\phi_k(\lambda)$  be defined as in (4.1). For  $k \in \mathcal{N}$ , define

$$(5.2) \quad \begin{array}{ll} \mathbf{r}_k = \phi_{g_n(k+1)}(\mathbf{A}) \widehat{\mathbf{r}}_k, & \mathbf{g}_k = \phi_{g_n(k+1)}(\mathbf{A}) \widehat{\mathbf{g}}_k, \\ \mathbf{u}_k = \phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{r}}_k, & \mathbf{w}_k = \mathbf{A} \mathbf{g}_k. \end{array}$$

and set

$$(5.3) \quad \mathbf{r}_0 = \widehat{\mathbf{r}}_0 \quad \text{and} \quad \mathbf{g}_0 = \widehat{\mathbf{g}}_0.$$

The vector  $\mathbf{r}_k$  is considered to be the residual of the approximate solution  $\mathbf{x}_k$  computed. We remark that  $\mathbf{r}_k = \mathbf{u}_k$  when  $r_n(k) < n$  since  $g_n(k+1) = g_n(k)$  in this case.

Definition (5.2) increases the degree of  $\phi$  at the end of a cycle. To see this, let  $n = 3$ . Then (5.2) yields

$$\begin{array}{lll} \mathbf{r}_1 = \phi_0(\mathbf{A}) \widehat{\mathbf{r}}_1, & \mathbf{r}_4 = \phi_1(\mathbf{A}) \widehat{\mathbf{r}}_4, & \mathbf{r}_7 = \phi_2(\mathbf{A}) \widehat{\mathbf{r}}_7, \\ \mathbf{r}_2 = \phi_0(\mathbf{A}) \widehat{\mathbf{r}}_2, & \mathbf{r}_5 = \phi_1(\mathbf{A}) \widehat{\mathbf{r}}_5, & \mathbf{r}_8 = \phi_2(\mathbf{A}) \widehat{\mathbf{r}}_8, \\ \mathbf{r}_3 = \phi_1(\mathbf{A}) \widehat{\mathbf{r}}_3, & \mathbf{r}_6 = \phi_2(\mathbf{A}) \widehat{\mathbf{r}}_6, & \mathbf{r}_9 = \phi_3(\mathbf{A}) \widehat{\mathbf{r}}_9. \end{array}$$

**5.2. Algorithm Derivation.** To derive the algorithm associated with (5.2), we first transform Algorithm 3.1 (forgetting Lines 1, 2, 5 and 11) into the following version which is computationally equivalent to Algorithm 3.1, but is more convenient for us to apply Proposition 2.1.

**Derivation Version #5.**

1. For  $k = 1, 2, \dots$ , until convergence:
2.  $\alpha_k = \mathbf{p}_k^H \widehat{\mathbf{r}}_{k-1} / \mathbf{p}_k^H \mathbf{A} \widehat{\mathbf{g}}_{k-1}$ ;
3. If  $r_n(k) < n$
4.  $\widehat{\mathbf{r}}_k = \widehat{\mathbf{r}}_{k-1} - \alpha_k \mathbf{A} \widehat{\mathbf{g}}_{k-1}$ ;
5. For  $s = \max(k - n, 0), \dots, g_n(k)n - 1$
6.  $\beta_s^{(k)} = -\mathbf{p}_{s+1}^H \mathbf{A} \left( \widehat{\mathbf{r}}_k + \sum_{t=\max(k-n,0)}^{s-1} \beta_t^{(k)} \widehat{\mathbf{g}}_t \right) / \mathbf{p}_{s+1}^H \mathbf{A} \widehat{\mathbf{g}}_s$ ;
7. End
8. For  $s = g_n(k)n, \dots, k - 1$
9.  $\beta_s^{(k)} = -\mathbf{p}_{s+1}^H \mathbf{A} \left( \widehat{\mathbf{r}}_k + \sum_{t=\max(k-n,0)}^{g_n(k)n-1} \beta_t^{(k)} \widehat{\mathbf{g}}_t + \sum_{t=g_n(k)n}^{s-1} \beta_t^{(k)} \widehat{\mathbf{g}}_t \right) / \mathbf{p}_{s+1}^H \mathbf{A} \widehat{\mathbf{g}}_s$ ;
10. End
11.  $\widehat{\mathbf{g}}_k = \widehat{\mathbf{r}}_k + \sum_{s=\max(k-n,0)}^{g_n(k)n-1} \beta_s^{(k)} \widehat{\mathbf{g}}_s + \sum_{s=g_n(k)n}^{k-1} \beta_s^{(k)} \widehat{\mathbf{g}}_s$ ;
12. Else
13.  $\widehat{\mathbf{r}}_k = \widehat{\mathbf{r}}_{k-1} - \alpha_k \mathbf{A} \widehat{\mathbf{g}}_{k-1}$ ;
14. For  $s = g_n(k)n, \dots, k - 1$
15.  $\beta_s^{(k)} = -\mathbf{p}_{s+1}^H \mathbf{A} \left( \widehat{\mathbf{r}}_k + \sum_{t=g_n(k)n}^{s-1} \beta_t^{(k)} \widehat{\mathbf{g}}_t \right) / \mathbf{p}_{s+1}^H \mathbf{A} \widehat{\mathbf{g}}_s$ ;
16. End
17.  $\widehat{\mathbf{g}}_k = \widehat{\mathbf{r}}_k + \sum_{s=g_n(k)n}^{k-1} \beta_s^{(k)} \widehat{\mathbf{g}}_s$ ;
18. End
19. End

Then we transform DV#5 as follows by Corollary 3.3.

**Derivation Version #6.**

1. For  $k = 1, 2, \dots$ , until convergence:
2.  $\alpha_k = \mathbf{q}_{r_n(k)}^H \phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{r}}_{k-1} / \mathbf{q}_{r_n(k)}^H \mathbf{A} \phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{g}}_{k-1}$ ;
3. If  $r_n(k) < n$
4.  $\phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{r}}_k = \phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{r}}_{k-1} - \alpha_k \mathbf{A} \phi_{g_n(k)}(\mathbf{A}) \widehat{\mathbf{g}}_{k-1}$ ;
5. For  $s = \max(k - n, 0), \dots, g_n(k)n - 1$
6.  $\beta_s^{(k)} = -\mathbf{q}_{r_n(s+1)}^H \left( \phi_{g_n(s+1)+1}(\mathbf{A}) \widehat{\mathbf{r}}_k \right. \\ \left. + \rho_{g_n(s+1)+1} \sum_{t=\max(k-n,0)}^{s-1} \beta_t^{(k)} \mathbf{A} \phi_{g_n(s+1)}(\mathbf{A}) \widehat{\mathbf{g}}_t \right) / \rho_{g_n(s+1)+1} \mathbf{q}_{r_n(s+1)}^H \mathbf{A} \phi_{g_n(s+1)}(\mathbf{A}) \widehat{\mathbf{g}}_s$ ;
7. End
8. For  $s = g_n(k)n, \dots, k - 1$
9.  $\beta_s^{(k)} = -\mathbf{q}_{r_n(s+1)}^H \mathbf{A} \left( \phi_{g_n(s+1)}(\mathbf{A}) \widehat{\mathbf{r}}_k + \sum_{t=\max(k-n,0)}^{g_n(k)n-1} \beta_t^{(k)} \phi_{g_n(s+1)}(\mathbf{A}) \widehat{\mathbf{g}}_t \right. \\ \left. + \sum_{t=g_n(k)n}^{s-1} \beta_t^{(k)} \phi_{g_n(s+1)}(\mathbf{A}) \widehat{\mathbf{g}}_t \right) / \mathbf{q}_{r_n(s+1)}^H \mathbf{A} \phi_{g_n(s+1)}(\mathbf{A}) \widehat{\mathbf{g}}_s$ ;
10. End
11.  $\phi_{g_n(k+1)}(\mathbf{A}) \widehat{\mathbf{g}}_k = \phi_{g_n(k+1)}(\mathbf{A}) \widehat{\mathbf{r}}_k + (\rho_{g_n(k+1)} \mathbf{A} + \mathbf{I}) \sum_{s=\max(k-n,0)}^{g_n(k)n-1} \beta_s^{(k)} \phi_{g_n(k+1)-1}(\mathbf{A}) \widehat{\mathbf{g}}_s + \\ \sum_{s=g_n(k)n}^{k-1} \beta_s^{(k)} \phi_{g_n(k+1)}(\mathbf{A}) \widehat{\mathbf{g}}_s$ ;
12. Else

13.  $\phi_{g_n(k)}(\mathbf{A})\widehat{\mathbf{r}}_k = \phi_{g_n(k)}(\mathbf{A})\widehat{\mathbf{r}}_{k-1} - \alpha_k \mathbf{A} \phi_{g_n(k)}(\mathbf{A})\widehat{\mathbf{g}}_{k-1};$
14.  $\phi_{g_n(k+1)}(\mathbf{A})\widehat{\mathbf{r}}_k = (\rho_{g_n(k+1)} \mathbf{A} + \mathbf{I}) \phi_{g_n(k+1)-1}(\mathbf{A})\widehat{\mathbf{r}}_k;$
15. For  $s = g_n(k)n, \dots, k-1$
16. 
$$\beta_s^{(k)} = -\mathbf{q}_{r_n(s+1)}^H \left( \phi_{g_n(s+1)+1}(\mathbf{A})\widehat{\mathbf{r}}_k + \rho_{g_n(s+1)+1} \sum_{t=g_n(k)n}^{s-1} \beta_t^{(k)} \mathbf{A} \phi_{g_n(s+1)}(\mathbf{A})\widehat{\mathbf{g}}_t \right) / \rho_{g_n(s+1)+1} \mathbf{q}_{r_n(s+1)}^H \mathbf{A} \phi_{g_n(s+1)}(\mathbf{A})\widehat{\mathbf{g}}_s;$$
17. End
18.  $\phi_{g_n(k+1)}(\mathbf{A})\widehat{\mathbf{g}}_k = \phi_{g_n(k+1)}(\mathbf{A})\widehat{\mathbf{r}}_k + (\rho_{g_n(k+1)} \mathbf{A} + \mathbf{I}) \sum_{s=g_n(k)n}^{k-1} \beta_s^{(k)} \phi_{g_n(k+1)-1}(\mathbf{A})\widehat{\mathbf{g}}_s;$
19. End
20. End

Lines 4, 11, 13 and 18, DV#6, were obtained from Lines 4, 11, 13 and 17, DV#5, by multiplying them with  $\phi_{g_n(k)}(\mathbf{A})$  and  $\phi_{g_n(k+1)}(\mathbf{A})$  respectively. Line 14, DV#6, is a direct result of the definition (4.1) of  $\phi$ .

Now we use Proposition 2.1 to write DV#6 as

### Derivation Version #7.

1. For  $k = 1, 2, \dots$ , until convergence:
2.  $\alpha_k = \mathbf{q}_{r_n(k)}^H \phi_{g_n(k)}(\mathbf{A})\widehat{\mathbf{r}}_{k-1} / \mathbf{q}_{r_n(k)}^H \mathbf{A} \phi_{g_n(k)}(\mathbf{A})\widehat{\mathbf{g}}_{k-1};$
3. If  $r_n(k) < n$
4.  $\phi_{g_n(k+1)}(\mathbf{A})\widehat{\mathbf{r}}_k = \phi_{g_n(k)}(\mathbf{A})\widehat{\mathbf{r}}_{k-1} - \alpha_k \mathbf{A} \phi_{g_n(k)}(\mathbf{A})\widehat{\mathbf{g}}_{k-1};$
5. For  $s = \max(k-n, 0), \dots, g_n(k)n-1$
6. 
$$\beta_s^{(k)} = -\mathbf{q}_{r_n(s+1)}^H \left( \phi_{g_n(k+1)}(\mathbf{A})\widehat{\mathbf{r}}_k + \rho_{g_n(k+1)} \sum_{t=\max(k-n, 0)}^{s-1} \beta_t^{(k)} \mathbf{A} \phi_{g_n(t+1)}(\mathbf{A})\widehat{\mathbf{g}}_t \right) / \rho_{g_n(k+1)} \mathbf{q}_{r_n(s+1)}^H \mathbf{A} \phi_{g_n(s+1)}(\mathbf{A})\widehat{\mathbf{g}}_s;$$
7. End
8. For  $s = g_n(k)n, \dots, k-1$
9. 
$$\beta_s^{(k)} = -\mathbf{q}_{r_n(s+1)}^H \mathbf{A} \left( \phi_{g_n(k+1)}(\mathbf{A})\widehat{\mathbf{r}}_k + \sum_{t=\max(k-n, 0)}^{g_n(k)n-1} \beta_t^{(k)} \phi_{g_n(t+1)+1}(\mathbf{A})\widehat{\mathbf{g}}_t + \sum_{t=g_n(k)n}^{s-1} \beta_t^{(k)} \phi_{g_n(t+1)}(\mathbf{A})\widehat{\mathbf{g}}_t \right) / \mathbf{q}_{r_n(s+1)}^H \mathbf{A} \phi_{g_n(s+1)}(\mathbf{A})\widehat{\mathbf{g}}_s;$$
10. End
11. 
$$\phi_{g_n(k+1)}(\mathbf{A})\widehat{\mathbf{g}}_k = \phi_{g_n(k+1)}(\mathbf{A})\widehat{\mathbf{r}}_k + (\rho_{g_n(k+1)} \mathbf{A} + \mathbf{I}) \sum_{s=\max(k-n, 0)}^{g_n(k)n-1} \beta_s^{(k)} \phi_{g_n(s+1)}(\mathbf{A})\widehat{\mathbf{g}}_s + \sum_{s=g_n(k)n}^{k-1} \beta_s^{(k)} \phi_{g_n(s+1)}(\mathbf{A})\widehat{\mathbf{g}}_s;$$
12. Else
13.  $\phi_{g_n(k)}(\mathbf{A})\widehat{\mathbf{r}}_k = \phi_{g_n(k)}(\mathbf{A})\widehat{\mathbf{r}}_{k-1} - \alpha_k \mathbf{A} \phi_{g_n(k)}(\mathbf{A})\widehat{\mathbf{g}}_{k-1};$
14.  $\phi_{g_n(k+1)}(\mathbf{A})\widehat{\mathbf{r}}_k = (\rho_{g_n(k+1)} \mathbf{A} + \mathbf{I}) \phi_{g_n(k)}(\mathbf{A})\widehat{\mathbf{r}}_k;$
15. For  $s = g_n(k)n, \dots, k-1$
16. 
$$\beta_s^{(k)} = -\mathbf{q}_{r_n(s+1)}^H \left( \phi_{g_n(k+1)}(\mathbf{A})\widehat{\mathbf{r}}_k + \rho_{g_n(k+1)} \sum_{t=g_n(k)n}^{s-1} \beta_t^{(k)} \mathbf{A} \phi_{g_n(t+1)}(\mathbf{A})\widehat{\mathbf{g}}_t \right) / \rho_{g_n(k+1)} \mathbf{q}_{r_n(s+1)}^H \mathbf{A} \phi_{g_n(s+1)}(\mathbf{A})\widehat{\mathbf{g}}_s;$$
17. End
18.  $\phi_{g_n(k+1)}(\mathbf{A})\widehat{\mathbf{g}}_k = \phi_{g_n(k+1)}(\mathbf{A})\widehat{\mathbf{r}}_k + (\rho_{g_n(k+1)} \mathbf{A} + \mathbf{I}) \sum_{s=g_n(k)n}^{k-1} \beta_s^{(k)} \phi_{g_n(s+1)}(\mathbf{A})\widehat{\mathbf{g}}_s;$
19. End
20. End

We remark that the term  $\phi_{g_n(t+1)+1}(\mathbf{A})\widehat{\mathbf{g}}_t$  in the first sum on Line 9 can be further written as

$$(5.4) \quad \begin{aligned} \phi_{g_n(t+1)+1}(\mathbf{A})\widehat{\mathbf{g}}_t &= (\rho_{g_n(t+1)+1} \mathbf{A} + \mathbf{I}) \phi_{g_n(t+1)}(\mathbf{A})\widehat{\mathbf{g}}_t \\ &= (\rho_{g_n(k+1)} \mathbf{A} + \mathbf{I}) \phi_{g_n(t+1)}(\mathbf{A})\widehat{\mathbf{g}}_t. \end{aligned}$$

Substituting (5.4) and (5.2) into DV#7 then yields a set of updating relations of the vectors defined by (5.2).

**Derivation Version #8.**

1. For  $k = 1, 2, \dots$ , until convergence:
2.  $\alpha_k = \mathbf{q}_{r_n(k)}^H \mathbf{r}_{k-1} / \mathbf{q}_{r_n(k)}^H \mathbf{w}_{k-1}$ ;
3. If  $r_n(k) < n$
4.  $\mathbf{r}_k = \mathbf{r}_{k-1} - \alpha_k \mathbf{w}_{k-1}$ ;
5. For  $s = \max(k-n, 0), \dots, g_n(k)n-1$
6.  $\beta_s^{(k)} = -\mathbf{q}_{r_n(s+1)}^H \left( \mathbf{r}_k + \rho_{g_n(k+1)} \sum_{t=\max(k-n, 0)}^{s-1} \beta_t^{(k)} \mathbf{w}_t \right) / \rho_{g_n(k+1)} \mathbf{q}_{r_n(s+1)}^H \mathbf{w}_s$ ;
7. End
8. For  $s = g_n(k)n, \dots, k-1$
9.  $\beta_s^{(k)} = -\mathbf{q}_{r_n(s+1)}^H \left( \mathbf{A} \mathbf{r}_k + \sum_{t=\max(k-n, 0)}^{g_n(k)n-1} \beta_t^{(k)} (\rho_{g_n(k+1)} \mathbf{A} + \mathbf{I}) \mathbf{w}_t + \sum_{t=g_n(k)n}^{s-1} \beta_t^{(k)} \mathbf{w}_t \right) / \mathbf{q}_{r_n(s+1)}^H \mathbf{w}_s$ ;
10. End
11.  $\mathbf{g}_k = \mathbf{r}_k + \rho_{g_n(k+1)} \sum_{s=\max(k-n, 0)}^{g_n(k)n-1} \beta_s^{(k)} \mathbf{w}_s + \sum_{s=\max(k-n, 0)}^{g_n(k)n-1} \beta_s^{(k)} \mathbf{g}_s + \sum_{s=g_n(k)n}^{k-1} \beta_s^{(k)} \mathbf{g}_s$ ;
12. Else
13.  $\mathbf{u}_k = \mathbf{r}_{k-1} - \alpha_k \mathbf{w}_{k-1}$ ;
14.  $\mathbf{r}_k = (\rho_{g_n(k+1)} \mathbf{A} + \mathbf{I}) \mathbf{u}_k$ ;
15. For  $s = g_n(k)n, \dots, k-1$
16.  $\beta_s^{(k)} = -\mathbf{q}_{r_n(s+1)}^H \left( \mathbf{r}_k + \rho_{g_n(k+1)} \sum_{t=g_n(k)n}^{s-1} \beta_t^{(k)} \mathbf{w}_t \right) / \rho_{g_n(k+1)} \mathbf{q}_{r_n(s+1)}^H \mathbf{w}_s$ ;
17. End
18.  $\mathbf{g}_k = \mathbf{r}_k + \rho_{g_n(k+1)} \sum_{s=g_n(k)n}^{k-1} \beta_s^{(k)} \mathbf{w}_s + \sum_{s=g_n(k)n}^{k-1} \beta_s^{(k)} \mathbf{g}_s$ ;
19. End
20. End

DV#8 does not contain any update about  $\mathbf{w}_k$ . For the updates, we multiply the equations on Lines 11 and 18 by  $\mathbf{A}$  to get

$$(5.5) \quad \mathbf{w}_k = \mathbf{A} \left( \mathbf{r}_k + \rho_{g_n(k+1)} \sum_{s=\max(k-n, 0)}^{g_n(k)n-1} \beta_s^{(k)} \mathbf{w}_s \right) + \sum_{s=\max(k-n, 0)}^{g_n(k)n-1} \beta_s^{(k)} \mathbf{w}_s + \sum_{s=g_n(k)n}^{k-1} \beta_s^{(k)} \mathbf{w}_s$$

if  $r_n(k) < n$ , and

$$(5.6) \quad \mathbf{w}_k = \mathbf{A} \left( \mathbf{r}_k + \rho_{g_n(k+1)} \sum_{s=g_n(k)n}^{k-1} \beta_s^{(k)} \mathbf{w}_s \right) + \sum_{s=g_n(k)n}^{k-1} \beta_s^{(k)} \mathbf{w}_s$$

if  $r_n(k) = n$ .

Again, we consider  $\mathbf{r}_k$  to be a residual. To be consistent with Lines 4, 13 and 14, we update the solution vector  $\mathbf{x}_k$  as

$$(5.7) \quad \mathbf{x}_k = \begin{cases} \mathbf{x}_{k-1} + \alpha_k \mathbf{g}_{k-1}, & \text{if } r_n(k) < n \\ -\rho_{g_n(k+1)} \mathbf{u}_k + \mathbf{x}_{k-1} + \alpha_k \mathbf{g}_{k-1}, & \text{if } r_n(k) = n. \end{cases}$$

Now adding (5.5), (5.6) and (5.7) to DV#8 and simplifying the operations appropriately, we then arrive at the following algorithm. The free parameter  $\rho_{g_n(k+1)}$  is chosen to minimize the 2-norm of  $\mathbf{r}_k$ .

**ALGORITHM 5.1. ML( $n$ )BiCGStab without preconditioning associated with definition (5.2)**

1. Choose an initial guess  $\mathbf{x}_0$  and  $n$  vectors  $\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n$ .
2. Compute  $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$  and  $\mathbf{g}_0 = \mathbf{r}_0$ ,  $\mathbf{w}_0 = \mathbf{A}\mathbf{g}_0$ ,  $c_0 = \mathbf{q}_1^H \mathbf{w}_0$ .
3. For  $k = 1, 2, \dots$ , until convergence:
  4.  $\alpha_k = \mathbf{q}_{r_n(k)}^H \mathbf{r}_{k-1} / c_{k-1}$ ;
  5. If  $r_n(k) < n$ 
    6.  $\mathbf{x}_k = \mathbf{x}_{k-1} + \alpha_k \mathbf{g}_{k-1}$ ;
    7.  $\mathbf{r}_k = \mathbf{r}_{k-1} - \alpha_k \mathbf{w}_{k-1}$ ;
    8.  $\mathbf{z}_w = \mathbf{r}_k$ ,  $\mathbf{g}_k = \mathbf{0}$ ;
    9. For  $s = \max(k - n, 0), \dots, g_n(k)n - 1$ 
      10.  $\beta_s^{(k)} = -\mathbf{q}_{r_n(s+1)}^H \mathbf{z}_w / (\rho_{g_n(k+1)} c_s)$ ;
      11.  $\mathbf{z}_w = \mathbf{z}_w + \rho_{g_n(k+1)} \beta_s^{(k)} \mathbf{w}_s$ ;
      12.  $\mathbf{g}_k = \mathbf{g}_k + \beta_s^{(k)} \mathbf{g}_s$ ;
    13. End
    14.  $\mathbf{g}_k = \mathbf{g}_k + \mathbf{z}_w$ ;
    15.  $\mathbf{w}_k = \mathbf{A}\mathbf{g}_k$ ;
    16. For  $s = g_n(k)n, \dots, k - 1$ 
      17.  $\beta_s^{(k)} = -\mathbf{q}_{r_n(s+1)}^H \mathbf{w}_k / c_s$ ;
      18.  $\mathbf{w}_k = \mathbf{w}_k + \beta_s^{(k)} \mathbf{w}_s$ ;
      19.  $\mathbf{g}_k = \mathbf{g}_k + \beta_s^{(k)} \mathbf{g}_s$ ;
    20. End
  21. Else
    22.  $\mathbf{x}_k = \mathbf{x}_{k-1} + \alpha_k \mathbf{g}_{k-1}$ ;
    23.  $\mathbf{u}_k = \mathbf{r}_{k-1} - \alpha_k \mathbf{w}_{k-1}$ ;
    24.  $\rho_{g_n(k+1)} = -(\mathbf{A}\mathbf{u}_k)^H \mathbf{u}_k / \|\mathbf{A}\mathbf{u}_k\|_2^2$ ;
    25.  $\mathbf{x}_k = \mathbf{x}_k - \rho_{g_n(k+1)} \mathbf{u}_k$ ;
    26.  $\mathbf{r}_k = \rho_{g_n(k+1)} \mathbf{A}\mathbf{u}_k + \mathbf{u}_k$ ;
    27.  $\mathbf{z}_w = \mathbf{r}_k$ ,  $\mathbf{g}_k = \mathbf{0}$ ;
    28. For  $s = g_n(k)n, \dots, k - 1$ 
      29.  $\beta_s^{(k)} = -\mathbf{q}_{r_n(s+1)}^H \mathbf{z}_w / (\rho_{g_n(k+1)} c_s)$ ;
      30.  $\mathbf{z}_w = \mathbf{z}_w + \rho_{g_n(k+1)} \beta_s^{(k)} \mathbf{w}_s$ ;
      31.  $\mathbf{g}_k = \mathbf{g}_k + \beta_s^{(k)} \mathbf{g}_s$ ;
    32. End
    33.  $\mathbf{g}_k = \mathbf{z}_w + \mathbf{g}_k$ ;
    34.  $\mathbf{w}_k = \mathbf{A}\mathbf{g}_k$ ;
    35. End
    36.  $c_k = \mathbf{q}_{r_n(k+1)}^H \mathbf{w}_k$ ;
    37. End

We remark that (i) the algorithm does not compute  $\mathbf{u}_k$  when  $r_n(k) < n$ . In fact,  $\mathbf{u}_k = \mathbf{r}_k$  when  $r_n(k) < n$  (see the remark right after (5.2)); (ii) if the  $\mathbf{u}_k$  on Line 23 happens to be zero, then the  $\mathbf{x}_k$  on Line 22 will be the exact solution to system (3.1) and the algorithm stops there.

The cost and storage requirement, obtained from its preconditioned version, Algorithm 9.2 in §9, are listed in Table 5.1. Compared to Algorithm 4.1, Algorithm 5.1 saves about 20% in saxpy. Since only three sets of vectors  $\{\mathbf{q}_1, \dots, \mathbf{q}_n\}$ ,

TABLE 5.1

Average cost per ( $k$ -)iteration step of Algorithm 9.2 and its storage requirement.

Preconditioning ( $\mathbf{M}^{-1}\mathbf{v}$ )	$1 + \frac{1}{n}$	Vector addition ( $\mathbf{u} \pm \mathbf{v}$ )	1
Matvec ( $\mathbf{A}\mathbf{v}$ )	$1 + \frac{1}{n}$	Saxpy ( $\mathbf{u} + \alpha\mathbf{v}$ )	$2n + 2 + \frac{2}{n}$
dot product ( $\mathbf{u}^H\mathbf{v}$ )	$n + 1 + \frac{2}{n}$	Storage	$\mathbf{A} + \mathbf{M} + (3n + 5)N + O(n)$

$\{\mathbf{g}_{k-n}, \dots, \mathbf{g}_{k-1}\}$  and  $\{\mathbf{w}_{k-n}, \dots, \mathbf{w}_{k-1}\}$  are needed in iteration  $k$ , the storage is about  $3nN$  besides storing  $\mathbf{A}$  and  $\mathbf{M}$ .

Numerical experiments have showed that Algorithm 5.1 converges faster than Algorithm 4.1 in general in terms of time. However, a disadvantage of the algorithm is that the computed relative error  $\|\mathbf{r}_k\|_2/\|\mathbf{b}\|_2$  easily diverges from the exact error  $\|\mathbf{b} - \mathbf{A}\mathbf{x}_k\|_2/\|\mathbf{b}\|_2$  (see §7), especially when  $n$  is not small.

**5.3. Properties.** We summarize the properties about Algorithm 5.1 below. Their proofs are similar to those in Proposition 4.2. Since  $\mathbf{r}_0 = \widehat{\mathbf{r}}_0$  by (5.3),  $\nu$  is also the degree of the minimal polynomial of  $\mathbf{r}_0$  with respect to  $\mathbf{A}$ .

**PROPOSITION 5.2.** *Under the assumptions of Proposition 3.2, if  $\rho_{g_n(k+1)} \neq 0$  and  $-1/\rho_{g_n(k+1)} \notin \sigma(\mathbf{A})$  for  $1 \leq k \leq \nu - 1$ , where  $\sigma(\mathbf{A})$  is the spectrum of  $\mathbf{A}$ , then Algorithm 5.1 does not break down by zero division for  $k = 1, 2, \dots, \nu$ , and the approximate solution  $\mathbf{x}_\nu$  at step  $k = \nu$  is exact to the system (3.1). Moreover, the computed quantities satisfy*

- $\mathbf{x}_k \in \mathbf{x}_0 + \text{span}\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \dots, \mathbf{A}^{g_n(k+1)+k-1}\mathbf{r}_0\}$  and  $\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k \in \mathbf{r}_0 + \text{span}\{\mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \dots, \mathbf{A}^{g_n(k+1)+k}\mathbf{r}_0\}$  for  $1 \leq k \leq \nu - 1$ .
- $\mathbf{r}_k \neq \mathbf{0}$  for  $1 \leq k \leq \nu - 1$ ;  $\mathbf{r}_\nu = \mathbf{0}$ .
- $\mathbf{r}_k \perp \text{span}\{\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_{r_n(k)}\}$  and  $\mathbf{r}_k \not\perp \mathbf{q}_{r_n(k)+1}$  for  $1 \leq k \leq \nu - 1$  with  $r_n(k) < n$ ;  $\mathbf{r}_k \not\perp \mathbf{q}_1$  for  $1 \leq k \leq \nu - 1$  with  $r_n(k) = n$ .
- $\mathbf{u}_k \perp \text{span}\{\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n\}$  for  $1 \leq k \leq \nu$  with  $r_n(k) = n$ .
- $\mathbf{A}\mathbf{g}_k \perp \text{span}\{\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_{r_n(k)}\}$  and  $\mathbf{A}\mathbf{g}_k \not\perp \mathbf{q}_{r_n(k)+1}$  for  $1 \leq k \leq \nu - 1$  with  $r_n(k) < n$ ;  $\mathbf{A}\mathbf{g}_k \not\perp \mathbf{q}_1$  for  $1 \leq k \leq \nu - 1$  with  $r_n(k) = n$ .

**6. Relations to Some Existing Methods.** In this section, we discuss the relations of ML( $n$ )BiCGStab with the FOM, BiCGStab and IDR( $n$ ) methods under the exact arithmetic environment.

### 6.1. Algorithm 4.1.

- Relation with FOM[19]. Consider the case where  $n \geq \nu$ . In this case,  $g_n(k) = 0$  and  $r_n(k) = k$  for  $k = 1, 2, \dots, \nu$ . Hence  $\mathbf{p}_k = \mathbf{q}_k$  by (3.2). If we choose  $\mathbf{q}_k = \widehat{\mathbf{r}}_{k-1}$  in Algorithm 3.1 (it is possible since  $\widehat{\mathbf{r}}_{k-1}$  is computed before  $\mathbf{q}_k$  is used), then the  $\widehat{\mathbf{x}}_k$  and  $\widehat{\mathbf{r}}_k$  computed by the algorithm satisfy

$$(6.1) \quad \begin{cases} \widehat{\mathbf{x}}_k \in \widehat{\mathbf{x}}_0 + \text{span}\{\widehat{\mathbf{r}}_0, \mathbf{A}\widehat{\mathbf{r}}_0, \dots, \mathbf{A}^{k-1}\widehat{\mathbf{r}}_0\}, \\ \widehat{\mathbf{r}}_k \perp \text{span}\{\widehat{\mathbf{r}}_0, \widehat{\mathbf{r}}_1, \dots, \widehat{\mathbf{r}}_{k-1}\} \end{cases}$$

for  $1 \leq k \leq \nu$  by Proposition 3.2(a), (d). (6.1) is exactly what the FOM approximate solution  $\mathbf{x}_k^{FOM}$  needs to satisfy. Therefore, when  $n \geq \nu$  and with the choice  $\mathbf{q}_k = \widehat{\mathbf{r}}_{k-1}$ , Algorithm 3.1 is mathematically equivalent to FOM.

Now, from (4.3), the  $\mathbf{r}_k$  computed by Algorithm 4.1 satisfies

$$\mathbf{r}_k = \phi_{g_n(k)+1}(\mathbf{A}) \widehat{\mathbf{r}}_k = \phi_1(\mathbf{A}) \widehat{\mathbf{r}}_k = (\rho_1 \mathbf{A} + \mathbf{I}) \widehat{\mathbf{r}}_k.$$

Thus, for  $1 \leq k \leq \nu$ ,  $\mathbf{r}_k$  is the factor  $\rho_1 \mathbf{A} + \mathbf{I}$  times the FOM residual  $\widehat{\mathbf{r}}_k$ .<sup>5</sup>

2. Relation with BiCGStab[30]. When  $n = 1$ , we have  $g_n(k) = k-1$  and  $r_n(k) = 1$  for  $k \in \mathcal{N}$ . Hence  $\mathbf{p}_k = (\mathbf{A}^H)^{k-1} \mathbf{q}_1$  by (3.2). By Proposition 3.2(a) and (d), the  $\widehat{\mathbf{x}}_k$  and  $\widehat{\mathbf{r}}_k$  computed by Algorithm 3.1 satisfy

$$(6.2) \quad \begin{cases} \widehat{\mathbf{x}}_k \in \widehat{\mathbf{x}}_0 + \text{span}\{\widehat{\mathbf{r}}_0, \mathbf{A}\widehat{\mathbf{r}}_0, \dots, \mathbf{A}^{k-1}\widehat{\mathbf{r}}_0\} \\ \widehat{\mathbf{r}}_k \perp \text{span}\{\mathbf{q}_1, \mathbf{A}^H \mathbf{q}_1, \dots, (\mathbf{A}^H)^{k-1} \mathbf{q}_1\} \end{cases}$$

for  $1 \leq k \leq \nu$ . (6.2) is exactly what the BiCG approximate solution  $\mathbf{x}_k^{BiCG}$  needs to satisfy. Therefore, when  $n = 1$ , Algorithm 3.1 is mathematically equivalent to BiCG.

Now, from (4.3), the  $\mathbf{r}_k$  computed by Algorithm 4.1 satisfies

$$\mathbf{r}_k = \phi_{g_n(k)+1}(\mathbf{A}) \widehat{\mathbf{r}}_k = \phi_k(\mathbf{A}) \widehat{\mathbf{r}}_k$$

which is exactly the definition of the BiCGStab residuals. Thus Algorithm 4.1 is mathematically equivalent to BiCGStab when  $n = 1$ .

3. Relation with IDR( $n$ )[29]. Write  $k = jn + i$  as in (2.1) with  $1 \leq i \leq n, 0 \leq j$ . Let  $\mathcal{G}_0 = K(\mathbf{A}, \mathbf{r}_0)$  be the complete Krylov space and let  $\mathcal{S} = \text{span}\{\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n\}^\perp$ . Define

$$\mathcal{G}_{j+1} = (\rho_{j+1} \mathbf{A} + \mathbf{I})(\mathcal{G}_j \cap \mathcal{S}) = (\rho_{g_n(k)+1} \mathbf{A} + \mathbf{I})(\mathcal{G}_j \cap \mathcal{S})$$

for  $j = 0, 1, 2, \dots$ . By (4.3), we have

$$\mathbf{r}_{jn+i} = \phi_{j+1}(\mathbf{A}) \widehat{\mathbf{r}}_{jn+i} = (\rho_{j+1} \mathbf{A} + \mathbf{I}) \phi_j(\mathbf{A}) \widehat{\mathbf{r}}_{jn+i} = (\rho_{j+1} \mathbf{A} + \mathbf{I}) \mathbf{u}_{jn+i}.$$

From Proposition 4.2(d),  $\mathbf{u}_{jn+i} \not\perp \mathbf{q}_{i+1}$  if  $i < n$ . Hence  $\mathbf{u}_{jn+i} \notin \mathcal{G}_j \cap \mathcal{S}$  and therefore  $\mathbf{r}_{jn+i} \notin \mathcal{G}_{j+1}$  when  $i < n$ . From this point of view, Algorithm 4.1 is not a IDR( $n$ ) algorithm.

However, if we regard  $\mathbf{r}_{jn+i}$  with  $1 \leq i < n$  as auxiliary vectors and instead, consider the followings as residuals

$$(6.3) \quad \mathbf{u}_{jn+1}, \dots, \mathbf{u}_{jn+n-1}, \mathbf{r}_{jn+n}$$

where  $j = 0, 1, 2, \dots$ . Then Algorithm 4.1 is a IDR( $n$ ) algorithm. In fact, by Corollary 3.3 and Proposition 3.2(d), we have

$$\begin{cases} \phi_t(\mathbf{A}) \widehat{\mathbf{r}}_{jn+i} \in \mathcal{S} & \text{if } 1 \leq i < n \text{ and } 0 \leq t < j, \\ \phi_t(\mathbf{A}) \widehat{\mathbf{r}}_{jn+n} \in \mathcal{S} & \text{if } 0 \leq t \leq j. \end{cases}$$

<sup>5</sup>In [35], a remark immediately following Theorem 4.1 states that, when  $n \geq \nu$  and with the choice that  $\mathbf{q}_1 = \phi_1(\mathbf{A}^H) \phi_1(\mathbf{A}) \mathbf{r}_0$  and  $\mathbf{q}_k = \phi_1(\mathbf{A}^H) \mathbf{r}_{k-1}$  for  $k \geq 2$ , the  $\mathbf{x}_k$  and  $\mathbf{r}_k$  computed by Algorithm 2 (which is mathematically equivalent to Algorithm 4.1 of this paper) will satisfies (6.1) and therefore Algorithm 2 is a FOM. The argument there of this remark is not correct. The author remembers that the referees of [35] were skeptical about that argument.

Thus,

$$\begin{cases} \phi_t(\mathbf{A})\widehat{\mathbf{r}}_{jn+i} \in \mathcal{G}_t \cap \mathcal{S} & \text{if } 1 \leq i < n \text{ and } 0 \leq t < j, \\ \phi_t(\mathbf{A})\widehat{\mathbf{r}}_{jn+n} \in \mathcal{G}_t \cap \mathcal{S} & \text{if } 0 \leq t \leq j. \end{cases}$$

Therefore, by (4.3),

$$(6.4) \begin{cases} \mathbf{u}_{jn+i} = \widehat{\mathbf{r}}_i \in \mathcal{G}_0 & \text{if } 1 \leq i < n, j = 0, \\ \mathbf{u}_{jn+i} = (\rho_j \mathbf{A} + \mathbf{I})\phi_{j-1}(\mathbf{A})\widehat{\mathbf{r}}_{jn+i} \in \mathcal{G}_j & \text{if } 1 \leq i < n, 1 \leq j, \\ \mathbf{r}_{jn+n} = (\rho_{j+1} \mathbf{A} + \mathbf{I})\phi_j(\mathbf{A})\widehat{\mathbf{r}}_{jn+n} \in \mathcal{G}_{j+1}. \end{cases}$$

So, the residuals in (6.3) lie in the  $\mathcal{G}$ 's.

That (6.4) holds becomes quite obvious if one applies the following result from [23] and Proposition 3.2(d),

$$(6.5) \quad \mathcal{G}_j = \{\phi_j(\mathbf{A})\mathbf{v} \mid \mathbf{v} \perp \text{span}\{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_{jn}\}\}.$$

The relation between  $\text{ML}(n)\text{BiCGStab}$  and  $\text{IDR}(n)$  was also discussed in details in [29].

## 6.2. Algorithm 5.1.

1. Relation with FOM. When  $n \geq \nu$ ,  $g_n(k) = 0$  and  $r_n(k) = k$  for  $1 \leq k \leq \nu$  and Algorithm 3.1, with the choice  $\mathbf{q}_k = \widehat{\mathbf{r}}_{k-1}$ , is a FOM algorithm as seen in §6.1. Now, from (5.2), the  $\mathbf{r}_k$  computed by Algorithm 5.1 satisfies

$$\mathbf{r}_k = \phi_{g_n(k+1)}(\mathbf{A})\widehat{\mathbf{r}}_k = \phi_0(\mathbf{A})\widehat{\mathbf{r}}_k = \widehat{\mathbf{r}}_k.$$

Thus Algorithm 5.1 is a FOM algorithm when we set  $\mathbf{q}_k = \mathbf{r}_{k-1}$ .

2. Relation with BiCGStab. When  $n = 1$ , we have  $g_n(k) = k - 1$  and  $r_n(k) = 1$  for  $k \in \mathcal{N}$  and Algorithm 3.1 is a BiCG algorithm. Now, from (5.2), the  $\mathbf{r}_k$  computed by Algorithm 5.1 satisfies

$$\mathbf{r}_k = \phi_{g_n(k+1)}(\mathbf{A})\widehat{\mathbf{r}}_k = \phi_k(\mathbf{A})\widehat{\mathbf{r}}_k$$

which is exactly the definition of the BiCGStab residuals. Thus Algorithm 5.1 is mathematically equivalent to BiCGStab.

3. Relation with IDR( $n$ ). Write  $k = jn + i$  as in (2.1) with  $1 \leq i \leq n, 0 \leq j$ . By (5.2), we have

$$\mathbf{r}_{jn+i} = \phi_{g_n(jn+i+1)}(\mathbf{A})\widehat{\mathbf{r}}_{jn+i} = \begin{cases} \phi_j(\mathbf{A})\widehat{\mathbf{r}}_{jn+i} & \text{if } 1 \leq i < n, \\ \phi_{j+1}(\mathbf{A})\widehat{\mathbf{r}}_{jn+i} & \text{if } i = n. \end{cases}$$

Thus, (6.5) and Proposition 3.2(d) yield

$$\mathbf{r}_{jn+i} \in \begin{cases} \mathcal{G}_j & \text{if } 1 \leq i < n, \\ \mathcal{G}_{j+1} & \text{if } i = n. \end{cases}$$

So, the residuals computed by Algorithm 5.1 lie in the  $\mathcal{G}$ 's and therefore it is a  $\text{IDR}(n)$  algorithm.

**7. Implementation Issues.** A preconditioned ML( $n$ )BiCGStab algorithm can be obtained by applying either Algorithm 4.1 or Algorithm 5.1 to the system

$$\mathbf{A}\mathbf{M}^{-1}\mathbf{y} = \mathbf{b}$$

where  $\mathbf{M}$  is nonsingular, then recovering  $\mathbf{x}$  through  $\mathbf{x} = \mathbf{M}^{-1}\mathbf{y}$ . The resulting algorithms, Algorithm 9.1 and Algorithm 9.2, together with their Matlab codes are presented in §9.1 and §9.2 respectively. To avoid calling the index functions  $r_n(k)$  and  $g_n(k)$  every  $k$ -iteration, we have split the  $k$ -loop into a  $i$ -loop and a  $j$ -loop where  $i, j, k$  are related by (2.1) with  $1 \leq i \leq n, 0 \leq j$ . Moreover, we have optimized the operations as possible as we can in the resulting preconditioned algorithms.

Since we have compared ML( $n$ )BiCGStab with some existing methods in [35], we will only concentrate on the performance of ML( $n$ )BiCGStab itself. The following test data were downloaded from Matrix Market.<sup>6</sup>

1. *e20r0100*, DRIVCAV Fluid Dynamics. *e20r0100* contains a  $4241 \times 4241$  real unsymmetric matrix  $\mathbf{A}$  with 131,556 nonzero entries and a real right-hand side  $\mathbf{b}$ .
2. *qc2534*, H2PLUS Quantum Chemistry, NEP Collection. *qc2534* contains a  $2534 \times 2534$  complex symmetric indefinite matrix with 463,360 nonzero entries, but does not provide the right-hand side  $\mathbf{b}$ . Following [23], we set  $\mathbf{b} = \mathbf{A}\mathbf{1}$  with  $\mathbf{1} = [1, 1, \dots, 1]^T$ .
3. *utm5940*, TOKAMAK Nuclear Physics (Plasmas). *utm5940* contains a  $5940 \times 5940$  real unsymmetric matrix  $\mathbf{A}$  with 83,842 nonzero entries and a real right-hand side  $\mathbf{b}$ .

Experiments were performed in Matlab Version 7.1 on a Windows XP machine with a Pentium 4 processor. *ILLU(0)* preconditioner (p.294, [18]) has been used in all the experiments. For *e20r0100*, the *U*-factor of the *ILLU(0)* decomposition of  $\mathbf{A}$  has some zeros along its main diagonal. In that experiment, we replaced those zeros with 1 so that the *U*-factor was invertible.

In all the experiments, initial guess  $\mathbf{x}_0 = \mathbf{0}$  and stopping criterion was

$$\|\mathbf{r}_k\|_2 / \|\mathbf{b}\|_2 < 10^{-7}$$

where  $\mathbf{r}_k$  is the computed residual. Except where specified, auxiliary vectors  $\mathbf{Q} \equiv [\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n]$  are chosen to be  $\mathbf{Q} = [\mathbf{r}_0, \text{randn}(N, n-1)]$  for *e20r0100* and *utm5940* and  $\mathbf{Q} = [\mathbf{r}_0, \text{randn}(N, n-1) + \text{sqr}t(-1) * \text{randn}(N, n-1)]$  for *qc2534*.

Moreover, for the convenience of our presentation, we introduce the following functions:

- (a)  $T_{conv}(n)$  is the time that a ML( $n$ )BiCGStab algorithm takes to converge.
- (b)  $I_{conv}(n)$  is the number of iterations that a ML( $n$ )BiCGStab algorithm takes to converge.
- (c)  $T_{iter}(n) := T_{conv}(n) / I_{conv}(n)$  is the time per iteration of a ML( $n$ )BiCGStab algorithm.
- (d)  $E(n) := \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2 / \|\mathbf{b}\|_2$  is the true relative error of  $\mathbf{x}$  where  $\mathbf{x}$  is the computed solution output by a ML( $n$ )BiCGStab algorithm when it converges.

<sup>6</sup><http://math.nist.gov/MatrixMarket/data/>

**7.1. Stability.** We plot the graphs of  $I_{conv}(n)$  in Figures 7.1(a), 7.2(a) and 7.3(a). For  $e20r0100$  and  $qc2534$ ,  $I_{conv}(n)$  decreases as  $n$  increases. However, the  $I_{conv}(n)$  for  $utm5940$  behaves very irregularly due to some of the  $\rho$ 's are too small. Recall that  $ML(n)BiCGStab$  performs  $1 + 1/n$  matrix-vector multiplications (MVs) per iteration on average. In terms of the number of MVs, both Algorithms 9.1 and 9.2 are still considerably faster than BiCGStab. BiCGStab required 455 MVs to converge.

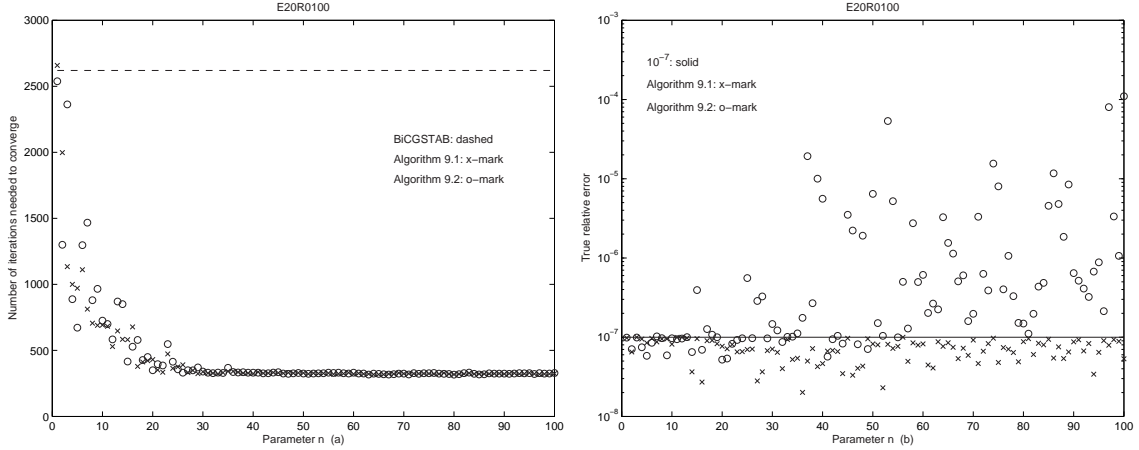


FIG. 7.1.  $e20r0100$ : (a) Graphs of  $I_{conv}(n)$  against  $n$ . BiCGStab took 2620 iterations/5240 matrix-vector multiplications (MVs) to converge. Full GMRES converged with 308 MVs; BiCGStab: dashed, Alg. 9.1: x-mark, Alg. 9.2: o-mark. (b) Graphs of  $E(n)$  against  $n$ .  $10^{-7}$ : solid, Alg. 9.1: x-mark, Alg. 9.2: o-mark.

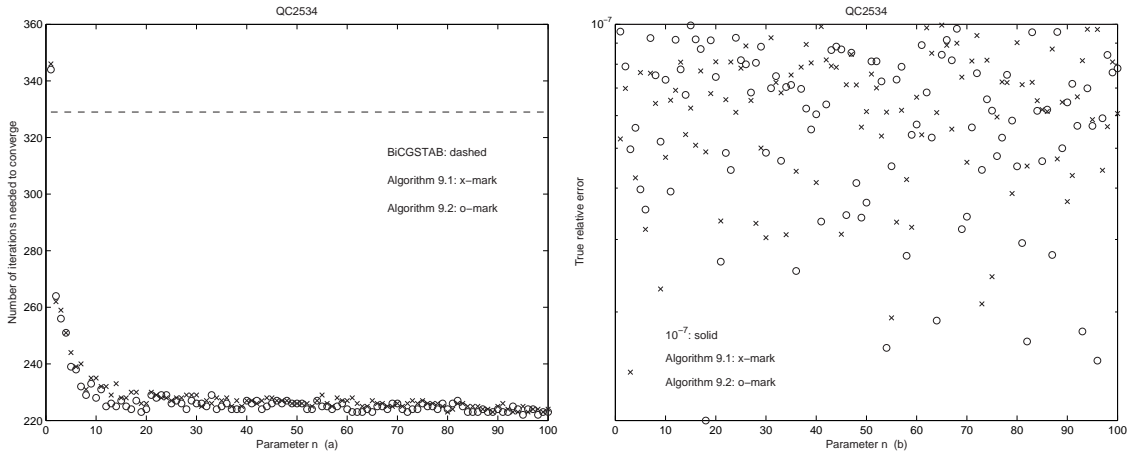


FIG. 7.2.  $qc2534$ : (a) Graphs of  $I_{conv}(n)$  against  $n$ . BiCGStab took 329 iterations/658 matrix-vector multiplications (MVs) to converge. Full GMRES converged with 439 MVs; BiCGStab: dashed, Alg. 9.1: x-mark, Alg. 9.2: o-mark. (b) Graphs of  $E(n)$  against  $n$ .  $10^{-7}$ : solid, Alg. 9.1: x-mark, Alg. 9.2: o-mark.

The graphs of  $E(n)$  are plotted in Figures 7.1(b), 7.2(b) and 7.3(b). It can be seen that the computed  $\mathbf{r}_k$  in Algorithm 9.2 easily diverges from its exact counterpart  $\mathbf{b} - \mathbf{A}\mathbf{x}_k$ . This divergence becomes significant when  $n \geq 15$  for  $e20r0100$  and  $n \geq 4$

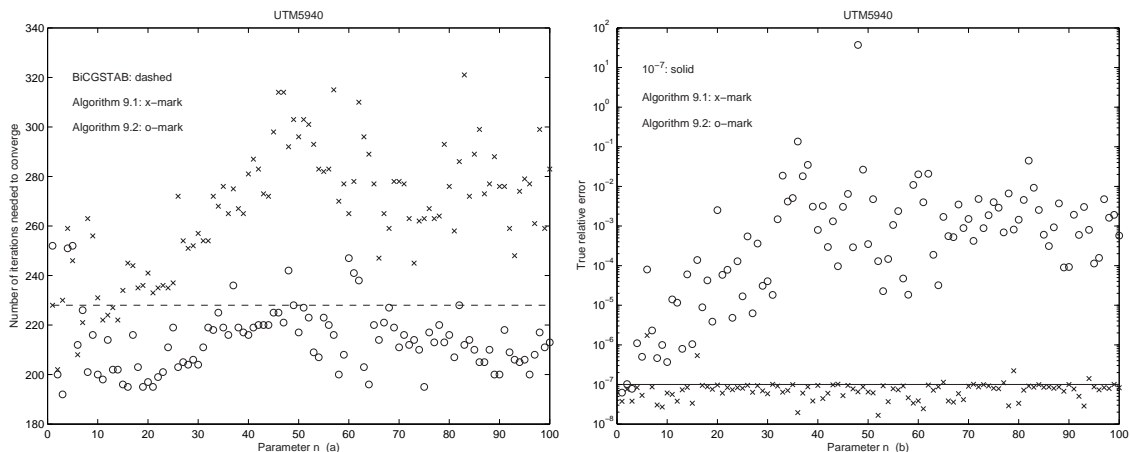


FIG. 7.3. *utm5940*: (a) Graphs of  $I_{conv}(n)$  against  $n$ . BiCGStab took 228 iterations/455 matrix-vector multiplications (MVs) to converge. Full GMRES converged with 176 MVs; BiCGStab: dashed, Alg. 9.1: x-mark, Alg. 9.2: o-mark. (b) Graphs of  $E(n)$  against  $n$ .  $10^{-7}$ : solid, Alg. 9.1: x-mark, Alg. 9.2: o-mark.

for *utm5940*. By contrast, the computed relative errors  $\|\mathbf{r}_k\|_2/\|\mathbf{b}\|_2$  by Algorithm 9.1 well approximate their corresponding true ones. Thus, from this point of view, we consider that Algorithm 9.1 is numerically more stable than Algorithm 9.2. We remark that the issues of divergence of computed residuals and corresponding remedy techniques have been discussed in details in [26, 32].

**7.2. Choice of  $n$ .** In this and the following subsections, we will focus on Algorithm 9.1.

ML( $n$ )BiCGStab has a free parameter  $n$ . It is ideal that  $n$  is chosen such that both  $T_{iter}(n)$  and  $I_{conv}(n)$  are the smallest. Since, however, it is difficult to predict the overall behavior of ML( $n$ )BiCGStab on a problem at the point of interest, minimization of  $I_{conv}(n)$  is almost impossible. In fact, there are some cases where ML( $n$ )BiCGStab does not converge for all the  $n$ 's. In the following, we therefore only make an effort on the minimization of  $T_{iter}(n)$ .

A ML( $n$ )BiCGStab algorithm mainly involves three types of operations: preconditioner system solving and matrix-vector multiplication ( $\mathbf{A}\mathbf{M}^{-1}\mathbf{v}$ ), dot product ( $\mathbf{u}^H\mathbf{v}$ ), and saxpy operation ( $\mathbf{u} + \alpha\mathbf{v}$ ). The algorithm requires  $1 + 1/n$  of  $\mathbf{A}\mathbf{M}^{-1}\mathbf{v}$  per iteration on average. For smaller values of  $n$ , the overall computational effort is dominated by  $\mathbf{A}\mathbf{M}^{-1}\mathbf{v}$ , and as  $n$  is increased, the computations of  $\mathbf{u}^H\mathbf{v}$  and  $\mathbf{u} + \alpha\mathbf{v}$  will start to take dominant. Therefore, the graph of  $T_{iter}(n)$  should have a parabolic shape, as illustrated in Figures 7.4(a), 7.5(a), 7.6(a). As a result, the minimizer  $n_{opt}$  of  $T_{iter}(n)$  always exists. For a comparison, we also plot the graphs of  $T_{conv}(n)$  in Figures 7.4(b), 7.5(b) and 7.6(b). It is clear that  $n_{opt}$  does not necessarily minimize  $T_{conv}(n)$ . In fact, there is no absolute connection between  $n_{opt}$  and the minimization of  $T_{conv}(n)$ .

No general formula is available for determining  $n_{opt}$  because  $n_{opt}$  depends on the problem, the coding language, the computer and etc.. In §9.1, we present a Matlab function *time\_per\_iter.m* which estimates the value of  $T_{iter}(n)$  at any given  $n$ . Note that one  $j$ -loop of Algorithm 9.1 contains  $n$  iterations — the algorithm computes  $\mathbf{x}_{jn+1}, \mathbf{x}_{jn+2}, \dots, \mathbf{x}_{jn+n}$  in a  $j$ -loop. In more detailed, Lines 4-10, 60-75

TABLE 7.1

Performance of *find\_n.m* plus Algorithm 9.1. In this experiment, we set  $n\_step = 10$  and  $n\_max = 10 * n\_step$  for *find\_n.m*. The 3rd column is the number of nonzero entries in  $\mathbf{A}$ , the 4th is the  $n$  found by *find\_n.m*, and the next four columns are the times and MVs required by the algorithms to converge.

Title	Size ( $N$ )	# non-zeros	<i>find_n.m</i> + Alg. 9.1			BiCGStab	
			$n$	Time (secs)	MVs	Time (secs)	MVs
e20r0100	4241	131,556	12	3.36	16 + 574	25.69	5240
qc2534	2534	463,360	12	5.05	16 + 252	11.04	658
utm5940	5940	83,842	12	1.56	16 + 243	2.22	455

compute  $\mathbf{x}_{jn+1}$  with two  $\mathbf{AM}^{-1}\mathbf{v}$  and Lines 12 - 58 compute  $\mathbf{x}_{jn+2}, \dots, \mathbf{x}_{jn+n}$  with one  $\mathbf{AM}^{-1}\mathbf{v}$  for each of them. In *time\_per\_iter.m*, the operation times for running Lines 4-10, 60-75, 12-58 with  $i = 1$  and 12-58 with  $i = n - 1$  are stored in  $t\_11, t\_12, t\_2$  and  $t\_n$  respectively. Thus, the time to compute  $\mathbf{x}_{jn+1}$  is  $t\_11 + t\_12$  and the times to  $\mathbf{x}_{jn+2}$  and  $\mathbf{x}_{jn+n}$  are  $t\_2$  and  $t\_n$  respectively. *time\_per\_iter.m* then takes the average  $(t\_2 + t\_n)/2$  as a guess to the time of computing each of  $\mathbf{x}_{jn+2}, \dots, \mathbf{x}_{jn+n}$ . So, the time to complete one  $j$ -loop is about  $t\_11 + t\_12 + 0.5(n - 1)(t\_2 + t\_n)$  and therefore the time for each iteration is roughly  $[t\_11 + t\_12 + 0.5(n - 1)(t\_2 + t\_n)]/n$ . The function *time\_per\_iter.m* requires four  $\mathbf{AM}^{-1}\mathbf{v}$  to implement.

Finding the minimizer  $n_{opt}$  of  $T_{iter}(n)$  is an one-dimensional minimization problem. Any optimization technique for one-dimensional problems can be applied. The function *find\_n.m* in §9.3 takes a simple idea. It searches for  $n_{opt}$  in an user-provided interval  $[1, n\_max]$ . *find\_n.m* first constructs an interval  $[n1, n4]$  that contains  $n_{opt}$ . It then picks two points  $n2, n3$  from  $[n1, n4]$  and computes the values of  $T_{iter}$  at the points  $n1, n2, n3$  and  $n4$  by *time\_per\_iter.m*. Then, *find\_n.m* fits a quadratic polynomial  $p(t)$  to the four data points  $\{(n1, T_{iter}(n1)), \dots, (n4, T_{iter}(n4))\}$  in the least-squares sense. The polynomial  $p(t)$  has a minimum in the interval  $[n1, n4]$ . This minimum, rounded to an integer, is the output of *find\_n.m* as an estimate of  $n_{opt}$ .

We can combine *find\_n.m* and a  $ML(n)$ BiCGStab algorithm to form a new algorithm. The performance of this new algorithm is demonstrated in Table 7.1. Also, a sample implementation is presented in §9.4.

Finally, we remark that Algorithm 9.2 is faster than Algorithm 9.1 in our experiments in terms of time in general.

**7.3. Choice of  $\rho$ .** The standard choice for the  $\rho_{j+1}$  in Algorithm 9.1 is

$$(7.1) \quad \rho_{j+1} = -(\mathbf{A}\tilde{\mathbf{u}}_{jn+1})^H \mathbf{u}_{jn+1} / \|\mathbf{A}\tilde{\mathbf{u}}_{jn+1}\|_2^2.$$

This choice of  $\rho_{j+1}$  minimizes the 2-norm of  $\mathbf{r}_{jn+1} = \rho_{j+1}\mathbf{A}\tilde{\mathbf{u}}_{jn+1} + \mathbf{u}_{jn+1}$ , but sometimes can cause instability due to that it can be very small during an implementation. A remedy as follows has been suggested in [24]:

$$(7.2) \quad \begin{aligned} \rho_{j+1} &= -(\mathbf{A}\tilde{\mathbf{u}}_{jn+1})^H \mathbf{u}_{jn+1} / \|\mathbf{A}\tilde{\mathbf{u}}_{jn+1}\|_2^2; \\ \omega &= (\mathbf{A}\tilde{\mathbf{u}}_{jn+1})^H \mathbf{u}_{jn+1} / (\|\mathbf{A}\tilde{\mathbf{u}}_{jn+1}\|_2 \|\mathbf{u}_{jn+1}\|_2); \\ \text{if } |\omega| < \kappa, \quad \rho_{j+1} &= \kappa \rho_{j+1} / |\omega|; \text{ end} \end{aligned}$$

where  $\kappa$  is a user-defined parameter. In Figures 7.7(a)(b), we compare the performances of Algorithm 9.1 with (7.1) and (7.2) respectively (we only plot the results of *qc2534* and *utm5940*. The result of *e20r0100* with  $\kappa = 0.1$  is similar to Figure 7.7(a)). Also, see the numerical experiments in [29] for more information about these  $\rho$  choices.

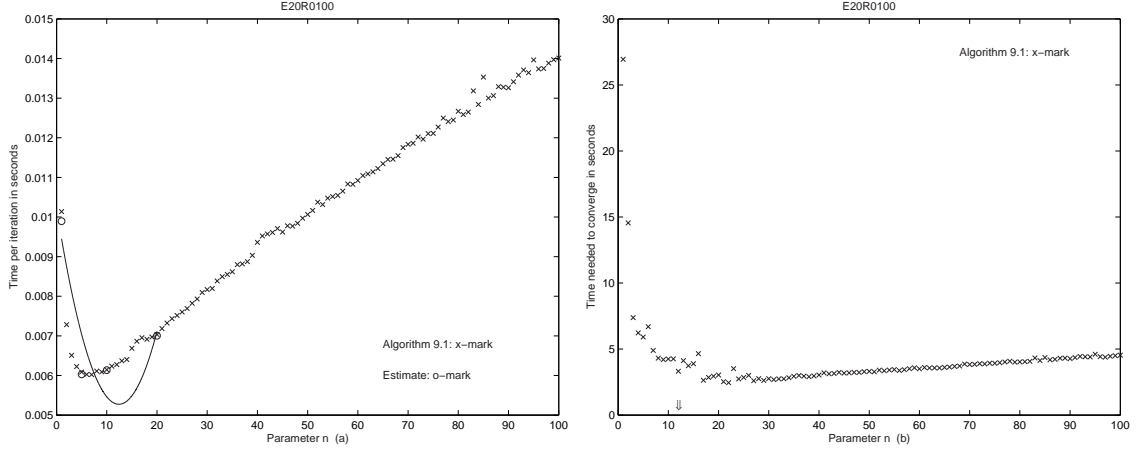


FIG. 7.4. *e20r0100*: (a) Graph of  $T_{iter}(n)$  of Algorithm 9.1 against  $n$ .  $T_{iter}(n)$  reaches its minimum at  $n = 7$ . The *o*-marks are the estimates of the values of  $T_{iter}(n)$  at  $n = 1, 5, 10, 20$  obtained by *time\_per\_iter.m*. The solid curve is the graph of the quadratic polynomial that fits the *o*-marks in the least-squares sense. The minimizer of the polynomial, rounded to an integer, is the estimate of  $n_{opt}$  output by *find\_n.m*; Alg. 9.1: *x*-mark, estimate: *o*-mark. (b) Graph of  $T_{conv}(n)$  of Algorithm 9.1 against  $n$ .  $T_{conv}(n)$  reaches its minimum at  $n = 22$ .  $\Downarrow$  in the figure marks the estimate of  $n_{opt}$  obtained by *find\_n.m*.

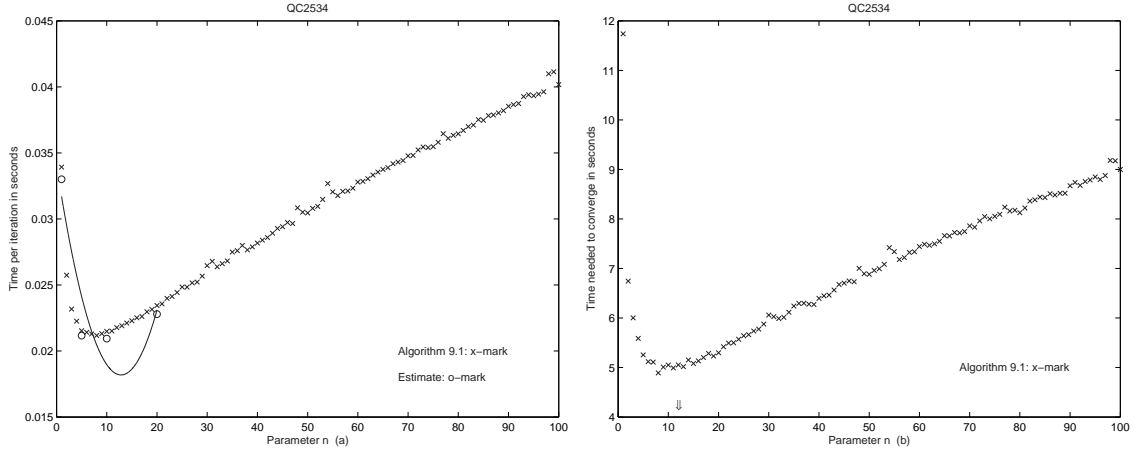


FIG. 7.5. *qc2534*: (a) Graph of  $T_{iter}(n)$  of Algorithm 9.1 against  $n$ .  $T_{iter}(n)$  reaches its minimum at  $n = 8$ . The *o*-marks are the estimates of the values of  $T_{iter}(n)$  at  $n = 1, 5, 10, 20$  obtained by *time\_per\_iter.m*. The solid curve is the graph of the quadratic polynomial that fits the *o*-marks in the least-squares sense. The minimizer of the polynomial, rounded to an integer, is the estimate of  $n_{opt}$  output by *find\_n.m*; Alg. 9.1: *x*-mark, estimate: *o*-mark. (b) Graph of  $T_{conv}(n)$  of Algorithm 9.1 against  $n$ .  $T_{conv}(n)$  reaches its minimum at  $n = 8$ .  $\Downarrow$  in the figure marks the estimate of  $n_{opt}$  obtained by *find\_n.m*.

**7.4. Choice of  $\mathbf{q}$ 's.** We usually pick  $\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n]$  as

$$(7.3) \quad \mathbf{Q} = [\mathbf{r}_0, \text{randn}(N, n - 1)]$$

for a real problem and

$$(7.4) \quad \mathbf{Q} = [\mathbf{r}_0, \text{randn}(N, n - 1) + \text{sqrt}(-1) * \text{randn}(N, n - 1)]$$

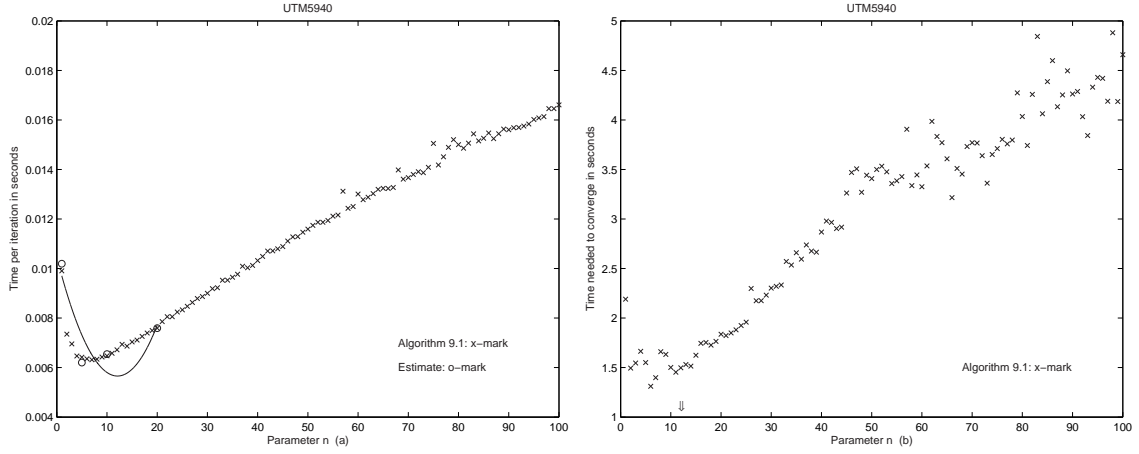


FIG. 7.6. *utm5940*: (a) Graph of  $T_{iter}(n)$  of Algorithm 9.1 against  $n$ .  $T_{iter}(n)$  reaches its minimum at  $n = 7$ . The *o*-marks are the estimates of the values of  $T_{iter}(n)$  at  $n = 1, 5, 10, 20$  obtained by *time\_per\_iter.m*. The solid curve is the graph of the quadratic polynomial that fits the *o*-marks in the least-squares sense. The minimizer of the polynomial, rounded to an integer, is the estimate of  $n_{opt}$  output by *find\_n.m*; Alg. 9.1: *x*-mark, estimate: *o*-mark. (b) Graph of  $T_{conv}(n)$  of Algorithm 9.1 against  $n$ .  $T_{conv}(n)$  reaches its minimum at  $n = 6$ . ↓ in the figure marks the estimate of  $n_{opt}$  obtained by *find\_n.m*.

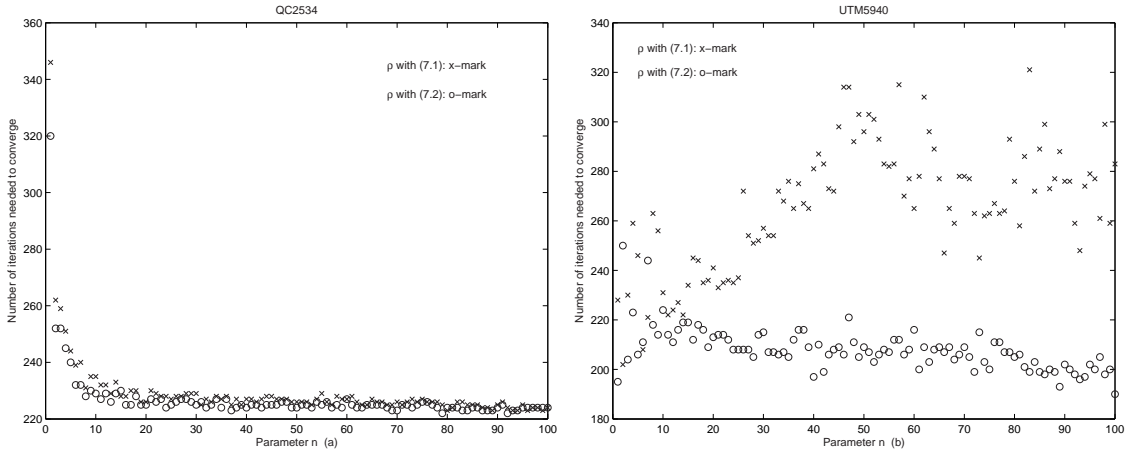


FIG. 7.7. (a) *qc2534*: Graphs of  $I_{conv}(n)$  of Algorithm 9.1 against  $n$  with choices (7.1) and (7.2) for  $\rho$  respectively. In this experiment, we picked  $\kappa = 0.7$ .  $\rho$  with (7.1): *x*-mark,  $\rho$  with (7.2): *o*-mark. (b) *utm5940*: Graphs of  $I_{conv}(n)$  of Algorithm 9.1 against  $n$  with choices (7.1) and (7.2) for  $\rho$  respectively. In this experiment, we picked  $\kappa = 0.7$ .  $\rho$  with (7.1): *x*-mark,  $\rho$  with (7.2): *o*-mark.

for a complex problem. In our experiments, however, we observed a comparable performance when we chose

$$(7.5) \quad \mathbf{Q} = [\mathbf{r}_0, \text{sign}(\text{randn}(N, n - 1))].$$

or

$$(7.6) \quad \mathbf{Q} = [\mathbf{r}_0, \text{sign}(\text{randn}(N, n - 1)) + \text{sqrt}(-1) * \text{sign}(\text{randn}(N, n - 1))].$$

See Figures 7.8(a)(b) (we only plot the results of  $qc2534$  and  $utm5940$ . The result of  $e20r0100$  is similar to Figure 7.8(a)).

The advantages of (7.5) and (7.6) over (7.3) and (7.4) are that (i) the storage of  $\mathbf{Q}$  is substantially reduced. In fact, we just need to store the random signs (except its first column); (ii) an inner product with  $\mathbf{q}_i$ ,  $2 \leq i \leq n$ , is now reduced to a sum without involving scalar multiplications.

For other choices for  $\mathbf{Q}$ , one is referred to [29].

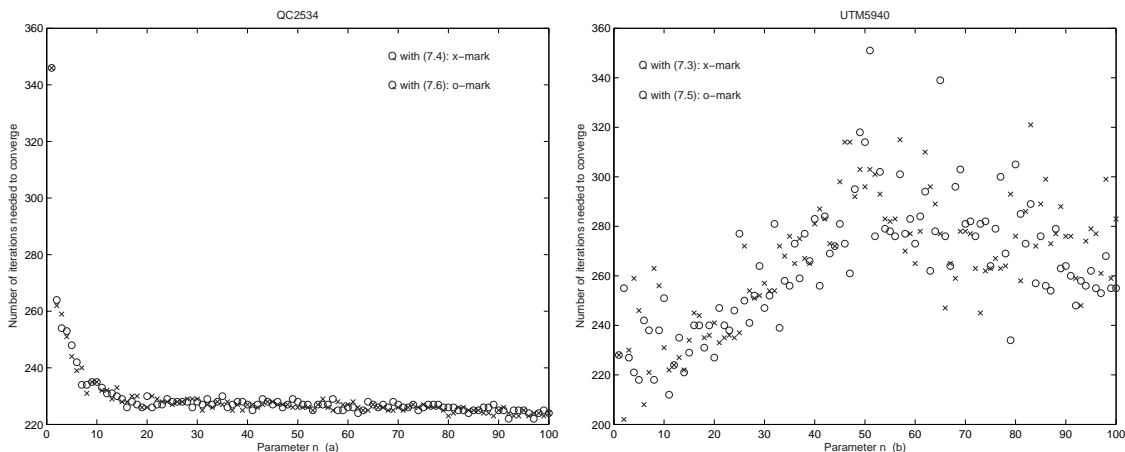


FIG. 7.8. (a)  $qc2534$ : Graphs of  $I_{conv}(n)$  of Algorithm 9.1 against  $n$  with choices (7.4) and (7.6) for  $\mathbf{Q}$  respectively.  $\mathbf{Q}$  with (7.4): x-mark,  $\mathbf{Q}$  with (7.6): o-mark. (b)  $utm5940$ : Graphs of  $I_{conv}(n)$  of Algorithm 9.1 against  $n$  with choices (7.3) and (7.5) for  $\mathbf{Q}$  respectively.  $\mathbf{Q}$  with (7.3): x-mark,  $\mathbf{Q}$  with (7.5): o-mark.

**8. Conclusions and Remarks.** With the help of index functions, we re-derived the ML( $n$ )BiCGStab algorithm in [35] in a more systematic way. We believe the derivation here is more transparent. Indeed, we have been able to find out some redundant operations existing in the algorithm in [35] and realized that there are at least  $n$  ways to define the ML( $n$ )BiCGStab residuals  $\mathbf{r}_k$ . We believe a different definition of  $\mathbf{r}_k$  will lead to a different algorithm. We demonstrated two definitions together with the derivations of their associated algorithms, namely, (i) definition (4.3) that increases the degree of  $\phi$  at the beginning of an iteration cycle and results in Algorithm 4.1; (ii) definition (5.2) with the degree of  $\phi$  increased at the end of a cycle, resulting in Algorithm 5.1. A comparison of the two algorithms showed that Algorithm 5.1 is cheaper in storage and computational cost, faster to converge, but the computed residuals  $\mathbf{r}_k$  can easily diverge from their corresponding true residuals, except that  $n$  is small. For a definition of  $\mathbf{r}_k$  that increases the degree of  $\phi$  somewhere within a cycle, we expect that the cost, storage and performance of the associated algorithm would lie between those of Algorithms 4.1 and 5.1.

Based on the experiments in this paper, we observed that there are two situations on which ML( $n$ )BiCGStab may be able to improve BiCGStab in terms of time: (i) BiCGStab behaves less stable on a problem. As  $n$  is increased, the stability of ML( $n$ )BiCGStab (especially Algorithms 4.1) will increase; (ii) the time to compute a  $\mathbf{A}\mathbf{M}^{-1}\mathbf{v}$  is significantly greater than the time to compute a dot product and a saxpy. In this case, one can increase  $n$  appropriately to reduce the number of  $\mathbf{A}\mathbf{M}^{-1}\mathbf{v}$  per iteration in ML( $n$ )BiCGStab. On the other hand, however, ML( $n$ )BiCGStab reduces

the residuals only once every  $n$  iterations. This could be a disadvantage in an application. When this happened, small  $n$  should be suggested.

In §7, we discussed in details on the choices of  $n$ ,  $\rho$  and  $\mathbf{q}_1, \dots, \mathbf{q}_n$ . We also summarized the properties of  $\text{ML}(n)\text{BiCGStab}$  and discussed its relations to some existing methods. In the case where (3.1) is a real system,  $n = 1$ ,  $\mathbf{q}_1$  is random with iid elements from  $N(0, 1)$  and the free parameter  $\rho$  is selected to be nonzero, we proved that  $\text{ML}(n)\text{BiCGStab}$  (or equivalently,  $\text{BiCGStab}$  in this case) almost surely works without breakdown by zero division to find a solution from the affine space  $\mathbf{x}_0 + \text{span}\{\mathbf{A}^t \mathbf{r}_0 | t \in \mathcal{N}_0\}$  provided that  $\mathbf{x}_0$  is chosen such that the affine space contains a solution to (3.1)<sup>7</sup>.

**9. Appendix.** In this section, we present preconditioned  $\text{ML}(n)\text{BiCGStab}$  algorithms and related Matlab codes.

**9.1.  $\text{ML}(n)\text{BiCGStab}$  Associated with (4.3) and Related Codes.** The following algorithm is a preconditioned version of Algorithm 4.1.

**ALGORITHM 9.1.  $\text{ML}(n)\text{BiCGStab}$  with preconditioning associated with definition (4.3).**

1. Choose an initial guess  $\mathbf{x}_0$  and  $n$  vectors  $\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n$ .
2. Compute  $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$  and set  $\mathbf{g}_0 = \mathbf{r}_0$ .  
Compute  $\tilde{\mathbf{g}}_0 = \mathbf{M}^{-1}\mathbf{g}_0$ ,  $\mathbf{w}_0 = \mathbf{A}\tilde{\mathbf{g}}_0$ ,  $c_0 = \mathbf{q}_1^H \mathbf{w}_0$  and  $e_0 = \mathbf{q}_1^H \mathbf{r}_0$ .
3. For  $j = 0, 1, 2, \dots$
4.  $\alpha_{jn+1} = e_{(j-1)n+n} / c_{(j-1)n+n}$ ;
5.  $\mathbf{u}_{jn+1} = \mathbf{r}_{(j-1)n+n} - \alpha_{jn+1} \mathbf{w}_{(j-1)n+n}$ ;
6.  $\mathbf{x}_{jn+1} = \mathbf{x}_{(j-1)n+n} + \alpha_{jn+1} \tilde{\mathbf{g}}_{(j-1)n+n}$ ;
7.  $\tilde{\mathbf{u}}_{jn+1} = \mathbf{M}^{-1} \mathbf{u}_{jn+1}$ ;
8.  $\rho_{j+1} = -(\mathbf{A}\tilde{\mathbf{u}}_{jn+1})^H \mathbf{u}_{jn+1} / \|\mathbf{A}\tilde{\mathbf{u}}_{jn+1}\|_2^2$ ;
9.  $\mathbf{x}_{jn+1} = \mathbf{x}_{jn+1} - \rho_{j+1} \tilde{\mathbf{u}}_{jn+1}$ ;
10.  $\mathbf{r}_{jn+1} = \rho_{j+1} \mathbf{A}\tilde{\mathbf{u}}_{jn+1} + \mathbf{u}_{jn+1}$ ;
11. For  $i = 1, 2, \dots, n-1$
12.  $f_{jn+i} = \mathbf{q}_{i+1}^H \mathbf{u}_{jn+i}$ ;
13. If  $j \geq 1$
14.  $\beta_{(j-1)n+i}^{(jn+i)} = -f_{jn+i} / c_{(j-1)n+i}$ ;
15. If  $i \leq n-2$
16.  $\mathbf{z}_d = \mathbf{u}_{jn+i} + \beta_{(j-1)n+i}^{(jn+i)} \mathbf{d}_{(j-1)n+i}$ ;
17.  $\mathbf{g}_{jn+i} = \beta_{(j-1)n+i}^{(jn+i)} \mathbf{g}_{(j-1)n+i}$ ;
18.  $\mathbf{z}_w = \beta_{(j-1)n+i}^{(jn+i)} \mathbf{w}_{(j-1)n+i}$ ;
19.  $\beta_{(j-1)n+i+1}^{(jn+i)} = -\mathbf{q}_{i+2}^H \mathbf{z}_d / c_{(j-1)n+i+1}$ ;
20. For  $s = i+1, \dots, n-2$
21.  $\mathbf{z}_d = \mathbf{z}_d + \beta_{(j-1)n+s}^{(jn+i)} \mathbf{d}_{(j-1)n+s}$ ;
22.  $\mathbf{g}_{jn+i} = \mathbf{g}_{jn+i} + \beta_{(j-1)n+s}^{(jn+i)} \mathbf{g}_{(j-1)n+s}$ ;
23.  $\mathbf{z}_w = \mathbf{z}_w + \beta_{(j-1)n+s}^{(jn+i)} \mathbf{w}_{(j-1)n+s}$ ;
24.  $\beta_{(j-1)n+s+1}^{(jn+i)} = -\mathbf{q}_{s+2}^H \mathbf{z}_d / c_{(j-1)n+s+1}$ ;
25. End

<sup>7</sup> $p_{\min}(0, \mathbf{A}, \mathbf{r}_0) \neq 0$  iff the affine space  $\mathbf{x}_0 + \text{span}\{\mathbf{A}^t \mathbf{r}_0 | t \in \mathcal{N}_0\}$  contains a solution to (3.1). For an example where the affine space contains no solution, consider  $\mathbf{A} = [0, 1; 0, 0]$ ,  $\mathbf{b} = [1, 0]^T$  and  $\mathbf{x}_0 = [1, 0]^T$ .

```

26.       $\mathbf{g}_{jn+i} = \mathbf{g}_{jn+i} + \beta_{(j-1)n+n-1}^{(jn+i)} \mathbf{g}_{(j-1)n+n-1};$ 
27.       $\mathbf{z}_w = \mathbf{z}_w + \beta_{(j-1)n+n-1}^{(jn+i)} \mathbf{W}_{(j-1)n+n-1};$ 
28.       $\mathbf{z}_w = \mathbf{r}_{jn+i} + \rho_{j+1} \mathbf{z}_w;$ 
29.      Else
30.           $\mathbf{g}_{jn+i} = \beta_{(j-1)n+n-1}^{(jn+i)} \mathbf{g}_{(j-1)n+n-1};$ 
31.           $\mathbf{z}_w = \mathbf{r}_{jn+i} + \rho_{j+1} \beta_{(j-1)n+n-1}^{(jn+i)} \mathbf{W}_{(j-1)n+n-1};$ 
32.      End
33.       $\beta_{(j-1)n+n}^{(jn+i)} = -\mathbf{q}_1^H \mathbf{z}_w / (\rho_{j+1} c_{(j-1)n+n});$ 
34.       $\mathbf{z}_w = \mathbf{z}_w + \rho_{j+1} \beta_{(j-1)n+n}^{(jn+i)} \mathbf{W}_{(j-1)n+n};$ 
35.       $\mathbf{g}_{jn+i} = \mathbf{g}_{jn+i} + \mathbf{z}_w + \beta_{(j-1)n+n}^{(jn+i)} \mathbf{g}_{(j-1)n+n};$ 
36.      Else
37.           $\beta_{(j-1)n+n}^{(jn+i)} = -\mathbf{q}_1^H \mathbf{r}_{jn+i} / (\rho_{j+1} c_{(j-1)n+n});$ 
38.           $\mathbf{z}_w = \mathbf{r}_{jn+i} + \rho_{j+1} \beta_{(j-1)n+n}^{(jn+i)} \mathbf{W}_{(j-1)n+n};$ 
39.           $\mathbf{g}_{jn+i} = \mathbf{z}_w + \beta_{(j-1)n+n}^{(jn+i)} \mathbf{g}_{(j-1)n+n};$ 
40.      End
41.      For  $s = 1, \dots, i - 1$ 
42.           $\beta_{jn+s}^{(jn+i)} = -\mathbf{q}_{s+1}^H \mathbf{z}_w / c_{jn+s};$ 
43.           $\mathbf{g}_{jn+i} = \mathbf{g}_{jn+i} + \beta_{jn+s}^{(jn+i)} \mathbf{g}_{jn+s};$ 
44.           $\mathbf{z}_w = \mathbf{z}_w + \beta_{jn+s}^{(jn+i)} \mathbf{d}_{jn+s};$ 
45.      End
46.      If  $i < n - 1$ 
47.           $\mathbf{d}_{jn+i} = \mathbf{z}_w - \mathbf{u}_{jn+i};$ 
48.           $c_{jn+i} = \mathbf{q}_{i+1}^H \mathbf{d}_{jn+i};$ 
49.           $\tilde{\alpha}_{jn+i+1} = f_{jn+i} / c_{jn+i};$           %  $\tilde{\alpha}_{jn+i+1} = \alpha_{jn+i+1} / \rho_{j+1}$ 
50.           $\mathbf{u}_{jn+i+1} = \mathbf{u}_{jn+i} - \tilde{\alpha}_{jn+i+1} \mathbf{d}_{jn+i};$ 
51.      Else
52.           $c_{jn+i} = \mathbf{q}_{i+1}^H (\mathbf{z}_w - \mathbf{u}_{jn+i});$ 
53.           $\tilde{\alpha}_{jn+i+1} = f_{jn+i} / c_{jn+i};$           %  $\tilde{\alpha}_{jn+i+1} = \alpha_{jn+i+1} / \rho_{j+1}$ 
54.      End
55.       $\tilde{\mathbf{g}}_{jn+i} = \mathbf{M}^{-1} \mathbf{g}_{jn+i};$ 
56.       $\mathbf{w}_{jn+i} = \mathbf{A} \tilde{\mathbf{g}}_{jn+i};$ 
57.       $\mathbf{x}_{jn+i+1} = \mathbf{x}_{jn+i} + \rho_{j+1} \tilde{\alpha}_{jn+i+1} \tilde{\mathbf{g}}_{jn+i};$ 
58.       $\mathbf{r}_{jn+i+1} = \mathbf{r}_{jn+i} - \rho_{j+1} \tilde{\alpha}_{jn+i+1} \mathbf{w}_{jn+i};$ 
59.      End
60.       $e_{jn+n} = \mathbf{q}_1^H \mathbf{r}_{jn+n};$ 
61.       $\beta_{(j-1)n+n}^{(jn+n)} = -e_{jn+n} / (\rho_{j+1} c_{(j-1)n+n});$ 
62.       $\mathbf{z}_w = \mathbf{r}_{jn+n} + \rho_{j+1} \beta_{(j-1)n+n}^{(jn+n)} \mathbf{W}_{(j-1)n+n};$ 
63.       $\mathbf{g}_{jn+n} = \mathbf{z}_w + \beta_{(j-1)n+n}^{(jn+n)} \mathbf{g}_{(j-1)n+n};$ 
64.      If  $n \geq 2$ 
65.           $\beta_{jn+1}^{(jn+n)} = -\mathbf{q}_2^H \mathbf{z}_w / c_{jn+1};$ 
66.          For  $s = 1, \dots, n - 2$ 
67.               $\mathbf{g}_{jn+n} = \mathbf{g}_{jn+n} + \beta_{jn+s}^{(jn+n)} \mathbf{g}_{jn+s};$ 
68.               $\mathbf{z}_w = \mathbf{z}_w + \beta_{jn+s}^{(jn+n)} \mathbf{d}_{jn+s};$ 
69.               $\beta_{jn+s+1}^{(jn+n)} = -\mathbf{q}_{s+2}^H \mathbf{z}_w / c_{jn+s+1};$ 

```

70. *End*  
71.  $\mathbf{g}_{jn+n} = \mathbf{g}_{jn+n} + \beta_{jn+n-1}^{(jn+n)} \mathbf{g}_{jn+n-1};$   
72. *End*  
73.  $\tilde{\mathbf{g}}_{jn+n} = \mathbf{M}^{-1} \mathbf{g}_{jn+n};$   
74.  $\mathbf{w}_{jn+n} = \mathbf{A} \tilde{\mathbf{g}}_{jn+n};$   
75.  $c_{jn+n} = \mathbf{q}_1^H \mathbf{w}_{jn+n};$   
76. *End*

A sample Matlab code of Algorithm 9.1 is provided as follows.

**Code #1: Matlab code of Algorithm 9.1**

```

1. function [x, err, iter, flag] = mlbicgstab(A, x, b, Q, M, max_it, tol)
2.
3. % input:  A    N-by-N matrix
4. %        M    N-by-N preconditioner matrix
5. %        Q    N-by-n auxiliary matrix [q1, ..., qn]
6. %        x    initial guess
7. %        b    right hand side vector
8. %        max_it (integer) maximum number of iterations
9. %        tol   error tolerance
10. % output: x     solution computed
11. %        err   error norm
12. %        iter  (integer) number of iterations performed
13. %        flag  (integer): 0 = solution found to tolerance
14. %                1 = no convergence given max_it
15. %                -1 = breakdown
16. % storage: D      N x (n - 2) matrix defined only when n > 2
17. %        G, Q, W   N x n matrices
18. %        A, M      N x N matrices
19. %        x, r, g_t, u, u_t, z, b --- N x 1 matrices
20. %        c         1 x n matrix
21.
22. N = size(A, 2); n = size(Q, 2);
23. G = zeros(N, n); W = zeros(N, n); % initialize workspace for d, g, w and c
24. if n > 2, D = zeros(N, n - 2); end
25. c = zeros(1, n); % end initialization
26.
27. iter = 0; flag = 1; bnorm2 = norm(b);
28. if bnorm2 == 0.0, bnorm2 = 1.0; end
29.
30. r = b - A * x; err = norm(r)/bnorm2;
31. if err < tol, flag = 0; return, end
32.
33. G(:, n) = r; g_t = M \ r; W(:, n) = A * g_t;
34. c(n) = Q(:, 1)' * W(:, n);
35. if c(n) == 0, flag = -1; return, end
36. e = Q(:, 1)' * r;
37.
38. for j = 0 : max_it
39.     alphas = e/c(n);
40.     x = x + alphas * g_t;

```

```

41.  $u = r - \text{alpha} * W(:, n);$ 
42.  $\text{err} = \text{norm}(u) / \text{bnrm2};$ 
43. if  $\text{err} < \text{tol}$ ,  $\text{flag} = 0$ ;  $\text{iter} = \text{iter} + 1$ ; return, end
44.
45.  $u.t = M \setminus u$ ;  $z = A * u.t$ ;  $\text{rhoo} = z' * z$ ;
46. if  $\text{rhoo} == 0$ ,  $\text{flag} = -1$ ; return, end
47.  $\text{rhoo} = -z' * u / \text{rhoo}$ ;
48.  $x = x - \text{rhoo} * u.t$ ;
49.  $r = \text{rhoo} * z + u$ ;
50.  $\text{err} = \text{norm}(r) / \text{bnrm2}$ ;
51.  $\text{iter} = \text{iter} + 1$ ;
52. if  $\text{err} < \text{tol}$ ,  $\text{flag} = 0$ ; return, end
53. if  $\text{iter} \geq \text{max\_it}$ , return, end
54.
55.  $\text{rc} = \text{rhoo} * c(n)$ ;
56. if  $\text{rc} == 0$ ,  $\text{flag} = -1$ ; return, end
57. for  $i = 1 : n - 1$ 
58.    $f = Q(:, i + 1)' * u$ ;
59.   if  $j \geq 1$ 
60.      $\text{betaa} = -f / c(i)$ ;
61.     if  $i \leq n - 2$ 
62.        $D(:, i) = u + \text{betaa} * D(:, i)$ ;
63.        $G(:, i) = \text{betaa} * G(:, i)$ ;
64.        $W(:, i) = \text{betaa} * W(:, i)$ ;
65.        $\text{betaa} = -Q(:, i + 2)' * D(:, i) / c(i + 1)$ ;
66.       for  $s = i + 1 : n - 2$ 
67.          $D(:, i) = D(:, i) + \text{betaa} * D(:, s)$ ;
68.          $G(:, i) = G(:, i) + \text{betaa} * G(:, s)$ ;
69.          $W(:, i) = W(:, i) + \text{betaa} * W(:, s)$ ;
70.          $\text{betaa} = -Q(:, s + 2)' * D(:, i) / c(s + 1)$ ;
71.       end
72.        $G(:, i) = G(:, i) + \text{betaa} * G(:, n - 1)$ ;
73.        $W(:, i) = W(:, i) + \text{betaa} * W(:, n - 1)$ ;
74.        $W(:, i) = r + \text{rhoo} * W(:, i)$ ;
75.     else
76.        $G(:, i) = \text{betaa} * G(:, n - 1)$ ;
77.        $W(:, i) = r + (\text{rhoo} * \text{betaa}) * W(:, n - 1)$ ;
78.     end
79.      $\text{betaa} = -Q(:, 1)' * W(:, i) / \text{rc}$ ;
80.      $W(:, i) = W(:, i) + (\text{rhoo} * \text{betaa}) * W(:, n)$ ;
81.      $G(:, i) = G(:, i) + W(:, i) + \text{betaa} * G(:, n)$ ;
82.   else
83.      $\text{betaa} = -Q(:, 1)' * r / \text{rc}$ ;
84.      $W(:, i) = r + (\text{rhoo} * \text{betaa}) * W(:, n)$ ;
85.      $G(:, i) = W(:, i) + \text{betaa} * G(:, n)$ ;
86.   end
87.   for  $s = 1 : i - 1$ 
88.      $\text{betaa} = -Q(:, s + 1)' * W(:, i) / c(s)$ ;
89.      $G(:, i) = G(:, i) + \text{betaa} * G(:, s)$ ;

```

```

90.          $W(:, i) = W(:, i) + \text{betaa} * D(:, s);$ 
91.     end
92.     if  $i < n - 1$ 
93.          $D(:, i) = W(:, i) - u;$ 
94.          $c(i) = Q(:, i + 1)' * D(:, i);$ 
95.         if  $c(i) == 0$ ,  $\text{flag} = -1$ ; return, end
96.          $\text{alphaa} = f/c(i);$ 
97.          $u = u - \text{alphaa} * D(:, i);$ 
98.     else
99.          $c(i) = Q(:, i + 1)' * (W(:, i) - u);$ 
100.        if  $c(i) == 0$ ,  $\text{flag} = -1$ ; return, end
101.         $\text{alphaa} = f/c(i);$ 
102.    end
103.     $g\_t = M \setminus G(:, i); W(:, i) = A * g\_t;$ 
104.     $\text{alphaa} = \text{rhoo} * \text{alphaa};$ 
105.     $x = x + \text{alphaa} * g\_t;$ 
106.     $r = r - \text{alphaa} * W(:, i);$ 
107.     $\text{err} = \text{norm}(r)/\text{bnrm2};$ 
108.     $\text{iter} = \text{iter} + 1;$ 
109.    if  $\text{err} < \text{tol}$ ,  $\text{flag} = 0$ ; return, end
110.    if  $\text{iter} \geq \text{max\_it}$ , return, end
111. end
112.  $e = Q(:, 1)' * r;$   $\text{betaa} = -e/rc;$ 
113.  $W(:, n) = r + (\text{rhoo} * \text{betaa}) * W(:, n);$ 
114.  $G(:, n) = W(:, n) + \text{betaa} * G(:, n);$ 
115. if  $n \geq 2$ 
116.      $\text{betaa} = -Q(:, 2)' * W(:, n)/c(1);$ 
117.     for  $s = 1 : n - 2$ 
118.          $G(:, n) = G(:, n) + \text{betaa} * G(:, s);$ 
119.          $W(:, n) = W(:, n) + \text{betaa} * D(:, s);$ 
120.          $\text{betaa} = -Q(:, s + 2)' * W(:, n)/c(s + 1);$ 
121.     end
122.      $G(:, n) = G(:, n) + \text{betaa} * G(:, n - 1);$ 
123. end
124.  $g\_t = M \setminus G(:, n); W(:, n) = A * g\_t;$ 
125.  $c(n) = Q(:, 1)' * W(:, n);$ 
126. if  $c(n) == 0$ ,  $\text{flag} = -1$ ; return, end
127. end

```

Remark:  $u\_t$  can be stored in the location of  $g\_t$  without changing the performance of the code.

The following function *time\_per\_iter* outputs the operation time in a single iteration of Code #1.

#### Code #2: Time per iteration.

1. function  $t = \text{time\_per\_iter}(A, b, Q, M)$
2. % input:  $A$       N-by-N coefficient matrix
3. %         $b$         right hand side vector
4. %         $Q$         N-by-n auxiliary matrix used by ML( $n$ )BiCGStab
5. %         $M$         N-by-N preconditioner matrix

```

6. % output: t          estimate of the operation time per iteration of Algorithm 9.1
7.
8.   N = size(A, 2); n = size(Q, 2); c = randn(1, n); x = randn(N, 1);
9.   r = x; g_t = x; G = randn(N, n); W = G;
10.  if n > 2, D = G(:, 1 : n - 2); end
11.  e = randn(1); bnorm2 = rand(1); iter = 0;
12.  max_it = 10; tol = 0.0; t_2 = 0; t_n = 0;
13.
14.  ii = 1; j = 1;
15.  tic, Lines 39 - 56 of Code #1 go here; t_11 = toc;
16.  for i = 1 : max(n - 2, 1) : n - 1
17.      tic
18.      Lines 58 - 110 of Code #1 go here;
19.      if i == 1, t_2 = toc; else, t_n = toc; ii = ii + 1; end
20.  end
21.  tic, Lines 112 - 126 of Code #1 go here; t_12 = toc;
22.  t = (t_11 + t_12 + ((t_2 + t_n)/ii) * (n - 1))/n;

```

**9.2. ML( $n$ )BiCGStab Associated with (5.2) and Related Codes.** The following algorithm is a preconditioned version of Algorithm 5.1.

**ALGORITHM 9.2. ML( $n$ )BiCGStab with preconditioning associated with definition (5.2).**

1. Choose an initial guess  $\mathbf{x}_0$  and  $n$  vectors  $\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n$ .
2. Compute  $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$ ,  $\tilde{\mathbf{g}}_0 = \mathbf{M}^{-1}\mathbf{r}_0$ ,  $\mathbf{w}_0 = \mathbf{A}\tilde{\mathbf{g}}_0$ ,  $c_0 = \mathbf{q}_1^H \mathbf{w}_0$  and  $e_0 = \mathbf{q}_1^H \mathbf{r}_0$ .
3. For  $j = 0, 1, 2, \dots$
4. For  $i = 1, 2, \dots, n - 1$
5.  $\alpha_{jn+i} = e_{jn+i-1}/c_{jn+i-1}$ ;
6.  $\mathbf{x}_{jn+i} = \mathbf{x}_{jn+i-1} + \alpha_{jn+i}\tilde{\mathbf{g}}_{jn+i-1}$ ;    %  $\tilde{\mathbf{g}} = \mathbf{M}^{-1}\mathbf{g}$
7.  $\mathbf{r}_{jn+i} = \mathbf{r}_{jn+i-1} - \alpha_{jn+i}\mathbf{W}_{jn+i-1}$ ;
8.  $e_{jn+i} = \mathbf{q}_{i+1}^H \mathbf{r}_{jn+i}$ ;
9. If  $j \geq 1$
10.  $\beta_{(j-1)n+i}^{(jn+i)} = -e_{jn+i}/(\rho_j c_{(j-1)n+i})$ ;
11.  $\mathbf{z}_w = \mathbf{r}_{jn+i} + \rho_j \beta_{(j-1)n+i}^{(jn+i)} \mathbf{w}_{(j-1)n+i}$ ;
12.  $\tilde{\mathbf{g}}_{jn+i} = \beta_{(j-1)n+i}^{(jn+i)} \tilde{\mathbf{g}}_{(j-1)n+i}$ ;
13. For  $s = i + 1, \dots, n - 1$
14.  $\beta_{(j-1)n+s}^{(jn+i)} = -\mathbf{q}_{s+1}^H \mathbf{z}_w / (\rho_j c_{(j-1)n+s})$ ;
15.  $\mathbf{z}_w = \mathbf{z}_w + \rho_j \beta_{(j-1)n+s}^{(jn+i)} \mathbf{w}_{(j-1)n+s}$ ;
16.  $\tilde{\mathbf{g}}_{jn+i} = \tilde{\mathbf{g}}_{jn+i} + \beta_{(j-1)n+s}^{(jn+i)} \tilde{\mathbf{g}}_{(j-1)n+s}$ ;
17. End
18.  $\tilde{\mathbf{g}}_{jn+i} = \tilde{\mathbf{g}}_{jn+i} + \mathbf{M}^{-1}\mathbf{z}_w$ ;
19. Else
20.  $\tilde{\mathbf{g}}_{jn+i} = \mathbf{M}^{-1}\mathbf{r}_{jn+i}$ ;
21. End
22.  $\mathbf{w}_{jn+i} = \mathbf{A}\tilde{\mathbf{g}}_{jn+i}$ ;
23. For  $s = 0, \dots, i - 1$
24.  $\beta_{jn+s}^{(jn+i)} = -\mathbf{q}_{s+1}^H \mathbf{w}_{jn+i} / c_{jn+s}$ ;

```

25.       $\mathbf{w}_{jn+i} = \mathbf{w}_{jn+i} + \beta_{jn+s}^{(jn+i)} \mathbf{w}_{jn+s};$ 
26.       $\tilde{\mathbf{g}}_{jn+i} = \tilde{\mathbf{g}}_{jn+i} + \beta_{jn+s}^{(jn+i)} \tilde{\mathbf{g}}_{jn+s};$ 
27.      End
28.       $c_{jn+i} = \mathbf{q}_{i+1}^H \mathbf{w}_{jn+i};$ 
29.      End
30.       $\alpha_{jn+n} = e_{jn+n-1}/c_{jn+n-1};$ 
31.       $\mathbf{x}_{jn+n} = \mathbf{x}_{jn+n-1} + \alpha_{jn+n} \tilde{\mathbf{g}}_{jn+n-1};$ 
32.       $\mathbf{u}_{jn+n} = \mathbf{r}_{jn+n-1} - \alpha_{jn+n} \mathbf{w}_{jn+n-1};$ 
33.       $\tilde{\mathbf{u}}_{jn+n} = \mathbf{M}^{-1} \mathbf{u}_{jn+n};$ 
34.       $\rho_{j+1} = -(\mathbf{A} \tilde{\mathbf{u}}_{jn+n})^H \mathbf{u}_{jn+n} / \|\mathbf{A} \tilde{\mathbf{u}}_{jn+n}\|_2^2;$ 
35.       $\mathbf{x}_{jn+n} = \mathbf{x}_{jn+n} - \rho_{j+1} \tilde{\mathbf{u}}_{jn+n};$ 
36.       $\mathbf{r}_{jn+n} = \rho_{j+1} \mathbf{A} \tilde{\mathbf{u}}_{jn+n} + \mathbf{u}_{jn+n};$ 
37.       $e_{jn+n} = \mathbf{q}_1^H \mathbf{r}_{jn+n};$ 
38.       $\beta_{(j-1)n+n}^{(jn+n)} = -e_{jn+n} / (\rho_{j+1} c_{(j-1)n+n});$ 
39.       $\mathbf{z}_w = \mathbf{r}_{jn+n} + \rho_{j+1} \beta_{(j-1)n+n}^{(jn+n)} \mathbf{w}_{(j-1)n+n};$ 
40.       $\tilde{\mathbf{g}}_{jn+n} = \beta_{(j-1)n+n}^{(jn+n)} \tilde{\mathbf{g}}_{(j-1)n+n};$ 
41.      For  $s = 1, \dots, n-1$ 
42.           $\beta_{jn+s}^{(jn+n)} = -\mathbf{q}_{s+1}^H \mathbf{z}_w / (\rho_{j+1} c_{jn+s});$ 
43.           $\mathbf{z}_w = \mathbf{z}_w + \rho_{j+1} \beta_{jn+s}^{(jn+n)} \mathbf{w}_{jn+s};$ 
44.           $\tilde{\mathbf{g}}_{jn+n} = \tilde{\mathbf{g}}_{jn+n} + \beta_{jn+s}^{(jn+n)} \tilde{\mathbf{g}}_{jn+s};$ 
45.      End
46.       $\tilde{\mathbf{g}}_{jn+n} = \mathbf{M}^{-1} \mathbf{z}_w + \tilde{\mathbf{g}}_{jn+n};$ 
47.       $\mathbf{w}_{jn+n} = \mathbf{A} \tilde{\mathbf{g}}_{jn+n};$ 
48.       $c_{jn+n} = \mathbf{q}_1^H \mathbf{w}_{jn+n};$ 
49. End

```

A sample Matlab code is as follows.

**Code #3: Matlab code of Algorithm 9.2**

```

1. function [x, err, iter, flag] = mlbicgstab(A, x, b, Q, M, max_it, tol)
2.
3. % input:  A      N-by-N matrix
4. %        M      N-by-N preconditioner matrix
5. %        Q      N-by-n auxiliary matrix with columns  $\mathbf{q}_1, \dots, \mathbf{q}_n$ 
6. %        x      initial guess
7. %        b      right hand side vector
8. %        max_it (integer) maximum number of iterations
9. %        tol    error tolerance
10. % output: x      solution computed
11. %         err   error norm
12. %         iter  (integer) number of iterations performed
13. %         flag  (integer): 0 = solution found to tolerance
14. %                    1 = no convergence given max_it
15. %                    -1 = breakdown
16. % storage: c, rc  1 x n matrices
17. %         x, r, b, u_t, z  N-by-1 matrices
18. %         A, M      N-by-N matrices
19. %         Q, G, W   N-by-n matrices
20.

```

```

21.  $N = \text{size}(A, 2)$ ;  $n = \text{size}(Q, 2)$ ;
22.  $G = \text{zeros}(N, n)$ ;  $W = \text{zeros}(N, n)$ ; % initialize workspace for  $\tilde{\mathbf{g}}$ ,  $\mathbf{w}$ ,  $c$  and  $\rho * c$ 
23.  $c = \text{zeros}(1, n)$ ;  $rc = \text{zeros}(1, n)$ ; % end initialization
24.
25.  $iter = 0$ ;  $flag = 1$ ;  $bnrm2 = \text{norm}(b)$ ;
26. if  $bnrm2 == 0.0$ ,  $bnrm2 = 1.0$ ; end
27.  $r = b - A * x$ ;  $err = \text{norm}(r)/bnrm2$ ;
28. if  $err < tol$ ,  $flag = 0$ ; return, end
29.
30.  $G(:, 1) = M \setminus r$ ;  $W(:, 1) = A * G(:, 1)$ ;  $c(1) = Q(:, 1)' * W(:, 1)$ ;
31. if  $c(1) == 0$ ,  $flag = -1$ ; return, end
32.  $e = Q(:, 1)' * r$ ;
33.
34. for  $j = 0 : max\_it$ 
35.     for  $i = 1 : n - 1$ 
36.          $alpha = e/c(i)$ ;
37.          $x = x + alpha * G(:, i)$ ;
38.          $r = r - alpha * W(:, i)$ ;
39.          $err = \text{norm}(r)/bnrm2$ ;
40.          $iter = iter + 1$ ;
41.         if  $err < tol$ ,  $flag = 0$ ; return, end
42.         if  $iter \geq max\_it$ , return, end
43.
44.          $e = Q(:, i + 1)' * r$ ;
45.         if  $j \geq 1$ 
46.              $beta = -e/rc(i + 1)$ ;
47.              $W(:, i + 1) = r + (rho * beta) * W(:, i + 1)$ ;
48.              $G(:, i + 1) = beta * G(:, i + 1)$ ;
49.             for  $s = i + 1 : n - 1$ 
50.                  $beta = -Q(:, s + 1)' * W(:, i + 1)/rc(s + 1)$ ;
51.                  $W(:, i + 1) = W(:, i + 1) + (rho * beta) * W(:, s + 1)$ ;
52.                  $G(:, i + 1) = G(:, i + 1) + beta * G(:, s + 1)$ ;
53.             end
54.              $G(:, i + 1) = G(:, i + 1) + (M \setminus W(:, i + 1))$ ;
55.         else
56.              $G(:, i + 1) = M \setminus r$ ;
57.         end
58.          $W(:, i + 1) = A * G(:, i + 1)$ ;
59.         for  $s = 0 : i - 1$ 
60.              $beta = -Q(:, s + 1)' * W(:, i + 1)/c(s + 1)$ ;
61.              $W(:, i + 1) = W(:, i + 1) + beta * W(:, s + 1)$ ;
62.              $G(:, i + 1) = G(:, i + 1) + beta * G(:, s + 1)$ ;
63.         end
64.          $c(i + 1) = Q(:, i + 1)' * W(:, i + 1)$ ;
65.         if  $c(i + 1) == 0$ ,  $flag = -1$ ; return, end
66.     end
67.      $alpha = e/c(n)$ ;
68.      $x = x + alpha * G(:, n)$ ;
69.      $r = r - alpha * W(:, n)$ ;

```

```

70.     err = norm(r)/bnrm2;
71.     if err < tol, flag = 0; iter = iter + 1; return, end
72.     u.t = M\r; z = A * u.t; rhoo = z' * z;
73.     if rhoo == 0, flag = -1; return, end
74.     rhoo = -z' * r/rhoo;
75.     x = x - rhoo * u.t;
76.     r = r + rhoo * z;
77.     err = norm(r)/bnrm2;
78.     iter = iter + 1;
79.     if err < tol, flag = 0; return, end
80.     if iter >= max_it, return, end
81.
82.     rc = rhoo * c;
83.     if rc(1) == 0, flag = -1; return, end
84.     e = Q(:,1)' * r; betaa = -e/rc(1);
85.     W(:,1) = r + (rhoo * betaa) * W(:,1);
86.     G(:,1) = betaa * G(:,1);
87.     for s = 1 : n - 1
88.         if rc(s + 1) == 0, flag = -1; return, end
89.         betaa = -Q(:,s+1)' * W(:,1)/rc(s + 1);
90.         W(:,1) = W(:,1) + (rhoo * betaa) * W(:,s+1);
91.         G(:,1) = G(:,1) + betaa * G(:,s+1);
92.     end
93.     G(:,1) = (M\W(:,1)) + G(:,1); W(:,1) = A * G(:,1);
94.     c(1) = Q(:,1)' * W(:,1);
95.     if c(1) == 0, flag = -1; return, end
96. end

```

The following function outputs the operation time per iteration of Code #3.

**Code #4: Time per iteration.**

```

1. function t = time_per_iter(A, b, Q, M)
2. % input:  A      N-by-N coefficient matrix
3. %        b      right hand side vector
4. %        Q      N-by-n auxiliary matrix used by ML(n)BiCGStab
5. %        M      N-by-N preconditioner matrix
6. % output: t      estimate of the operation time per iteration of Algorithm 9.2
7.
8.     N = size(A,2); n = size(Q,2); c = randn(1,n); rc = c;
9.     x = randn(N,1); r = x; G = randn(N,n); W = G;
10.    e = randn(1); bnrm2 = rand(1); rhoo = randn(1);
11.    iter = 0; max_it = 10; tol = 0.0; j = 1;
12.
13.    if n == 1
14.        tic, Lines 67 - 95 of Code #3 go here; t = toc;
15.    elseif n == 2
16.        tic, i = 1; Lines 36 - 42 of Code #3 go here; t_11 = toc;
17.        tic, Lines 44 - 65, 67-80 of Code #3 go here; t_2 = toc;
18.        tic, Lines 82 - 95 of Code #3 go here; t_12 = toc;
19.        t = (t_11 + t_12 + t_2)/2;
20.    else

```

```

21.     tic,  $i = 1$ ; Lines 36 - 42 of Code #3 go here;  $t_{11} = toc$ ;
22.     tic
23.      $i = n - 2$ ; Lines 44 - 65 of Code #3 go here;
24.      $i = n - 1$ ; Lines 36 - 42 of Code #3 go here;
25.      $t_{n-1} = toc$ ;
26.     tic, Lines 44 - 65, 67 - 80 of Code #3 go here;  $t_n = toc$ ;
27.     tic, Lines 82 - 95 of Code #3 go here;  $t_{12} = toc$ ;
28.      $t = ((t_{11} + t_{12}) + t_{n-1})/2 * (n - 1) + t_n/n$ ;
29.     end

```

**9.3. Codes for the Optimal  $n$ .** The function *find\_n* searches for the  $n$  at which ML( $n$ )BiCGStab takes the least operation time in a single iteration. *find\_n* needs to call the following function to implement.

**Code #5: fitting a quadratic polynomial.**

```

1. function  $z = least\_sq(x, y)$ 
2. % Input  $x, y$ , vectors of length  $k$ . Assume that  $x$  has increasing components.
3. % least_sq uses orthogonal polynomials to find a quadratic polynomial
4. %  $p(t) = e(1) + e(2)t + e(3)t^2$  that best fits the data  $\{(x(1), y(1)), \dots, (x(k), y(k))\}$ 
5. % in the least-squares sense. Then, it outputs the minimizer  $z \in [x(1), x(k)]$  of  $p(t)$ .
6.
7.      $k = length(x)$ ;
8.      $a = zeros(1, 3)$ ;  $b = zeros(1, 2)$ ;  $c2 = 0$ ;  $d = zeros(1, 3)$ ;  $d(1) = k$ ;
9.     for  $s = 1 : k$ ,  $b(1) = b(1) + x(s)$ ; end
10.     $b(1) = b(1)/d(1)$ ;
11.    for  $s = 1 : k$ 
12.         $d(2) = d(2) + (x(s) - b(1))^2$ ;  $b(2) = b(2) + x(s) * (x(s) - b(1))^2$ ;
13.         $c2 = c2 + x(s) * (x(s) - b(1))$ ;
14.    end
15.     $b(2) = b(2)/d(2)$ ;  $c2 = c2/d(1)$ ;
16.    for  $s = 1 : k$ 
17.         $a(1) = a(1) + y(s)$ ;  $a(2) = a(2) + y(s) * (x(s) - b(1))$ ;
18.         $a(3) = a(3) + y(s) * ((x(s) - b(2)) * (x(s) - b(1)) - c2)$ ;
19.         $d(3) = d(3) + ((x(s) - b(2)) * (x(s) - b(1)) - c2)^2$ ;
20.    end
21.     $a(1) = a(1)/d(1)$ ;  $a(2) = a(2)/d(2)$ ;  $a(3) = a(3)/d(3)$ ;
22.     $e = [a(1) + b(1) * (b(2) * a(3) - a(2)) - a(3) * c2, a(2) - a(3) * (b(1) + b(2)), a(3)]$ ;
23.    if  $e(3) > 0$ 
24.         $z = -e(2)/(2 * e(3))$ ; %  $p'(z) = 0$ 
25.    else
26.         $z = x(1) - 1$ ;
27.    end
28.    if  $(z < x(1)) | (z > x(k))$ 
29.         $p1 = e(1) + e(2) * x(1) + e(3) * x(1)^2$ ; %  $p1 = p(x(1))$ 
30.         $pk = e(1) + e(2) * x(k) + e(3) * x(k)^2$ ; %  $pk = p(x(k))$ 
31.        if  $p1 < pk$ ,  $z = x(1)$ ; else,  $z = x(k)$ ; end
32.    end

```

**Code #6: finding the optimal  $n$ .**

```

1. function  $n = find\_n(A, b, Q, M, n\_step, n\_max)$ 

```

```

2. % input:  n_step      positive integer  $\geq 4$ .
3. %        n_max      positive integer, a multiple of n_step
4. %        A, b, M    coefficient matrix, right hand side and preconditioner
5. %        Q         N-by-n_max auxiliary matrix
6. % output: n         $n \in [1, n\_max]$  at which ML(n)BiCGStab has about the
7. %                    smallest operation time in a single iteration
8.
9.   if (n_step <= 3) | (rem(n_max, n_step) ~ = 0)
10.      disp('n_step is assumed to be greater than 3 and n_max a multiple of n_step.');
```

```

11.      n = -1; return
12.   end
13.
14.   n1 = 1; t1 = time_per_iter(A, b, Q(:, 1 : n1), M);
15.   n4 = n_step; t4 = time_per_iter(A, b, Q(:, 1 : n4), M);
16.   if (n4 == n_max) | (t4 >= t1)           % optimal  $n \in [n1, n4]$ 
17.      n2 = -floor(-(n1 + n4)/3); t2 = time_per_iter(A, b, Q(:, 1 : n2), M);
18.      n3 = floor(2 * (n1 + n4)/3); t3 = time_per_iter(A, b, Q(:, 1 : n3), M);
19.      z = least_sq([n1, n2, n3, n4], [t1, t2, t3, t4]);
20.      n = floor(z); return
21.   end
22.
23.   n3 = n4; t3 = t4;
24.   n4 = n3 + n_step; t4 = time_per_iter(A, b, Q(:, 1 : n4), M);
25.   while (t4 < t3) & (n4 < n_max)
26.      n1 = n3; t1 = t3; n3 = n4; t3 = t4;
27.      n4 = n3 + n_step; t4 = time_per_iter(A, b, Q(:, 1 : n4), M);
28.   end           % optimal  $n \in [n1, n4]$ 
29.   n2 = floor((n1 + n3)/2); t2 = time_per_iter(A, b, Q(:, 1 : n2), M);
30.   z = least_sq([n1, n2, n3, n4], [t1, t2, t3, t4]);
31.   n = floor(z);
```

Remark: To obtain a better estimate of  $n_{opt}$ , one can add “ $t = \text{time\_per\_iter}(A, b, Q(:, 1 : n), M)$ ;  $[t, i] = \min([t, t1, t2, t3, t4])$ ;  $nn = [n, n1, n2, n3, n4]$ ;  $n = nn(i)$ ,” after Line 31.

**9.4. A Sample Run.** Here is a sample run of *find\_n.m* plug a ML(*n*)BiCGStab algorithm.

1.  $N = 100$ ;  $A = \text{randn}(N)$ ;  $M = \text{randn}(N)$ ;  $b = \text{randn}(N, 1)$ ;
2.  $\text{tol} = 1e-7$ ;  $\text{max\_it} = 3 * N$ ;  $n\_step = 4$ ;  $n\_max = 10 * n\_step$ ;
3.  $x = \text{zeros}(N, 1)$ ;  $r = b - A * x$ ;  $Q = [r, \text{sign}(\text{randn}(N, n\_max - 1))]$ ;
4.  $n = \text{find\_n}(A, b, Q, M, n\_step, n\_max)$ ;
5.  $[x, \text{err}, \text{iter}, \text{flag}] = \text{mlbicgstab}(A, x, b, Q(:, 1 : n), M, \text{max\_it}, \text{tol})$ ;

**Acknowledgements.** The index functions were introduced by Prof. Daniel Boley in [34]. Without the index functions, this research work would have become too complicated to be possible. The author wishes to thank Dr. Jens-Peter M. Zemke for the very helpful communications and the up-to-date information about Krylov subspace methods.

The author is grateful to his Ph.D. advisor Prof. Tony Chan for guiding him into the area of Krylov subspace methods. The method was originally named MLBiCGStab(*n*) by Tony. The current and more accurate name was from Prof. Henk A. van der Vorst.

## REFERENCES

- [1] J. ALIAGA, D. BOLEY, R. FREUND AND V. HERNÁNDEZ, *A Lanczos-type method for multiple starting vectors*, Math. Comp. 69 (2000), pp. 1577-1601.
- [2] R. E. BANK AND T. F. CHAN, *A composite step bi-conjugate gradient algorithm for nonsymmetric linear systems*, Numerical Algorithms 7(1994) 1-16.
- [3] C. BREZINSKI AND M. REDIVO-ZAGLIA, *Look-ahead in Bi-CGSTAB and other methods for linear systems*, BIT, 35(1995), pp. 169-201.
- [4] A. EDELMAN, *Eigenvalues and condition numbers of random matrices*, SIAM J. Matrix Anal. Appl., 9(1988), 543-560.
- [5] R. FLETCHER, *Conjugate gradient methods for indefinite systems*, volume 506 of Lecture Notes Math., pages 73-89. Springer-Verlag, Berlin-Heidelberg-New York, 1976.
- [6] R. FREUND AND M. MALHOTRA, *A Block-QMR algorithm for non-hermitian linear systems with multiple right-hand sides*, Linear Algebra Appl., 254(1997), pp. 119-157.
- [7] R. W. FREUND, M. GUTKNECHT AND N. M. NACHTIGAL, *An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices*, SIAM J. Sci. Statist. Comput., 14(1993), pp. 137-158.
- [8] A. EL GUENNOUNI, K. JBILOU AND H. SADOK, *A block version of BiCGSTAB for linear systems with multiple right-hand sides*, ETNA 16 (2003), 129-142.
- [9] M. H. GUTKNECHT, *Variants of BICGStab for matrices with complex spectrum*, SIAM J. Sci. Comput., 14 (1993), pp. 1020-1033.
- [10] —, *Lanczos-type solvers for nonsymmetric linear systems of equations*, Acta Numerica, 6 (1997), pp. 271-397.
- [11] —, *IDR Explained*, submitted to ETNA.
- [12] M. H. GUTKNECHT AND K. J. RESSEL, *Look-ahead procedures for Lanczos-type product methods based on three-term Lanczos recurrences*, SIAM J. Matrix Anal. Appl., 21(2000), pp. 1051-1078.
- [13] R. HORN AND C. JOHNSON, *Matrix analysis*, Cambridge University Press, 1985.
- [14] C. LANCZOS, *Solution of systems of linear equations by minimized iterations*, J. Research Nat. Bureau of Standards, 49 (1952), pp. 33-53.
- [15] K. MORIYA AND T. NODERA, *Breakdown-free ML( $k$ )BiCGStab algorithm for non-Hermitian linear systems*, O. Gervasi et al. (Eds.): ICCSA 2005, LNCS 3483, pp. 978-988, 2005.
- [16] D. O'LEARY, *The block conjugate gradient algorithm and related methods*, Linear Algebra Appl., 29(1980), pp. 293-322.
- [17] B. N. PARLETT, D. R. TAYLOR, AND Z. A. LIU, *A look-ahead Lanczos algorithm for unsymmetric linear systems*, Mathematics of Computation 1985; 44:105-124.
- [18] Y. SAAD, *Iterative methods for sparse linear systems*, 2nd edition, SIAM, Philadelphia, PA, 2003.
- [19] Y. SAAD AND M. H. SCHULTZ, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856-869.
- [20] Y. SAAD AND H. A. VAN DER VORST, *Iterative solution of linear systems in the 20-th century*, J. Comp. and Appl. Math., 123(1-2):1-33, 2000.
- [21] V. SIMONCINI AND D. B. SZYLD, *Recent computational developments in Krylov Subspace Methods for linear systems*, Numerical Linear Algebra with Applications, vol. 14(2007) pp. 1-59.
- [22] G. L. G. SLEIJPEN AND D. R. FOKKEMA, *BiCGSTAB( $l$ ) for linear equations involving unsymmetric matrices with complex spectrum*, ETNA, 1:11-32, 1993.
- [23] G. L. G. SLEIJPEN, P. SONNEVELD, AND M. B. VAN GIJZEN, *Bi-CGSTAB as an induced dimension reduction method*, Preprint 1369, Dep. Math., University Utrecht (April, 2008).
- [24] 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.
- [25] —, *An overview of approaches for the stable computation of hybrid BiCG methods*, Appl. Numer. Math., 19 (1995), pp. 235-254.
- [26] —, *Reliable updated residuals in hybrid Bi-CG methods*, Computing, 56 (1996), pp. 141-163.
- [27] G. L. G. SLEIJPEN, H. A. VAN DER VORST, AND D. R. FOKKEMA, *BiCGstab( $l$ ) and other hybrid Bi-CG methods*, Numerical Algorithms, 7 (1994), pp. 75-109. Received Oct. 29, 1993.
- [28] P. SONNEVELD, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 36-52.
- [29] P. SONNEVELD AND M. VAN GIJZEN, *IDR( $s$ ): a family of simple and fast algorithms for solving large nonsymmetric linear systems*, SIAM J. Sci. Comput. Vol. 31, No. 2, pp. 1035-1062.
- [30] H. A. VAN DER VORST, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the*

- solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 12 (1992), pp. 631–644.
- [31] —, *Iterative Krylov methods for large linear systems*, Cambridge University Press, Cambridge, April 2003.
- [32] H. VAN DER VORST AND Q. YE, *Residual Replacement Strategies for Krylov Subspace Iterative Methods for the Convergence of True Residuals*, SIAM J. Sci. Comput., 22 (2000):836–852.
- [33] P. WESSELING AND P. SONNEVELD, *Numerical experiments with a multiple grid and a preconditioned Lanczos type method*, in Approximation methods for Navier-Stokes problems (Proc. Sympos., Univ. Paderborn, Paderborn, 1979), vol. 771 of Lecture Notes in Math., Springer, Berlin, 1980, pp. 543–562.
- [34] M. YEUNG AND D. BOLEY, *Transpose-free multiple Lanczos and its application in Padé approximation*, Journal of Computational and Applied Mathematics, Vol 177/1 pp. 101–127, 2005.
- [35] M. YEUNG AND T. CHAN, *ML(k)BiCGSTAB: A BiCGSTAB variant based on multiple Lanczos starting vectors*, SIAM J. Sci. Comput., Vol. 21, No. 4, pp. 1263–1290, 1999.
- [36] SHAO-LIANG ZHANG, *GPBi-CG: Generalized product-type methods based on Bi-CG for solving nonsymmetric linear systems*, SIAM J. Sci. Comput., 18:537–551, 1997.