Electronic Transactions on Numerical Analysis.
Volume 46, pp. 460–473, 2017.
Copyright © 2017, Kent State University.
ISSN 1068–9613.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

THE BLOCK HESSENBERG PROCESS FOR MATRIX EQUATIONS[*]

M. ADDAM[†], M. HEYOUNI[†], AND H. SADOK[‡]

Abstract. In the present paper, we first introduce a block variant of the Hessenberg process and discuss its properties. Then, we show how to apply the block Hessenberg process in order to solve linear systems with multiple right-hand sides. More precisely, we define the block CMRH method for solving linear systems that share the same coefficient matrix. We also show how to apply this process for solving discrete Sylvester matrix equations. Finally, numerical comparisons are provided in order to compare the proposed new algorithms with other existing methods.

Key words. Block Krylov subspace methods, Hessenberg process, Arnoldi process, CMRH, GMRES, low-rank matrix equations.

AMS subject classifications. 65F10, 65F30

1. Introduction.

In this work, we are first interested in solving s systems of linear equations with the same coefficient matrix and different right-hand sides of the form

$$A\,x^{(i)} = y^{(i)}, \quad 1 \le i \le s, \tag{1.1}$$

where A is a large and sparse $n \times n$ real matrix, $y^{(i)}$ is a real column vector of length n, and $s \ll n$. Such linear systems arise in numerous applications in computational science and engineering such as wave propagation phenomena, quantum chromodynamics, and dynamics of structures [5, 9, 36, 39]. When n is small, it is well known that the solution of (1.1) can be computed by a direct method such as $L\,U$ or Cholesky factorization. Note that the factorization needs to be carried out only once and the resulting upper and lower triangular systems are solved at low cost.

Let $Y = [y^{(1)}, \dots, y^{(s)}] \in \mathbb{R}^{n\times s}$, $X = [x^{(1)}, \dots, x^{(s)}] \in \mathbb{R}^{n\times s}$, and assume that all s vectors $y^{(i)}$ are available simultaneously. Then the above systems can be written as

$$A\,X = Y. \tag{1.2}$$

In the last two decades, block Krylov subspace methods for block linear systems of the form (1.2) have been developed. These iterative methods are suitable when n is large and when the matrix A is not explicitly available. For symmetric and positive definite matrices A, O'Leary presented in [31] a block conjugate gradient (BCG) method. Other variants of the BCG algorithm and generalizations to nonsymmetric matrices were presented in [30, 31]. Generalizations of classical and robust Krylov methods for solving a linear system such as GMRES, QMR, and BiCG-Stab to the block case are respectively considered in [28, 38, 40, 16], [19], and [18]. Parallel implementations of block Krylov solvers are discussed in [3, 6, 7, 10] and in the references therein.

We also consider, in this paper, the solution of the low-rank Sylvester matrix equation

$$A\,X\,B - X = C\,F^T, \tag{1.3}$$

where $X \in \mathbb{R}^{n\times p}$, $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{p\times p}$, $C \in \mathbb{R}^{n\times r}$, and $F \in \mathbb{R}^{p\times r}$ with $r \ll \min\{n, p\}$.

[*]Received December 29, 2015. Accepted September 2, 2017. Published online on November 20, 2017. Recommended by L. Reichel.

[†]ENSA d'Al-Hoceima, Université Mohammed Premier, Oujda, Maroc ({addam.mohamed@gmail.com, mohammed.heyouni@gmail.com}).

[‡]L.M.P.A, Université du Littoral, 50 rue F. Buisson BP699, F-62228 Calais Cedex, France ({sadok@lmpa.univ-littoral.fr}).

Matrix Sylvester equations have numerous applications in filtering and image restoration [8]. They are also encountered in control and communication theory, model reduction problems, feedback stabilization, and pole-placement problems [4, 13, 12]. In order to ensure the existence of a unique solution, we assume that the matrices A and B of every Sylvester matrix equation satisfy $\mu_i(A)\mu_j(B) \neq 1$ for all $i = 1, \ldots, n,\ j = 1, \ldots, s$, where $\mu_k(Z)$ is the kth eigenvalue of the matrix Z.

2. The block Hessenberg process. If computations are carried out in exact arithmetic, then similar to the classical Hessenberg process with pivoting strategy [41], the block Hessenberg process generates a lower trapezoidal basis $\mathcal{V}_m = [V_1, \ldots, V_m] \in \mathbb{R}^{n\times ms}$ of the block Krylov subspace,

$$\mathcal{K}_m(A, R) = \{X \in \mathbb{C}^{n\times s}\,|X = \sum_{i=0}^{m-1} A^i R\,\Omega_i;\ \Omega_i \in \mathbb{R}^{s\times s} \text{ for } i = 0, \ldots, m-1\} \subset \mathbb{C}^{n\times s},$$

where R is a given $n \times s$ column block vector.

The first block vector V_1 is obtained by performing an LU decomposition with partial pivoting of the given block vector R. This means if $P_1\,R = L_1\,\Gamma$ is the PLU decomposition of R, where $P_1 \in \mathbb{R}^{n\times n}$ is a permutation matrix, $L_1 \in \mathbb{R}^{n\times s}$ is a unit lower trapezoidal matrix, and $\Gamma \in \mathbb{R}^{s\times s}$ is an upper triangular matrix, then

$$V_1 = P_1^T\,L_1 = R\,\Gamma^{-1}.$$

We note that Γ and V_1 can be computed using the lu Matlab function $[V_1, \Gamma] = \texttt{lu}(R)$.

Let i_j $(j = 1, \ldots, s)$ be the index of the row of V_1 corresponding to the j-th row of L_1, and let $e_i = [0, \ldots, 0, 1, 0 \ldots, 0]^T$ be the i-th vector of the canonical basis of $\mathbb{R}^n$. Then we define $p_1 = (i_1, \ldots, i_s)$ and the $n \times s$ matrix $\widetilde{E}_1 = [e_{i_1}, \ldots, e_{i_s}]^T$, which correspond to the s first columns of P_1. The vector p_1 can be obtained using the max Matlab function $[\sim, p_1] = \texttt{max}(V_1)$.

Now, suppose that block vectors $V_1, \ldots, V_k$ have been computed and the permutation vectors $p_2, \ldots, p_k$ updated. Then we can generate the block vectors $U_{k+1}^{(k)}$ via

$$U_{k+1}^{(0)} = A\,V_k, \quad \text{and} \quad U_{k+1}^{(i)} = A\,V_k - \sum_{j=1}^{i} V_j\,H_{j,k}, \quad \text{for } i = 1, \ldots, k,$$

where the square matrices $H_{j,k} \in \mathbb{R}^{s\times s}$, $j = 1, \ldots, k$, are such that

$$U_{k+1}^{(i)} \perp \widetilde{E}_1, \ldots,\ \widetilde{E}_i, \quad \text{for } i = 1, \ldots, k. \tag{2.1}$$

Thanks to the previous orthogonality condition, we have

$$H_{j,k} = (V_j(p_j, :))^{-1}\,U_{k+1}^{(j)}(p_j, :), \quad \text{for } j = 1, \ldots, k.$$

Again, letting $P_{k+1}\,U_{k+1}^{(k)} = L_{k+1}\,H_{k+1,k}$ be the PLU decomposition of $U_{k+1}^{(k)}$, we obtain

$$V_{k+1} = P_{k+1}^T\,L_{k+1} = U_{k+1}^{(k)}\,H_{k+1,k}^{-1},$$

and using the lu Matlab function, the $(k + 1)$-st block vector V_{k+1} and the upper square triangular matrix $H_{k+1,k}$ are given by

$$[V_{k+1}, H_{k+1,k}] = \texttt{lu}(U_{k+1}^{(k)}).$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

We end the derivation of the block Hessenberg process by letting i_j be the index row of V_{k+1} which corresponds to the j-th row of L_{k+1} ($j = ks+1, \dots, (k+1)s$), $p_{k+1} = (i_{ks+1} \, \dots, i_{(k+1)s})$, and define $\widetilde{E}_{k+1} = [e_{i_{ks+1}}, \dots, e_{i_{(k+1)s}}]$.

We also observe that the `max` Matlab function allows us to update p_{k+1} by

$$[\sim, p_{k+1}] = \mathtt{max}(V_{k+1}).$$

Finally, a complete statement of the resulting block Hessenberg algorithm reads as follows.

ALGORITHM 1: The block Hessenberg algorithm (with partial pivoting)

```
• Inputs: $A$ an $n \times n$ matrix, $R$ an $n \times s$ matrix and $m$ an integer.
• Step 0. $[V_1,\ \Gamma] = \mathtt{lu}(R)$; $[\sim,\ p_1] = \mathtt{max}(V_1)$;
• Step 1. For $k = 1, \dots, m$
              $U^{(0)}_{k+1} = A\, V_k$;
              for $j = 1, \dots, k$
                  $H_{j,k} = (V_{p_j}(p_j, :))^{-1}\, U^{(j-1)}_{k+1}(p_j, :)$;
                  $U^{(j)}_{k+1} = U^{(j-1)}_{k+1} - V_j\, H_{j,k}$;
              end(for)
              $[V_{k+1},\ H_{k+1,k}] = \mathtt{lu}(U^{(k)}_{k+1})$; $[\sim,\ p_{k+1}] = \mathtt{max}(V_{k+1})$;
          end(For).
```

In exact arithmetic and after m steps, the above block Hessenberg procedure leads to the following relation, for $k = 1, \dots, m$,

$$A\, \underbrace{[V_1, \dots, V_k]}_{=\mathbb{V}_k} = \underbrace{[V_1, \dots, V_k, V_{k+1}]}_{=\mathbb{V}_{k+1} = [\mathbb{V}_k,\ V_{k+1}]} \begin{bmatrix} H_{1,1} & H_{1,2} & \dots & H_{1,k} \\ H_{2,1} & H_{2,2} & \dots & H_{2,k} \\ 0_s & H_{3,2} & \dots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0_s & \dots & 0_s & H_{k+1,k} \end{bmatrix}.$$

For $k = m$, the above relation can be rewritten as

$$A\, \mathbb{V}_m = \mathbb{V}_{m+1}\, \widetilde{\mathbb{H}}_m \tag{2.2}$$
$$= \mathbb{V}_m\, \mathbb{H}_m + V_{m+1}\, H_{m+1,m}\, E_m^T, \tag{2.3}$$

where $\widetilde{\mathbb{H}}_m$, $\mathbb{H}_m$ are respectively the $(m+1)s \times ms$ and $ms \times ms$ block upper Hessenberg matrices whose non-zero block entries are the $H_{j,k}$ generated by Algorithm 1 and $E_m = [0_s, \dots, 0_s, I_s]^T$ is the $ms \times s$ rectangular matrix whose m-th block element is I_s, the identity matrix of size s.

Letting $\mathbb{P}_m = (\widetilde{E}_1, \dots, \widetilde{E}_m) \in \mathbb{R}^{n \times ms}$, which is a permutation matrix, and using (2.1) we also have

$$\mathbb{P}_m^T\, \mathbb{V}_m = \mathbb{L}_m, \qquad (\text{ with } \mathbb{L}_m \in \mathbb{R}^{ms \times ms}), \tag{2.4}$$

where $\mathbb{L}_m$ is a unit lower triangular matrix. Now introducing $\mathbb{V}_m^L := \mathbb{L}_m^{-1}\, \mathbb{P}_m^T \in \mathbb{R}^{ms \times n}$, we see that $\mathbb{V}_m^L$ is a left inverse of $\mathbb{V}_m$ since, according to (2.4), we have $\mathbb{V}_m^L \mathbb{V}_m = I_{ms}$. Pre-multiplying (2.2) and (2.3), respectively, by $\mathbb{V}_{m+1}^L$ and $\mathbb{V}_m^L$, we get

$$\mathbb{V}_{m+1}^L\, A\, \mathbb{V}_m = \widetilde{\mathbb{H}}_m \text{ and } \mathbb{V}_m^L\, A\, \mathbb{V}_m = \mathbb{H}_m. \tag{2.5}$$

3. The block Hessenberg and CMRH methods. To solve a single linear system, the author in [34] proposed the CMRH method. The CMRH method can be interpreted as a GMRES-like method but based on the Hessenberg reduction process with pivoting strategy instead of the Arnoldi process [34, 35, 21, 33, 32]. A variant for dense linear systems and a parallel implementation of the CMRH method are described in [14, 15, 24]. The analysis of the performance of the CMRH algorithm combined with the boundary element method when applied to acoustic problems is considered in [1]. In this section, we investigate the block CMRH method for solving multiple linear systems. This new block method uses the block Hessenberg process described in the last section.

Block Krylov subspace methods for solving (1.2) are iterative methods that generate approximate solutions $X_k \in \mathbb{R}^{n\times s}$ such that

$$X_k - X_0 \in \mathcal{K}_k(A, R_0),$$

where $R_0 := Y - A\,X_0$ is the residual bock vector associated to an initial guess X_0.

Let $\mathbb{V}_k = [V_1, \ldots, V_k]$ be the permuted trapezoidal matrix and $\widetilde{\mathbb{H}}_k$ be the upper block Hessenberg matrix produced after k iterations of the block Hessenberg process applied to the pair (A, R_0). Then using (3), we can write

$$X_k = X_0 + \mathbb{V}_k\, D_k, \text{ where } D_k \in \mathbb{R}^{ks\times s}.$$

Since $R_0 = V_1\,\Gamma$, we can use (2.3) and the first relation in (2.5) to get an expression for the residual R_k associated to X_k,

$$R_k = R_0 - A\,\mathbb{V}_k\, D_k = V_1\,\Gamma - \mathbb{V}_{k+1}\,\widetilde{\mathbb{H}}_k\, D_k = \mathbb{V}_{k+1}\left(E_1\,\Gamma - \widetilde{\mathbb{H}}_k\, D_k\right),$$

where $E_1 \in \mathbb{R}^{(k+1)s\times s}$ corresponds to the first s columns of the identity matrix $I_{(k+1)\,s}$. Now, using the last equality and requiring the minimal norm residual condition

$$X_k = \underset{X\in X_0+\mathcal{K}_k(A,R_0)}{\arg\min} \|Y - A\,X\|_F, \tag{3.1}$$

we see that D_m is the solution of the full $n \times (k+1)s$ least-squares problem

$$\min_{D\in\mathbb{R}^{ks\times s}} \|\mathbb{V}_{k+1}(E_1\,\Gamma - \widetilde{\mathbb{H}}_k\, D)\|. \tag{3.2}$$

Solving this problem requires computing the QR decomposition of $\mathbb{V}_{k+1}\,\mathbb{H}_k$, which would require $\mathcal{O}(n(ks)^2)$ work and $\mathcal{O}(nks)$ storage. We would then obtain a method that is mathematically equivalent to block GMRES but useless in practice. So instead of solving (3.2), we solve a smaller problem, namely minimizing just the norm of the coefficient block vector in (3.2). Hence, we obtain D_k from the minimization problem

$$\min_{D\in\mathbb{R}^{ks\times s}} \|E_1\,\Gamma - \widetilde{\mathbb{H}}_k\, D\|.$$

If the matrix $\widetilde{\mathbb{H}}_k$ is of full rank, then $\widetilde{\mathbb{H}}_k^+ = \left(\widetilde{\mathbb{H}}_k^T\widetilde{\mathbb{H}}_k\right)^{-1}\widetilde{\mathbb{H}}_k^T$, the pseudo-inverse of $\widetilde{\mathbb{H}}_k$, is well defined, and so $D_k = \widetilde{\mathbb{H}}_k^+\, E_1\,\Gamma$. Finally the k-th iterate of the block CMRH method is given by

$$X_k = X_0 + \mathbb{V}_k\,\widetilde{\mathbb{H}}_k^+\, E_1\,\Gamma.$$

It is clear that when k increases, the number of block vectors that must be stored increases with k, so the computational and storage requirements grow with each iteration. To remedy this difficulty, we can use the algorithm iteratively, i.e., we can restart the algorithm every m steps, where m is some fixed integer parameter. This restarted version of block CMRH is denoted by BCMRH(m) and is described below.

```
ALGORITHM 2: BCMRH($m$), the restarted block CMRH method
• Inputs: $A$ an $n \times n$ matrix, $Y$ an $n \times s$ block vector, $m$ an integer.
          $X_0$ an initial guess, $\varepsilon$ a desired tolerance.
• Step 0. Compute $R_0 = Y - AX_0$;
• Step 1. Apply Algorithm 1 to the pair $(A, R_0)$ to get $\Gamma$, $\mathbb{V}_m$ and $\widetilde{\mathbb{H}}_m$;
• Step 2. Determine $D_m$ as the solution of $\min_{D \in \mathbb{R}^{ms \times s}} \|E_1 \Gamma - \widetilde{\mathbb{H}}_m D\|$;
• Step 3. Compute the approximate solution $X_m = X_0 + \mathbb{V}_m D_m$;
          Compute $R_m = Y - A X_m$;
• Step 4. If $\|R_m\| \leq \varepsilon$ Stop; else $X_0 = X_m$, $R_0 = R_m$; goto Step 1.
```

Before ending this section, we observe that instead of imposing the minimal norm condition (3.1), we can use the following orthogonality condition

$$\widetilde{R}_k \perp \widetilde{E}_1, \ldots, \widetilde{E}_k,$$

which defines the block Hessenberg method. In this case (just as in the case of the block FOM method [32, 40]) the block vector D_k is given as the solution of the $ks \times ks$ block linear system

$$\mathbb{H}_k D_k = E_1 \Gamma,$$

where $E_1 \in \mathbb{R}^{ks \times s}$ corresponds to the first s columns of the identity matrix I_{ks}, meaning that the k-th approximate solution of the block Hessenberg method is

$$X_k = X_0 + \mathbb{V}_k \mathbb{H}_k^{-1} E_1 \Gamma.$$

Finally, we observe that we get the algorithm of the block Hessenberg method by replacing the instruction given in step 2 of Algorithm 2 by the following one:

- *Step 2.* Determine D_m as the solution of $\mathbb{H}_m D = E_1 \Gamma$;

4. The low-rank Sylvester block Hessenberg method.

In this section, we are concerned with the solution of the low-rank Sylvester matrix equation (1.3). We show how to apply the block Hessenberg process in order to obtain low-rank approximate solutions to (1.3).

Let $\mathbb{V}_m = (V_1, \ldots, V_m)$, $\mathbb{W}_m = (W_1, \ldots, W_m)$ be the permuted trapezoidal matrices generated by applying simultaneously m steps of Algorithm 1 applied to the pairs (A, C) and (B^T, F), respectively. We recall that these matrices satisfy the following relations

$$A \mathbb{V}_m = \mathbb{V}_{m+1} \widetilde{\mathbb{H}}^{A}_m = \mathbb{V}_m \mathbb{H}^{A}_m + V_{m+1} H^{A}_{m+1,m} E_m^T \tag{4.1}$$

and

$$B^T \mathbb{W}_m = \mathbb{W}_{m+1} \widetilde{\mathbb{H}}^{B}_m = \mathbb{W}_m \mathbb{H}^{B}_m + W_{m+1} H^{B}_{m+1,m} E_m^T, \tag{4.2}$$

where $\mathbb{H}^{A}_m = \mathbb{V}_m^L A \mathbb{V}_m$, $\mathbb{H}^{B}_m = \mathbb{W}_m^L B^T \mathbb{W}_m$ and $E_m \in \mathbb{R}^{mr \times r}$ is the m-th block of the identity matrix I_{mr}. We observe furthermore that there exists Γ^{A}, $\Gamma^{B} \in \mathbb{R}^{r \times r}$ such that V_1 and W_1, the first blocks of $\mathbb{V}_m$ and $\mathbb{W}_m$, are given by the LU decomposition of C and F, i.e.,

$$C = V_1 \Gamma^{A}, \quad \text{and} \quad F = W_1 \Gamma^{B}.$$

We also have

$$\mathbb{V}_m^L\, C = E_1\, \Gamma^{A} \quad \text{and} \quad \mathbb{W}_m^L\, F = E_1\, \Gamma^{B},$$

where E_1 is the $mr \times r$ matrix corresponding to the first r columns of the identity matrix I_{mr}.

Following the ideas developed in [25, 26], we seek approximate solutions to (1.3) that have the form

$$\mathcal{X}_m = \mathbb{V}_m\, \mathcal{Y}_m\, \mathbb{W}_m^T \tag{4.3}$$

and satisfy the Galerkin-type condition

$$\mathbb{V}_m^L\, \mathcal{R}_m\, (\mathbb{W}_m^L)^T = 0, \tag{4.4}$$

where $\mathcal{R}_m = A\, \mathcal{X}_m\, B - \mathcal{X}_m - C\, F^T$ is the residual associated with $\mathcal{X}_m$.

By multiplying $\mathcal{R}_m$ on the left by $\mathbb{V}_m^L$, on the right by $(\mathbb{W}_m^L)^T$, and using $\mathcal{X}_m$ given by (4.3), the Galerkin condition (4.4) can be rewritten as

$$\begin{aligned} 0 &= \mathbb{V}_m^L\, \left[A\, \mathbb{V}_m\, \mathcal{Y}_m\, \mathbb{W}_m^T\, B - \mathbb{V}_m\, \mathcal{Y}_m\, \mathbb{W}_m^T - C\, F^T\right]\, (\mathbb{W}_m^L)^T \\ &= (\mathbb{V}_m^L\, A\, \mathbb{V}_m)\, \mathcal{Y}_m\, (\mathbb{W}_m^T\, B\, \mathbb{W}_m^L) - \mathcal{Y}_m - (\mathbb{V}_m^L\, C)\, (\mathbb{W}_m^L\, F)^T \\ &= \mathbb{H}_m^{A}\, \mathcal{Y}_m\, (\mathbb{H}_m^{B})^T - \mathcal{Y}_m - (E_1\, \Gamma^{A})\, (E_1\, \Gamma^{B})^T. \end{aligned} \tag{4.5}$$

If $\mu_i(\mathbb{H}_m^{A}) \cdot \mu_j(\mathbb{H}_m^{B}) \neq 1$, for all $i, j = 1, \ldots, mr$, then the unique solution of the reduced discrete Sylvester equation (4.5) can be obtained using direct methods [2, 20]. We observe that the computation of the approximate solution $\mathcal{X}_m$ requires matrix products of the three matrices $\mathbb{V}_m$, $\mathcal{Y}_m$, $\mathbb{W}_m^T$, and this becomes very expensive when m increases. Moreover, in order to monitor convergence of $\mathcal{X}_m$, we need to compute the residual $\mathcal{R}_m$, which requires matrix products with the large matrices A and B. Hence, to avoid the computations of expensive matrix products, we derive an upper bound for the residual norm $\|\mathcal{R}_m\|_F$, which allows us to stop the iterations in the low-rank Sylvester extended block Hessenberg algorithm.

PROPOSITION 4.1. *Let $\mathcal{Y}_m$ be the exact solution of (4.5) and $\mathcal{X}_m = \mathbb{V}_m\, \mathcal{Y}_m\, \mathbb{W}_m^T$ the approximate solution to the discrete Sylvester equation (1.3) obtained after m iterations of the block Hessenberg process applied to the pairs (A, C) and (B^T, F). Then, the residual $\mathcal{R}_m$ associated with $\mathcal{X}_m$ satisfies*

$$\|\mathcal{R}_m\|_F \leq \sqrt{n\, p}\, (m+1) r\, \sqrt{(\alpha^2 + \beta^2 + \gamma^2)}, \tag{4.6}$$

$$\begin{aligned} &\alpha = \|\mathbb{H}_m^{A}\, \mathcal{Y}_m\, E_m\, (H_{m+1,m}^{B})^T\|_F, \qquad \beta = \|T_{m+1,m}^{A}\, E_m^T\, \mathcal{Y}_m\, \mathbb{H}_m^{B}\|_F, \qquad \textit{and} \\ &\gamma = \|H_{m+1,m}^{A}\, E_m^T\, \mathcal{Y}_m\, E_m\, (H_{m+1,m}^{B})^T\|_F. \end{aligned}$$

Proof. Since $\mathbb{V}_{m+1}$, $\mathbb{W}_{m+1}^T$ can be respectively partitioned as $\mathbb{V}_{m+1} = [\mathbb{V}_m, V_{m+1}]$ and $\mathbb{W}_{m+1}^T = [\mathbb{W}_m^T, W_{m+1}^T]$, using (4.1) and (4.2), we can show that $\mathcal{R}_m = \mathbb{V}_{m+1}\, \Omega\, \mathbb{W}_{m+1}^T$, where

$$\Omega = \begin{bmatrix} O_{mr} & \mathbb{H}_m^{A}\, \mathcal{Y}_m\, E_m\, (H_{m+1,m}^{B})^T \\ H_{m+1,m}^{A}\, E_m^T\, \mathcal{Y}_m\, \mathbb{H}_m^{B} & H_{m+1,m}^{A}\, E_m^T\, \mathcal{Y}_m\, E_m\, (H_{m+1,m}^{B})^T \end{bmatrix}.$$

Therefore, $\|\mathcal{R}_m\|_F \leq \|\mathbb{V}_{m+1}\|_F\, \|\Omega\|_F\, \|\mathbb{W}_{m+1}\|_F$. Observe that

$$\begin{aligned} \|\Omega\|_F^2 = \|\mathbb{H}_m^{A}\, \mathcal{Y}_m\, E_m\, (H_{m+1,m}^{B})^T\|_F^2 + \|H_{m+1,m}^{A}\, E_m^T\, \mathcal{Y}_m\, \mathbb{H}_m^{B}\|_F^2 \\ + \|H_{m+1,m}^{A}\, E_m^T\, \mathcal{Y}_m\, E_m\, (H_{m+1,m}^{B})^T\|_F^2. \end{aligned}$$

Recall that $\mathbb{V}_{m+1} \in \mathbb{R}^{n\times(m+1)r}$, $\mathbb{W}_{m+1} \in \mathbb{R}^{p\times(m+1)r}$ are obtained by applying permutation matrices and that these matrices are lower trapezoidal with entries $\mathbb{V}_{i,j}$, $\mathbb{W}_{i,j}$, respectively, such that $\mathbb{V}_{i,j},\ \mathbb{W}_{i,j} \le 1$. From this, we deduce that

$$\|\mathbb{V}_{m+1}\|_F \le \sqrt{n}\,\|\mathbb{V}_{m+1}\|_1 \le \sqrt{n}\,\sqrt{(m+1)r},$$
$$\|\mathbb{W}_{m+1}\|_F \le \sqrt{p}\,\|\mathbb{W}_{m+1}\|_1 \le \sqrt{p}\,\sqrt{(m+1)r},$$

and this yields (4.6). ☐

To reduce the computational cost, we observe that as $E_m^T = (O_{r\times(m-1)r}, I_r)$, the quantities α, β, and γ introduced in the previous theorem are be given by

$$\left.\begin{aligned}\alpha &= \|\mathbb{H}_m^{A}\,(\mathcal{Y}_m)_{:,m_r}\,(H_{m+1,m}^{B})^T\|_F,\\ \beta &= \|H_{m+1,m}^{A}\,(\mathcal{Y}_m)_{m_r,:}\,\mathbb{H}_m^{B}\|_F,\\ \gamma &= \|H_{m+1,m}^{A}\,E_m^T\,(\mathcal{Y}_m)_{m_r,m_r}\,(H_{m+1,m}^{B})^T\|_F,\end{aligned}\right\}\ \text{where } m_r = (m-1)r+1 : mr.$$

Our experiments suggest that the upper bound (4.6) is a pessimistic one and that is more appropriate to use the following estimate which we derived heuristically,

$$\|\mathcal{R}_m\|_F \le \sqrt{\max(n,p)}\; m\,\sqrt{(\alpha^2+\beta^2+\gamma^2)} =: r_m^h. \tag{4.7}$$

Finally, it is possible to get $\mathcal{X}_m$ as a product of two low-rank matrices. We proceed as suggested in [37, 23] by computing the singular value decomposition of $\mathcal{Y}_m$, i.e., $\mathcal{Y}_m = \widetilde{V}\,\Sigma\,\widetilde{W}^T$, where $\Sigma = \text{diag}[\sigma_1, \sigma_2, \dots, \sigma_{\text{mr}}]$ is the diagonal matrix of the singular values of $\mathcal{Y}_m$ sorted in decreasing order.

Now, let $\tilde{V}_l$ and $\tilde{W}_l$ be the $mr \times l$ matrices of the first l columns of $\widetilde{V}$ and $\widetilde{W}$ corresponding respectively to the l singular values of magnitude greater than some tolerance τ. More precisely, let l be the such that $\sigma_{l+1} \le \tau < \sigma_l$, then $\mathcal{Y}_m \approx \widetilde{V}_l\,\Sigma_l\,\widetilde{W}_l^T$, where $\Sigma_l = \text{diag}[\sigma_1, \dots, \sigma_l]$. Finally, setting $\mathcal{Z}_m^{A} = \mathbb{V}_m\,\tilde{V}_l\,{\Sigma_l}^{1/2}$ and $\mathcal{Z}_m^{B} = \mathbb{W}_m\,\tilde{W}_l\,{\Sigma_l}^{1/2}$, it follows that

$$\mathcal{X}_m \approx \mathcal{Z}_m^{A}\,{\mathcal{Z}_m^{B}}^T.$$

The algorithm of the Low-Rank Sylvester block Hessenberg method is summarized below.

```
ALGORITHM 4: The Low-Rank Sylvester Block Hessenberg method" (LRS-BH)
• Inputs:   $A$ an $n \times n$ matrix, $B$ a $p \times p$ matrix, $C$ an $n \times r$ matrix,
            and $F$ a $p \times r$ matrix.
• Step 0.   Choose a tolerance $\epsilon > 0$, $m_{max}$ a maximum number of iterations,
            $k$ a step-size, and $\tau$ the tolerance for the truncated SVD.
• Step 1.   For $m = 1,\ 2,\ \dots,\ m_{max}$
• Step 1.1          Update $\mathbb{V}_m$ and $\mathbb{W}_m$ and $\widetilde{\mathbb{H}}_m^{A}$ and $\widetilde{\mathbb{H}}_m^{B}$ by applying the $m$th step
                    of Algorithm 2 to the pairs $(A, C)$ and $(B^T, F)$ respectively.
• Step 1.2          If $m$ is a multiple of $k$
                            Solve (4.5) and compute $r_m^h$ given by (4.7);
• Step 1.3                  If $r_m^h < \epsilon$,
                                    go to Step 2,
                            end(If).
                    end(If).
            end(For).
• Step 2.  Compute the SVD of $\mathcal{Y}_m$, i.e., $\mathcal{Y}_m = \widetilde{V}\,\Sigma\,\widetilde{W}^T$,
           where $\Sigma = \text{diag}[\sigma_1, \dots, \sigma_{mr}]$ and $\sigma_1 \ge \dots \ge \sigma_{mr}$;
```

Find l such that $\sigma_{l+1} \leq \tau < \sigma_l$ and let $\Sigma_l = \mathrm{diag}[\sigma_1, \ldots, \sigma_l]$;
Form $\mathcal{Z}_m^{A} = \mathbb{V}_m\, \tilde{V}_l\, \Sigma_l^{1/2}$ and $\mathcal{Z}_m^{B} = \mathbb{W}_m\, \tilde{W}_l\, \Sigma_l^{1/2}$;

- *Step 3.* The approximate solution $\mathcal{X}_m$ is given by $\mathcal{X}_m \approx \mathcal{Z}_m^{A}\, {\mathcal{Z}_m^{B}}^T$.

5. Numerical experiments. We present the following numerical experiments to illustrate the behavior and performance of the proposed methods. The different algorithms have been implemented in Matlab 7.9 and have been executed on a computer with an Intel Pentium-4 3.4GHz processor and 2028MBytes of RAM. The machine precision was $2.22\,10^{-16}$.

Example 1. In this example, we provide some experimental results of using the restarted block CMRH(m) method applied to (1.2) and compare its performance to that of restarted block GMRES(m). In all examples, the starting guess was taken to be $X_0 = 0$, and a maximum number of 301 restarts was allowed for each algorithm. In all experiments, the right-hand sides of the different systems are such that $Y = A\,X^*$, where X^* is generated randomly with coefficients uniformly distributed in $[0,\ 1]$. This choice enables us to compare the error norm given by $e = \|X^* - X_m\|_F$. In all examples in this set of experiments, we took $\epsilon = 10^{-10}$, and the tests were stopped as soon as the residual norm R_m satisfies $\|R_m\| = \|Y - A\,X_m\|_2 \leq \epsilon \|Y\|_2$.

Before describing the examples used to to show the efficiency of the proposed methods, we list below some properties of the matrices that are used in Examples 1.1, 1.3, and 2.2. These matrices come from the Matrix Market web collection which is a visual repository of test data for use in comparative studies of algorithms for numerical linear algebra [29]. All these matrices are real and nonsymmetric.

- `rdb 32001`. This matrix comes from a computational fluid dynamics problem. It is of size $n = 3200$ with a symmetric non-zero pattern and $nnz = 18880$ nonzero entries. The average nonzeros per row and column is 5.9. The estimation of the condition number is $2.71\,10^3$.
- `add32`. This matrix comes from a circuit simulation problem. It is real of size $n = 4960$ with a symmetric non-zero pattern and $nnz = 19848$ nonzero entries. The average nonzeros per row and column is 4. The estimation of the condition number is $2.14\,10^2$.
- `appu`. This matrix is a random sparse matrix used in a set of benchmark examples from the NASA AMES research center. It comes from a directed weighted random graph and is of size $n = 14000$ with a symmetric non-zero pattern and $nnz = 1853104$ nonzero entries. The estimation of the condition number is $1.71\,10^2$.
- `memplus`. This matrix comes from a memory circuit problem. It is of size $n = 17758$ with a symmetric non-zero pattern and $nnz = 126150$ nonzero entries. The average nonzeros per row and column is 5.6. The estimation of the condition number is $1.29\,10^5$.
- `psmigr_3`. This matrix comes from a memory circuit problem. It is of size $n = 3140$ and has $nnz = 543160$ nonzero entries. The average nonzeros per row and column is $1.7\,10^2$. The estimation of the condition number is $1.00\,10^2$.
- `pde2961`. This matrix comes from an economic problem. It is of size $n = 2961$ with a symmetric non-zero pattern and $nnz = 14585$ nonzero entries. The average nonzeros per row and column is 4.9. The estimation of the condition number is $9.49\,10^2$.

Example 1.1. In this first set of experiments, we report the results obtained with four matrices: `rdb32001`, `add32`, `appu`, and `memplus`. The results are summarized in Table 5.1. For the matrix `memplus`, we also compared the behavior of the preconditioned block CMRH

(BC) and block GMRES (BG) algorithms. The obtained results with an ILUO preconditioner [32] are listed in Table 5.2.

TABLE 5.1
Example 1.1. Results obtained with the Matrix-Market matrices `rdb32001`, `add32`, `appu`, *and* `memplus`.

A	$m,\ s$		# restarts	err. norm	res. norm	CPU time
A=`rdb32001`	$m = 10$	BC	301	$4.24\,10^{-6}$	$1.37, 10^{-8}$	29.93
$n = 3200$	$s = 5$	BG	137	$3.71\,10^{-7}$	$9.37, 10^{-8}$	**20.35**
$nnz = 18880$	$m = 10$	BC	301	$2.57\,10^{-5}$	$7.5, 10^{-6}$	**48.29**
	$s = 10$	BG	145	$5.06\,10^{-7}$	$1.15, 10^{-7}$	51.76
	$m = 20$	BC	59	$2.92\,10^{-7}$	$7.77\,10^{-8}$	**4.28**
	$s = 5$	BG	67	$3.96\,10^{-7}$	$8.72\,10^{-8}$	6.26
	$m = 20$	BC	50	$3.25\,10^{-7}$	$8.04\,10^{-8}$	**12.67**
	$s = 10$	BG	47	$6.32\,10^{-7}$	$1.05\,10^{-7}$	19.20
A=`add32`	$m = 10$	BC	15	$1.24\,10^{-8}$	$1.48\,10^{-11}$	**0.31**
$n = 4960$	$s = 2$	BG	16	$4.28\,10^{-8}$	$2.39\,10^{-11}$	0.34
$nnz = 23884$	$m = 10$	BC	18	$6.13\,10^{-8}$	$6.97\,10^{-11}$	**1.06**
	$s = 5$	BG	16	$5.70\,10^{-8}$	$3.14\,10^{-11}$	1.07
	$m = 10$	BC	23	$6.67\,10^{-8}$	$9.18\,10^{-11}$	**12.84**
	$s = 20$	BG	16	$8.88\,10^{-8}$	$4.91\,10^{-11}$	14.31
	$m = 10$	BC	25	$7.67\,10^{-8}$	$1.33\,10^{-10}$	**35.09**
	$s = 40$	BG	16	$1.22\,10^{-7}$	$6.89\,10^{-11}$	45.98
A=`appu`	$m = 5$	BC	38	$1.22\,10^{-6}$	$1.53\,10^{-7}$	**11.62**
$n = 14000$	$s = 5$	BG	45	$1.47\,10^{-6}$	$1.70\,10^{-7}$	15.31
$nnz = 1853104$	$m = 5$	BC	48	$1.50, 10^{-6}$	$1.76\,10^{-7}$	**26.46**
	$s = 10$	BG	44	$1.97\,10^{-6}$	$2.20\,10^{-7}$	29.64
	$m = 10$	BC	14	$8.26\,10^{-7}$	$9.76\,10^{-8}$	**10.39**
	$s = 5$	BG	14	$4.40\,10^{-7}$	$6.25\,10^{-8}$	12.17
	$m = 10$	BC	15	$6.90\,10^{-7}$	$1.07\,10^{-7}$	**26.87**
	$s = 10$	BG	13	$1.57\,10^{-6}$	$2.29\,10^{-7}$	32.53
A=`memplus`	$m = 20$	BC	290	$1.38\,10^{-2}$	$3.26\,10^{-7}$	736.48
$n = 17758$	$s = 5$	BG	145	$2.02\,10^{-2}$	$3.27\,10^{-7}$	**516.03**
$nnz = 126150$	$m = 20$	BC	301	$8.19\,10^{-2}$	$3.17\,10^{-6}$	**1680.91**
	$s = 10$	BG	301	$2.47\,10^{-4}$	$2.96\,10^{-9}$	2758.09
	$m = 40$	BC	118	$6.81\,10^{-6}$	$3.24\,10^{-10}$	**1076, 89**
	$s = 5$	BG	87	$2.32\,10^{-5}$	$2.97\,10^{-10}$	1123.91
	$m = 40$	BC	130	$5.20\,10^{-6}$	$3.22\,10^{-10}$	**2504.11**
	$s = 10$	BG	82	$3.31\,10^{-5}$	$4.11\,10^{-10}$	2515.09

As indicated in the tables of results, and except for a few cases, both methods are able to obtain approximate solutions that satisfy the stopping criterion. In some instances, we also note that even if the block CMRH method needs more restarts than the block GMRES method, the CPU time is in favor of the block Hessenberg based method. Indeed, the proposed block Hessenberg process is cheaper than the block Arnoldi process.

TABLE 5.2
Example 1.1. Results obtained for the matrix `memplus` *with ILU0 preconditioning.*

A	$m,\ s$		# restarts	err. norm	res. norm	CPU time
	$m=20$	BC	29	$4.44\,10^{-3}$	$2.45\,10^{-7}$	87.57
	$s=5$	BG	18	$1.88\,10^{-2}$	$3.20\,10^{-7}$	**71.37**
A=`memplus`	$m=20$	BC	51	$4.16\,10^{-6}$	$3.15\,10^{-10}$	326.22
$n=17758$	$s=10$	BG	35	$1.81\,10^{-5}$	$3.07\,10^{-10}$	**325.58**
$nnz=126150$	$m=40$	BC	14	$1.37\,10^{-6}$	$6.22\,10^{-11}$	**132.06**
	$s=5$	BG	11	$3.01\,10^{-6}$	$7.72\,10^{-11}$	147.02
	$m=40$	BC	14	$1.79\,10^{-6}$	$7.80\,10^{-11}$	**287.80**
	$s=10$	BG	10	$5.13\,10^{-6}$	$9.97\,10^{-11}$	316.16

Example 1.2. In this second set of experiments, the matrix A is obtained from the centered finite difference discretization of the operator

$$L_A(u) = \Delta u - (x+y^2)\,\frac{\partial u}{\partial x} - (y-x^2)\,\frac{\partial u}{\partial y} - \sqrt{x^2+y^2}\,u$$

on the unit square $[0,1]\times[0,1]$ with homogeneous Dirichlet boundary conditions. The number of inner grid points in each direction is n_0. Therefore, the dimension of the matrix A is $n=n_0^2$. The results reported in Table 5.3 are those obtained for $n_0=150$ (i.e., $n=22500$) and for different values of m and s.

TABLE 5.3
Example 1.2. Results obtained with matrix A generated by discretizing the operator L_A.

$m,\ s$		# restarts	err. norm	res. norm	CPU time
$s=5$	BC	65	$1.74\,10^{-5}$	$3.98\,10^{-4}$	**509.16**
$m=40$	BG	49	$1.51\,10^{-5}$	$3.32\,10^{-4}$	756.16
$s=5$	BC	19	$1.38\,10^{-5}$	$4.17\,10^{-4}$	**378.06**
$m=80$	BG	13	$1.65\,10^{-5}$	$3.64,10^{-4}$	380.72
$s=10$	BC	87	$1.77\,10^{-5}$	$4.29\,10^{-4}$	2539.91
$m=40$	BG	49	$1.63\,10^{-5}$	$3.51\,10^{-4}$	**2292.31**
$s=10$	BC	19	$7.89\,10^{-6}$	$2.48\,10^{-4}$	7353.39
$m=80$	BG	10	$1.61\,10^{-6}$	$9.94,10^{-5}$	**6277.01**

The results show that the block CMRH method gives better results than block GMRES when the number of right-hand sides s is small. However, for relatively large values of s, the block Arnoldi-based method is less time-consuming than the block Hessenberg based method.

Example 1.3. In Table 5.4 we compare the proposed method with matrix Krylov subspace methods [22, 27], and we report the results of applying the global CMRH(m) (GC) and global GMRES(m) (GG) to multiple linear systems with the matrices `appu` and `psmigr_3`.

TABLE 5.4
Example 1.3. Results obtained when comparing with matrix Krylov subspace methods.

A	$m,\ s$		# restarts	err. norm	res. norm	CPU time
		BC	41	$1.18\,10^{-6}$	$1.32\,10^{-7}$	**15.49**
A=`appu`	$m=5$	BG	45	$1.41\,10^{-6}$	$1.63\,10^{-7}$	21.84
$n=14000$	$s=5$	GC	40	$6.47\,10^{-7}$	$1.18\,10^{-7}$	21.54
$nnz=1853104$		GG	48	$9.53\,10^{-7}$	$1.09\,10^{-7}$	29.91
		BC	5	$1.02\,10^{-8}$	$2.41\,10^{-9}$	1.53
A=`psmigr_3`	$m=10$	BG	4	$4.84\,10^{-8}$	$3.68,10^{-9}$	**1.39**
$n=3140$	$s=10$	GC	14	$4.15\,10^{-8}$	$4.94\,10^{-9}$	3.51
$nnz=543162$		GG	15	$6.96\,10^{-8}$	$3.19,10^{-9}$	4.17

Example 2. We report experimental results obtained by comparing the performances of the Low-Rank Sylvester Block Hessenberg (LRSBH) given by Algorithm 4, the Low-Rank Sylvester Block Arnoldi (LRSBA), and the Low-Rank Sylvester Global Arnoldi (LRSGA) described in [17, 26]. In all the examples, the starting guess was taken to be $X_0 = 0$, the maximum size of the constructed Krylov subspaces was 150, and the matrices C and F of the matrix Sylvester equations were generated randomly with coefficients uniformly distributed in $[0,\ 1]$. All iterations were stopped as soon as the heuristic residual norm r_m^h, given by (4.7) for the LRSBH algorithm, or the residual norm r_m^{ba} (for the LRSBA algorithm) or the upper bound r_m^{ga} (for the LRSGA algorithm) were less than $\varepsilon\,\|C\,F^T\|_2$ with $\varepsilon = 10^{-9}$.

Example 2.1. We compare the sharpness of the upper bounds (4.6) and (4.7) by plotting the exact residual norm $\|\mathcal{R}_m\|_F = \|\,A\,\mathcal{X}_m\,B - \mathcal{X}_m - C\,F^T\|_F$.

The four plots correspond to the results obtained when solving the discrete Sylvester equation (1.3) with matrices A and B obtained from the centered finite difference discretization of the operators

$$L_A(u) = \Delta u - x\,\sqrt{1+y^2}\,\frac{\partial u}{\partial x} - (1+y^2)\,\frac{\partial u}{\partial y} - (1+x\,y)\,u,$$
$$L_B(u) = \Delta u - (y-x)\,\frac{\partial u}{\partial x} - \sqrt{(x^2+y^2)}\,\frac{\partial u}{\partial y} - (x^2-y^2)\,u,$$

on the unit square $[0,1]\times[0,1]$ with homogeneous Dirichlet boundary conditions. The number of inner grid points in each direction was n_0 for the operator L_A and p_0 for the operator L_B. Hence, the dimension of the matrices A and B are $n = n_0^2$ and $p = p_0^2$. The size of the matrices and the rank of the matrices C, F is indicated in the title of each plot.

As seen from Figure 5.1 and as indicated in the previous section, the theoretical upper bound (4.6) is pessimistic, and the heuristic upper bound (4.7) seems to be more accurate than the first one.

Example 2.2. In this experiment, we report in Table 5.5 the results obtained for different pairs of matrices $(\tilde{A},\ \tilde{B})$ coming from the Matrix Market repository [29], and we compare the performance of the LRSBH, LRSBA, and LRSGA methods. To ensure existence and uniqueness of the tested discrete Sylvester equations, we take

$$A = \frac{\tilde{A}}{\|\tilde{A}\|_1} \qquad \text{and} \qquad B = \frac{\tilde{B}}{\|\tilde{B}\|_1}.$$

We also note that for these numerical tests, the low-order Sylvester equations are solved every k iterations, and we took $\tau = 10^{-12}$ in Step 2 of Algorithm 4.

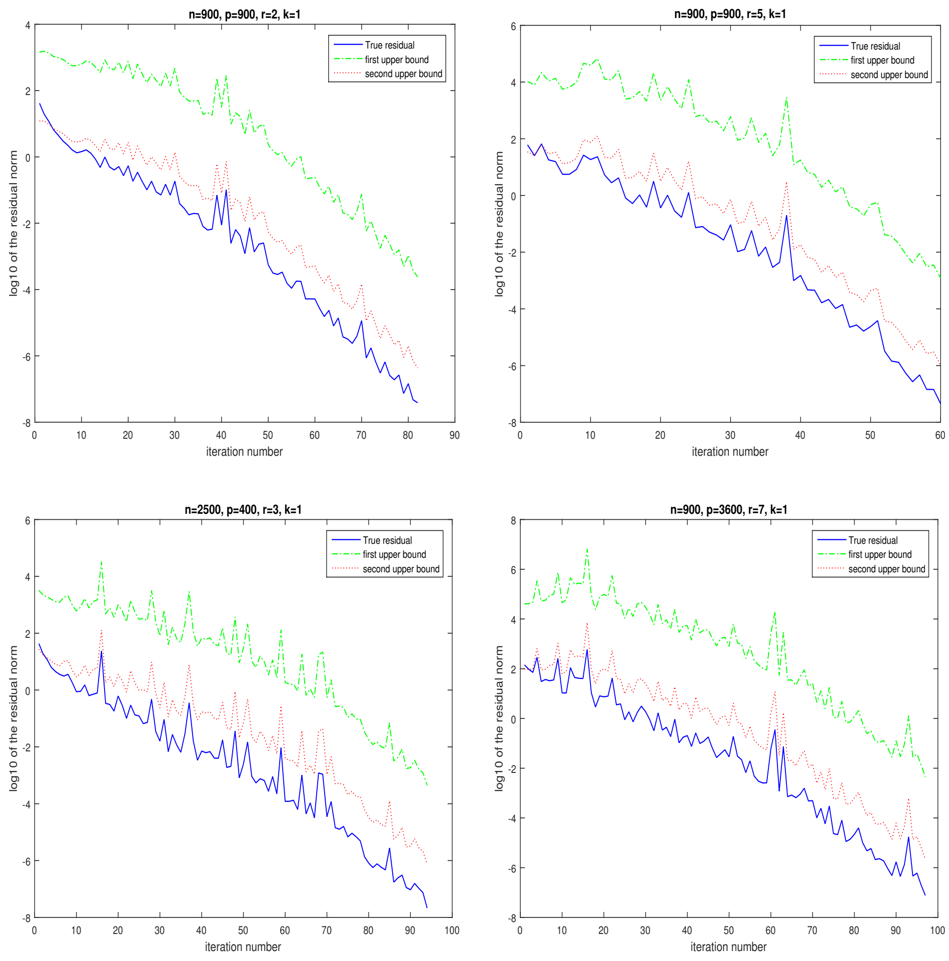

FIG. 5.1. *Comparison of the upper bounds (4.6) and (4.7) with the true residual norm.*

Acknowledgments. The authors thank the anonymous reviewers for their comments. The authors would like to express their gratitude to Laura Dykes for her careful reading of the paper and for her helpful comments and suggestions which helped improve the presentation of this paper.

REFERENCES

[1] A. ALIA, H. SADDOK, AND M. SOULI, *CMRH method as iterative solver for boundary element acoustic systems*, Eng. Anal. Bound. Elem., 36 (2012), pp. 346–350.

[2] R. H. BARTELS AND G. W. STEWART, *Algorithm 432: Solution of the matrix equation $AX + XB = C$*, Comm. ACM, 15 (1972), pp. 820–826.

[3] A. BASERMANN, B. REICHEL, AND C. SCHELTHOFF, *Preconditioned CG methods for sparse matrices on massively parallel machines*, Parallel Computing, 23 (1997), pp. 381–393.

[4] P. BENNER, *Factorized solution of Sylvester equations with applications in control*, in Proceedings of the 16th International Symposium on Mathematical Theory of Networks and Systems MTNS 2004, V. Blondel, P. Van Dooren, J. Willems, B. Motmans, and B. De Moor, eds., University of Leuven, Leuven, Belgium, 2004 (10 pages).

[5] J. C. R. BLOCH AND S. HEYBROCK, *A nested Krylov subspace method to compute the sign function of large complex matrices*, Comput. Phys. Commun., 182 (2011), pp. 878–889.

TABLE 5.5
Results for Example 2.2 with matrices from the Matrix-Market collection.

Test problem	Method	Iter.	$r_m/\|C\,F^T\|_2$	CPU-Time
$\tilde{A} = \texttt{appu}, n = 14000$	LRSBH	9	$1.46\,10^{-6}$	**1.01**
$\tilde{B} = -\texttt{pde2961}, p = 2961$	LRSBA	8	$3.55\,10^{-6}$	1.44
$r = 3, k = 1$	LRSGA	9	$5.54\,10^{-7}$	1.30
$\tilde{A} = \texttt{appu}, n = 14000$	LRSBH	10	$2.23\,10^{-7}$	**4.92**
$\tilde{B} = -\texttt{pde2961}, p = 2961$	LRSBA	10	$2.66\,10^{-8}$	6.52
$r = 10, k = 5$	LRSGA	10	$1.61\,10^{-7}$	5.70
$\tilde{A} = \texttt{psmigr\_3}, n = 3140$	LRSBH	12	$7.33\,10^{-8}$	0.96
$\tilde{B} = -\texttt{add32}, p = 4960$	LRSBA	10	$2.12\,10^{-6}$	1.11
$r = 5, k = 2$	LRSGA	10	$3.83\,10^{-6}$	**0.95**
$\tilde{A} = \texttt{psmigr\_3}, n = 3140$	LRSBH	10	$4.49\,10^{-6}$	**0.73**
$\tilde{B} = -\texttt{add32}, p = 4960$	LRSBA	10	$2.05\,10^{-6}$	0.98
$r = 5, k = 5$	LRSGA	10	$3.75\,10^{-6}$	0.79
$\tilde{A} = \texttt{psmigr\_3}, n = 3140$	LRSBH	10	$9.61\,10^{-8}$	**0.89**
$\tilde{B} = -\texttt{appu}, p = 14000$	LRSBA	8	$3.74\,10^{-6}$	0.92
$r = 5, k = 2$	LRSGA	12	$3.52\,10^{-7}$	1.13
$\tilde{A} = \texttt{psmigr\_3}, n = 3140$	LRSBH	8	$1.29\,10^{-5}$	**1.69**
$\tilde{B} = -\texttt{add32}, p = 14000$	LRSBA	8	$2.68\,10^{-6}$	2.45
$r = 10, k = 2$	LRSGA	10	$7.29\,10^{-7}$	2.66

[6] J. BOLZ, I. FARMER, E. GRINSPUN, AND P. SCHRÖDER, *Sparse matrix solvers on the GPU: conjugate gradients and multigrid*, ACM Trans. Graphics, 22 (2003), pp. 917–924.

[7] H. CALANDRA, S. GRATTON, J. LANGOU, X. PINEL, AND X. VASSEUR, *Flexible variants of block restarted GMRES methods with application to geophysics*, SIAM J. Sci. Comput., 34 (2012), pp. A714–A736.

[8] D. CALVETTI AND L. REICHEL, *Application of ADI iterative methods to the restoration of noisy images*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 165–186.

[9] R.W. CLOUGH AND J. PENZIEN, *Dynamics of Structures*, McGraw-Hill, 1975.

[10] R. D. DA CUNHA AND D. BECKER, *Dynamic block GMRES: an iterative method for block linear systems*, Adv. Comput. Math., 27 (2007), pp. 423–448.

[11] H. DAI, *Two algorithms for symmetric linear systems with multiple right-hand sides*, Numer. Math. J. Chin. Univ. (English Ser.), 9 (2000), pp. 91–110.

[12] B. N. DATTA, *Numerical Methods for Linear Control Systems. Design and Analysis*, Elsevier, San Diego, 2004.

[13] B. N. DATTA AND K. DATTA, *Theoretical and computational aspects of some linear algebra problems in control theory*, in Computational and Combinatorial Methods in Systems Theory (Stockholm, 1985), C. I. Byrnes and A. Lindquist, eds, North-Holland, Amsterdam, 1986, pp. 201–212.

[14] S. DUMINIL, *A parallel implementation of the CMRH method for dense linear systems*, Numer. Algorithms, 63 (2013), pp. 127–142.

[15] S. DUMINIL, M. HEYOUNI, P. MARION, AND H. SADOK, *Algorithms for the CMRH method for dense linear systems*, Numer. Algorithms, 71 (2016), pp. 383–394.

[16] L. ELBOUYAHYAOUI, A. MESSAOUDI, AND H. SADOK, *Algebraic properties of the block GMRES and block Arnoldi methods*, Electron. Trans. Numer. Anal., 33 (2008/09), pp. 207–220. http://etna.ricam.oeaw.ac.at/vol.33.2008-2009/pp207-220.dir/pp207-220.pdf

[17] A. EL GUENNOUNI, K. JBILOU, AND A. J. RIQUET, *Block Krylov subspace methods for solving large Sylvester equations*, Numer. Algorithms, 29 (2002), pp. 75–96.

[18] A. EL GUENNOUNI, K. JBILOU, AND H. SADOK, *A block version of BICGSTAB for linear systems with multiple right-hand sides*, Electron. Trans. Numer. Anal., 16 (2003), pp. 129–142. http://etna.ricam.oeaw.ac.at/vol.16.2003/pp129-142.dir/pp129-142.pdf

[19] 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.
[20] G. H. GOLUB, S. NASH, AND C. VAN LOAN, *A Hessenberg-Schur method for the problem* $AX + XB = C$, IEEE Trans. Automat. Control, 24 (1979), pp. 909–913.
[21] M. HEYOUNI. *Méthode de Hessenberg Généralisée et Applications*, PhD. Thesis, Université des Sciences et Technologies de Lille, Lille, 1996.
[22] ———, *The global Hessenberg and global CMRH methods for linear systems with multiple right-hand sides*, Numer. Algorithms, 26 (2001), pp. 317–332.
[23] ———, *Extended Arnoldi methods for large low-rank Sylvester matrix equations*, Appl. Numer. Math., 60 (2010), pp. 1171–1182.
[24] M. HEYOUNI AND H. SADOK, *A new implementation of the CMRH method for solving dense linear systems*, J. Comput. Appl. Math., 213 (2008), pp. 387–399.
[25] I. M. JAIMOUKHA AND E. M. KASENALLY, *Krylov subspace methods for solving large Lyapunov equations*, SIAM J. Numer. Anal., 31 (1994), pp. 227–251.
[26] K. JBILOU, *Low rank approximate solutions to large Sylvester matrix equations*, Appl. Math. Comput., 177 (2006), pp. 365–376.
[27] K. JBILOU, A. MESSAOUDI, AND H. SADOK, *Global FOM and GMRES algorithms for matrix equations*, Appl. Numer. Math., 31 (1999), pp. 49–63.
[28] H. L. LIU AND B. J. ZHONG, *Simpler block GMRES for nonsymmetric systems with multiple right-hand sides*, Electron. Trans. Numer. Anal., 30 (2008), pp. 1–9.
`http://etna.ricam.oeaw.ac.at/vol.30.2008/pp1-9.dir/pp1-9.pdf`
[29] MATRIX MARKET WEB COLLECTION, *`http://math.nist.gov/MatrixMarket/`*.
[30] A. NIKISHIN AND A. YEREMIN, *Variable block CG algorithms for solving large sparse symmetric positive definite linear systems on parallel computers. I: General Iterative Scheme*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 1135–1153.
[31] D. P. O'LEARY, *The block conjugate gradient algorithm and related methods*, Linear Algebra Appl., 29 (1980), pp. 293–322.
[32] Y. SAAD, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, New York, 1995.
[33] 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.
[34] H. SADOK, *CMRH: a new method for solving nonsymmetric linear systems based on the Hessenberg reduction algorithm*, Numer. Algorithms, 20 (1999), pp. 303–321.
[35] H. SADOK AND D. B. SZYLD, *A new look at CMRH and its relation to GMRES*, BIT, 52 (2012), pp. 485–501.
[36] T. SAKURAI, H. TADANO, AND Y. KURAMASHI, *Application of block Krylov subspace algorithms to the Wilson-Dirac equation with multiple right-hand sides in lattice QCD*, Comput. Phys. Comm., 181 (2010), pp. 113–117.
[37] V. SIMONCINI, *A new iterative method for solving large-scale Lyapunov matrix equations*, SIAM J. Sci. Comput., 29 (2007), pp. 1268–1288.
[38] V. SIMONCINI AND E. GALLOPOULOS, *Convergence properties of block GMRES and matrix polynomials*, Linear Algebra Appl., 247 (1996), pp. 97–119.
[39] J. VIRIEUX AND S. OPERTO, *An overview of full waveform inversion in exploration geophysics*, Geophysics, 74 (2009). pp. WCC127–WCC152.
[40] B. VITAL, *Etude de quelques méthodes de résolution de problèmes linéaires de grande taille sur multiprocesseur*, PhD. Thesis, Université de Rennes, Rennes, 1990.
[41] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1988.