Computers and Mathematics with Applications 76 (2018) 2011–2021

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

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

Weighted and flexible versions of block CMRH method for solving nonsymmetric linear systems with multiple right-hand sides

S. Amini, F. Toutounian *

Department of Applied Mathematics, School of Mathematical Sciences, Ferdowsi University of Mashhad, Iran

ARTICLE INFO

Article history:
Received 4 October 2017
Received in revised form 18 July 2018
Accepted 29 July 2018
Available online 18 August 2018

Keywords:
Block CMRH method
Block Krylov subspace
Weighting strategy
Flexible preconditioning
Multiple right-hand sides

ABSTRACT

Block Krylov subspace methods are the most popular algorithms for solving large nonsymmetric linear systems with multiple right-hand sides. One of them is the block CMRH method. This method generates a (non orthogonal) basis of the Krylov subspace through the block Hessenberg process. To accelerate the convergence of the block CMRH method, we will introduce two new methods. First, we present the block CMRH method with weighting strategy. In this method, the block CMRH method uses a different product at each restart. Second, we introduce a flexible version of the block CMRH algorithm that allows varying preconditioning at every step of the algorithm. Numerical experiments illustrate the benefits of the presented methods.

1. Introduction

Block iterative methods are used for large systems with multiple right-hand sides of the form

$$AX = B, \tag{1}$$

where $A \in \mathbb{R}^{n\times n}$ is a large nonsymmetric real matrix and X and B are rectangular matrices of dimension $n \times s$, and s is of moderate size (i.e., $s \ll n$). This problem arises in many areas of science and engineering, such as computational biology, electromagnetic structure computation, control theory, and so on [1–4].

When A is a large sparse matrix, block iterative methods, e.g., block CG method [5], block GMRES method [6], block Lanczos method [7], block QMR method [1], block BiCGSTAB method [8], block LSQR method [9], the block OSGCR(s)/OSOmin(s,k) methods [10,11], or block CMRH method [12,13] are natural candidates for solving (1). The purpose of these block methods is to provide the solutions of a multiple right-hand sides system faster than their single right-hand side counterparts. They are generally more efficient when the matrix of the linear system is relatively dense or when preconditioners are used.

The global methods form another family that can be applied to the solution of multiple linear systems. These methods are based on the use of a global projection process onto a matrix Krylov subspace and they are particularly suitable for sparse multiple linear systems. References on this class include global FOM and GMRES methods [14,15], global BCG and BiCGSTAB methods [16,17], global CGS algorithm [18,19], Gl-LSQR algorithm [20], Gl-BCR and Gl-CRS algorithms [21], global Hessenberg and CMRH methods [22], and global SCD algorithm [23]. In order to improve the convergence property of the Krylov subspace methods the weighted and flexible versions of these methods have been proposed. The weighting strategy has been successfully developed for solving linear systems [24] and matrix equations [24–28]. Several flexible versions of

* Corresponding author.
E-mail addresses: s.amini@mail.um.ac.ir (S. Amini), toutouni@math.um.ac.ir (F. Toutounian).

https://doi.org/10.1016/j.camwa.2018.07.045
0898-1221/

Krylov Subspace Methods have been implemented successfully. These include the flexible GMRES method [29], GMRESR [30], flexible CG [31–33], flexible QMR [34], flexible BiCG and flexible BiCGSTAB [35]. See also [36] for a general theory where the preconditioner itself is a Krylov subspace method.

For nonsymmetric problems, the block CMRH [12,13] is one such method, but it may need restarting. Here we give two new versions of the restarted block CMRH method to improve convergence. First to accelerate the convergence of the block CMRH method we apply a weighting technique. We introduce a weighted block Hessenberg process for constructing a basis of the block Krylov subspace by using the weighting matrix D. Second we propose a flexible version of the block CMRH.

Throughout the paper, all vectors and matrices are assumed to be real. For a matrix X, $\|X\|_F$ denotes the Frobenius norm $\|X\|_F = \sqrt{\mathrm{tr}(X^TX)}$. For a matrix $V \in \mathbb{R}^{n\times s}$, the block Krylov subspace $\mathcal{K}_k(A, V)$ is the subspace generated by the columns of the matrices $V, AV, A^2V, \ldots, A^{k-1}V$. Some MATLAB notation is used; for instance, $H_k(i+1 : m+1, 1 : m)$ denotes the portion of H_k with rows from $i+1$ to $m+1$ and columns from 1 to m. Finally, $0_{m\times n}$ and I_s will denote the zero and the identity matrices in $\mathbb{R}^{m\times n}$ and $\mathbb{R}^{s\times s}$, respectively.

As in [37], we need the definition of the left inverse of a rectangular matrix. Let Z_k be the $n \times k$ matrix. We partition this matrix as follows:

$$Z_k = \begin{bmatrix} Z1_k \\ Z2_k \end{bmatrix},$$

where $Z1_k$ is a $k \times k$ square matrix. If the matrix $Z1_k$ is nonsingular, we define Z_k^L a left inverse of Z_k by

$$Z_k^L = \left[Z1_k^{-1}, 0_{k\times(n-k)}\right].$$

The structure of the paper is as follows. In Section 2, we briefly describe the block CMRH method for solving nonsymmetric linear systems with multiple right-hand sides. A weighted version of the block CMRH algorithm is presented in Section 3. In Section 4, we propose the fixed and flexible preconditioned block CMRH algorithm. In Section 5, we demonstrate the effectiveness of the proposed methods. Finally, conclusions are summarized in Section 6.

2. Block CMRH method

The block CMRH method [12,13] is a generalization of the well-known CMRH method [37]. The essential component of the block CMRH method is the block Hessenberg process. Let $X_0 \in \mathbb{R}^{n\times s}$ be an initial matrix for the solution of system (1) and $R_0 = B - AX_0$ its residual. The block Hessenberg process computes a unit trapezoidal matrix $\mathcal{L}_m = [L_1, L_2, \ldots, L_m]$, whose matrices $L_i \in \mathbb{R}^{n\times s}$, for $i = 1, 2, \ldots, m$, form a basis of the Krylov subspace $\mathcal{K}_m(A, R_0) = \mathrm{span}\{R_0, AR_0, \ldots, A^{m-1}R_0\}$, by using the following formulas:

$$\begin{cases} R_0 = L_1U_1, \\ L_{k+1}H_{k+1,k} = AL_k - \sum_{j=1}^{k} L_jH_{j,k}, \qquad \text{for } k = 1, \ldots, m, \end{cases} \tag{2}$$

where the unit trapezoidal matrix L_{k+1} and the upper triangular matrix $H_{k+1,k} \in \mathbb{R}^{s\times s}$ are determined by the LU factorization of $W = AL_k - \sum_{j=1}^{k} L_jH_{j,k}$, and the matrices $H_{j,k} \in \mathbb{R}^{s\times s}$ are determined such that

$$L_{k+1} \perp E_1, E_2, \ldots, E_k, \tag{3}$$

where E_i, for $i = 1, 2, \ldots, k$, is the $n \times s$ matrix which is zero except for the ith s rows, which are the $s \times s$ identity matrix. Let $\overline{\mathcal{H}}_m \equiv (H_{i,j})_{1\le i\le m+1, 1\le j\le m}$ be an $(m+1)s \times ms$ block upper Hessenberg matrix. From the block Hessenberg process, we can deduce the relation

$$A\mathcal{L}_m = \mathcal{L}_{m+1}\overline{\mathcal{H}}_m = \mathcal{L}_m\mathcal{H}_m + L_{m+1}H_{m+1,m}E_m^T, \tag{4}$$

where $\mathcal{H}_m$ is the $ms\times ms$ matrix obtained from $\overline{\mathcal{H}}_m$ by deleting the last s rows and E_m is the $ms\times s$ matrix which is zero except for the mth s rows, which are the $s \times s$ identity matrix.

The block Hessenberg process can breakdown if the LU factorization of R_0 or W does not exist [13]. For avoiding such a breakdown, we use pivoting strategy.

The block CMRH method constructs an approximate solution of the form $X_m^{BC} = X_0 + \mathcal{L}_m Y_m^{BC}$, where Y_m^{BC} is the solution of the minimizing problem

$$\min_{Y\in\mathbb{R}^{ms\times s}} \|E_1U_1 - \overline{\mathcal{H}}_m Y\|_F, \tag{5}$$

where $E_1 \in \mathbb{R}^{(m+1)s\times s}$ is the first s columns of the identity matrix.

In Algorithm 1, we summarize the restarted block CMRH method with pivoting strategy (denoted by BCMRH(m)). More detail can be found in [13]. We mention that the block CMRH algorithm given in [12] is similar to the Algorithm 1. The main difference between these algorithms is the generation of the matrices H_m and W in the block Hessenberg processes.

We end this section by giving a relation between the residual norms of the block CMRH method and the block GMRES method denoted by $\|R_m^{BC}\|_F$ and $\|R_m^{BG}\|_F$, respectively, which is stated in the following lemma [13].

Algorithm 1 BCMRH(m)

1. Start: Choose X_0, Compute $R_0 = B - AX_0$.
2. Block Hessenberg process for constructing the basis $\mathcal{L}_m$ and the upper Hessenberg matrix $\overline{\mathcal{H}}_m$:
4. Compute the LU factorization of R_0 with pivoting strategy, i.e., $P_1R_0 = \bar{L}_1U_1$.
5. Set $L_1 = P_1^T\bar{L}_1$, $\mathcal{L}_1 = L_1$, and $\overline{\mathcal{H}}_0 = \phi$.
6. Set $q = 0_{1\times(m+1)s}$.
7. For $j = 1, \ldots, s$
8. Find the row index t such that $P_1^T(t, j) = 1$. Set $q(j) = t$.
9. end
10. For $k = 1, \ldots, m$
11. Compute $T = AL_k$.
12. Set $E = T(q(1 : ks), 1 : s)$, $F = \mathcal{L}_k(q(1 : ks), 1 : ks)$.
13. Compute $H_k = F^{-1}E$.
14. Compute $W = T - \mathcal{L}_kH_k$.
15. Compute the LU factorization of W with pivoting strategy, i.e., $P_{k+1}W = \bar{L}_{k+1}U_{k+1}$.
16. Set $L_{k+1} = P_{k+1}^T\bar{L}_{k+1}$ and $\mathcal{L}_{k+1} = [\mathcal{L}_k, L_{k+1}]$.
17. For $j = 1, \ldots, s$
18. Find the row index t such that $P_{k+1}^T(t, j) = 1$. Set $q(ks + j) = t$
19. end
20. $D_k = U_{k+1}$,
21. Set $\overline{\mathcal{H}}_k = \begin{bmatrix} \overline{\mathcal{H}}_{k-1} & H_k \\ O_{s\times(k-1)s} & D_k \end{bmatrix}$
22. end
23. Approximate solution:
24. Determine Y_m^{BC} as the solution of $\min_{Y\in\mathbb{R}^{ms\times s}} \| E_1U_1 - \overline{\mathcal{H}}_mY \|_F$.
25. Compute the approximation solution $X_m^{BC} = X_0 + \mathcal{L}_mY_m^{BC}$.
26. Compute $R_m^{BC} = B - AX_m^{BC}$.
27. Restart:
28. If $\| R_m^{BC} \|_F \leq \epsilon$ Stop,
29. else $X_0 = X_m^{BC}$, $R_0 = R_m^{BC}$ and go to 2.

Lemma 1. *If the initial guesses in the block CMRH and the block GMRES methods are equal, i.e., $X_0^{BC} = X_0^{BG} = X_0$, then*

$$\|R_m^{BC}\|_F \leq \kappa_F(\mathcal{L}_{m+1}^{BC})\|R_m^{BG}\|_F,$$

where $\kappa_F(\mathcal{L}_{m+1}^{BC}) = \|(\mathcal{L}_{m+1}^{BC})^\dagger\|_F\|\mathcal{L}_{m+1}^{BC}\|_F$.

3. The weighted block CMRH method

Before giving a complete description of the weighted block CMRH method, we introduce, as in [26], the inner product $\langle\cdot,\cdot\rangle_D$. Let $D \in \mathbb{R}^{n\times n}$ be a positive definite diagonal matrix. For two matrices X and Y in $\mathbb{R}^{n\times s}$, the inner product $\langle\cdot,\cdot\rangle_D$ is defined as follows:

$$\langle X, Y\rangle_D = \mathrm{tr}(X^TDY). \tag{6}$$

Then the corresponding matrix norm $\|\cdot\|_D$ is defined as

$$\|X\|_D = \sqrt{\mathrm{tr}(X^TDX)}.$$

In this case, we define the weighted LU decomposition of matrix $V \in \mathbb{R}^{n\times s}$ by $D^{\frac{1}{2}}V = LU$. By using this decomposition and the inner product $\langle\cdot,\cdot\rangle_D$, the weighted block Hessenberg process constructs a basis $\tilde{\mathcal{L}}_m = [\tilde{L}_1, \ldots, \tilde{L}_m]$ of the subspace $\mathcal{K}_m(A, V)$, whose weighted matrix $D^{\frac{1}{2}}\tilde{\mathcal{L}}_m$ is a unit trapezoidal matrix. This is done by computing the weighted LU decomposition of $R_0 = B - AX_0$ with pivoting strategy, (i.e., $\hat{P}_1D^{\frac{1}{2}}R_0 = \hat{L}_1\tilde{U}_1$, where $\hat{P}_1$ is a permutation matrix), setting $\tilde{L}_1 = D^{-\frac{1}{2}}\hat{P}_1^T\hat{L}_1$, and imposing the following orthogonality condition:

$$\tilde{L}_{k+1} \perp_D \tilde{E}_{p_1}, \ldots, \tilde{E}_{p_k},$$

where $\tilde{E}_{p_i} = \hat{P}_i^TE_i$, for $i = 1, 2, \ldots, k$, $\hat{P}_i$ is the permutation matrix obtained in the iteration i, and $X \perp_D Y$ means that $\langle X, Y\rangle_D = 0$.

The block Hessenberg matrix $\overline{\tilde{\mathcal{H}}}_m = (\tilde{H}_{i,j}) \in \mathbb{R}^{(m+1)s\times ms}$ constructed by the weighted block Hessenberg process can be partitioned as

$$\overline{\tilde{\mathcal{H}}}_m = \begin{pmatrix} \tilde{\mathcal{H}}_m \\ \tilde{H}_{m+1,m}E_m^T \end{pmatrix}. \tag{7}$$

As in the block Hessenberg process, the matrices constructed by the weighted block Hessenberg process satisfy the relationship

$$A\tilde{\mathcal{L}}_m = \tilde{\mathcal{L}}_{m+1}\overline{\tilde{\mathcal{H}}}_m. \tag{8}$$

Using the initial guess X_0 and the corresponding initial residual $R_0 = B - AX_0$, the weighted block CMRH method constructs the approximate solution

$$X_m = X_0 + \tilde{\mathcal{L}}_m Y_m^{WBC}, \tag{9}$$

where the matrix $Y_m^{WBC} \in \mathbb{R}^{ms\times s}$ is defined by

$$Y_m^{WBC} = \underset{Y\in\mathbb{R}^{ms\times s}}{\operatorname{argmin}} \|E_1\tilde{U}_1 - \overline{\tilde{\mathcal{H}}}_m Y\|_F,$$

As was pointed out in [24,25], the optimal choice of D in the weighted approaches is still an open problem and needs further investigation. Some choices for the weighting matrix have been considered in [24,27,28,38]. In order to test the influence of weighting matrix D on convergence, we consider two choices of matrix D based on the residual R_m, which could be updated during the iterations. As in [27], for numerical examples (Section 5) we set $D = D_1$, where

$$D_1 = diag(d), \qquad d_i = \frac{\sqrt{n}}{\|R_0\|_F}\sqrt{\sum_{j=1}^{n} |(R_0)_{i,j}|^2}, \tag{10}$$

with $\|D_1\|_F = \sqrt{n}$ and, as in [38], we use the mean of the block residual R_m, i.e., $D = D_2$, where

$$D_2 = diag(d), \qquad d = \left|\frac{\sum_{i=1}^{s} R_m(:,i)}{s}\right|. \tag{11}$$

Note that the restarted weighted block CMRH method can also be regarded as the BCMRH(m) for solving system $(D^{\frac{1}{2}}AD^{-\frac{1}{2}})(D^{\frac{1}{2}}X) = D^{\frac{1}{2}}B$, where the matrix D is dynamically set in each restart cycle. We can summarize the restarted weighted block CMRH (denoted with WBCMRH(m)) as follows.

3.1. Relations between BCMRH(m) and WBCMRH(m)

First, we can give relations between the bases and block Hessenberg matrices generated by Algorithms 1 and 2.

Proposition 1. *Assume that Algorithms 1 and 2 do not break down before the mth step. Then there exists a block upper triangular matrix*

$$\mathcal{F}_m = \begin{pmatrix} F_{11} & F_{12} & \cdots & F_{1m} \\ & F_{22} & \cdots & F_{2m} \\ & & \ddots & \vdots \\ & & & F_{mm} \end{pmatrix},$$

such that

$$\tilde{\mathcal{L}}_m = \mathcal{L}_m\mathcal{F}_m, \tag{12}$$

$$\mathcal{F}_m = \mathcal{L}_m^L\tilde{\mathcal{L}}_m, \tag{13}$$

and we can express $\overline{\tilde{\mathcal{H}}}_m$ *in terms of* $\overline{\mathcal{H}}_m$ *as*

$$\overline{\tilde{\mathcal{H}}}_m = \mathcal{F}_{m+1}^{-1}\overline{\mathcal{H}}_m\mathcal{F}_m. \tag{14}$$

Proof. For proving the relation (12), we use induction on j. For $j = 1$ by using the first step of the block Hessenberg process and its weighted algorithm we have

$$R_0 = L_1U_1 \quad \text{and} \quad R_0 = \tilde{L}_1\tilde{U}_1,$$

then we have $\tilde{L}_1 = L_1F_{11}$, where $F_{11} = U_1\tilde{U}_1^{-1} \in \mathbb{R}^{s\times s}$.

We suppose that for $j = 1, \ldots, m$, the relation (12) is true, so we have

$$\tilde{L}_j = \sum_{i=1}^{j} L_iF_{ij},$$

where $F_{ij} \in \mathbb{R}^{s\times s}$. Using (4) and (8), we obtain

$$AL_m = \sum_{k=1}^{m+1} L_kH_{km} \quad \text{and} \quad A\tilde{L}_m = \sum_{k=1}^{m+1} \tilde{L}_k\tilde{H}_{km}.$$

Algorithm 2 WBCMRH(m)

1. Start: Choose X_0, Compute $R_0 = B - AX_0$.
2. Compute D, e.g., by using (10) or (11).
3. Weighted Block Hessenberg process for constructing the basis $\tilde{\mathcal{L}}_m$ and the upper Hessenberg matrix $\overline{\tilde{\mathcal{H}}}_m$:
4. Compute the weighted LU factorization of R_0 with pivoting strategy, i.e., $\hat{P}_1 D^{\frac{1}{2}} R_0 = \hat{L}_1 \tilde{U}_1$.
5. Set $\tilde{L}_1 = D^{-\frac{1}{2}} \hat{P}_1^T \hat{L}_1$, $\tilde{\mathcal{L}}_1 = \tilde{L}_1$, and $\overline{\tilde{\mathcal{H}}}_0 = \phi$.
6. Set $q = 0_{1\times(m+1)s}$.
7. For $j = 1, \ldots, s$
8. Find the row index t such that $\hat{P}_1^T(t, j) = 1$. Set $q(j) = t$.
9. end
10. For $k = 1, \ldots, m$
11. Compute $\tilde{T} = A\tilde{L}_k$.
12. Set $\tilde{E} = \tilde{T}(q(1 : ks), 1 : s)$, $\tilde{F} = \tilde{\mathcal{L}}_k(q(1 : ks), 1 : ks)$.
13. Compute $\tilde{H}_k = \tilde{F}^{-1}\tilde{E}$.
14. Compute $\tilde{W} = \tilde{T} - \tilde{\mathcal{L}}_k \tilde{H}_k$.
15. Compute the weighted LU factorization of $\tilde{W}$ with pivoting strategy, i.e., $\hat{P}_{k+1} D^{\frac{1}{2}} \tilde{W} = \hat{L}_{k+1} \tilde{U}_{k+1}$.
16. Set $\tilde{L}_{k+1} = D^{-\frac{1}{2}} \hat{P}_{k+1}^T \hat{L}_{k+1}$ and $\tilde{\mathcal{L}}_{k+1} = [\tilde{\mathcal{L}}_k, \tilde{L}_{k+1}]$.
17. For $j = 1, \ldots, s$
18. Find the row index t such that $\hat{P}_{k+1}^T(t, j) = 1$. Set $q(ks + j) = t$
19. end
20. $D_k = \tilde{U}_{k+1}$,
21. Set $\overline{\tilde{\mathcal{H}}}_k = \begin{bmatrix} \overline{\tilde{\mathcal{H}}}_{k-1} & \tilde{H}_k \\ O_{s\times(k-1)s} & D_k \end{bmatrix}$
22. end
23. Approximate solution:
24. Determine Y_m^{WBC} as the solution of $\min_{Y\in\mathbb{R}^{ms\times s}} \| E_1\tilde{U}_1 - \overline{\tilde{\mathcal{H}}}_m Y \|_F$.
25. Compute the approximation solution $X_m^{WBC} = X_0 + \tilde{\mathcal{L}}_m Y_m^{WBC}$.
26. Compute $R_m^{WBC} = B - AX_m^{WBC}$.
27. Restart:
28. If $\| R_m^{WBC} \|_F \leq \epsilon$ Stop,
29. else $X_0 = X_m^{WBC}$, $R_0 = R_m^{WBC}$ and go to 2.

From the last relation, by a simple computation, we deduce that

$$\begin{aligned}\tilde{L}_{m+1}\tilde{H}_{m+1,m} &= A\tilde{L}_m - \sum_{k=1}^{m} \tilde{L}_k \tilde{H}_{km} \\ &= \sum_{k=1}^{m} L_{k+1} H_{k+1,k} F_{km} + \sum_{k=1}^{m}\sum_{i=1}^{k} L_i (H_{ik}F_{km} - F_{ik}\tilde{H}_{km}).\end{aligned}$$

So, we have

$$\tilde{L}_{m+1} = \sum_{k=1}^{m} L_{k+1} H_{k+1,k} F_{km} \tilde{H}_{m+1,m}^{-1} + \sum_{k=1}^{m}\sum_{i=1}^{k} L_i (H_{ik}F_{km} - F_{ik}\tilde{H}_{km})\tilde{H}_{m+1,m}^{-1}$$

and

$$\tilde{L}_{m+1} = \sum_{i=1}^{m+1} L_i F_{i,m+1}.$$

Thus the relation (12) is true for $j = m + 1$ and we get (12). By using $\mathcal{L}_m^L \mathcal{L}_m = I$, we obtain the relation (13). From (8) and (12), we have

$$A\mathcal{L}_m\mathcal{F}_m = \mathcal{L}_{m+1}\mathcal{F}_{m+1}\overline{\tilde{\mathcal{H}}}_m,$$

which implies that

$$\mathcal{L}_{m+1}\overline{\mathcal{H}}_m\mathcal{F}_m = \mathcal{L}_{m+1}\mathcal{F}_{m+1}\overline{\tilde{\mathcal{H}}}_m,$$

by premultiplying it by $\mathcal{L}_{m+1}^L$, we get

$$\overline{\mathcal{H}}_m\mathcal{F}_m = \mathcal{F}_{m+1}\overline{\tilde{\mathcal{H}}}_m,$$

premultiplying the above equation by $\mathcal{F}_{m+1}^{-1}$, yields the relation (14). □

Now, by partitioning $\mathcal{F}_{m+1}$ as

$$\mathcal{F}_{m+1} = \begin{pmatrix} \mathcal{F}_m & F_{m+1} \\ 0_{s\times ms} & F_{m+1,m+1} \end{pmatrix}, \tag{15}$$

where $F_{m+1} \in \mathbb{R}^{ms\times s}$ and $F_{m+1,m+1} \in \mathbb{R}^{s\times s}$, we can state the following proposition.

Proposition 2. *Under the same assumptions as in Proposition 1, we can express $\tilde{\mathcal{H}}_m$ and $\mathcal{H}_m$ in terms of each other by*

$$\tilde{\mathcal{H}}_m = \mathcal{F}_m^{-1}\mathcal{H}_m\mathcal{F}_m - \mathcal{F}_m^{-1}F_{m+1}F_{m+1,m+1}^{-1}H_{m+1,m}F_{mm}E_m^T, \tag{16}$$

$$\mathcal{H}_m = \mathcal{F}_m\tilde{\mathcal{H}}_m\mathcal{F}_m^{-1} + F_{m+1}F_{m+1,m+1}^{-1}H_{m+1,m}E_m^T. \tag{17}$$

Proof. By using (15) we have

$$\mathcal{F}_{m+1}^{-1} = \begin{pmatrix} \mathcal{F}_m^{-1} & -\mathcal{F}_m^{-1}F_{m+1}F_{m+1,m+1}^{-1} \\ 0_{s\times ms} & F_{m+1,m+1}^{-1} \end{pmatrix}.$$

From (14), we can write

$$\tilde{\mathcal{H}}_m = \mathcal{F}_m^{-1}\mathcal{H}_m\mathcal{F}_m - \mathcal{F}_m^{-1}F_{m+1}F_{m+1,m+1}^{-1}H_{m+1,m}E_m^T\mathcal{F}_m.$$

We get (16), since $\mathcal{F}_m$ is block upper triangular matrix and we have $E_m^T\mathcal{F}_m = F_{mm}E_m^T$. Multiplying the last equation on the left by $\mathcal{F}_m$ and on the right by $\mathcal{F}_m^{-1}$, yields the relation (17). □

Also, we can describe a link between BCMRH(m) and its weighted version. From (9) and (12), we have the relation

$$X_m^{WBC} - X_0 = \tilde{\mathcal{L}}_m Y_m^{WBC} = \mathcal{L}_m\hat{Y}_m^{WBC},$$

where $Y_m^{WBC}, \hat{Y}_m^{WBC} \in \mathbb{R}^{ms\times s}$ satisfy the relation $Y_m^{WBC} = \mathcal{F}_m^{-1}\hat{Y}_m^{WBC}$. Since Y_m^{WBC} is the solution of the minimization problem

$$\min_{Y\in\mathbb{R}^{ms\times s}} \|E_1\tilde{U}_1 - \overline{\tilde{\mathcal{H}}}_m Y\|_F,$$

we have

$$\begin{aligned}
\hat{Y}_m^{WBC} &= \operatorname*{argmin}_{\hat{Y}\in\mathbb{R}^{ms\times s}} \|E_1\tilde{U}_1 - \overline{\tilde{\mathcal{H}}}_m\mathcal{F}_m^{-1}\hat{Y}\|_F \\
&= \operatorname*{argmin}_{\hat{Y}\in\mathbb{R}^{ms\times s}} \|E_1\tilde{U}_1 - \mathcal{F}_{m+1}^{-1}\overline{\mathcal{H}}_m\hat{Y}\|_F \\
&= \operatorname*{argmin}_{\hat{Y}\in\mathbb{R}^{ms\times s}} \|\mathcal{F}_{m+1}^{-1}\left(E_1U_1 - \overline{\mathcal{H}}_m\hat{Y}\right)\|_F \\
&= \operatorname*{argmin}_{\hat{Y}\in\mathbb{R}^{ms\times s}} \|E_1U_1 - \overline{\mathcal{H}}_m\hat{Y}\|_{\mathcal{F}_{m+1}^{-T}\mathcal{F}_{m+1}^{-1}}.
\end{aligned}$$

So, $\hat{Y}_m^{WBC}$ is the solution of the same optimization problem as for BCMRH(m) but with the norm corresponding to the symmetric matrix $\mathcal{F}_{m+1}^{-T}\mathcal{F}_{m+1}^{-1}$ instead of the Frobenius one.

4. The fixed and flexible preconditioned block CMRH method

It is well known that the performance of a Krylov subspace method can be improved when combined with a suitable preconditioner. In this section we consider the block CMRH algorithm with fixed and flexible right preconditioners (denoted by BCMRH(m) and FBCMRH(m), respectively) and we interpret the relation between them. In the sequel in order to avoid confusion, we use the superscript BC and FBC to denote the quantities from BCMRH and FBCMRH algorithms, respectively.

Consider the following modified linear system with multiple right hand sides:

$$AM^{-1}(MX) = B, \tag{18}$$

where M is some appropriate preconditioner. We apply Algorithm 1 to solve the system (18). Let $X_0 \in \mathbb{R}^{n\times s}$ be the initial matrix and R_0 its residual. The block Hessenberg process generates a block upper Hessenberg matrix $\overline{\mathcal{H}}_m \in \mathbb{R}^{(m+1)s\times ms}$, and a basis $\mathcal{L}_m^{BC} = [L_1^{BC}, \ldots, L_m^{BC}]$ of the Krylov subspace $\mathcal{K}_m(AM^{-1}, R_0)$ such that

$$\begin{cases} R_0 = L_1^{BC}U_1, \\ L_{k+1}^{BC}H_{k+1,k} = AM^{-1}L_k^{BC} - \sum_{j=1}^{k} L_j^{BC}H_{j,k}, \qquad \text{for } k = 1, \ldots, m. \end{cases} \tag{19}$$

By defining

$$Z_k^{BC} = M^{-1}L_k^{BC}, \tag{20}$$

then the following relation holds:

$$A\mathcal{Z}_k^{BC} = AM^{-1}\mathcal{L}_k^{BC} = \mathcal{L}_{k+1}^{BC}\overline{\mathcal{H}}_k, \tag{21}$$

where $\mathcal{Z}_m^{BC} = [Z_1^{BC}, \ldots, Z_m^{BC}] = M^{-1}\mathcal{L}_m^{BC}$. The block CMRH method computes, from an initial guess X_0, the approximation $X_m^{BC} = X_0 + M^{-1}\mathcal{L}_m^{BC}Y_m^{BC}$ to the matrix equation (18), where Y_m^{BC} is taken as the solution of the minimizing problem $\min_{Y\in\mathbb{R}^{ms\times s}}\|E_1U_1 - \overline{\mathcal{H}}_mY\|_F$. Using the relation (21), the residual R_m^{BC} can be written as $R_m^{BC} = \mathcal{L}_{m+1}^{BC}\left(E_1U_1 - \overline{\mathcal{H}}_mY_m^{BC}\right)$. In this case, the main part of Algorithm 1 remains the same except that the lines 11 and 25 must be changed as follows:

11. Compute $Z_k^{BC} = M^{-1}L_k^{BC}$ and $T = AZ_k^{BC}$,
25. Compute the approximation solution $X_m^{BC} = X_0 + M^{-1}\mathcal{L}_m^{BC}Y_m^{BC}$.

Suppose now that the preconditioner could change at every step, i.e., that Z_k is given by

$$Z_k^{FBC} = M_k^{-1}L_k^{FBC}. \tag{22}$$

Then it would be natural to compute the approximate solution as

$$X_m^{FBC} = X_0 + \mathcal{Z}_m^{FBC}Y_m^{FBC},$$

in which $\mathcal{Z}_m^{FBC} = [Z_1^{FBC}, \ldots, Z_m^{FBC}]$, and Y_m^{FBC} is computed as before, as the solution to the least-squares problem in line 24 of Algorithm 1. In this case the lines 11 and 25 are, respectively, changed to

11. Compute $Z_k^{FBC} = M_k^{-1}L_k^{FBC}$ and $T = AZ_k^{FBC}$.
25. Set $\mathcal{Z}_m^{FBC} = [Z_1^{FBC}, \ldots, Z_m^{FBC}]$ and compute the approximation solution $X_m^{FBC} = X_0 + \mathcal{Z}_m^{FBC}Y_m^{FBC}$.

Suppose that a second iterative solver is employed to solve approximately $M_kZ_k^{FBC} = L_k^{FBC}$. As in [31,34,39], we can rewrite (22) as

$$Z_k^{FBC} = M_k^{-1}L_k^{FBC} = M^{-1}L_k^{FBC} + G_k^{FBC}, \tag{23}$$

where G_k^{FBC} is the error matrix associated with the inner solver at step k.

Lemma 2. *Let Z_k^{BC} and Z_k^{FBC} defined by (20) and (22), respectively. If G_k^{FBC} is the kth error defined in (23), then*

$$Z_k^{BC} \in \mathcal{S}_m, \qquad \text{for } k = 1, \ldots, m, \tag{24}$$

where the subspace $\mathcal{S}_m = \operatorname{span}\{Z_i^{FBC}, (M^{-1}A)^jG_i^{FBC}\}_{j=0,\ldots,m-1}^{i=1,\ldots,m}$.

Proof. The proof is similar to that of Lemma 1 in [39]. □

By using Lemma 2, we give the following theorem that relates the mth residual norm of BCMRH(m) and FBCMRH(m).

Theorem 1. *Let R_m^{BC} and R_m^{FBC} be the residuals derived from the mth steps of BCMRH(m) and FBCMRH(m), respectively. Define the block error matrix $\mathcal{G}_m^{FBC} = [G_1^{FBC}, \ldots, G_m^{FBC}]$, where G_k^{FBC} are the error matrices defined in (23), for $k = 1, \ldots, m$. Then there exists $\mathcal{D}_j^{FBC} \in \mathbb{R}^{ms\times s}$ such that*

$$\|R_m^{FBC}\|_F \le \kappa_F(\mathcal{L}_{m+1}^{FBC})\left(\|R_m^{BC}\|_F + \sum_{j=0}^{m-1}\|A(M^{-1}A)^j\mathcal{G}_m^{FBC}\mathcal{D}_j^{FBC}\|_F\right), \tag{25}$$

where the condition number $\kappa_F(\mathcal{L}_{m+1}^{FBC}) = \|(\mathcal{L}_{m+1}^{FBC})^L\|_F\|\mathcal{L}_{m+1}^{FBC}\|_F$.

Proof. The proof is similar to that of Theorem 1 in [39]. □

4.1. Relation between FBCMRH(m) and FBGMRES(m)

In this section, we express a relation between the flexible block CMRH(m) (FBCMRH(m)) method and the flexible block GMRES(m) (FBGMRES(m)) [40] method about the corresponding residual norm. As done in the previous section, we denote by the superscripts BG and FBG the Block GMRES(m) (with fixed right preconditioning) and FBGMRES(m) variables, respectively.

Note that in FBGMRES(m), by defining

$$Z_k^{FBG} = M_k^{-1}V_k^{FBG}, \quad \text{for } k = 1, \ldots, m, \tag{26}$$

the following relation, similar to (21), holds:

$$A\mathcal{Z}_m^{FBG} = \mathcal{V}_{m+1}^{FBG}\overline{\mathcal{H}}_m^{FBG}, \tag{27}$$

where $\mathcal{Z}_m^{FBG} = [Z_1^{FBG}, \ldots, Z_m^{FBG}]$. The following lemma will be useful for the main result of the section.

Table 1
Results for Example 1.

Matrix	Algorithm	$s = 5$			$s = 10$		
		Cycle	CPU	Residual norm	Cycle	CPU	Residual norm
	BCMRH	†	–	–	476	5.88	2.64e−07
A_1	WBCMRH (D_1)	1080	6.96	1.93e−07	166	2.13	2.46e−07
$m = 20$	WBCMRH (D_2)	1237	7.41	1.70e−07	229	3.01	2.34e−07
	FBCMRH	216	40.78	1.81e−07	4	1.44	5.15e−11
	BCMRH	29	0.14	2.46e−07	33	0.39	4.94e−07
A_2	WBCMRH (D_1)	23	0.14	2.07e−07	11	0.15	5.55e−07
$m = 20$	WBCMRH (D_2)	20	0.11	1.09e−07	12	0.16	2.63e−07
	FBCMRH	1	0.03	2.40e−14	1	0.04	6.96e−14
	BCMRH	486	15.55	4.05e−07	652	40.00	5.29e−07
A_3	WBCMRH (D_1)	363	11.86	3.73e−07	259	16.26	5.27e−07
$m = 20$	WBCMRH (D_2)	375	12.19	3.41e−07	254	15.76	5.65e−07
	FBCMRH	1	0.35	5.28e−14	1	0.49	8.89e−14
	BCMRH	149	1.60	3.69e−07	113	2.96	3.63e−07
A_4	WBCMRH (D_1)	104	1.27	3.67e−07	36	1.06	5.31e−07
$m = 30$	WBCMRH (D_2)	143	1.58	2.13e−07	29	0.77	4.20e−07
	FBCMRH	1	0.04	2.75e−14	1	0.07	5.82e−14

Lemma 3. *Let $\mathcal{Z}_m^{FBC}$ and $\mathcal{Z}_m^{FBG}$ be defined in (21) and (27), respectively. Suppose that G_k^{FBC} is the kth error matrix given by (23), and analogously, G_k^{FBG} is the kth error matrix associated with*

$$Z_k^{FBG} = M^{-1}V_k^{FBG} + G_k^{FBG}. \tag{28}$$

Then

$$Z_k^{FBG} \in \mathcal{P}_m, \qquad \text{for } k = 1, \ldots, m, \tag{29}$$

where the subspace $\mathcal{P}_m = \text{span}\{Z_i^{FBC}, (M^{-1}A)^j G_i^{FBC}, (M^{-1}A)^j G_i^{FBG}\}_{j=0,\ldots,m-1}^{i=1,\ldots,m}$.

Proof. The proof is analogous to that of Lemma 3 in [39]. □

Next we present the result concerning the residual norm between FBCMRH(m) and FBGMRES(m).

Theorem 2. *Let R_m^{FBC} and R_m^{FBG} be the residuals derived from the mth steps of FBCMRH(m) and FBGMRES(m), respectively. Suppose that $\mathcal{G}_m^{FBC} = [G_1^{FBC}, \ldots, G_m^{FBC}]$ and $\mathcal{G}_m^{FBG} = [G_1^{FBG}, \ldots, G_m^{FBG}]$ where G_k^{FBC} and G_k^{FBG} are the error matrices defined in (23) and (28), respectively, for $k = 1, \ldots, m$. Then there exists $\mathcal{D}_j^{FBC}$, $\mathcal{D}_j^{FBG} \in \mathbb{R}^{ms\times s}$ such that*

$$\|R_m^{FBC}\|_F \le \kappa_F(\mathcal{L}_{m+1}^{FBC}) \left(\|R_m^{FBG}\|_F + \sum_{j=0}^{m-1} \|A(M^{-1}A)^j \mathcal{G}_m^{FBC} \mathcal{D}_j^{FBC}\|_F + \sum_{j=0}^{m-1} \|A(M^{-1}A)^j \mathcal{G}_m^{FBG} \mathcal{D}_j^{FBG}\|_F \right), \tag{30}$$

where the condition number $\kappa_F(\mathcal{L}_{m+1}^{FBC}) = \|(\mathcal{L}_{m+1}^{FBC})^L\|_F \|\mathcal{L}_{m+1}^{FBC}\|_F$.

Proof. The proof is analogous to that of Theorem 4 in [39]. □

5. Numerical results

In this section, we give some experimental results. We compare the performance of BCMRH(m) (unpreconditioned block CMRH(m) method), WBCMRH(m), FBCMRH(m), FBGMRES(m), and WBGMRES(m) (the restarted weighted block GMRES method [27]) for solving the multiple linear system (1). We have freedom in choosing the inner linear system solver for flexible methods. For illustration, we only consider the block BiCGSTAB (Bl-BiCGSTAB) method ([8], Algorithm 3) since it is easy to be implemented and needs no restarting. The tolerance for inner iteration is 10^{-1} and a maximum of 20 inner iterations is allowed. We mention that this inner linear system solver for FBCMRH method takes very long time in each cycle, but it provides the approximations that have high accuracy and allow FBCMRH to converge very fast (see the number of cycles and CPU time in Tables 1 and 3 for FBCMRH method).

All the numerical experiments were performed in double precision floating point arithmetic in MATLAB R2014a. The machine we have used is a Intel(R) Core(TM) i7, CPU 3.60 GHz, 16.00 GB of RAM. In all the examples, the starting guess was taken to be zero. We consider the right-hand side $B = rand(n, s)$, where function *rand* creates an $n \times s$ random matrix with coefficients uniformly distributed in [0, 1]. We set the number of linear systems $s = 5$ and 10. The stopping criterion

$$\frac{\|R_k\|_F}{\|R_0\|_F} \le 10^{-8}$$

was used and a maximum of 3000 restarts was allowed.

Table 2
Test problems information.

	Matrix property	order	nnz	cond
1	cdde6	961	4681	1.77e+02
2	pde2961	2961	14585	6.42e+02
3	rdb3200l	3200	18880	1.11e+03
4	epb2	25228	175027	2.62e+03
5	Pd	8081	13036	2.62e+11
6	psmigr_3	3140	543160	1.33e+02
7	appu	14000	1853104	1.71e+02
8	Poisson3Db	85623	2374949	1.66e+05
9	FEM_3D_thermal2	147900	3489300	7.69e+03

Table 3
Results for Example 2.

Matrix	Algorithm	$s = 5$			$s = 10$		
		Cycle	CPU	Residual norm	Cycle	CPU	Residual norm
	BCMRH	106	0.52	3.52e−07	124	1.55	4.45e−07
cdde6	WBCMRH (D_1)	43	0.26	3.30e−07	48	0.64	4.70e−07
$m = 20$	WBCMRH (D_2)	49	0.24	2.82e−07	51	0.63	3.66e−07
	FBCMRH	2	0.17	6.97e−09	2	0.27	1.61e−11
	BCMRH	54	0.83	6.72e−07	114	4.12	6.21e−07
pde2961	WBCMRH (D_1)	60	1.10	4.75e−07	79	2.90	9.47e−07
$m = 20$	WBCMRH (D_2)	65	1.02	6.26e−07	81	2.88	8.02e−07
	FBCMRH	2	0.34	2.30e−10	2	0.66	1.54e−12
	BCMRH	395	2.25	6.92e−07	1076	12.94	9.80e−07
rdb3200l	WBCMRH (D_1)	384	3.29	7.20e−07	730	11.36	9.45e−07
$m = 10$	WBCMRH (D_2)	351	2.11	7.28e−07	681	8.59	1.02e−06
	FBCMRH	5	0.46	1.28e−08	5	0.81	9.71e−10
	BCMRH	97	30.37	2.05e−06	150	106.79	2.88e−06
epb2	WBCMRH (D_1)	72	25.01	1.90e−06	104	77.38	2.64e−06
$m = 30$	WBCMRH (D_2)	81	26.32	1.74e−06	113	82.22	2.86e−06
	FBCMRH	3	8.46	6.05e−09	3	26.62	1.01e−07
	BCMRH	830	36.46	9.62e−07	867	84.66	1.58e−06
Pd	WBCMRH (D_1)	356	18.51	2.27e−07	139	15.04	1.05e−06
$m = 20$	WBCMRH (D_2)	191	8.71	6.54e−07	88	8.99	8.10e−07
	FBCMRH	5	2.08	1.58e−08	2	1.82	2.27e−07

In the tables of results, Cycle, CPU, and Residual norm denote, respectively, the number of restarts, the runtime in terms of seconds, and the Frobenius norm of residual.

Example 1. In this example we consider four matrices. The first matrix is given in [37,41]:

$$A_1 = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ a_1 & 1 & 1 & \cdots & 1 & 1 \\ a_1 & a_2 & 1 & \cdots & 1 & 1 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ a_1 & a_2 & a_3 & \cdots & a_{n-1} & 1 \end{pmatrix},$$

with $a_i = 1 + i\epsilon$. We consider a system of size $n = 300$ and we take $\epsilon = 10^{-2}$. The second matrix is a tridiagonal matrix with entries 1, 2, 3, …, 999, 1000 on the main diagonal and -0.1's on the super diagonal and 0.1's on the lower diagonal. We consider the $n \times n$ matrix $A_3 = SBS^{-1}$, where

$$S = \begin{pmatrix} 1 & \beta & & & \\ & 1 & \beta & 0 & \\ & & \ddots & \ddots & \\ & & & \ddots & \beta \\ & 0 & & & 1 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 1 & & & & \\ & 1+\alpha & & 0 & \\ & & 3 & & \\ & 0 & & \ddots & \\ & & & & n \end{pmatrix},$$

with $\beta = 0.9$, $\alpha = 1$, and $n = 1000$. This matrix was also considered in [37,42]. The matrix A_4 is a bidiagonal with entries 0.1, 1, 2, 3, …, 999 on the main diagonal and 1's on the super diagonal. The results obtained for these matrices are presented in Table 1. In this Table, "†" signifies that the maximum allowed number of restarts was reached before the convergence.

Table 4
Results for Example 3.

Matrix	Algorithm	$s = 5$			$s = 10$		
		Cycle	CPU	Residual norm	Cycle	CPU	Residual norm
	WBCMRH (D_1)	5	0.15	2.57e−08	5	0.26	2.04e−08
psmigr_3	WBGMRES (D_1)	4	0.12	9.93e−08	4	0.20	8.04e−09
$m = 10$	FBCMRH	1	0.21	1.30e−11	1	0.34	6.56e−10
	FBGMRES	1	0.25	1.01e−13	1	0.41	1.89e−13
	WBCMRH (D_1)	6	1.62	3.08e−07	6	3.41	9.49e−07
appu	WBGMRES (D_1)	5	1.26	7.93e−07	5	3.50	8.49e−07
$m = 20$	FBCMRH	1	3.42	4.57e−13	1	7.01	8.58e−13
	FBGMRES	1	3.29	6.80e−13	1	7.08	1.08e−12
	WBCMRH (D_1)	23	56.71	3.67e−06	29	147.59	3.52e−06
Poisson3Db	WBGMRES (D_1)	16	77.40	1.59e−06	15	158.68	4.31e−06
$m = 40$	FBCMRH	1	41.00	1.67e−10	1	80.88	2.87e−10
	FBGMRES	1	40.71	1.66e−10	1	66.86	2.54e−10
	WBCMRH (D_1)	31	81.01	2.68e−06	31	157.06	6.48e−06
FEM_3D_thermal2	WBGMRES (D_1)	17	89.97	2.96e−06	16	172.39	4.30e−06
$m = 30$	FBCMRH	1	35.21	7.58e−10	1	64.83	1.32e−10
	FBGMRES	1	42.35	4.96e−12	1	79.95	6.96e−12

For Examples 2 and 3, we used some nonsymmetric matrices from the University of Florida Sparse Matrix Collection [43]. These matrices with their properties are shown in Table 2.

Example 2. In this example, we use the nonsymmetric matrices 1–5 of Table 2. Table 3 reports the results of these test problems.

In the following, we summarize the observation from Tables 1 and 3. In all cases, we observe that the weighted BCMRH algorithm with different weighting matrices needs much fewer iterations and much less CPU time than the standard BCMRH algorithm and they reach about the same accuracy in terms of the residual norm. More precisely, WBCMRH performs better than the standard BCMRH algorithm, using D_1 and D_2 as the weighting matrices. Furthermore, FBCMRH(m) is superior to the other two algorithms in terms of the number of restarts, CPU time, and accuracy (except for example A_1 with $s = 5$). All these demonstrate that the WBCMRH(m) and FBCMRH(m) algorithms have the potential to improve the convergence, and also they are more robust and efficient than the standard block CMRH algorithm. According to the CPU time given in Tables 1 and 3, we can see that the compared methods often give a similar behavior when s increases.

Finally in the following example we compare FBCMRH, WBCMRH, FBGMRES, and WBGMRES methods in terms of the number of restarts, CPU time, and accuracy.

Example 3. The test problems 6–9 of Table 2 are used for this example. The results are shown in Table 4. For weighted methods we presented the results obtained with the weighting matrix D_1. Similar results are obtained with the weighting matrix D_2.

As can be seen from Table 4, FBCMRH, WBCMRH methods and FBGMRES, WBGMRES methods give a similar behavior, respectively. In addition, for all problems, the number of restarts of the WBGMRES is less than that of the WBCMRH and for large matrices *Poisson3Db* and *FEM_3D_thermal2*, the CPU time for the WBCMRH is smaller than the one for the WBGMRES. Finally for these large matrices, we observe that FBCMRH and FBGMRES are superior to the other methods in terms of the number of restarts, CPU time, and accuracy.

6. Conclusion

Two new variants of the block CMRH algorithm for solving nonsymmetric systems with multiple right-hand sides were presented and studied. The relations between WBCMRH(m) and BCMRH(m), FBCMRH(m) and BCMRH(m) are examined. Theoretical results concerning the residual norm between FBGMRES(m) and FBCMRH(m) are presented. Finally, the numerical experiments presented in this paper show that the weighted block CMRH method and the flexible block CMRH method are more robust than the block CMRH method. In addition, the WBCMRH, FBCMRH algorithms are comparable to the WBGMRES, FBGMRES algorithms in terms of the number of restarts and the CPU time needed for convergence.

Acknowledgments

We would like to thank the referees for their valuable remarks and helpful suggestions.

References

[1] R.W. Freund, M. Malhotra, A block QMR algorithm for non-Hermitian systems with multiple right-hand sides, Linear Algebra Appl. 254 (1997) 119–157.
[2] Y. Saad, Iterative Methods for Sparse Linear Systems, SIAM, 2003.
[3] V. Simoncini, E. Gallopoulos, An iterative method for nonsymmetric systems with multiple right-hand sides, SIAM J. Sci. Comput. 16 (1995) 917–933.
[4] V. Simoncini, E. Gallopoulos, A hybrid block GMRES method for nonsymmetric systems with multiple right-hand sides, J. Comput. Appl. Math. 66 (1996) 457–469.
[5] D.P. O'Leary, The block conjugate gradient algorithm and related methods, Linear Algebra Appl. 29 (1980) 293–322.
[6] B. Vital, Etude de quelques méthodes de résolution de problemes linéaires de grande taille sur multiprocesseur, (Ph.D. thesis), Université de Rennes, 1990.
[7] A. El Guennouni, K. Jbilou, H. Sadok, The block lanczos method for linear systems with multiple right-hand sides, Appl. Numer. Math. 51 (2004) 243–256.
[8] A. El Guennouni, K. Jbilou, H. Sadok, A block version of bicgstab for linear systems with multiple right-hand sides, Electron. Trans. Numer. Anal. 16 (2003) 129–142.
[9] S. Karimi, F. Toutounian, The block least squares method for solving nonsymmetric linear systems with multiple right-hand sides, Appl. Math. Comput. 177 (2006) 852–862.
[10] A.T. Chronopoulos, S-step iterative methods for (non)symmetric (in)definite linear systems, SIAM J. Numer. Anal. 28 (1991) 1776–1789.
[11] A.T. Chronopoulos, A.B. Kucherov, Block s-step Krylov iterative methods, Numer. Linear Algebra Appl. 17 (2010) 3–15.
[12] M. Addam, M. Heyouni, H. Sadok, The block Hessenberg process for matrix equations, Electron. Trans. Numer. Anal. 46 (2017) 460–473.
[13] S. Amini, F. Toutounian, M. Gachpazan, The block CMRH method for solving nonsymmetric linear systems with multiple right-hand sides, J. Comput. Appl. Math. 337 (2018) 166–174.
[14] M. Bellalij, K. Jbilou, H. Sadok, New convergence results on the global GMRES method for diagonalizable matrices, J. Comput. Appl. Math. 219 (2008) 350–358.
[15] K. Jbilou, A. Messaoudi, H. Sadok, Global FOM and GMRES algorithms for matrix equations, Appl. Numer. Math. 31 (1999) 49–63.
[16] K. Jbilou, H. Sadok, Global Lanczos-Based Methods with Applications, Technical Report LMA 42, Université du Littoral, Calais, France, 1997.
[17] K. Jbilou, H. Sadok, A. Tinzefte, Oblique projection methods for linear systems with multiple right-hand sides, Electron. Trans. Numer. Anal. 20 (2005) 119–138.
[18] J. Zhang, H. Dai, Global CGS algorithm for linear systems with multiple right-hand sides, Numer. Math. A: J. Chin. Univ. 30 (2008) 390–399.
[19] J. Zhang, H. Dai, J. Zhao, Generalized global conjugate gradient squared algorithm, Appl. Math. Comput. 216 (2010) 3694–3706.
[20] F. Toutounian, S. Karimi, Global least squares method (Gl-LSQR) for solving general linear systems with several right-hand sides, Appl. Math. Comput. 178 (2006) 452–460.
[21] J. Zhang, H. Dai, J. Zhao, A new family of global methods for linear systems with multiple right-hand sides, J. Comput. Appl. Math. 236 (2011) 1562–1575.
[22] M. Heyouni, The global Hessenberg and CMRH methods for linear systems with multiple right-hand sides, Numer. Algorithms 26 (2001) 317–332.
[23] C. Gu, Z. Yang, Global SCD algorithm for real positive definite linear systems with multiple right-hand sides, Appl. Math. Comput. 189 (2007) 59–67.
[24] M. Heyouni, A. Essai, Matrix Krylov subspace methods for linear systems with multiple right-hand sides, Numer. Algorithms 40 (2005) 137–156.
[25] A. Essai, Weighted FOM and GMRES for solving nonsymmetric linear systems, Numer. Algorithms 18 (1998) 277–292.
[26] M. Mohseni Mohgadam, F. Panjeh Ali Beik, A new weighted global full orthogonalization method for solving nonsymmetric linear systems with multiple right-hand sides, Int. Electron. J. Pure Appl. Math. 2 (2010) 47–67.
[27] A. Imakura, L. Du, H. Tadano, A weighted block GMRES method for solving linear systems with multiple right-hand sides, JSIAM Lett. 5 (2013) 65–68.
[28] F. Panjeh Ali Beik, D. Khojasteh Salkuyeh, Weighted versions of Gl-FOM and Gl-GMRES for solving general coupled linear matrix equations, Comput. Math. Math. Phys. 55 (2015) 1606–1618.
[29] Y. Saad, A flexible inner-outer preconditioned GMRES algorithm, SIAM J. Sci. Comput. 14 (1993) 461–469.
[30] H.A. Van der Vorst, C. Vuik, GMRESR: a family of nested GMRES methods, Numer. Linear Algebra Appl. 1 (1994) 369–386.
[31] G.H. Golub, Q. Ye, Inexact preconditioned conjugate gradient method with inner-outer iteration, SIAM J. Sci. Comput. 21 (1999) 1305–1320.
[32] A.V. Knyazev, I. Lashuk, Steepest descent and conjugate gradient methods with variable preconditioning, SIAM J. Matrix Anal. Appl. 29 (2007) 1267–1280.
[33] Y. Notay, Flexible conjugate gradients, SIAM J. Sci. Comput. 22 (2000) 1444–1460.
[34] D.B. Szyld, J.A. Vogel, FQMR: A flexible quasi-minimal residual method with inexact preconditioning, SIAM J. Sci. Comput. 23 (2001) 363–380.
[35] J.A. Vogel, Flexible BiCG and flexible Bi-CGSTAB for nonsymmetric linear systems, Appl. Math. Comput. 188 (2007) 226–233.
[36] V. Simoncini, D.B. Szyld, Flexible inner-outer Krylov subspace methods, SIAM J. Numer. Anal. 40 (2002) 2219–2239.
[37] H. Sadok, CMRH: A new method for solving nonsymmetric linear systems based on the Hessenberg reduction algorithm, Numer. Algorithms 20 (1999) 303–321.
[38] N. Azizi zadeh, A. Tajaddini, G. Wu, A weighted global GMRES algorithm with deflation for solving large Sylvester matrix equations, 2017, arXiv: 1706.01176.
[39] K. Zhang, C. Gu, A flexible CMRH algorithm for nonsymmetric linear systems, J. Appl. Math. Comput. 45 (2014) 43–61.
[40] H. Calandra, S. Gratton, J. Langou, X. Pinel, X. Vasseur, Flexible variants of block restarted GMRES methods with application to geophysics, SIAM J. Sci. Comput. 34 (2012) A714–A736.
[41] R.T. Gregory, D.L. Karney, A Collection of Matrices for Testing Computational Algorithms, Wiley, New York, 1969.
[42] Y. Huang, H.A. Van der Vorst, Some Observations on the Convergence Behaviour of GMRES, Report 89-09, Delft Univ. Technology, 1989.
[43] T. Davis, Y. Hu, The university of florida sparse matrix collection, ACM Trans. Math. Software 38 (2011) 1–25 Avaiable online at http://www.cise.ufl.edu/research/sparse/matrices/.