Journal of Computational and Applied Mathematics 321 (2017) 78–89

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

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

Global LSMR(Gl-LSMR) method for solving general linear systems with several right-hand sides

M. Mojarrab [a,*], F. Toutounian [b,c]

[a] Department of Mathematics, University of Sistan and Baluchestan, Zahedan, Iran

[b] The Center of Excellence on Modeling and Control Systems, Ferdowsi University of Mashhad, Iran

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

ARTICLE INFO

Article history:
Received 22 April 2015
Received in revised form 28 May 2016

MSC:
15A06
65F10
65F20

Keywords:
LSMR method
Bidiagonalization
Global methods
Iterative methods
Multiple right-hand sides

ABSTRACT

The global solvers are an attractive class of iterative solvers for solving linear systems with multiple right-hand sides. In this paper, first, a new global method for solving general linear systems with several right-hand sides is presented. This method is the global version of the LSMR algorithm presented by Fong and Saunders (2011). Then, some theoretical properties of the new method are discussed. Finally, numerical experiments from real applications are used to confirm the effectiveness of the proposed method.

1. Introduction

We consider the linear systems of the form

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

where A is an $n \times n$ nonsingular and nonsymmetric real matrix, B and X are $n \times s$ rectangular matrices with usually $s \ll n$. It is well known that if the coefficient matrix A is large and sparse or sometimes not readily available, then iterative solvers may become the only choice. These solvers are possibly classifiable into the three classes which are briefly described in the next three paragraphs.

One class of solvers for solving the problem (1) is the block solvers which are much more efficient when the matrix A is relatively dense and preconditioners are used. The first block solvers are the block conjugate gradient algorithm (B-CG) and the block biconjugate gradient algorithm (B-BCG) proposed in [1]. The generalized of the block CG algorithm and its variable version can be observed in [2]. Haase and Reitzinger showed that applied problems can be solved by using the block variants of the CG method and combining them with cache aware algorithms [3]. Vital (1990) designed the block generalized minimal residual algorithm (Bl-GMRES) [4]. Then the block variants of the GMRES method developed by some authors [5–12]. For symmetric linear systems, an adaptive block Lanczos algorithm and a block version of minimal residual method (Minres) are devised in [13]. For nonsymmetric problems, the B-BCG algorithm [1,14], the block quasi-minimal residual algorithm

* Corresponding author.
E-mail addresses: ma_mojarrab@math.usb.ac.ir (M. Mojarrab), toutouni@math.um.ac.ir (F. Toutounian).

http://dx.doi.org/10.1016/j.cam.2017.02.011
0377-0427/

(Bl-QMR) [15], the block BiCGSTAB algorithm (Bl-BiCGSTAB) [16], the block Lanczos method [17], and the block least squares with QR decomposition algorithm (Bl-LSQR) have been developed [18]. Recently, two block variants of the least squares minimal residual method (Bl-LSMR) have been presented [19,20].

Another class is the seed methods, which consist of selecting a single system as the seed system and generating the corresponding Krylov subspace and then projecting all the residuals of the other linear systems onto the same Krylov subspace to find new approximate solutions as initial approximations. These methods were presented by some authors [21–23]. The seed conjugate gradient method was studied in [24,25]. Also, the seed GMRES method was offered in [11]. Abdel-Rehim et al. (2008) proposed the improved seed conjugate gradient for symmetric positive definite linear equations with multiple right-hand sides [26].

The last class is the global methods, which are based on using a global projection process onto a matrix Krylov subspace. Jbilou et al. (1997) obtained global Lanczos-based methods [27]. Jbilou et al. (1999) presented the global full orthogonalization method (Gl-FOM) and the global GMRES method (Gl-GMRES) for the nonsymmetric matrix equation (1) [28]. Heyouni (2001) introduced two new methods for solving large sparse non-symmetric linear systems with several right-hand sides that are called the global hessenberg (Gl-Hess) and the global changing minimal residual method based on the hessenberg process (Gl-CMRH). Both of these methods are less expensive than the Gl-FOM and Gl-GMRES [29]. In the case where the coefficient matrix A is nonsymmetric, Jbilou et al. (2005) introduced the global biconjugate gradient algorithm (Gl-BCG) and the global BiCGSTAB algorithm (Gl-BiCGSTAB) [30]. In order to speed-up the convergence rate and retention of information Gl-FOM and Gl-GMRES methods, Lin (2005) proposed the implicitly restarted Gl-FOM (GLFOM-IR) and the implicitly restarted Gl-GMRES (GLGMRES-IR) [31]. By definition the global generalized hessenberg method for solving multiple linear systems, Heyouni and Essai (2005) showed how to derive global Galerkin methods (like Gl-FOM and Gl-BCG) and global minimal residual norm methods (such as Gl-GMRES and Gl-CMRH). Also, in order to speed-up the convergence of the global methods, they applied a weighted technique [32]. Toutounian and Karimi (2006) proposed the global least-squares method (Gl-LSQR). In this method, the basis vectors are not needed to store [33]. Salkuyeh (2006) extracted the global conjugate gradient method (Gl-CG) and the global conjugate residual method (Gl-CR) from the Gl-FOM and the Gl-GMRES algorithms for solving symmetric linear systems of equations with multiple right-hand sides, respectively [34]. By introduction of the concept of semi conjugate direction matrices, Gu and Yang (2007) presented the global semi-conjugate direction method (Gl-SCD). In the case where the coefficient A is symmetric positive definite, the method does not fail [35]. Zhang and Dai (2008) presented a new transpose-free global method called the global conjugate gradient squared algorithm (Gl-CGS) for solving large non-symmetric linear systems with multiple right-hand sides [36]. Bellalij et al. (2008) developed new convergence results for the global GMRES method when applied to multiple linear systems. They considered the case where the matrix A is diagonalizable and the case of normal matrices [37]. Zhang et al. (2010) introduced the generalized Gl-CGS algorithm that is less expensive the Gl-CGS method [38]. Zhang et al. (2011) derived a new family of global A-biorthogonal methods which contain the global bi-conjugate residual algorithm (Gl-BCR). Also, they offered its improved version which is called the global conjugate residual squared algorithm (Gl-CRS) [39].

In this paper, we present a global version of the LSMR algorithm [40] for solving the problem (1). The algorithm will be derived from the LSMR algorithm in the same way as the global LSQR method was derived from the LSQR algorithm [33]. In practice we observe that although the global LSQR and the global LSMR algorithms ultimately converge to similar points, it is safer to use the global LSMR in situations where the solver must be terminated early.

The article is organized as follows. In Section 2, first, we recall some notations of the global methods and Global Bidiag 1 Algorithm [33], then we present the global version of the LSMR algorithm and some properties. Section 3 is devoted to numerical experiments to illustrate the efficiency of the new method. Finally, some concluding remarks are the subject of Section 4.

2. Notations, Global Bidiag 1 algorithm, and the global LSMR method

In this section, first, we give some notations and properties used in the global methods. Then, we remind Global Bidiag 1 [33] which reduces matrix A to the lower bidiagonal form. Finally, we present the global LSMR method for solving Eq. (1).

2.1. Notations and Global Bidiag 1 algorithm

Throughout this paper, we use the following notations. For two $n \times s$ matrices X and Y, the inner product between X and Y is denoted by $\langle X, Y\rangle$ and defined by $\langle X, Y\rangle = \mathrm{tr}(X^T Y)$, where $\mathrm{tr}(Z)$ denotes the trace of the square matrix Z. The associated norm is the Frobenius norm defined by $\|X\|_F = \left(\langle X, X\rangle\right)^{\frac{1}{2}}$. We will use the notation $\langle \cdot, \cdot\rangle_2$ for the usual inner product in $\mathbb{R}^n$ and the associated norm denoted by $\|\cdot\|_2$. Vectors $e_1, \ldots, e_k$ denote the columns of an identity matrix. Notation $A \otimes B$ is Kronecker product and for an $m \times n$ matrix A, $vec(A)$ is defined by the following

$$vec(A) = \begin{pmatrix} a_1^T & a_2^T & \cdots & a_n^T \end{pmatrix}^T,$$

where a_k is the kth column of A. Finally, 0_s and I_s will denote the zero and the identity matrices in $\mathbb{R}^{s\times s}$.

We use the notation $*$ for the following product:

$$\mathcal{V}_k * \gamma = \sum_{j=1}^{k} V_j \gamma_j,$$

where $\mathcal{V}_k = (V_1, V_2, \ldots, V_k)$ with $V_i \in \mathbb{R}^{n\times s}$, $i = 1, 2, \ldots, k$, and $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_k)^T$ is a vector of $\mathbb{R}^k$. By the same way, we set

$$\mathcal{V}_k * T = (\mathcal{V}_k * T_{.,1}, \mathcal{V}_k * T_{.,2}, \ldots, \mathcal{V}_k * T_{.,k}),$$

where $T_{.,j}$ is jth column of the matrix $T \in \mathbb{R}^{k\times k}$. It is easy to see that the following relations are satisfied:

$$\mathcal{V}_k * (\gamma + \eta) = \mathcal{V}_k * \gamma + \mathcal{V}_k * \eta \quad \text{and} \quad (\mathcal{V}_k * T) * \gamma = \mathcal{V}_k * (T\gamma),$$

where γ and η are two vectors of $\mathbb{R}^k$.

Now, we remind Global Bidiag 1 [33] which is based on Bidiag 1 [41] and reduces A to the lower bidiagonal form. The global Bidiag 1 procedure constructs the sets of the $n \times s$ block vectors $V_1, V_2, \ldots$ and $U_1, U_2, \ldots$ such that $\langle V_i, V_j\rangle_F = 0$, $\langle U_i, U_j\rangle_F = 0$, for $i \neq j$, and $\|V_i\|_F = 1$, $\|U_i\|_F = 1$, i.e., they form two F-orthonormal basis of $\mathbb{R}^{n\times ks}$.

Global Bidiag 1 algorithm (Starting matrix B; reduction to lower bidiagonal form)

$$\begin{aligned} &\beta_1 U_1 = B, \qquad \alpha_1 V_1 = A^T U_1, \\ &\left.\begin{aligned} &\beta_{i+1} U_{i+1} = AV_i - \alpha_i U_i, \\ &\alpha_{i+1} V_{i+1} = A^T U_{i+1} - \beta_{i+1} V_i, \end{aligned}\right\} \quad i = 1, 2, \ldots. \end{aligned} \tag{2}$$

The scalars $\alpha_i \geq 0$ and $\beta_i \geq 0$ are chosen so that $\|U_i\|_F = \|V_i\|_F = 1$.

We set

$$\mathcal{U}_k \equiv \begin{pmatrix} U_1, & U_2, & \cdots, & U_k \end{pmatrix}, \qquad \mathcal{V}_k \equiv \begin{pmatrix} V_1, & V_2, & \cdots, & V_k \end{pmatrix}, \qquad T_k \equiv \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_k & \alpha_k \\ & & & \beta_{k+1} \end{pmatrix}.$$

Now, according to the notation $*$, the recurrence relations (2) may be written as

$$\begin{aligned} &\mathcal{U}_{k+1} * (\beta_1 e_1) = B, \\ &A\mathcal{V}_k = \mathcal{U}_{k+1} * T_k, \\ &A^T \mathcal{U}_{k+1} = \mathcal{V}_k * T_k^T + \alpha_{k+1} V_{k+1} * e_{k+1}^T. \end{aligned}$$

Using the above notations, we have also the following results which have been proved in [33,28], respectively.

Proposition 1. *Suppose that k step of the procedure Global Bidiag 1 have been taken, then the block vectors $V_1, V_2, \ldots, V_k$ and $U_1, U_2, \ldots, U_{k+1}$ are F-orthonormal basis of the Krylov subspaces $\mathcal{K}_k(A^TA, V_1)$ and $\mathcal{K}_{k+1}(AA^T, U_1)$, respectively.*

Proposition 2. *Let $\mathcal{V}_k$ be the matrix defined by $\mathcal{V}_k = (V_1, V_2, \ldots, V_k)$, where the $n \times s$ matrices V_i, $i = 1, \ldots, k$, are defined by the global Bidiag 1 process. Then we have $\|\mathcal{V}_k * \eta\|_F = \|\eta\|_2$, where η is a vector of $\mathbb{R}^k$.*

2.2. Global LSMR (Gl-LSMR) algorithm

The Gl-LSMR method is an efficient extension of the LSMR method [40] for (1). The LSMR algorithm is based on Golub and Kahan bidiagonalization process [41] for solving the nonsymmetric linear system

$$Ax = b, \quad \text{with } A \in \mathbb{R}^{m\times n} \text{ and } x, b \in \mathbb{R}^n.$$

This method is similar in style to the well-known LSQR method [42] and is equivalent to MINRES [43] applied to the normal equation, so that the quantities $\|A^T r_k\|$ are monotonically decreasing (where $r_k = b - Ax_k$ is the residual for current iterate x_k). In practice, although LSQR and LSMR ultimately converge to similar points, it is safer to use LSMR in situations where the solver must be terminated early [40]. More details about the LSMR algorithm can be found in [40].

Now we describe a Gl-LSMR algorithm for solving linear system (1). At iteration k we seek an approximate solution X_k of the form

$$X_k = \mathcal{V}_k * y_k, \tag{3}$$

where y_k is in $\mathbb{R}^k$.

Let $\overline{\beta}_k \equiv \alpha_k\beta_k$ for all k. Using the notation $*$, from (2), we have

$$\begin{aligned} A^TR_k &= A^TB - A^TAX_k \\ &= \alpha_1\beta_1V_1 - A^TA\mathcal{V}_k * y_k \\ &= \overline{\beta}_1V_1 - \mathcal{V}_{k+1} * \left(\begin{pmatrix} T_k^TT_k \\ \overline{\beta}_{k+1}e_k^T \end{pmatrix} y_k\right) \\ &= \mathcal{V}_{k+1} * \left(\overline{\beta}_1e_1 - \left(\begin{pmatrix} T_k^TT_k \\ \overline{\beta}_{k+1}e_k^T \end{pmatrix} y_k\right)\right), \end{aligned} \tag{4}$$

where $e_1 \in \mathbb{R}^{k+1}$. The global LSMR approximation is the unique matrix of $\mathcal{K}_k(A^TA, V_1)$ which minimizes the Frobenius norm of the matrix A^TR_k. Therefore, from (4), the Frobenius norm of the matrix A^TR_k can be written as

$$\|A^TR_k\|_F = \left\|\mathcal{V}_{k+1} * \left(\overline{\beta}_1e_1 - \left(\begin{pmatrix} T_k^TT_k \\ \overline{\beta}_{k+1}e_k^T \end{pmatrix} y_k\right)\right)\right\|_F.$$

From Proposition 2, we have

$$\|A^TR_k\|_F = \left\|\overline{\beta}_1e_1 - \left(\begin{pmatrix} T_k^TT_k \\ \overline{\beta}_{k+1}e_k^T \end{pmatrix} y_k\right)\right\|_2.$$

Thus

$$\min_{y\in\mathbb{R}^k} \|A^TR_k\|_F = \min_{y\in\mathbb{R}^k} \left\|\overline{\beta}_1e_1 - \left(\begin{pmatrix} T_k^TT_k \\ \overline{\beta}_{k+1}e_k^T \end{pmatrix} y_k\right)\right\|_2. \tag{5}$$

A common technique to solve the least-squares problem (5) is to transform the coefficient matrix $\begin{pmatrix} T_k^TT_k \\ \overline{\beta}_{k+1}e_k^T \end{pmatrix}$ into upper triangular form using the QR decomposition [44]. First, we define a unitary matrix Q_k such that

$$Q_kT_k = \begin{pmatrix} \mathcal{R}_k \\ 0 \end{pmatrix} = \begin{pmatrix} \rho_1 & \theta_2 & & & \\ & \rho_2 & \theta_3 & & \\ & & \ddots & \ddots & \\ & & & \rho_{k-1} & \theta_k \\ & & & & \rho_k \\ & & & & 0 \end{pmatrix},$$

where ρ_l and θ_l are scalers. This QR factorization can be done in a progressive manner, i.e., at iteration k, we can eliminate β_{k+1}, by using the simple recurrence relation

$$\begin{pmatrix} c_k & s_k \\ -s_k & c_k \end{pmatrix} \begin{pmatrix} \widehat{\widehat{\rho}}_k & 0 \\ \beta_{k+1} & \alpha_{k+1} \end{pmatrix} = \begin{pmatrix} \rho_k & \theta_{k+1} \\ 0 & \widehat{\widehat{\rho}}_{k+1} \end{pmatrix}, \tag{6}$$

where $\widehat{\widehat{\rho}}_1 \equiv \alpha_1$, and the scalers c_k and s_k satisfy the relation $c_k^2 + s_k^2 = 1$. The quantity $\widehat{\widehat{\rho}}_k$ is an intermediate scaler that is subsequently replaced by ρ_k. Now, if we define $f_k = \mathcal{R}_ky_k$, we have

$$\begin{aligned} \|A^TR_k\|_F &= \left\|\overline{\beta}_1e_1 - \left(\begin{pmatrix} \mathcal{R}_k^T\mathcal{R}_k \\ \overline{\beta}_{k+1}e_k^T \end{pmatrix} y_k\right)\right\|_2 \\ &= \left\|\overline{\beta}_1e_1 - \left(\begin{pmatrix} \mathcal{R}_k^T \\ \overline{\beta}_{k+1}e_k^T\mathcal{R}_k^{-1} \end{pmatrix} f_k\right)\right\|_2 \\ &= \left\|\overline{\beta}_1e_1 - \left(\begin{pmatrix} \mathcal{R}_k^T \\ \alpha_{k+1}\beta_{k+1}\rho_k^{-1}e_k^T \end{pmatrix} f_k\right)\right\|_2. \end{aligned}$$

On the other hand, from (6), we have $\theta_{k+1} = s_k\alpha_{k+1}$ and

$$\begin{pmatrix} \widehat{\widehat{\rho}}_k & 0 \\ \beta_{k+1} & \alpha_{k+1} \end{pmatrix} = \begin{pmatrix} c_k & -s_k \\ s_k & c_k \end{pmatrix} \begin{pmatrix} \rho_k & \theta_{k+1} \\ 0 & \widehat{\widehat{\rho}}_{k+1} \end{pmatrix}.$$

This implies that $\beta_{k+1} = s_k\rho_k$. So, $\theta_{k+1} = \alpha_{k+1}\beta_{k+1}\rho_k^{-1}$ and

$$\|A^T R_k\|_F = \left\| \overline{\beta}_1 e_1 - \begin{pmatrix} \mathcal{R}_k^T \\ \theta_{k+1} e_k^T \end{pmatrix} f_k \right\|_2 .$$

Then, we perform a second QR factorization

$$\overline{Q}_k \begin{pmatrix} \mathcal{R}_k^T & \overline{\beta}_1 e_1 \\ \theta_{k+1} e_k^T & 0 \end{pmatrix} = \begin{pmatrix} \overline{\mathcal{R}}_k & z_k \\ 0 & \overline{\zeta}_{k+1} \end{pmatrix}, \qquad \overline{\mathcal{R}}_k = \begin{pmatrix} \overline{\rho}_1 & \overline{\theta}_2 & & & \\ & \overline{\rho}_2 & \overline{\theta}_3 & & \\ & & \ddots & \ddots & \\ & & & \overline{\rho}_{k-1} & \overline{\theta}_k \\ & & & & \overline{\rho}_k \end{pmatrix}$$

where $\overline{\rho}_l$ and $\overline{\theta}_l$ are scalers. Similarly, this QR factorization can be done in progressive manner. At iteration k we eliminate θ_{k+1} by using the simple recurrence relation

$$\begin{pmatrix} \overline{c}_k & \overline{s}_k \\ -\overline{s}_k & \overline{c}_k \end{pmatrix} \begin{pmatrix} \widetilde{\overline{\rho}}_k & 0 & \overline{\zeta}_k \\ \theta_{k+1} & \rho_{k+1} & 0 \end{pmatrix} = \begin{pmatrix} \overline{\rho}_k & \overline{\theta}_{k+1} & \zeta_k \\ 0 & \widetilde{\overline{\rho}}_{k+1} & \overline{\zeta}_{k+1} \end{pmatrix},$$

where $\widetilde{\overline{\rho}}_1 \equiv \rho_1, \overline{\zeta}_1 \equiv \overline{\beta}_1$, and the scalers $\overline{c}_k$ and $\overline{s}_k$ satisfy the relation $\overline{c}_k^2 + \overline{s}_k^2 = 1$. The quantities $\widetilde{\overline{\rho}}_k$ and $\overline{\zeta}_k$ are intermediate scalers that are subsequently replaced by $\overline{\rho}_k$ and ζ_k.

Combining what we have with (5) gives:

$$\min_{y_k} \|A^T R_k\|_F = \min_{f_k} \left\| \begin{pmatrix} z_k \\ \overline{\zeta}_{k+1} \end{pmatrix} - \begin{pmatrix} \overline{\mathcal{R}}_k \\ 0 \end{pmatrix} f_k \right\|_2 .$$

With setting

$$f_k = \overline{\mathcal{R}}_k^{-1} z_k,$$

the approximate solution is given by

$$X_k = \mathcal{V}_k * \left((\overline{\mathcal{R}}_k \mathcal{R}_k)^{-1} z_k \right) = \left(\mathcal{V}_k * (\overline{\mathcal{R}}_k \mathcal{R}_k)^{-1} \right) z_k.$$

By defining

$$\mathcal{P}_k \equiv \mathcal{V}_k * (\overline{\mathcal{R}}_k \mathcal{R}_k)^{-1} \equiv \left(P_1, \quad P_2, \quad \ldots, \quad P_k \right),$$

we have

$$X_k = \mathcal{P}_k * z_k.$$

The $n \times s$ matrix P_k, the last block of $\mathcal{P}_k$, can be derived from the previous P_{k-2}, P_{k-1} and V_k, via the relation

$$P_k = \left(V_k - \overline{\theta}_{k-1}\theta_k P_{k-2} - (\overline{\rho}_{k-1}\theta_k + \overline{\theta}_k \rho_k) P_{k-1} \right) (\overline{\rho}_k \rho_k)^{-1}.$$

In addition, we note that

$$z_k = \begin{pmatrix} z_{k-1} \\ \zeta_k \end{pmatrix},$$

in which $\zeta_k = \overline{c}_k \overline{\zeta}_k$. As a result, X_k can be updated at each step as

$$X_k = X_{k-1} + \zeta_k P_k, \tag{7}$$

where P_k is defined above. Moreover, the Frobenius norm of residual matrix of normal equation, $\|A^T R_k\|_F$, is the absolute value of the last element of the right hand side $\begin{pmatrix} z_k \\ \overline{\zeta}_{k+1} \end{pmatrix}$, i.e.,

$$\|A^T R_k\|_F = |\overline{\zeta}_{k+1}|. \tag{8}$$

From (7), the matrix residual R_k is given by

$$R_k = R_{k-1} - \zeta_k A P_k,$$

Now, we show that the $\|R_k\|_F$ can be estimated by a simple formula. We transform $\overline{\mathcal{R}}_k^T$ to upper-bidiagonal form using a third QR factorization:

$$\widetilde{Q}_k\overline{\mathcal{R}}_k^T = \widetilde{R}_k = \begin{pmatrix} \widetilde{\rho}_1 & \widetilde{\theta}_2 & & & \\ & \widetilde{\rho}_2 & \widetilde{\theta}_3 & & \\ & & \ddots & \ddots & \\ & & & \widetilde{\rho}_{k-1} & \widetilde{\theta}_k \\ & & & & \widetilde{\rho}_k \end{pmatrix},$$

where $\widetilde{\rho}_l$ and $\widetilde{\theta}_l$ are scalers. The above QR factorization is determined by constructing the kth plane rotation $\widetilde{Q}_{k,k+1}$ to operate on rows k and $k+1$ of the transformed $\overline{\mathcal{R}}_k^T$ to annihilate $\overline{\theta}_{k+1}$. This gives the following simple recurrence relation:

$$\begin{pmatrix} \widetilde{c}_k & \widetilde{s}_k \\ -\widetilde{s}_k & \widetilde{c}_k \end{pmatrix}\begin{pmatrix} \dot{\rho}_k & 0 \\ \overline{\theta}_{k+1} & \overline{\rho}_{k+1} \end{pmatrix} = \begin{pmatrix} \widetilde{\rho}_k & \widetilde{\theta}_{k+1} \\ 0 & \dot{\rho}_{k+1} \end{pmatrix},$$

where $\dot{\rho}_1 \equiv \overline{\rho}_1$, and the scalers $\widetilde{c}_k$ and $\widetilde{s}_k$ are the nontrivial elements of $\widetilde{Q}_{k,k+1}$. The quantity $\dot{\rho}_k$ is an intermediate scaler that is subsequently replaced by $\widetilde{\rho}_k$. Now let

$$\widetilde{f}_k = \widetilde{Q}_k f_k, \qquad \widetilde{b}_k = \begin{pmatrix} \widetilde{Q}_k & \\ & 1 \end{pmatrix} Q_k e_1 \beta_1.$$

Then

$$\begin{aligned} R_k &= B - AX_k \\ &= \beta_1 U_1 - A\mathcal{V}_k * y_k \\ &= \beta_1 U_1 - \mathcal{U}_{k+1} * T_k * y_k \\ &= \mathcal{U}_{k+1} * (\beta_1 e_1 - T_k y_k) \\ &= \mathcal{U}_{k+1} * \left(\beta_1 e_1 - Q_k^T \begin{pmatrix} \mathcal{R}_k \\ 0 \end{pmatrix} y_k\right) \\ &= \mathcal{U}_{k+1} * \left(\beta_1 e_1 - Q_k^T \begin{pmatrix} f_k \\ 0 \end{pmatrix}\right) \\ &= \mathcal{U}_{k+1} * \left(Q_k^T \begin{pmatrix} \widetilde{Q}_k^T & \\ & 1 \end{pmatrix} \widetilde{b}_k - Q_k^T \begin{pmatrix} \widetilde{Q}_k^T \widetilde{f}_k \\ 0 \end{pmatrix}\right) \\ &= \mathcal{U}_{k+1} * \left(Q_k^T \begin{pmatrix} \widetilde{Q}_k^T & \\ & 1 \end{pmatrix} \left(\widetilde{b}_k - \begin{pmatrix} \widetilde{f}_k \\ 0 \end{pmatrix}\right)\right). \end{aligned}$$

According to Proposition 2, we have

$$\|R_k\|_F = \left\| \widetilde{b}_k - \begin{pmatrix} \widetilde{f}_k \\ 0 \end{pmatrix} \right\|_2. \tag{9}$$

The vectors $\widetilde{b}_k$ and $\widetilde{f}_k$ can be written in the form

$$\widetilde{b}_k = \begin{pmatrix} \widetilde{\beta}_1 & \cdots & \widetilde{\beta}_{k-1} & \dot{\beta}_k & \ddot{\beta}_{k+1} \end{pmatrix}^T, \qquad \widetilde{f}_k = \begin{pmatrix} \widetilde{\tau}_1 & \cdots & \widetilde{\tau}_{k-1} & \dot{\tau}_k \end{pmatrix}^T. \tag{10}$$

The vector $\widetilde{f}_k$ can be computed by forward substitution from $\widetilde{\mathcal{R}}_k^T \widetilde{f}_k = z_k$. The following lemma shows that we can estimate $\|R_k\|_F$ from just two elements of $\widetilde{b}_k$ and the last element of $\widetilde{f}_k$.

Lemma 3. *In (9) and (10), $\widetilde{\beta}_i = \widetilde{\tau}_i$ for $i = 1, 2, \ldots, k-1$.*

The proof of this lemma is similar to that given in [40] for the LSMR algorithm. The main steps of Gl-LSMR algorithm can be summarized as follows.

Algorithm 1 Gl-LSMR algorithm

Set $X_0 = 0_{n\times s}$
$\beta_1 = \|B\|_F,\ U_1 = B/\beta_1,\ \alpha_1 = \|A^T U_1\|_F, V_1 = A^T U_1/\alpha_1, \overline{\beta}_1 = \alpha_1\beta_1$
Set $\overline{\zeta}_1 = \overline{\beta}_1,\ \widehat{\widehat{\rho}}_1 = \alpha_1$
Set $\overline{c}_0 = 1,\ \overline{s}_0 = 0,\ P_{-1} = P_0 = 0_{n\times s},\ \overline{\theta}_0 = \theta_1 = 0,\ \overline{\rho}_0 = 1$
For $i = 1, 2, \ldots$ until convergence Do:
$\mathcal{W}_i = AV_i - \alpha_i U_i$
$\beta_{i+1} = \|\mathcal{W}_i\|_F$
$U_{i+1} = \mathcal{W}_i/\beta_{i+1}$
$w_i = A^T U_{i+1} - \beta_{i+1} V_i$
$\alpha_{i+1} = \|w_i\|_F$
$V_{i+1} = w_i/\alpha_{i+1}$
$\rho_i = \left(\widehat{\widehat{\rho}}_i^2 + \beta_{i+1}^2\right)^{\frac{1}{2}}$
$c_i = \widehat{\widehat{\rho}}_i/\rho_i$
$s_i = \beta_{i+1}/\rho_i$
$\theta_{i+1} = s_i\alpha_{i+1}$
$\widehat{\widehat{\rho}}_{i+1} = c_i\alpha_{i+1}$
$\widetilde{\overline{\rho}}_i = \overline{c}_{i-1}\rho_i$
$\overline{\rho}_i = \left(\widetilde{\overline{\rho}}_i^2 + \theta_{i+1}^2\right)^{\frac{1}{2}}$
$\overline{c}_i = \widetilde{\overline{\rho}}_i/\overline{\rho}_i$
$\overline{s}_i = \theta_{i+1}/\overline{\rho}_i$
$\overline{\theta}_i = \overline{s}_{i-1}\rho_i$
$\zeta_i = \overline{c}_i\overline{\zeta}_i$
$\overline{\zeta}_{i+1} = -\overline{s}_i\overline{\zeta}_i$
$P_i = \left(V_i - \overline{\theta}_{i-1}\theta_i P_{i-2} - (\overline{\rho}_{i-1}\theta_i + \overline{\theta}_i\rho_i)P_{i-1}\right)/\rho_i\overline{\rho}_i$
$X_i = X_{i-1} + \zeta_i P_i$
If $|\overline{\zeta}_{i+1}|$ is small enough then stop
End Do.

As we observe, at each iteration, the approximate solution and residual matrix can be cheaply updated and the Frobenius norm of residual matrix of normal equation can also be obtained with virtually no additional arithmetic operations. From (8), we have $\|A^T R_k\|_F = |\overline{\zeta}_{k+1}|$ and by using Lemma 3, we can show that the Frobenius norm of residual matrix, $\|R_k\|_F$, can be computed by Algorithm 2 which is similar to the pseudo-code given in [40] for computing $\|r_k\|_2$.

2.3. Complexity

The storage requirements and the computational costs for Gl-LSMR and Gl-LSQR algorithms are summarized in Table 1. For each method, the cost represents the computational costs required per iteration. For example, Gl-LSMR method requires $2s$ matrix-by-vector products at each iteration. For the storage, the $n \times s$ matrix AV represents working storage to hold products of the form AV and $A^T U$. As we observe, the work and storage requirements of Gl-LSMR are not significantly higher than for Gl-LSQR method.

The following propositions establish some results on the global LSMR algorithm.

Proposition 4. *Assume that $A^T A = U\Lambda U^T$, where U is an orthonormal matrix, $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_k)$ and λ_i's are eigenvalues of $A^T A$. Consider the eigen-decomposition of the $A^T R_0 = U\beta$ where β is an $n \times s$ matrix. Then, at step k of the Gl-LSMR, we have*

$$\|A^T R_k\|_F \le \overline{\beta}_1 \min_{P\in\mathcal{P}_k, P(0)=1} \left(\max_{i=1,\ldots,n} |P(\lambda_i)|\right),$$

where $\mathcal{P}_k$ is the set of polynomials of degree less than or equal to k.

The proof of this proposition is similar to that of Theorem 7 in [37].

Proposition 5. *In the Gl-LSMR method, we have the following properties:*

Table 1
Storage and computational requirements for Gl-LSMR and Gl-LSQR.

Cost	Gl-LSMR	Gl-LSQR
$M \times v$	$2s$	$2s$
n-vector DOT	$2s$	$2s$
Mult. of n-vector and scaler	$8s$	$6s$
Scaler costs	18	10
Storage		
$n \times s$-matrix	AV, U, V, P_0, P_1, X	AV, U, V, W, X

(i) *If* $\alpha_{k+1} = 0$, *then* $\|A^T R_k\|_F = 0$.
(ii) *If* $\beta_{k+1} = 0$, *then* $\|R_k\|_F = 0$.
(iii) *For singular systems* $AX = B$, *Gl-LSMR gives the minimum-F-norm LS solution, i.e., it solves the optimization problem* $\min \|X\|_F$ *such that* $A^T AX = A^T B$.

Algorithm 2 Pseudo-code for computing $\|R_i\|_F$

Set $\ddot{\beta}_1 = \beta_1,\ \dot{\beta}_0 = 0,\ \dot{\rho}_0 = 1,\ \widetilde{\tau}_{-1} = 0,\ \widetilde{\theta}_0 = 0,\ \zeta_0 = 0$
For the ith iteration of the algorithm 1, perform the following steps:

$\widehat{\beta}_i = c_i \ddot{\beta}_i$
$\ddot{\beta}_{i+1} = -s_i \ddot{\beta}_i$
$\widetilde{\rho}_{i-1} = (\dot{\rho}_{i-1}^2 + \overline{\theta}_i^2)^{\frac{1}{2}}$
$\widetilde{c}_{i-1} = \dot{\rho}_{i-1}/\widetilde{\rho}_{i-1}$
$\widetilde{s}_{i-1} = \overline{\theta}_i/\widetilde{\rho}_{i-1}$
$\theta_i = \widetilde{s}_{i-1}\overline{\rho}_i$
$\dot{\rho}_i = \widetilde{c}_{i-1}\overline{\rho}_i$
$\widetilde{\beta}_{i-1} = \widetilde{c}_{i-1}\dot{\beta}_{i-1} + \widetilde{s}_{i-1}\widehat{\beta}_i$
$\dot{\beta}_i = -\widetilde{s}_{i-1}\dot{\beta}_{i-1} + \widetilde{c}_{i-1}\widehat{\beta}_i$
$\widetilde{\tau}_{i-1} = (\zeta_{i-1} - \theta_{i-1}\widetilde{\tau}_{i-2})/\widetilde{\rho}_{i-1}$
$\dot{\tau}_i = (\zeta_i - \widetilde{\theta}_i\widetilde{\tau}_{i-1})/\dot{\rho}_i$
$\gamma = (\dot{\beta}_i - \dot{\tau}_i)^2 + \ddot{\beta}_{i+1}^2$
$\|R_i\|_F = \sqrt{\gamma}$

The proof of this proposition is similar to that stated in [40].

3. Numerical examples

To compare the behavior of the proposed method discussed in the previous section with the LSMR method [40] and the global LSQR iterative method (Gl-LSQR) [33], we present in this section numerical results for four examples. All the numerical computations are performed in MATLAB 7. Moreover, the initial guess is selected $X_0 = 0$ with suitable size.

Example 1. In this example, we compare the Gl-LSMR algorithm for solving problem (1) with the standard LSMR algorithm applied to one right-hand side. The coefficient matrices are taken from the University of Florida Sparse Matrix Collection (Davis [45]). These matrices with their properties are shown in Table 2. Diagonal scaling was applied to the columns of $\begin{pmatrix} A & B \end{pmatrix}$ to give a scaled problem $AX = B$ in which the columns of $\begin{pmatrix} A & B \end{pmatrix}$ have unit 2-norm. By scaling, the number of iterations of Gl-LSMR for convergence reduced satisfactorily. We use the right-hand side: $B = \text{rand}(n, s)$, where function rand creates an $n \times s$ random matrix with coefficients uniformly distributed in [0, 1]. The stopping criteria is set to $\|A^T R_k\|_F \leq 10^{-10}$.

In Table 3, we give the ratio $t(s)/t(1)$, for $s = 5$, 10, 15, and 20, where $t(s)$ is the CPU time for Gl-LSMR algorithm and $t(1)$ is the CPU time obtained when applying LSMR for one right-hand side linear system. Note that the time obtained by LSMR for one right-hand side depends on which right-hand was used. So, $t(1)$ is the average of the times needed for the s right-hand sides using LSMR. We note that Gl-LSMR is effective if the indicator $t(s)/t(1)$ is less than s. As shown in Table 3 the Gl-LSMR algorithm is effective and less expensive than the LSMR algorithm applied to each right-hand side.

Example 2. In this example, we display the convergence history of Gl-LSMR and Gl-LSQR in Fig. 1 for the systems corresponding to the matrices Hamm/hcircuit and HB/sherman3. As Example 1, $B = rand(n, s)$ and the tests stopped as soon as $\|A^T R_k\|_F \leq 10^{-10}$. Fig. 1(left) shows both solvers reduce $\|R_k\|_F$ monotonically. Fig. 1 (right) shows $\|A^T R_k\|_F$ for each solver. The plateaus for Gl-LSMR correspond to Gl-LSQR gaining ground with $\|R_k\|_F$, but falling significantly backward

Table 2
Test problems information.

Matrix	Property				
	Order	sym	nnz	id	Discipline
Hamm/add32	4 960	No	19 848	540	Electronic circuit design
HB/fs6801	680	No	2 184	149	Chemical kinetics
HB/gre115	115	No	421	161	Simulation studies in computer systems
Hamm/hcircuit	105 676	No	513 072	542	Simulation studies in circuit parasitics
Bai/pde2961	2 961	No	14 585	324	Partial differential equations
HB/orsirr1	1 030	No	6 858	225	Oil reservoir simulation
Bai/rdb3200l	3 200	No	18 880	1633	Chemical engineering
HB/sherman3	5 005	No	20 033	244	Oil reservoir simulation
HB/sherman4	1 104	No	3 786	245	Oil reservoir modeling
IBM EDA/trans5	116 835	No	749 800	1324	Subsequent circuit simulation problem

Table 3
Effectiveness of Gl-LSMR algorithm measured $t(s)/t(1)$.

Matrix	S			
	5	10	15	20
Hamm/add32	1.92	3.54	5.94	7.98
HB/fs6801	2.63	4.11	5.06	5.95
HB/gre115	1.60	3.01	4.04	4.26
Hamm/hcircuit	1.92	5.68	8.17	10.26
Bai/pde2961	1.89	3.16	4.47	5.50
HB/orsirr1	2.57	3.84	5.29	6.67
Bai/rdb3200l	2.40	3.72	5.90	6.83
HB/sherman3	2.44	3.96	6.75	9.06
HB/sherman4	1.99	4.50	5.24	8.34
IBM EDA/trans5	1.71	4.50	6.60	9.26

Table 4
CPU time for the convergence of the Gl-LSMR and the Gl-LSQR.

Matrix	Method	
	Gl-LSMR	Gl-LSQR
Hamm/hcircuit	8.84e+03 (6500)	1.33e+04 (7250)
HB/sherman3	45.34 (9375)	72.56 (9750)

by the $\|A^T R_k\|_F$ measure. As can be seen, Gl-LSQR decreases $\|A^T R_k\|_F$ oscillatory, whereas Gl-LSMR decreases $\|A^T R_k\|_F$ monotonically. In addition, we give CPU times and the number of iterations (in parentheses) for convergence the Gl-LSMR and the Gl-LSQR methods in Table 4. As can be observed, the Gl-LSMR is terminated sooner than the Gl-LSQR.

Example 3. In this example, the matrix A represents the 5-point discretization of the operator

$$L(u) := -u_{xx} - u_{yy} + \delta u_x$$

on the unit square $[0, 1] \times [0, 1]$ with homogeneous Dirichlet boundary conditions. The mesh size in x-direction and y-direction is $h = \frac{1}{61}$ which yields a matrix of dimension $n = 3600$. We consider three matrices for the right-hand sides: $B = B_1 = I_{n,s}$, $B = B_2 = rand(n, s)$, and $B = B_3$, where B_3 is the $n \times s$ matrix whose ith column has all components equal to one except the ith component which is zero. The tests stopped as soon as $\frac{\|R_k\|_F}{\|R_0\|_F} \leq 10^{-10}$. Fig. 2(left) shows the behavior of the Frobenius residual norms in the logarithmic scale, versus the number of iteration for Gl-LSMR and Gl-LSQR. The figures from top to bottom are related to running with the right-hand side matrix B_1, B_2 and B_3, respectively. As can be observed the Gl-LSMR method has similar behavior to the Gl-LSQR method.

Example 4. In order to illustrate the numerical performance of the Gl-LSMR and the Gl-LSQR, we consider the convection diffusion equation with the Dirichlet boundary conditions

$$\begin{cases} -\Delta u + 2p_1 u_x + 2p_2 u_y - 2p_3 u = F \quad \text{on } \Omega, \\ u = 0 \quad \text{on } \partial\Omega. \end{cases}$$

Here Ω is the unit square $(0, 1) \times (0, 1)$. The parameters p_1, p_2 and p_3 are 25, 50, 50, respectively. The operator Laplacian is discretized using standard five-point stencil on Ω, and the first-order derivatives is discretized by centered finite differences with mesh size $h_1 = 1/(n + 1)$ in the "x" direction and $h_2 = 1/(s + 1)$ in the "y" direction. This yields a linear system of

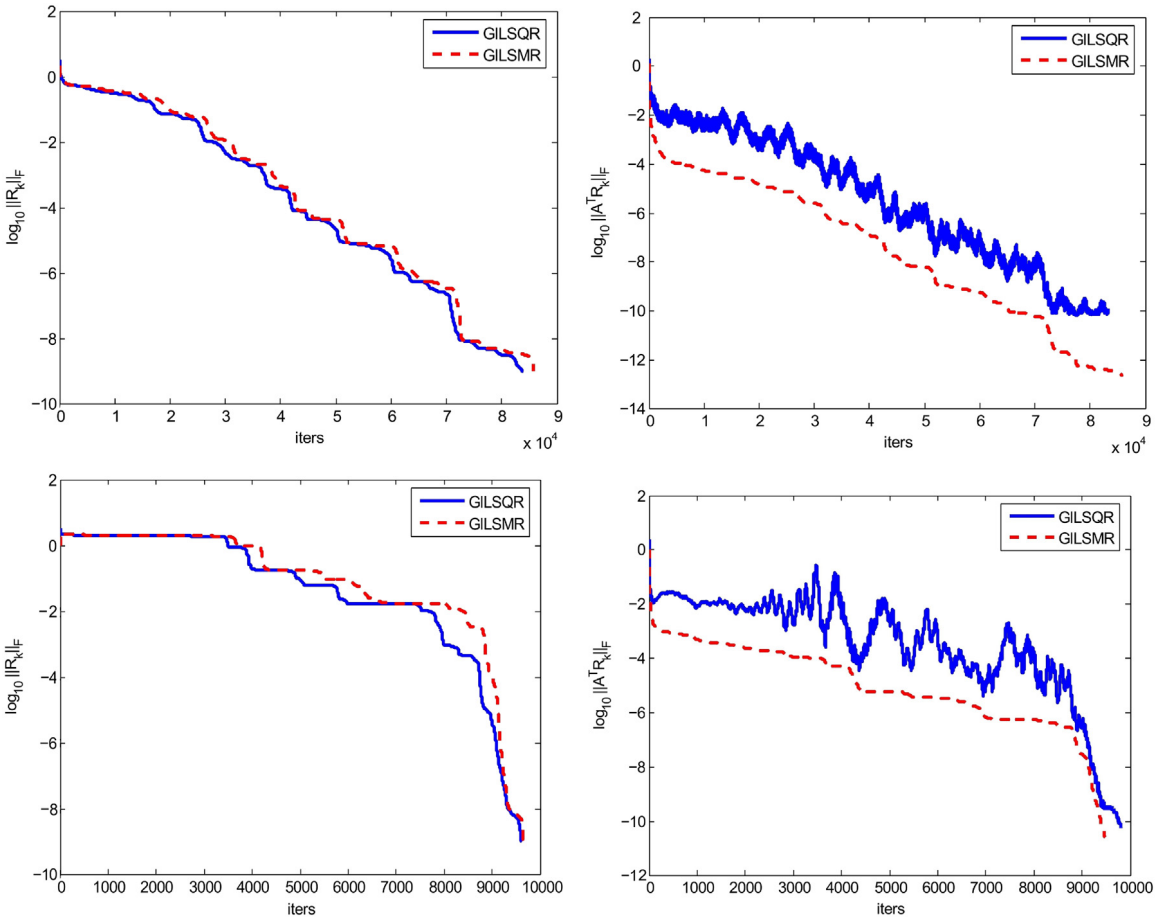

Fig. 1. Gl-LSMR and Gl-LSQR solving two square nonsingular systems $AX = B$ with $s = 10$: problems Hamm/hcircuit (top) and HB/sherman3 (bottom). Left: $\log_{10} \|R_k\|_F$ for both solvers, with prolonged plateaus for Gl-LSMR. Right: $\log_{10} \|A^T R_k\|_F$ (preferable for Gl-LSMR).

algebraic equations that can be written as a Sylvester matrix equation

$$AX + XC = B, \tag{11}$$

where tridiagonal matrices A and C given by

$$A_{n\times n} = \frac{-1}{h_1^2} tridiag\big(-1 - p_1h_1, 2 - p_3h_1^2, -1 + p_1h_1\big), \qquad C_{s\times s} = \frac{1}{h_2^2} tridiag\big(-1 + p_2h_2, 2 - p_3h_2^2, -1 - p_2h_2\big)$$

and $B = \big(F(x_i, y_j)\big)_{n\times s}$. The function F is chosen such that the exact solution is $u(x, y) = \sin(\pi xy)$ on the domain Ω. As Example 3, the stopping criteria is set $\frac{\|R_k\|_F}{\|R_0\|_F} \leq 10^{-10}$. It is easy to prove that the matrix equation (11) is equivalent to the $ns \times ns$ linear system

$$\widetilde{\mathcal{A}}\widetilde{x} = \widetilde{b}, \tag{12}$$

where $\widetilde{\mathcal{A}} = I_s \otimes A + C^T \otimes I_n$, $\widetilde{b} = vec(B)$ and $\widetilde{x} = vec(X)$. The matrix equation (12) can be solved by the Gl-LSMR(or any other global method) by replacing in Gl-LSMR the matrix product AX by $AX + XC$. Fig. 2 (right-top to bottom) shows the convergence history of the Gl-LSMR and the Gl-LSQR for different performances of these methods with $n = 10$, 100 and 200, respectively. As can be seen, both methods have almost the same operation and by increasing n the number of iterations is increased.

4. Conclusion

In this paper, we have presented a global version of LSMR algorithm for solving general linear systems with multiple right-hand sides. As can be observed, the new method constructs a simple recurrence formula for generating the sequence of approximate solutions $\{X_k\}$. In addition, $\|R_k\|_F$, and $\|A^T R_k\|_F$ are cheaply computable. Experimental results confirm that

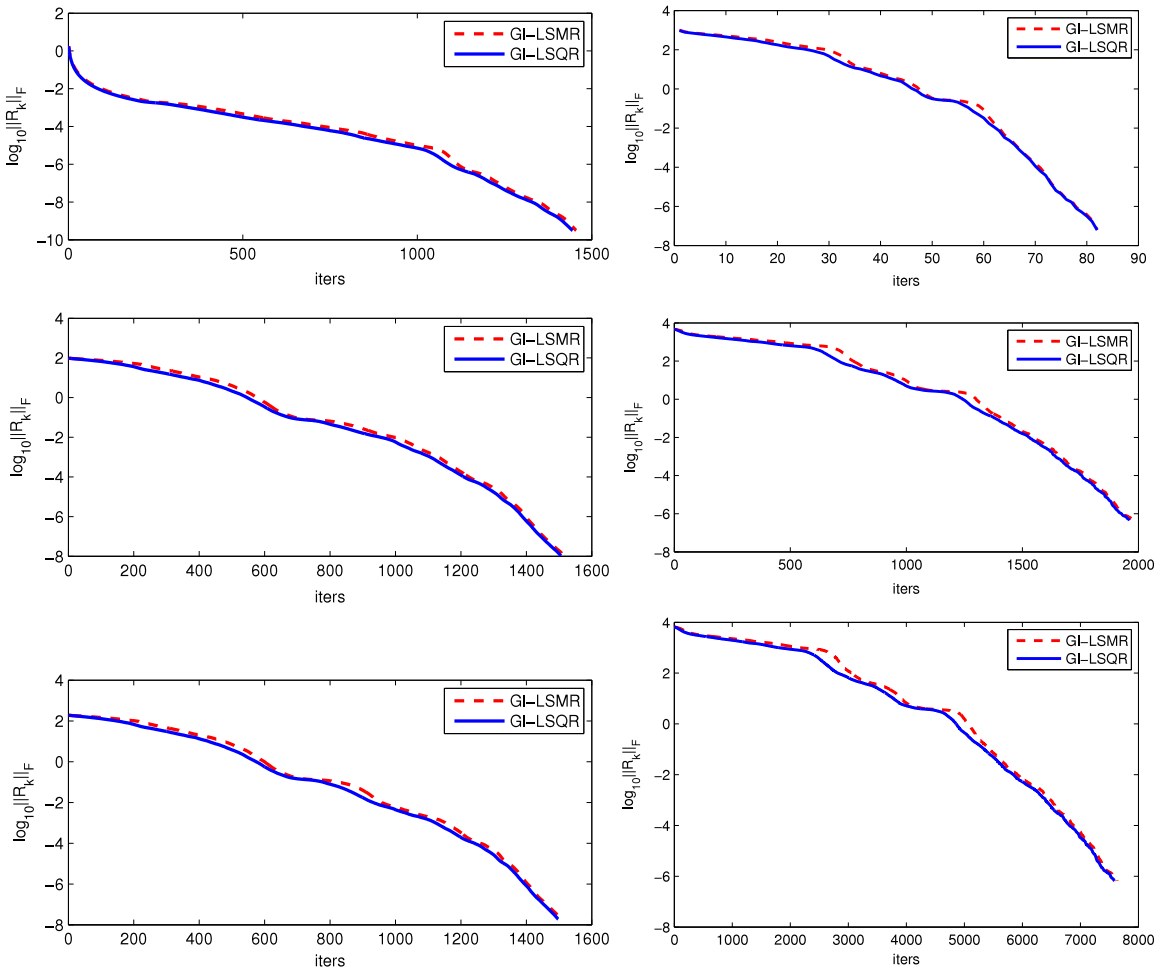

Fig. 2. Convergence history of the Gl-LSMR and Gl-LSQR with $s = 10$: Example 3 (left) and Example 4 (right).

the proposed method is effective and less expensive than the LSMR algorithm applied to each right-hand side. In other hand, the numerical examples show that the developed method has advantages over the Gl-LSQR algorithm. Finally, to reinforcing the performance of the proposed method, a good preconditioner for the symmetric positive definite matrix A^TA can be used.

Acknowledgments

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

References

[1] D. ÓLeary, The block conjugate gradient algorithm and related methods, Linear Algebra Appl. 29 (1980) 293–322.
[2] A. Nikishin, 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. 16 (1995) 1135–1153.
[3] G. Haase, S. Reitzinger, Cache issues of algebraic multigrid methods for linear systems with multiple right-hand sides, SIAM J. Sci. Comput. 27 (2005) 1–18.
[4] B. Vital, Etude de quelques méthodes de résolution de problémes linéaires de grande taille sur multiprocesseur (Ph.D. thesis), Université de Rennes, 1990.
[5] V. Simoncini, E. Gallopoulos, Convergence properties of block GMRES and matrix polynomials, Linear Algebra Appl. 247 (1996) 97–119.
[6] H.-L. Liu, B.-J. Zhong, Simpler block GMRES for nonsymmetric systems with multiple right-hand sides, Electron. Trans. Numer. Anal. 30 (2008) 1–9.
[7] D. Darnell, R.B. Morgan, W. Wilcox, Deflated GMRES for systems with multiple shifts and multiple right-hand sides, Linear Algebra Appl. 429 (2008) 2415–2434.
[8] G. Gu, Z. Cao, A block GMRES method augmented with eigenvectors, Appl. Math. Comput. 121 (2001) 271–289.
[9] M.H. Gutknecht, Block Krylov space methods for linear systems with multiple right-hand sides: an introduction, in: A.H. Siddiqi, I.S. Duff, O. Christensen (Eds.), Modern Mathematical Models, Methods and Algorithms for Real Word Systems, Anamaya Publishers, New Delhi, India, 2007, pp. 420–447.
[10] R.B. Morgan, Restarted block- GMRES with deflation of eignvalues, Appl. Numer. Math. 54 (2005) 222–236.
[11] V. Simoncini, E. Gallopoulos, An iterative method for nonsymmetric systems with multiple right-hand sides, SIAM J. Sci. Comput. 16 (1995) 917–933.
[12] M. Robbé, M. Sadkane, Exact and inexact breakdowns in the block GMRES method, Linear Algebra Appl. 419 (2006) 265–285.

[13] H. Dai, Two algorithms for symmetric linear systems with multiple right-hand sides, Numer. Math. J. Chin. Univ. (Engl. Ser.) 9 (2000) 91–110.
[14] V. Simoncini, A stabilized QMR version of block BICG, SIAM J. Matrix Anal. Appl. 18 (1997) 419–434.
[15] R. Freund, M. Malhotra, A Block-QMR algorithm for non-hermitian linear systems with multiple right-hand sides, Linear Algebra Appl. 254 (1997) 119–157.
[16] 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.
[17] 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.
[18] 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.
[19] F. Toutounian, M. Mojarrab, The block LSMR algorithm for solving linear systems with multiple right-hand sides, Iranian J. Numer. Anal. Optim. 5 (2015) 11–28.
[20] F. Toutounian, M. Mojarrab, The block LSMR method: a novel efficient algorithm for solving non-symmetric linear systems with multiple right-hand sides, Iranian J. Sci. Technol. 39 (2015) 69–78.
[21] T.F. Chan, W.L. Wan, Analysis of projection methods for solving linear systems with multiple right-hand sides, SIAM J. Sci. Comput. 18 (1997) 1698–1721.
[22] Y. Saad, On the Lanczos method for solving symmetric linear systems with several right-hand sides, Math. Comp. 48 (1987) 651–662.
[23] C. Smith, A. Peterson, R. Mittra, A conjugate gradient algorithm for treatment of multiple incident electromagnetic fields, IEEE Trans. Antennas and Propagation 37 (1989) 1490–1493.
[24] P. Joly, Résolution de Systems Linéaires Avec Plusieurs Second Members par la Methode du Gradient Conjugue, Tech. Rep. R-91012, Publications du Laboratire d'Analyse Numerique, Université Pierre et Marie Curie, Paris, 1991.
[25] H. Van Der Vorst, An iterative solution method for solving $f(A)x = b$, using Krylov subspace information obtained for the symmetric positive definite matrix A, J. Comput. Appl. 18 (1987) 249–263.
[26] A.M. Abdel-Rehim, R.B. Morgan, W. Wilcox, Improved seed methods for symmetric positive definite linear equations with multiple right-hand sides, 2008, Arxiv Preprint arXiv:0810.0330.
[27] K. Jbilou, H. Sadok, Global Lanczos-based methods with applications, Technical Report LMA 42, Université du Littoral, Calais, France, 1997.
[28] K. Jbilou, A. Messaoudi, H. Sadok, Global FOM and GMRES algorithms for matrix equations, Appl. Numer. Math. 31 (1999) 49–63.
[29] M. Heyouni, The global Hessenberg and global CMRH methods for linear systems with multiple right-hand sides, Numer. Algorithms 26 (2001) 317–332.
[30] 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.
[31] Y. Lin, Implicitly restarted global FOM and GMRES for nonsymmetric matrix equations and Sylvester equations, Appl. Math. Comput. 167 (2005) 1004–1025.
[32] M. Heyouni, A. Essai, Matrix Krylov subspace methods for linear systems with multiple right-hand sides, Numer. Algorithms 40 (2005) 137–156.
[33] 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.
[34] D.K. Salkuyeh, CG-type algorithms to solve symmetric matrix equations, Appl. Math. Comput. 172 (2006) 985–999.
[35] 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.
[36] J. Zhang, H. Dai, Global CGS algorithm for linear systems with multiple right-hand sides, Numer. Math. J. Chinese Univ. 30 (2008) 390–399. (in Chinese).
[37] 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.
[38] J. Zhang, H. Dai, J. Zhao, Generalized global conjugate gradient squared algorithm, Appl. Math. Comput. 216 (2010) 3694–3706.
[39] 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.
[40] D. Chin-Lung Fong, M. Saunders, LSMR: An iterative algorithm For Sparse Least-Squares Problems, SIAM J. Sci. Comput. 33 (2011) 2950–2971.
[41] G.H. Golub, W. Kahan, Algorithm LSQR is based on the Lanczos process and bidiagonalization procedure, SIAM J. Numer. Anal. 2 (1965) 205–224.
[42] C.C. Paige, M.A. Saunders, LSQR: an algorithm for sparse linear equations and sparse least squares, ACM Trans. Math. Software 8 (1) (1982) 43–71.
[43] C.C. Paige, M.A. Saunders, Solution of sparse indefinite systems of linear equations, SIAM J. Numer. Anal. 12 (1975) 617–629.
[44] G.H. Golub, C.F. Van Loan, Matrix Computations, Johns Hopkins University Press, Baltimore, MD, 1983.
[45] T.A. Davis, University of Florida Sparse Matrix Collection, http://www.cise.ufl.edu/research/sparse/matrices.