The block least squares method for solving nonsymmetric linear systems with multiple right-hand sides

S. Karimi *, F. Toutounian

Department of Mathematics, Ferdowsi University of Mashhad, P.O. Box 1159-91775, Mashhad, Iran

Abstract

In this paper, we present the block least squares method for solving nonsymmetric linear systems with multiple right-hand sides. This method is based on the block bidiagonalization. We first derive two algorithms by using two different convergence criteria. The first one is based on independently minimizing the 2-norm of each column of the residual matrix and the second approach is based on minimizing the Frobenius norm of residual matrix. We then give some properties of these new algorithms. Finally, some numerical experiments on test matrices from Harwell–Boeing collection are presented to show the efficiency of the new method.

© 2005 Elsevier Inc. All rights reserved.

Keywords: LSQR method; Bidiagonalization; Block methods; Iterative methods; Multiple right-hand sides

1. Introduction

Many applications require the solution of several sparse systems of equations

$$Ax^{(i)} = b^{(i)}, \quad i = 1, 2, \ldots, s$$

with the same coefficient matrix and different right-hand sides. When all the $b^{(i)}$'s are available simultaneously, Eq. (1) can be written as

$$AX = B,$$

where $A$ is an $n \times n$ nonsingular and nonsymmetric real matrix, $B$ and $X$ are $n \times s$ rectangular matrices whose columns are $b^{(1)}, b^{(2)}, \ldots, b^{(s)}$ and $x^{(1)}, x^{(2)}, \ldots, x^{(s)}$, respectively. In practice, $s$ is of moderate size $s \ll n$. Instead of applying an iterative method to each linear system, it is more efficient to use a method for all the systems simultaneously. In the last years, generalizations of the classical Krylov subspace methods have been developed. The first class of these methods contains the block solvers such as the BL-BCG algorithm [2,10], the
The second class contains the global GMRES [7] algorithm and global Lanczos-based methods [8]. A third class of methods use a single linear system named the seed system and then consider the corresponding Krylov subspace. The initial residuals of the other linear systems are projected onto this Krylov subspace. The process is repeated until convergence; see [12,1,9,13,14].

In the present paper, we give a new method for solving the problem (2). Our iterative method will be defined as block least squares (LSQR) method [11]. Algorithm LSQR is based on the bidiagonalization procedure of Golub and Kahan [5]. It generates a sequence of approximations \( \{x_k\} \) such that the residual norm \( \|r_k\|_2 \) decreases monotonically, where \( r_k = b - Ax_k \). Analytically, the sequence \( \{x_k\} \) is identical to the sequence generated by the standard CG algorithm and by several other published algorithms.

We define the block bidiagonalization procedure. This procedure generates two sets of the \( n \times s \) block vectors, \( V_1, V_2, \ldots \) and \( U_1, U_2, \ldots \) such that \( V_i^T V_j = 0_s \), \( U_i^T U_j = 0_s \), \( i \neq j \), and \( V_i^T V_i = I_s \), \( U_i^T U_i = I_s \). We derive two simple recurrence formulas for generating the sequences of approximations \( \{X_k\} \) such that the \( \max_j \|\text{col}(R_k)\|_2 \), where \( \text{col}(R_k) \) represents the \( j \)th column of \( R_k \), or the Frobenius norm of \( R_k \) decreases monotonically, where \( R_k = B - AX_k \).

Throughout this paper, we use the following notations. For two \( n \times s \) matrices \( X \) and \( Y \), we define the following inner product: \( \langle X, Y \rangle_F = \text{tr}(X^T Y) \), where \( \text{tr}(Z) \) denotes the trace of the square matrix \( Z \). The associated norm is the Frobenius norm denoted by \( ||\cdot||_F \). 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 \( ||\cdot||_2 \). Finally, \( 0_s \) and \( I_s \) will denote the zero and the identity matrices in \( \mathbb{R}^{s \times s} \).

The outline of this paper is as follows. In Section 2 we give a quick overview of LSQR method and its properties. In Section 3 we present the block version of the LSQR algorithm and two different convergence criteria. In Section 4 some numerical experiments on test matrices from Harwell–Boeing collection are presented to show the efficiency of the method. Finally, we make some concluding remarks in Section 5.

2. The LSQR algorithm

In this section, we recall some fundamental properties of LSQR algorithm [11], which is an iterative method for solving real linear systems of the form

\[
Ax = b,
\]

where \( A \) is a nonsymmetric matrix of order \( n \) and \( x, b \in \mathbb{R}^n \).

LSQR algorithm uses an algorithm of Golub and Kahan [5], which states as procedure Bidiag 1 in [11], to reduce \( A \) to the lower diagonal form. The procedure Bidiag 1 can be described as follows.

Bidiag 1 (Starting vector \( b \); reduction to lower bidiagonal form)

\[
\begin{align*}
\beta_1 u_1 &= b, \\
\alpha_2 v_1 &= A^T u_1, \\
\beta_{i+1} u_{i+1} &= A v_i - \alpha_i u_i, \\
\alpha_{i+1} v_{i+1} &= A^T u_{i+1} - \beta_{i+1} v_i,
\end{align*}
\]

\( i = 1, 2, \ldots \)

The scalars \( \alpha_i \geq 0 \) and \( \beta_i \geq 0 \) are chosen so that \( ||u||_2 = ||v||_2 = 1 \). With the definitions

\[
U_k \equiv [u_1, u_2, \ldots, u_k], \quad V_k \equiv [v_1, v_2, \ldots, v_k], \quad B_k \equiv \begin{bmatrix}
\alpha_1 \\
\beta_2 & \alpha_2 \\
& \ddots \\
& & \ddots \\
& & & \ddots \\
& & & & \ddots \\
& & & & & \ddots \\
& & & & & & \beta_k & \alpha_k \\
& & & & & & & \beta_{k+1}
\end{bmatrix},
\]
the recurrence relations (3) may be rewritten as
\[ U_{k+1}(\beta_t e_t) = b, \]
\[ AV_k = U_{k+1}B_k, \]
\[ A^T U_{k+1} = V_k B_k^T + \alpha_{k+1} v_{k+1} e_{k+1}^T. \]

In exact arithmetic, we have \( U_{k+1}^T U_{k+1} = I \) and \( V_k^T V_k = I \), where \( I \) is the identity matrix. By using the procedure Block Bidiag 1 the LSQR method constructs an approximation solution of the form \( x_k = V_k y_k \) which solves the least-squares problem, \( \min ||b - A x||_2 \). The main steps of the LSQR algorithm can be summarized as follows.

**Algorithm 1 (LSQR algorithm)**

Set \( x_0 = 0 \)
Compute \( \beta_1 = ||b||_2, u_1 = b/\beta_1, x_1 = ||A^T u_1||_2, v_1 = A^T u_1/x_1 \)
Set \( w_1 = v_1, \phi_1 = \beta_1, \rho_1 = x_1 \)
For \( i = 1, 2, \ldots, \) until convergence Do:
\[ w = Av_i - \rho_i u_i \]
\[ \beta_{i+1} = ||w||_2 \]
\[ u_{i+1} = w/\beta_{i+1} \]
\[ z_i = A^T u_{i+1} - \beta_{i+1} v_i \]
\[ x_{i+1} = ||z_i||_2 \]
\[ v_{i+1} = z_i/x_{i+1} \]
\[ \rho_{i+1} = (\rho_i^2 + \beta_{i+1}^2)^{1/2} \]
\[ c_i = \rho_{i+1}/\rho_i \]
\[ s_i = \beta_{i+1}/\rho_i \]
\[ \theta_{i+1} = s_i \beta_{i+1} \]
\[ \rho_{i+1} = -c_i \theta_{i+1} \]
\[ \phi_i = c_i \phi_i \]
\[ \phi_{i+1} = s_i \phi_i \]
\[ x_i = x_{i-1} + (\phi_i/\rho_i) w_i \]
\[ w_{i+1} = v_{i+1} - (\theta_{i+1}/\rho_i) w_i \]
(EndDo)

More details about the LSQR algorithm can be found in [11].

### 3. The block least square method

In this section, we propose a new method for solving the linear equation (2). This method is based on the block LSQR method. We first introduce a new procedure, based on Bidiag 1, for reducing \( A \) to the block lower bidiagonal form.

The block 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 \( V_i^T V_j = 0, U_i^T U_j = 0, \) for \( i \neq j, \) and \( V_i^T V_i = I_s, U_i^T U_i = I_s; \) and they form the orthonormal basis of \( \mathbb{R}^{n \times k}. \)

**Block Bidiag 1 (Starting matrix \( B; \) reduction to block lower bidiagonal form).**

\[
\begin{align*}
U_1 B_1 &= B, & V_1 A_1 &= A^T U_1, \\
U_{i+1} B_{i+1} &= A V_i - U_i A_i^T, & V_{i+1} A_{i+1} &= A^T U_{i+1} - V_i B_{i+1}^T, & i &= 1, 2, \ldots, k,
\end{align*}
\]

where \( U_i, V_i \in \mathbb{R}^{n \times s}; \) \( B_i, A_i \in \mathbb{R}^{s \times s}; \) and \( U_1 B_1, V_1 A_1, U_{i+1} B_{i+1}, V_{i+1} A_{i+1} \) are the QR decompositions of the matrices \( B, A^T U_1, A V_i - U_i A_i^T, A^T U_{i+1} - V_i B_{i+1}^T, \) respectively. With the definitions
the recurrence relations (4) may be rewritten as:

\[
\begin{align*}
\overline{U}_{k+1}E_1B_1 & = B, \\
A\overline{V}_k & = \overline{U}_{k+1}T_k, \\
A^T\overline{U}_{k+1} & = \overline{V}_k^TT_k + V_{k+1}A_{k+1}E_{k+1}^T,
\end{align*}
\]

where \(E_i\) is the \(n \times s\) matrix which is zero except for the \(i\)th \(s\) rows, which are the \(s \times s\) identity matrix. we have also \(\overline{V}_k^TV_k = I_k\), and \(\overline{U}_{k+1}\overline{U}_{k+1} = I_{(k+1)s}\), where \(I_l\) is the \(l \times l\) identity matrix.

At iteration \(k\) we seek an approximate solution \(X_k\) of the form

\[
X_k = \overline{V}_kY_k,
\]

where \(Y_k\) is an \(ks \times s\) matrix. The residual matrix for this approximate solution is given by

\[
R_k = B - AX_k = B - A\overline{V}_kY_k = U_{k+1}B_1 - \overline{U}_{k+1}T_kY_k = \overline{U}_{k+1}(E_1B_1 - T_kY_k),
\]

where \(E_i\) is the \((k+1)s \times s\) matrix, which is zero except for the first \(s\) rows, which are the \(s \times s\) identity matrix.

Now we propose two algorithms for computing the approximate solution to (2). The new algorithms are named block LSQR 1 and 2 (BL-LSQR 1, 2) since they are the generalizations of the single right-hand side LSQR algorithm. In the block LSQR 1 algorithm we would like to choose \(Y_k \in \mathbb{R}^{ks \times s}\) such that \(\|\text{col}(R_k)\|_2\) is a minimum independent for \(j = 1, 2, \ldots, s\). In the block LSQR 2 algorithm, the matrix \(Y_k\) is selected by imposing a minimization of the Frobenius norm of the residual, \(\|R_k\|_F\).

In Section 3.1, we will show how to solve these problems.

3.1. The block LSQR 1 (BL-LSQR 1) algorithm

The block LSQR 1 algorithm chooses \(Y_k\) so that

\[
\|\text{col}(R_k)\|_2 = \|\overline{U}_{k+1}(E_1B_1 - T_kY_k)e_j\|_2
\]

is a minimum independent for \(j = 1, 2, \ldots, s\). Since the columns of the matrix \(\overline{U}_{k+1}\) are orthonormal it follows that

\[
\|\text{col}(R_k)\|_2 = \|(E_1B_1 - T_kY_k)e_j\|_2.
\]

This minimization problem is accomplished by using the QR decomposition [6], where a unitary matrix \(Q_k\) is determined so that

\[
Q_kT_k = \begin{bmatrix}
R_k \\
0
\end{bmatrix} = \begin{bmatrix}
\rho_1 & \theta_2 & & \\
\rho_2 & \theta_3 & & \\
& & \ddots & \\
& & & \rho_{k-1} & \theta_k \\
& & & \rho_k & \\
0 & & & & 0
\end{bmatrix},
\]
where $\rho_l$ and $\theta_l$ are the $s \times s$ matrices. The matrix $Q_k$ is updated from the previous iteration by setting

$$Q_k = \begin{bmatrix} I_{(k-1)s} & 0 \\ 0 & Q(a_k, b_k, c_k, d_k) \end{bmatrix} \begin{bmatrix} Q_{k-1} & 0 \\ 0 & I_s \end{bmatrix},$$

where

$$Q(a_k, b_k, c_k, d_k) = \begin{bmatrix} a_k & b_k \\ c_k & d_k \end{bmatrix}$$

is an $2s \times 2s$ unitary matrix written as four $s \times s$ blocks, and $I_s$ is the $s \times s$ identity matrix. The unitary matrix $Q(a_k, b_k, c_k, d_k)$ is computed such that

$$Q(a_k, b_k, c_k, d_k) = \begin{bmatrix} \tilde{\rho}_k & 0 \\ B_{k+1} & A_{k+1}^T \end{bmatrix} = \begin{bmatrix} \rho_k & \theta_k \\ 0 & \tilde{\rho}_{k+1} \end{bmatrix}.$$ 

A problem equivalent to minimizing (7) may then be written as minimizing

$$\left\| \text{col}_j \left( \begin{bmatrix} \tilde{R}_k \\ 0 \end{bmatrix} Y_k - Q_k E_1 B_1 \right) \right\|_2$$
for $j = 1, 2, \ldots, s$. This is easily done by defining

$$Q_k E_1 B_1 = \begin{bmatrix} F_k \\ \tilde{\phi}_{k+1} \end{bmatrix},$$
with $F_k = [\phi_1 \phi_2 \ldots \phi_k]^T$,

and setting

$$Y_k = \tilde{R}_k^{-1} F_k.$$ 

So the approximate solution is given by

$$X_k = \tilde{V}_k \tilde{R}_k^{-1} F_k.$$ 

Letting

$$\tilde{P}_k \equiv \tilde{V}_k \tilde{R}_k^{-1} \equiv [P_1 P_2 \cdots P_k]$$
then

$$X_k = \tilde{P}_k F_k.$$ 

The $n \times s$ matrix $P_k$, the last block column of $\tilde{P}_k$, can be computed from the previous $P_{k-1}$ and $V_k$, by the simple update

$$P_k = (V_k - P_{k-1} \theta_k) \rho_k^{-1}. \quad (8)$$ 

Also note that,

$$F_k = \begin{bmatrix} F_{k-1} \\ \phi_k \end{bmatrix}$$

in which

$$\phi_k = a_k \tilde{\phi}_k.$$ 

Thus, $X_k$ can be updated at each step, via the relation

$$X_k = X_{k-1} + P_k \phi_k.$$ 

The equation residual $\text{RES}_k = \max_j \| \text{col}_j (AX_k - B) \|_2$ is computed directly from the quantity $\tilde{\phi}_{k+1}$ as

$$\text{RES}_k = \max_j \| \text{col}_j (\tilde{\phi}_{k+1}) \|_2.$$ 

Some of the work in (8) can be eliminated by using matrices $W_k \equiv P_k \rho_k$ in place of $P_k$. The main steps of BL-LSQR 1 algorithm can be summarized as follows.
**Algorithm 2 (BL-LSQR 1)**

Set $X_0 = 0$

$U_1B_1 = B$, $V_1A_1 = A^TU_1$, (QR decomposition of $B$ and $A^TU_1$)

Set $W_1 = V_1$, $\tilde{\phi}_1 = B_1$, $\tilde{\rho}_1 = A_1$

For $i = 1, 2, \ldots$ until convergence Do:

$\tilde{W}_i = AV_i - U_iA^T$

$U_{i+1}B_{i+1} = \tilde{W}_i$, (QR decomposition of $\tilde{W}_i$)

$\tilde{S}_i = A^TU_{i+1} - V_iB_{i+1}^T$

$V_{i+1}A_{i+1} = \tilde{S}_i$, (QR decomposition of $\tilde{S}_i$)

Compute a unitary matrix $Q(a_i, b_i, c_i, d_i)$ such that

$Q(a_i, b_i, c_i, d_i) \left[ \begin{array}{c} \tilde{\rho}_i \\ B_{i+1} \end{array} \right] = \left[ \begin{array}{c} \rho_i \\ 0 \end{array} \right]$

$0_{i+1} = b_iA_{i+1}^T$

$\tilde{\rho}_{i+1} = d_iA_{i+1}^T$

$\tilde{\phi}_i = a_i\tilde{\phi}_i$

$\tilde{\phi}_{i+1} = c_i\tilde{\phi}_i$

$P_i = W_i\tilde{\rho}_i^{-1}$

$X_i = X_{i-1} + P_i\tilde{\phi}_i$

$W_{i+1} = V_{i+1} - P_i\tilde{\rho}_{i+1}$

If $\max_j \| \text{col}_j(\tilde{\phi}_{i+1}) \|_2$ is small enough then stop

EndDo.

The algorithm is a generalization of the LSQR algorithm. It reduces to the classical LSQR when $s = 1$. The BL-LSQR 1 algorithm will be breakdown at step $i$, if $\tilde{\rho}_i$ is singular. This happens when the matrix $\left[ \begin{array}{c} \tilde{\rho}_i \\ B_{i+1} \end{array} \right]$ is not full rank. So the BL-LSQR 1 algorithm will not breakdown at step $i$, if $B_{i+1}$ is nonsingular.

We will not treat the problem of breakdown in this paper and we assume that the matrices $B_i$'s produced by the BL-LSQR 1 algorithm are nonsingular.

### 3.2. The block LSQR 2 (BL-LSQR 2) algorithm

The block LSQR 2 algorithm chooses the matrix $Y_k$ which minimizes the Frobenius norm of the residual $R_k$. From (6), the residual $R_k$ can be written as

$R_k = \tilde{U}_{k+1} \begin{bmatrix} \tilde{E}_kB_k & -\tilde{\bar{T}}_kY_k \\ E_{k+1}^TT_kY_k \end{bmatrix}$,

where $\tilde{T}_k$ (respectively, $\tilde{E}_1$) is the matrix obtained from $T_k$ (respectively, $E_1$) by deleting its last block row and $E_{k+1}$ is the $(k+1)s \times s$ matrix, which is zero except for the last $s$ rows, which are the $s \times s$ identity matrix. But since the columns of the matrix $\tilde{U}_{k+1}$ are orthonormal it follows that

$\|R_k\|_F^2 = \left\| \begin{bmatrix} \tilde{E}_kB_k & -\tilde{\bar{T}}_kY_k \\ E_{k+1}^TT_kY_k \end{bmatrix} \right\|_F^2 = \|\tilde{E}_kB_k + \tilde{\bar{T}}_kY_k\|_F^2 + \|E_{k+1}^TT_kY_k\|_F^2$. \hspace{1cm} (9)

We now define the linear operators $\varphi_k$ and $\psi_k$ as follows:

For $Y \in \mathbb{R}^{k\times s}$

$\varphi_k(Y) = \tilde{T}_kY$

and

$\psi_k(Y) = E_{k+1}^TT_kY$. 

Then the relation (9) can be expressed as
\[ \| R_k \|_F^2 = \| \phi_k(Y_k) - \tilde{E}_1 B_1 \|_F^2 + \| \psi_k(Y_k) \|_F^2. \] (10)

Therefore, \( Y_k \) minimizes the Frobenius norm of the residual if and only if it satisfies the following linear matrix equation.
\[ \phi_k^T(\phi_k(Y_k) - \tilde{E}_1 B_1) + \psi_k^T(\psi_k(Y_k)) = 0, \] (11)

where the linear operators \( \phi_k^T \) and \( \psi_k^T \) are the transpose of the operators \( \phi_k \) and \( \psi_k \), respectively. Therefore, (11) is also written as
\[ \tilde{T}_k^T(\tilde{T}_k Y_k - \tilde{E}_1 B_1) + T_k^T E_{k+1}(E_{k+1}^T T_k Y_k) = 0. \]

Hence, \( Y_k \) is given by
\[ Y_k = \tilde{T}_k^{-1} F, \]

where
\[ \tilde{T}_k = \tilde{T}_k^T + T_k^T E_{k+1} E_{k+1}^T T_k, \quad F = \tilde{T}_k^T \tilde{E}_1 B_1 = [A_1 B_1 0 \cdots 0]^T. \]

The matrix \( \tilde{T}_k \) is a symmetric positive definite block tridiagonal matrix of the form
\[
\begin{bmatrix}
\tilde{A}_1 & \tilde{B}_2^T \\
\tilde{B}_2 & \tilde{A}_2 & \ddots \\
& \ddots & \ddots & \tilde{B}_k^T \\
& & \tilde{B}_k & \tilde{A}_k
\end{bmatrix},
\]

where \( \tilde{A}_i = A_i A_i^T + B_{i+1}^T B_{i+1}, \) for \( i = 1, 2, \ldots, k \) and \( \tilde{B}_i = A_i B_i, \) for \( i = 2, 3, \ldots, k. \)

The approximate solution of the system (2) is given by
\[ X_k = \bar{T}_k \tilde{T}_k^{-1} F. \]

Suppose that, by using the QR decomposition [6], we obtain a unitary matrix \( \bar{Q}_k \) such that
\[
\bar{Q}_k \tilde{T}_k = \begin{bmatrix}
\hat{a}_1 & \hat{b}_2 & \hat{b}_3 & \hat{b}_4 \\
\hat{a}_2 & \hat{b}_3 & \hat{b}_4 & \ddots \\
& & \ddots & \hat{b}_4 \\
& & & \hat{b}_4 \\
\hat{a}_{k-2} & \hat{b}_{k-2} & \hat{b}_k & \hat{b}_k \\
\hat{a}_{k-1} & \hat{b}_k & \hat{b}_k & \ddots \\
\hat{a}_k & \hat{b}_k & \hat{b}_k & \ddots
\end{bmatrix} = \hat{R}_k,
\]

where \( \hat{R}_k \) is upper triangular as shown and \( \hat{a}_i, \hat{b}_i, \hat{b}_i, \hat{b}_i \) are the \( s \times s \) matrices. So, by setting
\[ P_k = \bar{V}_k \tilde{T}_k^{-1} \bar{Q}_k^T \equiv [P_1 P_2 \ldots P_k] \]

and
\[ \hat{F}_k = Q_k F = [\phi_1 \phi_2 \ldots \phi_k]^T, \]

we have
\[ P_k = (V_k - P_{k-2} \hat{0}_k - P_{k-1} \hat{b}_k) \hat{a}_k^{-1}, \]
\[ X_k = X_{k-1} + P_i \phi_k. \] (12)
From (12) the residual $R_k$ is given by
\[ R_k = R_{k-1} - AP_k \varphi_k, \]
where $AP_k$ can be computed from the previous $AP_k$'s and $AV_k$ by the simple update
\[ AP_k = (AV_k - AP_{k-2} \hat{\theta}_k - AP_{k-1} \hat{\beta}_k) \tilde{\varphi}_k^{-1}. \]

Now we can summarize the above descriptions as the following algorithm.

**Algorithm 3 (BL-LSQR 2)**

Set $X_0 = 0$
Set $c_0 = a_0 = d_0 = I_s$, $b_0 = b_{-1} = d_{-1} = 0$, $\hat{\theta}_0 = \tilde{\varphi}_0 = 0_{n \times s}$
Compute $U_1 B_1 = B$, $V_1 A_1 = A^T U_1$, (QR decomposition of $B$ and $A^T U_1$)
Set $\varphi_1 = A_1 B_1$, $\tilde{\varphi}_1 = 0_{n \times s}$
For $i = 1, 2, \ldots$, until convergence Do:
\[
\begin{align*}
\tilde{W}_i &= AV_i - U_i A_i^T \\
U_{i+1} B_{i+1} &= \tilde{W}_i \text{ (QR decomposition of } \tilde{W}_i) \\
\tilde{A}_i &= A_i A_i^T + B_{i+1}^T B_{i+1} \\
\tilde{S}_i &= A_i^T U_{i+1} - V_i B_{i+1}^T \\
V_{i+1} A_{i+1} &= \tilde{S}_i \text{ (QR decomposition of } \tilde{S}_i) \\
\hat{\beta}_i &= a_{i-1} d_{i-1} B_1 + b_{i-1} A_i \\
\hat{\theta}_i &= b_{i-1} \tilde{B}_i^T + d_{i-1} \tilde{A}_i \\
\tilde{\xi}_i &= c_{i-1} d_{i-2} \tilde{B}_i^T + d_{i-1} \tilde{A}_i \\
\tilde{B}_{i+1} &= A_{i+1} B_{i+1} \text{ Compute the unitary matrix } Q(a_i, b_i, c_i, d_i) \text{ such that} \\
&\begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix} \begin{bmatrix} \tilde{\xi}_i \\ \tilde{B}_{i+1} \end{bmatrix} = \begin{bmatrix} \hat{\xi}_i \\ 0 \end{bmatrix} \\
\varphi_i &= a_i \tilde{\varphi}_i \\
\tilde{\varphi}_{i+1} &= c_i \tilde{\varphi}_i \\
P_i &= (V_i - P_{i-2} \hat{\theta}_i - P_{i-1} \hat{\beta}_i) \hat{\varphi}_i^{-1} \\
X_i &= X_{i-1} + P_i \varphi_i \\
R_i &= R_{i-1} - AP_i \tilde{\varphi}_i \\
&\text{If } \max \| \text{col}(R_i) \|_2 \text{ is small enough then stop} 
\end{align*}
\]
EndDo

The BL-LSQR 2 algorithm will be breakdown at step $i$, if $\hat{\xi}_i$ is singular. This happens when the matrix
\[
\begin{bmatrix} \tilde{\xi}_i \\ \tilde{B}_{i+1} \end{bmatrix}
\]
is not full rank. So the BL-LSQR 2 algorithm will not breakdown at step $i$, if $\tilde{B}_{i+1}$ is nonsingular. As we mentioned in Section 3.1, we will not treat the problem of breakdown in this paper and we also assume that the matrices $\tilde{B}_i$'s produced by the BL-LSQR 2 algorithm are nonsingular.

The following proposition establishes a relation between the residual matrices associated with the approximate solutions obtained by the BL-LSQR 1 and 2 algorithms.

**Proposition 1.** Let $X_k$ and $X_k'$ be the approximate solutions of (2) which obtained by BL-LSQR 1 and BL-LSQR 2 algorithm, respectively. Then we have
\[
\| R_k \|_F \leq \| R_k' \|_F^2,
\]
where $R_k$ and $R_k'$ are the residual matrices associated with the approximate solutions $X_k$ and $X'_k$, respectively.

**Proof.** By noting that BL-LSQR 1 algorithm chooses $Y_k \in \mathbb{R}^{k \times s}$ such that $\| \text{col}(R_k) \|_2$ is a minimum independent for $j = 1, 2, \ldots, s$ and BL-LSQR 2 algorithm selects $Y_k \in \mathbb{R}^{k \times s}$ such that $\| R_k \|_F$ is a minimum; from (6), (7) and (9) we have
\[
\sum_{j=1}^{s} \min_{y_j \in \mathbb{R}^n} \| (E_j B y_j - T_j y_j) \|_2^2 \leq \min_{y_j \in \mathbb{R}^n} \| E_j B y_j - T_j y_j \|_F^2,
\]

which completes the proof. \(\square\)

4. Numerical examples

In this section, we give some experimental results. Our examples have been coded in Matlab with double precision and have been executed on a PIV/1.8 GHz/Full workstation. For all the experiments, the initial guess was \(X_0 = 0\) and \(B = r\) and \((n,s)\), where function \(\text{rand}\) creates an \(n \times s\) random matrix with coefficients uniformly distributed in \([0,1]\). In all test problems, we have not used any preconditioning. All the tests were stopped as soon as, \(\max_{1 \leq j \leq s} (\| \text{col}(R_j) \|_2 / \| \text{col}(R_0) \|_2) \leq 10^{-7}\).

As [7], the first matrix test \(A_1\) represents the 5-point discretization of the operator

\[
L(u) = -u_{xx} - u_{yy} + \delta u,
\]

on the unit square \([0,1] \times [0,1]\) with homogeneous Dirichlet boundary conditions. The discretization was performed using a grid size of \(h = 1/61\) which yields a matrix of dimension \(n = 3600\); we chose \(\delta = 0.5\).

Also, we use some matrices from Harwell–Boeing collection. These matrices with their properties are shown in Table 1.

In Table 2 (respectively, Table 3), we give the ratio \(t(s)/t(1)\), where \(t(s)\) is the CPU time for BL-LSQR 1 (respectively, BL-LSQR 2) algorithm and \(t(1)\) is the CPU time obtained when applying LSQR for one right-hand side linear system. Note that the time obtained by LSQR for one right-hand side depends on which right-hand was used. So, in our experiments, \(t(1)\) is the average of the times needed for the \(s\) right-hand sides.

<table>
<thead>
<tr>
<th>Matrix \ property</th>
<th>Order</th>
<th>sym</th>
<th>nnz</th>
<th>cond</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDE900</td>
<td>900</td>
<td>No</td>
<td>4380</td>
<td>2.9e+02</td>
</tr>
<tr>
<td>PDE2961</td>
<td>2961</td>
<td>No</td>
<td>14,585</td>
<td>9.49e+02</td>
</tr>
<tr>
<td>SHERMAN4</td>
<td>1104</td>
<td>No</td>
<td>3786</td>
<td>7.2e+03</td>
</tr>
<tr>
<td>SHERMAN5</td>
<td>3312</td>
<td>No</td>
<td>20,793</td>
<td>3.9e+05</td>
</tr>
<tr>
<td>GR-30-30</td>
<td>900</td>
<td>Yes</td>
<td>4322</td>
<td>35.8e+02</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Matrix</th>
<th>(S)</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A_1)</td>
<td></td>
<td>1.52</td>
<td>1.82</td>
<td>2.02</td>
<td>2.60</td>
<td>4.02</td>
</tr>
<tr>
<td>PDE900</td>
<td></td>
<td>2.05</td>
<td>2.51</td>
<td>2.82</td>
<td>3.17</td>
<td>3.66</td>
</tr>
<tr>
<td>PDE2961</td>
<td></td>
<td>1.66</td>
<td>2.10</td>
<td>2.33</td>
<td>2.55</td>
<td>4.02</td>
</tr>
<tr>
<td>SHERMAN4</td>
<td></td>
<td>1.34</td>
<td>1.21</td>
<td>1.33</td>
<td>1.40</td>
<td>1.64</td>
</tr>
<tr>
<td>GR-30-30</td>
<td></td>
<td>2.05</td>
<td>2.29</td>
<td>2.70</td>
<td>3.02</td>
<td>3.93</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Matrix (S)</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A_1)</td>
<td>1.90</td>
<td>2.22</td>
<td>2.43</td>
<td>3.10</td>
<td>4.51</td>
</tr>
<tr>
<td>PDE900</td>
<td>2.74</td>
<td>3.22</td>
<td>3.81</td>
<td>3.91</td>
<td>4.50</td>
</tr>
<tr>
<td>PDE2961</td>
<td>2.50</td>
<td>3.06</td>
<td>3.34</td>
<td>3.70</td>
<td>4.89</td>
</tr>
<tr>
<td>SHERMAN4</td>
<td>1.77</td>
<td>1.65</td>
<td>1.71</td>
<td>1.84</td>
<td>2.10</td>
</tr>
<tr>
<td>GR-30-30</td>
<td>2.93</td>
<td>3.32</td>
<td>3.82</td>
<td>4.37</td>
<td>5.61</td>
</tr>
</tbody>
</table>
using LSQR. We note that BL-LSQR 1 (or 2) is effective if the indicator $t(s)/t(1)$ is less than $s$. In Table 2 (respectively, Table 3), we list the ratio $t(s)/t(1)$, for $s = 5, 10, 15, 20,$ and $30$, for the BL-LSQR 1 (respectively, BL-LSQR 2). As shown in Tables 2 and 3 the BL-LSQR 1 and 2 algorithms are effective and less expensive than the LSQR algorithm applied to each right-hand side. In addition, the BL-LSQR 1 algorithm is more effective than the BL-LSQR 2 algorithm which is in good agreement with Proposition 1.

For test problem SHERMAN5, we present in Table 4, the number of iterations to convergence of BL-LSQR 1 and 2 algorithms for $s = 5, 10, 15, 20,$ and $30$. In this table, $s = 1$ corresponds to the standard LSQR method. For this test problem the stopping criterion
\[ \max_{1 < j < s} \| \text{col}_j(R_k) \|_2 / \| \text{col}_j(R_0) \|_2 \leq 10^{-5} \]
was used and the maximum number of 2000 and 20,000 iterations was allowed for BL-LSQR 1 (or 2) and LSQR algorithm, respectively. A dagger signifies that the maximum allowed number of iteration was reached before convergence. We observed that the convergence of LSQR deteriorates significantly and the residual only satisfies $\| r_k \|_2 / \| r_0 \|_2 \leq 1.5 \times 10^{-2}$, even after 20,000 iterations. As can be seen from Table 4, The BL-LSQR 1 and 2 algorithms are able to obtain the solution with desired accuracy.

5. Conclusion

We have proposed in this paper two new block LSQR algorithms for nonsymmetric linear systems with multiple right-hand sides. To define these new algorithms we use the procedure which generate two sets of the orthonormal block vectors and we derive two simple recurrence formulas for generating the sequences of approximations $\{ X_k \}$ such that $\max_j \| \text{col}_j(R_k) \|_2$ or the Frobenius norm of residual $R_k$ decreases monotonically. Experimental results show that the proposed methods are effective and less expensive than the LSQR algorithm applied to each right-hand side. From Proposition 1 and the experimental results we can conclude that the BL-LSQR 1 is better than BL-LSQR 2.

References