The block LSMR method: a novel efficient algorithm for solving non-symmetric linear systems with multiple right-hand sides

F. Toutounian$^{1,2}$, M. Mojarrab$^{1,3,*}$

$^1$Department of Applied Mathematics, School of Mathematical Sciences, Ferdowsi University of Mashhad, Iran
$^2$The Center of Excellence on Modeling and Control Systems, Ferdowsi University of Mashhad, Iran
$^3$Department of Mathematics, University of Sistan and Bluchestan, Zahedan, Iran

E-mail: Maryam.Mojarrab@gmail.com

Abstract

It is well known that if the coefficient matrix in a linear system is large and sparse or sometimes not readily available, then iterative solvers may become the only choice. The block solvers are an attractive class of iterative solvers for solving linear systems with multiple right-hand sides. In general, the block solvers are more suitable for dense systems with preconditioner. In this paper, we present a novel block LSMR (least squares minimal residual) algorithm for solving non-symmetric linear systems with multiple right-hand sides. This algorithm is based on the block bidiagonalization and LSMR algorithm and derived by minimizing the 2-norm of each column of normal equation. Then, we give some properties of the new algorithm. In addition, the convergence of the stated algorithm is studied. In practice, we also observe that the Frobenius norm of residual matrix decreases monotonically. Finally, some numerical examples are presented to show the efficiency of the new method in comparison with the traditional LSMR method.

Keywords: LSMR method; bidiagonalization; block methods; iterative methods; multiple right-hand sides

1. Introduction

Many applications require the solution of several sparse systems of equations

$$Ax^{(i)} = h^{(i)}, \quad i = 1, 2, ..., s,$$  \hspace{1cm} (1)

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

$$AX = B,$$  \hspace{1cm} (2)

where $A$ is an $n \times n$ nonsingular and nonsymmetric real matrix, $B$ and $X$ are $n \times s$ rectangular matrices whose columns are $b^{(1)}, b^{(2)}, ..., b^{(s)}$ and $x^{(1)}, x^{(2)}, ..., x^{(s)}$, 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.

One class of solvers for solving the problem (2) 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 block conjugate gradient (Bl-CG) algorithm and block biconjugate gradient (Bl-BCG) algorithm proposed in (Oleary, 1980). Variable Bl-CG algorithms for symmetric positive definite problems are implemented on parallel computers (Haase and Reitzinger, 2005; Nikishin and Yeremin, 1995). If the matrix is only symmetric, an adaptive block Lanczos algorithm and a block version of Minres method are devised in (Dai, 2000). For nonsymmetric problems, the Bl-BCG algorithm (Oleary, 1980; Simoncini, 1997), the block generalized minimal residual (Bl-GMRES) algorithm (Darnel et al., 2008; Gu and Cao, 2001; Gutknecht, 2007; Liu and Zhong, 2008; Morgan, 2005; Robbe and Sadkane, 2006; Simoncini and Gallopoulos, 1995; Simoncini and Gallopoulos, 1996; Vital, 1990), the block quasi minimum residual (Bl-QMR) algorithm (Freund and Malhotra, 1997), the block BiCGStab (Bl-BICGSTAB) algorithm (Guennouni et al., 2003), the block Lanczos method (Guennouni et al., 2004) and the block least squares (Bl-LSQR) algorithm (Karimi and Toutounian, 2006) have been developed.

Another class is the global methods, which are based on the use of a global projection process onto a matrix Krylov subspace, including global FOM and GMRES methods (Bellalij et al., 2008; Jbilou et al., 1999), global conjugate gradient type
methods (Salkuyeh, 2006), global BCG and BiCGStab methods (Jbilou et al., 1997; Jbilou et al., 2005), global CGS algorithm (Zhang and Dai, 2008; Zhang et al., 2010), GI-LSQR algorithm (Toutounian and Karimi, 2006), GI-BCR and GI-CRS algorithms (Zhang et al., 2011), global Hessenberg and CMRH methods (Heyouni, 2001; Lin, 2005), global SCD algorithm (Gu and Yang, 2007), and weighted global methods (Heyouni and Essai, 2005). Global methods are more suitable for sparse linear systems (Heyouni and Essai, 2005).

The other 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. References on this class include (Abdel-Rehim et al., 2008; Chan and Wang, 1997; Joly, 1991; Saad, 1987; Simoncini and Gallopoulos, 1995; Smith et al., 1989; Van Der Vorst, 1987).

In this paper, based on the block bidiagonalization, we derive a simple recurrence formula for generating a sequence of approximations \( \{X_k\} \) such that the \( \| \text{col}_i(A^T R_k) \|_2 \) decreases monotonically, where \( R_k = B - A X_k \) and \( \text{col}_i(A^T R_k) \) represents the \( i \)th column of \( A^T R_k \).

Throughout this paper, the following notations are used. For two \( n \times s \) matrices \( X \) and \( Y \), we define the following inner product: \( \langle X, Y \rangle = \text{tr}(X^T Y) \), where \( \text{tr}(Z) \) denoted the trace of the square matrix \( Z \). The associated norm is the Frobenius norm denoted by \( \| \cdot \|_F \). We will use the notation \( \langle \cdot, \cdot \rangle \) for the usual inner product in \( \mathbb{R}^n \) and the associated norm denoted by \( \| \cdot \|_2 \). Finally, \( 0 \) 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 LSMR method and its properties. In Section 3, we present the block version of the LSQR algorithm. In Section 4, the convergence of the presented algorithm is considered. In Section 5, some numerical experiments on test matrices from the University of Florida Sparse Matrix Collection (Davis, 2011) are presented to show the efficiency of the method. Finally, we make some concluding remarks in Section 6.

2. The LSMR algorithm

In this section, we recall some fundamental properties of LSMR algorithm (Chin-Lung Fong and Saunders, 2011), which is an iterative method for solving real linear systems of the form \( A x = b \), where \( A \) is a nonsymmetric matrix of order \( n \) and \( x, b \in \mathbb{R}^n \).

LSMR algorithm uses an algorithm of Golub and Kahan (Golub and Kahan, 1965), which is stated as procedure Bidiag 1 (Paige and Saunders, 1982), to reduce \( (b, A) \) to the upper-diagonal form \( (\beta, e, B_k) \). The procedure Bidiag 1 can be described as follows.

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

\[
\begin{align*}
\beta_1 u_1 &= b, \\
\alpha_1 v_1 &= A^T u_1, \\
\beta_{i+1} u_{i+1} &= A v_i - \alpha_i u_i, \\
\alpha_{i+1} v_{i+1} &= A^T u_{i+1} - \beta_{i+1} v_i, \\
\| v_i \| &= 1, 2, ..., (3)
\end{align*}
\]

The scalars \( \alpha_i \geq 0 \) and \( \beta_i \geq 0 \) are chosen so that \( \| u_i \|_2 = \| v_i \|_2 = 1 \). With the definition

\[
\begin{align*}
U_k &= \begin{bmatrix} u_1, u_2, ..., u_s \end{bmatrix}, \\
V_k &= \begin{bmatrix} v_1, v_2, ..., v_s \end{bmatrix}, \\
B_k &= \begin{bmatrix} \beta_1 & \alpha_2 & \vdots & \alpha_k \\ \beta_2 & \alpha_3 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \alpha_k \\ \beta_k & \vdots & \ddots & \beta_{k+1} \end{bmatrix}, \\
L_{k+1} &= (B_k \alpha_{k+1} e_{k+1}), \\
V_{k+1} &= (V_k \ V_{k+1})
\end{align*}
\]

the recurrence relations (3) may be rewritten as

\[
\begin{align*}
U_{k+1} (\beta_1 e_1) &= b, \\
A V_k &= U_{k+1} B_k, \\
A^T U_{k+1} &= V_k^T B_k^T + \alpha_{k+1} v_{k+1} e_{k+1}^T = V_{k+1} L_{k+1}^T, \\
A^T A V_k &= A^T U_{k+1} B_k = V_{k+1} L_{k+1}^T B_k \\
&= V_{k+1} \begin{bmatrix} b_1^T \\
\alpha_{k+1} e_{k+1}^T \\
\vdots \\
\alpha_k e_{k+1}^T \\
\beta_k e_{k+1} \end{bmatrix} B_k \\
&= V_{k+1} \begin{bmatrix} B_k^T \\
\alpha_{k+1} e_{k+1}^T \\
\vdots \\
\alpha_k e_{k+1}^T \\
\beta_k e_{k+1} \end{bmatrix}
\end{align*}
\]

This is equivalent to what would be generated by the symmetric Lanczos process with matrix \( A^T A \) and starting vector \( A^T b \).

Hence, by using the procedure Bidiag 1 the LSMR method constructs an approximation solution of the form \( x_k = V_k y_k \) which solves the least-squares problem, \( \min_{y_k} \| A^T y_k \|_2 \). The main steps of the LSMR algorithm can be summarized as follows.

**Algorithm 1 LSMR algorithm**

Set \( \beta_1 u_1 = b, \alpha_1 v_1 = A^T u_1, \alpha_1 = 1, v_1 = 1, x_0 = 0, h_1 = v_1, \beta_0 = 0, x_0 = 0 \)

For \( k = 1, 2, ..., \) until convergence Do:

\[
\begin{align*}
\beta_{k+1} u_{k+1} &= A V_k - \alpha_k u_k, \\
\alpha_{k+1} v_{k+1} &= A^T u_{k+1} - \beta_{k+1} v_k, \\
\rho_k &= (\alpha_k + \beta_{k+1})^2
\end{align*}
\]
$$c_k = \frac{\overline{a}_k}{\overline{p}_k}$$
$$s_k = \frac{\overline{p}_{k+1}}{\overline{p}_k}$$
$$\theta_{k+1} = s_k a_{k+1}$$
$$\overline{a}_{k+1} = c_k a_{k+1}$$
$$\overline{p}_k = s_{k-1} p_k$$
$$\overline{c}_k = (\overline{c}_{k-1} p_k)^2 + \theta_{k+1}^2$$
$$\overline{s}_k = \frac{\overline{c}_{k-1} p_k}{\overline{p}_k}$$
$$\xi = \overline{c}_k$$
$$\Xi_{k+1} = -s_{k-1} \overline{c}_k$$
$$h_k = h_{k-1} - \left( \frac{\overline{p}_k}{(p_{k-1} - s_{k-1})} \right) h_{k-1}$$
$$x_k = x_{k-1} + \left( \frac{\xi_k}{p_{k-1}} \right) h_k$$
$$h_{k+1} = v_{k+1} - \left( \frac{\theta_{k+1}}{\rho_k} \right) h_k.$$ 

If $|w_k|_1$ is small enough then stop.

End Do.

More details about the LSQR algorithm can be found in (Chin-Lung Fong and Saunders, 2011).

3. The block LSQR method

In this section, we present a new method for solving the linear equation (2). This method is based on the LSQR method. We use the block Bidiag I (Karimi and Toutounian, 2006), based on Bidiag I, for reducing $A$ to the block lower bidiagonal form.

The block Bidiag I procedure constructs the sets of the $n \times n$ block vectors $V_1, V_2, \ldots$ and $U_1, U_2, \ldots$ such that $V_i^T V_j = 0_s$, $U_i^T U_j = 0_s$, 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 I.** (Starting matrix $B$; reduction to block lower bidiagonal form).

$$U_1 B_1 = B, \quad V_i A_i = A^T U_i,$$
$$U_{i+1} B_{i+1} = A V_i - U_i A_i,$$
$$V_{i+1} A_{i+1} = A^T U_{i+1} - V_i B_{i+1}.$$

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 $B_i A_i U_i - A_i V_i$ and $A^T U_{i+1} - V_i B_{i+1}$, respectively. With the definitions

$$U_k \equiv [U_1, \ldots, U_k],$$

$$V_k \equiv [V_1, V_2, \ldots, V_k],$$

$$T_k \equiv \begin{bmatrix} A_1^T & B_2 & A_2^T & \cdots & B_k & A_k^T \\ & B_2 & A_2^T & \cdots & B_k & A_k^T \\ & & \ddots & \ddots & \cdots & \cdots \\ & & & B_k & A_k^T \\ & & & & B_{k+1} \\ & & & & & B_{k+1} \end{bmatrix},$$

the recurrence relations (4) may be rewritten as:

$$\overline{U}_{k+1} E_1 B_1 = B,$$
$$A \overline{V}_k = \overline{U}_{k+1} T_k,$$

$$A^T \overline{U}_{k+1} = \overline{V}_k T_k^T + V_{k+1} A_{k+1} E_{k+1}^T,$$

where $E_i$ is the $(k + 1) \times s$ matrix which is zero except for the $i$th $s \times s$ block, which are the $s \times s$ identity matrix. We also have $V_k^T V_k = I_{ks}$ and $\overline{U}_{k+1}^T \overline{U}_{k+1} = I_{(k+1)s}$, where $I_1$ is the $1 \times 1$ identity matrix. We define

$$\overline{L}_{k+1} \equiv \begin{bmatrix} T_k & E_{k+1} A_k^T \end{bmatrix},$$

then

$$A^T \overline{U}_{k+1} = \overline{V}_{k+1} \overline{L}_{k+1}^T,$$

$$A^T A \overline{V}_k = A^T \overline{U}_{k+1} T_k = \overline{V}_{k+1} \overline{L}_{k+1} T_k =$$

$$\overline{L}_{k+1} \begin{bmatrix} T_k^T & A_{k+1} E_{k+1}^T \end{bmatrix} T_k$$

$$= \overline{L}_{k+1} \begin{bmatrix} T_k^T T_k & T_k^T A_{k+1} E_{k+1}^T \end{bmatrix}.$$  \hspace{1cm} (5)

At iteration $k$ we seek an approximate solution $X_k$ of the form

$$X_k = \overline{V}_k Y_k,$$  \hspace{1cm} (6)

where $Y_k$ is an $k \times s$ matrix. Let $\overline{B}_k \equiv A_k B_k$ for all $k$.

$$A^T R_k = A^T B - A^T A X_k,$$

$$= V_i A_1 B_1 - A^T A \overline{V}_k Y_k,$$

we have

$$A^T R_k = V_i \overline{B}_1 - \overline{V}_{k+1} \begin{bmatrix} T_k^T & A_{k+1} E_{k+1}^T \end{bmatrix} Y_k,$$

$$\overline{V}_{k+1} \begin{bmatrix} E_i \overline{B}_1 - \begin{bmatrix} T_k^T & A_{k+1} E_{k+1}^T \end{bmatrix} Y_k \end{bmatrix}.$$  \hspace{1cm} (7)

where $E_i$ is the $k \times s$ matrix, which is zero except for the $i$th $s \times s$ block, which are the $s \times s$ identity matrix. Now, we use the QR decomposition (Golub and Van Loan, 1983), where a unitary matrix $Q_{k+1}$ is determined so that
\( Q_{k+1}T_k = \begin{bmatrix} \overline{R}_k \\ 0 \end{bmatrix}, \overline{R}_k = \begin{bmatrix} \rho_1 & \theta_2 & \theta_3 & \cdots & \theta_{k-1} & \theta_k \\ \rho_2 & \rho_1 & & & & \\ \vdots & & \ddots & & & \\ \rho_{k-1} & & & & \rho_1 & \theta_k \\ \rho_k & & & & & \rho_1 \end{bmatrix} \) \hspace{1cm} (8)

where \( \rho_i \) and \( \theta_i \) are the \( s \times s \) matrices. As (Karimi and Toutounian, 2006), the matrix \( Q_{k+1} \) is updated from the previous iteration by setting

\[
Q_{k+1} = \begin{bmatrix} I^{(k-1)s} & 0 \\ 0 & Q(a_k, b_k, c_k, d_k) \end{bmatrix} \begin{bmatrix} Q_k \\ 0 \\ 1_s \end{bmatrix}
\]

where

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

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

\[
Q(a_k, b_k, c_k, d_k) \begin{bmatrix} \hat{\rho}_k & 0 \\ B_{k+1}^T & A_{k+1}^T \end{bmatrix} = \begin{bmatrix} \rho_k & \theta_{k+1} \\ 0 & \hat{\rho}_{k+1} \end{bmatrix} \begin{bmatrix} \rho_k & \theta_{k+1} \\ 0 & \hat{\rho}_{k+1} \end{bmatrix}^{-1}.
\]

(9)

From (7), we have

\[
A^T R_k = \overline{V}_{k+1}(E_1 \overline{E}_1^{T} - \begin{bmatrix} T_k & B_{k+1}^T E_k \\ B_{k+1} & E_k \end{bmatrix} Y_k)
\]

\[
= \overline{V}_{k+1}(E_1 \overline{E}_1^{T} - \begin{bmatrix} \rho_k & 0 \\ 0 & \hat{\rho}_{k+1} \end{bmatrix}^{-1} F_k).
\]

If we define \( F_k \equiv \overline{R}_k Y_k, \) we have

\[
A^T R_k = \overline{V}_{k+1}(E_1 \overline{E}_1^{T} - \begin{bmatrix} R_k & 0 \\ 0 & \hat{\rho}_{k+1} \end{bmatrix}^{-1} F_k).
\]

(10)

\[
\overline{V}_{k+1}(E_1 \overline{E}_1^{T} - A_{k+1}^T B_{k+1} F_k).
\]

(11)

On the other hand, from (9), we have \( \theta_{k+1} = b_k A_{k+1} \) and

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

This implies that \( B_{k+1} = b_k \rho_k. \) So, \( \theta_{k+1} = \rho_k^{-1} B_{k+1}^T A_{k+1} \)

(12)

and

\[
A^T R_k = \overline{V}_{k+1}(E_1 \overline{E}_1^{T} - \begin{bmatrix} R_k & 0 \\ 0 & \hat{\rho}_{k+1} \end{bmatrix}^{-1} F_k).
\]

(13)

Then we perform a second QR factorization

\[
\overline{Q}_{k+1} \begin{bmatrix} R_k^{T} & E_1 \overline{E}_1 \\ \theta_{k+1}^{T} E_k \end{bmatrix} = \begin{bmatrix} \overline{R}_k & z_k \\ 0 & \overline{\tau}_{k+1} \end{bmatrix}.
\]

(14)

where \( \overline{\rho}_i \) and \( \overline{\theta}_i \) are the \( s \times s \) matrices. The matrix \( \overline{Q}_{k+1} \) is updated from the previous iteration by setting

\[
\overline{Q}_{k+1} = \begin{bmatrix} I^{(k-1)s} & 0 \\ 0 & \overline{Q}(a_k, b_k, c_k, d_k) \end{bmatrix} \begin{bmatrix} \overline{Q}_k \\ 0 \\ 1_s \end{bmatrix}
\]

where

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

is a \( 2s \times 2s \) unitary matrix written as four \( s \times s \) blocks. The unitary matrix \( \overline{Q}(a_k, b_k, c_k, d_k) \) is computed such that

\[
Q(\overline{a}_k, \overline{b}_k, \overline{c}_k, \overline{d}_k) \begin{bmatrix} \overline{\rho}_k & 0 \\ \overline{B}_{k+1}^T & \overline{A}_{k+1}^T \end{bmatrix} = \begin{bmatrix} \overline{\rho}_k & \overline{\theta}_{k+1} \\ 0 & \overline{\rho}_{k+1} \end{bmatrix}.
\]

Combining what we have with (13) gives

\[
A^T R_k = \overline{V}_{k+1} \overline{Q}_{k+1} \begin{bmatrix} \overline{V}_{k+1}^{T} \\ 0 \end{bmatrix} - \overline{\overline{R}}_k F_k.
\]

(15)

In the block LSMR algorithm we would like to choose \( Y_k \in \mathbb{R}^{k \times s} \) and correspondingly \( F_k = \overline{R}_k Y_k \) such that \( \left\| \text{col}(A^T R_k) \right\|_2 \) is a minimum independent for \( j = 1, 2, \ldots, s. \) Since \( \overline{Q}_{k+1} \) is a unitary matrix and \( \overline{V}_{k+1} \) is orthogonal, we get

\[
\min_{Y_k} \left\| \text{col}(A^T R_k) \right\|_2 = \min_{F_k} \left\| \text{col}(\overline{V}_{k+1}^{T} \overline{Q}_{k+1}^{T} \begin{bmatrix} \overline{V}_{k+1}^{T} \\ 0 \end{bmatrix} - \overline{\overline{R}}_k F_k) \right\|_2.
\]

(16)

The subproblem is solved by choosing \( F_k \) from

\[
\overline{R}_k F_k = z_k.
\]

Therefore,

\[
A^T R_k = \overline{V}_{k+1} \overline{Q}_{k+1} \overline{V}_{k+1}^{T} \begin{bmatrix} \overline{V}_{k+1}^{T} \\ 0 \end{bmatrix} - \overline{\overline{R}}_k F_k.
\]

(17)

and the approximate solution is given by

\[
X_k = \overline{V}_{k+1} \overline{Q}_{k+1} \overline{V}_{k+1}^{T} \begin{bmatrix} \overline{V}_{k+1}^{T} \\ 0 \end{bmatrix}.
\]

Letting

\[
\overline{F}_k \equiv \overline{V}_{k+1} \overline{R}^{-1} \overline{R}_k z_k.
\]

then
\[ X_k = \overline{F}_k z_k. \]

The \( n \times s \) matrix \( P_k \), the last block column of \( \overline{F}_k \), can be computed from the previous \( P_{k-1} \), \( P_{k-2} \) and \( V_k \), by the update

\[
P_k = (V_k - P_{k-2} \overline{b}_{k-1} \theta_k - P_{k-1} (\overline{b}_{k-1} \theta_k + \overline{b}_k \rho_k)) (\overline{p}_k \rho_k)^{-1}.
\]  

Also note that,

\[
z_k = \begin{bmatrix} z_{k-1} \\ \zeta_k \end{bmatrix}
\]

in which

\[
\zeta_k = \overline{a}_k \overline{r}_k.
\]

Thus, \( X_k \) can be updated at each step, via the relation

\[ X_k = X_{k-1} + P_k \zeta_k. \tag{20} \]

The equation \( \text{RES}_k = \max_i || \text{col}_i (A_k^T R_k) ||_2 \) is computed directly from the quantity \( \overline{r}_{k+1} \) as

\[ \text{RES}_k = \max_i || \text{col}_i (\overline{r}_{k+1}) ||_2. \]

From (20), the residual \( R_k \) is given by

\[ R_k = R_{k-1} - A P_k \zeta_k, \tag{21} \]

where \( A P_k \) can be computed from the previous \( A P_k \)'s and \( A V_k \) by the simple update

\[ A P_k = (A V_k - A P_{k-2} \overline{b}_{k-1} \theta_k - A P_{k-1} (\overline{b}_{k-1} \theta_k + \overline{b}_k \rho_k)) (\overline{p}_k \rho_k)^{-1}. \]

In addition, we show that the \( || \text{col}_i (R_k) ||_2 \) can be estimated by a simple formula. We transform \( \overline{R}_k \) to block upper-bidiagonal form using a third QR factorization: \( \overline{R}_k = \overline{Q}_k \overline{F}_k \) with \( \overline{Q}_k = \overline{P}_{k-1} \ldots \overline{P}_1 \). By defining

\[ \overline{F}_k = \overline{Q}_k F_k, \quad \overline{B}_k = \begin{bmatrix} \overline{Q}_k \\ I_s \end{bmatrix} Q_{k+1} E_1 B_k \]

we have

\[ R_k = B - A X_k V_k \]

\[ = U_{k+1} (E_1 B_k - T_k Y_k) \]

\[ = \overline{U}_{k+1} (E_1 B_k - Q_{k+1}^{\top} \begin{bmatrix} \overline{R}_k \\ 0_s \end{bmatrix} Y_k) \]

\[ = \overline{U}_{k+1} (E_1 B_k - Q_{k+1}^{\top} \begin{bmatrix} \overline{F}_k \\ 0_s \end{bmatrix}) \]

\[ = \overline{U}_{k+1} (Q_{k+1}^{\top} \begin{bmatrix} Q_{k+1}^{\top} \\ I_s \end{bmatrix} \overline{B}_k - Q_{k+1}^{\top} \begin{bmatrix} Q_{k+1}^{\top} \overline{F}_k \\ 0_s \end{bmatrix}). \]

\[ = \overline{U}_{k+1} Q_{k+1}^{\top} \begin{bmatrix} Q_{k+1}^{\top} \\ I_s \end{bmatrix} \left[ B_k - \left[ \begin{bmatrix} \overline{F}_k \\ 0_s \end{bmatrix} \right] \right] \]

\[ = \overline{U}_{k+1} Q_{k+1}^{\top} \begin{bmatrix} Q_{k+1}^{\top} \\ I_s \end{bmatrix} \left[ B_k - \left[ \begin{bmatrix} \overline{F}_k \\ 0_s \end{bmatrix} \right] \right] \]

Since the columns of the matrices \( Q_{k+1} \), \( \overline{Q}_{k+1} \), and \( \overline{U}_{k+1} \) are orthonormal, we have

\[ || \text{col}_i (R_k) ||_2 = || B_k - \left[ \begin{bmatrix} \overline{F}_k \\ 0_s \end{bmatrix} \right] e_i ||_2. \tag{23} \]

The matrices \( \overline{B}_k \) and \( \overline{F}_k \) can be written in the form

\[ \overline{B}_k = \begin{bmatrix} \overline{b}_1^{\top} \\ \overline{b}_k^{\top} \end{bmatrix}, \quad \overline{F}_k = \begin{bmatrix} \overline{f}_1^{\top} \\ \overline{f}_k^{\top} \end{bmatrix} \]

The following Lemma shows that we can estimate \( || \text{col}_i (R_k) ||_2 \) from just the last two blocks of \( \overline{B}_k \) and the last block of \( \overline{F}_k \).

**Lemma 1.** In (23) and (24), \( \overline{b}_i = \overline{r}_i \) for \( i = 1, 2, ..., k - 1 \).

**Proof:** Appendix A proves the lemma by induction.

The main steps of BI-LSMR algorithm can be summarized as follows.

**Algorithm 2 BI-LSMR algorithm**

Set \( X_0 = 0, \rho_0 = I_s, \overline{p}_0 = I_s, \overline{d}_0 = I_s, \overline{y}_0 = 0, \overline{b}_1 = 0, \theta_1 = 0 \)

\( U_1 B_1 = B, V_1 A_1 = A_1^T U_1 \) (QR decomposition of \( B \) and \( A_1^T U_1 \))

Set \( \overline{B}_1 = A_1 B_1, \overline{p}_1 = A_1^T \zeta_1 = \overline{B}_1, \overline{d}_1 = 0 \)

For \( i = 1, 2, \ldots \) until convergence, do:

\[ \overline{w}_i = A_1 V_i - U_i A_1^T \]

\[ U_{i+1} B_{i+1} = \overline{w}_i \] (QR decomposition of \( \overline{w}_i \))

\[ S_i = A_1^T U_{i+1} - V_i B_{i+1}^\top \]

\[ V_{i+1} A_{i+1} = S_i \] (QR decomposition of \( S_i \))

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

\[ Q(a_i, b_i, c_i, d_i) \begin{bmatrix} \overline{b}_i \\ \overline{d}_i \end{bmatrix} = \begin{bmatrix} \overline{p}_i \\ \overline{p}_i \end{bmatrix} \]

\[ \overline{b}_i = b_i A_{i+1}^T \]

\[ \overline{d}_i = d_i A_{i+1}^T \]

\[ \overline{p}_i = \overline{d}_i - \overline{b}_i \theta_i \]

Compute a unitary matrix \( Q(\overline{a}_i, \overline{b}_i, \overline{c}_i, \overline{d}_i) \) such that

\[ Q(\overline{a}_i, \overline{b}_i, \overline{c}_i, \overline{d}_i) \begin{bmatrix} \overline{b}_i \\ \overline{d}_i \end{bmatrix} = \begin{bmatrix} \overline{p}_i \\ \overline{p}_i \end{bmatrix} \]

\[ \overline{a}_i = \overline{a}_i \]

\[ \overline{b}_i = b_i A_{i+1}^T \]

\[ \overline{d}_i = d_i A_{i+1}^T \]

\[ \overline{p}_i = \overline{d}_i - \overline{b}_i \theta_i \]
$$X_1 = X_{i-1} + P_i ζ_i$$

If $\max \| \text{col}(P_{i+1}) \|_2$ is small enough then stop.

End Do.

The algorithm is a generalization of the LSMR algorithm. It reduces to the classical LSMR when $s = 1$. The Bl-LSMR algorithm will breakdown at step $i$, if $P_i$ or $P_i$ are singular. This happens when the matrix $[\tilde{P}_i \ | \ B_{i+1}]$ or matrix $[\tilde{P}_i \ | \ I_{i+1}]$ is not full rank. So the Bl-LSMR algorithm will not breakdown at step $i$, if $B_{i+1}$ and $θ_{i+1}$ are nonsingular. We will not treat the problem of breakdown in this paper and we assume that the matrices $B_i$’s and $θ_{i+1}$’s produced by the Bl-LSMR algorithm are nonsingular.

4. The convergence of the Bl-LSMR algorithm

In this section, the convergence of the Bl-LSMR algorithm is studied. In order to get this goal, we first give the following lemma.

Lemma 2. Let $P_i, i = 1, 2, ..., k$, be the $n \times s$ auxiliary matrices produced by the Bl-LSMR algorithm and $R_k$ be the residual matrix associated with the approximate solution $X_k$ of the matrix equation (2). Then, we have

(a) $P_i^T A T A T P_j = \begin{cases} I_s, & i = j \\ 0_s, & i ≠ j \end{cases}$

(b) $(A^T R_{k+1})^T A T P_k = ζ_k^T$

Proof: (a) It is enough to show that

$$P_k A^T A T A^T P_k = I_{ks},$$

where $I_{ks}$ is a $ks \times ks$ identity matrix and $P_k = [P_1, P_2, ..., P_k]$ is an $n \times ks$ matrix whose columns are defined by (19). By using $P_k = \bar{V}_k R_k^{-1} P_k$ and equation (5), we have

$$A^T A P_k = A^T A \bar{V}_k R_k^{-1} P_k$$

$$= \bar{V}_{k+1} T_k \begin{bmatrix} T_k & T_k A k+1 E_k + T_k \end{bmatrix} A^T P_k^{-1} R_k^{-1}$$

$$= \bar{V}_{k+1} \begin{bmatrix} R_k^T P_k \theta_k & \theta_k \end{bmatrix} R_k^{-1} P_k^{-1}$$

$$= \bar{V}_{k+1} \begin{bmatrix} R_k^T \theta_k \end{bmatrix} R_k^{-1}$$

(Proof)

$$= \bar{V}_{k+1} \begin{bmatrix} \theta_k & I_{ks} \end{bmatrix} R_k^{-1}$$

From (18), (25), and (15), we have

$$\bar{P}_k^T A^T A P_k = \bar{V}_{k+1} Q_{k+1}^T I_{ks} \bar{E}_k$$

$$= \bar{V}_{k+1} Q_{k+1}^T I_{ks} \bar{E}_k$$

Since $\bar{V}_{k+1}$ is orthonormal and $Q_{k+1}$ is a unitary matrix, we get

$$\bar{P}_k^T A^T A P_k = I_{ks}.$$

(b) From (18), (25), and (15), we have

$$(A^T R_{k-1})^T A^T P_k = (\bar{V}_k Q_k \begin{bmatrix} 0 & I_{ks} \end{bmatrix} K_{k+1})^T \bar{V}_{k+1} Q_{k+1}^T I_{ks} \bar{E}_k$$

$$= \bar{V}_{k+1} Q_{k+1}^T I_{ks} \bar{E}_k$$

By using Lemma 2, we get

$$(A^T R_k)^T (A^T R_k) = (A^T R_{k-1})^T (A^T R_{k-1}) - ζ_k^T ζ_k.$$
Our examples have been coded in Matlab with double precision. For all the experiments, the initial guess was $X_0 = 0$ and $B = \text{rand}(n,s)$, where function rand creates an $n \times s$ random matrix with coefficients uniformly distributed in $[0,1]$. No preconditioning has been used in any of the test problems. All the tests were stopped as soon as, $\|A^T R_{k}\|_F \leq \|s_{k+1}\|_F \leq 10^{-10}$.

We use some matrices from the University of Florida Sparse Matrix Collection (Davis [7]). These matrices with their properties are shown in Table 1. In Table 2, for $s = 10,20,30,$ and $40$, we give the ratio $t(s)/t(1)$, where $t(s)$ is the CPU time for BI-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, in our experiments, $t(1)$ is the average of the times needed for the $s$ right-hand sides using LSMR. We note that BI-LSMR is effective if the indicator $t(s)/t(1)$ is less than $s$. Table 2 shows that the BI-LSMR algorithm is effective and less expensive than the LSMR algorithm applied to each right-hand side. To show that the Frobenius norm of residual matrix decreases monotonically, we display the convergence history in Fig. 1 for the systems corresponding to the matrices of Table 2 and BI-LSMR algorithm, respectively. In this figure, the vertical axis and horizontal axis are the logarithm of the Frobenius norm of residual matrix and the number of iterations to convergence, respectively. We observe that for all matrices the Frobenius norm of residual matrix decreases monotonically.

<table>
<thead>
<tr>
<th>Matrix\Property</th>
<th>Order</th>
<th>Sym</th>
<th>nnz</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rajat/rajan03</td>
<td>7602</td>
<td>no</td>
<td>32653</td>
<td>1187</td>
</tr>
<tr>
<td>HB/</td>
<td>1104</td>
<td>no</td>
<td>3786</td>
<td>245</td>
</tr>
<tr>
<td>sherman4</td>
<td>20414</td>
<td>no</td>
<td>165979</td>
<td>925</td>
</tr>
<tr>
<td>FEMLAB/ns3Da</td>
<td>14000</td>
<td>no</td>
<td>1853104</td>
<td>811</td>
</tr>
<tr>
<td>Simon/appu</td>
<td>2961</td>
<td>no</td>
<td>14585</td>
<td>34</td>
</tr>
<tr>
<td>Bai/pde2961</td>
<td>14000</td>
<td>no</td>
<td>1853104</td>
<td>811</td>
</tr>
</tbody>
</table>

| Table 1. Test problems information |

<table>
<thead>
<tr>
<th>Matrix(s)</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rajat03</td>
<td>1.98</td>
<td>6.31</td>
<td>3.54</td>
<td>18.98</td>
</tr>
<tr>
<td>Sherman4</td>
<td>0.15</td>
<td>0.16</td>
<td>0.18</td>
<td>0.45</td>
</tr>
<tr>
<td>Ns3Da</td>
<td>2.15</td>
<td>4.50</td>
<td>7.00</td>
<td>10.25</td>
</tr>
<tr>
<td>appu</td>
<td>0.93</td>
<td>5.96</td>
<td>0.98</td>
<td>15.10</td>
</tr>
<tr>
<td>Pde2961</td>
<td>0.70</td>
<td>1.04</td>
<td>0.81</td>
<td>0.87</td>
</tr>
</tbody>
</table>

| Table 2. Effectiveness of BI-LSMR algorithm measured by \(t(s)/t(1)\) |

6. Conclusion

We have proposed in this paper a new block LSMR algorithm for solving nonsymmetric linear systems with multiple right-hand sides. To define this new algorithm, we used the block BiDiag 1 procedure (Karimi and Toutounian, 2006) and derived a simple recurrence formulas for generating the sequences of approximations \(\{X_k\}\) such that $\|\text{col}_1(A^TR_{k})\|_2$ decreases monotonically. In practice we also observed that the Frobenius norm of residual matrix decreased monotonically. Experimental results showed that the proposed methods are effective and less expensive than the LSMR algorithm applied to each right-hand side.

Fig. 1. Convergence history of the BI-LSMR algorithm with \(s=10\)

Appendix A. Proof of Lemma 1.

Let $Q_k = \tilde{P}_k\tilde{P}_{k-1}...\tilde{P}_1$, $Q_k = \bar{P}_k\bar{P}_{k-1}...\bar{P}_1$, and

$R_k = \begin{bmatrix} \bar{\rho}_1 & \bar{\rho}_2 & \cdots & \bar{\rho}_{k-1} & \bar{\rho}_k \end{bmatrix}$

(26)

Then, with $\bar{\rho}_1 = B_1$, $\bar{\rho}_k = \rho_k^T$, $\bar{\beta}_1 = \bar{\beta}_1$, $\bar{\beta}_k = \bar{\beta}_k$, and $\bar{\beta}_1 = A_1^T$, the effects of the rotations $\tilde{P}_k, \tilde{P}_{k}$ and $\bar{P}_{k-1}$ can be summarized as

$A_k \begin{bmatrix} a_k & b_k \end{bmatrix} = \begin{bmatrix} \tilde{P}_k & 0 \\ 0 & \tilde{P}_{k+1} \end{bmatrix} \begin{bmatrix} \rho_{k+1} \\ \beta_{k+1} \end{bmatrix}$

(27)

and

$\begin{bmatrix} a_k & b_k \end{bmatrix} = \begin{bmatrix} \tilde{P}_k & 0 \\ 0 & \tilde{P}_{k+1} \end{bmatrix} \begin{bmatrix} \rho_{k+1} \\ \beta_{k+1} \end{bmatrix}$

(28)

where $a_k, b_k, c_k, d_k, \bar{a}_k, \bar{b}_k, \bar{c}_k$ and $\bar{d}_k$ are defined in section 3. From (27), (28), and (29), with $d_0 = 1$ we have
From the definitions of $F_k$, $\bar{F}_k$, $R_k$, and $z_k$, we have

$$R_k^T F_k = \bar{R}_k F_k = z_k = [I_k \circ 0_{k,\infty}] \vec{\rho} \bar{M}_k E_1 \bar{A}_1.$$  

(30)

Therefore by induction, we know that $\bar{\tau}_i = \bar{\beta}_i$ for $i = 1,2,...,k-1$. From (24), we see that at iteration $k$, the first $k-1$ blocks of $\bar{B}_k$ and $\bar{F}_k$ are equal.

References


