2013

# An Augmented LSQR Method

James Baglama
*University of Rhode Island*, jbaglama@uri.edu

L. Reichel

***See next page for additional authors***



Citation/Publisher Attribution
Baglama, J., Rechel, L. & Richmond, D. (2013). An Augmented LSQR Method. *Numerical Algorithms*, 64(2), 263-293.
Available at http://link.springer.com/article/10.1007%2Fs11075-012-9665-8

# AN AUGMENTED LSQR METHOD

J. BAGLAMA*, L. REICHEL†, AND D. RICHMOND‡

**Abstract.** The LSQR iterative method for solving least-squares problems may require many iterations to determine an approximate solution with desired accuracy. This often depends on the fact that singular vector components of the solution associated with small singular values of the matrix require many iterations to be determined. Augmentation of Krylov subspaces with harmonic Ritz vectors often makes it possible to determine the singular vectors associated with small singular values with fewer iterations than without augmentation. This paper describes how Krylov subspaces generated by the LSQR iterative method can be conveniently augmented with harmonic Ritz vectors. Computed examples illustrate the competitiveness of the augmented LSQR method proposed.

**Key words.** Partial singular value decomposition, iterative method, large-scale computation, least-squares approximation, LSQR, precondition, Krylov subspace, augmentation

**AMS subject classifications.** 65F15, 15A18

## 1. Introduction.

We consider the solution of least-squares (LS) problems

$$\min_{x\in\mathbb{R}^n} \|Ax-b\|, \tag{1.1}$$

where $A \in \mathbb{R}^{\ell\times n}$ is a large sparse matrix with $\ell \geq n$ and $b \in \mathbb{R}^\ell$. Throughout, $\|\cdot\|$ denotes the Euclidean vector norm or the associated induced matrix norm. The matrix $A$ is assumed to be too large to be factored. We therefore seek to solve (1.1) by an iterative method. Unless stated otherwise, $A$ is assumed to have full column rank. Problem (1.1) then has a unique solution, which we denote by $x^+$. The associated residual vector $r^+ = b - Ax^+$ vanishes if and only if $b$ lies in the range of $A$, denoted by $\mathcal{R}(A)$.

Many iterative methods have been proposed for the solution of (1.1); see, e.g., [5, 8, 10, 11, 12, 26, 27] and references therein. A popular method is LSQR by Paige and Saunders [26]. This method does not require the matrix $A$ to be stored; instead each iteration requires that one matrix-vector product with $A$ and one matrix-vector product with $A^T$ be evaluated. A mathematically, but not numerically, equivalent method is CGLS proposed by Björck; see, e.g., [5] for a discussion of CGLS.

LSQR [26] is based on Golub-Kahan (GK) bidiagonalization of $A$. Let $x_0$ be an initial approximate solution of (1.1) and define $r_0 = b - Ax_0$. Generically, $m \ll \min\{\ell, n\}$ steps of GK bidiagonalization determine orthonormal bases $\{q_1, q_2, \ldots, q_m\}$ and $\{p_1, p_2, \ldots, p_m\}$ for the Krylov subspaces

$$\begin{aligned} \mathcal{K}_m(AA^T, q_1) &= \mathrm{span}\{q_1, AA^Tq_1, (AA^T)^2q_1, \ldots, (AA^T)^{m-1}q_1\} \\ \mathcal{K}_m(A^TA, p_1) &= \mathrm{span}\{p_1, A^TAp_1, (A^TA)^2p_1, \ldots, (A^TA)^{m-1}p_1\} \end{aligned} \tag{1.2}$$

respectively, with initial vectors $q_1 = r_0/\|r_0\|$ and $p_1 = A^Tq_1/\|A^Tq_1\|$. LSQR computes an approximate solution $x_m$ of (1.1) by minimizing $\|Ax - b\|$ over the set

[0]Version October 30, 2012

*Department of Mathematics, University of Rhode Island, Kingston, RI 02881. E-mail: jbaglama@math.uri.edu. Home page: http://www.math.uri.edu/∼jbaglama

†Department of Mathematical Sciences, Kent State University, Kent, OH 44242. E-mail: reichel@math.kent.edu. Home page: http://www.math.kent.edu/∼reichel

‡Department of Mathematics, University of Rhode Island, Kingston, RI 02881. E-mail: dan@math.uri.edu. Home page: http://www.math.uri.edu/∼dan

$x_0 + \mathcal{K}_m(A^TA, p_1)$. The associated residual vector $r_m = b - Ax_m$ lies in $\mathcal{K}_m(AA^T, q_1)$; see [26] or section 4 for details.

GK bidiagonalization, and therefore also LSQR, will in exact arithmetic terminate before $m$ steps have been carried out if the Krylov subspace $\mathcal{K}_m(A^TA, p_1)$ is of dimension less than $m$. LSQR delivers, in this situation, the solution of (1.1). However, early termination is rare and it is common for LSQR to require many iterations before an approximation of the solution $x^+$ of (1.1) of desired accuracy has been determined. The rate of convergence of LSQR depends on the condition number of $A$ and on the distribution of the singular values of the matrix; convergence may be slow when $A$ has a large condition number; see [5] or section 2 for details.

The rate of convergence of LSQR can be improved by using a preconditioner. Instead of solving (1.1), one may solve the right-preconditioned LS problem

$$\min_{y\in\mathbb{R}^n} \|AMy - b\|. \tag{1.3}$$

The preconditioner $M \in \mathbb{R}^{n\times n}$ should be nonsingular and such that i) the condition number of $AM$ is smaller than the condition number of $A$, or $AM$ has improved clustering of its singular values, and ii) matrix-vector products with the matrices $M$ and $M^T$ can be evaluated fairly quickly; see, e.g., [4, 5, 6, 11, 16, 28] and references therein for several approaches to constructing preconditioners. Many such preconditioners are constructed prior to solution of the LS problem, and their determination may require significant computational effort and storage. Preconditioners affect the Krylov subspaces in which approximate solutions are determined. We describe another approach for modifying Krylov subspaces in which approximate solutions are computed. Specifically, we determine approximations of singular vectors of $A$ associated with the smallest singular values and augment the Krylov subspaces (1.2) by these vectors. This augmentation is carried out while improved approximate solutions of (1.1) are computed, and changes the Krylov subspaces to improve convergence. Our method can be used in conjunction with a preconditioner.

The idea of augmenting a Krylov subspace with vectors to improve convergence was first discussed by Morgan [21], who considered the solution of linear systems of equations with a square nonsingular matrix by GMRES. Morgan proposed to augment the Krylov subspaces used by GMRES with harmonic Ritz vectors associated with the Ritz values of smallest magnitude to increase the rate of convergence. Subsequently, Morgan showed in [22, 23] that the residual vectors associated with the harmonic Ritz vectors are multiples of the residual vector at every restart of the (standard) GMRES method and that, therefore, the augmented Krylov subspace is a Krylov subspace generated by a different starting vector. This result suggested that the augmenting vectors should be chosen to be harmonic Ritz vectors.

The initial iterations of our augmentation method for LSQR is analogous to Morgan's augmented method for GMRES [23] in that we augment the Krylov subspaces (1.2) with harmonic Ritz vectors for $AA^T$ and associated vectors for $A^TA$. During the initial iterations with LSQR, we compute both improved approximations of the solution of (1.1) and improved approximations to harmonic Ritz vectors. When the latter approximations are deemed accurate enough, we stop updating these vectors and carry out LSQR iterations using augmented Krylov subspaces until a solution of (1.1) with desired accuracy has been found; the solution subspaces are augmented with fixed harmonic Ritz vectors.

Section 2 discusses convergence of LSQR when the Krylov subspaces (1.2) are augmented with singular vectors of $A$ associated with the smallest singular values.

These singular vectors generally are not explicitly known. We therefore describe in section 3 how approximations of these vectors can be computed by a restarted GK bidiagonalization method, which is augmented by harmonic Ritz vectors of $AA^T$ associated with the smallest harmonic Ritz values and with related vectors for $A^TA$. The method is related to a scheme described in [1], but differs in certain design aspects to fit better with the restarted LSQR method described in section 4. In section 5 we show that all residual vectors of the harmonic Ritz vectors are multiples of the residual vector of the restarted LSQR method. This result is important for the design of our augmented LSQR method. It implies that the augmented Krylov subspaces also are Krylov subspaces. Moreover, section 5 describes our augmented LSQR method. Application of this algorithm to LS problems (1.1) with a rank-deficient matrix $A$ is discussed in section 6. A few numerical examples are presented in section 7 and concluding remarks can be found in section 8.

We would like to emphasize that the proposed iterative method is not a restarted LSQR method. Restarting may lead to stagnation; see [10, Section 7.3.1] for remarks on restarting the related LSMR method. Our method consists of two stages: i) the augmenting stage, which uses restarted LSQR to approximate the singular vectors associated with the smallest singular values of $A$ and simultaneously improve an available approximation of the solution of (1.1), and ii) the LSQR stage, in which LSQR is applied using the augmented Krylov subspaces with fixed harmonic Ritz vectors to solve the LS problem (1.3).

**2. Convergence of LSQR using augmented Krylov subspaces.** Let $u_i$ and $v_i$ denote the left and right singular vectors of $A$ associated with the singular value $\sigma_i$. Define $U_n = [u_1, u_2, \ldots, u_n] \in \mathbb{R}^{\ell\times n}$ and $V_n = [v_1, v_2, \ldots, v_n] \in \mathbb{R}^{n\times n}$ with orthonormal columns, as well as $\Sigma_n = \mathrm{diag}[\sigma_1, \sigma_2, \ldots, \sigma_n] \in \mathbb{R}^{n\times n}$. Then

$$AV_n = U_n\Sigma_n \qquad \text{and} \qquad A^TU_n = V_n\Sigma_n \tag{2.1}$$

are singular value decompositions (SVDs) of $A$ and $A^T$, respectively. We assume the singular values to be ordered from the smallest to the largest one, i.e.,

$$0 < \sigma_1 \leq \sigma_2 \leq \ldots \leq \sigma_n.$$

While this ordering is nonstandard, it simplifies the notation in the sequel. The condition number of $A$ is given by $\kappa(A) = \sigma_n/\sigma_1$.

The residual $r_m = b - Ax_m$ associated with the $m^{\text{th}}$ iterate, $x_m$, determined by LSQR with initial approximate solution $x_0$ satisfies

$$\|r_m - r^+\| \leq 2\left(\frac{\sigma_n - \sigma_1}{\sigma_n + \sigma_1}\right)^m \|r_0 - r^+\| = 2\left(\frac{\kappa(A) - 1}{\kappa(A) + 1}\right)^m \|r_0 - r^+\|, \tag{2.2}$$

where $x^+$ denotes the solution of (1.1) and $r^+$ is the corresponding residual; see [5]. Furthermore, if $b \in \mathcal{R}(A)$, then

$$\|r_m\| \leq 2\left(\frac{\sigma_n - \sigma_1}{\sigma_n + \sigma_1}\right)^m \|r_0\|.$$

For well-conditioned LS problems, LSQR converges quickly. However, ill-conditioned problems may require a prohibitively large number of iterations. The use of a preconditioner $M$ with $\kappa(AM) \ll \kappa(A)$ may alleviate this difficulty.

We first describe how augmentation of the Krylov subspaces (1.2) by singular vectors of $A$ associated with the smallest singular values reduces the bound (2.2) and therefore can be expected to speed up convergence. Thus, consider the augmented Krylov subspaces

$$\begin{aligned} \mathcal{K}_m(AA^T, u_1, \ldots, u_k, q_1) &= \text{span}\{u_1, \ldots, u_k, q_1, AA^T q_1, \ldots, (AA^T)^{m-k-1} q_1\} \\ \mathcal{K}_m(A^TA, v_1, \ldots, v_k, p_1) &= \text{span}\{v_1, \ldots, v_k, p_1, A^TAp_1, \ldots, (A^TA)^{m-k-1} p_1\} \end{aligned} \tag{2.3}$$

obtained by augmenting the Krylov subspace $\mathcal{K}_{m-k}(AA^T, q_1)$ by the left singular vectors $u_1, \ldots, u_k$ associated with the $k$ smallest singular values, and by augmenting $\mathcal{K}_{m-k}(A^TA, p_1)$ by the corresponding right singular vectors $v_1, \ldots, v_k$. At iteration $m$, the augmented method determines an approximate solution in a subspace of at most dimension $m$. The following result shows that the upper bound for the residual error (2.2) may be reduced considerably by augmentation.

THEOREM 2.1. *Let $A \in \mathbb{R}^{\ell \times n}$ have the SVD (2.1) and let $x_m$ minimize $\|Ax - b\|$ over the augmented and shifted Krylov subspace $x_0 + \mathcal{K}_m(A^TA, v_1, \ldots, v_k, p_1)$. Then with $r_m = b - Ax_m$,*

$$\|r_m - r^+\| \leq 2\left(\frac{\sigma_n - \sigma_{k+1}}{\sigma_n + \sigma_{k+1}}\right)^{m-k} \|r_0 - r^+\|.$$

*Proof.* Let $x_m$ be *any* vector from $x_0 + \mathbb{K}_m(A^TA, v_1, \ldots, v_k, p_1)$ and define $r_m = b - Ax_m$. Then

$$x_m = x_0 + \sum_{i=1}^{k} \tau_i v_i + \phi(A^TA)A^T r_0, \tag{2.4}$$

where $\phi$ is a polynomial of degree at most $m - k - 1$ and $\tau_i \in \mathbb{R}$. Let $P_{\mathcal{R}(A)}$ and $P_{\mathcal{N}(A^T)}$ denote the orthogonal projectors onto the range of $A$ and the null space of $A^T$, respectively. Split the vector $b$ according to

$$b = P_{\mathcal{R}(A)} b + P_{\mathcal{N}(A^T)} b = \sum_{i=1}^{n} \omega_i u_i + P_{\mathcal{N}(A^T)} b,$$

where the $u_i$ are the left singular vectors of $A$; cf. (2.1). Then

$$A^T r_0 = A^T b - A^T A x_0 = \sum_{i=1}^{n} \omega_i A^T u_i - A^T A x_0 = \sum_{i=1}^{n} \tilde{\omega}_i v_i \tag{2.5}$$

since $\{v_1, \ldots, v_n\}$ is an orthonormal basis for $\mathbb{R}^n$. Using (2.4) and (2.5) we obtain

$$A^T r_m = \psi(A^TA)A^T r_0 - \sum_{i=1}^{k} \tau_i \sigma_i^2 v_i = \sum_{i=1}^{n} \tilde{\omega}_i \psi(\sigma_i^2) v_i - \sum_{i=1}^{k} \tau_i \sigma_i^2 v_i, \tag{2.6}$$

where $\psi(x) = 1 - x\phi(x)$. Let $\gamma_i = -\tau_i \sigma_i^2 + \tilde{\omega}_i \psi(\sigma_i^2)$. Then

$$A^T r_m = \sum_{i=1}^{k} \gamma_i v_i + \sum_{i=k+1}^{n} \tilde{\omega}_i \psi(\sigma_i^2) v_i.$$

We may now choose $\tau_i = \frac{\tilde{\omega}_i \psi(\sigma_i^2)}{\sigma_i^2}$ to define $x_m$ in (2.4). This yields $\gamma_i = 0$ and, therefore,

$$A^T r_m = \sum_{i=k+1}^{n} \tilde{\omega}_i \psi(\sigma_i^2) v_i. \tag{2.7}$$

Now let $\psi$ be the shifted Chebyshev polynomial of degree $m-k-1$ for the interval $[\sigma_{k+1}^2, \sigma_n^2]$, scaled so that $\psi(0) = 1$, and take the $(A^TA)^{-1}$ norm of both sides of (2.7). Using properties of the scaled and shifted Chebyshev polynomial, we obtain

$$\|A^T r_m\|_{(A^TA)^{-1}} \leq 2\left(\frac{\sigma_n - \sigma_{k+1}}{\sigma_n + \sigma_{k+1}}\right)^{m-k} \|A^T r_0\|_{(A^TA)^{-1}}.$$

The desired result follows from the observations that

$$\|A^T r_m\|_{(A^TA)^{-1}} = \|r_m - r^+\| \tag{2.8}$$

and that the norm of the residual vector $r_m = b - Ax_m$ associated with the vector $x_m$ in the statement of the theorem is at least as small as the norm obtained for our choices of $\tau$ and $\psi$. □

Morgan [21] discussed the use of augmented Krylov subspaces of the form $\text{span}\{b, Ab, \ldots, A^{m-1}b, z_1, \ldots z_k\}$, where $z_1, \ldots, z_k$ are eigenvectors of $A$, to increase the rate of convergence of restarted GMRES, and showed a result analogous to Theorem 2.1 for this situation.

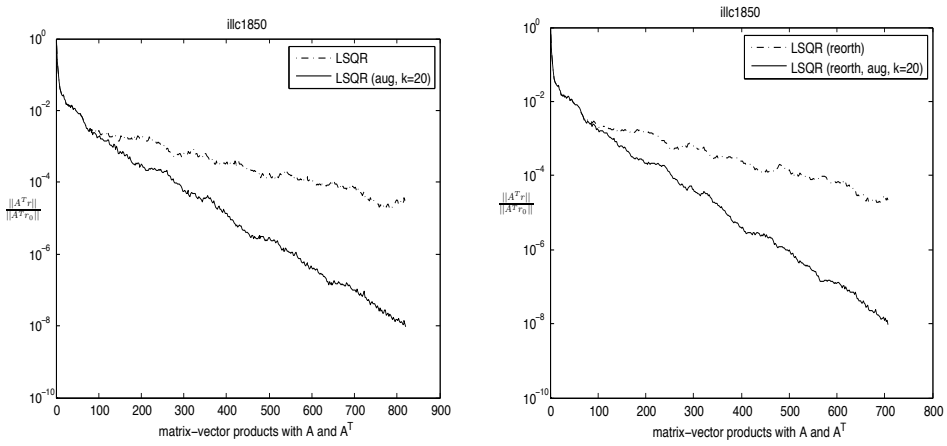


FIG. 2.1. *Example 2.1: A comparison of augmented and standard LSQR.*

*Example 2.1.* Let $A \in \mathbb{R}^{1850\times 721}$ be the matrix ILLC1850 and let $b$ be the vector ILLC1850_RHS1 from the LSQ set of the Matrix Market Collection [7, 9]. Figure 2.1 compares the augmented LSQR method using (2.3) with $k = 20$ and the standard LSQR method, with $x_0 = 0$ for both methods. Figure 2.1 displays the convergence of the quotients $\|A^T r\|/\|A^T r_0\|$ as a function of the number of matrix-vector products with $A$ and $A^T$. Here $r_0 = b$ is the residual associated with the initial iterate $x_0$, and $r$ is the residual associated with the currently available iterate. The right graphs show

implementations of the methods with full reorthogonalization, while the left graphs display the performance of the methods without reorthogonalization. In this case, we see that reorthogonalization does not change the convergence behavior much, but that augmentation as described in Theorem 2.1 may increase the rate of convergence significantly.

The initial vector $q_1$ for the Krylov subspace in the augmented Krylov subspace $\mathcal{K}_m(AA^T, u_1, \ldots, u_{20}, q_1)$ is orthogonalized against the $k = 20$ left singular vectors $\{u_1, \ldots, u_{20}\}$. This makes the vector $p_1 = A^T q_1/\|A^T q_1\|$ in the augmented Krylov subspace $\mathcal{K}_m(A^T A, v_1, \ldots, v_k, p_1)$ orthogonal to the right singular vectors $\{v_1, \ldots, v_{20}\}$. □

The singular vectors $\{u_1, \ldots, u_k\}$ and $\{v_1, \ldots, v_k\}$ associated with the $k$ smallest singular values of $A$ are generally not explicitly known. We therefore seek to determine approximations of these vectors while simultaneously computing improved approximations of the solution of (1.1). This is achieved with a restarted LSQR method. Typically augmenting vectors do not have to be accurate approximations of the singular vectors of $A$ to yield beneficial results. This is illustrated by the following theorem as well as by numerical examples in section 7. The theorem is an analog of a result by Morgan [21], concerned with augmenting a Krylov subspace by approximate eigenvectors to increase the rate of convergence of restarted GMRES.

THEOREM 2.2. *Let $A \in \mathbb{R}^{\ell\times n}$ have the SVD (2.1) and let $x_m$ minimize $\|Ax - b\|$ over the augmented and shifted Krylov subspace $x_0 + \mathcal{K}_m(A^T A, y_1, p_1)$, where the unit-length vector $y_1 \in \mathbb{R}^n$ is an approximation of the right singular vector $v_1$. Let $\zeta$ be the angle between $y_1$ and $v_1$, and let $\tilde{\omega}_1$ be defined in (2.5) from Theorem 2.1. Then with $r_m = b - Ax_m$,*

$$\|r_m - r^+\| \le 2\left(\frac{\sigma_n - \sigma_2}{\sigma_n + \sigma_2}\right)^{m-1} \|r_0 - r^+\| + \frac{\|A^T A\|}{\sigma_1^2} \tan(\zeta)|\tilde{\omega}_1|. \tag{2.9}$$

*Proof.* Similarly to (2.4) and (2.6) we have

$$\begin{aligned} x_m &= x_0 + \tau_1 y_1 + \phi(A^T A)A^T r_0, \\ A^T r_m &= \sum_{i=1}^{n} \tilde{\omega}_i \psi(\sigma_i^2) v_i - \tau_1 A^T A y_1, \end{aligned} \tag{2.10}$$

where $\phi(x)$ is a polynomial of degree at most $m - 2$ and $\psi(x) = 1 - x\phi(x)$ is a polynomial of degree at most $m - 1$. Let

$$y_1 = \cos(\zeta)v_1 + \sin(\zeta)z, \tag{2.11}$$

where $z \in \text{span}\{v_2, \ldots, v_n\}$ is a unit-length vector. Using (2.11) and the SVD of $A$, equation (2.10) becomes

$$A^T r_m = \sum_{i=1}^{n} \tilde{\omega}_i \psi(\sigma_i^2) v_i - \tau_1 \sigma_1^2 v_1 \cos(\zeta) - \tau_1 A^T A z \sin(\zeta).$$

With $\tau_1 = \frac{\tilde{\omega}_1 \psi(\sigma_1^2)}{\sigma_1^2 \cos(\zeta)}$, we obtain

$$A^T r_m = \sum_{i=2}^{n} \tilde{\omega}_i \psi(\sigma_i^2) v_i - \frac{\tilde{\omega}_1 \psi(\sigma_1^2) A^T A z \tan(\zeta)}{\sigma_1^2}. \tag{2.12}$$

Let $\psi$ be the shifted Chebyshev polynomial for the interval $[\sigma_2^2, \sigma_n^2]$, scaled so that $\psi(0) = 1$, and take the $(A^TA)^{-1}$ norm of both sides of (2.12). Using properties of the shifted and scaled Chebyshev polynomials, we get

$$\|A^T r_m\|_{(A^TA)^{-1}} \leq \Big(\frac{\sigma_n - \sigma_2}{\sigma_n + \sigma_2}\Big)^{m-1} \|A^T r_0\|_{(A^TA)^{-1}} + \frac{\|A^TA\|}{\sigma_1^2} \tan(\zeta)|\tilde{\omega}_1|.$$

The theorem now follows from (2.8). □

We remark that the right-hand side of (2.9) shows that if the smallest singular value $\sigma_1$ is very close to zero or to $\sigma_2$, then $y_1$ has to be a fairly accurate approximation of the singular vector $v_1$ in order to be effective.

**3. A restarted augmented GK bidiagonalization method.** This section describes a restarted GK bidiagonalization method for approximating the singular triplets $\{\sigma_i, u_i, v_i\}_{i=1}^k$ associated with the $k$ smallest singular values of $A$. We refer to these singular triplets as the $k$ smallest singular triplets. Let the matrices $U_k \in \mathbb{R}^{\ell\times k}$ and $V_k \in \mathbb{R}^{n\times k}$ consist of the first $k$ columns of the matrices $U_n$ and $V_n$ in the SVD (2.1) of $A$, and introduce $\Sigma_k = \mathrm{diag}[\sigma_1, \ldots, \sigma_k] \in \mathbb{R}^{k\times k}$. Then, analogously to (2.1), we have the partial SVDs

$$AV_k = U_k\Sigma_k \qquad \text{and} \qquad A^TU_k = V_k\Sigma_k.$$

There are numerous methods available for computing approximations of the singular triplets $\{\sigma_i, u_i, v_i\}_{i=1}^k$; see, e.g., [1, 2, 3, 12, 14, 15, 17, 18, 19] and references therein. We are interested in using a method that is related to LSQR, so that while computing these approximations, we also can determine improved approximate solutions of (1.1). Therefore, we will use a restarted augmented harmonic GK bidiagonalization method to determine approximations of the desired singular triplets. We show in section 4 why this approach is attractive.

The restarted augmented harmonic GK bidiagonalization method of this paper is closely related to the method presented in [1]; it differs in that here we use a lower bidiagonal matrix. This makes it easier to connect our method to LSQR. The following algorithm describes the computations required for partial GK bidiagonalization. We comment on the algorithm below.

ALGORITHM 3.1. A PARTIAL GK BIDIAGONALIZATION ALGORITHM

*Input:* $A \in \mathbb{R}^{\ell\times n}$ *or functions for evaluating products with* $A$ *and* $A^T$,
$q_1 \in \mathbb{R}^\ell$ : *initial vector,*
$m$ : *number of bidiagonalization steps.*

*Output:* $P_m = [p_1, \ldots, p_m] \in \mathbb{R}^{n\times m}$ : *matrix with orthonormal columns,*
$Q_{m+1} = [q_1, \ldots, q_{m+1}] \in \mathbb{R}^{\ell\times(m+1)}$ : *matrix with orthonormal columns,*
$B_{m+1,m} \in \mathbb{R}^{(m+1)\times m}$ : *lower bidiagonal matrix (3.2),*
$p_{m+1} \in \mathbb{R}^n$ : *residual vector,*
$\alpha_{m+1} \in \mathbb{R}$.

1. *Compute* $\beta_1 := \|q_1\|$; $q_1 := q_1/\beta_1$; $Q_1 := q_1$
2. *Compute* $p_1 := A^Tq_1$; $\alpha_1 := \|p_1\|$; $p_1 := p_1/\alpha_1$; $P_1 := p_1$
3. *For* $j = 1 : m$
   4. *Compute* $q_{j+1} := Ap_j - q_j\alpha_j$

*5. Reorthogonalize:* $q_{j+1} := q_{j+1} - Q_{(1:j)}(Q_{(1:j)}^T q_{j+1})$
*6. Compute* $\beta_{j+1} := \|q_{j+1}\|; \quad q_{j+1} := q_{j+1}/\beta_{j+1}; \quad Q_{j+1} := [Q_j, q_{j+1}]$
*7. Compute* $p_{j+1} := A^T q_{j+1} - p_j \beta_{j+1}$
*8. Reorthogonalize:* $p_{j+1} := p_{j+1} - P_{(1:j)}(P_{(1:j)}^T p_{j+1})$
*9. Compute* $\alpha_{j+1} := \|p_{j+1}\|; \quad p_{j+1} := p_{j+1}/\alpha_{j+1}$
*10. if* $j < m$
*11.* $P_{j+1} := [P_j, p_{j+1}]$
*12. End*
*13. End*

To avoid loss of orthogonality due to finite precision arithmetic, we reorthogonalize in lines 5 and 8 of the algorithm; see section 5 for a few remarks on reorthogonalization in the context of GK bidiagonalization.

A matrix interpretation of the computations of Algorithm 3.1 shows that the algorithm determines the decompositions

$$\begin{aligned} AP_m &= Q_{m+1}B_{m+1,m}, \\ A^T Q_{m+1} &= P_m B_{m+1,m}^T + \alpha_{m+1} p_{m+1} e_{m+1}^T, \end{aligned} \tag{3.1}$$

where the matrices $P_m = [p_1, \ldots, p_m] \in \mathbb{R}^{n\times m}$ and $Q_{m+1} = [q_1, \ldots, q_{m+1}] \in \mathbb{R}^{\ell\times(m+1)}$ have orthonormal columns, the residual vector $p_{m+1} \in \mathbb{R}^n$ satisfies $P_m^T p_{m+1} = 0$, and $e_{m+1}$ is the $(m+1)^{\text{st}}$ axis vector of appropriate dimension. The matrix

$$B_{m+1,m} = \begin{bmatrix} \alpha_1 & & & & 0 \\ \beta_2 & \alpha_2 & & & \\ & \beta_3 & \alpha_3 & & \\ & & \ddots & \ddots & \\ & & & & \alpha_m \\ 0 & & & & \beta_{m+1} \end{bmatrix} \in \mathbb{R}^{(m+1)\times m} \tag{3.2}$$

is lower bidiagonal. We refer to (3.1) as a partial GK bidiagonalization of $A$. The number of bidiagonalization steps $m \ll \min\{\ell, n\}$ is assumed to be small enough so that the partial bidiagonalization (3.1) with the stated properties exists. We assume in the following that Algorithm 3.1 does not terminate early, i.e., that all $\alpha_j > 0$ and $\beta_j > 0$ for $1 \le j \le m+1$. Early termination will be commented on in section 5.

The decompositions (3.1) are closely related to partial Lanczos tridiagonalization of $A^TA$ and $AA^T$. For instance, multiplying the first equation in (3.1) by $A^T$ yields the partial Lanczos tridiagonalization of $A^TA$,

$$A^T A P_m = P_m B_{m+1,m}^T B_{m+1,m} + (\alpha_{m+1}\beta_{m+1}) p_{m+1} e_m^T. \tag{3.3}$$

Analogously, multiplying the second equation in (3.1) by $A$ gives

$$AA^T Q_{m+1} = Q_{m+1} B_{m+1,m} B_{m+1,m}^T + \alpha_{m+1} A p_{m+1} e_{m+1}^T,$$

and then equating the first $m$ columns yields the partial Lanczos tridiagonalization of $AA^T$,

$$AA^T Q_m = Q_m B_m B_m^T + \alpha_m \beta_{m+1} q_{m+1} e_m^T, \tag{3.4}$$

where $B_m$ is the leading $m \times m$ principal submatrix of $B_{m+1,m}$, $Q_m \in \mathbb{R}^{\ell \times m}$ consists of the first $m$ columns of the matrix $Q_{m+1}$, and $q_{m+1}$ is the last column of $Q_{m+1}$.

The LSQR method is started or restarted with Krylov subspaces of the form (1.2). We therefore consider the decomposition (3.4) for determining harmonic Ritz vectors. The harmonic Ritz values $\hat{\theta}_j$ of $AA^T$ determined by (3.4) are the eigenvalues $\hat{\theta}_j$ of the generalized eigenvalue problem

$$((B_m B_m^T) + \alpha_m^2 \beta_{m+1}^2 (B_m B_m^T)^{-1} e_m e_m^T)\tilde{g}_j = \hat{\theta}_j \tilde{g}_j, \quad 1 \leq j \leq m, \tag{3.5}$$

where $\tilde{g}_j \in \mathbb{R}^m \backslash \{0\}$ is an eigenvector; see, e.g., [20, 25] for properties of and discussions on harmonic Ritz values.

The eigenpairs $\{\hat{\theta}_j, \tilde{g}_j\}_{j=1}^m$ of (3.5) can be computed without forming the matrix $B_m B_m^T$. Instead, determine the SVD of $B_{m+1,m}$, which satisfies

$$\begin{aligned} B_{m+1,m}\tilde{V}_m &= \left[\tilde{U}_{m+1,m}\ \tilde{u}_{m+1}\right] \begin{bmatrix} \tilde{\Sigma}_m \\ 0 \end{bmatrix}, \\ B_{m+1,m}^T \left[\tilde{U}_{m+1,m}\ \tilde{u}_{m+1}\right] &= \tilde{V}_m \left[\tilde{\Sigma}_m \quad 0\right], \end{aligned} \tag{3.6}$$

where the matrices $\tilde{V}_m = [\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_m] \in \mathbb{R}^{m \times m}$ and $\tilde{U}_{m+1,m} = [\tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_m] \in \mathbb{R}^{(m+1)\times m}$ have orthonormal columns, $\tilde{u}_{m+1} \in \mathbb{R}^{m+1}$ is a unit-length vector such that $\tilde{u}_{m+1}^T \tilde{U}_{m+1,m} = 0$, and $\tilde{\Sigma}_m = \mathrm{diag}[\tilde{\sigma}_1, \tilde{\sigma}_2, \ldots, \tilde{\sigma}_m] \in \mathbb{R}^{m \times m}$. We order the $m$ singular values according to

$$0 < \tilde{\sigma}_1 \leq \tilde{\sigma}_2 \leq \ldots \leq \tilde{\sigma}_m.$$

The vector $\tilde{u}_{m+1}$ lies in $\mathcal{N}(B_{m+1,m}^T)$ and we will refer to it as the null space vector of $B_{m+1,m}^T$.

Consider the $(m+1) \times (m+1)$ symmetric tridiagonal matrix

$$B_{m+1,m} B_{m+1,m}^T = \left[\begin{array}{c|c} B_m B_m^T & \alpha_m \beta_{m+1} e_m \\ \hline \alpha_m \beta_{m+1} e_m^T & \beta_{m+1}^2 \end{array}\right].$$

The $m$ nonvanishing eigenvalues of this matrix are harmonic Ritz values, i.e., they are the eigenvalues of (3.5). We have $\hat{\theta}_j = \tilde{\sigma}_j^2$; see [25]. The harmonic Ritz vectors of $AA^T$ can be computed by using the matrix

$$S = \begin{bmatrix} I_m & \alpha_m \beta_{m+1} (B_m B_m^T)^{-1} e_m \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} I_m & \beta_{m+1} B_m^{-T} e_m \\ 0 & 1 \end{bmatrix}$$

and noticing that

$$S B_{m+1,m} B_{m+1,m}^T S^{-1} = \left[\begin{array}{c|c} B_m B_m^T + \alpha_m^2 \beta_{m+1}^2 (B_m B_m^T)^{-1} e_m e_m^T & \begin{matrix} 0 \\ \vdots \end{matrix} \\ \hline \alpha_m \beta_{m+1} e_m^T & 0 \end{array}\right].$$

Thus, the first $m$ rows of $S\tilde{U}_{m+1,m}$ are the eigenvectors in (3.5), i.e.,

$$[\tilde{g}_1, \tilde{g}_2, \ldots, \tilde{g}_m] = \left[I_m \quad \beta_{m+1} B_m^{-T} e_m\right] \tilde{U}_{m+1,m}.$$

It follows that a harmonic Ritz vector of $AA^T$ associated with the harmonic Ritz value $\hat{\theta}_j$ is given by

$$\hat{u}_j := Q_m \tilde{g}_j. \tag{3.7}$$

Morgan [22] pointed out that the residual vectors associated with different harmonic Ritz pairs $\{\hat{\theta}_j, \hat{u}_j\}$ are parallel in the context of the Arnoldi process and GMRES. We show this result for the problem at hand, because this property is central for our augmentation method. Using (3.4), (3.5), and (3.7), we obtain

$$\begin{aligned}
AA^T\hat{u}_j - \hat{\theta}_j\hat{u}_j &= AA^T Q_m\tilde{g}_j - \hat{\theta}_j Q_m\tilde{g}_j \\
&= (Q_m B_m B_m^T + \alpha_m\beta_{m+1}q_{m+1}e_{m+1}^T)\tilde{g}_j - \hat{\theta}_j Q_m\tilde{g}_j \\
&= Q_m(B_m B_m^T - \hat{\theta}_j I_m)\tilde{g}_j + \alpha_m\beta_{m+1}q_{m+1}e_m^T\tilde{g}_j \\
&= Q_m(-(\alpha_m\beta_{m+1})^2(B_m B_m^T)^{-1}e_m e_m^T)\tilde{g}_j + \alpha_m\beta_{m+1}q_{m+1}e_m^T\tilde{g}_j \\
&= (\alpha_m\beta_{m+1}e_m^T\tilde{g}_j)Q_{m+1}\begin{bmatrix} -\alpha_m\beta_{m+1}(B_m B_m^T)^{-1}e_m \\ 1 \end{bmatrix} \\
&= (\alpha_m\beta_{m+1}e_m^T\tilde{g}_j)Q_{m+1}\begin{bmatrix} -\beta_{m+1}B_m^{-T}e_m \\ 1 \end{bmatrix}.
\end{aligned}$$

This shows that all the residuals for the harmonic Ritz pairs for $AA^T$ are multiples of the same vector.

Define the residual vector for the harmonic Ritz pairs,

$$r_m^{\text{harm}} = Q_{m+1}\begin{bmatrix} -\beta_{m+1}B_m^{-T}e_m \\ 1 \end{bmatrix} \tag{3.8}$$

and assume that we are interested in the $k$ smallest singular triplets. Our augmentation process can now be described by considering the starting matrix

$$\left[\hat{u}_1, \ldots, \hat{u}_k, r_m^{\text{harm}}\right] = Q_{m+1}\begin{bmatrix} \left[I_m \quad \beta_{m+1}B_m^{-T}e_m\right]\tilde{U}_{m+1,k} & -\beta_{m+1}B_m^{-T}e_m \\ 0 & 1 \end{bmatrix}. \tag{3.9}$$

The columns of the matrix in (3.9) are not orthogonal. We therefore compute its QR decomposition

$$\begin{bmatrix} \left[I_m \quad \beta_{m+1}B_m^{-T}e_m\right]\tilde{U}_{m+1,k} & -\beta_{m+1}B_m^{-T}e_m \\ 0 & 1 \end{bmatrix} = \tilde{Q}\tilde{R}, \tag{3.10}$$

where $\tilde{Q} \in \mathbb{R}^{(m+1)\times(k+1)}$ has orthonormal columns and $\tilde{R} \in \mathbb{R}^{(k+1)\times(k+1)}$ is upper triangular, and use

$$\hat{Q}_{k+1} = Q_{m+1}\tilde{Q} \tag{3.11}$$

as starting matrix. Application of (3.1), (3.6), (3.8), and (3.10) yields

$$A^T\hat{Q}_{k+1} = A^T Q_{m+1}\tilde{Q} = \left[P_m\tilde{V}_k\tilde{\Sigma}_k \quad A^T r_m^{\text{harm}}\right]\tilde{R}^{-1}, \tag{3.12}$$

where $\tilde{V}_k = [\tilde{v}_1, \ldots, \tilde{v}_k]$ and $\tilde{\Sigma}_k = \mathrm{diag}[\tilde{\sigma}_1, \tilde{\sigma}_2, \ldots, \tilde{\sigma}_k]$.

The relation

$$A^T r_m^{\mathrm{harm}} = \alpha_{m+1} p_{m+1} \tag{3.13}$$

can be shown by using

$$A^T \hat{Q}_{k+1} = (P_m B_{m+1,m}^T + \alpha_{m+1} p_{m+1} e_{m+1}^T)\tilde{Q} \tag{3.14}$$

and by equating the right-hand sides of (3.12) and (3.14) and applying (3.10). Therefore, we have

$$\begin{aligned} A^T \hat{Q}_{k+1} &= \begin{bmatrix} P_m \tilde{V}_k & p_{m+1} \end{bmatrix} \begin{bmatrix} \tilde{\sigma}_1 & & & & 0 \\ & \tilde{\sigma}_2 & & & \\ & & \ddots & & \\ & & & \tilde{\sigma}_k & \\ 0 & & & & \alpha_{m+1} \end{bmatrix} \tilde{R}^{-1} \\ &= \hat{P}_k(\tilde{\Sigma}_k \tilde{R}_{k,k+1}^{-1}) + \frac{\alpha_{m+1}}{\tilde{r}_{k+1,k+1}} p_{m+1} e_{k+1}^T, \end{aligned} \tag{3.15}$$

where

$$\hat{P}_k = P_m \tilde{V}_k, \tag{3.16}$$

the matrix $\tilde{R}_{k,k+1}^{-1}$ is the leading $k \times (k+1)$ submatrix of $\tilde{R}^{-1}$, and $\tilde{r}_{k+1,k+1}$ is the $(k+1)^{\mathrm{st}}$ diagonal entry of $\tilde{R}$. It follows from the structure of the matrix on the left-hand side of (3.10) that $1/\tilde{r}_{k+1,k+1} = \tilde{q}_{m+1,k+1}$, the $(m+1, k+1)$-element of the matrix $\tilde{Q}$. It follows from $\hat{P}_k^T p_{m+1} = 0$ that

$$\hat{P}_k^T A^T \hat{Q}_{k+1} = \tilde{\Sigma}_k \tilde{R}_{k,k+1}^{-1}. \tag{3.17}$$

The decomposition (3.15) is important for the derivation of our iterative method; it is analogous to the second decomposition in (3.1).

We now derive a decomposition for $A\hat{P}_k$ that is analogous to the first decomposition in (3.1). Using (3.1), (3.6), and (3.16), we obtain

$$A\hat{P}_k = Q_{m+1} \tilde{U}_{m+1,k} \tilde{\Sigma}_k. \tag{3.18}$$

This gives

$$B_{m+1,m}^T = B_m^T [I_m \quad \beta_{m+1} B_m^{-T} e_m],$$

and from (3.6) it follows that

$$\begin{bmatrix} I_m & \beta_{m+1} B_m^{-T} e_m \end{bmatrix} \tilde{U}_{m+1,k} = B_m^{-T} \tilde{V}_k \tilde{\Sigma}_k \tag{3.19}$$

and therefore

$$\tilde{U}_{m+1,k} = \begin{bmatrix} B_m^{-T} \tilde{V}_k \tilde{\Sigma}_k & -\beta_{m+1} B_m^{-T} e_m \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I_k \\ e_{m+1}^T \tilde{U}_{m+1,k} \end{bmatrix}. \tag{3.20}$$

We obtain from (3.10), (3.19), and (3.20) that

$$\tilde{U}_{m+1,k} = \tilde{Q}\tilde{Q}^T\tilde{U}_{m+1,k}, \tag{3.21}$$

and inserting (3.21) into (3.18) yields

$$A\hat{P}_k = Q_{m+1}\tilde{Q}\tilde{Q}^T\tilde{U}_{m+1,k}\tilde{\Sigma}_k = \hat{Q}_{k+1}\tilde{Q}^T\tilde{U}_{m+1,k}\tilde{\Sigma}_k. \tag{3.22}$$

Now using (3.17) and (3.22), we get

$$\hat{Q}_{k+1}^T A\hat{P}_k = \tilde{Q}^T\tilde{U}_{m+1,k}\tilde{\Sigma}_k = (\tilde{\Sigma}_k\tilde{R}_{k,k+1}^{-1})^T. \tag{3.23}$$

Let

$$\hat{B}_{k+1,k} = \tilde{Q}^T\tilde{U}_{m+1,k}\tilde{\Sigma}_k, \tag{3.24}$$

$$\hat{\alpha}_{k+1} = \alpha_{m+1}\tilde{q}_{m+1,k+1}. \tag{3.25}$$

Then from (3.15) and (3.22)–(3.25), we obtain

$$\begin{aligned} A\hat{P}_k &= \hat{Q}_{k+1}\hat{B}_{k+1,k}, \\ A^T\hat{Q}_{k+1} &= \hat{P}_k\hat{B}_{k+1,k}^T + \hat{\alpha}_{k+1}\hat{p}_{k+1}e_{k+1}^T, \end{aligned} \tag{3.26}$$

where $\hat{p}_{k+1} = p_{m+1}$ and $\hat{B}_{k+1,k} \in \mathbb{R}^{(k+1)\times k}$ is lower triangular. This is the desired analogue of (3.1).

Starting with (3.26), computations with GK bidiagonalization can be continued using Algorithm 3.1 with $\hat{q}_{k+1}$, the $(k+1)^{\text{st}}$ column of $\hat{Q}_{k+1}$. Application of $m-k$ steps of GK bidiagonalization yields the new decompositions

$$\begin{aligned} A^T[\hat{Q}_{k+1} \;\; \hat{Q}_{m-k}] &= [\hat{P}_k \;\; \hat{P}_{m-k}]\hat{B}_{m+1,m}^T + \hat{\alpha}_{m+1}\hat{p}_{m+1}e_{m+1}^T, \\ A[\hat{P}_k \;\; \hat{P}_{m-k}] &= [\hat{Q}_{k+1} \;\; \hat{Q}_{m-k}]\hat{B}_{m+1,m}, \end{aligned} \tag{3.27}$$

where the first column of $\hat{P}_{m-k}$ is $\hat{p}_{k+1}$,

$$\hat{B}_{m+1,m} = \begin{bmatrix} \hat{B}_{k+1,k} & \hat{\alpha}_{k+1} & & & 0 \\ & \hat{\beta}_{k+2} & \hat{\alpha}_{k+2} & & \\ & & \ddots & \ddots & \\ & & & & \hat{\alpha}_m \\ 0 & & & & \hat{\beta}_{m+1} \end{bmatrix} \in \mathbb{R}^{(m+1)\times m}, \tag{3.28}$$

and the matrices $[\hat{Q}_{k+1} \;\; \hat{Q}_{m-k}] \in \mathbb{R}^{\ell\times(m+1)}$ and $[\hat{P}_k \;\; \hat{P}_{m-k}] \in \mathbb{R}^{n\times m}$ have orthonormal columns. We now proceed by computing the SVD of $\hat{B}_{m+1,m}$, harmonic Ritz vectors of $AA^T$, cf. (3.7), and then new decompositions analogous to (3.26) and (3.27). The $k$ smallest singular triplets

$$\{\tilde{\sigma}_j, \hat{q}_j, \hat{p}_j\}_{j=1}^k, \tag{3.29}$$

where $\hat{q}_j$, $j = 1, \ldots, k$, are the first $k$ columns of $\hat{Q}_{k+1}$ and the $\hat{p}_j$, $j = 1, \ldots, k$, are the first $k$ columns of $\hat{P}_k$, furnish approximations of the $k$ smallest singular triplets $\{\sigma_j, u_j, v_j\}_{j=1}^k$ of $A$.

A singular triplet $\{\tilde{\sigma}_j, \hat{q}_j, \hat{p}_j\}$ defined by (3.29) is accepted as an approximate singular triplet of $A$ if

$$\begin{aligned}
&\sqrt{\|A\hat{p}_j - \tilde{\sigma}_j\hat{q}_j\|^2 + \|A^T\hat{q}_j - \tilde{\sigma}_j\hat{p}_i\|^2} \\
&\qquad = \sqrt{\tilde{\sigma}_j^2\|\tilde{u}_j - \tilde{q}_j\|^2 + \|B_{m+1,m}^T\tilde{q}_j - \tilde{\sigma}_j\tilde{v}_j\|^2 + |\alpha_{m+1}e_{m+1}^T\tilde{q}_j|^2} \\
&\qquad \leq \delta^{\text{harm}}\|A\|,
\end{aligned} \tag{3.30}$$

where $\tilde{q}_j$ is the $j^{\text{th}}$ column of $\tilde{Q}$ from (3.10), $\tilde{u}_j$ and $\tilde{v}_j$ are the $j^{\text{th}}$ columns of $\tilde{U}_{m+1,m}$ and $\tilde{V}_m$ respectively in the SVD (3.6) of $\hat{B}_{m,m+1}$, and $\delta^{\text{harm}} > 0$ is a user-specified tolerance. In (3.30) $\|A\|$ can be approximated by $\tilde{\sigma}_m$, the largest singular value of $\hat{B}_{m+1,m}$. Typically, several matrices $\hat{B}_{m+1,m}$ are generated during the iterations and therefore an acceptable approximation of $\|A\|$ can be obtained from the largest singular value of all the matrices $\hat{B}_{m+1,m}$ generated.

We remark that accurate computation of the vector $B_m^{-T}e_m$, used in (3.10), might be difficult when $B_m$ has a large condition number. This computation can be avoided by noticing that the vector

$$\begin{bmatrix} -\beta_{m+1}B_m^{-T}e_m \\ 1 \end{bmatrix} \tag{3.31}$$

is in the null space of $[I_m \quad \beta_{m+1}B_m^{-T}e_m] \in \mathbb{R}^{m\times(m+1)}$, and

$$B_{m+1,m}^T = B_m^T\left[I_m \quad \beta_{m+1}B_m^{-T}e_m\right].$$

Therefore, the vector (3.31) is a multiple of the null space vector $\tilde{u}_{m+1}$ of $B_{m+1,m}^T$, cf. (3.6). We have

$$\begin{bmatrix} -\beta_{m+1}B_m^{-T}e_m \\ 1 \end{bmatrix} = (1/\tilde{u}_{m+1,m+1})\tilde{u}_{m+1}, \tag{3.32}$$

where $\tilde{u}_{m+1,m+1}$ is the last element of the vector $\tilde{u}_{m+1}$. It follows that any multiple of the matrix

$$\left[\begin{array}{c|c} [\tilde{u}_{m+1,m+1}I_m \quad -\tilde{u}_{m+1,1:m}]\,\tilde{U}_{m+1,k} & \tilde{u}_{m+1} \\ \hline 0 & \end{array}\right] \tag{3.33}$$

can be used in place of the left-hand side of (3.10). Here $\tilde{u}_{m+1,1:m}$ denotes the vector consisting of the first $m$ elements of $\tilde{u}_{m+1}$.

The restarted GK bidiagonalization method described above will be combined with the restarted LSQR method reviewed in the following section.

**4. A restarted LSQR method.** We describe a restarted LSQR method for solving the LS problem (1.1). The method will be used in conjunction with the restarted GK bidiagonalization method for computing harmonic Ritz vectors presented in the previous section. The description of our restarted LSQR method parallels as much as possible that of the standard LSQR method [26].

Application of $k$ steps of Algorithm 3.1 with starting vector $q_1 \in \mathbb{R}^\ell$ yields the decompositions

$$\begin{aligned}
AP_k &= Q_{k+1}B_{k+1,k}, \\
A^TQ_{k+1} &= P_kB_{k+1,k}^T + \alpha_{k+1}p_{k+1}e_{k+1}^T.
\end{aligned} \tag{4.1}$$

Let $r_k = b - Ax_k$ for some vector $x_k \in \mathbb{R}^n$ such that $r_k = Q_{k+1} f_{k+1}$ for some $f_{k+1} \in \mathbb{R}^{k+1}$; if $k = 0$, then we let $r_0 = q_1 f_1$ where $f_1 = \|r_0\|$.

Extend the $k$ step decompositions (4.1) by carrying out $m - k$ additional GK bidiagonalization steps to obtain $m$ step decompositions (3.1). Let $x_m = x_k + P_m y_m$ and notice that

$$\begin{aligned} r_m = b - Ax_m &= b - A(x_k + P_m y_m) \\ &= r_k - AP_m y_m \\ &= r_k - Q_{m+1} B_{m+1,m} y_m \\ &= Q_{m+1}\left( \begin{bmatrix} f_{k+1} \\ 0 \end{bmatrix} - B_{m+1,m} y_m \right). \end{aligned}$$

It follows that

$$\min_{x_m \in x_k + \mathcal{K}_m(A^T A, p_1)} \|b - Ax_m\| = \min_{y \in \mathbb{R}^m} \left\| \begin{bmatrix} f_{k+1} \\ 0 \end{bmatrix} - B_{m+1,m} y \right\|. \tag{4.2}$$

We solve (4.2) with the aid of the QR decomposition

$$B_{m+1,m} = \tilde{Q}^{(B)}_{m+1} \tilde{R}^{(B)}_{m+1,m}, \tag{4.3}$$

where $\tilde{Q}^{(B)}_{m+1} \in \mathbb{R}^{(m+1)\times(m+1)}$ is orthogonal and $\tilde{R}^{(B)}_{m+1,m} \in \mathbb{R}^{(m+1)\times m}$ is upper triangular. Substituting (4.3) into (4.2) yields the equivalent minimization problem

$$\min_{y \in \mathbb{R}^m} \left\| (\tilde{Q}^{(B)}_{m+1})^T \begin{bmatrix} f_{k+1} \\ 0 \end{bmatrix} - \tilde{R}^{(B)}_{m+1,m} y \right\|. \tag{4.4}$$

Since the last row of $\tilde{R}^{(B)}_{m+1,m}$ vanishes, the LS solution $y_m$ of (4.4) satisfies the first $m$ rows exactly. The residual norm for (4.4) is given by

$$\bar{\phi}_{m+1} = e^T_{m+1} (\tilde{Q}^{(B)}_{m+1})^T \begin{bmatrix} f_{k+1} \\ 0 \end{bmatrix}.$$

This yields the residual vector for the LSQR method

$$\begin{aligned} r^{\mathrm{lsqr}}_m &= b - Ax_m \\ &= Q_{m+1}\left( \begin{bmatrix} f_{k+1} \\ 0 \end{bmatrix} - B_{m+1,m} y_m \right) \\ &= Q_{m+1} \tilde{Q}^{(B)}_{m+1} \left( (\tilde{Q}^{(B)}_{m+1})^T \begin{bmatrix} f_{k+1} \\ 0 \end{bmatrix} - \tilde{R}^{(B)}_{m+1,m} y_m \right) \\ &= Q_{m+1} \bar{\phi}_{m+1} \tilde{Q}^{(B)}_{m+1} e_{m+1}. \end{aligned} \tag{4.5}$$

The process can be restarted with the vectors $x_k = x_m$ and $r_k = r^{\mathrm{lsqr}}_m$, where we again assume that $r_k$ is a linear combination of the columns of the matrix $Q_{k+1}$ in (4.1). Section 5 shows how this condition can be guaranteed.

There are several ways to compute the QR decomposition in (4.3). In the context of the restarted GK bidiagonalization method of section 3, the first $k + 1$ rows and $k$

columns of $\hat{B}_{m+1,m}$ in (3.28) is the matrix $\hat{B}_{k+1,k}$ in (3.24), which is lower triangular and typically not lower bidiagonal. We compute a QR decomposition of $\hat{B}_{k+1,k}$ by an arbitrary method and then switch to using Givens rotations when carrying out $m-k$ GK bidiagonalization steps to produce the bottom part of the matrix $\hat{B}_{m+1,m}$. This approach allows our algorithm to incorporate all of the formulas, e.g., for computing residual norms, of the standard LSQR algorithm [26] from step $k+1$ and onwards.

The following algorithm describes our restarted LSQR method, where we assume that the starting residual vector $r_k$ is in $\mathcal{R}(Q_{k+1})$. The algorithm uses the elegant formulas of the LSQR method by Paige and Saunders [26] whenever possible to reduce the computational cost and storage requirements. We comment further on the algorithm below.

ALGORITHM 4.1. A RESTARTED LSQR METHOD

*Input:* $A \in \mathbb{R}^{\ell \times n}$ *or functions for evaluating products with* $A$ *and* $A^T$,
$k$*-step GK bidiagonalization decomposition (4.1),*
$x_k \in \mathbb{R}^n$ : *initial approximate solution of (1.1),*
$f_{k+1} \in \mathbb{R}^{k+1}$ : *where* $r_k = b - Ax_k = Q_{k+1} f_{k+1}$, $Q_{k+1}$ *is given in (4.1),*
$m \geq k+2$ : *maximum number of iterations,*
$m_{\text{reorth}}$ : *maximum number of vectors for reorthogonalization in steps 25 and 28,*
$\delta^{\text{lsqr}}$ : *tolerance for accepting an approximate solution to (1.1).*

*Output: Approximate solution* $x_m$ *to (1.1),*
*(optional)* $\bar{\phi}_{m+1}$, $c_m$, *and* $m$*-step GK bidiagonalization (3.1).*

1. *If* $k = 0$
   2. *Compute* $q_1 := r_0/f_1$; $\quad Q_1 := q_1$
   3. *Compute* $p_1 := A^T q_1$; $\quad \alpha_1 := \|p_1\|$; $\quad p_1 := p_1/\alpha_1$; $\quad P_1 := p_1$
   4. *Set* $B_{1,0} := [\,]$
5. *End*
6. *Compute* $q_{k+2} := Ap_{k+1} - q_{k+1}\alpha_{k+1}$
7. *Reorthogonalize:* $q_{k+2} := q_{k+2} - Q_{(1:k+1)}(Q_{(1:k+1)}^T q_{k+2})$
8. *Compute* $\beta_{k+2} := \|q_{k+2}\|$; $\quad q_{k+2} := q_{k+2}/\beta_{k+2}$; $\quad Q_{k+2} := [Q_{k+1}, q_{k+2}]$
9. *Compute* $p_{k+2} := A^T q_{k+2} - p_{k+1}\beta_{k+2}$
10. *Reorthogonalize:* $p_{k+2} := p_{k+2} - P_{(1:k+1)}(P_{(1:k+1)}^T p_{k+2})$
11. *Compute* $\alpha_{k+2} := \|p_{k+2}\|$; $\quad p_{k+2} := p_{k+2}/\alpha_{k+2}$; $\quad P_{k+2} := [P_{k+1}, p_{k+2}]$
12. *Compute QR decomposition* $B_{k+2,k+1} = \widetilde{Q}\widetilde{R}$ *of*

$$B_{k+2,k+1} := \begin{bmatrix} B_{k+1,k} & \alpha_{k+1} \\ 0 & \beta_{k+2} \end{bmatrix} \in \mathbb{R}^{(k+2)\times(k+1)},$$

*where* $\widetilde{Q} \in \mathbb{R}^{(k+2)\times(k+2)}$ *and* $\widetilde{R} \in \mathbb{R}^{(k+2)\times(k+1)}$

13. *Compute* $\tilde{f}_{k+2} := \widetilde{Q}^T \begin{bmatrix} f_{k+1} \\ 0 \end{bmatrix}$
14. *Compute* $\bar{\rho}_{k+2} := \alpha_{k+2}(e_{k+2}^T \widetilde{Q} e_{k+2})$
15. *Compute* $\bar{\phi}_{k+2} := e_{k+2}^T \tilde{f}_{k+2}$
16. *Solve* $\widetilde{R}_{k+1,k+1} y = \tilde{f}_{1:k+1}$, *where* $\widetilde{R}_{k+1,k+1} \in \mathbb{R}^{(k+1)\times(k+1)}$ *is the*

*leading submatrix of* $\widetilde{R}$

17. *Update solution vector* $x_{k+1} := x_k + P_{(1:k+1)}y$
18. *Compute* $\|r_{k+1}\| := |\bar{\phi}_{k+2}|$
19. *Compute* $\|A^T r_{k+1}\| := \alpha_{k+2}\beta_{k+2}|e_{k+1}^T y|$
20. *Check convergence: if (4.6) is satisfied, then exit.*
21. *Compute* $\theta_{k+2} := e_{k+1}^T \widetilde{Q}^T \begin{bmatrix} B_{k+2,k+1} \\ 0 \quad \alpha_{k+2} \end{bmatrix} e_{k+2}$
22. *Compute* $w := p_{k+2} - P_{(1:k+1)}y(\theta_{k+2}/f_{k+1,k+1})$
23. *For* $j = k+2 : m$
    24. *Compute* $q_{j+1} := Ap_j - q_j\alpha_j$
    25. *Reorthogonalize:*
        *Compute* $i := \max\{1, j - m_{\text{reorth}} + 1\}$
        *Compute* $q_{j+1} := q_{j+1} - Q_{(i:j)}(Q_{(i:j)}^T q_{j+1})$
    26. *Compute* $\beta_{j+1} := \|q_{j+1}\|; \quad q_{j+1} := q_{j+1}/\beta_{j+1}; \quad Q_{j+1} := [Q_j, q_{j+1}];$
    27. *Compute* $p_{j+1} := A^T q_{j+1} - p_j\beta_{j+1}$
    28. *Reorthogonalize:*
        *Compute* $i := \max\{1, j - m_{\text{reorth}} + 1\}$
        *Compute* $p_{j+1} := p_{j+1} - P_{(i:j)}(P_{(i:j)}^T p_{j+1})$
    29. *Compute* $\alpha_{j+1} := \|p_{j+1}\|; \quad p_{j+1} := p_{j+1}/\alpha_{j+1}$
    30. *if* $j < m$
        31. $P_{j+1} := [P_j, p_{j+1}]$
    32. *End*
    33. *Compute* $\rho_j := \sqrt{\beta_{j+1}^2 + \bar{\rho}_j^2}$; $c_j := \bar{\rho}_j/\rho_j$; $s_j := \beta_{j+1}/\rho_j$
    34. *Compute* $\theta_j := s_j\alpha_{j+1}$
    35. *Compute* $\bar{\rho}_{j+1} := -c_j\alpha_{j+1}$
    36. *Compute* $\phi_j := c_j\bar{\phi}_j; \quad \bar{\phi}_{j+1} := s_j\bar{\phi}_j$
    37. *Compute* $x_j := x_{j-1} + (\phi_j/\rho_j)w; \quad w := p_{j+1} - (\theta_{j+1}/\rho_j)w$
    38. *Compute* $\|r_j\| := |\bar{\phi}_{j+1}|$
    39. *Compute* $\|A^T r_j\| := |\bar{\phi}_{j+1}\bar{\rho}_{j+1}|$
    40. *Check convergence: if (4.6) is satisfied, then exit.*
41. *End*

When $k = 0$ on input to Algorithm 4.1 and no reorthogonalization and accumulation of the matrices $B_{m+1,m}$, $P_m$, and $Q_{m+1}$ is carried out, $m$ steps of the algorithm are equivalent to $m$ steps of the LSQR method of Paige and Saunders [26]. In particular, Algorithm 4.1 can be used as a restarted or nonrestarted LSQR method.

The stopping criteria outlined in [10, 26] can be used in the convergence tests (lines 20 and 40) of Algorithm 4.1. This is recommend for public domain implementations of the algorithm. For ease of comparison with other methods, we terminate the computations in the examples reported in section 7 when in lines 20 or 40 the inequality

$$\|A^T r_j\| \le \delta^{\text{lsqr}} \|A^T r_0\| \tag{4.6}$$

holds, where $\delta^{\text{lsqr}} > 0$ is a user-specified tolerance.

The formula for $\|r_{k+1}\|$ in line 18 follows from (4.5), and the expression for $\|A^T r_{k+1}\|$ in line 19 is taken from Jia [13]. The formulas for $\|r_j\|$ and $\|A^T r_j\|$ in lines 35 and 36, respectively, are obtained from [26]. If $\alpha_{j+1} = 0$ or $\beta_{j+1} = 0$ for some $j$, then $\|A^T r_j\| = 0$; see [24] and more recently [13, Theorem 2].

We reorthogonalize in lines 25 and 28 of Algorithm 4.1 to avoid loss of orthogonality due to finite precision arithmetic. Reorthogonalization requires the accumulation of the matrices $Q_{(i:j)}$ in line 25 and $P_{(i:j)}$ in line 28. Both these matrices have a fixed maximum number of columns, denoted by $m_{\rm reorth}$. Several reorthogonalization strategies are discussed in [1, 18, 29]. When $\ell \gg n$, reorthogonalization of the columns of $P_{(i:j)}$ only, reduces the computational effort required to compute the decompositions (3.1) considerably, compared with reorthogonalization of the columns of both the matrices $P_{(i:j)}$ and $Q_{(i:j)}$. We refer to reorthogonalization of the columns of $P_{(i:j)}$ only as one-sided reorthogonalization. Algorithm 4.1 can easily be modified to implement one-sided reorthogonalization; see [1, 29] for discussions on this reorthogonalization approach.

We are interested in combining Algorithm 4.1 with the augmented harmonic GK bidiagonalization method of section 3. In this context, we assume that $m \ll \min\{\ell, n\}$ and apply one-sided reorthogonalization as described in [1] and applied in the MATLAB code `irlba` accompanying [2]. When, instead, Algorithm 4.1 is used as a non-restarted LSQR algorithm, either no reorthogonalization is carried out or only the last generated $m_{\rm reorth}$ columns of $P_{(i:j)}$ are reorthogonalized. The latter reorthogonalization approach also is implemented by Fong and Saunders [10] in their MATLAB code `lsmr`. Reorthogonalization in lines 7 and 10 of Algorithm 4.1 is always carried out when $k > 0$. Moreover, when $k > 0$ we use a $k$-step GK bidiagonalization (4.1) as input. To be able to apply the formulas of the LSQR algorithm [26], we carry out the $(k+1)^{\rm st}$ step of GK bidiagonalization separately, i.e., we perform the computations of lines 6–11 of Algorithm 4.1, and subsequently determine the quantities $\bar{\rho}_{k+2}$ in line 14, $\bar{\phi}_{k+2}$ in line 15, $\theta_{k+2}$ in line 21, and $w$ in line 22 by formulas analogous to [26, equations (4.6)–(4.12)].

Line 12 of Algorithm 4.1 computes the QR decomposition of the matrix $B_{k+2,k+1}$. This can be done with MATLAB's internal `qr` function. The input restriction $m \geq k+2$ ensures that the For-loop (lines 23–38) is executed at least once. Typically, $k$ is quite small; in the computed examples of section 7, we let $k \leq 20$.

**5. An augmented LSQR algorithm.** In order to be able to conveniently combine the restarted LSQR method of Section 4 with the restarted augmented GK bidiagonalization method of section 3, the residual vector from restarted LSQR, $r_m^{\rm lsqr}$ in (4.5), should be in the range of the matrix $\hat{Q}_{k+1}$ defined in (3.11). We now show that the residual vector $r_m^{\rm harm}$ of the harmonic Ritz vectors, defined by (3.8), and $r_m^{\rm lsqr}$ are parallel. It then follows from (3.8)–(3.11) that $r_m^{\rm lsqr} \in \mathcal{R}(\hat{Q}_{k+1})$.

THEOREM 5.1. *The residual vector of the harmonic Ritz vectors $r_m^{\rm harm}$, defined by (3.8), and the residual vector of the restarted LSQR method $r_m^{\rm lsqr}$, given by (4.5), are parallel provided that the lower bidiagonal matrix $B_{m+1,m}$ (3.2) from GK bidiagonalization (3.1) is unreduced. Moreover, $r_m^{\rm harm}$ and $r_m^{\rm lsqr}$ are multiples of $Q_{m+1}\tilde{u}_{m+1}$, where $\tilde{u}_{m+1} \in \mathcal{N}(B_{m+1,m}^T)$, cf. (3.6).*

*Proof.* Consider the $(m+1)$-vector

$$\tilde{Q}_{m+1}^{(B)} \bar{\phi}_{m+1} e_{m+1} \tag{5.1}$$

of $r_m^{\text{lsqr}}$ and note that this vector is in $\mathcal{N}(B_{m+1,m}^T)$, i.e.,

$$\begin{aligned} B_{m+1,m}^T \tilde{Q}_{m+1}^{(B)} \bar{\phi}_{m+1} e_{m+1} &= \bar{\phi}_{m+1} (e_{m+1}^T (\tilde{Q}_{m+1}^{(B)})^T B_{m+1,m})^T \\ &= \bar{\phi}_{m+1} (e_{m+1}^T \tilde{R}_{m+1,m}^{(B)})^T \\ &= 0. \end{aligned} \tag{5.2}$$

It is easy to see that the $(m+1)$-vector

$$\begin{bmatrix} -\beta_{m+1} B_m^{-T} e_m \\ 1 \end{bmatrix} \tag{5.3}$$

in the definition (3.8) of $r_m^{\text{harm}}$ lies in $\mathcal{N}(B_{m+1,m}^T)$:

$$\begin{bmatrix} B_m^T & \beta_{m+1} e_m \end{bmatrix} \begin{bmatrix} -\beta_{m+1} B_m^{-T} e_m \\ 1 \end{bmatrix} = 0. \tag{5.4}$$

The matrix $B_{m+1,m}$ is unreduced by assumption. Therefore, it has rank $m$ and so does its transpose $B_{m+1,m}^T$. Equations (5.2) and (5.4) show that the vectors

$$\tilde{Q}_{m+1}^{(B)} \bar{\phi}_{m+1} e_{m+1} \quad \text{and} \quad \begin{bmatrix} -\beta_{m+1} B_m^{-T} e_m \\ 1 \end{bmatrix}$$

are in $\mathcal{N}(B_{m+1,m}^T)$. It follows that they are multiples of each other and of the vector $\tilde{u}_{m+1}$ defined in (3.6). □

We can easily determine the scalar multiplier between $r_m^{\text{harm}}$ (3.8) and $r_m^{\text{lsqr}}$ (4.5) by examining the For-loop (lines 23–38) in Algorithm 4.1. LSQR eliminates the subdiagonal element of the lower bidiagonal matrix via Givens rotations, but does not explicitly form the orthogonal matrix made up by the products of these rotations. If this matrix were generated, then in the last iteration (lines 23–41) of Algorithm 4.1, we would obtain

$$\tilde{Q}_{m+1}^{(B)} := \begin{bmatrix} I_{m-1} & 0 \\ 0 & \begin{bmatrix} c_m & s_m \\ s_m & -c_m \end{bmatrix} \end{bmatrix} \begin{bmatrix} \tilde{Q}_m^{(B)} & 0 \\ 0 & 1 \end{bmatrix}, \tag{5.5}$$

where $\tilde{Q}_m^{(B)} \in \mathbb{R}^{m \times m}$ is the orthogonal matrix from the QR factorization of $B_{m,m-1}$. It follows from (5.5) that the last element of the vector (5.1) is $-c_m \bar{\phi}_{m+1}$. Moreover, the last element of the vector (5.3) is one. Therefore,

$$r_m^{\text{lsqr}} = -c_m \bar{\phi}_{m+1} r_m^{\text{harm}}.$$

Using (3.32), we also have that

$$\begin{aligned} \tilde{Q}_{m+1}^{(B)} \bar{\phi}_{m+1} e_{m+1} &= -c_m \bar{\phi}_{m+1} \begin{bmatrix} -\beta_{m+1} B_m^{-T} e_m \\ 1 \end{bmatrix} \\ &= (-c_m \bar{\phi}_{m+1} / \tilde{u}_{m+1,m+1}) \tilde{u}_{m+1}. \end{aligned}$$

If $\tilde{Q}$ is the matrix with orthonormal columns in the QR decomposition of (3.33), then

$$r_m^{\text{lsqr}} = \hat{Q}_{k+1} f_{k+1},$$

where $f_{k+1} = (-c_m \bar{\phi}_{m+1}/\tilde{u}_{m+1,m+1})\tilde{Q}^T \tilde{u}_{m+1}$.

We are now in a position to describe our augmented LSQR algorithm that combines the methods of sections 3 and 4. We assume that augmentation is carried out with vectors that approximate the singular vectors associated with the smallest singular values.

ALGORITHM 5.2. AN AUGMENTED LSQR METHOD

*Input:* $A \in \mathbb{R}^{\ell \times n}$ *or functions for evaluating products with* $A$ *and* $A^T$,
$x_0 \in \mathbb{R}^n$ : *initial approximate solution of (1.1),*
$r_0 := b - Ax_0 \in \mathbb{R}^\ell$ : *initial residual vector,*
$k$ : *number of augmenting vectors,*
$m \geq k+2$ : *maximum length GK bidiagonalization,*
$max_{\text{aug}}$: *maximum number of iteration for augmenting stage,*
$max_{\text{lsqr}}$: *maximum number of iteration for the non-restarted LSQR method,*
$\delta^{\text{lsqr}}$ : *tolerance for accepting an approximate solution to (1.1).*
$\delta^{\text{harm}}$ : *tolerance for accepting computed approximate singular triplet, cf. (3.30),*

*Output: Approximate solution* $x$ *to (1.1).*

1. *Call Algorithm 4.1*
   *Input:* $A$, $k := 0$, $x_0$, $f_1 := \|r_0\|$, $q_1 := r_0/f_1$, $m_{\text{reorth}} := m$, $m$ *and* $\delta^{\text{lsqr}}$
   *Output:* $x_m$, $\bar{\phi}_{m+1}$, $c_m$, *and an* $m$*-step GK bidiagonalization (3.1)*
2. *For* $i = 1 : max_{\text{aug}}$
   3. *Compute the singular value decomposition (3.6) of* $B_{m+1,m}$
   4. *Compute the augmenting vectors:*
      *Compute the QR factorization of (3.33).*
      *Determine the matrices* $\hat{Q}_{k+1}$, $\hat{P}_k$, *and* $\hat{B}_{k+1,k}$ *by (3.11), (3.16) and (3.24), respectively and* $\hat{\alpha}_{k+1}$ *by (3.25) to get (3.26).*
   5. *Check convergence: if all* $k$ *desired singular triplets satisfy (3.30), then goto 9.*
   6. *Call Algorithm 4.1*
      *Input:* $A$, $x_k := x_m$, $f_{k+1} := (-c_m \bar{\phi}_{m+1}/\tilde{u}_{m+1,m+1})\tilde{Q}^T \tilde{u}_{m+1}$, $m_{\text{reorth}} := m$, $m$, $\delta^{\text{lsqr}}$, *and a* $k$*-step GK bidiagonalization (3.26)*
      *Output:* $x_m$, $\bar{\phi}_{m+1}$, $c_m$, *and an* $m$*-step GK bidiagonalization (3.27)*
   7. *Set*

$$\begin{aligned} B_{m+1} &:= \hat{B}_{m+1,m} \\ Q_{m+1} &:= [\hat{Q}_{k+1} \quad \hat{Q}_{m-k}] \\ P_m &:= [\hat{P}_k \quad \hat{P}_{m-k}] \\ p_{m+1} &:= \hat{p}_{m+1} \\ \alpha_{m+1} &:= \hat{\alpha}_{m+1} \end{aligned}$$

8. *End*
9. *Call Algorithm 4.1*
   *Input:* $A$, $x_k := x_m$, $f_{k+1} := (-c_m \bar{\phi}_{m+1}/\tilde{u}_{m+1,m+1})\tilde{Q}^T \tilde{u}_{m+1}$, $m_{\text{reorth}} := m$, $m := max_{\text{lsqr}}$, $\delta^{\text{lsqr}}$ *and a* $k$*-step GK bidiagonalization (3.26)*

*Output:* $x_m$

The above algorithm describes a simplification of the actual computations carried out. For instance, the number of augmenting vectors used at each restart is typically chosen to be larger than the number of desired singular triplets. This often yields faster convergence without increasing the memory requirement; see [1, 2] for a discussion. The number of vectors to be reorthogonalized, $m_{\rm reorth}$, is set to the maximum number of columns of the computed GK bidiagonalization. This is to ensure that accurate approximations of the singular vectors are computed.

In the nonrestarted LSQR stage of Algorithm 5.2, i.e., in line 9, the reorthogonalization applied is that of the nonrestarted LSQR method described by Algorithm 4.1. We set $m_{\rm reorth} = m$. Letting $0 \leq m_{\rm reorth} < m$ instead would reduce the computational work for each iteration, but could require more iterations to satisfy the convergence criterion and, therefore, may require more computational effort in total. The choice $m_{\rm reorth} > m$ increases the storage requirement and therefore is avoided.

**6. Rank-deficient LS problems.** A least-squares problem (1.1) is said to be rank-deficient if $A$ has linearly dependent columns. We are interested in determining the unique solution, $x^+$, of minimal Euclidean norm. This solution is orthogonal to $\mathcal{N}(A)$ and therefore lies in $\mathcal{R}(A^T)$; see, e.g., [5] for a discussion on rank-deficient LS problems.

The standard LSQR algorithm [26] produces a sequence of iterates that lie in $\mathcal{R}(A^T)$ provided the initial iterate $x_0$ does. To ensure the latter one may choose $x_0 = 0$. Note that the iterates determined in lines 17 and 34 of Algorithm 4.1 are in $\mathcal{R}(A^T)$ if the initial approximation $x_k$ of $x^+$ used in Algorithm 4.1 is in $\mathcal{R}(A^T)$. In order to show that the approximate solutions determined by Algorithm 5.2 are in $\mathcal{R}(A^T)$ when this holds for the first iterate $x_0$, it remains to establish that the harmonic Ritz vectors used to augment the Krylov subspace in Algorithm 5.2 also lie in $\mathcal{R}(A^T)$. Observe that the restarted augmented harmonic method of section 3 does not determine approximations of eigenvectors associated with the eigenvalue zero. The reason for this is that the harmonic Ritz values are the square of the nonvanishing singular values of $B_{m+1,m}$ (3.2). The singular values are nonvanishing, since by assumption all $\alpha_j$ and $\beta_j$ are nonzero. The situation when some $\alpha_j$ or $\beta_j$ vanish is discussed in section 4.

The iterations with the augmented Krylov subspaces of Algorithm 5.2 determine approximate solutions $x_m$ of (1.1) in subspaces of the form

$$\mathcal{K}_m(A^TA, \hat{p}_1, \ldots, \hat{p}_k, \hat{p}_{k+1}) = \mathrm{span}\{\hat{p}_1, \ldots, \hat{p}_k, \hat{p}_{k+1}, A^TA\hat{p}_{k+1}, \ldots, (A^TA)^{m-k-1}\hat{p}_{k+1}\},$$

where $\hat{p}_1, \ldots, \hat{p}_k$ are approximate right singular vectors of $A$ associated with nonvanishing singular values, and $\hat{p}_{k+1} = p_{m+1}$ is the residual vector of GK bidiagonalization (3.1); see also Algorithm 3.1. Using (3.3) and (3.13), we have for $j \leq k$,

$$\begin{aligned}\hat{p}_j &= \frac{1}{\tilde{\sigma}_j^2}(A^TA\hat{p}_j - (\beta_{m+1}e_m^T\tilde{v}_k)\alpha_{m+1}p_{m+1}) \\ &= \frac{1}{\tilde{\sigma}_j^2}A^T(A\hat{p}_j - (\beta_{m+1}e_m^T\tilde{v}_k)r_m^{\rm harm}).\end{aligned}$$

It follows that $\mathcal{K}_m(A^TA, \hat{p}_1, \ldots, \hat{p}_k, \hat{p}_{k+1}) \subset \mathcal{R}(A^T)$. Example 7.6 in section 7 illustrates the performance of Algorithm 5.2 when applied to a rank-deficient LS problem.

**7. Numerical examples.** We describe a few numerical experiments that illustrate the performance of Algorithm 5.2 as implemented by the MATLAB code `alsqr`[1]. This code uses the following user-specified parameters:

| | |
|---|---|
| *adjust* | Additional vectors used together with $k$ augmenting vectors to speed up convergence; see [1] for comments on the inclusion of additional vectors. |
| $k$ | Number of augmenting vectors. |
| *maxitp* | Maximum number of iterations in the augmenting stage. |
| *maxitl* | Maximum number of iterations with the nonrestarted LSQR method when the augmented vectors are kept fixed. |
| $m$ | Maximum number of GK vectors. |
| *reorth*012 | String deciding whether no, one, or two-sided reorthogonalization is used in either stage. |
| *mreorth* | Number of vectors to be reorthogonalized during the nonrestarted LSQR stage, when the augmented vectors are kept fixed. If $mreorth > 0$, then one-sided reorthogonalization is applied to the "short" vectors. |
| *tollsqr* | Tolerance $\delta^{\text{lsqr}}$ in (4.6) for accepting a computed approximate solution as the solution of (1.1). |
| *tolharm* | Tolerance $\delta^{\text{harm}}$ in (3.30) for accepting an approximate singular triplet as a singular triplet of $A$ and use it for augmentation. |

We compare `alsqr` to the MATLAB code `lsqr`[2] for the standard LSQR method by Paige and Saunders [26] and to the MATLAB code `lsmr`[3] by Fong and Saunders [10]. We remark that the performance of the methods in our comparisons depends on the machine architecture, coding style, and stopping criteria. These may significantly affect the performance, regardless of the theoretical properties of the methods. We therefore do not report CPU times, but instead measure performance in terms of the required number of matrix-vector product evaluations with the matrices $A$ and $A^T$. We set all common parameters for different methods to the same values for each example, and reorthogonalize only against the last $m$ vectors in each method. We use the initial approximate solution $x_0 = 0$ for all methods and examples.

There are many preconditioned iterative methods available for the solution of (1.1). It is difficult to make a fair comparison, because the construction of many preconditioners is determined by several parameters, including drop tolerance and available storage. Here we only note that our method is unique in that an approximate solution to the LS problem is computed already during the construction of the augmented Krylov subspaces.

We present six numerical examples with matrices from the Matrix Market collection [7, 9]. The matrices $A$, their properties, as well as the definition of the vector $b$, are described in Table 7.1. All matrices are of full column rank except for the matrix of Example 7.6. In Table 7.1 "$\ell$" denotes the number of rows, "$n$" the number of columns, and "$nnz$" the number of nonzero entries of the matrices. The column labeled "Cond. #" shows the condition number estimate computed by the MATLAB function `condest` when $A$ is square. For the rectangular matrix ILLC1850, we deter-

[1] Code is available at http://www.math.uri.edu/∼jbaglama

[2] The `lsqr` MATLAB code is not the code that comes with MATLAB. The used code was adapted to output the norm of the residual error in each iteration and to carry out reorthogonalization as described in section 4.

[3] http://www.stanford.edu/group/SOL/software/lsmr.html. The code was adapted to output the norm of the residual error in each iteration.

mined the condition number with the MATLAB function `cond`. The vectors $b$ also were chosen from the Matrix Market collection when available, otherwise we computed the vector $b$ with the MATLAB function `b=rand(size(A,1),1)`. This yields a vector $b$ with uniformly distribution entries in the interval $(0, 1)$. All computations were carried out using MATLAB version 7.12.0.0635 R2011a on a Dell XPS workstation with an Intel Core2 Quad processor and 4 GB of memory running under the Windows Vista operating system. Machine precision is $2.2 \cdot 10^{-16}$. One-sided reorthogonalization is used in both stages for all examples except for Example 7.3 where two-sided reorthogonalization is used in the augmenting stage and one-sided reorthogonalization is used in the LSQR stage. The matrix $A$ in Example 7.3 is very ill-conditioned, see Table 7.1; hence two-sided reorthogonalization is required during the iteration process to approximate singular vectors. See [1, 29] for remarks on requiring two-sided reorthogonalization during the GK process for singular triplet approximation.

TABLE 7.1

*Matrix Market collection of matrices $A$, properties, and vectors $b$ used in the numerical examples. The rank-deficient matrix ILLC1850* was obtained from ILLC1850 by replacing the second column by twice the first column of the latter.*

| Example | Matrix | $\ell$ | $n$ | $nnz$ | Cond. # | $b$ |
|---|---|---|---|---|---|---|
| *Example 7.1* | ILLC1850 | 1850 | 712 | 8758 | $1.4 \cdot 10^3$ | ILLC1850_RHS1 |
| *Example 7.2* | E05R0000 | 236 | 236 | 5856 | $5.9 \cdot 10^4$ | E05R0000_RHS1 |
| *Example 7.3* | E20R0100 | 4241 | 4241 | 131566 | $2.2 \cdot 10^{10}$ | E20R0100_RHS1 |
| *Example 7.4* | NOS5 | 468 | 468 | 2820 | $2.9 \cdot 10^4$ | `rand(468,1)` |
| *Example 7.5* | CK656 | 656 | 656 | 3884 | $1.2 \cdot 10^7$ | `rand(656,1)` |
| *Example 7.6* | ILLC1850* | 1850 | 712 | 8645 | – | ILLC1850_RHS1 |

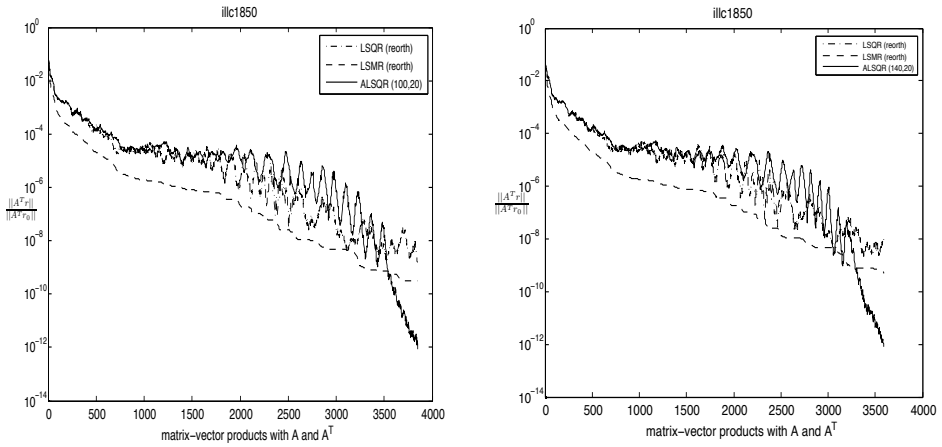


FIG. 7.1. *Example 7.1: LSQR(reorth) and LSMR(reorth) denote that reorthogonalization was applied to the last $m$ vectors. ALSQR(100,20) denotes* `alsqr` *with parameters $m = 100$ and $k = 20$, and ALSQR(140,20) shows the performance of* `alsqr` *with $m = 140$ and $k = 20$.* `alsqr` *switched to nonrestarted LSQR at $2,840$ matrix-vector products in the left-hand side graph and at $2,680$ matrix-vector products for the right-hand side graph.*

*Example 7.1.* This example uses the same matrix $A$ and vector $b$ as Example 2.1 of section 2. The vector $b$ is not in $\mathcal{R}(A)$. The left-hand side graph of Figure

7.1 is determined with the code `alsqr` using the parameter values $k = 20$, $adjust = 40$, and $m = 100$. The right-hand side graph of Figure 7.1 is obtained with `alsqr` using the parameters $k = 20$, $adjust = 70$, and $m = 140$. We used $tolharm = 5 \cdot 10^{-2}$ to determine when to accept approximate singular vectors. The iterations were continued until the residual vectors $r$ generated by `alsqr` for the first time satisfied $\|A^T r\|/\|A^T r_0\| \leq 10^{-12}$. The graphs of Figure 7.1 show the quotient $\|A^T r\|/\|A^T r_0\|$ versus the number of matrix-vector products with $A$ and $A^T$ for each iteration of each method. The graphs marked `lsqr(reorth)` and `lsmr(reorth)` are for iteration with reorthogonalization. All methods reorthogonalized the last 100 vectors for the left-hand side graphs and the last 140 vectors for the right-hand side graphs of Figure 7.1. The `alsqr` algorithm exited the augmenting stage with all $k = 20$ approximate singular vectors converged after $2,840$ matrix-vector product evaluations for the left-hand side graph, and after $2,680$ matrix-vector product evaluations for the right-hand side graph. Having computed these approximate singular vectors, `alsqr` continued the iterations as a nonrestarted augmented LSQR method. The graphs show that augmentation by approximate singular vectors led to faster convergence and that `alsqr` converged before `lsqr` and `lsmr`. □

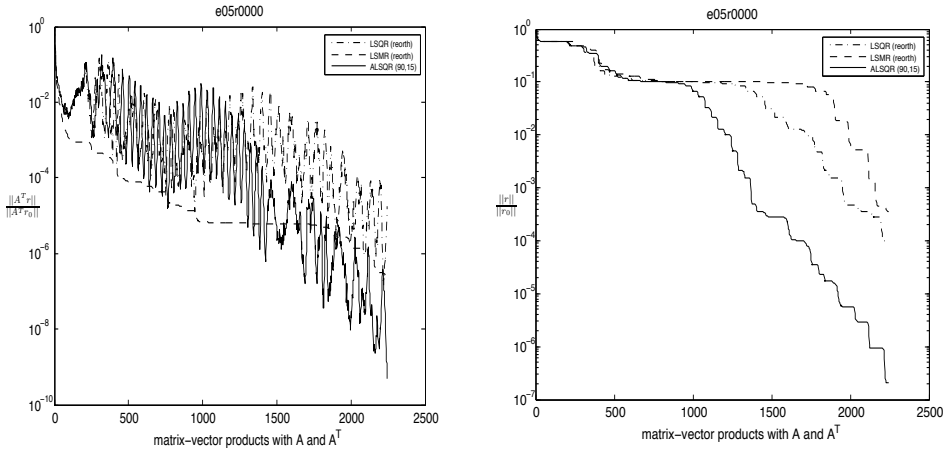


FIG. 7.2. *Example 7.2: LSQR(reorth) and LSMR(reorth) indicates that reorthogonalization of the last m vectors was carried out. ALSQR(90,20) denotes* `alsqr` *with parameters* $m = 90$ *and* $k = 15$. `alsqr` *switched to nonrestarted LSQR at* $1,230$ *matrix-vector product evaluations. The left-hand side graph shows* $\|A^T r\|/\|A^T r_0\|$ *for each iteration and the right-hand side graph displays* $\|r\|/\|r_0\|$ *for each iteration.*

*Example 7.2.* We let the matrix $A$ and vector $b$ be E05R0000 and E05R0000_RHS1, respectively, from the DRIVCAV set of the Matrix Market collection. The intended use of the linear systems in this collection is for testing iterative Krylov solvers, because it is difficult to find suitable preconditioners for the matrices. Since the linear system of equations is consistent, we can show convergence of both the quotients $\|A^T r\|/\|A^T r_0\|$ and $\|r\|/\|r_0\|$, where as usual $r$ denotes the generated residual vector and $r_0$ the initial residual vector. We use the parameters $k = 15$, $adjust = 40$, $m = 90$ for `alsqr`. The value $tolharm = 3.5 \cdot 10^{-3}$ was used when deciding when to accept computed approximate singular vectors as converged. `alsqr` exited the augmening stage with all $k = 15$ approximate singular vectors converged when the matrix-vector product count was $1,230$. The iterations were continued with the fixed augmenting

vectors until a residual vector satisfied $\|A^T r\|/\|A^T r_0\| \leq 10^{-9}$.

The left-hand side graph of Figure 7.2 displays $\|A^T r\|/\|A^T r_0\|$ versus the number of matrix-vector products with the matrices $A$ and $A^T$ for each iteration and for each method in our comparison. The right-hand side graph is analogous; it displays the quotients $\|r\|/\|r_0\|$ instead of $\|A^T r\|/\|A^T r_0\|$. This graph shows a fast steady decrease of the residual norm when `alsqr` carries out LSQR iterations with the fixed augmenting vectors. □

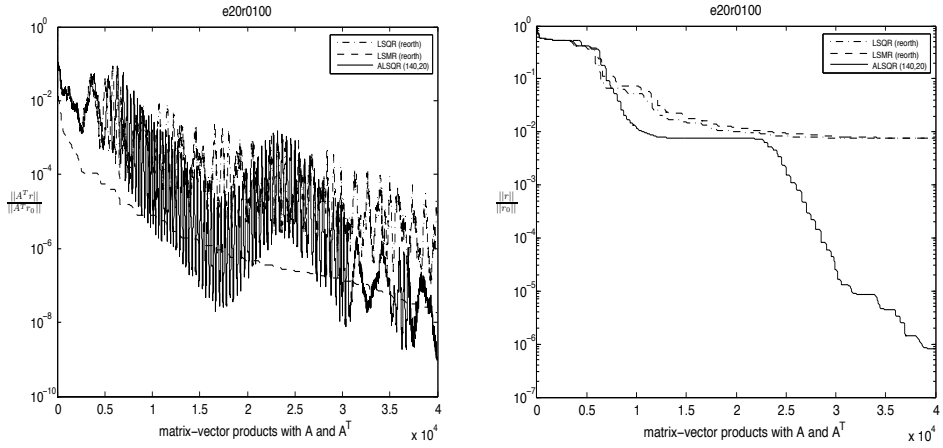


FIG. 7.3. *Example 7.3: LSQR(reorth) and LSMR(reorth) indicate that reorthogonalization of the m last vectors was carried out. The method ALSQR(m,k) for $m = 140$ and $k = 20$ is compared with LSQR and LSMR.* `alsqr` *switched to nonrestarted LSQR after* $30,280$ *matrix-vector product evaluations. The left-hand side graph depicts $\|A^T r\|/\|A^T r_0\|$ for each iteration, while the right-hand side graph shows $\|r\|/\|r_0\|$ for each iteration.*

*Example 7.3.* Let the matrix $A$ and vector $b$ be E20R0100 and E20R0100_RHS1, respectively, from the DRIVCAV set of the Matrix Market collection; see Example 7.2 for comments on this set of linear systems of equations. The code `alsqr` used the parameter values $k = 20$, $adjust = 90$, and $m = 140$. The matrix has a large condition number, $2.2 \cdot 10^{10}$, which leads to large oscillations in the quotients $\|A^T r\|/\|A^T r_0\|$ and very slow convergence. We used the same stopping criterion as in Example 7.2. Figure 7.3 is analogous to Figure 7.2.

We used the parameter value $tolharm = 1.22 \cdot 10^{-4}$ to decide when approximate singular vectors could be considered converged. The code `alsqr` exited the augmenting stage with $k = 20$ converged approximate singular vectors when $30,280$ matrix-vector products with $A$ and $A^T$ had been computed. Notice that the residual curve in the right-hand side graph starts to decrease steadily long before the augmenting stage ends. This illustrates the positive effect of augmentation already while the augmenting vectors are computed. □

*Example 7.4.* The matrix $A$ is NOS5 from the LANPRO set in the Matrix Market collection. The matrices in this set stem from linear equations in structural engineering. This matrix set does not contain vectors $b$ that can be used in (1.1). We therefore let $b$ be a random vector with uniformly distributed entries in the interval $(0, 1)$. We use the parameter values $k = 20$, $adjust = 60$, $m = 120$, and $tolharm = 10^{-2}$ for the code `alsqr`. The augmenting stage, which lasted until $k = 20$ approximate singular vectors had converged, required $4,000$ matrix-vector product evaluations with

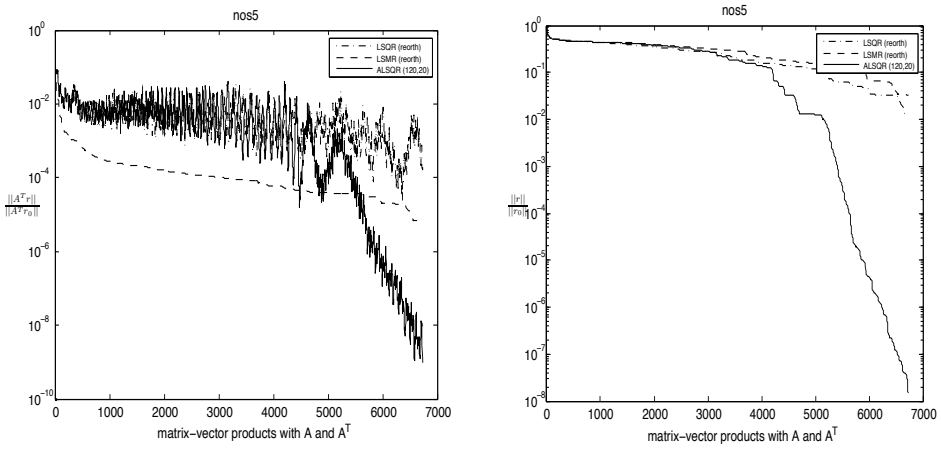


FIG. 7.4. *Example 7.4: LSQR(reorth) and LSMR(reorth) denote that reorthogonalization of the last m vectors was performed. The method ALSQR(m,k) is for* $m = 120$ *and* $k = 20$ *compared to LSQR and LSMR.* `alsqr` *switched to nonrestarted LSQR after* $4,000$ *matrix-vector product evaluations. The left-hand side graph shows* $\|A^T r\|/\|A^T r_0\|$ *for each iteration and the right-hand side graph displays* $\|r\|/\|r_0\|$ *for each iteration.*

$A$ and $A^T$. Iterations were then continued with the augmented LSQR method until $\|A^T r\|/\|A^T r_0\| \leq 10^{-9}$. Figure 7.4 is analogous to Figure 7.3. The right-hand side graph displays fast and steady decrease of $\|r\|/\|r_0\|$ of the nonrestarted LSQR method with fixed augmented vectors. □

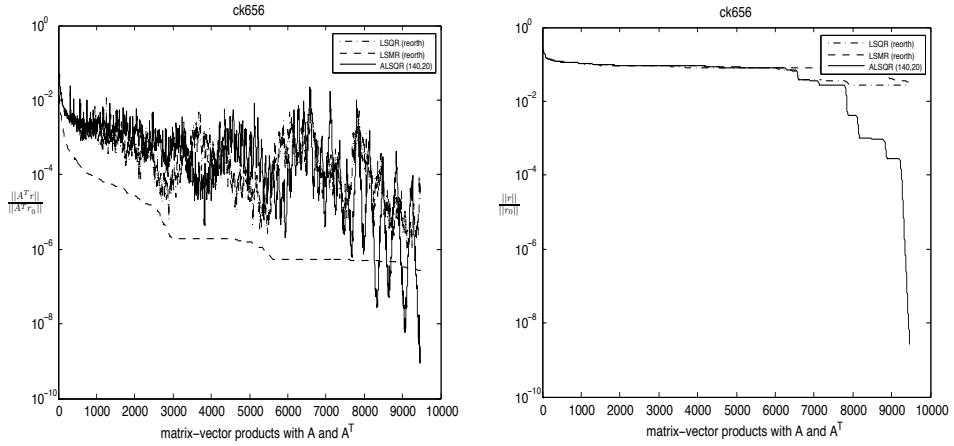


FIG. 7.5. *Example 7.5: LSQR(reorth) and LSMR(reorth) denotes that reorthogonalization of the last m vectors was carried out. ALSQR(140,20) indicates that* `alsqr` *is applied with* $m = 140$ *and* $k = 20$. *The code* `alsqr` *did not switch to nonrestarted LSQR before the convergence criterion was satisfied. The left-hand side graph displays* $\|A^T r\|/\|A^T r_0\|$ *for each iteration, and the right-hand side graph shows* $\|r\|/\|r_0\|$ *for each iteration.*

*Example 7.5.* The matrix $A$ is chosen to be CK656, which is the largest matrix in the CHUCK set of the Matrix Market collection. This matrix has many clustered and multiple eigenvalues. The matrices in this collection arise from linear systems of

equations in structural engineering. This collection does not contain right-hand side vectors. Therefore, we let $b$ be a vector with random entries as in Example 7.4. We use the parameters $k = 20$, $adjust = 80$, $m = 140$, and $tolharm = 10^{-4}$ for `alsqr`. Iterations were terminated when $\|A^T r\|/\|A^T r_0\| \leq 10^{-9}$. The left-hand side graph of Figure 7.4 depicts $\|A^T r\|/\|A^T r_0\|$ versus the number of matrix-vector products with $A$ and $A^T$. Figure 7.5 is analogous to Figure 7.4. In this example, `alsqr` did not exit the augmenting stage before the stopping criterion was satisfied, i.e., the stopping condition was satisfied before $k = 20$ approximate singular vectors had converged. □

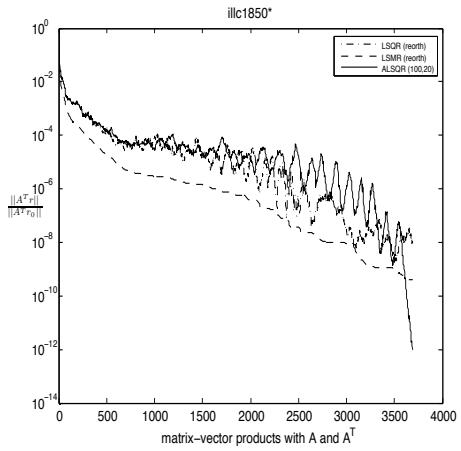


FIG. 7.6. *Example 7.6: The matrix $A$ in this example is rank-deficient and the right-hand size $b$ is not in the column space of $A$. Therefore, we show only the graph $\|A^T r\|/\|A^T r_0\|$ versus the number of matrix-vector products with $A$ and $A^T$. The graphs LSQR(reorth) and LSMR(reorth) display results obtained when reorthogonalization of the last $m$ vectors was carried out. ALSQR(100,20) denotes that* `alsqr` *is applied with the parameters $m = 100$ and $k = 20$.* `alsqr` *switched over to nonrestarted LSQR after $3,080$ matrix-vector product evaluation.*

*Example 7.6.* The matrix $A$ used in this example is obtained from the matrix ILLC1850 of Example 7.1 by letting the second column be twice the first column. We refer to the rank-deficient matrix so obtained as ILLC1850*. The vector $b$ is the same as in Example 7.1. The LS problem (1.1) is inconsistent. We chose the parameters $k = 20$, $adjust = 40$, and $m = 100$ for `alsqr`, and used $tolharm = 4 \cdot 10^{-2}$ to decide when to accept approximate singular vectors as converged. All methods reorthogonalized the 100 last vectors. The required $k = 20$ approximate singular vectors had converge after $3,080$ matrix-vector product evaluations with $A$ and $A^T$. At this point the code switched to run as an augmented nonrestarted LSQR method. The iterations were terminated as soon as $\|A^T r\|/\|A^T r_0\| \leq 10^{-11}$.

Figure 7.6 shows $\|A^T r\|/\|A^T r_0\|$ versus the number of matrix-vector product evaluations with $A$ and $A^T$. This example illustrates that `alsqr` can be competitive also when applied to a rank-deficient LS problem. □

**8. Conclusion.** We have described a new augmented LSQR method for large-scale linear LS problems or linear systems of equations. During the initial iterations, the method computes approximations of harmonic Ritz vectors that are used for augmenting the solution subspaces. Simultaneously, the method computes improved approximate solutions of the LS problem (1.1). Subsequently, the augmented vectors are kept fixed and used to form nonstandard Krylov subspaces used by a nonrestarted

LSQR method. Numerical examples show the proposed method to be competitive.

**Acknowledgment.** We would like to thank the referees for carefully reading the paper and for many comments that improved the presentation. Research in part supported by NSF grant DMS-1115385.

REFERENCES

[1] J. Baglama and L. Reichel, *Augmented implicitly restarted Lanczos bidiagonalization methods*, SIAM J. Sci. Comput., 27 (2005), pp. 19–42.
[2] J. Baglama and L. Reichel, *Restarted block Lanczos bidiagonalization methods*, Numer. Algorithms, 43 (2006), pp. 251–272.
[3] J. Baglama and L. Reichel, *An implicitly restarted block Lanczos bidiagonalization method using Leja shifts*, submitted for publication.
[4] M. Benzi and M. Tuma, *A robust preconditioner with low memory requirements for large sparse least squares problems*, SIAM J. Sci. Comput., 25 (2003), pp. 499–512.
[5] Å. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.
[6] Å. Björck and J. Y. Yuan, *Preconditioners for least squares problems by LU factorization*, Electron. Trans. Numer. Anal., 8 (1997), pp. 26–35.
[7] R. Boisvert, R. Pozo, K. Remington, B. Miller, and R. Lipman, MatrixMarket, 1996. The matrices are available at http://math.nist.gov/MatrixMarket/
[8] S.-C. Choi, *Iterative Methods for Singular Linear Equations and Least Squares*, Ph.D. thesis, Institute for Computational and Mathematical Engineering, Stanford University, 2006.
[9] I. S. Duff, R. G. Grimes, and J. G. Lewis, *User's Guide for the Harwell-Boeing Sparse Matrix Collection (Release I)*, Technical Report TR/PA/92/86, CERFACS, Toulouse, France, 1992. Matrices available at http://math.nist.bov/MatrixMarket/
[10] D. C.-L. Fong and M. A. Saunders, *LSMR: An iterative algorithm for sparse least-squares problems*, SIAM J. Sci. Comput., 33 (2011), pp. 2950–2971.
[11] K. Hayami, J.-F. Yin, and T. Ito, *GMRES methods for least squares problems*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2400–2430.
[12] M. E. Hochstenbach, *Harmonic and refined extraction methods for the singular value problem, with applications in least squares problems*, BIT, 44 (2004), pp. 721–754.
[13] Z. Jia, *Some properties of LSQR for large sparse linear least squares problems*, J. Sys. Sci. Complex., 23 (2010), pp. 815–821.
[14] Z. Jia and D. Niu, *An implicitly restarted refined bidiagonalization Lanczos method for computing a partial singular value decomposition*, SIAM J. Matrix Anal. Appl., 25 (2003), pp. 246–265.
[15] Z. Jia and D. Niu, *A refined harmonic Lanczos bidiagonalization method and an implicitly restarted algorithm for computing the smallest singular triplets of large matrices*, SIAM J. Sci. Comput., 32 (2010), pp. 714–744.
[16] S. Karimi, D. K. Salkuyeh, and F. Toutounian, *A preconditioner for the LSQR algorithm*, J. Appl. Math. Informatics, 26 (2008), No. 1-2, pp. 213–222.
[17] E. Kokiopoulou, C. Bekas, and E. Gallopoulos, *Computing smallest singular triplets with implicitly restarted Lanczos bidiagonalization*, Appl. Numer. Math., 49 (2004), pp. 39–61.
[18] R. M. Larsen, *Lanczos bidiagonalization with partial reorthogonalization*, Ph.D. thesis, Dept. Computer Science, University of Aarhus, Aarhus, Denmark, 1998.
[19] R. M. Larsen, *Combining implicit restarts and partial reorthogonalization in Lanczos bidiagonalization*, http://soi.stanford.edu/∼rmunk/PROPACK/
[20] R. B. Morgan, *Computing interior eigenvalues of large matrices*, Linear Algebra Appl., 154-156 (1991), pp. 289–309.
[21] R. B. Morgan, *A restarted GMRES method augmented with eigenvectors*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 1154–1171.
[22] R. B. Morgan, *Implicitly restarted GMRES and Arnoldi methods for nonsymmetric systems of equations*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1112–1135.
[23] R. B. Morgan, *GMRES with deflated restarting*, SIAM J. Sci. Comput., 24 (2002), pp. 20–37.
[24] C. C. Paige, *Bidiagonalization of matrices and solution of linear equations*, SIAM J. Numer. Anal., 11 (1974), pp. 197–209.
[25] C. C. Paige, B. N. Parlett, and H. A. van der Vorst, *Approximate solutions and eigenvalue bounds from Krylov subspaces*, Numer. Linear Algebra Appl., 2 (1995), pp. 115–134.
[26] C. C. Paige and M. A. Saunders, *LSQR: An algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Software, 8 (1982), pp. 43–71.

[27] L. Reichel and Q. Ye, *A generalized LSQR algorithm*, Numer. Linear Algebra Appl., 15 (2008), pp. 643–660.
[28] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.
[29] H. D. Simon and H. Zha, *Low rank matrix approximation using the Lanczos bidiagonalization process with applications*, SIAM J. Sci. Comput., 21 (2000), pp. 2257–2274.