A GENERALIZED LSQR ALGORITHM

LOTHAR REICHEL* AND QIANG YE†

Abstract. LSQR is a popular iterative method for the solution of large linear systems of equations and least-squares problems. This paper presents a generalization of LSQR that allows the choice of an arbitrary initial vector for the solution subspace. Computed examples illustrate the benefit of being able to choose this vector.

Key words. linear system of equations, least-squares problem, iterative method, linear discrete ill-posed problem.

AMS subject classifications. 65F10, 65F20, 65F22

1. Introduction. This paper is concerned with the solution of least-squares problems

$$\min_{x\in\mathbb{R}^n} \|b-Ax\|, \qquad A\in\mathbb{R}^{m\times n}, \qquad b\in\mathbb{R}^m, \qquad m\geq n, \tag{1.1}$$

with a matrix of size too large to allow rapid factorization. We therefore consider iterative solution methods. When $m > n$, LSQR by Paige and Saunders [10] is the iterative method of choice, and it is sometimes also applied to systems with a square matrix ($m = n$). LSQR is a particular implementation of the conjugate gradient method applied to the normal equations associated with (1.1). Throughout this paper $\|\cdot\|$ denotes the Euclidean vector norm or the associated induced matrix norm, $\mathcal{R}(M)$ the range of the matrix M, and $e_j = [0,\dots,0,1,0,\dots,0]^T$ the jth axis vector of appropriate dimension.

The Lanczos bidiagonalization algorithm is at the core of the LSQR method. Application of ℓ steps of this algorithm to the matrix A with initial vector $u_1 = b/\|b\|$ yields the decompositions

$$AV_\ell = U_{\ell+1}\bar{B}_\ell, \qquad A^TU_\ell = V_\ell B_\ell^T, \tag{1.2}$$

where $U_i = [u_1,u_2,\dots,u_i]\in\mathbb{R}^{m\times i}$, $i=\ell,\ell+1$, and $V_\ell=[v_1,v_2,\dots,v_\ell]\in\mathbb{R}^{n\times\ell}$ have orthonormal columns, $B_\ell\in\mathbb{R}^{\ell\times\ell}$ is lower bidiagonal, and $\bar{B}_\ell$ is obtained by appending a suitable multiple of e_ℓ^T to B_ℓ. Moreover,

$$\mathcal{R}(V_\ell) = \mathcal{K}_\ell(A^TA, A^Tb) = \mathrm{span}\{A^Tb, (A^TA)A^Tb, \dots, (A^TA)^{\ell-1}A^Tb\} \tag{1.3}$$

is an ℓ-dimensional Krylov subspace. In particular, the first column of V_ℓ is a normalization of A^Tu_1. LSQR is based on the decompositions (1.2). We remark that if the Lanczos bidiagonalization algorithm breaks down because for some ℓ the decompositions (1.2) with the stated properties do not exist, then the solution of (1.1) lives in the subspace (1.3) available at breakdown.

Application of LSQR to the solution of (1.1) yields a sequence of approximate solutions $x_1, x_2, x_3, \dots$, such that

$$\|b-Ax_\ell\| = \min_{x\in\mathcal{R}(V_\ell)} \|b-Ax\|, \qquad x_\ell\in\mathcal{R}(V_\ell). \tag{1.4}$$

*Department of Mathematical Sciences, Kent State University, Kent, OH 44242 (reichel@math.kent.edu). Supported in part by an OBR Research Challenge Grant.

†Department of Mathematics, University of Kentucky, Lexington, KY 40506 (qye@ms.uky.edu). Supported in part by National Science Foundation grant DMS-0411502.

The minimization problem (1.4) can be solved by using the Lanczos decomposition (1.2). We have

$$\min_{x\in\mathcal{R}(V_\ell)} \|b - Ax\| = \min_{y\in\mathbb{R}^\ell} \|b - AV_\ell y\| = \min_{y\in\mathbb{R}^\ell} \|\|b\|e_1 - \bar{B}_\ell y\|. \tag{1.5}$$

Let y_ℓ denote the solution of the minimization problem on the right-hand side of (1.5). Then $x_\ell = V_\ell y_\ell$. LSQR only requires the simultaneous storage of a few columns of $U_{\ell+1}$ and V_ℓ. The computation of the decomposition (1.2), and therefore also the determination of x_ℓ, requires the evaluation of 2ℓ matrix-vector products; ℓ with A and A^T each.

All vectors in the Krylov subspace $\mathcal{R}(V_\ell)$, from which the approximate solution x_ℓ is taken, can be expressed as a polynomial in A^TA of degree at most $\ell-1$ times the vector A^Tb; see (1.3). In some applications, the vector A^Tb represents the discretization of a much smoother function than the vector b, but the solution of (1.1) is not the discretization of a smooth function. Then the computation of a useful approximate solution of (1.1) typically requires a large number of iterations. This situation is discussed in Example 3.2 of Section 3. Observe that when A is very ill-conditioned due to the presence of one or several "tiny" singular values, the vector $A^Tb = A^TAx$ has significantly smaller components of singular vectors associated with these singular values than the solution x of (1.1) and this generally makes it difficult to approximate x well by a vector from low-dimensional subspaces of the form (1.3). In this situation, LSQR typically requires a large number of iterations to arrive at a good approximate solution $x_\ell \in \mathcal{R}(V_\ell)$. This observation suggests that it may be advantageous to use an ℓ-dimensional subspace of $\mathbb{R}^n$ different from (1.3). In particular, it might be desirable to use a vector other than A^Tb to construct the subspace $\mathcal{R}(V_\ell)$.

Section 2 describes a generalization of LSQR that allows us to choose the initial vector v_1 of the matrix V_ℓ and construct a decomposition analogous to (1.2) with $\mathcal{R}(V_\ell) \neq \mathcal{K}_\ell(A^TA, A^Tb)$. Computed examples in Section 3 illustrate that a suitable choice of the initial vector can reduce the number of iterations ℓ required to determine an approximate solution of (1.1) of desired accuracy substantially. Moreover, in the context of linear discrete ill-posed problems with a right-hand side b that is contaminated by an error, a suitable choice of initial vector of V_ℓ can increase the achievable quality of the computed approximate solutions.

Lanczos bidiagonalization was first described by Golub and Kahan [5], who discuss decompositions different from (1.2); in their decompositions the analog of the bidiagonal matrix B_ℓ is upper bidiagonal. Their decompositions allow the choice of an arbitrary initial vector of the solution subspace, however, due to numerical instability the decomposition in [5] is not well suited for the solution of least-squares problems. Paige and Saunders [10] found the decompositions (1.2) to perform better in the context of least-squares computations. We therefore base the iterative method of this paper on a modification of the decompositions (1.2).

A different approach to modifying the subspace (1.3) in which approximate solutions of (1.1) are determined is described by Calvetti et al. [3], who augment the spaces (1.3), for $\ell = 1, 2, 3, \dots$, by a user-chosen subspace in the context of the CGLS algorithm. The latter algorithm is mathematically, but not numerically, equivalent to LSQR. Paige and Saunders [10] point out that LSQR performs better for linear systems (1.1) with ill-conditioned matrices. We find the simplicity of the algorithm of the present paper and the fact that the algorithm generalizes LSQR, rather than CGLS, attractive, and believe that our generalization of LSQR will prove itself useful in various applications.

2. Generalized LSQR.

2. Generalized LSQR. We first describe a decomposition similar to (1.2), which allows the first column of the matrix V_ℓ to be different from a normalization of $A^T u_1$. This decomposition is the basis for our generalization of LSQR.

Given arbitrary unit vectors u_1 and v_1, we construct, in the generic case, the matrices

$$U_i = [u_1, u_2, \ldots, u_i] \in \mathbb{R}^{m\times i}, \qquad V_i = [v_1, v_2, \ldots, v_i] \in \mathbb{R}^{n\times i}, \qquad i = \ell, \ell+1,$$

with orthonormal columns, such that

$$AV_\ell = U_{\ell+1} T_{\ell+1,\ell}, \qquad A^T U_\ell = V_{\ell+1} S_{\ell+1,\ell}, \tag{2.1}$$

where the matrices $S_{\ell+1,\ell} = [s_{jk}] \in \mathbb{R}^{(\ell+1)\times\ell}$ and $T_{\ell+1,\ell} = [t_{jk}] \in \mathbb{R}^{(\ell+1)\times\ell}$ are tridiagonal. We refer to the decompositions (2.1) as a Partial Lanczos Bi-Tri-Diagonalization (PLBTD). Algorithm 1 below determines the decompositions (2.1) or a modification thereof in case of occurrence of an exceptional case. The vectors w in the algorithm are used for temporary storage of intermediate quantities.

Algorithm 1. Partial Lanczos Bi-tridiagonalization:

1 *Input:* $A \in \mathbb{R}^{m\times n}$, *unit vectors* $u_1 \in \mathbb{R}^m$ *and* $v_1 \in \mathbb{R}^n$, *number of steps* ℓ;
2 *Output:* $S_{\ell-d+1,\ell} = [s_{jk}] \in \mathbb{R}^{(\ell-d+1)\times\ell}$, $T_{\ell+1,\ell} = [t_{jk}] \in \mathbb{R}^{(\ell+1)\times\ell}$,
  $U_{\ell+1} = [u_1, u_2, \ldots, u_{\ell+1}] \in \mathbb{R}^{m\times(\ell+1)}$,
  $V_{\ell-d+1} = [v_1, v_2, \ldots, v_{\ell-d+1}] \in \mathbb{R}^{n\times(\ell-d+1)}$,
  where $d = 0$ *generically and* $d = 1$ *in case of an exceptional case;*
3 *Initialization:* $d := 0$; $u_0 := 0$; $v_0 := 0$; $S_{\ell+1,\ell} = [s_{jk}] := 0$; $T_{\ell+1,\ell} = [t_{jk}] := 0$;
4 *for* $k := 1, 2, \ldots, \ell$
5   $w := A^T u_k$; $s_{k-1,k} := v_{k-1}^T w$; $s_{k-d,k} := v_{k-d}^T w$;
6   $w := w - \Sigma_{j=k-1}^{k-d} s_{jk} v_j$; $s_{k-d+1,k} := \|w\|$;
7   *if* $s_{k-d+1,k} \neq 0$, *then* $v_{k-d+1} := w/s_{k-d+1,k}$;
8   *if* $s_{k-d+1,k} = 0$ *and* $d = 1$, *then stop;*
9   *if* $s_{k-d+1,k} = 0$ *and* $d = 0$, *then* $d := 1$;
10   $w := Av_k$; $t_{k-1,k} := u_{k-1}^T w$; $t_{kk} := u_k^T w$;
11   $w := w - t_{kk} u_k - t_{k-1,k} u_{k-1}$; $t_{k+1,k} := \|w\|$;
12   *if* $t_{k+1,k} \neq 0$, *then* $u_{k+1} := w/t_{k+1,k}$;
13   *if* $t_{k+1,k} = 0$, *then stop;*
14 *endfor*

Remark 1: The variable d is initialized to 0 in the algorithm, and is set to 1 (and then stays 1) if the conditions on line 9 hold. At the beginning of the for-loop, u_k and v_{k-d} are available and we either generate v_{k-d+1} (on line 7) or increase the value of d from 0 to 1 (on line 9). In either case, v_{k-d+1} (with the current value of d) is available. In particular, v_k is available at the end of line 9 and is used to generate u_{k+1} on lines 10-12. Thus, at the end of the for-loop, u_{k+1} and v_{k-d+1} are available. □

Remark 2: If the conditions on lines 8 or 13 hold, then the algorithm breaks down. These are benign breakdowns, however, since the solution of (1.1) can be recovered from the partial decompositions available at breakdown. Details are discussed below. □

We first consider the case when no breakdown occurs. Then the constructions on lines 5-6 and lines 10-11 yield, respectively, that

$$A^T u_k = \Sigma_{j=k-1}^{k-d+1} s_{jk} v_j, \tag{2.2}$$

$$Av_k = t_{k-1,k} u_{k-1} + t_{kk} u_k + t_{k+1,k} u_{k+1}. \tag{2.3}$$

The coefficient t_{ij} and s_{ij} are chosen to enforce orthogonality of u_{k+1} to u_k, u_{k-1}, and of v_{k-d+1} to v_{k-d}, v_{k-1}. For clarity of presentation, we have used classical Gram-Schmidt orthogonalization in the description of the algorithm; see below for further comments on this. We now show that the matrices $U_{\ell+1}$ and $V_{\ell+1}$ determined by Algorithm 1 have orthogonal columns.

THEOREM 2.1. *Assume that no breakdown occurs during the execution of Algorithm 1 (i.e., the conditions on lines 8 and 13 are false). Let d take on the value defined at the end of the algorithm, and let $T_{ij} = [t_{pq}] \in \mathbb{R}^{i\times j}$ and $S_{ij} = [s_{pq}] \in \mathbb{R}^{i\times j}$. Then at the end of the algorithm, we have generated two orthonormal sequences $u_1, u_2, \ldots, u_{\ell+1}$ and $v_1, v_2, \ldots, v_{\ell-d+1}$, such that*

$$AV_\ell = U_{\ell+1}T_{\ell+1,\ell}, \qquad A^T U_\ell = V_{\ell-d+1}S_{\ell-d+1,\ell}, \tag{2.4}$$

where $U_i = [u_1, u_2, \ldots, u_i] \in \mathbb{R}^{m\times i}$ and $V_i = [v_1, v_2, \ldots, v_i] \in \mathbb{R}^{n\times i}$. Furthermore,

$$S_{\ell\ell}^T = T_{\ell\ell}. \tag{2.5}$$

Proof. By Remark 1 above, we have at the end of the algorithm (step ℓ) generated the vectors $u_1, u_2, \ldots, u_{\ell+1}$ and $v_1, v_2, \ldots, v_{\ell-d+1}$. Then the first equation of (2.4) follows directly from (2.3). For the second equation of (2.4), we note that if $d = 1$ in step k, then $s_{k+1,k} = 0$. Thus, equation (2.2) always can be expressed as

$$A^T u_k = s_{kk}v_k + s_{k-1,k}v_{k-1} + s_{k+1,k}v_{k+1}. \tag{2.6}$$

This shows the first $\ell - 1$ columns of the second equation of (2.4), and equation (2.2) gives the last column. Hence, equation (2.4) is proved.

We now show the orthogonality by induction. Clearly, u_2 is orthogonal to u_1, and v_2 is orthogonal to v_1. In the beginning of step k, assume that the vectors $v_1, v_2, \ldots, v_{k-d}$ and $u_1, u_2, \ldots, u_k$, which already have been generated, are orthogonal, in fact orthonormal,

$$u_i^T u_j = v_i^T v_j = \begin{cases} 0, & i \neq j, \\ 1, & i = j. \end{cases}$$

Then, we either determine v_{k-d+1} (on line 7) or we do not compute a new v-vector and instead set $d = 1$ (on line 9). In the former case, v_{k-d+1} is orthogonal to v_{k-d} and v_{k-1} by construction. Furthermore, for $1 \leq i \leq k-2$, we obtain from (2.2) and (2.3) that

$$\begin{aligned} s_{k-d+1,k}v_i^T v_{k-d+1} &= v_i^T A^T u_k - \Sigma_{j=k-1}^{k-d} s_{jk}v_i^T v_j = (Av_i)^T u_k \\ &= (t_{ii}u_i + t_{i-1,i}u_{i-1} + t_{i+1,i}u_{i+1})^T u_k = 0, \end{aligned}$$

which implies that $v_i^T v_{k-d+1} = 0$. Thus, in either cases (lines 7 or 9), the v-vectors available at the end of step k are orthogonal. Similarly, u_{k+1} as determined in step k is orthogonal to u_k and u_{k-1}, because of the definition of the coefficients $t_{k-1,k}$ and t_{kk}. For $1 \leq i \leq k-2$, we obtain from (2.2) and (2.3) that

$$\begin{aligned} t_{k+1,k}u_i^T u_{k+1} &= u_i^T Av_k - t_{kk}u_i^T u_k - t_{k-1,k}u_i^T u_{k-1} = (A^T u_i)^T v_k \\ &= (s_{ii}v_i + s_{i-1,i}v_{i-1} + s_{i+1,i}v_{i+1})^T v_k = 0, \end{aligned}$$

where we have used (2.6) with $s_{0,1}v_0 = 0$. Thus, $u_1, u_2, \ldots, u_k, u_{k+1}$ are orthogonal. This completes the proof of orthogonality.

Finally, we obtain from (2.4) that

$$T_{\ell\ell} = U_\ell^T A V_\ell = (V_\ell^T A^T U_\ell)^T = S_{\ell\ell}^T.$$

This concludes the proof of the theorem. $\square$

If the conditions on line 9 of Algorithm 1 hold in step $k = p$, then the value of d is increased to 1 in that step and we have $s_{p+1,p} = 0$. Furthermore, $s_{k+1,k} = 0$ for all $k \geq p$. Therefore, the (sub)matrix $S_{p:\ell,p:\ell} = [s_{ij}]_{i,j=p}^{\ell}$ is upper bidiagonal. By (2.5), the (sub)matrix $T_{p:\ell,p:\ell} = [t_{ij}]_{i,j=p}^{\ell}$ is lower bidiagonal. Hence, $t_{k,k+1} = 0$ for all $k \geq p$. The latter property should be incorporated on lines 10-11 of an implementation of Algorithm 1.

The computation of the decompositions (2.1) by Algorithm 1 requires the evaluation of 2ℓ matrix-vector products, similarly as the computation of the decompositions (1.2). The number of inner product evaluations in Algorithm 1 can be reduced by using the fact that $s_{jk} = t_{kj}$. For instance, on line 5, we may replace the evaluation of $s_{k-1,k}$ by $s_{k-1,k} := t_{k,k-1}$. Analogously, on line 10, we may use $t_{kk} := s_{kk}$ and $t_{k-1,k} := s_{k,k-1}$.

Classical Gram-Schmidt orthogonalization used in the description of Algorithm 1 may be replaced by modified Gram-Schmidt orthogonalization. For instance, we may replace lines 5-6 of the algorithm by

5a $\quad w := A^T u_k - t_{k,k-1} v_{k-1};$
5b $\quad$ *if* $d = 0$, *then* $s_{kk} := v_k^T w$; $w := w - s_{kk} v_k$ *endif*;
6 $\quad s_{k-d+1,k} := \|w\|;$

where we have used (2.5). We remark that one step of reorthogonalization of u_{k+1} against u_k and u_{k-1}, and of v_{k+1} against v_k and v_{k-1}, for all k, as described by Parlett [11, Section 6.9] may be beneficial.

Given the decompositions (2.1) with $u_1 = b/\|b\|$ and v_1 a user-chosen unit vector, we compute an approximate solution $x_\ell \in \mathcal{R}(V_\ell)$ of (1.1) by solving the minimization problem

$$\min_{x \in \mathcal{R}(V_\ell)} \|b - Ax\| = \min_{y \in \mathbb{R}^\ell} \|b - AV_\ell y\| = \min_{y \in \mathbb{R}^\ell} \|\beta_1 e_1 - T_{\ell+1,\ell} y\| \tag{2.7}$$

for y_ℓ and letting $x_\ell = V_\ell y_\ell$. This minimization problem is analogous to (1.5). We determine y_ℓ by using the QR-factorization of $T_{\ell+1,\ell}$, which can be computed cheaply with the aid of Givens rotations. The computations can be organized so that the iterates $x_\ell = V_\ell y_\ell$ can be updated with a recurrence formula that uses the last columns of V_ℓ only, similarly as in LSQR. Details are described in Subsection 2.1. We refer to the solution method obtained in this manner as Generalized LSQR (GLSQR).

The computations with Algorithm 1 terminate at steps 8 or 13. We refer to these terminations as breakdowns. The following theorems show that these breakdowns are benign, because the solution of (1.1) can be determined from the decompositions available at breakdown.

THEOREM 2.2. *Apply Algorithm 1 with $u_1 = b/\beta_1$ to the solution of (1.1). Let* $\text{rank}(A) = n$ *and assume that breakdown occurs in step k on line 13. Then, we have generated two orthonormal sequences $u_1, u_2, \ldots, u_k$ and $v_1, v_2, \ldots, v_{k-d+1}$ (with d having the value defined at the end of the computations), such that*

$$AV_k = U_k T_{kk}, \qquad A^T U_k = V_{k-d+1} S_{k-d+1,k}. \tag{2.8}$$

Furthermore, T_{kk} is nonsingular and $x = V_k T_{kk}^{-1} e_1 \|b\|$ solves $Ax = b$.

Proof. Application of the relations (2.4) of Theorem 2.1 to the vectors u_j and v_j generated before breakdown yields $AV_{k-1} = U_k T_{k,k-1}$ and $A^T U_k = V_{k-d+1} S_{k-d+1,k}$. In step k at line 13, we have

$$Av_k = t_{k-1,k} u_{k-1} + t_{kk} u_k.$$

Combining this equation with $AV_{k-1} = U_k T_{k,k-1}$ gives $AV_k = U_k T_{kk}$. This shows (2.8).

It follows from (2.8) that

$$\begin{aligned} V_k^T A^T A V_k &= V_k^T A^T U_k T_{kk} = V_k^T V_{k-d+1} S_{k-d+1,k} T_{kk} \\ &= S_{kk} T_{kk} = T_{kk}^T T_{kk}. \end{aligned}$$

Since A has full column rank, T_{kk} is nonsingular. Thus,

$$b - Ax = b - AV_k T_{kk}^{-1} e_1 \|b\| = b - U_k T_{kk} T_{kk}^{-1} e_1 \|b\| = 0.$$

$\square$

We note that if $d = 1$, then $\mathcal{R}(V_k)$ is an invariant subspace of $A^T A$ and $\mathcal{R}(U_k)$ is an invariant subspace of AA^T. However, if $d \neq 1$, then no invariant subspace has been found. The solution can be determined because the choice of v_1 leads to $x \in \mathcal{R}(V_k)$.

THEOREM 2.3. *Apply Algorithm 1 with $u_1 = b/\beta_1$ to the solution of (1.1). Assume that breakdown occurs in step $k+1$ on line 8. Then we have generated two orthonormal sequences $u_1, u_2, \ldots, u_{k+1}$ and $v_1, v_2, \ldots, v_k$, such that*

$$AV_k = U_{k+1} T_{k+1,k}, \qquad A^T U_{k+1} = V_k S_{k,k+1}. \tag{2.9}$$

Furthermore, $x = V_k y_k$ is a solution of (1.1), where y_k solves $\min_{y \in \mathbb{R}^k} \| \|b\| e_1 - T_{k+1,k} y \|$.

Proof. For future reference, we note that y_k satisfies

$$T_{k+1,k}^T (\|b\| e_1 - T_{k+1,k} y_k) = 0. \tag{2.10}$$

When breakdown occurs in step $k+1$ on line 8, we have $d = 1$ and

$$A^T u_{k+1} - s_{kk+1} v_k = 0. \tag{2.11}$$

No new vector v_{k+1} is generated in this step. However, Theorem 2.1 applied to step k shows that

$$AV_k = U_{k+1} T_{k+1,k}, \qquad A^T U_k = V_k S_{kk},$$

i.e., equation (2.4) holds with $\ell = k$ and $d = 1$. The second equation and (2.11) yield the right-hand side decomposition of (2.9).

Now, $A^T u_1 = A^T U_{k+1} e_1 = V_k S_{k,k+1} e_1$ and it follows from (2.9) and (2.10) that

$$\begin{aligned} A^T(b - Ax) &= A^T(b - AV_k y_k) \\ &= A^T(b - U_{k+1} T_{k+1,k} y_k) \\ &= \|b\| A^T u_1 - V_k S_{k,k+1} T_{k+1,k} y_k \\ &= \|b\| V_k S_{k,k+1} e_1 - V_k S_{k,k+1} T_{k+1,k} y_k \\ &= V_k S_{k,k+1} (\|b\| e_1 - T_{k+1,k} y_k) \\ &= V_k T_{k+1,k}^T (\|b\| e_1 - T_{k+1,k} y_k) = 0. \end{aligned}$$

Hence, x satisfies the normal equations associated with (1.1) and the proof is complete. $\square$

When the breakdown of Theorem 2.3 occurs, $\mathcal{R}(V_k)$ is an invariant subspace of A^TA and $A^Tb = \|b\|A^Tu_1 \in \mathcal{R}(V_k)$. Thus, the solution of (1.1) lives in $\mathcal{R}(V_k)$ and is found.

2.1. The GLSQR algorithm. We derive the recurrence relations for updating the solution $x_k = V_k y_k$ and the associated residual norm $\|b - Ax_k\|$ determined in step k of the GLSQR algorithm. The derivation parallels that for standard LSQR presented in [10]; it is included for completeness. For ease of comparison, we use the notation of [10].

Let $\mathcal{Q}_k = Q_{k,k+1} \dots Q_{2,3}Q_{1,2}$ be the product of the Givens rotations that transforms $T_{k+1,k}$ to upper triangular form, where $Q_{i,i+1}$ denotes the rotation acting on rows i and $i+1$. Thus, $Q_{i,i+1}$ is defined by

$$Q_{i,i+1}(i:i+1, i:i+1) = \begin{bmatrix} c_i & s_i \\ s_i & -c_i \end{bmatrix}.$$

We write

$$\mathcal{Q}_k T_{k+1,k} = \begin{array}{c} \\ k \\ 1 \end{array} \!\!\begin{array}{c} k \\ \begin{pmatrix} R_k \\ 0 \end{pmatrix} \end{array} = \begin{bmatrix} \rho_1 & \theta_2 & \gamma_3 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \rho_{k-2} & \theta_{k-1} & \gamma_k \\ & & & \rho_{k-1} & \theta_k \\ & & & & \rho_k \\ & & & & 0 \end{bmatrix} \tag{2.12}$$

and

$$\mathcal{Q}_k(\beta_1 e_1) = \begin{array}{c} \\ k \\ 1 \end{array} \!\!\begin{array}{c} 1 \\ \begin{pmatrix} f_k \\ \hat{\phi}_{k+1} \end{pmatrix} \end{array} = \begin{bmatrix} \phi_1 \\ \vdots \\ \phi_k \\ \hat{\phi}_{k+1} \end{bmatrix}.$$

These matrices are available at step k of GLSQR. Thus, $R_k \in \mathbb{R}^{k\times k}$ is an upper triangular banded matrix. The vector $y_k = R_k^{-1} f_k$ solves $\min_y \|T_{k+1,k}y - \|b\|e_1\|$ and $x_k = V_k y_k$ satisfies

$$\min_{x\in\mathcal{R}(V_k)} \|b - Ax\| = \|b - Ax_k\| = |\hat{\phi}_{k+1}|.$$

Define

$$D_k = [d_1, d_2, \cdots, d_k] = V_k R_k^{-1}.$$

The columns of D_k can be generated recursively by

$$d_k = (v_k - \theta_k d_{k-1} - \gamma_k d_{k-2})/\rho_k.$$

Let $R_{k-1} \in \mathbb{R}^{(k-1)\times(k-1)}$ be the upper triangular matrix obtained from (2.12) with k replaced by $k-1$. Thus, R_{k-1} is obtained by applying the product of Givens rotations $\mathcal{Q}_{k-1}$ to the matrix $T_{k,k-1}$. We note that R_{k-1} is related to R_k by

$$R_k = \begin{array}{c} \\ k-1 \\ 1 \end{array} \!\!\begin{array}{c} \begin{array}{cc} k-1 & 1 \end{array} \\ \begin{pmatrix} R_{k-1} & g_k \\ & \rho_k \end{pmatrix} \end{array}.$$

Similarly, $f_k = [f_{k-1}^T,\ \phi_k]^T$. Letting $D_{k-1} = V_{k-1}R_{k-1}^{-1}$, we obtain

$$D_k = V_k R_k^{-1} = [V_{k-1},\ v_k] \begin{bmatrix} R_{k-1}^{-1} & \tilde{g}_k \\ 0 & \rho_k^{-1} \end{bmatrix} = [D_{k-1},\ d_k]$$

and it follows that

$$\begin{aligned} x_k &= V_k y_k = V_k R_k^{-1} f_k \\ &= D_k f_k = D_{k-1} f_{k-1} + \phi_k d_k \\ &= x_{k-1} + \phi_k d_k. \end{aligned}$$

To complete the formulas, we need to determine the scalars $\gamma_k, \theta_k, \rho_k$, and ϕ_k. The first three scalars are obtained from

$$\begin{aligned} \mathcal{Q}_k T_{k+1,k} e_k &= Q_{k,k+1} Q_{k-1,k} Q_{k-2,k-1} \, [0\ \ldots\ 0\ t_{k-1,k}\ t_{kk}\ t_{k+1,k}]^T \\ &= Q_{k,k+1} Q_{k-1,k} \, [0\ \ldots\ \gamma_k\ \hat{\theta}_k\ t_{kk}\ t_{k+1,k}]^T \\ &= Q_{k,k+1} \, [0\ \ldots\ \gamma_k\ \theta_k\ \hat{\rho}_k\ t_{k+1,k}]^T \\ &= [0\ \ldots\ \gamma_k\ \theta_k\ \rho_k\ 0]^T, \end{aligned}$$

where

$$\begin{array}{lcllcl} \gamma_k & = & s_{k-2} t_{k-1,k}, & \hat{\theta}_k & = & -c_{k-2} t_{k-1,k}, \\ \theta_k & = & c_{k-1} \hat{\theta}_k + s_{k-1} t_{kk}, & \hat{\rho}_k & = & s_{k-1} \hat{\theta}_k - c_{k-1} t_{kk}, \\ \rho_k & = & c_k \hat{\rho}_k + s_k t_{k+1,k} = (\hat{\rho}_k^2 + t_{k+1,k}^2)^{1/2} & & & \end{array}$$

with $c_k = \hat{\rho}_k / \rho_k$ and $s_k = t_{k+1,k}/\rho_k$. Expressing $\mathcal{Q}_k(\beta_1 e_1)$ in a similar way yields

$$\phi_k = c_k \hat{\phi}_k, \qquad \hat{\phi}_{k+1} = s_k \hat{\phi}_k$$

with $\hat{\phi}_1 = \|b\|$. This completes the derivation of the recurrence formula for x_k.

Algorithm 2 below summarizes the computations required by GLSQR. The algorithm combines recursive updating of the iterates x_k with partial Lanczos bi-tridiagonalization. It is not necessary to store the full matrices V_k and $T_{k+1,k}$ in order to be able to compute x_k; only the last two columns v_{k-1} and v_k of V_k and the nontrivial entries of the last column of $T_{k+1,k}$ are required. At iteration k, Algorithm 2 stores v_k in v_0 and v_{k-1} in v_{-1}, as well as $t_{k-1,k}$ in τ_{-1} and t_{kk} in τ_0. Other required quantities are stored similarly. In order to save a scalar-vector multiplication, the algorithm uses the vector $w_i = \rho_i d_i$ instead of d_i.

Algorithm 2. Generalized LSQR:

Input: $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, *initial unit vector for solution subspace* $v_1 \in \mathbb{R}^n$;
Output: solution x_0, *residual norm* $|\hat{\phi}| = \|b - Ax_0\|$
Initialization: $d := 0$, $u_0 := v_0 := 0$, $\hat{\phi} := \|b\|$, $u_1 := b/\hat{\phi}$;
Initialization: $x_0 := 0$; $w_0 := w_{-1} := 0$, $\rho_0 := \rho_{-1} := 1$;
Initialization: $c_0 := c_{-1} := -1$; $s_0 := s_{-1} := 0$;
for $k := 1, 2, \ldots$ *until convergence*
    % shifting indices for storage
    $u_{-1} := u_0$, $u_0 := u_1$, $v_{-1} := v_0$, $v_0 := v_1$,
    $c_{-2} := c_{-1}$, $c_{-1} := c_0$, $s_{-2} := s_{-1}$, $s_{-1} := s_0$,
    $x_{-1} := x_0$, $w_{-2} := w_{-1}$, $w_{-1} := w_0$; $\rho_{-2} := \rho_{-1}$, $\rho_{-1} := \rho_0$,
    % Lanczos bi-tridiagonalization

$z := A^T u_0$, $\sigma_{-1} := v_{-1}^T z$; $z := z - \sigma_{-1} v_{-1}$;
if $d = 0$, *then* $\sigma_0 := v_0^T z$; $z := z - \sigma_0 v_0$ *endif;*
$\sigma_{-d+1} := \|z\|$;
if $\sigma_{-d+1} \neq 0$, *then* $v_{-d+1} := z/\sigma_{-d+1}$;
if $\sigma_{-d+1} = 0$ *and* $d = 1$, *then stop (least-squares solution found);*
if $\sigma_{-d+1} = 0$ *and* $d = 0$, *then* $d := 1$;
$z := A v_0$; $\tau_{-1} := u_{-1}^T z$; $z := z - \tau_{-1} u_{-1}$;
$\tau_0 := u_0^T z$; $z := z - \tau_0 u_0$; $\tau_1 := \|z\|$;
if $\tau_1 \neq 0$, *then* $u_1 := z/\tau_1$;
% updating the solution
$\gamma := s_{-2}\tau_{-1}$, $\theta := -c_{-1}c_{-2}\tau_{-1} + s_{-1}\tau_0$; $\hat{\rho} := -s_{-1}c_{-2}\tau_{-1} - c_{-1}\tau_0$;
$\rho_0 := (\hat{\rho}^2 + \tau_1^2)^{1/2}$; $c_0 := \hat{\rho}/\rho_0, s_0 := \tau_1/\rho_0$;
$\phi := c_0\hat{\phi}$; $\hat{\phi} := s_0\hat{\phi}$;
$w_0 := v_0 - \frac{\theta}{\rho_{-1}} w_{-1} - \frac{\gamma}{\rho_{-2}} w_{-2}$;
$x_0 := x_{-1} + (\phi/\rho_0) w_0$;
if $\tau_1 = 0$, *then stop (exact solution found);*
endfor

We conclude this section with some comments on the stability of Algorithm 2 in a finite precision arithmetic. The decompositions (2.4) are computed by recurrence formulas and are therefore satisfied to the order of machine precision, similarly as in other Lanczos-type algorithms. The solution y_k of $\min_y \|T_{k+1,k}y - \|b\|e_1\|$ is computed by QR-factorization of $T_{k+1,k}$ and therefore is backwards stable. The only stability problem that may arise is loss of orthogonality of the columns the matrices U_{k+1} and V_k. Equation (2.7) is not valid when the columns of the matrix U_{k+1} are not orthonormal. Instead, we have

$$\|b - Ax_k\| \approx \|U_{k+1}(\beta_1 e_1 - T_{k+1,k} y_k)\| \leq \|U_{k+1}\| |\hat{\phi}_k| \leq \sqrt{n} |\hat{\phi}_k|,$$

where we note that each column of U_{k+1} is of approximately unit length. The solution obtained might not be accurate, however, its residual norm is still bounded by $\sqrt{n}|\hat{\phi}_k|$. Note that $|\hat{\phi}_k|$ provides an estimate of the residual norm and can be used to decide when to terminate the iterations. Formulas for the quantities $\|A^T(b - Ax_k)\|$ and $\|x_k\|$ can be derived in the same manner; see [10] for details.

3. Numerical examples. This section illustrates the performance of GLSQR for different choices of the initial vector v_1 of the matrix V_ℓ in the decompositions (2.1). Both GLSQR and LSQR use the zero-vector as initial approximate solution in all examples. The computations were carried out on a PC in Matlab with machine epsilon $2 \cdot 10^{-16}$.

The first example is a modification of a linear discrete ill-posed problem from the program package Regularization Tools by Hansen [6]. Linear discrete ill-posed problems are linear systems of equations (1.1) with a matrix of ill-determined rank. The singular values of A "cluster" at the origin making the matrix very ill-conditioned and possibly singular. Linear discrete ill-posed problems arise, e.g., from the discretization of linear ill-posed problems, such as Fredholm integral equations of the first kind. The right-hand side b of linear discrete ill-posed problems that arise in applications often represents a measured quantity and therefore is contaminated by a measurement error $e \in \mathbb{R}^m$; see Hansen [7] for an introduction to linear discrete ill-posed problems.

Let $\hat{b} \in \mathbb{R}^m$ denote the error-free right-hand side associated with b, i.e.,

$$b = \hat{b} + e. \tag{3.1}$$

The vector $\hat{b}$ is assumed to be in $\mathcal{R}(A)$ and unknown. We would like to determine a solution $\hat{x}$ of the linear system of equations

$$Ax = \hat{b}. \tag{3.2}$$

Since $\hat{b}$ is not available, we apply a few, say ℓ, steps of GLSQR or LSQR to the available linear system (1.1) with contaminated right-hand side. Let ℓ_{opt} be the smallest integer, such that

$$\|x_{\ell_{\text{opt}}} - \hat{x}\| = \min_{\ell \geq 1} \|x_\ell - \hat{x}\|.$$

The difference $x_\ell - \hat{x}$ typically decreases as ℓ increases and $\ell \leq \ell_{\text{opt}}$, and increases with ℓ for $\ell \geq \ell_{\text{opt}}$. The latter depends on that the right-hand side in (1.1) is not $\hat{b}$. We report for GLSQR and LSQR the values of ℓ_{opt} and the relative errors $\|x_{\ell_{\text{opt}}} - \hat{x}\|/\|\hat{x}\|$, which measures the capability of the iterative methods to solve the discrete ill-posed problem.

When applied to "real" problems, ℓ_{opt} typically is not explicitly known and has to be estimated. Several approaches to determine an estimate of ℓ_{opt}, including the discrepancy principle and the L-curve, have been described in the literature, see, e.g., [2, 4, 7, 8, 9], and can be applied in conjunction with GLSQR. In applications to image restoration, ℓ_{opt} also may be determined through visual inspection.

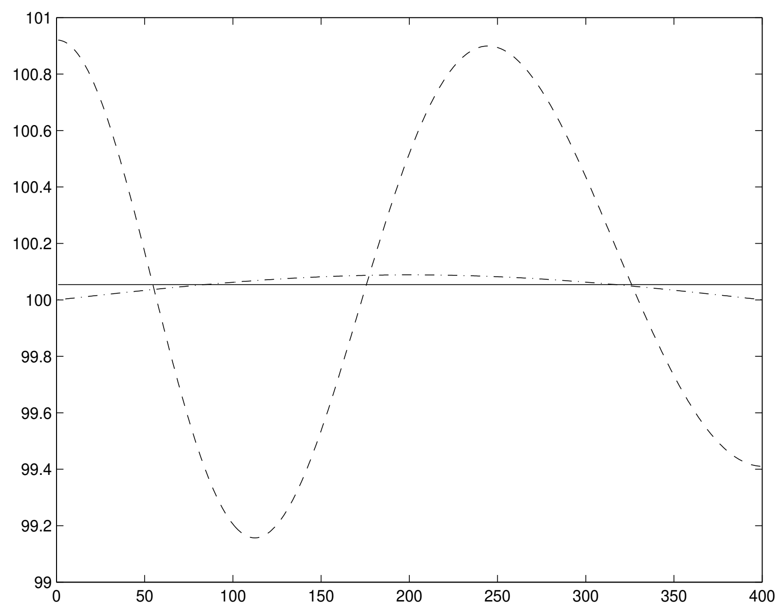

FIG. 3.1. *Example 3.1: Solution $\hat{x}$ of the error-free linear system (3.2) (dash-dotted curve), computed approximate solutions x_1^{GLSQR} (continuous curve) and x_3^{LSQR} (dashed curve).*

Example 3.1. Consider the Fredholm integral equation of the first kind,

$$\int_0^\pi \kappa(s,t)x(t)dt = b(s), \qquad 0 \leq s \leq \frac{\pi}{2}, \tag{3.3}$$

with $\kappa(s,t) = \exp(s\cos(t))$, $b(s) = 2\sinh(s)/s$, and solution $x(t) = \sin(t)$, discussed by Baart [1]. The Matlab function baart in [6] provides a discretization. We used baart to determine the matrix $A \in \mathbb{R}^{400\times 400}$ of the linear system of equations (1.1) and the solution $\tilde{x} \in \mathbb{R}^{400}$ of the discretized system, and define $\hat{x} = \tilde{x} + 100$ and $\hat{b} = A\hat{x}$. The matrix A so obtained is nonsymmetric and has many singular values close to the origin; it is numerically singular. We add an "error vector" $e \in \mathbb{R}^{400}$ with normally distributed random entries with zero mean, scaled to yield the noise level $\|e\|/\|\hat{b}\| = 1 \cdot 10^{-3}$, to $\hat{b}$ to obtain the noise-contaminated right-hand side vector b of (1.1), cf. (3.1).

Application of GLSQR with v_1 a normalization of the vector $[1, 1, \ldots, 1]^T \in \mathbb{R}^{400}$ to (1.1) gives the iterates x_ℓ^{GLSQR}, $\ell = 1, 2, 3, \ldots$, and $\ell_{\text{opt}} = 1$. LSQR applied to (1.1) yields the iterates x_ℓ^{LSQR}, $\ell = 1, 2, 3, \ldots$, and $\ell_{\text{opt}} = 3$. Figure 3.1 displays the desired solution $\hat{x}$ (dash-dotted curve), and the iterates x_1^{GLSQR} (continuous curve) and x_3^{LSQR} (dash-dotted curve). The figure shows GLSQR to give the highest accuracy; we have $\|x_1^{\text{GLSQR}} - \hat{x}\|/\|\hat{x}\| = 2.73 \cdot 10^{-4}$ and $\|x_3^{\text{LSQR}} - \hat{x}\|/\|\hat{x}\| = 5.83 \cdot 10^{-3}$. Thus, GLSQR yields higher accuracy with fewer iterations.

The reason for the poor performance of LSQR is that the constant function cannot be well approximated by vectors in low-dimensional Krylov subspaces determined by the method. Similar poor performance of LSQR, and good performance of GLSQR, also can be observed when solving many other Fredholm integral equations of the first kind, whose solutions are a small perturbation of a constant. More generally, when the desired solution is a small perturbation of a known vector, say w, then it may be beneficial to apply GLSQR with v_1 chosen to be a normalization of w.

Assume that we do not know at the beginning of the computations that the solution is a small perturbation of a constant. We then use LSQR and obtain the dashed curve of Figure 3.1. The graph shows the solution to oscillate around the value 100 and suggests the application of GLSQR with an initial vector of the solution subspace that represents a constant. □

Example 3.2. We consider the restoration of a discrete image that has been contaminated by blur and noise. The discrete "original" noise- and blur-free image, which is assumed to be unavailable, displays a "star cluster" with 470 "stars," which is shown in Figure 3.2. The image is represented by an array of 256×256 floating point numbers (pixels) in the interval $[2.6 \cdot 10^{-11}, 3.2 \cdot 10^4]$.[1] In order to avoid oversaturation in the displayed images, we truncate all pixel values p to $\max\{p, 500\}$. The pixel values, ordered row-wise, make up the vector $\hat{x} \in \mathbb{R}^n$ with $n = 256^2$.

The Matlab function blur from [6] with parameters band=6 and sigma=1 is applied to generate a blurring operator $A \in \mathbb{R}^{n\times n}$ that models spatially invariant Gaussian blur, a commonly used approximation of atmospheric blur. Specifically, $A\hat{x}$ models a convolution with a Gaussian kernel, and the vector $\hat{b} = A\hat{x}$ represents a blurred version of the "original" image $\hat{x}$. Let $e \in \mathbb{R}^n$ represent normally distributed random noise with zero mean, scaled to yield the noise level $\|e\|/\|\hat{b}\| = 1 \cdot 10^{-2}$. Adding e to $\hat{b}$ gives the vector $b \in \mathbb{R}^n$, cf. (3.1), which represents the blur- and noise-contaminated image of the star cluster shown in Figure 3.3. This image is assumed to be available.

The matrix A and right-hand side vector b defined in this manner determine the linear system of equations (1.1). Our task is to compute an approximation of the original image in Figure 3.2 by determining an approximate solution of (1.1). Figures 3.4 and 3.5 display the images represented by the iterates x_{40}^{LSQR} and x_{40}^{GLSQR} computed by 40 iterations with LSQR and GLSQR applied to (1.1), respectively. Both

[1] A file that represents this image can be obtained by ftp from the Space Telescope Science Institute address `ftp.stsci.edu`. It is in the directory `/pub/software/stsdas/testdata/restore/sims/star_cluster`.

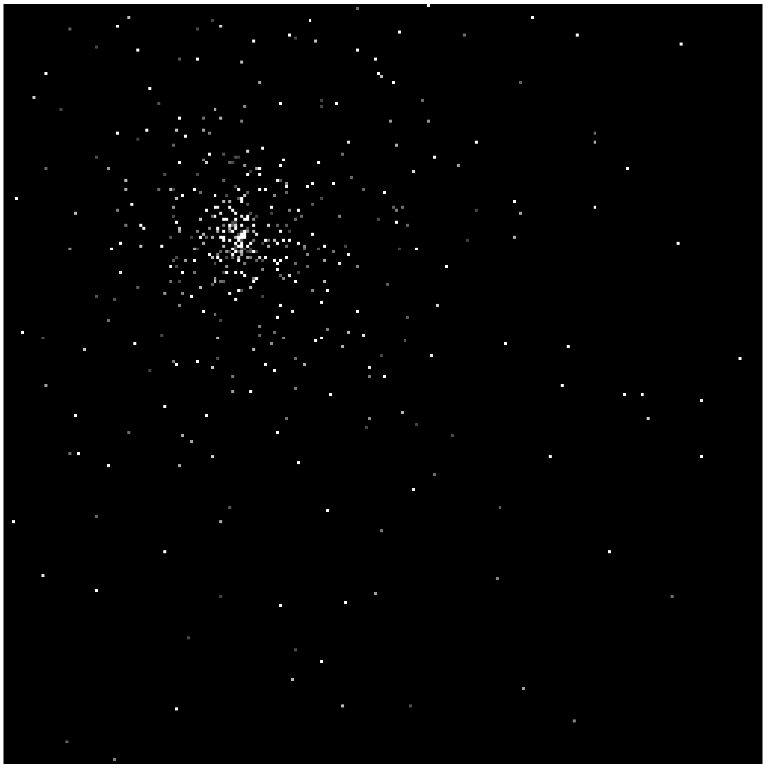

FIG. 3.2. *Example 3.2: Original blur- and noise-free image of "star cluster," assumed to be unavailable.*

methods use the initial approximate solution $x_0 = 0$, and GLSQR uses $v_1 = b/\|b\|$. The image determined by GLSQR is seen to be a better approximation of the original image than the image obtained with LSQR. Figure 3.6 displays the graphs for the errors $\|x_\ell^{\text{LSQR}} - \hat{x}\|/\|\hat{x}\|$ and $\|x_\ell^{\text{GLSQR}} - \hat{x}\|/\|\hat{x}\|$ as a function of the iteration number ℓ and shows LSQR to give a three times larger error than GLSQR in many of the corresponding iterates.

This example illustrates that GLSQR with the right-hand side b in the solution subspace may yield higher accuracy than LSQR when the matrix A represents a smoothing operator and the desired solution is not smooth.

We finally note that when $m \neq n$, the vector $v_1 = b/\|b\|$ cannot be used. In this situation, one can let $v_1 = Pb/\|Pb\|$, where P is a restriction or prolongation operator chosen so that $Pb \in \mathbb{R}^n$. $\square$

Example 3.3. We discuss the possibility of determining a suitable initial vector v_1 by solving a small system of equations determined by a coarse discretization of an integral equation. Consider the Fredholm integral equation of the first kind

$$\int_0^1 k(s,t)x(t)dt = \exp(s) + (1-e)s - 1, \qquad 0 \leq s \leq 1, \tag{3.4}$$

where

$$k(s,t) = \begin{cases} s(t-1), & s < t, \\ t(s-1), & s \geq t. \end{cases}$$

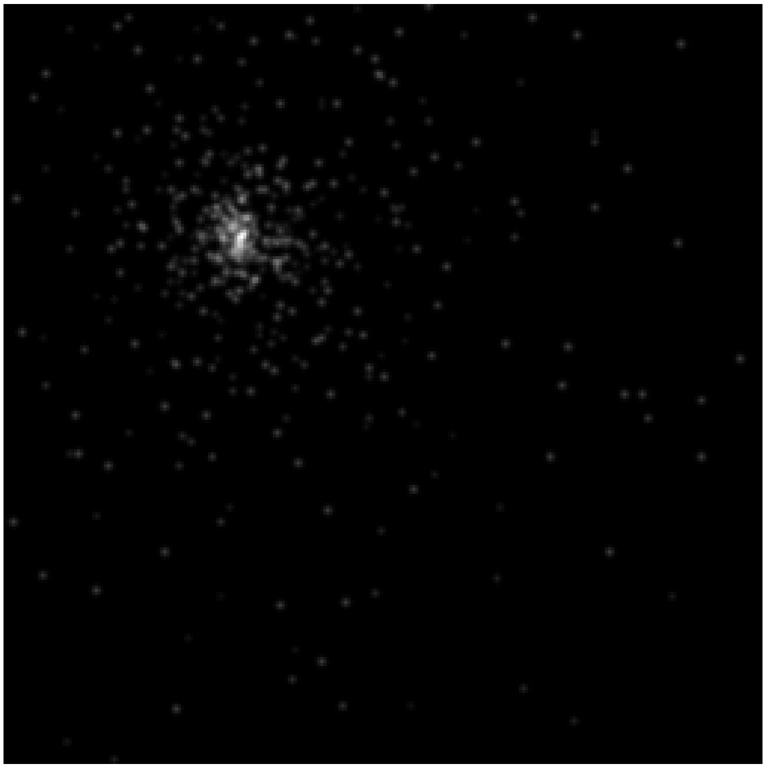

FIG. 3.3. *Example 3.2: Blur- and noise-contaminated image of "star cluster," assumed to be available.*

We discretize this integral equation by a Galerkin method with $n = 1024$ orthonormal box functions as test and trial functions using the Matlab program deriv2 from [6]. This yields the matrix $A \in \mathbb{R}^{n\times n}$. The code deriv2 also determines a scaled discrete approximation $\hat{x} \in \mathbb{R}^n$ of the solution $x(t) = \exp(t)$. The condition number of A, given by $\kappa(A) = \|A\|\|A^{-1}\|$, is $1.3 \cdot 10^6$. Figure 3.7 displays $\hat{x}$ (dash-dotted curve); the graph for $\hat{x}$ cannot be distinguished from the continuous curve of the figure. The error-free right-hand side vector is given by $\hat{b} = A\hat{x}$, and the right-hand side vector b in (1.1) is obtained by adding an error vector $e \in \mathbb{R}^n$ to b, cf. (3.1), where e has normally distributed random entries with zero mean, scaled to yield the noise level $\|e\|/\|\hat{b}\| = 1 \cdot 10^{-3}$. LSQR applied to (1.1) yields the iterates x_ℓ^{LSQR}, $\ell = 1, 2, 3, \ldots$, with $\ell_{\mathrm{opt}} = 22$ and $\|x_{22}^{\mathrm{LSQR}} - \hat{x}\|/\|\hat{x}\| = 1.32 \cdot 10^{-1}$. The dashed curve of Figure 3.7 shows x_{22}^{LSQR}.

We turn to the computation of an approximate solution of (1.1) by using a coarse-grid discretization. The code deriv2 is used to discretize (3.4) using a Galerkin method with only 4 box functions as test and trial functions. This gives the matrix $A_c \in \mathbb{R}^{4\times 4}$. We determine an associated right-hand side $b_c \in \mathbb{R}^4$ by projecting the available noisy right-hand side $b \in \mathbb{R}^n$ into $\mathbb{R}^4$ by local averaging. The small linear system of equations

$$A_c x = b_c \tag{3.5}$$

obtained in this manner is solved by a direct solution method without regularization. Regularization is not required because A_c is not ill-conditioned; we have $\kappa(A_c) =$

FIG. 3.4. *Example 3.2: Restored "star cluster" image represented by the vector* x_{40}^{LSQR}.

18. Prolongation of the solution $x_c \in \mathbb{R}^4$ of (3.5) by piecewise linear interpolation, using the Matlab function interp1, yields the approximation $x_{\mathrm{prl}} \in \mathbb{R}^n$ with error $\|x_{\mathrm{prl}} - \hat{x}\|/\|\hat{x}\| = 7.27 \cdot 10^{-3}$. GLSQR with $v_1 = x_{\mathrm{prl}}/\|x_{\mathrm{prl}}\|$ gives the approximate solutions x_ℓ^{GLSQR}, $\ell = 1, 2, 3, \ldots$, with $\ell_{\mathrm{opt}} = 4$ and $\|x_4^{\mathrm{GLSQR}} - \hat{x}\|/\|\hat{x}\| = 2.69 \cdot 10^{-3}$. The continuous curve of Figure 3.7 displays x_4^{GLSQR}. Thus, GLSQR determines a much better approximation of $\hat{x}$ than LSQR. □

The above example illustrates that it can be attractive to use an inexpensively computable coarse-grid solution to define the vector v_1 in GLSQR. However, there are several open questions, which we presently are investigating, such as how coarse the coarse grid can be chosen. Moreover, the approximate solution x_{prl} also can be used to improve the performance of LSQR. Let $r = b - Ax_{\mathrm{prl}}$ and apply LSQR to solve $Az = r$. Denote the computed iterates by z_ℓ^{LSQR}, $\ell = 1, 2, 3, \ldots$. Then $\widetilde{x}_\ell^{\mathrm{LSQR}} = x_{\mathrm{prl}} + z_\ell^{\mathrm{LSQR}}$, $\ell = 1, 2, 3, \ldots$, approximate $\hat{x}$. For A, b, and x_{prl} defined as in Example 3.3, we obtain $\min_{\ell \geq 1} \|\widetilde{x}_\ell^{\mathrm{LSQR}} - \hat{x}\|/\|\hat{x}\| = 5.11 \cdot 10^{-3}$, where the minimum is achieved for $\ell = 5$. Thus, $\widetilde{x}_5^{\mathrm{LSQR}}$ approximates $\hat{x}$ much better than x_{22}^{LSQR}, but not quite as well as x_4^{GLSQR}. These approaches to determine approximations of $\hat{x}$ are presently being investigated.

We remark that numerical experiments, which compare the decompositions (2.1) with the decompositions discussed by Golub and Kahan [5] for least-squares approximation, using the same initial vector for the solution subspaces, showed the former decomposition to provide more accurate solutions. Since Paige and Saunders [10] al-

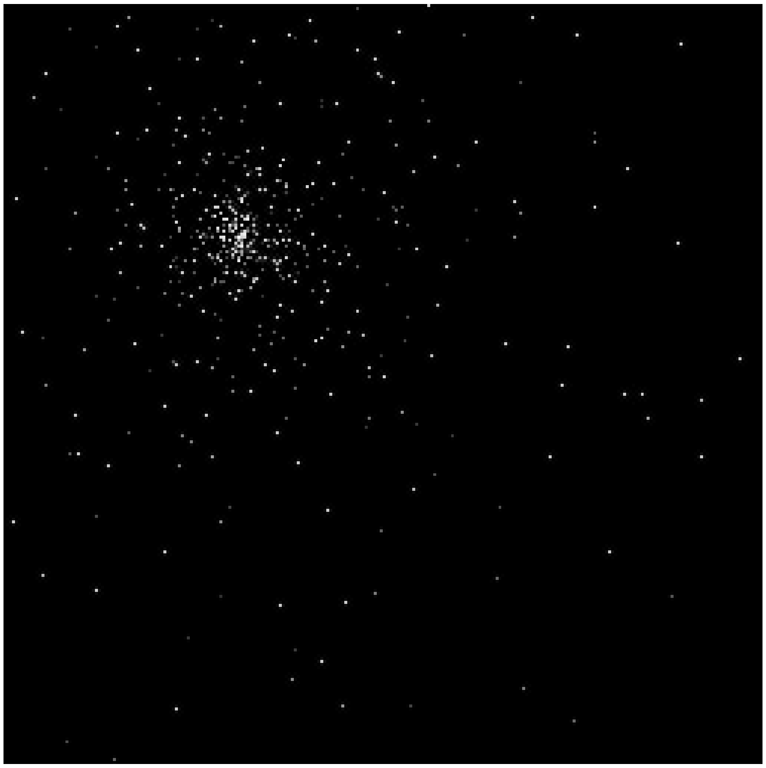

FIG. 3.5. *Example 3.2: Restored "star cluster" image represented by the vector* x_{40}^{GLSQR}.

ready pointed out that the decompositions in [5] may not perform well in the context of least-squares computations, we omit computed examples.

The examples of this section illustrate that modification of the subspace in which LSQR determines its iterates may yield higher accuracy or reduce the number of iterations required to compute an approximate solution with prescribed accuracy. The modification is beneficial when the initial vector of the solution subspace represents pertinent information about the desired solution. Knowledge about properties of the desired solution is helpful for determining a suitable initial vector. For instance, if the desired solution is known to have a jump discontinuity of unknown size at a known location, then the initial vector can be chosen to represent a step function. Examples 3.1-3.3 illustrate different approaches to determining a suitable initial vector when little a priori knowledge of the desired solution is available.

Acknowledgment. We would like to thank Bryan Lewis for discussions.

REFERENCES

[1] M. L. Baart, The use of auto-correlation for pseudo-rank determination in noisy ill-conditioned least-squares problems, IMA J. Numer. Anal., 2 (1982), pp. 241–247.
[2] D. Calvetti, B. Lewis, and L. Reichel, GMRES, L-curves and discrete ill-posed problems, BIT, 42 (2002), pp. 44–65.
[3] D. Calvetti, L. Reichel, and A. Shuibi, Enriched Krylov subspace methods for ill-posed problems, Linear Algebra Appl., 362 (2003), pp. 257–273.

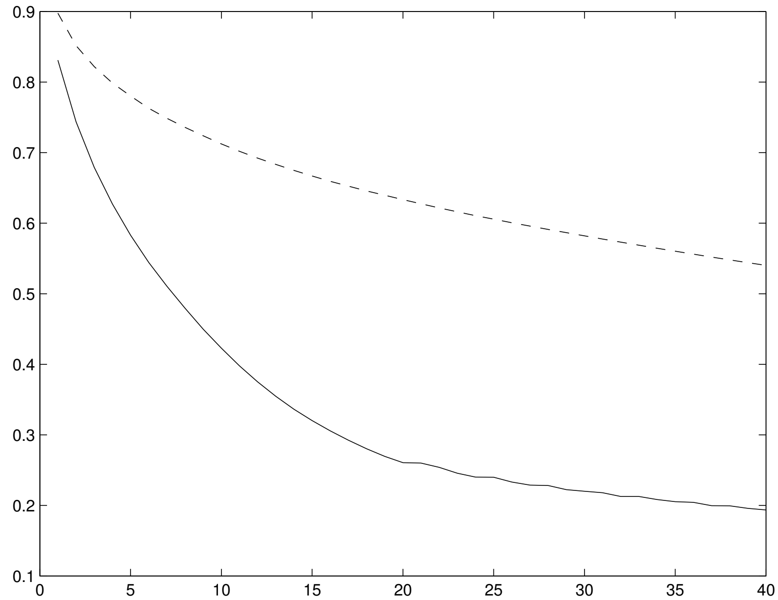

FIG. 3.6. *Example 3.2: The dashed curve shows the relative error* $\|x_\ell^{\mathrm{LSQR}} - \hat{x}\|/\|\hat{x}\|$ *for iterates determined by LSQR as a function of the iteration number* ℓ*, and the continuous curve displays the relative error* $\|x_\ell^{\mathrm{GLSQR}} - \hat{x}\|/\|\hat{x}\|$ *for iterates determined by GLSQR, for* $1 \leq \ell \leq 40$.

[4] J. L. Castellanos, S. Gómez, and V. Guerra, The triangle method for finding the corner of the L-curve, Appl. Numer. Math., 43 (2002), pp. 359–373.

[5] G. Golub and W. Kahan, Calculating the singular values and pseudo-inverse of a matrix, SIAM J. Numer. Anal., 2 (1965), pp. 205–224.

[6] P. C. Hansen, Regularization tools: A Matlab package for analysis and solution of discrete ill-posed problems, Numer. Algorithms, 6 (1994), pp. 1–35. Software is available in Netlib at `http://www.netlib.org`.

[7] P. C. Hansen, Rank-Deficient and Discrete Ill-Posed Problems, SIAM, Philadelphia, 1998.

[8] S. Morigi, L. Reichel, F. Sgallari, and F. Zama, Iterative methods for ill-posed problems and semiconvergent sequences, J. Comput. Appl. Math., 193 (2006), pp. 157–167.

[9] A. S. Nemirovskii, The regularization properties of the adjoint gradient method in ill-posed problems, USSR Comput. Math. Math. Phys., 26(2) (1986), pp. 7–16.

[10] 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.

[11] B. N. Parlett, The Symmetric Eigenvalue Problem, SIAM, Philadelphia, 1998.

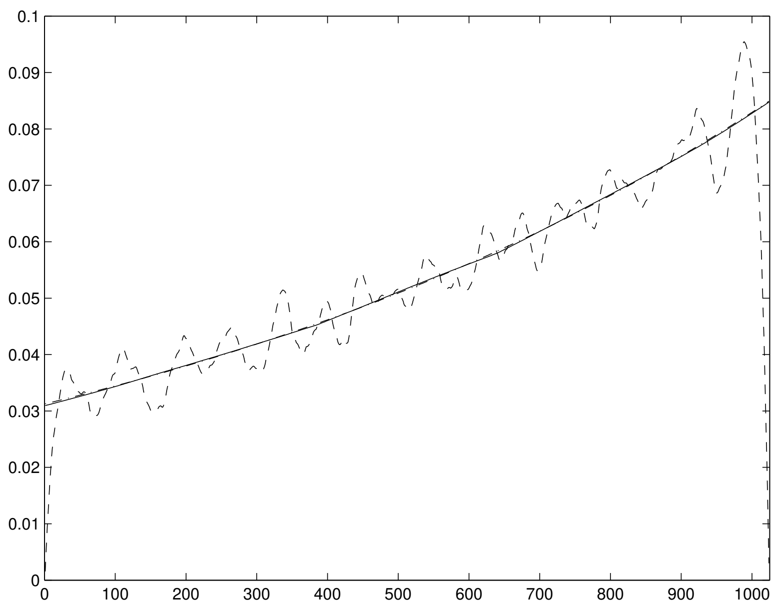

FIG. 3.7. *Example 3.3: Solution $\hat{x}$ of the error-free linear system (3.2) (dash-dotted curve), computed approximate solutions x_4^{GLSQR} (continuous curve) and x_{22}^{LSQR} (dashed curve).*