LSMR: AN ITERATIVE ALGORITHM FOR SPARSE LEAST-SQUARES PROBLEMS*

DAVID CHIN-LUNG FONG† AND MICHAEL SAUNDERS‡

Abstract. An iterative method LSMR is presented for solving linear systems $Ax = b$ and least-squares problem $\min \|Ax - b\|_2$, with A being sparse or a fast linear operator. LSMR is based on the Golub-Kahan bidiagonalization process. It is analytically equivalent to the MINRES method applied to the normal equation $A^TAx = A^Tb$, so that the quantities $\|A^Tr_k\|$ are monotonically decreasing (where $r_k = b - Ax_k$ is the residual for the current iterate x_k). In practice we observe that $\|r_k\|$ also decreases monotonically. Compared to LSQR, for which only $\|r_k\|$ is monotonic, it is safer to terminate LSMR early. Improvements for the new iterative method in the presence of extra available memory are also explored.

Key words. least-squares problem, sparse matrix, LSQR, MINRES, Krylov subspace method, Golub-Kahan process, conjugate-gradient method, minimum-residual method, iterative method

AMS subject classifications. 15A06, 65F10, 65F20, 65F22, 65F25, 65F35, 65F50, 93E24

DOI. xxx/xxxxxxxxx

1. Introduction.

We present a numerical method called LSMR for computing a solution x to the following problems:

Unsymmetric equations:	solve $Ax = b$
Linear least squares:	minimize $\|Ax - b\|_2$
Regularized least squares:	minimize $\left\| \begin{pmatrix} A \\ \lambda I \end{pmatrix} x - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $\lambda \geq 0$. The matrix A is used as an operator for which products of the form Av and A^Tu can be computed for various v and u. Thus A is normally large and sparse and need not be explicitly stored.

LSMR is similar in style to the well known method LSQR [11, 12] in being based on the Golub-Kahan bidiagonalization of A [4]. LSQR is equivalent to the conjugate-gradient (CG) method applied to the normal equation $(A^TA+\lambda^2I)x = A^Tb$. It has the property of reducing $\|r_k\|$ monotonically, where $r_k = b - Ax_k$ is the residual for the approximate solution x_k. (For simplicity, we are letting $\lambda = 0$.) In contrast, LSMR is equivalent to MINRES [10] applied to the normal equation, so that the quantities $\|A^Tr_k\|$ are monotonically decreasing. In practice we observe that $\|r_k\|$ also decreases monotonically, and is never very far behind the corresponding value for LSQR. Hence, although LSQR and LSMR ultimately converge to similar points, it is safer to use LSMR in situations where the solver must be terminated early.

Stopping conditions are typically based on *backward error*: the norm of some perturbation to A for which the current iterate x_k solves the perturbed problem exactly. Experiments on many sparse least-squares test problems show that for LSMR, a certain *cheaply computable* backward error for each x_k is close to the *optimal* (smallest possible) backward error. This is an unexpected but highly desirable advantage.

*Version of May 28, 2010. Technical Report SOL 2010-2
http://www.siam.org/journals/sisc/ for Copper Mountain Special Issue 2010

†iCME, Stanford University (clfong@stanford.edu). Partially supported by a Stanford Graduate Fellowship.

‡Systems Optimization Laboratory, Department of Management Science and Engineering, Stanford University, CA 94305-4026 (saunders@stanford.edu). Partially supported by Office of Naval Research grant N00014-08-1-0191 and by the U.S. Army Research Laboratory, through the Army High Performance Computing Research Center, Cooperative Agreement W911NF-07-0027.

1.1. Notation. Matrices are denoted by $A, B, \ldots$, vectors by $v, w, \ldots$, and scalars by $\alpha, \beta, \cdots$. Two exceptions are c and s, which denote the significant components of a plane rotation matrix, with $c^2 + s^2 = 1$. For a vector v, $\|v\|$ always denotes the 2-norm of v. For a matrix A, $\|A\|$ usually denotes the Frobenius norm, and the condition number of a matrix A is defined by $\text{cond}(A) = \|A\|\|A^+\|$, where A^+ denotes the pseudoinverse of A.

2. Derivation of LSMR.

2.1. Golub-Kahan process. We begin with the Golub-Kahan process [4].

1. Set $\beta_1 u_1 = b$ (shorthand for $\beta_1 = \|b\|$, $u_1 = b/\beta_1$) and $\alpha_1 v_1 = A^T u_1$.
2. For $k = 1, 2, \ldots$, set

$$\beta_{k+1} u_{k+1} = A v_k - \alpha_k u_k \quad \text{and} \quad \alpha_{k+1} v_{k+1} = A^T u_{k+1} - \beta_{k+1} v_k. \tag{2.1}$$

After k steps, we have

$$AV_k = U_{k+1} B_k \quad \text{and} \quad A^T U_{k+1} = V_{k+1} L_{k+1}^T,$$

where we define

$$L_k = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_k & \alpha_k \end{pmatrix}, \qquad B_k = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_k & \alpha_k \\ & & & \beta_{k+1} \end{pmatrix} = \begin{pmatrix} L_k \\ \beta_{k+1} e_k^T \end{pmatrix}.$$

Now consider

$$\begin{aligned} A^T A V_k = A^T U_{k+1} B_k &= (V_{k+1} L_{k+1}^T) B_k \\ &= V_{k+1} \begin{pmatrix} L_k^T & \beta_{k+1} e_k \\ 0 & \alpha_{k+1} \end{pmatrix} \begin{pmatrix} L_k \\ \beta_{k+1} e_k^T \end{pmatrix} \\ &= V_{k+1} \begin{pmatrix} L_k^T L_k + \beta_{k+1}^2 e_k e_k^T \\ \alpha_{k+1} \beta_{k+1} e_k^T \end{pmatrix} \\ &= V_{k+1} \begin{pmatrix} B_k^T B_k \\ \alpha_{k+1} \beta_{k+1} e_k^T \end{pmatrix}. \end{aligned}$$

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

2.2. Using Golub-Kahan to solve the normal equation. Krylov subspace methods for solving linear equations form solution estimates $x_k = V_k y_k$ for some y_k, where (as above) the columns of V_k are an expanding set of theoretically orthonormal vectors.

For the equation $A^T A x = A^T b$, any solution x has the property of minimizing $\|r\|$, where $r = b - Ax$ is the corresponding residual vector. Thus, in the development of LSQR it was natural to choose y_k to minimize $\|r_k\|$ at each stage. Since

$$r_k = b - AV_k y_k = \beta_1 u_1 - U_{k+1} B_k y_k = U_{k+1}(\beta_1 e_1 - B_k y_k),$$

where U_{k+1} is theoretically orthonormal, the subproblem $\min_{y_k} \|\beta_1 e_1 - B_k y_k\|$ easily arose. In contrast, for LSMR we wish to minimize $\|A^T r_k\|$. Let $\bar{\beta}_k = \alpha_k \beta_k$ for all k.

Since

$$\begin{aligned} A^T r_k = A^T b - A^T A x_k &= \beta_1 A^T u_1 - A^T A x_k \\ &= \beta_1 \alpha_1 v_1 - A^T A V_k y_k \\ &= \bar{\beta}_1 v_1 - V_{k+1} \begin{pmatrix} B_k^T B_k \\ \alpha_{k+1}\beta_{k+1} e_k^T \end{pmatrix} y_k \\ &= V_{k+1} \left(\bar{\beta}_1 e_1 - \begin{pmatrix} B_k^T B_k \\ \bar{\beta}_{k+1} e_k^T \end{pmatrix} y_k \right), \end{aligned}$$

we are led to the subproblem

$$\min_{y_k} \|A^T r_k\| = \min_{y_k} \left\| \bar{\beta}_1 e_1 - \begin{pmatrix} B_k^T B_k \\ \bar{\beta}_{k+1} e_k^T \end{pmatrix} y_k \right\|. \tag{2.2}$$

Efficient solution of this least-squares subproblem is the heart of algorithm LSMR.

2.3. Two QR factorizations. As in LSQR, we form the QR factorization

$$Q_{k+1} B_k = \begin{pmatrix} R_k \\ 0 \end{pmatrix}, \qquad R_k = \begin{pmatrix} \rho_1 & \theta_2 & & \\ & \rho_2 & \ddots & \\ & & \ddots & \theta_k \\ & & & \rho_k \end{pmatrix}. \tag{2.3}$$

If we define $t_k = R_k y_k$ and solve $R_k^T q_k = \bar{\beta}_{k+1} e_k$, we have $q_k = (\bar{\beta}_{k+1}/\rho_k) e_k = \varphi_k e_k$ with $\rho_k = (R_k)_{kk}$ and $\varphi_k = \bar{\beta}_{k+1}/\rho_k$. Then we perform a second QR factorization

$$\bar{Q}_{k+1} \begin{pmatrix} R_k^T & \bar{\beta}_1 e_1 \\ \varphi_k e_k^T & 0 \end{pmatrix} = \begin{pmatrix} \bar{R}_k & z_k \\ 0 & \bar{\zeta}_{k+1} \end{pmatrix}, \qquad \bar{R}_k = \begin{pmatrix} \bar{\rho}_1 & \bar{\theta}_2 & & \\ & \bar{\rho}_2 & \ddots & \\ & & \ddots & \bar{\theta}_k \\ & & & \bar{\rho}_k \end{pmatrix}. \tag{2.4}$$

Combining what we have gives

$$\begin{aligned} \min_{y_k} \left\| \bar{\beta}_1 e_1 - \begin{pmatrix} B_k^T B_k \\ \bar{\beta}_{k+1} e_k^T \end{pmatrix} y_k \right\| &= \min_{y_k} \left\| \bar{\beta}_1 e_1 - \begin{pmatrix} R_k^T R_k \\ q_k^T R_k \end{pmatrix} y_k \right\| \\ &= \min_{t_k} \left\| \bar{\beta}_1 e_1 - \begin{pmatrix} R_k^T \\ \varphi_k e_k^T \end{pmatrix} t_k \right\| \\ &= \min_{t_k} \left\| \begin{pmatrix} z_k \\ \bar{\zeta}_{k+1} \end{pmatrix} - \begin{pmatrix} \bar{R}_k \\ 0 \end{pmatrix} t_k \right\|. \end{aligned} \tag{2.5}$$

The subproblem is solved by choosing t_k from $\bar{R}_k t_k = z_k$.

2.4. Recurrence for x_k. Let W_k and $\bar{W}_k$ be computed by forward substitution from $R_k^T W_k^T = V_k^T$ and $\bar{R}_k^T \bar{W}_k^T = W_k^T$. Then from $x_k = V_k y_k$, $R_k y_k = t_k$, and $\bar{R}_k t_k = z_k$, we have

$$x_k = W_k R_k y_k = W_k t_k = \bar{W}_k \bar{R}_k t_k = \bar{W}_k z_k = x_{k-1} + \zeta_k \bar{w}_k.$$

2.5. Recurrence for W_k and $\bar{W}_k$. If we write

$$V_k = \begin{pmatrix} v_1 & v_2 & \cdots & v_k \end{pmatrix}, \qquad W_k = \begin{pmatrix} w_1 & w_2 & \cdots & w_k \end{pmatrix},$$
$$\bar{W}_k = \begin{pmatrix} \bar{w}_1 & \bar{w}_2 & \cdots & \bar{w}_k \end{pmatrix} \qquad z_k = \begin{pmatrix} \zeta_1 & \zeta_2 & \cdots & \zeta_k \end{pmatrix}^T,$$

an important fact is that when k increases to $k+1$, all quantities remain the same except for one additional term.

The first QR factorization proceeds as follows. At iteration k, we write

$$Q_{l,l+1} = \begin{pmatrix} I_{l-1} & & & \\ & c_l & s_l & \\ & -s_l & c_l & \\ & & & I_{k-l-1} \end{pmatrix}.$$

Now if $Q_{k+1} = Q_{k,k+1} \dots Q_{3,2} Q_{1,2}$, we have

$$Q_{k+1} B_{k+1} = Q_k \begin{pmatrix} B_k & \alpha_{k+1} e_{k+1} \\ & \beta_{k+2} \end{pmatrix} = \begin{pmatrix} R_k & \theta_{k+1} e_k \\ 0 & \bar{\alpha}_{k+1} \\ & \beta_{k+2} \end{pmatrix}$$
$$Q_{k+2} B_{k+1} = Q_{k+1,k+2} \begin{pmatrix} R_k & \theta_{k+1} e_k \\ 0 & \bar{\alpha}_{k+1} \\ & \beta_{k+2} \end{pmatrix} = \begin{pmatrix} R_k & \theta_{k+1} e_k \\ 0 & \rho_{k+1} \\ 0 & 0 \end{pmatrix}$$

and we see that $\theta_{k+1} = s_k \alpha_{k+1} = (\beta_{k+1}/\rho_k)\alpha_{k+1} = \bar{\beta}_{k+1}/\rho_k = \varphi_k$. Therefore we can now write θ_{k+1} instead of φ_k.

For the second QR factorization, if $\bar{Q}_{k+1} = \bar{Q}_{k,k+1} \dots \bar{Q}_{3,2} \bar{Q}_{1,2}$, we know that

$$\bar{Q}_{k+1} \begin{pmatrix} R_k^T \\ \theta_{k+1} e_k^T \end{pmatrix} = \begin{pmatrix} \bar{R}_k \\ 0 \end{pmatrix}.$$

Therefore we would have

$$\bar{Q}_{k+2} \begin{pmatrix} R_{k+1}^T \\ \theta_{k+2} e_{k+1}^T \end{pmatrix} = \bar{Q}_{k+1,k+2} \begin{pmatrix} \bar{R}_k & \bar{\theta}_{k+1} e_k \\ & \bar{c}_k \rho_{k+1} \\ & \theta_{k+2} \end{pmatrix} = \begin{pmatrix} \bar{R}_k & \bar{\theta}_{k+1} e_k \\ & \bar{\rho}_{k+1} \\ & 0 \end{pmatrix}.$$

By considering the last row of the matrix equation $R_{k+1}^T W_{k+1}^T = V_{k+1}^T$ we obtain

$$\theta_{k+1} w_k^T + \rho_{k+1} w_{k+1}^T = v_{k+1}^T,$$

and from the last row of the matrix equation $\bar{R}_{k+1}^T \bar{W}_{k+1}^T = W_{k+1}^T$ we obtain

$$\bar{\theta}_{k+1} \bar{w}_k^T + \bar{\rho}_{k+1} \bar{w}_{k+1}^T = w_{k+1}^T.$$

These equations serve to define w_{k+1} and $\bar{w}_{k+1}$.

2.6. The two rotations. To summarize, the rotations $Q_{k,k+1}$ and $\bar{Q}_{k,k+1}$ have the following effects on our computation:

$$\begin{pmatrix} c_k & s_k \\ -s_k & c_k \end{pmatrix} \begin{pmatrix} \bar{\alpha}_k & \\ \beta_{k+1} & \alpha_{k+1} \end{pmatrix} = \begin{pmatrix} \rho_k & \theta_{k+1} \\ 0 & \bar{\alpha}_{k+1} \end{pmatrix}$$
$$\begin{pmatrix} \bar{c}_k & \bar{s}_k \\ -\bar{s}_k & \bar{c}_k \end{pmatrix} \begin{pmatrix} \bar{c}_{k-1}\rho_k & & \bar{\zeta}_k \\ \theta_{k+1} & \rho_{k+1} & \end{pmatrix} = \begin{pmatrix} \bar{\rho}_k & \bar{\theta}_{k+1} & \zeta_k \\ 0 & \bar{c}_k \rho_{k+1} & \bar{\zeta}_{k+1} \end{pmatrix}.$$

2.7. Speeding up forward substitution. The forward substitutions for computing w and $\bar{w}$ can be made more efficient if we define $h_k = \rho_k w_k$ and $\bar{h}_k = \rho_k \bar{\rho}_k \bar{w}_k$. We then obtain the updates described in part 6 of the pseudo-codes below.

3. Algorithm LSMR. The following summarizes the main steps of algorithm LSMR for solving $Ax \approx b$, excluding the norm estimates and stopping rules developed later.

1. (Initialize)

$$\begin{array}{llll} \beta_1 u_1 = b & \alpha_1 v_1 = A^T u_1 & \bar{\alpha}_1 = \alpha_1 & \bar{\zeta}_1 = \alpha_1 \beta_1 \\ \rho_0 = 1 & \bar{\rho}_0 = 1 & \bar{c}_0 = 1 & \bar{s}_0 = 0 \\ h_1 = v_1 & \bar{h}_0 = \vec{0} & x_0 = \vec{0} & \end{array}$$

2. For $k = 1, 2, 3 \ldots$, repeat steps 3–6.
3. (Continue the bidiagonalization)

$$\begin{aligned} \beta_{k+1} u_{k+1} &= A v_k - \alpha_k u_k \\ \alpha_{k+1} v_{k+1} &= A^T u_{k+1} - \beta_{k+1} v_k \end{aligned}$$

4. (Construct and apply rotation $Q_{k,k+1}$)

$$\rho_k = \left(\bar{\alpha}_k^2 + \beta_{k+1}^2\right)^{\frac{1}{2}} \tag{3.1}$$

$$c_k = \bar{\alpha}_k / \rho_k \qquad s_k = \beta_{k+1} / \rho_k \tag{3.2}$$

$$\theta_{k+1} = s_k \alpha_{k+1} \qquad \bar{\alpha}_{k+1} = c_k \alpha_{k+1} \tag{3.3}$$

5. (Construct and apply rotation $\bar{Q}_{k,k+1}$)

$$\bar{\theta}_k = \bar{s}_{k-1} \rho_k \qquad \bar{\rho}_k = \left((\bar{c}_{k-1} \rho_k)^2 + \theta_{k+1}^2\right)^{\frac{1}{2}} \tag{3.4}$$

$$\bar{c}_k = \bar{c}_{k-1} \rho_k / \bar{\rho}_k \qquad \bar{s}_k = \theta_{k+1} / \bar{\rho}_k \tag{3.5}$$

$$\zeta_k = \bar{c}_k \bar{\zeta}_k \qquad \bar{\zeta}_{k+1} = -\bar{s}_k \bar{\zeta}_k \tag{3.6}$$

6. (Update h, $\bar{h}$ x)

$$\begin{aligned} \bar{h}_k &= h_k - (\bar{\theta}_k \rho_k / (\rho_{k-1} \bar{\rho}_{k-1})) \bar{h}_{k-1} \\ x_k &= x_{k-1} + (\zeta_k / (\rho_k \bar{\rho}_k)) \bar{h}_k \\ h_{k+1} &= v_{k+1} - (\theta_{k+1} / \rho_k) h_k \end{aligned}$$

4. Estimation of norms. Here we derive estimates of the quantities $\|r_k\|$, $\|A^T r_k\|$, $\|x_k\|$, $\|A\|$ for use within stopping rules.

4.1. Estimate of $\|A^T r_k\|$. One can see from (2.2) and (2.5) that $\|A^T r_k\| = |\bar{\zeta}_{k+1}|$, which can be computed at no additional cost and is monotonically decreasing.

4.2. Estimate of $\|r_k\|$. Here we transform $\bar{R}^T$ to upper-bidiagonal form using a third QR factorization: $\widetilde{R}_k = \widetilde{Q}_k \bar{R}_k^T$. This amounts to one additional rotation per iteration. Now let

$$\tilde{t}_k = \widetilde{Q}_k t_k, \qquad \tilde{b}_k = \begin{pmatrix} \widetilde{Q}_k & \\ & 1 \end{pmatrix} Q_{k+1} e_1 \beta_1.$$

Then we have

$$\begin{aligned} r_k &= b - Ax_k \\ &= \beta_1 u_1 - AV_k y_k \\ &= U_{k+1} e_1 \beta_1 - U_{k+1} B_k y_k \\ &= U_{k+1} \left(e_1 \beta_1 - Q_{k+1}^T \begin{pmatrix} R_k \\ 0 \end{pmatrix} y_k \right) \\ &= U_{k+1} \left(e_1 \beta_1 - Q_{k+1}^T \begin{pmatrix} t_k \\ 0 \end{pmatrix} \right) \\ &= U_{k+1} \left(Q_{k+1}^T \begin{pmatrix} \widetilde{Q}_k^T & \\ & 1 \end{pmatrix} \tilde{b}_k - Q_{k+1}^T \begin{pmatrix} \widetilde{Q}_k^T \tilde{t}_k \\ 0 \end{pmatrix} \right) \\ &= U_{k+1} Q_{k+1}^T \begin{pmatrix} \widetilde{Q}_k^T & \\ & 1 \end{pmatrix} \left(\tilde{b}_k - \begin{pmatrix} \tilde{t}_k \\ 0 \end{pmatrix} \right). \end{aligned}$$

Therefore, assuming orthogonality of U_{k+1}, we have

$$\|r_k\| = \left\| \tilde{b}_k - \begin{pmatrix} \tilde{t}_k \\ 0 \end{pmatrix} \right\|. \tag{4.1}$$

The vectors $\tilde{b}_k$ and $\tilde{t}_k$ can be written in the form

$$\tilde{b}_k = \begin{pmatrix} \tilde{\beta}_1 & \cdots & \tilde{\beta}_{k-1} & \dot{\beta}_k & \ddot{\beta}_{k+1} \end{pmatrix}^T \qquad \tilde{t}_k = \begin{pmatrix} \tilde{\tau}_1 & \cdots & \tilde{\tau}_{k-1} & \dot{\tau}_k \end{pmatrix}^T. \tag{4.2}$$

The vector $\tilde{t}_k$ can be computed by forward substitution from $\widetilde{R}_k^T \tilde{t}_k = z_k$.

4.2.1. Effects of the rotations. If we write

$$\widetilde{R}_k = \begin{pmatrix} \tilde{\rho}_1 & \tilde{\theta}_2 & & \\ & \ddots & \ddots & \\ & & \tilde{\rho}_{k-1} & \tilde{\theta}_k \\ & & & \dot{\rho}_k \end{pmatrix},$$

the effects of the rotations $Q_{k,k+1}$ and $\widetilde{Q}_{k-1,k}$ can be summarized as

$$\begin{pmatrix} c_k & s_k \\ -s_k & c_k \end{pmatrix} \begin{pmatrix} \ddot{\beta}_k \\ 0 \end{pmatrix} = \begin{pmatrix} \hat{\beta}_k \\ \ddot{\beta}_{k+1} \end{pmatrix},$$

$$\begin{pmatrix} \tilde{c}_k & \tilde{s}_k \\ -\tilde{s}_k & \tilde{c}_k \end{pmatrix} \begin{pmatrix} \dot{\rho}_{k-1} & & \dot{\beta}_{k-1} \\ \bar{\theta}_k & \bar{\rho}_k & \hat{\beta}_k \end{pmatrix} = \begin{pmatrix} \tilde{\rho}_{k-1} & \tilde{\theta}_k & \tilde{\beta}_{k-1} \\ 0 & \dot{\rho}_k & \dot{\beta}_k \end{pmatrix},$$

where $\ddot{\beta}_1 = \beta_1$, $\dot{\rho}_1 = \bar{\rho}_1$, $\dot{\beta}_1 = \hat{\beta}_1$ and c_k, s_k are defined in section 2.6.

4.2.2. Relationship between $\tilde{t}_k$ and $\tilde{b}_k$. We define $s^{(k)} = s_1 \cdots s_k$ and $\bar{s}^{(k)} = \bar{s}_1 \cdots \bar{s}_k$. Then from

$$\begin{aligned} \widetilde{R}_k^T \tilde{t}_k &= z_k \\ &= \begin{pmatrix} I_k & 0 \end{pmatrix} \bar{Q}_{k+1} e_{k+1} \bar{\beta}_1 \qquad \text{from (2.4)} \\ &= \begin{pmatrix} \bar{c}_1 \\ -\bar{s}_1 \bar{c}_2 \\ \bar{s}_1 \bar{s}_2 \bar{c}_3 \\ \vdots \\ (-1)^{k+1} \bar{s}^{(k-1)} \bar{c}_k \end{pmatrix} \bar{\beta}_1, \end{aligned}$$

we see that

$$\tilde{\tau}_1 = \tilde{\rho}_1^{-1}\bar{c}_1\bar{\beta}_1 \tag{4.3}$$
$$\tilde{\tau}_{k-1} = \tilde{\rho}_{k-1}^{-1}((-1)^k\bar{s}^{(k-2)}\bar{c}_{k-1}\bar{\beta}_1 - \tilde{\theta}_{k-1}\tilde{\tau}_{k-2}) \tag{4.4}$$
$$\dot{\tau}_k = \dot{\rho}_k^{-1}((-1)^{k+1}\bar{s}^{(k-1)}\bar{c}_k\bar{\beta}_1 - \tilde{\theta}_k\tilde{\tau}_{k-1}). \tag{4.5}$$

Also, from

$$\begin{pmatrix}\hat{\beta}_1\\ \vdots \\ \hat{\beta}_k \\ \ddot{\beta}_{k+1}\end{pmatrix} = Q_{k+1}e_{k+1}b_1 = \begin{pmatrix} c_1 \\ -s_1c_2 \\ \vdots \\ (-1)^{k+1}s^{(k-1)}c_k \\ (-1)^{k+2}s^{(k)}\end{pmatrix}\beta_1, \qquad \begin{pmatrix}\tilde{\beta}_1\\ \vdots \\ \tilde{\beta}_{k-1} \\ \dot{\beta}_k\end{pmatrix} = \widetilde{Q}_k\begin{pmatrix}\hat{\beta}_1\\ \vdots \\ \hat{\beta}_k\end{pmatrix}$$

we see that

$$\dot{\beta}_1 = \hat{\beta}_1 = c_1\beta_1 \tag{4.6}$$
$$\dot{\beta}_k = -\tilde{s}_{k-1}\dot{\beta}_{k-1} + \tilde{c}_{k-1}(-1)^{k-1}s^{(k-1)}c_k\beta_1 \tag{4.7}$$
$$\tilde{\beta}_k = \tilde{c}_k\dot{\beta}_k + \tilde{s}_k(-1)^k s^{(k)}c_{k+1}\beta_1. \tag{4.8}$$

We want to show by induction that $\tilde{\tau}_i = \tilde{\beta}_i$ for all i. When $i = 1$,

$$\begin{aligned}
\tilde{\beta}_1 &= \tilde{c}_1c_1\beta_1 - \tilde{s}_1s_1c_2\beta_1 \\
&= \tilde{\rho}_1^{-1}\beta_1(c_1\bar{\rho}_1 - \bar{\theta}_2s_1c_2) \\
&= \tilde{\rho}_1^{-1}\beta_1(c_1\bar{\rho}_1 - \bar{\theta}_2s_1\frac{c_1\alpha_2}{\rho_2}) \\
&= \tilde{\rho}_1^{-1}\beta_1c_1(\bar{\rho}_1 - \bar{\theta}_2s_1\frac{\alpha_2}{\rho_2}) \\
&= \tilde{\rho}_1^{-1}\beta_1\alpha_1\rho_1^{-1}(\bar{\rho}_1 - \frac{1}{\rho_2}\bar{\theta}_2(s_1\alpha_2)) \\
&= \tilde{\rho}_1^{-1}\bar{\beta}_1\rho_1^{-1}(\bar{\rho}_1 - \frac{1}{\rho_2}(\bar{s}_1\rho_2)\theta_2) \\
&= \tilde{\rho}_1^{-1}\bar{\beta}_1\rho_1^{-1}(\bar{\rho}_1 - \frac{\theta_2}{\bar{\rho}_1}\theta_2) \\
&= \tilde{\rho}_1^{-1}\bar{\beta}_1(\rho_1\bar{\rho}_1)^{-1}(\bar{\rho}_1^2 - \theta_2^2) \\
&= \tilde{\rho}_1^{-1}\bar{\beta}_1(\rho_1\bar{\rho}_1)^{-1}(\rho_1^2 + \theta_2^2 - \theta_2^2) \\
&= \tilde{\rho}_1^{-1}\bar{\beta}_1\frac{\rho_1}{\bar{\rho}_1} \\
&= \tilde{\rho}_1^{-1}\bar{\beta}_1\bar{c}_1 \\
&= \tilde{\tau}_1.
\end{aligned}$$

Suppose $\tilde{\tau}_k = \tilde{\beta}_k$. We consider the expression

$$\begin{aligned} s^{(k)} c_{k+1} \bar{\rho}_{k+1}^{-1} \bar{c}_k^2 \rho_{k+1}^2 \beta_1 &= \frac{\bar{c}_k \rho_{k+1}}{\bar{\rho}_{k+1}} (s^{(k)} c_{k+1}) \bar{c}_k \rho_{k+1} \beta_1 \\ &= \bar{c}_{k+1} \frac{\theta_2 \cdots \theta_{k+1} \alpha_1}{\rho_1 \cdots \rho_{k+1}} \frac{\rho_1 \cdots \rho_k}{\bar{\rho}_1 \cdots \bar{\rho}_k} \rho_{k+1} \beta_1 \\ &= \bar{c}_{k+1} \frac{\theta_2}{\bar{\rho}_1} \cdots \frac{\theta_{k+1}}{\bar{\rho}_k} \bar{\beta}_1 \\ &= \bar{c}_{k+1} \bar{s}_1 \cdots \bar{s}_k \bar{\beta}_1 \\ &= \bar{c}_{k+1} \bar{s}^{(k)} \bar{\beta}_1. \end{aligned} \tag{4.9}$$

Then we would have

$$\begin{aligned} \tilde{\tau}_{k+1} &= \tilde{\rho}_{k+1}^{-1} \left((-1)^{k+2} \bar{s}^{(k)} \bar{c}_{k+1} \bar{\beta}_1 - \tilde{\theta}_{k+1} \tilde{\tau}_k \right) \\ &= \tilde{\rho}_{k+1}^{-1} \left((-1)^{k+2} \bar{s}^{(k)} \bar{c}_{k+1} \bar{\beta}_1 - \tilde{\theta}_{k+1} \left(\tilde{c}_k \dot{\beta}_k + \tilde{s}_k (-1)^k s^{(k)} c_{k+1} \beta_1 \right) \right), \end{aligned}$$

with the last equality obtained by the induction hypothesis. Then we continue by rearranging terms and using (4.9) in the second equality below:

$$\begin{aligned} \tilde{\tau}_{k+1} &= \tilde{\rho}_{k+1}^{-1} \tilde{\theta}_{k+1} \tilde{c}_k \dot{\beta}_k + (-1)^{k+2} \tilde{\rho}_{k+1}^{-1} \left(\bar{s}^{(k)} \bar{c}_{k+1} \bar{\beta}_1 - \tilde{\theta}_{k+1} \tilde{s}_k s^{(k)} c_{k+1} \beta_1 \right) \\ &= \tilde{\rho}_{k+1}^{-1} (\bar{\rho}_{k+1} \tilde{s}_k) \tilde{c}_k \dot{\beta}_k + (-1)^{k+2} \tilde{\rho}_{k+1}^{-1} \left(s^{(k)} c_{k+1} \bar{\rho}_{k+1}^{-1} \bar{c}_k^2 \rho_{k+1}^2 \beta_1 - (\tilde{s}_k \bar{\rho}_{k+1}) \tilde{s}_k s^{(k)} c_{k+1} \beta_1 \right) \\ &= \frac{\tilde{c}_k \bar{\rho}_{k+1}}{\tilde{\rho}_{k+1}} \tilde{s}_k \dot{\beta}_k + (-1)^{k+2} \tilde{\rho}_{k+1}^{-1} s^{(k)} c_{k+1} \beta_1 \bar{\rho}_{k+1}^{-1} \left(\bar{c}_k^2 \rho_{k+1}^2 - \tilde{s}_k^2 \bar{\rho}_{k+1}^2 \right) \\ &= \tilde{c}_{k+1} \tilde{s}_k \dot{\beta}_k + (-1)^{k+2} \tilde{\rho}_{k+1}^{-1} s^{(k)} c_{k+1} \beta_1 \bar{\rho}_{k+1}^{-1} \left(\left(\bar{\rho}_{k+1}^2 - \theta_{k+2}^2 \right) - \tilde{s}_k^2 \bar{\rho}_{k+1}^2 \right) \\ &= \tilde{c}_{k+1} \tilde{s}_k \dot{\beta}_k + (-1)^{k+2} \tilde{\rho}_{k+1}^{-1} s^{(k)} c_{k+1} \beta_1 \bar{\rho}_{k+1}^{-1} \left(\bar{\rho}_{k+1}^2 (1 - \tilde{s}_k^2) - \theta_{k+2}^2 \right) \\ &= \tilde{c}_{k+1} \tilde{s}_k \dot{\beta}_k + (-1)^{k+2} \tilde{\rho}_{k+1}^{-1} s^{(k)} c_{k+1} \beta_1 \bar{\rho}_{k+1}^{-1} \left(\bar{\rho}_{k+1}^2 \tilde{c}_k^2 - \theta_{k+2}^2 \right) \\ &= \tilde{c}_{k+1} \tilde{s}_k \dot{\beta}_k + (-1)^{k+2} \tilde{\rho}_{k+1}^{-1} s^{(k)} \beta_1 \left(\bar{\rho}_{k+1} \tilde{c}_k^2 c_{k+1} - \frac{\theta_{k+2}}{\bar{\rho}_{k+1}} \theta_{k+2} c_{k+1} \right) \\ &= \tilde{c}_{k+1} \tilde{s}_k \dot{\beta}_k + (-1)^{k+2} \tilde{\rho}_{k+1}^{-1} s^{(k)} \beta_1 \left(\dot{\rho}_{k+1} \tilde{c}_k c_{k+1} - \frac{\theta_{k+2} \rho_{k+2}}{\bar{\rho}_{k+1}} s_{k+1} \alpha_{k+2} \frac{c_{k+1}}{\rho_{k+2}} \right) \\ &= \tilde{c}_{k+1} \tilde{s}_k \dot{\beta}_k + (-1)^{k+2} \tilde{\rho}_{k+1}^{-1} s^{(k)} \beta_1 \left(\dot{\rho}_{k+1} \tilde{c}_k c_{k+1} - \bar{\theta}_{k+2} s_{k+1} c_{k+2} \right) \\ &= \tilde{c}_{k+1} \tilde{s}_k \dot{\beta}_k + (-1)^{k+2} s^{(k)} \beta_1 \left(\tilde{c}_{k+1} \tilde{c}_k c_{k+1} - \tilde{s}_{k+1} s_{k+1} c_{k+2} \right) \\ &= \tilde{c}_{k+1} \left(-\tilde{s}_k \dot{\beta}_k + \tilde{c}_k (-1)^{k+2} s^{(k)} c_{k+1} \beta_1 \right) + \tilde{s}_{k+1} (-1)^{k+2} s^{(k+1)} c_{k+2} \beta_1 \\ &= \tilde{c}_{k+1} \dot{\beta}_{k+1} + \tilde{s}_{k+1} (-1)^{k+2} s^{(k+1)} c_{k+2} \beta_1 \\ &= \tilde{\beta}_{k+1}. \end{aligned}$$

Therefore by induction, we know that $\tilde{\tau}_i = \tilde{\beta}_i$ for $i = 1, 2, \ldots$. From (4.2), we see that at iteration k, the first $k-1$ elements of $\tilde{b}_k$ and $\tilde{t}_k$ are equal. Hence from (4.1), we can estimate $\|r_k\|$ from just the last two elements of $\tilde{b}_k$ and the last element of $\tilde{t}_k$, as shown in step 6 below.

4.2.3. Pseudo-code for estimating $\|r_k\|$. The following shows how $\|r_k\|$ may be estimated from quantities arising from the first and third QR factorizations.

1. (Initialize)

$$\ddot{\beta}_1 = \beta_1 \qquad \dot{\beta}_0 = 0 \qquad \dot{\rho}_0 = 1$$
$$\tilde{\tau}_{-1} = 0 \qquad \tilde{\theta}_0 = 0 \qquad \zeta_0 = 0$$

2. (For the kth iteration, repeat steps 3–6)
3. (Apply rotation $Q_{k,k+1}$)

$$\hat{\beta}_k = c_k \ddot{\beta}_k \qquad \ddot{\beta}_{k+1} = -s_k \ddot{\beta}_k \tag{4.10}$$

4. (If $k \geq 2$, construct and apply rotation $\widetilde{Q}_{k-1,k}$)

$$\tilde{\rho}_{k-1} = \left(\dot{\rho}_{k-1}^2 + \bar{\theta}_k^2\right)^{\frac{1}{2}} \tag{4.11}$$
$$\tilde{c}_{k-1} = \dot{\rho}_{k-1}/\tilde{\rho}_{k-1} \qquad \tilde{s}_{k-1} = \bar{\theta}_k/\tilde{\rho}_{k-1} \tag{4.12}$$
$$\tilde{\theta}_k = \tilde{s}_{k-1}\bar{\rho}_k \qquad \dot{\rho}_k = \tilde{c}_{k-1}\bar{\rho}_k \tag{4.13}$$
$$\tilde{\beta}_{k-1} = \tilde{c}_{k-1}\dot{\beta}_{k-1} + \tilde{s}_{k-1}\hat{\beta}_k \qquad \dot{\beta}_k = -\tilde{s}_{k-1}\dot{\beta}_{k-1} + \tilde{c}_{k-1}\hat{\beta}_k \tag{4.14}$$

5. (Update $\tilde{t}_k$ by forward substitution)

$$\tilde{\tau}_{k-1} = (\zeta_{k-1} - \tilde{\theta}_{k-1}\tilde{\tau}_{k-2})/\tilde{\rho}_{k-1}$$
$$\dot{\tau}_k = (\zeta_k - \tilde{\theta}_k\tilde{\tau}_{k-1})/\dot{\rho}_k$$

6. (Estimate $\|r_k\|$)

$$\|r_k\| = \left((\dot{\beta}_k - \dot{\tau}_k)^2 + \ddot{\beta}_{k+1}^2\right)^{\frac{1}{2}}$$

4.3. Estimate of $\|A\|$ and cond(A). It is known that the singular values of B_k are interlaced by those of A and are bounded above and below by the largest and smallest nonzero singular values of A [11]. Therefore we can estimate $\|A\|$ and cond(A) by $\|B_k\|$ and cond(B_k) respectively. Considering the Frobenius norm of B_k, we have the recurrence relation

$$\|B_{k+1}\|_F^2 = \|B_k\|_F^2 + \alpha_k^2 + \beta_{k+1}^2.$$

From (2.3)–(2.4), we know that the minimum and maximum singular values of $\bar{Q}_k R_k^T$ and B_k are approximately the same respectively [17]. Since $\bar{R}_k$ is upper triangular with positive diagonals,

$$\sigma_{\max}(B_k) \approx \max\big(\max_{1\leq j\leq k-1}(\bar{\rho}_j), \bar{c}_{k-1}\rho_k\big),$$
$$\sigma_{\min}(B_k) \approx \min\big(\min_{1\leq j\leq k-1}(\bar{\rho}_j), \bar{c}_{k-1}\rho_k\big).$$

This gives us the approximation cond$(A) \approx \sigma_{\max}(B_k)/\sigma_{\min}(B_k)$, which can be obtained in constant time per iteration.

4.4. Estimate of $\|x_k\|$. From the definition in section 2.4, we have the relation $x_k = V_k R_k^{-1}\bar{R}_k^{-1} z_k$. From the third QR factorization $\widetilde{Q}_k\bar{R}^T = \widetilde{R}_k$ in section 4.2 and a fourth QR factorization $\hat{Q}_k(\widetilde{Q}_k R_k)^T = \hat{R}_k$ we can write

$$\begin{aligned} x_k &= V_k R_k^{-1}\bar{R}_k^{-1} z_k \\ &= V_k R_k^{-1}\bar{R}_k^{-1}\bar{R}_k\widetilde{Q}_k^T\tilde{z}_k \\ &= V_k R_k^{-1}\widetilde{Q}_k^T\widetilde{Q}_k R_k\hat{Q}_k^T\hat{z}_k \\ &= V_k\hat{Q}_k^T\hat{z}_k, \end{aligned}$$

where $\tilde{z}_k$ and $\hat{z}_k$ are defined by forward substitutions $\widetilde{R}_k^T \tilde{z}_k = z_k$ and $\hat{R}_k \hat{z}_k = \tilde{z}_k$. Then assuming orthogonality of V_k, we arrive at the estimate $\|x_k\| = \|\hat{z}_k\|$. Note that since only the lower-rightmost entries in R_k and $\bar{R}_k$ change each iteration, this estimate of $\|x_k\|$ requires only a constant number of multiplications per iteration. The pseudo-code, omitted here, can be derived as in section 4.2.3.

5. Stopping criteria. With exact arithmetic, the Golub-Kahan process terminates when either $\alpha_{k+1} = 0$ or $\beta_{k+1} = 0$. For certain data b, this could happen in practice when k is small (but is unlikely later). We show that LSMR will have solved the problem at that point and should therefore terminate.

When $\alpha_{k+1} = 0$, we have

$$\begin{aligned}
\|A^T r_k\| &= \bar{\zeta}_{k+1} && \text{(from sec. 4.1)} \\
&= -\bar{s}_k \bar{\zeta}_k && \text{(from (3.6))} \\
&= -\theta_{k+1} \bar{\rho}_k^{-1} \bar{\zeta}_k && \text{(from (3.5))} \\
&= -s_k \alpha_{k+1} \bar{\rho}_k^{-1} \bar{\zeta}_k && \text{(from (3.3))} \\
&= 0.
\end{aligned}$$

Thus, a least-squares solution has been obtained.

When $\beta_{k+1} = 0$, we have

$$\begin{aligned}
s_k &= \beta_{k+1} \rho_k^{-1} && \text{(from (3.2))} \\
&= 0. && (5.1) \\
\ddot{\beta}_{k+1} &= -s_k \ddot{\beta}_k && \text{(from (4.10))} \\
&= 0. && \text{(from (5.1))} \quad (5.2) \\
\dot{\tau}_k &= \dot{\rho}_k^{-1} \tilde{\rho}_k \tilde{\tau}_k && \text{(from (4.4), (4.5))} \quad (5.3) \\
\dot{\beta}_k &= \tilde{c}_k^{-1} \left(\tilde{\beta}_k - \tilde{s}_k (-1)^k s^{(k)} c_{k+1} \beta_1 \right) && \text{(from (4.8))} \\
&= \tilde{c}_k^{-1} \tilde{\beta}_k && \text{(from (5.1))} \\
&= \dot{\rho}_k^{-1} \tilde{\rho}_k \tilde{\beta}_k && \text{(from (4.12))} \\
&= \dot{\rho}_k^{-1} \tilde{\rho}_k \tilde{\tau}_k && \text{(from sec. 4.2.2)} \\
&= \dot{\tau}_k. && \text{(from (5.3))} \quad (5.4)
\end{aligned}$$

Therefore, by equation (5.2) and (5.4), we conclude that

$$\|r_k\| = \left((\dot{\beta}_k - \dot{\tau}_k)^2 + \ddot{\beta}_{k+1}^2 \right)^{\frac{1}{2}} = 0.$$

It follows that the system is compatible and we have solved $Ax = b$.

5.1. Practical stopping criteria. In practice, the stopping rules in LSQR [11] are used for LSMR. Three dimensionless quantities are needed: ATOL, BTOL, CONLIM. The first stopping rule applies to compatible systems, the second rule applies to incompatible systems, and the third rule applies to both.

S1: Stop if $\|r_k\| \leq \text{BTOL}\|b\| + \text{ATOL}\|A\|\|x_k\|$

S2: Stop if $\frac{\|A^T r_k\|}{\|A\|\|r_k\|} \leq \text{ATOL}$

S3: Stop if $\text{cond}(A) \geq \text{CONLIM}$

6. Characteristics of solution on singular systems. The least-squares problem $\min \|Ax - b\|$ has a unique solution when A has full column rank. If A does not have full column rank, there are many distinct x that give the same minimum value of $\|Ax - b\|$. In particular, the corresponding normal equation $A^TAx = A^Tb$ is a singular system. Here we show that both LSQR and LSMR give the minimum-norm least-squares solution at convergence. That is, both LSQR and LMSR solve the optimization problem

$$\min_x \|x\|_2 \quad \text{such that} \quad A^TAx = A^Tb.$$

Let $\mathrm{N}(A)$ and $\mathrm{R}(A)$ denote the nullspace and range of a matrix A.

LEMMA 6.1. *If $A \in \mathbb{R}^{m\times n}$ and $p \in \mathbb{R}^n$ satisfy $A^TAp = 0$, then $p \in N(A)$.*

Proof. $A^TAp = 0 \Rightarrow p^TA^TAp = 0 \Rightarrow (Ap)^TAp = 0 \Rightarrow Ap = 0$. ∎

THEOREM 6.2. *The converged solution returned by LSQR on a least-squares system is the minimum-norm solution.*

Proof. Let x_k^{LSQR} be the solution returned by LSQR on $\min \|Ax - b\|$ at convergence; i.e., x_k^{LSQR} satisfies

$$A^TAx_k^{\mathrm{LSQR}} = A^Tb. \tag{6.1}$$

Since the solution lies in the Krylov subspace, we also have

$$x_k^{\mathrm{LSQR}} = V_k y_k^{\mathrm{LSQR}} \tag{6.2}$$

for some y_k^{LSQR}. Consider any other solution to the least-squares system; i.e., $\widehat{x}$ satisfying

$$A^TA\widehat{x} = A^Tb. \tag{6.3}$$

Let $p = \widehat{x} - x_k^{\mathrm{LSQR}}$. The difference between (6.3) and (6.1) gives $A^TAp = 0$, so that $Ap = 0$ by Lemma 6.1. From the Golub-Kahan process, $\alpha_1 v_1 = A^Tu_1$ and $\alpha_{k+1}v_{k+1} = A^Tu_{k+1} - \beta_{k+1}v_k$, we know that $v_1, \ldots, v_k \in \mathrm{R}(A^T)$. With $Ap = 0$, this implies

$$p^TV_k = 0. \tag{6.4}$$

Now we consider

$$\begin{aligned}
\|\widehat{x}\|_2^2 - \|x_k^{\mathrm{LSQR}}\|_2^2 &= \|x_k^{\mathrm{LSQR}} + p\|_2^2 - \|x_k^{\mathrm{LSQR}}\|_2^2 \\
&= p^Tp + 2p^Tx_k^{\mathrm{LSQR}} \\
&= p^Tp + 2p^TV_k y_k^{\mathrm{LSQR}} && \text{by (6.2)} \\
&= p^Tp && \text{by (6.4)} \\
&\geq 0,
\end{aligned}$$

which shows that x_k^{LSQR} has minimum norm among all possible solutions. ∎

COROLLARY 6.3. *The converged solution returned by LSMR on a least-squares system is the minimum-norm solution.*

Proof. At convergence, $\alpha_{k+1} = 0$ or $\beta_{k+1} = 0$. Thus $\bar{\beta}_{k+1} = \alpha_{k+1}\beta_{k+1} = 0$, which means equation (2.5) becomes

$$\min_{y_k} \left\|\bar{\beta}_1 e_1 - B_k^TB_k y_k\right\| = 0 \qquad \Rightarrow \qquad B_k^TB_k y_k = \bar{\beta}_1 e_1,$$

since B_k has full rank. This is the normal equation for $\min_{y_k} \|B_k y_k - \beta_1 e_1\|$, the same least-squares subproblem solved by LSQR. We conclude that at convergence, $y_k = y_k^{\text{LSQR}}$ and thus $x_k = V_k y_k = V_k y_k^{\text{LSQR}} = x_k^{\text{LSQR}}$. By Theorem 6.2, LSMR converges to the minimum-norm solution. □

7. Complexity. We compare the storage requirement and computational complexity for LSMR and LSQR on $Ax \approx b$ and MINRES on the normal equation $A^TAx = A^Tb$. In Table 7.1, we list the vectors needed during each iteration (excluding storage for A and b). Recall that A is $m \times n$ and for least-squares systems m may be considerably larger than n. Av denote the working storage for matrix-vector products. Work represents the number of floating-point multiplications required at each iteration.

TABLE 7.1
Storage and computational requirements for various least-squares methods

	Storage		Work	
	m	n	m	n
LSMR	Av, u	$x, v, h, \bar{h}$	3	6
LSQR	Av, u	x, v, w	3	5
MINRES on $A^TAx = A^Tb$	Av_1	$x, v_1, v_2, w_1, w_2, w_3$		8

8. Regularized least squares. In this section, we extend LSMR to the regularized least-squares problem:

$$\min \left\| \begin{pmatrix} A \\ \lambda I \end{pmatrix} x - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2. \tag{8.1}$$

If $\bar{A} = \begin{pmatrix} A \\ \lambda I \end{pmatrix}$ and $\bar{r}_k = \begin{pmatrix} b \\ 0 \end{pmatrix} - \bar{A}x_k$, then

$$\begin{aligned}
\bar{A}^T\bar{r}_k &= A^Tr_k - \lambda^2 x_k \\
&= V_{k+1}\left(\bar{\beta}_1 e_1 - \begin{pmatrix} B_k^TB_k \\ \bar{\beta}_{k+1}e_k^T \end{pmatrix} y_k - \lambda^2 \begin{pmatrix} y_k \\ 0 \end{pmatrix}\right) \\
&= V_{k+1}\left(\bar{\beta}_1 e_1 - \begin{pmatrix} B_k^TB_k + \lambda^2 I \\ \bar{\beta}_{k+1}e_k^T \end{pmatrix} y_k\right) \\
&= V_{k+1}\left(\bar{\beta}_1 e_1 - \begin{pmatrix} R_k^TR_k \\ \bar{\beta}_{k+1}e_k^T \end{pmatrix} y_k\right)
\end{aligned}$$

and the rest of the main algorithm follows the same as in the unregularized case. In the last equality, R_k is defined by the QR factorization

$$Q_{2k+1}\begin{pmatrix} B_k \\ \lambda I \end{pmatrix} = \begin{pmatrix} R_k \\ 0 \end{pmatrix},$$

where Q_{2k+1} has the form $Q_{2k+1} = Q_{k,k+1}\hat{Q}_{k,2k+1}\cdots Q_{2,3}\hat{Q}_{2,k+3}Q_{1,2}\hat{Q}_{1,k+2}$. The effects of $\hat{Q}_{1,k+2}$ and $Q_{1,2}$ are illustrated here:

$$\hat{Q}_{1,k+2}\begin{pmatrix} \alpha_1 & \\ \beta_2 & \alpha_2 \\ & \beta_3 \\ \lambda & \\ & \lambda \end{pmatrix} = \begin{pmatrix} \hat{\alpha}_1 & \\ \beta_2 & \alpha_2 \\ & \beta_3 \\ 0 & \\ & \lambda \end{pmatrix}, \qquad Q_{1,2}\begin{pmatrix} \hat{\alpha}_1 & \\ \beta_2 & \alpha_2 \\ & \beta_3 \\ & \\ & \lambda \end{pmatrix} = \begin{pmatrix} \rho_1 & \theta_2 \\ & \bar{\alpha}_2 \\ & \beta_3 \\ & \\ & \lambda \end{pmatrix}.$$

8.1. Effects on estimation of $\|\bar{r}_k\|$. The introduction of regularization changes the estimate of $\|\bar{r}_k\|$ as follows:

$$\begin{aligned}
\bar{r}_k &= \begin{pmatrix} b \\ 0 \end{pmatrix} - \begin{pmatrix} A \\ \lambda I \end{pmatrix} x_k \\
&= \begin{pmatrix} u_1 \\ 0 \end{pmatrix} \beta_1 - \begin{pmatrix} AV_k \\ \lambda V_k \end{pmatrix} y_k \\
&= \begin{pmatrix} u_1 \\ 0 \end{pmatrix} \beta_1 - \begin{pmatrix} U_{k+1}B_k \\ \lambda V_k \end{pmatrix} y_k \\
&= \begin{pmatrix} U_{k+1} & \\ & V_k \end{pmatrix} \left(e_1\beta_1 - \begin{pmatrix} B_k \\ \lambda I \end{pmatrix} y_k \right) \\
&= \begin{pmatrix} U_{k+1} & \\ & V_k \end{pmatrix} \left(e_1\beta_1 - Q_{2k+1}^T \begin{pmatrix} R_k \\ 0 \end{pmatrix} y_k \right) \\
&= \begin{pmatrix} U_{k+1} & \\ & V_k \end{pmatrix} \left(e_1\beta_1 - Q_{2k+1}^T \begin{pmatrix} t_k \\ 0 \end{pmatrix} \right) \\
&= \begin{pmatrix} U_{k+1} & \\ & V_k \end{pmatrix} Q_{2k+1}^T \begin{pmatrix} \widetilde{Q}_k^T & \\ & 1 \end{pmatrix} \left(\tilde{b}_k - \begin{pmatrix} \tilde{t}_k \\ 0 \end{pmatrix} \right)
\end{aligned}$$

with $\tilde{b}_k = \begin{pmatrix} \widetilde{Q}_k & \\ & 1 \end{pmatrix} Q_{2k+1} e_1 \beta_1$, where we adopt the notation

$$\tilde{b}_k = \begin{pmatrix} \tilde{\beta}_1 & \cdots & \tilde{\beta}_{k-1} & \dot{\beta}_k & \ddot{\beta}_{k+1} & \check{\beta}_1 & \cdots & \check{\beta}_k \end{pmatrix}^T.$$

Then we conclude

$$\|r_k\| = \check{\beta}_1^2 + \cdots + \check{\beta}_k^2 + (\dot{\beta}_k - \tau_k)^2 + \ddot{\beta}_{k+1}^2.$$

The effect of regularization on the rotations is summarized as

$$\begin{pmatrix} \hat{c}_k & \hat{s}_k \\ -\hat{s}_k & \hat{c}_k \end{pmatrix} \begin{pmatrix} \bar{\alpha}_k & \ddot{\beta}_k \\ \lambda & \end{pmatrix} = \begin{pmatrix} \hat{\alpha}_k & \acute{\beta}_k \\ & \check{\beta}_k \end{pmatrix}$$

$$\begin{pmatrix} c_k & s_k \\ -s_k & c_k \end{pmatrix} \begin{pmatrix} \hat{\alpha}_k & & \acute{\beta}_k \\ \beta_{k+1} & \alpha_{k+1} & \end{pmatrix} = \begin{pmatrix} \rho_k & \theta_{k+1} & \hat{\beta}_k \\ & \bar{\alpha}_{k+1} & \ddot{\beta}_{k+1} \end{pmatrix}.$$

8.2. Pseudo-code for regularized LSMR. The following summarizes algorithm LSMR for solving the regularized problem (8.1) with given λ. Our MATLAB implementation is based on these steps.

1. (Initialize)

$$\begin{array}{llll}
\beta_1 u_1 = b & \alpha_1 v_1 = A^T u_1 & \bar{\alpha}_1 = \alpha_1 & \bar{\zeta}_1 = \alpha_1\beta_1 \\
\rho_0 = 1 & \bar{\rho}_0 = 1 & \bar{c}_0 = 1 & \bar{s}_0 = 0 \\
\ddot{\beta}_1 = \beta_1 & \dot{\beta}_0 = 0 & \dot{\rho}_0 = 1 & \tilde{\tau}_{-1} = 0 \\
\tilde{\theta}_0 = 0 & \zeta_0 = 0 & d_0 = 0 & \\
h_1 = v_1 & \bar{h}_0 = \vec{0} & x_0 = \vec{0} &
\end{array}$$

2. For $k = 1, 2, 3, \ldots$ repeat steps 3–12.

3. (Continue the bidiagonalization)

$$\begin{aligned}\beta_{k+1}u_{k+1} &= Av_k - \alpha_k u_k\\ \alpha_{k+1}v_{k+1} &= A^T u_{k+1} - \beta_{k+1}v_k\end{aligned}$$

4. (Construct rotation $\hat{Q}_{k,2k+1}$)

$$\begin{aligned}\hat{\alpha}_k &= \left(\bar{\alpha}_k^2 + \lambda^2\right)^{\frac{1}{2}}\\ c_k &= \bar{\alpha}_k/\hat{\alpha}_k & s_k &= \lambda/\hat{\alpha}_k\end{aligned}$$

5. (Construct and apply rotation $Q_{k,k+1}$)

$$\begin{aligned}\rho_k &= \left(\hat{\alpha}_k^2 + \beta_{k+1}^2\right)^{\frac{1}{2}}\\ c_k &= \hat{\alpha}_k/\rho_k & s_k &= \beta_{k+1}/\rho_k\\ \theta_{k+1} &= s_k\alpha_{k+1} & \bar{\alpha}_{k+1} &= c_k\alpha_{k+1}\end{aligned}$$

6. (Construct and apply rotation $\bar{Q}_{k,k+1}$)

$$\begin{aligned}\bar{\theta}_k &= \bar{s}_{k-1}\rho_k & \bar{\rho}_k &= \left((\bar{c}_{k-1}\rho_k)^2 + \theta_{k+1}^2\right)^{\frac{1}{2}}\\ \bar{c}_k &= \bar{c}_{k-1}\rho_k/\bar{\rho}_k & \bar{s}_k &= \theta_{k+1}/\bar{\rho}_k\\ \zeta_k &= \bar{c}_k\bar{\zeta}_k & \bar{\zeta}_{k+1} &= -\bar{s}_k\bar{\zeta}_k\end{aligned}$$

7. (Update h, $\bar{h}$, d, x)

$$\begin{aligned}\bar{h}_k &= h_k - (\bar{\theta}_k\rho_k/(\rho_{k-1}\bar{\rho}_{k-1}))\bar{h}_{k-1}\\ x_k &= x_{k-1} + (\zeta_k/(\rho_k\bar{\rho}_k))\bar{h}_k\\ h_{k+1} &= v_{k+1} - (\theta_{k+1}/\rho_k)h_k\end{aligned}$$

8. (Apply rotation $\hat{Q}_{k,2k+1}, Q_{k,k+1}$)

$$\begin{aligned}\acute{\beta}_k &= \hat{c}_k\ddot{\beta}_k & \check{\beta}_k &= -\hat{s}_k\ddot{\beta}_k\\ \hat{\beta}_k &= c_k\acute{\beta}_k & \ddot{\beta}_{k+1} &= -s_k\acute{\beta}_k\end{aligned}$$

9. (If $k \geq 2$, construct and apply rotation $\widetilde{Q}_{k-1,k}$)

$$\begin{aligned}\tilde{\rho}_{k-1} &= \left(\dot{\rho}_{k-1}^2 + \bar{\theta}_k^2\right)^{\frac{1}{2}}\\ \tilde{c}_{k-1} &= \dot{\rho}_{k-1}/\tilde{\rho}_{k-1} & \tilde{s}_{k-1} &= \bar{\theta}_k/\tilde{\rho}_{k-1}\\ \tilde{\theta}_k &= \tilde{s}_{k-1}\bar{\rho}_k & \dot{\rho}_k &= \tilde{c}_{k-1}\bar{\rho}_k\\ \tilde{\beta}_{k-1} &= \tilde{c}_{k-1}\dot{\beta}_{k-1} + \tilde{s}_{k-1}\hat{\beta}_k & \dot{\beta}_k &= -\tilde{s}_{k-1}\dot{\beta}_{k-1} + \tilde{c}_{k-1}\hat{\beta}_k\end{aligned}$$

10. (Update $\tilde{t}_k$ by forward substitution)

$$\begin{aligned}\tilde{\tau}_{k-1} &= (\zeta_{k-1} - \tilde{\theta}_{k-1}\tilde{\tau}_{k-2})/\tilde{\rho}_{k-1}\\ \dot{\tau}_k &= (\zeta_k - \tilde{\theta}_k\tilde{\tau}_{k-1})/\dot{\rho}_k\end{aligned}$$

11. (Estimate $\|r_k\|$)

$$\begin{aligned}d_k &= d_{k-1} + \check{\beta}_k^2\\ \|r_k\| &= \left(d_k + (\dot{\beta}_k - \dot{\tau}_k)^2 + \ddot{\beta}_{k+1}^2\right)^{\frac{1}{2}}\end{aligned}$$

12. (Estimate $\|A^T r_k\|$, $\|x_k\|$, $\|A\|$, cond(A) and test for termination)

$\|A^T r_k\| = |\bar{\zeta}_{k+1}|$ (section 4.1)
$\|x_k\|^2 = \|x_{k-1}\|^2 + \hat{\zeta}_k^2$ (section 4.4)
Estimate $\sigma_{\max}(B_k), \sigma_{\min}(B_k)$ and hence $\|A\|, \text{cond}(A)$ (section 4.3)
Terminate if any of the stopping criteria are satisfied (section 5.1)

9. Backward errors. For inconsistent LS problems, the optimal backward error norm

$$\mu(x) \equiv \min_E \|E\| \quad \text{s.t.} \quad (A+E)^T(A+E)x = (A+E)^T b$$

is known to be the smallest singular value of a certain $m \times (n+m)$ matrix C; see Waldén et al. [18] and Higham [7, pp. 392–393]:

$$\mu(x) = \sigma_{\min}(C), \qquad C \equiv \begin{bmatrix} A & \frac{\|r\|}{\|x\|}\Big(I - \frac{rr^T}{\|r\|^2}\Big) \end{bmatrix}.$$

This is generally considered too expensive to evaluate.

9.1. Approximate backward errors E_1 and E_2. In 1975, Stewart [15] discussed a particular backward error estimate that we will call E_1. Let $\widehat{x}$ and $\widehat{r} = b - A\widehat{x}$ be the exact least-squares solution and residual. Stewart showed that any *approximate* solution x with residual $r = b - Ax$ is the exact least-squares solution of the perturbed problem $\min \|b - (A + E_1)x\|$, where E_1 is the rank-one matrix

$$E_1 = \frac{ex^T}{\|x\|^2}, \qquad \|E_1\| = \frac{\|e\|}{\|x\|},$$

with $e \equiv \widehat{r} - r$ and $\|r\|^2 = \|\widehat{r}\|^2 + \|e\|^2$.

Soon after, Stewart [16] gave a further important result that can be used within any least-squares solver. The approximate x and a vector $\tilde{r}$ are the exact least-squares solution and residual of the perturbed problem $\min \|b - (A + E_2)x\|$, where

$$E_2 = -\frac{rr^T A}{\|r\|^2}, \qquad \|E_2\| = \frac{\|A^T r\|}{\|r\|}, \qquad \tilde{r} = b - (A + E_2)x.$$

This estimate is used in LSQR for each approximation x_k and residual $r_k = b - Ax_k$ because the current $\|r_k\|$ and $\|A^T r_k\|$ can be accurately estimated at essentially no cost. An added feature is that the associated $\tilde{r} = b - (A + E_2)x_k = r_k$ because $E_2 x_k = 0$ in LSQR (assuming orthogonality of V_k). We can show the same for LSMR, that (x_k, r_k) are theoretically exact for the perturbed problem $(A + E_2)x \approx b$.

We now show that $\|E_2\|$ is smaller for LSMR than for LSQR. As Figure 10.2 illustrates, this property gives LSMR a vital practical advantage for stopping early.

THEOREM 9.1. $\|E_2^{LSMR}\| \le \|E_2^{LSQR}\|$.

Proof. This follows directly from $\|A^T r_k\|^{\text{LSMR}} \le \|A^T r_k\|^{\text{LSQR}}$ and $\|r_k\|^{\text{LSMR}} \ge \|r_k\|^{\text{LSQR}}$. ∎

9.2. Approximate optimal backward error $\widetilde{\mu}(x)$. Various authors have derived expressions for $\widetilde{\mu}(x)$, a quantity that has proved to be a very accurate approximation to $\mu(x)$, the optimal backward error for $Ax \approx b$, when x is at least moderately

close to the exact least-squares solution. Grcar, Saunders, and Su [5] show that the full-rank least-squares problem

$$K = \begin{bmatrix} A \\ \frac{\|r\|}{\|x\|} I \end{bmatrix}, \qquad v = \begin{bmatrix} r \\ 0 \end{bmatrix}, \qquad \min_y \|Ky - v\| \tag{9.1}$$

has a solution y such that

$$\widetilde{\mu}(x) = \|Ky\|/\|x\|, \tag{9.2}$$

and give the following MATLAB script for computing $c \equiv Ky$ and thence $\widetilde{\mu}(x)$ using sparse QR factors of K:

```
[m,n] = size(A);           r   = b - A*x;
normx = norm(x);           eta = norm(r)/normx;
p     = colamd(A);
K     = [A(:,p);  eta*speye(n)];
v     = [    r ;    zeros(n,1)];
[c,R] = qr(K,v,0);         mutilde = norm(c)/normx;
```

In our experiments we use this script to estimate the optimal backward error for each approximate x generated by LSQR and LSMR.

10. Numerical results. For test examples, we have drawn from the University of Florida Sparse Matrix Collection (Davis [3]). The `LPnetlib` group provides data for 138 linear programming problems of widely varying origin, structure, and size. The constraint matrix and objective function may be used to define a sparse least-squares problem $\min \|Ax - b\|$. Each example was downloaded in MATLAB format, and a sparse matrix A and dense vector b were extracted from the data structure via `A = (Problem.A)'` and `b = Problem.c`.

Five examples had $b = 0$, and a further six gave $A^T b = 0$. The remaining 127 problems had up to 243000 rows, 10000 columns, and 1.4M nonzeros in A. LSQR and LSMR were run on each of those 127, generating sequences of approximate solutions $\{x_k^{\text{LSQR}}\}$ and $\{x_k^{\text{LSMR}}\}$. The iteration indices k are omitted below. The associated residual vectors are denoted by r without ambiguity. x^* is the solution to the least-squares problem, or the minimum-norm solution to the least-squares problem if the system is singular.

10.1. Observations.

1. $\|r\|^{\text{LSQR}}$ is monotonic by design. $\|r\|^{\text{LSMR}}$ seems to be monotonic (no counter-examples were found) and *nearly* as small as $\|r\|^{\text{LSQR}}$ for all iterations on almost all problems. Figure 10.1 illustrates a typical example and a rare case.
2. (Theorem) If $\|r\|$ is monotonic, then $\|E_1\|$ is monotonic.
3. $\|E_1^{\text{LSQR}}\|$ is monotonic because $\|r\|$ is montonic.
   $\|E_1^{\text{LSMR}}\|$ seems to be monotonic because $\|r\|$ seems to be montonic.
4. $\|E_2^{\text{LSQR}}\|$ is *not* monotonic.
   $\|E_2^{\text{LSMR}}\|$ seems to be monotonic almost always. Figure 10.2 shows a typical case. The sole exception for this observation is also shown.
5. $\|E_1^{\text{LSQR}}\| \leq \|E_2^{\text{LSQR}}\|$ often. Not so for LSMR. Some examples are shown on Figure 10.3, along with $\widetilde{\mu}(x_k)$, the accurate estimate (9.1)–(9.2) of the optimal backward error for each point x_k.

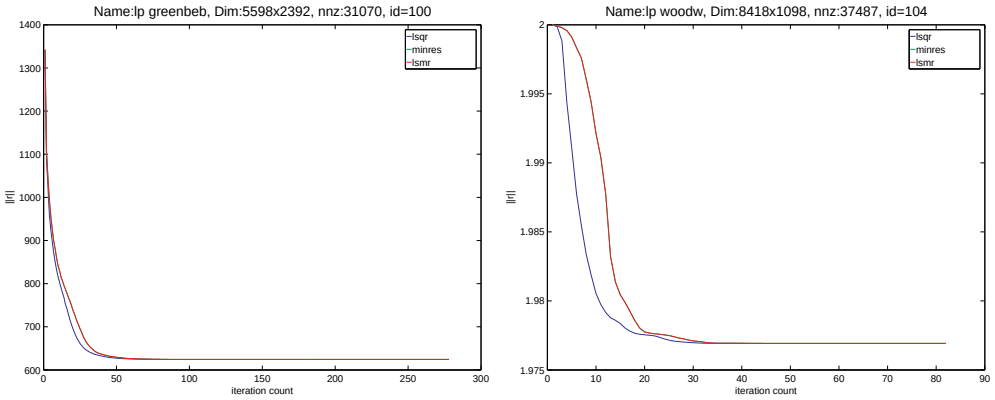

FIG. 10.1. *For most iterations,* $\|r^{LSMR}\|$ *appears to be monotonic and nearly as small as* $\|r^{LSQR}\|$. *Left: A typical case (problem* `lp_greenbeb`*). Right: A rare case (problem* `lp_woodw`*). LSMR's residual norm is significantly larger than LSQR's during early iterations.*

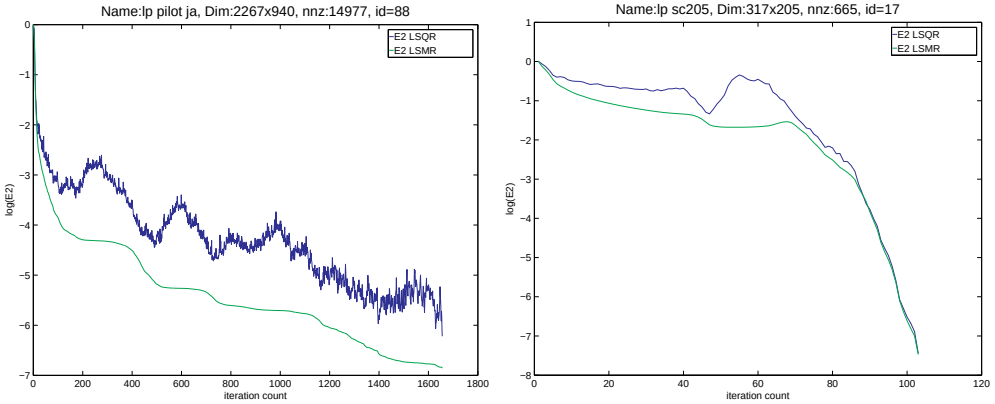

FIG. 10.2. *For most iterations,* $\|E_2^{LSMR}\|$ *appears to be monotonic (but* $\|E_2^{LSQR}\|$ *is not). Left: A typical case (problem* `lp_pilot_ja`*). Right: Sole exception (problem* `lp_sc205`*) at iterations 54–67. The exception remains even if* U_k *and/or* V_k *are reorthogonalized.*

6. $\|E_2^{\mathrm{LSMR}}\| \approx \widetilde{\mu}(x^{\mathrm{LSMR}})$ almost always. Figure 10.4 shows a typical example and a rare case. In all such "rare" cases, $\|E_1^{\mathrm{LSMR}}\| \approx \widetilde{\mu}(x^{\mathrm{LSMR}})$ instead!
7. $\widetilde{\mu}(x^{\mathrm{LSQR}})$ is not always monotonic. $\widetilde{\mu}(x^{\mathrm{LSMR}})$ does seem to be monotonic. See Figure 10.5 for examples.
8. $\widetilde{\mu}(x^{\mathrm{LSMR}}) \leq \widetilde{\mu}(x^{\mathrm{LSQR}})$ almost always. See Figure 10.6 for examples.
9. The errors $\|x^{\mathrm{LSQR}} - x^*\|$ and $\|x^{\mathrm{LSMR}} - x^*\|$ are both monotonically decreasing. $\|x^{\mathrm{LSQR}} - x^*\| \leq \|x^{\mathrm{LSMR}} - x^*\|$. x^{LSQR} and x^{LSMR} both converge to the minimum-norm solution for singular systems. See Figure 10.7 for examples.

10.2. Comparison with MINRES on the normal equation. Benbow [2] gave numerical results comparing a generalized form of LSQR with application of MINRES to the corresponding normal equation. The curves in Figure 3 of [2] are a preview of the comparisons shown above (where LSMR serves as our more reliable implementation of MINRES).

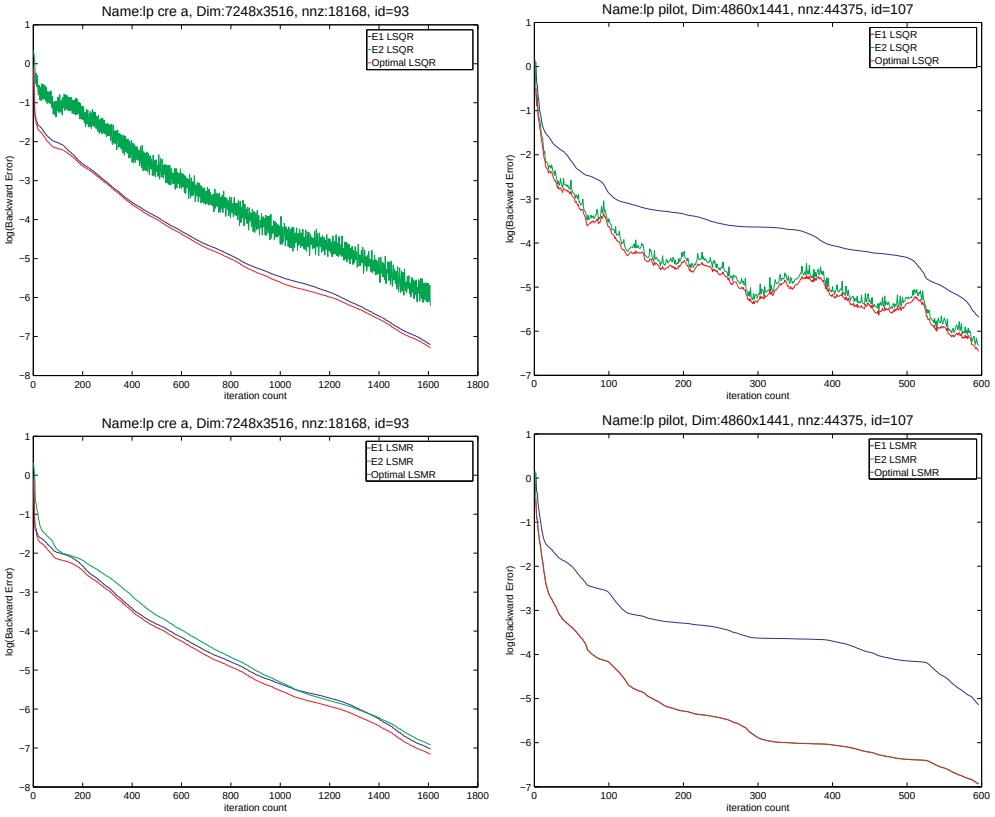

FIG. 10.3. $\|E_1\|$, $\|E_2\|$, *and* $\widetilde{\mu}(x_k)$ *for LSQR (top figures) and LSMR (bottom figures). Top left: A typical case (problem* `lp_cre_a`*).* $\|E_1^{LSQR}\|$ *is close to the optimal backward error, but the computable* $\|E_2^{LSQR}\|$ *is not. Top right: A rare case (problem* `lp_pilot`*) in which* $\|E_2^{LSQR}\|$ *is close to optimal. Bottom left: (problem* `lp_cre_a`*).* $\|E_1^{LSMR}\|$ *and* $\|E_2^{LSMR}\|$ *are often both close to the optimal backward error. Bottom right: (problem* `lp_pilot`*).* $\|E_1^{LSMR}\|$ *is far from optimal, but the computable* $\|E_2^{LSMR}\|$ *is almost always close (too close to distinguish in the plot!).*

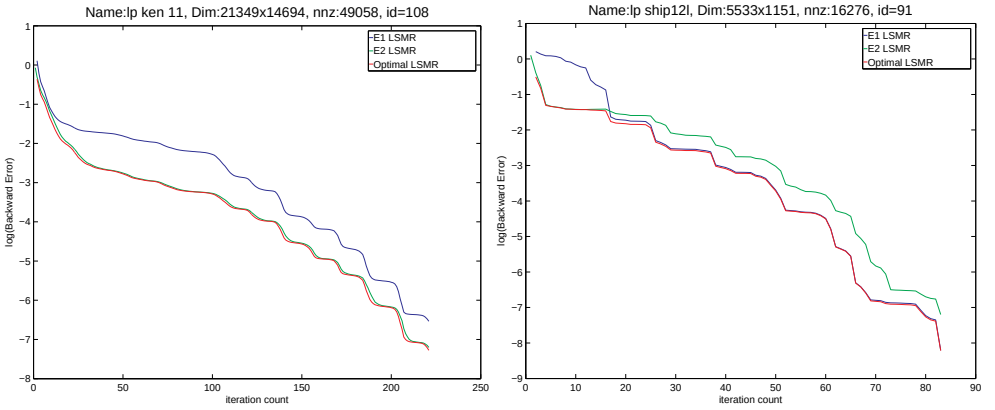

FIG. 10.4. *Again,* $\|E_2^{LSMR}\| \approx \widetilde{\mu}(x^{LSMR})$ *almost always (the computable backward error estimate is essentially optimal). Left: A typical case (problem* `lp_ken11`*). Right: A rare case (problem* `lp_ship12l`*). Here,* $\|E_1^{LSMR}\| \approx \widetilde{\mu}(x^{LSMR})$*!*

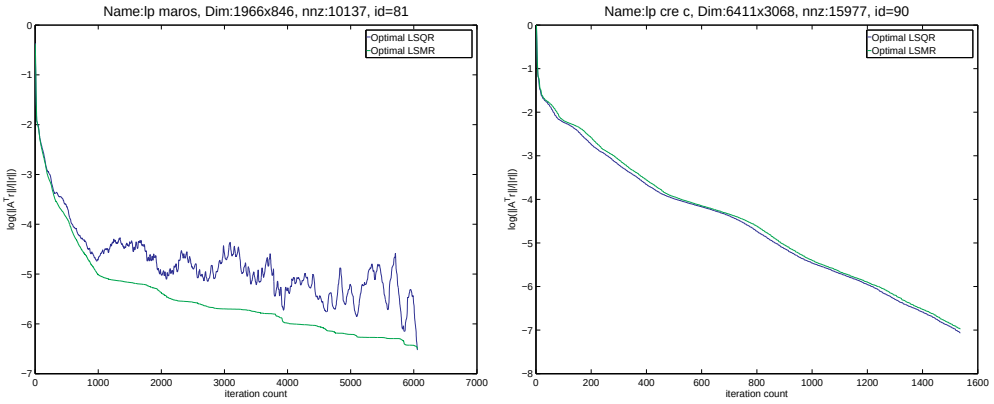

FIG. 10.5. *$\widetilde{\mu}(x^{LSMR})$ seems to be always monotonic, but $\widetilde{\mu}(x^{LSQR})$ is usually not. Left: A typical case for both LSQR and LSMR (problem* `lp_maros`*). Right: A rare case for LSQR, typical for LSMR (problem* `lp_cre_c`*).*

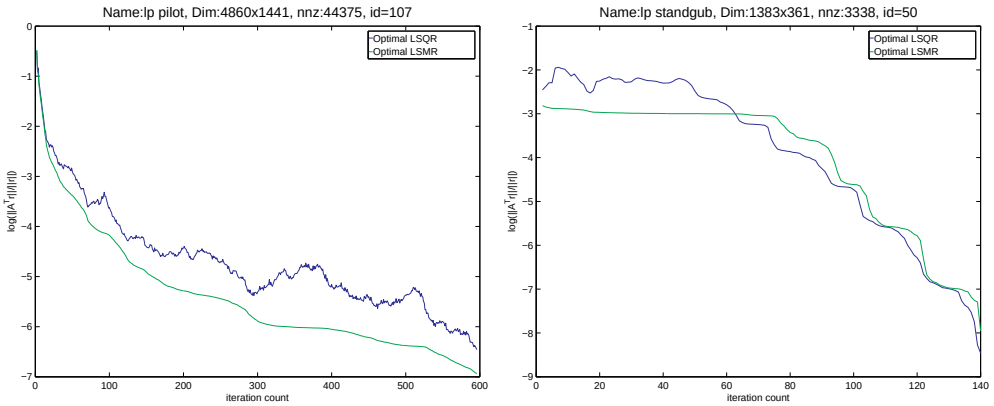

FIG. 10.6. *$\widetilde{\mu}(x^{LSMR}) \leq \widetilde{\mu}(x^{LSQR})$ almost always. Left: A typical case (problem* `lp_pilot`*). Right: A rare case (problem* `lp_standgub`*).*

10.3. Reorthogonalization. It is well known that in practice, certain Krylov-subspace methods can take arbitrarily many iterations on some data because of loss of orthogonality of the vectors involved. For the Golub-Kahan bidiagonalization, both sets of vectors—that is, matrices U_k and V_k—may lose orthogonality as k increases.

As an experiment, we implemented the following options in LSMR:

1. No reorthogonalization.
2. Reorthogonalize V_k (that is, reorthogonalize v_k with respect to V_{k-1}).
3. Reorthogonalize U_k (that is, reorthogonalize u_k with respect to U_{k-1}).
4. Both 2 and 3.

Figure 10.8 shows an "easy" case in which all options converge equally well (convergence before significant loss of orthogonality), and an extreme case in which reorthogonalization makes a large difference. Unexpectedly, options 2, 3, and 4 are indistinguishable in the extreme case (and the same was observed for *all* cases).

We can explain this effect by noting that for both LSQR and LSMR, the relations $x_k = V_k y_k$, $r_k = U_{k+1} p_{k+1}$, and $A^T r_k = V_{k+1} q_{k+1}$ hold accurately in practice for various vectors y_k, p_{k+1}, q_{k+1}. Thus, on compatible *or* incompatible systems, both

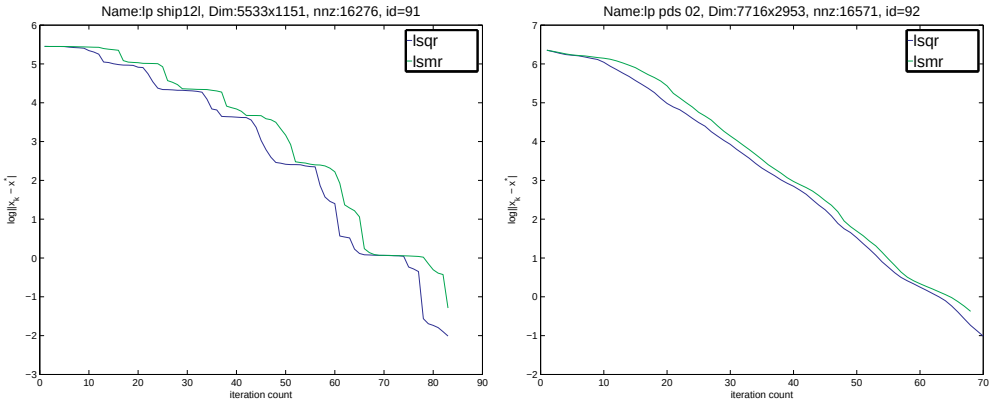

FIG. 10.7. *Both* $\|x^{LSQR} - x^*\|$ *and* $\|x^{LSMR} - x^*\|$ *are monotonically decreasing.* $\|x^{LSQR} - x^*\| \leq \|x^{LSMR} - x^*\|$. *Left: A nonsingular least-squares system (problem* `lp_ship12l`*). Right:* x^{LSQR} *and* x^{LSMR} *both converge to the minimum-norm least-squares solution of a singular system (problem* `lp_pds`*).*

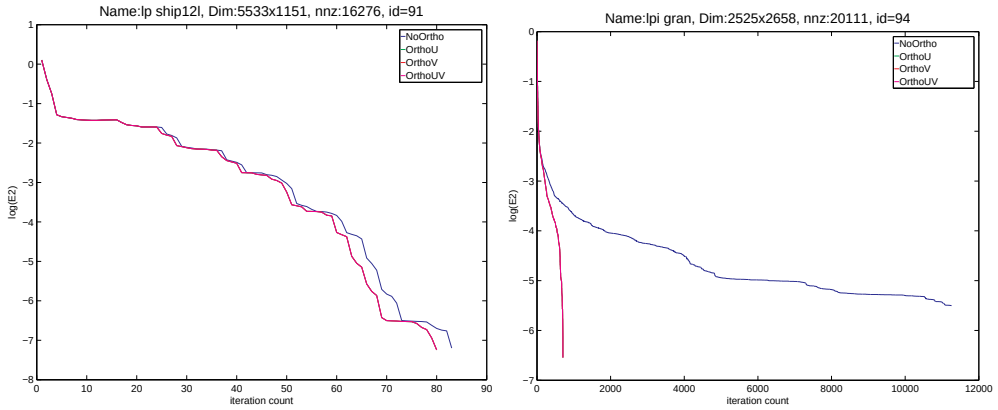

FIG. 10.8. *LSMR with and without reorthogonalization of* V_k *and/or* U_k. *Left: An easy case (problem* `lp_ship12l`*). Right: A helpful case (problem* `lp_gran`*).*

methods must converge in a finite number of iterations if V_k *and/or* U_k are essentially orthogonal.

The argument is nontrivial for incompatible systems when U_k is reorthogonalized but not V_k. If the maximum of m iterations occurred, the next iteration would give $\beta_{m+1} \approx 0$ and hence $r_{m+1} \approx 0$. If the true r is nonzero, this is a contradiction. Thus, iterations must terminate earlier with some $\alpha_{k+1} \approx 0$, in which case $\|A^T r_k\| \approx 0$ and the problem has been solved (section 5).

Other authors have presented numerical results on this effect. For example, on some randomly generated least-squares problems of increasing condition number, Hayami *et al.* [6] compare their BA-GMRES method with an implementation of CGLS (equivalent to LSQR [11]) in which V_k is reorthogonalized, and find that the methods require essentially the same number of iterations. The preconditioner chosen for BA-GMRES made that method equivalent to GMRES on $A^T Ax = A^T b$. Thus, GMRES without reorthogonalization was seen to converge essentially as well

as CGLS or LSQR with reorthogonalization of V_k (option 2 above).

This coincides with the analysis by Paige *et al.* [9], who conclude that MGS-GMRES does not need reorthogonalization of the Arnoldi vectors V_k.

10.4. Singular vectors. In [1], Barlow et al. describe a reliable procedure for obtaining the factorization $X = UBV^T$ of a dense matrix $X \in R^{m\times n}$, where B is upper bidiagonal and U and V have orthonormal columns. The aim is to estimate the singular values of X accurately from those of B. Supposing $m > n$, our results in this section suggest that an effective alternative would be to apply the Golub-Kahan process to X with a random starting vector b, reorthogonalizing the columns of V_k and saving U_k without reorthogonalization. After n steps, the process will terminate with $XV_n = U_{n+1}B_n$, with V_n orthonormal to machine precision ϵ and the columns of U_{n+1} orthonormal to $O(\sqrt{\epsilon})$. An SVD $B_n = \bar{U}\bar{S}\bar{V}^T$ gives

$$X = U_{n+1}B_nV_n^T = (U_{n+1}\bar{U})\bar{S}(V_n\bar{V})^T.$$

We anticipate that the left-most columns of $U_{n+1}\bar{U}$ would provide accurate left singular vectors associated with the largest singular values of X.

10.5. Modifications to reorthogonalization. With full reorthogonalization, the storage requirement grows linearly and the computational cost grows quadratically with respect to the iteration number. To utilize (possibly limited) storage for speeding up LSMR, we consider some of the standard variations. In view of the preceding results, we focus on reorthogonalizing V_k but not U_k.

10.5.1. Restarting. A simple approach is to restart the algorithm every l steps, as proposed for GMRES in [14]. To be precise, we set

$$r_l = b - Ax_l, \qquad \min \|A\Delta x - r_l\|, \qquad x_l \leftarrow x_l + \Delta x$$

and repeat the same process until convergence. Our numerical test in Figure 10.9 shows that restarting LSMR even with full reorthogonalization (of V_k) may lead to stagnation. In this example, convergence with restarting is much slower than LSMR without reorthogonalization. Restarting does not seem a useful approach to lowering computational and storage cost.

10.5.2. Local reorthogonalization. Here we reorthogonalize each new v_k with respect to the previous l vectors, where l is a specified parameter. An example is shown in Figure 10.10.

With $l = 5$, 10, and 50 we see that partial speedup can be achieved with local reorthogonalization of v_k. This allows full utilization of available memory to obtain faster convergence. It should be emphasized that the potential speedup achieved by reorthogonalizing V_k depends strongly on the computational cost of Av and A^Tu. If the matrix-vector products are expensive, reorthogonalization is preferable. Otherwise, LSMR without reorthogonalization may converge faster in terms of total CPU time.

10.5.3. Partial reorthogonalization. Larsen [13] uses partial reorthogonalization of both V_k and U_k within his PROPACK software for computing a set of singular values and vectors for a sparse rectangular matrix A. Similar techniques could be included within LSMR to reduce the iteration count at the expense of storage for a limited number of earlier vectors u_k and v_k.

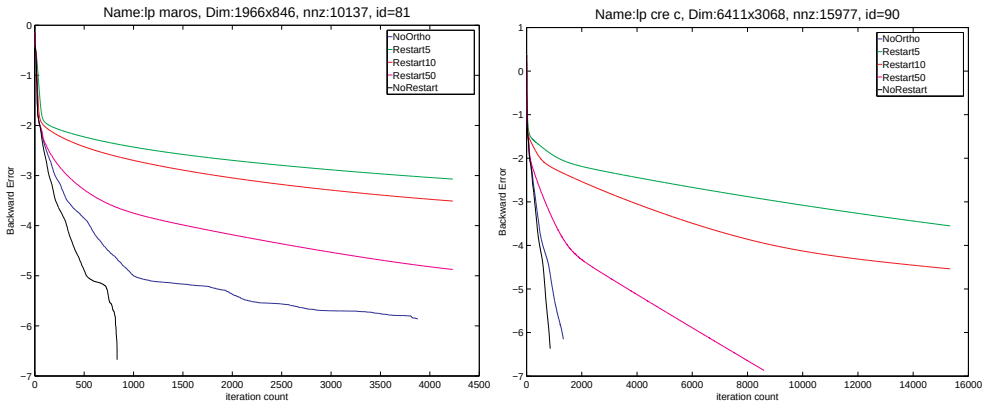

FIG. 10.9. *LSMR with reorthogonalized V_k and restarting. NoOrtho represents LSMR without reorthogonalization. Restart5, Restart10, and Restart50 represents reorthogonalized LSMR with restarting every 5, 10 or 50 iterations. NoRestart represents reorthogonalized LSMR without restarting. Left: Problem* `lp_ship12l`*. Right: Problem* `lp_gran`*.*

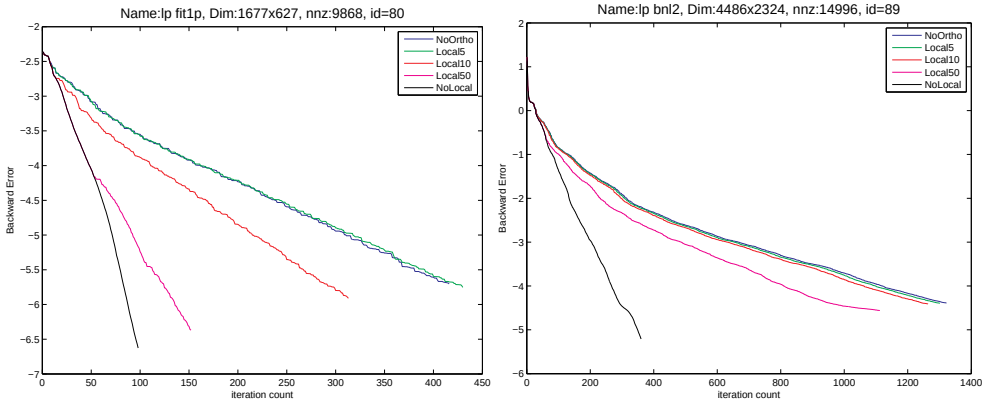

FIG. 10.10. *LSMR with local reorthogonalization of V_k. NoOrtho represents LSMR without reorthogonalization. Local5, Local10, and Local50 represent LSMR with local reorthogonalization of each v_k with respect to the previous 5, 10, or 50 vectors. NoLocal represents LSMR with reorthogonalized V_k without restarting. Left: Problem* `lp_ship12l`*. Right: Problem* `lp_gran`*.*

11. Summary. We have presented LSMR, an iterative algorithm for least-squares systems, along with details of its implementation and experimental results to suggest that it improves noticeably upon the widely adopted LSQR algorithm.

As in LSQR, theoretical and practical stopping criteria are provided for solving the problems $Ax = b$, $\min \|Ax - b\|$, and least-squares with Tikhonov regularization, using estimates of $\|r_k\|$ and $\|A^T r_k\|$ that are cheaply computable. For least-squares problems, the Stewart backward error estimate $\|E_2\|$ (section 9.1) seems experimentally to be very close to the *optimal* backward error at each iterate x_k^{LSMR}. This is likely to terminate LSMR significantly sooner than the same stopping rule in LSQR.

In experiments with reorthogonalization, we found that the Golub-Kahan process retains high accuracy if the columns of either V_k or U_k are reorthogonalized. There is no need to reorthogonalize both. In addition to speeding up reorthogonalized LSMR, this discovery could be used to design more reliable algorithms for computing singular values and vectors.

To conclude, we make the following recommendations:

1. For least-squares problems, current users of LSQR are recommended to try LSMR because it provides faster and smoother convergence of $\|A^T r_k\|$ (since it is equivalent to MINRES on the normal equation).
2. For least-squares problems that need to be stopped early (e.g., if only limited computational time is available), LSMR is preferable to LSQR because the backward error estimate at each iteration is always smaller (sometimes by two orders of magnitude).
3. Should extra memory be available, LSMR with reorthogonalized V_k could be used to reduce the number of iterations and possibly the computational time.

A MATLAB implementation of LSMR is available from [8].

Acknowledgement. We are grateful to Chris Paige for his helpful comments on reorthogonalization and other aspects of this work.

REFERENCES

[1] J. BARLOW, N. BOSNER, AND Z. DRMAC, *A new stable bidiagonal reduction algorithm*, Linear Alg. Applics., 397 (2005), pp. 35–84.

[2] S. J. BENBOW, *Solving generalized least-squares problems with LSQR*, SIAM J. Matrix Anal. Appl., 21 (1999), pp. 166–177.

[3] T. A. DAVIS, *University of Florida Sparse Matrix Collection.* `http://www.cise.ufl.edu/research/sparse/matrices`.

[4] G. H. GOLUB AND W. KAHAN, *Calculating the singular values and pseudo-inverse of a matrix*, J. of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis, 2 (1965), pp. 205–224.

[5] J. F. GRCAR, M. A. SAUNDERS, AND Z. SU, *Estimates of optimal backward perturbations for linear least squares problems*, Report SOL 2007-1, Department of Management Science and Engineering, Stanford University, Stanford, CA, 2007. 21 pp.

[6] K. HAYAMI, J.-F. YIN, AND T. ITO, *GMRES methods for least squares problems*, SIAM J. Matrix Anal. Appl., n (to appear), pp. n–n.

[7] N. J. HIGHAM, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, second ed., 2002.

[8] *LSMR software for linear systems and least squares.* `http://www.stanford.edu/group/SOL/software.html`.

[9] C. C. PAIGE, M. ROZLOZNIK, AND Z. STRAKOS, *Modified Gram-Schmidt (MGS), least squares, and backward stability of MGS-GMRES*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 264–284.

[10] C. C. PAIGE AND M. A. SAUNDERS, *Solution of sparse indefinite systems of linear equations*, SIAM J. on Numerical Analysis, 12 (1975), pp. 617–629.

[11] ———, *LSQR: An algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Softw., 8 (1982), pp. 43–71.

[12] ———, *Algorithm 583; LSQR: Sparse linear equations and least-squares problems*, ACM Trans. Math. Softw., 8 (1982), pp. 195–209.

[13] *PROPACK software for SVD of sparse matrices.* `http://soi.stanford.edu/~rmunk/PROPACK/`.

[14] Y. SAAD AND M. H. SCHULTZ, *GMRES: a generalized minimum residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. and Statist. Comput., 7 (1986), pp. 856–869.

[15] G. W. STEWART, *An inverse perturbation theorem for the linear least squares problem*, SIGNUM Newsletter, 10 (1975), pp. 39–40.

[16] ———, *Research, development and LINPACK*, in Mathematical Software III, J. R. Rice, ed., Academic Press, New York, 1977, pp. 1–14.

[17] ———, *The QLP approximation to the singular value decomposition*, SIAM J. Sci. Comput., 20 (1999), pp. 1336–1348.

[18] B. WALDÉN, R. KARLSON, AND J.-G. SUN, *Optimal backward perturbation bounds for the linear least squares problem*, Numerical Linear Algebra with Applications, 2 (1995), pp. 271–286.