BIT 37:1(1997), 96–104.

# COMPUTING PROJECTIONS WITH LSQR*

MICHAEL A. SAUNDERS†

*Systems Optimization Laboratory, Department of EES & OR*
*Stanford University, Stanford, CA 94305-4023, USA.*
*email: mike@SOL-michael.stanford.edu*

**Abstract.**

LSQR uses the Golub-Kahan bidiagonalization process to solve sparse least-squares problems with and without regularization. In some cases, projections of the right-hand side vector are required, rather than the least-squares solution itself. We show that projections may be obtained from the bidiagonalization as linear combinations of (theoretically) orthogonal vectors. Even the least-squares solution may be obtained from orthogonal vectors, perhaps more accurately than the usual LSQR solution. (However, LSQR has proved equally good in all examples so far.)

*AMS subject classification:* 65F10, 65F20, 65F50, 65F05.

*Key words:* Least squares, conjugate-gradient method, Golub-Kahan process, regularization.

## 1 Introduction.

LSQR [11, 12] is a conjugate-gradient-like method for solving linear least-squares problems

$$\min_x \|b - Ax\|_2, \tag{1.1}$$

where $A$ is a real $m \times n$ matrix and $b$ is a real vector. Typically $m \geq n$ and $\operatorname{rank}(A) = n$, though not necessarily. LSQR uses the Golub-Kahan bidiagonalization of $A$ [6] with starting vector $b$, forming a sequence of iterates $\{x_k\}$ to approximate $x$.

For problem (1.1), let us define the following items:

$$P = A(A^TA)^{-1}A^T, \tag{1.2}$$
$$x = (A^TA)^{-1}A^Tb, \tag{1.3}$$
$$p = Pb = Ax, \tag{1.4}$$
$$r = (I - P)b = b - Ax, \tag{1.5}$$

*Received July 1996. Revised December 1996. Presented at the Cornelius Lanczos International Centenary Conference, North Carolina State University, Raleigh, NC, December 1993.

†Partially supported by Department of Energy grant DE-FG03-92ER25117, National Science Foundation grants DMI-9204208 and DMI-9500668, and Office of Naval Research grants N00014-90-J-1242 and N00014-96-1-0274.

where $P$ and $(I - P)$ are both projection operators. Since some applications need $p$ or $r$ rather than $x$ itself, and since these projections are less sensitive than $x$ to perturbations in the data [7], it seems reasonable to compute the projections directly from the Golub-Kahan process, rather than from LSQR's final approximation to $x$.

Section 3.1 shows how to compute $p$ and $r$ for problem (1.1). Section 4.1 does the same for regularized or *damped* least-squares problems, and suggests some unexpected new ways for computing $x$.

### *1.1 Orthogonal steps.*

If a sequence of approximations $\{x_k\}$ is computed in the form

$$x_k = V_k y_k = x_{k-1} + \eta_k v_k, \tag{1.6}$$

where the columns of $V_k$ are (at least theoretically) orthonormal ($V_k^T V_k = I$), we say that $x$ *is computed by orthogonal steps.* For example, Craig's method [4, 5, 11] solves unsymmetric equations $Ax = b$ using orthogonal steps (1.6) to update each $x_k$. In contrast, the normal LSQR iterates have the form

$$x_k = (V_k R_k^{-1}) z_k \equiv W_k z_k = x_{k-1} + \zeta_k w_k, \tag{1.7}$$

where $V_k$ is orthonormal but $W_k$ is not. If the triangular matrix $R_k$ is ill-conditioned, we would expect a certain loss of precision (via cancellation) in forming $x_k$ that way.

A contribution of this paper is to show that for least-squares problems with and without damping, $x$, $p$ and $r$ can all be computed by orthogonal steps.

## 2 Bidiagonalization.

Given a general matrix $A$ and a starting vector $b$, the Golub-Kahan process generates two sequences of vectors $\{u_k\}$, $\{v_k\}$ and positive scalars $\{\alpha_k\}$, $\{\beta_k\}$ such that after $k$ steps,

$$\begin{array}{rcl} AV_k &=& U_{k+1} B_k, \\ A^T U_{k+1} &=& V_k B_k^T + \alpha_{k+1} v_{k+1} e_{k+1}^T, \\ U_k &=& (u_1 \; u_2 \; \dots \; u_k), \\ V_k &=& (v_1 \; v_2 \; \dots \; v_k), \end{array} \qquad B_k = \begin{pmatrix} \alpha_1 & & \\ \beta_2 & \ddots & \\ & \ddots & \alpha_k \\ & & \beta_{k+1} \end{pmatrix}, \tag{2.1}$$

where $B_k$ is $(k+1) \times k$ and lower bidiagonal. The starting condition is $\beta_1 u_1 = b$, so that $U_k \beta_1 e_1 = b$ exactly for all $k$, and with exact arithmetic the columns of $U_k$ and $V_k$ would be orthonormal.

## 3 Least squares.

To solve problem (1.1), LSQR defines a sequence of approximations $x_k = V_k y_k$, where each $y_k$ is defined by a subproblem, $\min \|\beta_1 e_1 - B_k y_k\|$ [11, 13]. The subproblem is reliably solved via a QR factorization of $B_k$:

$$Q_k(\,B_k \;\; \beta_1 e_1\,) = \begin{pmatrix} R_k & z_k \\ & \bar{\zeta}_k \end{pmatrix}, \qquad R_k y_k = z_k,$$

where $R_k$ is $k \times k$ and upper bidiagonal. The matrix $Q_k$ is nominally a product of $k$ plane rotations, requiring little work. In LSQR we work with symmetric transformations for simplicity. The $k$th transformation is of the form

$$\begin{pmatrix} c_k & s_k \\ s_k & -c_k \end{pmatrix} \begin{pmatrix} \bar{\rho}_{k-1} & 0 & \bar{\zeta}_{k-1} \\ \beta_{k+1} & \alpha_{k+1} & 0 \end{pmatrix} = \begin{pmatrix} \rho_k & \theta_k & \zeta_k \\ & \bar{\rho}_k & \bar{\zeta}_k \end{pmatrix},$$

where $\bar{\zeta}_k$ later becomes $\zeta_{k+1}$ (and similarly for other barred items). To keep storage to a minimum, $y_k$ is eliminated and $x_k$ is formed as in (1.7).

### *3.1 Projections.*

As approximations to the projections $p = Pb$ and $r = (I - P)b$, we use the vectors $p_k = Ax_k$ and $r_k = b - Ax_k$. Let us write the (theoretically orthonormal) matrix $U_{k+1}Q_k^T$ as

$$U_{k+1}Q_k^T = (\,U1_k \;\; \bar{u}_k\,), \tag{3.1}$$

in which the $k$th transformation has the form

$$(\,\bar{u}_{k-1} \;\; u_{k+1}\,) \begin{pmatrix} c_k & s_k \\ s_k & -c_k \end{pmatrix} = (\,u1_k \;\; \bar{u}_k\,).$$

It follows that

$$\begin{aligned} p_k &= Ax_k = AV_k y_k = U_{k+1} B_k y_k \\ &= U_{k+1} Q_k^T Q_k B_k y_k \\ &= (\,U1_k \;\; \bar{u}_k\,) \begin{pmatrix} R_k \\ 0 \end{pmatrix} y_k \\ &= (\,U1_k \;\; \bar{u}_k\,) \begin{pmatrix} z_k \\ 0 \end{pmatrix} = U1_k z_k \end{aligned}$$

and

$$\begin{aligned} r_k &= b - Ax_k \\ &= U_{k+1} Q_k^T Q_k (\beta_1 e_1 - B_k y_k) \\ &= (\,U1_k \;\; \bar{u}_k\,) \left\{ \begin{pmatrix} z_k \\ \bar{\zeta}_k \end{pmatrix} - \begin{pmatrix} R_k \\ 0 \end{pmatrix} y_k \right\} \\ &= (\,U1_k \;\; \bar{u}_k\,) \begin{pmatrix} 0 \\ \bar{\zeta}_k \end{pmatrix} = \bar{\zeta}_k \bar{u}_k. \end{aligned}$$

Thus, the sequences $\{p_k\}$ and $\{r_k\}$ are obtained by orthogonal steps. The main expense beyond the bidiagonalization lies in forming the columns of $U1_k$ in (3.1). Note that $x_k$ need not be formed.

## 4 Damped least squares.

The damped least-squares problem is

$$\min \|b - Ax\|^2 + \|\delta x\|^2 \quad \equiv \quad \min \left\| \begin{pmatrix} b \\ 0 \end{pmatrix} - \begin{pmatrix} A \\ \delta I \end{pmatrix} x \right\|^2, \tag{4.1}$$

where $\delta > 0$ is a small scalar that regularizes the problem if $\operatorname{rank}(A) < n$ or $A$ is ill-conditioned. For such problems, LSQR uses the same bidiagonalization to obtain approximations $x_k = V_k y_k$, where $y_k$ is defined by the subproblem

$$\min \left\| \begin{pmatrix} \beta_1 e_1 \\ 0 \end{pmatrix} - \begin{pmatrix} B_k \\ \delta I \end{pmatrix} y_k \right\|,$$

which is solved via an extended QR factorization [2, 12, 13]:

$$Q_k \begin{pmatrix} B_k & \beta_1 e_1 \\ \delta I & 0 \end{pmatrix} = \begin{pmatrix} R_k & z_k \\ & \bar{\zeta}_k \\ & q_k \end{pmatrix}, \qquad R_k y_k = z_k.$$

The matrix $Q_k$ now involves a product of $2k$ transformations, but the total work and storage is essentially the same as when $\delta = 0$. As before, $y_k$ is eliminated and $x_k$ is formed as in (1.7).

### *4.1 Projections.*

The damped least-squares solution satisfies $(A^T A + \delta^2 I)x = A^T b$. With

$$\bar{A} = \begin{pmatrix} A \\ \delta I \end{pmatrix}, \qquad \bar{b} = \begin{pmatrix} b \\ 0 \end{pmatrix},$$

the definitions analogous to (1.2)–(1.5) are

$$\bar{P} = \bar{A}(\bar{A}^T \bar{A})^{-1} \bar{A}^T, \tag{4.2}$$

$$x = (\bar{A}^T \bar{A})^{-1} A^T b, \tag{4.3}$$

$$\begin{pmatrix} p \\ s \end{pmatrix} = \bar{P}\bar{b} = \begin{pmatrix} Ax \\ \delta x \end{pmatrix}, \tag{4.4}$$

$$\begin{pmatrix} r \\ t \end{pmatrix} = (I - \bar{P})\bar{b} = \begin{pmatrix} b - Ax \\ -\delta x \end{pmatrix}, \tag{4.5}$$

where we see that $s = -t = \delta x$. Now define the (theoretically orthonormal) matrix

$$\begin{pmatrix} U_{k+1} & \\ & V_k \end{pmatrix} Q_k^T = \begin{pmatrix} U1_k & \bar{u}_k & U2_k \\ V1_k & \bar{v}_k & V2_k \end{pmatrix}, \tag{4.6}$$

where the next two transformations defining $Q_{k+1}$ leave $U1_k$, $U2_k$, $V1_k$, $V2_k$ unaltered. It follows that

$$\begin{aligned}
\begin{pmatrix} p_k \\ \delta x_k \end{pmatrix} &= \begin{pmatrix} Ax_k \\ \delta x_k \end{pmatrix} = \begin{pmatrix} AV_k \\ \delta V_k \end{pmatrix} y_k = \begin{pmatrix} U_{k+1}B_k \\ \delta V_k \end{pmatrix} y_k \\
&= \begin{pmatrix} U_{k+1} & \\ & V_k \end{pmatrix} Q_k^T Q_k \begin{pmatrix} B_k \\ \delta I \end{pmatrix} y_k \\
&= \begin{pmatrix} U1_k & \bar{u}_k & U2_k \\ V1_k & \bar{v}_k & V2_k \end{pmatrix} \begin{pmatrix} z_k \\ 0 \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} U1_k z_k \\ V1_k z_k \end{pmatrix},
\end{aligned}$$

and

$$\begin{aligned}
\begin{pmatrix} r_k \\ -\delta x_k \end{pmatrix} &= \begin{pmatrix} b \\ 0 \end{pmatrix} - \begin{pmatrix} Ax_k \\ \delta x_k \end{pmatrix} \\
&= \begin{pmatrix} U_{k+1} & \\ & V_k \end{pmatrix} Q_k^T Q_k \left\{ \begin{pmatrix} \beta_1 e_1 \\ 0 \end{pmatrix} - \begin{pmatrix} B_k \\ \delta I \end{pmatrix} y_k \right\} \\
&= \begin{pmatrix} U1_k & \bar{u}_k & U2_k \\ V1_k & \bar{v}_k & V2_k \end{pmatrix} \left\{ \begin{pmatrix} z_k \\ \bar{\zeta}_k \\ q_k \end{pmatrix} - \begin{pmatrix} R_k \\ 0 \\ 0 \end{pmatrix} y_k \right\} \\
&= \begin{pmatrix} \bar{u}_k & U2_k \\ \bar{v}_k & V2_k \end{pmatrix} \begin{pmatrix} \bar{\zeta}_k \\ q_k \end{pmatrix}.
\end{aligned}$$

Thus, the sequences $\{p_k\}$, $\{r_k\}$ and $\{\delta x_k\}$ are obtained by orthogonal steps:

$$p_k = U1_k z_k, \tag{4.7}$$

$$r_k = U2_k q_k + \bar{\zeta}_k \bar{u}_k, \tag{4.8}$$

$$\delta x_k = V1_k z_k, \tag{4.9}$$

$$-\delta x_k = V2_k q_k + \bar{\zeta}_k \bar{v}_k. \tag{4.10}$$

We see that the "damped" projections have led to two new sequences for approximating $x$. We shall denote these by $\{x1_k\}$ and $\{x2_k\}$. To use (4.7)–(4.10) in the usual way, we form

$$p_k = p_{k-1} + \zeta_k u1_k, \tag{4.11}$$

$$\widehat{r}_k = \widehat{r}_{k-1} + \psi_k u2_k, \tag{4.12}$$

$$\widehat{x1}_k = \widehat{x1}_{k-1} + \zeta_k v1_k, \tag{4.13}$$

$$\widehat{x2}_k = \widehat{x2}_{k-1} + \psi_k v2_k, \tag{4.14}$$

and upon termination at step $k$ we make some final adjustments:

$$r_k = \widehat{r}_k + \bar{\zeta}_k \bar{u}_k, \tag{4.15}$$

$$x1_k = (1/\delta)\widehat{x1}_k, \tag{4.16}$$

$$x2_k = -(1/\delta)(\widehat{x2}_k + \bar{\zeta}_k \bar{v}_k). \tag{4.17}$$

*4.2 Discussion.*

1. The approximations $x_k$, $p_k$ and $r_k$ are defined for all $\delta \geq 0$, but $x1_k$ and $x2_k$ require $\delta > 0$.

2. In (4.16)–(4.17), the divisions by $\delta$ may appear hazardous as $\delta \to 0$. However, the norm of each column of $V1_k$ and $V2_k$ is of order $\delta$, and $\|z_k\|$, $\|q_k\|$ and $|\bar{\zeta}_k|$ are all bounded by $\|b\|$. Values as small as $\delta = 10^{-10}$ (say) seem to be safe in practice. Hence, $x1_k$ or $x2_k$ may be used to estimate $x$ for both normal and damped least squares.

3. The Golub-Kahan process requires work vectors $u$ and $v$ ($m + n$ storage locations) and $3m + 3n$ floating-point operations (flops) per step, as well as the usual products $u \leftarrow Av + u$, $v \leftarrow A^T u + v$.

4. Table 4.1 shows the additional storage and work needed to estimate various vectors. For example, to estimate $x$, LSQR uses work vectors $x$ and $w$ ($2n$ storage locations) and $2n$ flops per step, for all values of $\delta$. The other quantities are somewhat more expensive.

5. To implement reliable stopping rules, LSQR uses the vectors $w_k$ to estimate $\mathrm{cond}(\bar{A})$. When $x$ is being estimated, this involves no additional storage and $2n$ additional flops per step. If $p$, $r$, $x1$ or $x2$ are estimated but not $x$, the extra cost to estimate $\mathrm{cond}(\bar{A})$ is $n$ locations and $3n$ flops per step.

6. $x1$ is slightly cheaper to compute than $x2$, and to date the computational results have not favored one over the other. It is probably sufficient to consider $x1$.

In summary, computing all of $p$, $r$ and $x1$ requires about twice the storage and work compared to the usual LSQR $x$. This may not be significant if the matrix-vector products dominate.

## 5 Relationship to Craig's method.

Craig's method [4, 5] solves *compatible* rectangular systems of the form

$$\min \|x\| \quad \text{subject to} \quad Ax = b, \tag{5.1}$$

where we typically have $m \leq n$ and $\mathrm{rank}(A) = m$. As described in [10, 11], the method may be implemented via $\mathrm{Bidiag}(A, b)$, the Golub-Kahan bidiagonalization of $A$ with starting vector $b$. This seems to be a reliable approach, but an outstanding question has been: What if the right-hand side is of the

Table 4.1: Storage and work per step needed (excluding the bidiagonalization) to estimate the normal LSQR solution $x$, the projections $p$ and $r$, and the new solution estimates *x1* and *x2*.

| | Vectors | Storage | Work $\delta = 0$ | Work $\delta > 0$ |
|---|---|---|---|---|
| $x$ | $x, w$ | $2n$ | $2n$ | $2n$ |
| $p$ | $p, \bar{u}$ | $2m$ | $3m$ | $5m$ |
| $r$ | $r, \bar{u}$ | $2m$ | $2m$ | $4m$ |
| $p$ and $r$ | $p, r, \bar{u}$ | $3m$ | $4m$ | $6m$ |
| *x1* | *x1*, $\bar{v}$ | $2n$ | | $4n$ |
| *x2* | *x2*, $\bar{v}$ | $2n$ | | $5n$ |

form $b = Ac$? The method is then using Bidiag$(A^T, Ac)$, which is *not* a reliable approach [11, 3].

This curiosity is now resolved by noting that when $b = Ac$, the solution to (5.1) is $x = A(AA^T)^{-1}Ac$, which is the projection $p = Pc$ associated with the least-squares problem $\min_y \|c - A^T y\|$. The method of Section 3 may be applied. Similarly, minimum-length problems of the form

$$\min \|x\|^2 + \|s\|^2 \quad \text{subject to} \quad Ax + \delta s = b, \tag{5.2}$$

may be treated by LSQR or by an extension of Craig's method as described in [13], but if $b = Ac$, then the method of Section 4 may be applied to compute $(x, s)$ as a projection.

## 6 Computational results.

The test problems described in [11] were generalized slightly to include damping and arbitrary values of $m$ and $n$. They use a matrix of the form $A = YDZ$, where $Y$ and $Z$ are Householder transformations and $D$ is diagonal with prescribed singular values. Preliminary conclusions follow.

Note that when $m = n$ and $\delta = 0$, the exact projections are $p = b$ and $r = 0$. Also, when $\delta = 0$, *x1* and *x2* are undefined. These cases were not considered.

For the results obtained, the machine precision was $\epsilon \approx 10^{-16}$; the damping parameter was in the range $10^{-11} \le \delta \le 10^{-8}$; $\|A\|$, $\|b\|$ and $\|x\|$ were all $O(1)$; and the condition of the "damped" matrix was in the range $10^6 \le \text{cond}(\bar{A}) \le 10^{11}$. The stopping tolerances for LSQR were $\mathtt{atol} = \mathtt{btol} = \epsilon^{0.9} \approx 10^{-14}$.

Below, $p$, $r$, $x$, *x1* and *x2* mean the final computed estimates of $p$, $r$ and $x$.

### 6.1 *Observations.*

1. When $m = n$ and $\|r\| = O(\epsilon)$, the errors in $p$ and $r$ were $O(\mathtt{atol})$, and the errors in $x$, *x1* and *x2* grew in proportion to $\text{cond}(\bar{A})$. This matches the sensitivity of the problem itself, indicating stability [7].

2. When $m > n$ or $m < n$ and $\|r\| = O(10^{-6})$, the same results were observed.

3. When $m > n$ and $\|r\| > 10^{-3}$, the errors in $x$, *x1* and *x2* grew in proportion to $\mathrm{cond}(\bar{A})^2$. Again this matches the sensitivity of least-squares problems.

4. In the same cases ($\|r\|$ large), the errors in $p$ and $r$ grew with $\mathrm{cond}(\bar{A})$ in accordance with sensitivity analysis, but they were significantly smaller than could be expected from the actual size of $\mathrm{cond}(\bar{A})$.

5. The final $p$ and $r$ closely matched $Ax$ and $b - Ax$ computed from the final LSQR estimate of $x$.

6. Surprisingly, this was true even when $x$ had essentially no digits of precision.

7. More surprisingly, the three estimates $x$, *x1* and *x2* matched each other very closely in all cases, even when they all had no correct digits. In extreme cases, $x$ and *x1* agreed more closely than $x$ and *x2*.

Support for Observations 4 and 5 has been given by Björck *et al.* [1, 3], who study the "recursive residuals" for various CG methods including CGLS, the original least-squares algorithm of Hestenes and Stiefel [9]. For updates such as (1.7), the recursive residuals are defined by

$$\begin{aligned} x_k &= x_{k-1} + \zeta_k w_k, \\ \tilde{r}_k &= \tilde{r}_{k-1} - \zeta_k A w_k, \end{aligned} \tag{6.1}$$

where we use $\tilde{r}_k$ to distinguish from $r_k$ in Sections 3–4. In CGLS the residuals are an integral part of the iteration. In LSQR they are not normally needed, but they may be computed for interest.

Following Greenbaum [8], Björck *et al.* [3] prove for CGLS and LSQR that $\tilde{r}_k$ closely approximates $b - Ax_k$ for all $k$. This matches Observation 5.

They also conjecture from experimental evidence that $\tilde{r}_k$ is ultimately very close to the true residual $r$. This is confirmed by Observation 4; for example, with $\mathrm{cond}(\bar{A}) = 10^{11}$ and $\|r\| = 10$, the final value of $\|r - \tilde{r}_k\|/\|r\|$ was $10^{-9}$ rather than the expected $10^{-5}$.

## 7 Conclusions.

We have shown how to obtain projections $p = Ax$ and $r = b - Ax$ from the Golub-Kahan process, as well as two different estimates of $x$, using orthogonal steps for all quantities. We were motivated by the concern that updates of the form (6.1) could entail significant cancellation if both $\zeta_k$ and $\|w_k\|$ are large.

In LSQR, we know that some of the vectors $w_k$ can be large, because $\|W_k\|$ is used to estimate $\mathrm{cond}(A)$. However, for the present test problems the corresponding multipliers $\zeta_k$ were always small (see [13]). Thus, we have not yet seen a benefit from obtaining $p$, $r$, *x1* and *x2* by orthogonal steps.

Since the new approach for computing projections involves additional work and storage, it is probably best to compute $x$ via the standard CGLS or LSQR iterations and then form $p$ or $r$ directly. We recommend this even in ill-conditioned cases where the computed $x$ has no accuracy. If cases arise in which the errors in $p$, $r$ or $x$ exceed whatever can be expected from $\mathrm{cond}(A)$, the methods of this paper should be reconsidered.

**Acknowledgements.**

As in the past, I am grateful to Chris Paige for showing how to work with the Lanczos and Golub-Kahan processes. I am also grateful to Åke Björck for bringing [3] to my attention and causing a closer look at the computed residuals (*cf.* Observation 4).

## REFERENCES

1. Å. Björck, *Conjugate gradient methods for sparse least squares problems*, unpublished notes, Stanford University, 1979.
2. Å. Björck, *A bidiagonalization algorithm for solving ill-posed systems of linear equations*, Report LITH-MAT-R-80-33, Dept. of Mathematics, Linköping University, Linköping, Sweden, 1980.
3. Å. Björck, T. Elfving, and Z. Strakoš, *Stability of conjugate gradient and Lanczos methods for linear least squares problems*, SIAM J. Matrix Anal. Appl., to appear.
4. J. E. Craig, *The N-step iteration procedures*, J. Math. and Phys., 34, 1 (1955), pp. 64–73.
5. D. K. Faddeev and V. N. Faddeeva, *Computational Methods of Linear Algebra*, Freeman, London, 1963.
6. G. H. Golub and W. Kahan, *Calculating the singular values and pseudoinverse of a matrix*, SIAM J. Numer. Anal., 2 (1965), pp. 205–224.
7. G. H. Golub and C. F. Van Loan, *Matrix Computations*, Second Edition, The Johns Hopkins University Press, Baltimore and London, 1989.
8. A. Greenbaum, *Estimating the attainable accuracy of recursively computed residual methods*, SIAM J. Matrix Anal. Appl., to appear.
9. M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Stds., B49 (1952), pp. 409–436.
10. C. C. Paige, *Bidiagonalization of matrices and solution of linear equations*, SIAM J. Numer. Anal., 11 (1974), pp. 197–209.
11. C. C. Paige and M. A. Saunders, *LSQR: An algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Software, 8(1) (1982), pp. 43–71.
12. C. C. Paige and M. A. Saunders, *Algorithm 583. LSQR: Sparse linear equations and least squares problems*, ACM Trans. Math. Software, 8(2) (1982), pp. 195–209.
13. M. A. Saunders, *Solution of sparse rectangular systems using LSQR and CRAIG*, BIT, 35 (1995), pp. 588–604.