ALGORITHM 583
LSQR: Sparse Linear Equations and Least Squares Problems

CHRISTOPHER C. PAIGE
McGill University, Canada
and
MICHAEL A. SAUNDERS
Stanford University

Categories and Subject Descriptors: G.1.3 [**Numerical Analysis**] Numerical Linear Algebra—*linear systems* (*direct and iterative methods*), G.3 [**Mathematics of Computing**]: Probability and Statistics—*statistical computing, statistical software*, G.m [**Mathematics of Computing**]: Miscellaneous—*FORTRAN program units*

General Terms Algorithms

Additional Key Words and Phrases: Analysis of variance, conjugate-gradient method, least squares, linear equations, regression, sparse matrix

---

1. INTRODUCTION

LSQR finds a solution x to the following problems:

Unsymmetric equations:	solve	$Ax = b$	(1.1)
Linear least squares:	minimize	$\|Ax - b\|_2$	(1.2)
Damped least squares:	minimize	$\left\| \begin{bmatrix} A \\ \lambda I \end{bmatrix} x - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\|_2$	(1.3)

where A is a matrix with m rows and n columns, b is an m-vector, λ is a scalar, and the given data A, b, λ are real. The matrix A will normally be large and sparse. It is defined by means of a user-written subroutine APROD, whose

---

Received 4 June 1980; revised 23 September 1981, accepted 28 February 1982

This work was supported by Natural Sciences and Engineering Research Council of Canada Grant A8652, by the New Zealand Department of Scientific and Industrial Research; and by U S. National Science Foundation Grants MCS-7926009 and ECS-8012974, the Department of Energy under Contract AM03-76SF00326, PA No. DE-AT03-76ER72018, the Office of Naval Research under Contract N00014-75-C-0267, and the Army Research Office under Contract DAA29-79-C-0110.

Authors' addresses: C. C. Paige, School of Computer Science, McGill University, Montreal, Quebec, Canada H3A 2K6; M. A Saunders, Department of Operations Research, Stanford University, Stanford, CA 94305.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

© 1982 ACM 0098-3500/82/0600-0195 $00 75

Table I. Comparison of CGLS and LSQR

	Storage	Work per iteration
CGLS, $\lambda = 0$	$2m + 2n$	$2m + 3n$
CGLS, $\lambda \neq 0$	$2m + 2n$	$2m + 5n$
LSQR, any λ	$m + 2n$	$3m + 5n$

essential function is to compute products of the form Ax and $A^T y$ for given vectors x and y.

Problems (1.1) and (1.2) are treated as special cases of (1.3), which we shall write as

$$\min \|\bar{A}x - \bar{b}\|_2, \qquad \bar{A} = \begin{bmatrix} A \\ \lambda I \end{bmatrix}, \qquad \bar{b} = \begin{bmatrix} b \\ 0 \end{bmatrix}. \tag{1.4}$$

An earlier successful method for such problems is the *conjugate-gradient method for least squares systems* given by Hestenes and Stiefel [3]. (This method is described as algorithm CGLS in [6, sect. 7.1].) CGLS and LSQR are iterative methods with similar qualitative properties. Their computational requirements are summarized in Table I. In addition they require a product Ax and a product $A^T y$ each iteration.

In order to achieve the storage shown for LSQR, we ask the user to implement the matrix-vector products in the form

$$y \leftarrow y + Ax \qquad \text{and} \qquad x \leftarrow x + A^T y, \tag{1.5}$$

where $\leftarrow$ means that one of the given vectors is overwritten by the expression shown. (A parameter specifies which expression the user's subroutine APROD should compute on any given entry.) We see that LSQR has a storage advantage if the operations (1.5) can be performed with no additional storage beyond that required to represent A. For least squares applications with many observations ($m \gg n$), this could be useful.

The work shown in Table I is the number of floating-point multiplications per iteration, excluding the work involved in the products Ax, $A^T y$. Since CGLS is somewhat more efficient, we would not discourage using that method whenever A or $\bar{A}$ is well conditioned. However, LSQR is likely to obtain a more accurate solution in fewer iterations if $\bar{A}$ is moderately or severely ill-conditioned.

Let $\bar{r}_k = \bar{b} - \bar{A}x_k$ be the residual vector associated with the kth iteration. LSQR provides estimates of $\|x_k\|_2$, $\|\bar{r}_k\|_2$, $\|\bar{A}^T\bar{r}_k\|_2$, the norm of $\bar{A}$, the condition number of $\bar{A}$, and standard errors for the components of x. The last two items require a further $2n$ multiplications per iteration and an additional n-vector of storage.

Subroutine LSQR is written in the PFORT subset of American National Standard FORTRAN. It contains no machine-dependent constants. Auxiliary routines required are APROD, NORMLZ, SCOPY, SNRM2, and SSCAL. The last three correspond to members of the BLAS collection [5].

2. MATHEMATICAL BACKGROUND

Algorithmic details are given in [6], mainly for the case $\lambda = 0$. We summarize these here with λ reintroduced, and show that a given value of λ may be dealt with at negligible cost. The vector norm $\|v\|_2 = (v^T v)^{1/2}$ is used throughout.

LSQR uses an algorithm of Golub and Kahan to reduce A to lower bidiagonal form. The quantities produced from A and b after $k + 1$ steps of the bidiagonalization (procedure Bidiag 1 [6]) are

$$\beta_1 = \|b\|, \quad U_{k+1} = [u_1, u_2, \dots, u_{k+1}], \quad V_{k+1} = [v_1, v_2, \dots, v_{k+1}], \qquad B_k = \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \ddots & \\ & & \ddots & \alpha_k \\ & & & \beta_{k+1} \end{bmatrix}. \tag{2.1}$$

The kth approximation to the solution x is then defined to be $x_k = V_k y_k$, where y_k solves the subproblem

$$\min \left\| \begin{bmatrix} B_k \\ \lambda I \end{bmatrix} y_k - \begin{bmatrix} \beta_1 e_1 \\ 0 \end{bmatrix} \right\|. \tag{2.2}$$

Letting the associated residual vectors be

$$\begin{aligned} t_{k+1} &= \beta_1 e_1 - B_k y_k \\ r_k &= b - A x_k \\ \bar{r}_k &= \bar{b} - \bar{A} x_k, \end{aligned} \tag{2.3}$$

we find that the relations

$$\begin{aligned} r_k &= U_{k+1} t_{k+1} \\ A^T r_k &= \lambda^2 x_k + \alpha_{k+1} \tau_{k+1} v_{k+1} \end{aligned} \tag{2.4}$$

will hold to machine accuracy, where τ_{k+1} is the last component of t_{k+1}, and we therefore conclude that (r_k, x_k) will be an acceptable solution of (1.4) if the computed value of either $\|t_{k+1}\|$ or $|\alpha_{k+1}\tau_{k+1}|$ is suitably small.

Bjorck [1] has previously observed that subproblem (2.2) is the appropriate generalization of $\min\|B_k y_k - \beta_1 e_1\|$, when $\lambda \neq 0$. He also discusses methods for computing y_k and x_k efficiently for various λ and k.

In LSQR we assume that a single value of λ is given, and to save storage and work, we do not compute y_k, r_k, or t_{k+1}. The orthogonal factorization

$$Q_k \begin{bmatrix} B_k & \beta_1 e_1 \\ \lambda I & 0 \end{bmatrix} = \begin{bmatrix} R_k & f_k \\ 0 & \bar{\phi}_{k+1} \\ 0 & q_k \end{bmatrix} \tag{2.5}$$

is computed ($Q_k^T Q_k = I$; R_k upper bidiagonal, $k \times k$) and this would give $R_k y_k = f_k$, but instead we solve $R_k^T D_k^T = V_k^T$ and form $x_k = D_k f_k$.

The factorization (2.5) is formed similarly to the case $\lambda = 0$ in [6], except that two rotations are required per step instead of one. For $k = 2$, the factorization proceeds according to

$$\begin{bmatrix} \alpha_1 & & \beta_1 \\ \beta_2 & \alpha_2 & \\ & \beta_3 & \\ \lambda & & \\ & \lambda & \end{bmatrix} \rightarrow \begin{bmatrix} \tilde{\rho}_1 & & \tilde{\phi}_1 \\ \beta_2 & \alpha_2 & \\ & \beta_3 & \\ & & \psi_1 \\ & \lambda & \end{bmatrix} \rightarrow \begin{bmatrix} \rho_1 & \theta_2 & \phi_1 \\ & \bar{\rho}_2 & \bar{\phi}_2 \\ & \beta_3 & \\ & & \psi_1 \\ & \lambda & \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} \rho_1 & \theta_2 & \phi_1 \\ & \tilde{\rho}_2 & \tilde{\phi}_2 \\ & \beta_3 & \\ & & \psi_1 \\ & & \psi_2 \end{bmatrix} \rightarrow \begin{bmatrix} \rho_1 & \theta_2 & \phi_1 \\ & \rho_2 & \phi_2 \\ & & \bar{\phi}_3 \\ & & \psi_1 \\ & & \psi_2 \end{bmatrix}.$$

Note that the first λ is rotated into the diagonal element α_1. This alters the right-hand side $\beta_1 e_1$ to produce ψ_1, the first component of q_k. An alternative is to rotate λ into β_2 (and similarly for later λ), since this does not affect the right-hand side and it more closely simulates the algorithm that results when LSQR is applied to $\bar{A}$ and $\bar{b}$ directly. However, the rotations then have a greater effect on B_k, and in practice the first option has proved to give marginally more accurate results.

The estimates required to implement the stopping criteria are

$$\|\bar{r}_k\|^2 = \|r_k\|^2 + \lambda^2\|x_k\|^2 \approx \bar{\phi}_{k+1}^2 + \|q_k\|^2,$$

$$\|\bar{A}^T\bar{r}_k\| = \|A^T r_k - \lambda^2 x_k\| \approx \left|\frac{\alpha_{k+1}\beta_{k+1}\phi_k}{\rho_k}\right|.$$

This is a simple generalization of the case $\lambda = 0$. No additional storage is needed for q_k, since only its norm is required. In short, although the presence of λ complicates the algorithm description, it adds essentially nothing to the storage and work per iteration.

3. REGULARIZATION AND RELATED WORK

Introducing λ as in (1.3) is just one way of "regularizing" the solution x, in the sense that it can reduce the size of the computed solution and make its components less sensitive to changes in the data. LSQR is applicable when a value of λ is known a priori. The value is entered via the subroutine parameter DAMP. A second method for regularizing x is available through LSQR's parameter ACOND, which can cause iterations to terminate before $\|x_k\|$ becomes large. A similar approach has recently been described by Wold et al. [9], who give an illuminating interpretation of the bidiagonalization as a partial least squares procedure. Their description will also be useful to those who prefer the notation of multiple regression.

Methods for choosing λ, and other approaches to regularization, are given in [1, 2, 4, 8] and elsewhere. For a philosophical discussion, see [7].

4. CODING APROD

The best way to compute $y + Ax$ and $x + A^T y$ depends upon the origin of the matrix A. We shall illustrate a case that commonly arises, in which A is a sparse matrix whose nonzero coefficients are stored by *rows* in a simple list. Let A have

M rows, N columns, and NZ nonzeros. Conceptually we need three arrays dimensioned as REAL RA(NZ) and INTEGER JA(NZ), NA(M),.where

RA(L) is the Lth nonzero of A, counting across row 1, then across row 2, and so on;
JA(L) is the column in which the Lth nonzero of A lies;
NA(I) is the number of nonzero coefficients in the Ith row of A.

These quantities may be used in a straightforward way, as shown in Figure 1 (a FORTRAN implementation). We assume that they are made available to APROD through COMMON, and that the actual array dimensions are suitably large.

Blank or labeled COMMON will often be convenient for transmitting data to APROD. (Of course, some of the data could be local to APROD.) For greater generality, the parameter lists for LSQR and APROD include two workspace arrays IW, RW and their lengths LENIW, LENRW. LSQR does not use these parameters directly; it just passes them to APROD.

Figure 2 illustrates their use on the same example (sparse A stored by rows). An auxiliary subroutine APROD1 is needed to make the code readable. A similar scheme should be used to initialize the workspace parameters prior to calling LSQR.

Returning to the example itself, it may often be natural to store A by *columns* rather than rows, using analogous data structures. However, we note that in sparse least squares applications, A may have many more rows than columns (M $\gg$ N). In such cases it is vital to store A by rows as shown, if the machine being used has a paged (virtual) memory. Random access is then restricted to arrays of length N rather than M, and page faults will therefore be kept to a minimum.

Note also that the arrays RA, JA, NA are adequate for computing both Ax and A^Ty; we do not need to store A by rows *and* by columns.

Regardless of the application, it will be apparent when coding APROD for the two values of MODE that the matrix A is effectively being defined *twice*. Great care must be taken to avoid coding inconsistent expressions $y + A_1x$ and $x + A_2^Ty$, where either A_1 or A_2 is different from the desired A. (If $A_1 \neq A_2$, algorithm LSQR will not converge.) Parameters ANORM, ACOND, and CONLIM provide a safeguard for such an event.

5. PRECONDITIONING

It is well known that conjugate-gradient methods can be accelerated if a nonsingular matrix M is available to approximate A in some useful sense. When A is square and nonsingular, the system $Ax = b$ is equivalent to both of the following systems:

$$(M^{-1}A)x = c \quad \text{where} \quad Mc = b; \tag{5.1}$$

$$(AM^{-1})z = b \quad \text{where} \quad Mx = z. \tag{5.2}$$

For least squares systems (undamped), only the analogue of (5.2) is applicable:

$$\min\|Ax - b\|_2 = \min\|(AM^{-1})z - b\|_2, \qquad \text{where} \quad Mx = z. \tag{5.3}$$

```
      SUBROUTINE APROD( MODE,M,N,X,Y,
     *                  LENIW,LENRW,IW,RW )
C
      INTEGER    MODE,M,N,LENIW,LENRW
      INTEGER    IW(LENIW)
      REAL       X(N),Y(M),RW(LENRW)
C
C     APROD  PERFORMS THE FOLLOWING FUNCTIONS:
C
C        IF MODE = 1, SET  Y = Y + A*X
C        IF MODE = 2, SET  X = X + A(TRANSPOSE)*Y
C
C     WHERE  A  IS A MATRIX STORED BY ROWS IN
C     THE ARRAYS  RA, JA, NA.  IN THIS EXAMPLE,
C     RA, JA, NA  ARE STORED IN COMMON.
C
      REAL       RA
      INTEGER    JA,NA
      COMMON     RA(9000),JA(9000),NA(1000)
C
C     INTEGER    I,J,L,L1,L2
      REAL       SUM,YI,ZERO
C
      ZERO = 0.0
      L2   = 0
      IF (MODE .NE.1) GO TO 400
C
C     ------------------------------------------
C     MODE = 1  --  SET  Y = Y + A*X.
C     ------------------------------------------
      DO 200 I = 1, M
         SUM = ZERO
         L1  = L2 + 1
         L2  = L2 + NA(I)
         DO 100 L = L1, L2
            J   = JA(L)
            SUM = SUM + RA(L)*X(J)
  100    CONTINUE
         Y(I) = Y(I) + SUM
  200 CONTINUE
      RETURN
C
C     ------------------------------------------
C     MODE = 2  --  SET  X = X + A(TRANSPOSE)*Y.
C     ------------------------------------------
  400 DO 600 I = 1, M
         YI = Y(I)
         L1 = L2 + 1
         L2 = L2 + NA(I)
         DO 500 L = L1, L2
            J    = JA(L)
            X(J) = X(J) + RA(L)*YI
  500    CONTINUE
  600 CONTINUE
      RETURN
C
C     END OF APROD
      END
```

Fig. 1. Computation of $y + Ax$, $x + A^T y$, where A is a sparse matrix stored compactly by rows. For convenience, the data structure for A is held in COMMON.

```
      SUBROUTINE APROD( MODE,M,N,X,Y,
     *                  LENIW,LENRW,IW,RW )
C
      INTEGER    MODE,M,N,LENIW,LENRW
      INTEGER    IW(LENIW)
      REAL       X(N),Y(M),RW(LENRW)
C
C     APROD  PERFORMS THE FOLLOWING FUNCTIONS:
C
C       IF MODE = 1, SET  Y = Y + A*X
C       IF MODE = 2, SET  X = X + A(TRANSPOSE)*Y
C
C     WHERE  A  IS A MATRIX STORED BY ROWS IN
C     THE ARRAYS  RA, JA, NA.  IN THIS EXAMPLE,
C     APROD  IS AN INTERFACE BETWEEN  LSQR  AND
C     ANOTHER USER ROUTINE THAT DOES THE WORK.
C     THE WORKSPACE ARRAY RW CONTAINS RA.
C     THE FIRST M COMPONENTS OF IW CONTAIN NA,
C     AND THE REMAINDER OF IW CONTAINS JA.
C     THE DIMENSIONS OF RW AND IW ARE ASSUMED
C     TO BE SUFFICIENTLY LARGE.
C
      INTEGER    LENJA,LENRA,LOCJA
C
      LOCJA  = M + 1
      LENJA  = LENIW - LOCJA + 1
      LENRA  = LENRW
      CALL APROD1( MODE,M,N,X,Y,
     *             LENJA,LENRA,IW,IW(LOCJA),RW )
      RETURN
C
C     END OF APROD
      END
```

Fig. 2 Same as Figure 1, with the data structure for A held in the workspace parameters.

```
      SUBROUTINE APROD1( MODE,M,N,X,Y,
     *                   LENJA,LENRA,NA,JA,RA )
C
      INTEGER    MODE,M,N,LENJA,LENRA
      INTEGER    NA(M),JA(LENJA)
      REAL       X(N),Y(M),RA(LENRA)
C
C     APROD1  DOES THE WORK FOR  APROD.
C
C     INTEGER    I,J,L,L1,L2
      REAL       SUM,YI,ZERO
C
C
      < the same code as in APROD in Figure 1 >
C
C
C     END OF APROD1
      END
```

```
LEAST-SQUARES TEST PROBLEM      P(   20   10    1    1    1.00E-03 )

CONDITION NO. =  9.9995E 00     RESIDUAL FUNCTION =  9.812157000E-01

                    LSQR   --   LEAST-SQUARES SOLUTION OF  A*X = B

                    THE MATRIX  A  HAS    20 ROWS   AND    10 COLS
                    THE DAMPING PARAMETER IS     DAMP    =  1.00E-03

                    ATOL    =  1.00E-06          CONLIM =  1.00E 02
                    BTOL    =  1.00E-06          ITNLIM =        80
```

ITN	X(1)	FUNCTION	COMPATIBLE	INCOMPATIBLE	NORM(ABAR)	COND(ABAR)
0	0.0000000000E-01	6.3410580000E 00	1.000E 00	9.135E-02		
1	-6.3564250000E-01	4.0387670000E 00	6.369E-01	7.244E-01	7.51E-01	1.00E 00
2	-4.7282630000E-01	2.3303970000E 00	3.675E-01	3.349E-01	1.10E 00	2.42E 00
3	-2.8075080000E-01	1.8962160000E 00	2.990E-01	2.462E-01	1.30E 00	3.63E 00
4	2.6825070000E-01	1.5905200000E 00	2.508E-01	1.424E-01	1.49E 00	5.27E 00
5	1.2649560000E 00	1.4032960000E 00	2.213E-01	1.160E-01	1.59E 00	7.09E 00
6	2.0648040000E 00	1.2910880000E 00	2.036E-01	9.273E-02	1.70E 00	9.03E 00
7	3.0031450000E 00	1.2072930000E 00	1.904E-01	8.294E-02	1.79E 00	1.12E 01
8	3.7526340000E 00	1.1551220000E 00	1.822E-01	5.138E-02	1.90E 00	1.34E 01
9	5.4443550000E 00	1.0780640000E 00	1.700E-01	3.155E-02	1.95E 00	1.72E 01
10	8.9918140000E 00	9.8151740000E-01	1.548E-01	1.283E-02	1.96E 00	2.44E 01
11	8.9998990000E 00	9.8120520000E-01	1.547E-01	2.017E-04	2.20E 00	2.75E 01
12	8.9999290000E 00	9.8120540000E-01	1.547E-01	4.361E-06	2.38E 00	2.98E 01
13	8.9999280000E 00	9.8120550000E-01	1.547E-01	9.078E-07	2.48E 00	3.13E 01

```
NO. OF ITERATIONS =    13         STOPPING CONDITION =  2

THE LEAST-SQRS SOLN IS GOOD ENOUGH, GIVEN  ATOL

                  RESIDUAL NORM (ABAR*X - BBAR)   RESIDUAL NORM (NORMAL EQNS)   SOLUTION NORM (X)
                  -----------------------------   ---------------------------   -----------------

ESTIMATED BY LSQR          9.812055E-01                  2.206693E-06              1.688187E 01
COMPUTED FROM X            9.812157E-01                  1.083419E-05              1.688184E 01


SOLUTION
    1   8.99993       2   7.99997       3   6.99998       4   5.99999       5   4.99999
    6   3.99999       7   3.00000       8   2.00000       9  0.999994      10 -0.204206E-05

STANDARD ERRORS
    1   2.11589       2  0.888101       3  0.685644       4  0.556184       5  0.614104
    6  0.409182       7  0.565480       8  0.519385       9  0.375466      10  0.589787
```

Fig. 3. Example output from test program and LSQR on a damped least squares problem.

We note only that *subroutine LSQR may be applied without change* to systems (5.1)–(5.3). The effect of M is localized to the user's own subroutine APROD. For example, when MODE = 1, APROD for the last two systems should compute $y + (AM^{-1})x$ by first solving $Mw = x$ and then computing $y + Aw$. Clearly it must be possible to solve systems involving M and M^T very efficiently.

6. OUTPUT

Subroutine LSQR produces printed output on file NOUT, if the parameter NOUT is positive. This is illustrated in Figure 3, in which the least squares problem solved is $P(20, 10, 1, 1)$ as defined in [6], with a slight generalization to include a damping parameter $\lambda = 10^{-3}$. (Single precision was used on an IBM 370/168.) The items printed at the kth iteration are as follows.

ITN	The iteration number k. Results are always printed for the first 10 and last 10 iterations. Intermediate results are printed if $m \le 40$ or $n \le 40$, or if one of the convergence conditions is nearly satisfied. Otherwise, information is printed every 10th iteration.
X(1)	The value of the first element of the approximate solution x_k.
FUNCTION	The value of the function being minimized, namely $\lVert \bar{r}_k \rVert = (\lVert r_k \rVert^2 + \lambda^2 \lVert x_k \rVert^2)^{1/2}$.
COMPATIBLE	A dimensionless quantity which should converge to zero *if and only if* $Ax = b$ is compatible. It is an estimate of $\lVert \bar{r}_k \rVert / \lVert b \rVert$, which decreases monotonically.
INCOMPATIBLE	A dimensionless quantity which should converge to zero *if and only if* the optimum $\lVert \bar{r}_k \rVert$ is nonzero. It is an estimate of $\lVert \bar{A}^T \bar{r}_k \rVert / (\lVert \bar{A} \rVert_F \lVert \bar{r}_k \rVert)$, which is usually *not* monotonic.
NORM(ABAR)	A monotonically increasing estimate of $\lVert \bar{A} \rVert_F$.
COND(ABAR)	A monotonically increasing estimate of $\text{cond}(\bar{A}) = \lVert \bar{A} \rVert_F \lVert \bar{A}^+ \rVert_F$, the condition number of $\bar{A}$.

ACKNOWLEDGMENT

The authors are grateful to Richard Hanson for suggestions that prompted several improvements to the implementation of LSQR.

REFERENCES

1. BJORCK, Å. A bidiagonalization algorithm for solving ill-posed systems of linear equations Rep LITH-MAT-R-80-33, Dep. Mathematics, Linkoping Univ., Linkoping, Sweden, 1980.
2 ELDÉN, L. Algorithms for the regularization of ill-conditioned least squares problems *BIT 17* (1977), 134–145.
3. HESTENES, M.R , AND STIEFEL, E Methods of conjugate gradients for solving linear systems. *J Res. N.B.S. 49* (1952), 409–436.
4 LAWSON, C.L , AND HANSON, R.J *Solving Least Squares Problems.* Prentice-Hall, Englewood Cliffs, N.J., 1974.
5 LAWSON, C.L , HANSON, R J., KINCAID, D.R , AND KROGH, F T Basic linear algebra subprograms for Fortran usage *ACM Trans Math Softw 5*, 3 (Sept 1979), 308–323 and (Algorithm) 324–325.
6. PAIGE, C.C., AND SAUNDERS, M A LSQR An algorithm for sparse linear equations and sparse least squares *ACM Trans Math Softw 8*, 1 (March 1982), 43–71

7. SMITH, G., AND CAMPBELL, F. A critique of some ridge regression methods. *J. Am. Stat. Assoc. 75*, 369 (March 1980), 74-81
8 VARAH, J.M A practical examination of some numerical methods for linear discrete ill-posed problems *SIAM Rev. 21* (1979), 100-111.
9. WOLD, S., WOLD, H., DUNN, W.J., AND RUHE, A. The collinearity problem in linear and nonlinear regression. The partial least squares (PLS) approach to generalized inverses. Rep. UMINF-83.80, Univ. Umeå, Umeå, Sweden, 1980.

ALGORITHM

[A part of the listing is printed here. The complete listing is available from the ACM Algorithms Distribution Service (see page 227 for order form).]

```
      SUBROUTINE LSQR( M,N,APROD,DAMP,
     1                 LENIW,LENRW,IW,RW,
     2                 U,V,W,X,SE,
     3                 ATOL,BTOL,CONLIM,ITNLIM,NOUT,
     4                 ISTOP,ANORM,ACOND,RNORM,ARNORM,XNORM )
C
      EXTERNAL    APROD
      INTEGER     M,N,LENIW,LENRW,ITNLIM,NOUT,ISTOP
      INTEGER     IW(LENIW)
      REAL        RW(LENRW),U(M),V(N),W(N),X(N),SE(N),
     1            ATOL,BTOL,CONLIM,DAMP,ANORM,ACOND,RNORM,ARNORM,XNORM
C     -----------------------------------------------------------------
C
C     LSQR  FINDS A SOLUTION  X  TO THE FOLLOWING PROBLEMS...
C
C     1. UNSYMMETRIC EQUATIONS --    SOLVE  A*X = B
C
C     2. LINEAR LEAST SQUARES  --    SOLVE  A*X = B
C                                    IN THE LEAST-SQUARES SENSE
C
C     3. DAMPED LEAST SQUARES  --    SOLVE  (   A    )*X = ( B )
C                                           ( DAMP*I )     ( 0 )
C                                    IN THE LEAST-SQUARES SENSE
C
C     WHERE  A  IS A MATRIX WITH  M  ROWS AND  N  COLUMNS, B  IS AN
C     M-VECTOR, AND  DAMP  IS A SCALAR (ALL QUANTITIES REAL).
C     THE MATRIX  A  IS INTENDED TO BE LARGE AND SPARSE.  IT IS ACCESSED
C     BY MEANS OF SUBROUTINE CALLS OF THE FORM
C
C                 CALL APROD( MODE,M,N,X,Y,LENIW,LENRW,IW,RW )
C
C     WHICH MUST PERFORM THE FOLLOWING FUNCTIONS...
C
C                 IF MODE = 1, COMPUTE  Y = Y + A*X.
C                 IF MODE = 2, COMPUTE  X = X + A(TRANSPOSE)*Y.
C
C     THE VECTORS X AND Y ARE INPUT PARAMETERS IN BOTH CASES.
C     IF MODE = 1, Y SHOULD BE ALTERED WITHOUT CHANGING X.
C     IF MODE = 2, X SHOULD BE ALTERED WITHOUT CHANGING Y.
C     THE PARAMETERS LENIW, LENRW, IW, RW MAY BE USED FOR WORKSPACE
C     AS DESCRIBED BELOW.
C
C     THE RHS VECTOR  B  IS INPUT VIA  U,  AND SUBSEQUENTLY OVERWRITTEN.
C
C
```

```
C     NOTE.  LSQR USES AN ITERATIVE METHOD TO APPROXIMATE THE SOLUTION.
C     THE NUMBER OF ITERATIONS REQUIRED TO REACH A CERTAIN ACCURACY
C     DEPENDS STRONGLY ON THE SCALING OF THE PROBLEM.  POOR SCALING OF
C     THE ROWS OR COLUMNS OF  A  SHOULD THEREFORE BE AVOIDED WHERE
C     POSSIBLE.
C
C     FOR EXAMPLE, IN PROBLEM 1 THE SOLUTION IS UNALTERED BY
C     ROW-SCALING.  IF A ROW OF  A  IS VERY SMALL OR LARGE COMPARED TO
C     THE OTHER ROWS OF  A,  THE CORRESPONDING ROW OF  ( A  B )  SHOULD
C     BE SCALED UP OR DOWN.
C
C     IN PROBLEMS 1 AND 2, THE SOLUTION  X  IS EASILY RECOVERED
C     FOLLOWING COLUMN-SCALING.  IN THE ABSENCE OF BETTER INFORMATION,
C     THE NONZERO COLUMNS OF  A  SHOULD BE SCALED SO THAT THEY ALL HAVE
C     THE SAME EUCLIDEAN NORM (E.G.  1.0).
C
C     IN PROBLEM 3, THERE IS NO FREEDOM TO RE-SCALE IF  DAMP  IS
C     NONZERO.  HOWEVER, THE VALUE OF  DAMP  SHOULD BE ASSIGNED ONLY
C     AFTER ATTENTION HAS BEEN PAID TO THE SCALING OF  A.
C
C     THE PARAMETER  DAMP  IS INTENDED TO HELP REGULARIZE
C     ILL-CONDITIONED SYSTEMS, BY PREVENTING THE TRUE SOLUTION FROM
C     BEING VERY LARGE.  ANOTHER AID TO REGULARIZATION IS PROVIDED BY
C     THE PARAMETER  ACOND,  WHICH MAY BE USED TO TERMINATE ITERATIONS
C     BEFORE THE COMPUTED SOLUTION BECOMES VERY LARGE.
C
C
C     NOTATION
C     --------
C
C     THE FOLLOWING QUANTITIES ARE USED IN DISCUSSING THE SUBROUTINE
C     PARAMETERS...
C
C     ABAR   =  (   A    ),          BBAR  =  ( B )
C               ( DAMP*I )                    ( 0 )
C
C     R      =  B  -  A*X,           RBAR  =  BBAR  -  ABAR*X
C
C     RNORM  =  SQRT( NORM(R)**2  +  DAMP**2 * NORM(X)**2 )
C            =  NORM( RBAR )
C
C     RELPR  =  THE RELATIVE PRECISION OF FLOATING-POINT ARITHMETIC
C               ON THE MACHINE BEING USED.  FOR EXAMPLE, ON THE IBM 370,
C               RELPR  IS ABOUT 1.0E-6 AND 1.0D-16 IN SINGLE AND DOUBLE
C               PRECISION RESPECTIVELY.
C
C     LSQR  MINIMIZES THE FUNCTION  RNORM  WITH RESPECT TO  X.
C
C
C     PARAMETERS
C     ----------
C
C     M       INPUT      THE NUMBER OF ROWS IN  A.
C
C     N       INPUT      THE NUMBER OF COLUMNS IN  A.
C
C     APROD   EXTERNAL   SEE ABOVE.
C
C     DAMP    INPUT      THE DAMPING PARAMETER FOR PROBLEM 3 ABOVE.
C                        (DAMP  SHOULD BE 0.0 FOR PROBLEMS 1 AND 2.)
```

```
C                       IF THE SYSTEM  A*X = B  IS INCOMPATIBLE, VALUES
C                       OF  DAMP  IN THE RANGE Ø TO SQRT(RELPR)*NORM(A)
C                       WILL PROBABLY HAVE A NEGLIGIBLE EFFECT.
C                       LARGER VALUES OF  DAMP  WILL TEND TO DECREASE
C                       THE NORM OF  X  AND TO REDUCE THE NUMBER OF
C                       ITERATIONS REQUIRED BY LSQR.
C
C                       THE WORK PER ITERATION AND THE STORAGE NEEDED
C                       BY LSQR ARE THE SAME FOR ALL VALUES OF  DAMP.
C
C     LENIW   INPUT     THE LENGTH OF THE WORKSPACE ARRAY  IW.
C     LENRW   INPUT     THE LENGTH OF THE WORKSPACE ARRAY  RW.
C     IW      WORKSPACE AN INTEGER ARRAY OF LENGTH  LENIW.
C     RW      WORKSPACE A REAL ARRAY OF LENGTH  LENRW.
C
C             NOTE.  LSQR DOES NOT EXPLICITLY USE THE PREVIOUS FOUR
C             PARAMETERS, BUT PASSES THEM TO SUBROUTINE APROD FOR
C             POSSIBLE USE AS WORKSPACE.  IF APROD DOES NOT NEED
C             IW  OR  RW,  THE VALUES  LENIW = 1  OR  LENRW = 1  SHOULD
C             BE USED, AND THE ACTUAL PARAMETERS CORRESPONDING TO
C             IW  OR  RW  MAY BE ANY CONVENIENT ARRAY OF SUITABLE TYPE.
C
C     U(M)    INPUT     THE RHS VECTOR  B.  BEWARE THAT  U  IS
C                       OVER-WRITTEN BY LSQR.
C
C     V(N)    WORKSPACE
C     W(N)    WORKSPACE
C
C     X(N)    OUTPUT    RETURNS THE COMPUTED SOLUTION  X.
C
C     SE(N)   OUTPUT    RETURNS STANDARD ERROR ESTIMATES FOR THE
C                       COMPONENTS OF  X.  FOR EACH  I,  SE(I)  IS SET
C                       TO THE VALUE  RNORM * SQRT( SIGMA(I,I) / T ),
C                       WHERE  SIGMA(I,I)  IS AN ESTIMATE OF THE I-TH
C                       DIAGONAL OF THE INVERSE OF ABAR(TRANSPOSE)*ABAR
C                       AND  T = 1      IF  M .LE. N,
C                            T = M - N  IF  M .GT. N  AND  DAMP = Ø,
C                            T = M      IF  DAMP .NE. Ø.
C
C     ATOL    INPUT     AN ESTIMATE OF THE RELATIVE ERROR IN THE DATA
C                       DEFINING THE MATRIX  A.  FOR EXAMPLE,
C                       IF  A  IS ACCURATE TO ABOUT 6 DIGITS, SET
C                       ATOL = 1.ØE-6 .
C
C     BTOL    INPUT     AN ESTIMATE OF THE RELATIVE ERROR IN THE DATA
C                       DEFINING THE RHS VECTOR  B.  FOR EXAMPLE,
C                       IF  B  IS ACCURATE TO ABOUT 6 DIGITS, SET
C                       BTOL = 1.ØE-6 .
C
C     CONLIM  INPUT     AN UPPER LIMIT ON  COND(ABAR),  THE APPARENT
C                       CONDITION NUMBER OF THE MATRIX  ABAR.
C                       ITERATIONS WILL BE TERMINATED IF A COMPUTED
C                       ESTIMATE OF  COND(ABAR)  EXCEEDS  CONLIM.
C                       THIS IS INTENDED TO PREVENT CERTAIN SMALL OR
C                       ZERO SINGULAR VALUES OF  A  OR  ABAR  FROM
C                       COMING INTO EFFECT AND CAUSING UNWANTED GROWTH
C                       IN THE COMPUTED SOLUTION.
C
C                       CONLIM  AND  DAMP  MAY BE USED SEPARATELY OR
C                       TOGETHER TO REGULARIZE ILL-CONDITIONED SYSTEMS.
```

```
C
C                       NORMALLY,  CONLIM  SHOULD BE IN THE RANGE
C                       1000  TO  1/RELPR.
C                       SUGGESTED VALUE --
C                       CONLIM = 1/(100*RELPR)  FOR COMPATIBLE SYSTEMS,
C                       CONLIM = 1/(10*SQRT(RELPR))  FOR LEAST SQUARES.
C
C            NOTE.  IF THE USER IS NOT CONCERNED ABOUT THE PARAMETERS
C            ATOL, BTOL  AND  CONLIM,  ANY OR ALL OF THEM MAY BE SET
C            TO ZERO.  THE EFFECT WILL BE THE SAME AS THE VALUES
C            RELPR, RELPR  AND  1/RELPR  RESPECTIVELY.
C
C     ITNLIM  INPUT     AN UPPER LIMIT ON THE NUMBER OF ITERATIONS.
C                       SUGGESTED VALUE --
C                       ITNLIM = N/2     FOR WELL CONDITIONED SYSTEMS,
C                       ITNLIM = 4*N     OTHERWISE.
C
C     NOUT    INPUT     FILE NUMBER FOR PRINTER.  IF POSITIVE,
C                       A SUMMARY WILL BE PRINTED ON FILE  NOUT.
C
C     ISTOP   OUTPUT    AN INTEGER GIVING THE REASON FOR TERMINATION...
C
C               0       X = 0  IS THE EXACT SOLUTION.
C                       NO ITERATIONS WERE PERFORMED.
C
C               1       THE EQUATIONS  A*X = B  ARE PROBABLY
C                       COMPATIBLE.  NORM(A*X - B)  IS SUFFICIENTLY
C                       SMALL, GIVEN THE VALUES OF  ATOL  AND  BTOL.
C
C               2       THE SYSTEM  A*X = B  IS PROBABLY NOT
C                       COMPATIBLE.  A LEAST-SQUARES SOLUTION HAS
C                       BEEN OBTAINED WHICH IS SUFFICIENTLY ACCURATE,
C                       GIVEN THE VALUE OF  ATOL.
C
C               3       AN ESTIMATE OF  COND(ABAR)  HAS EXCEEDED
C                       CONLIM.   THE SYSTEM  A*X = B  APPEARS TO BE
C                       ILL-CONDITIONED.  OTHERWISE, THERE COULD BE AN
C                       AN ERROR IN SUBROUTINE APROD.
C
C               4       THE EQUATIONS  A*X = B  ARE PROBABLY
C                       COMPATIBLE.  NORM(A*X - B)  IS AS SMALL AS
C                       SEEMS REASONABLE ON THIS MACHINE.
C
C               5       THE SYSTEM  A*X = B  IS PROBABLY NOT
C                       COMPATIBLE.  A LEAST-SQUARES SOLUTION HAS
C                       BEEN OBTAINED WHICH IS AS ACCURATE AS SEEMS
C                       REASONABLE ON THIS MACHINE.
C
C               6       COND(ABAR)  SEEMS TO BE SO LARGE THAT THERE IS
C                       NOT MUCH POINT IN DOING FURTHER ITERATIONS,
C                       GIVEN THE PRECISION OF THIS MACHINE.
C                       THERE COULD BE AN ERROR IN SUBROUTINE APROD.
C
C               7       THE ITERATION LIMIT  ITNLIM  WAS REACHED.
C
C     ANORM   OUTPUT    AN ESTIMATE OF THE FROBENIUS NORM OF  ABAR.
C                       THIS IS THE SQUARE-ROOT OF THE SUM OF SQUARES
C                       OF THE ELEMENTS OF  ABAR.
C                       IF  DAMP  IS SMALL AND IF THE COLUMNS OF  A
C                       HAVE ALL BEEN SCALED TO HAVE LENGTH  1.0,
C                       ANORM  SHOULD INCREASE TO ROUGHLY  SQRT(N).
```

```
C                      A RADICALLY DIFFERENT VALUE FOR  ANORM  MAY
C                      INDICATE AN ERROR IN SUBROUTINE APROD (THERE
C                      MAY BE AN INCONSISTENCY BETWEEN MODES 1 AND 2).
C
C     ACOND   OUTPUT   AN ESTIMATE OF  COND(ABAR),  THE CONDITION
C                      NUMBER OF  ABAR.  A VERY HIGH VALUE OF  ACOND
C                      MAY AGAIN INDICATE AN ERROR IN APROD.
C
C     RNORM   OUTPUT   AN ESTIMATE OF THE FINAL VALUE OF NORM(RBAR),
C                      THE FUNCTION BEING MINIMIZED (SEE NOTATION
C                      ABOVE).  THIS WILL BE SMALL IF  A*X = B  HAS
C                      A SOLUTION.
C
C     ARNORM  OUTPUT   AN ESTIMATE OF THE FINAL VALUE OF
C                      NORM( ABAR(TRANSPOSE)*RBAR ), THE NORM OF
C                      THE RESIDUAL FOR THE USUAL NORMAL EQUATIONS.
C                      THIS SHOULD BE SMALL IN ALL CASES.  (ARNORM
C                      WILL OFTEN BE SMALLER THAN THE TRUE VALUE
C                      COMPUTED FROM THE OUTPUT VECTOR  X.)
C
C     XNORM   OUTPUT   AN ESTIMATE OF THE NORM OF THE FINAL
C                      SOLUTION VECTOR  X.
C
C
C     SUBROUTINES AND FUNCTIONS USED
C     ------------------------------
C
C     USER       APROD
C     LSQR       NORMLZ
C     BLAS       SCOPY,SNRM2,SSCAL  (SEE LAWSON ET AL. BELOW)
C                (SNRM2 IS USED ONLY IN NORMLZ)
C     FORTRAN    ABS,MOD,SQRT
C
C
C     PRECISION
C     ---------
C
C     THE NUMBER OF ITERATIONS REQUIRED BY LSQR WILL USUALLY DECREASE
C     IF THE COMPUTATION IS PERFORMED IN HIGHER PRECISION.  TO CONVERT
C     LSQR  AND  NORMLZ  BETWEEN SINGLE- AND DOUBLE-PRECISION, CHANGE
C     THE WORDS
C                SCOPY, SNRM2, SSCAL
C                ABS, REAL, SQRT
C     TO THE APPROPRIATE BLAS AND FORTRAN EQUIVALENTS.
C
C
C     REFERENCES
C     ----------
C
C     PAIGE, C.C. AND SAUNDERS, M.A.  LSQR: AN ALGORITHM FOR SPARSE
C        LINEAR EQUATIONS AND SPARSE LEAST SQUARES.
C        ACM TRANSACTIONS ON MATHEMATICAL SOFTWARE 8, 1 (MARCH 1982).
C
C     LAWSON, C.L., HANSON, R.J., KINCAID, D.R. AND KROGH, F.T.
C        BASIC LINEAR ALGEBRA SUBPROGRAMS FOR FORTRAN USAGE.
C        ACM TRANSACTIONS ON MATHEMATICAL SOFTWARE 5, 3 (SEPT 1979),
C        308-323 AND 324-325.
C
C
C     LSQR.      THIS VERSION DATED 22 FEBRUARY 1982.
C     ------------------------------------------------------------------
```