LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares

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

An iterative method is given for solving $Ax = b$ and $\min\|Ax - b\|_2$, where the matrix A is large and sparse. The method is based on the bidiagonalization procedure of Golub and Kahan. It is analytically equivalent to the standard method of conjugate gradients, but possesses more favorable numerical properties.

Reliable stopping criteria are derived, along with estimates of standard errors for x and the condition number of A. These are used in the FORTRAN implementation of the method, subroutine LSQR. Numerical tests are described comparing LSQR with several other conjugate-gradient algorithms, indicating that LSQR is the most reliable algorithm when A is ill-conditioned.

Categories and Subject Descriptors: G.1.2 **[Numerical Analysis]**: Approximation—*least squares approximation*; G.1.3 **[Numerical Analysis]**: Numerical Linear Algebra—*linear systems (direct and iterative methods); sparse and very large systems*

General Terms: Algorithms

Additional Key Words and Phrases: analysis of variance

The Algorithm: *LSQR*: Sparse Linear Equations and Least Square Problems. *ACM Trans. Math. Softw. 8,* 2 (June 1982).

1. INTRODUCTION

A numerical method is presented here for computing a solution x to either of the following problems:

Unsymmetric equations: solve $Ax = b$

Linear least squares: minimize $\|Ax - b\|_2$

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada Grant A8652, the New Zealand Department of Scientific and Industrial Research, the U.S. Office of Naval Research under Contract N00014-75-C-0267, National Science Foundation Grants MCS 76-20019 and ENG 77-06761, and the Department of Energy under Contract DE-AC03-76SF00326, PA No. DE-AT03-76ER72018.

Authors' addresses: C.C. Paige, School of Computer Science, McGill University, Montreal, P.Q., Canada H3C 3G1; 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 0098-3500/82/0300-0043 $00.75

where A is a real matrix with m rows and n columns and b is a real vector. It will usually be true that $m \geq n$ and $\text{rank}(A) = n$, but these conditions are not essential. The method, to be called algorithm LSQR, is similar in style to the well-known method of conjugate gradients (CG) as applied to the least-squares problem [10]. The matrix A is used only to compute products of the form Av and $A^T u$ for various vectors v and u. Hence A will normally be large and sparse or will be expressible as a product of matrices that are sparse or have special structure. A typical application is to the large least-squares problems arising from analysis of variance (e.g., [12]).

CG-like methods are iterative in nature. They are characterized by their need for only a few vectors of working storage and by their theoretical convergence within at most n iterations (if exact arithmetic could be performed). In practice such methods may require far fewer or far more than n iterations to reach an acceptable approximation to x. The methods are most useful when A is well conditioned and has many nearly equal singular values. These properties occur naturally in many applications. In other cases it is often possible to divide the solution procedure into a direct and an iterative part, such that the iterative part has a better conditioned matrix for which CG-like methods will converge more quickly. Some such transformation methods are considered in [21].

Algorithm LSQR is based on the bidiagonalization procedure of Golub and Kahan [9]. It generates a sequence of approximations $\{x_k\}$ such that the residual norm $\|r_k\|_2$ decreases monotonically, where $r_k = b - Ax_k$. Analytically, the sequence $\{x_k\}$ is identical to the sequence generated by the standard CG algorithm and by several other published algorithms. However, LSQR is shown by example to be numerically more reliable in various circumstances than the other methods considered.

The FORTRAN implementation of LSQR [22] is designed for practical application. It incorporates reliable stopping criteria and provides the user with computed estimates of the following quantities: x, $r = b - Ax$, $A^T r$, $\|r\|_2$, $\|A\|_F$, standard errors for x, and the condition number of A.

1.1 Notation

Matrices are denoted by $A, B, \ldots$, vectors by $v, w, \ldots$, and scalars by $\alpha, \beta, \ldots$. Two exceptions are c and s, which denote the significant components of an elementary orthogonal matrix, such that $c^2 + s^2 = 1$. For a vector v, $\|v\|$ always denotes the Euclidean norm $\|v\|_2 = (v^T v)^{1/2}$. For a matrix A, $\|A\|$ will usually mean the Frobenius norm, $\|A\|_F = (\Sigma a_{ij}^2)^{1/2}$, and the condition number for an unsymmetric matrix A is defined by $\text{cond}(A) = \|A\| \|A^+\|$, where A^+ denotes the pseudoinverse of A. The relative precision of floating-point arithmetic is ϵ, the smallest machine-representable number such that $1 + \epsilon > 1$.

2. MOTIVATION VIA THE LANCZOS PROCESS

In this section we review the symmetric Lanczos process [13] and its use in solving symmetric linear equations $Bx = b$. Algorithm LSQR is then derived by applying the Lanczos process to a particular symmetric system. Although a more direct development is given in Section 4, the present derivation may remain useful for a future error analysis of LSQR, since many of the rounding error properties of the Lanczos process are already known [18].

Given a symmetric matrix B and a starting vector b, the Lanczos process is a method for generating a sequence of vectors $\{v_i\}$ and scalars $\{\alpha_i\}$, $\{\beta_i\}$ such that B is reduced to tridiagonal form. A reliable computational form of the method is the following.

The Lanczos process (reduction to tridiagonal form):

$$\left.\begin{aligned} \beta_1 v_1 &= b, \\ w_i &= Bv_i - \beta_i v_{i-1} \\ \alpha_i &= v_i^{\mathrm{T}} w_i \\ \beta_{i+1} v_{i+1} &= w_i - \alpha_i v_i \end{aligned}\right\}, \qquad i = 1, 2, \ldots, \tag{2.1}$$

where $v_0 \equiv 0$ and each $\beta_i \geq 0$ is chosen so that $\| v_i \| = 1$ $(i > 0)$. The situation after k steps is summarized by

$$BV_k = V_k T_k + \beta_{k+1} v_{k+1} e_k^{\mathrm{T}} \tag{2.2}$$

where $T_k \equiv \text{tridiag}(\beta_i, \alpha_i, \beta_{i+1})$ and $V_k \equiv [v_1, v_2, \ldots, v_k]$. If there were no rounding error we would have $V_k^{\mathrm{T}} V_k = I$, and the process would therefore terminate with $\beta_{k+1} = 0$ for some $k \leq n$. Some other stopping criterion is needed in practice, since β_{k+1} is unlikely to be negligible for any k. In any event, eq. (2.2) holds to within machine precision.

Now suppose we wish to solve the symmetric system $Bx = b$. Multiplying (2.2) by an arbitrary k-vector y_k, whose last element is η_k, gives $BV_k y_k = V_k T_k y_k + \beta_{k+1} v_{k+1} \eta_k$. Since $V_k(\beta_1 e_1) = b$ by definition, it follows that if y_k and x_k are defined by the equations

$$\begin{aligned} T_k y_k &= \beta_1 e_1, \\ x_k &= V_k y_k, \end{aligned} \tag{2.3}$$

then we shall have $Bx_k = b + \eta_k \beta_{k+1} v_{k+1}$ to working accuracy. Hence x_k may be taken as the exact solution to a perturbed system and will solve the original system whenever $\eta_k \beta_{k+1}$ is negligibly small.

The above arguments are not complete, but they provide at least some motivation for defining the sequence of vectors $\{x_k\}$ according to (2.3). It is now possible to derive several iterative algorithms for solving $Bx = b$, each characterized by the manner in which y_k is eliminated from (2.3) (since it is usually not practical to compute each y_k explicitly). In particular, the method of conjugate gradients is known to be equivalent to using the Cholesky factorization $T_k = L_k D_k L_k^{\mathrm{T}}$ and is reliable when B (and hence T_k) is positive definite, while algorithm SYMMLQ employs the orthogonal factorization $T_k = \bar{L}_k Q_k$ to retain stability for arbitrary symmetric B. (See [20] for further details of these methods.)

2.1 The Least-Squares System

We now turn to a particular system that arises from the "damped least-squares" problem

$$\min \left\| \begin{bmatrix} A \\ \lambda I \end{bmatrix} x - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\|_2, \tag{2.4}$$

where A and b are given data and λ is an arbitrary real scalar. The solution of (2.4) satisfies the symmetric (but indefinite) system

$$\begin{bmatrix} I & A \\ A^T & -\lambda^2 I \end{bmatrix} \begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}, \tag{2.5}$$

where r is the residual vector $b - Ax$. When the Lanczos process is applied to this system, a certain structure appears in relations (2.1)–(2.3). In particular, (2.1) reduces to the procedure defined as Bidiag 1 in Section 3, while (2.3) can be permuted after $2k + 1$ iterations to the form

$$\begin{bmatrix} I & B_k \\ B_k^T & -\lambda^2 I \end{bmatrix} \begin{bmatrix} t_{k+1} \\ y_k \end{bmatrix} = \begin{bmatrix} \beta_1 e_1 \\ 0 \end{bmatrix}, \qquad \begin{bmatrix} r_k \\ x_k \end{bmatrix} = \begin{bmatrix} U_{k+1} & 0 \\ 0 & V_k \end{bmatrix} \begin{bmatrix} t_{k+1} \\ y_k \end{bmatrix}, \tag{2.6}$$

where B_k is $(k + 1) \times k$ and lower bidiagonal. We see that y_k is the solution of another damped least-squares problem,

$$\min \left\| \begin{bmatrix} B_k \\ \lambda I \end{bmatrix} y_k - \begin{bmatrix} \beta_1 e_1 \\ 0 \end{bmatrix} \right\|_2, \tag{2.7}$$

which can be solved reliably using orthogonal transformations. This observation forms the basis of algorithm LSQR.

2.2 A Different Starting Vector

For completeness we note that a second least-squares algorithm can be derived in an analogous way. Defining $s = -Ax$, we can write (2.5) as

$$\begin{bmatrix} I & A \\ A^T & -\lambda^2 I \end{bmatrix} \begin{bmatrix} s \\ x \end{bmatrix} = \begin{bmatrix} 0 \\ -A^T b \end{bmatrix} \tag{2.8}$$

and apply the Lanczos process again, using the same matrix as before but with the new starting vector shown. This time, (2.1) reduces to Bidiag 2 as defined in Section 3, while (2.3) can be permuted to the form

$$\begin{bmatrix} I & R_k \\ R_k^T & -\lambda^2 I \end{bmatrix} \begin{bmatrix} q_k \\ y_k \end{bmatrix} = \begin{bmatrix} 0 \\ -\theta_1 e_1 \end{bmatrix}, \qquad \begin{bmatrix} s_k \\ x_k \end{bmatrix} = \begin{bmatrix} P_k & 0 \\ 0 & V_k \end{bmatrix} \begin{bmatrix} q_k \\ y_k \end{bmatrix}, \tag{2.9}$$

after $2k$ iterations, where R_k is $k \times k$ and upper bidiagonal. (The odd-numbered systems are not useful because T_{2k-1} is singular when $\lambda = 0$.) We see that y_k satisfies the system

$$(R_k^T R_k + \lambda^2 I) y_k = \theta_1 e_1, \tag{2.10}$$

which could easily be solved. However, (2.10) proves to be one form of the normal equations associated with (2.7), and the algorithm it suggests for computing x

therefore has unsatisfactory numerical properties. We clarify this matter in Section 7.4.

2.3 The Role of λ

The quantities generated by the Lanczos process from (2.5) and (2.8) are B_k, U_{k+1}, V_k and R_k, P_k, V_k, respectively. These are all *independent of* λ, which means that they are the same as those generated when $\lambda = 0$. We shall therefore assume from now on that $\lambda = 0$. A given λ can be accommodated when solving (2.7), as shown in [22]. Methods for actually choosing λ are beyond the scope of this paper.

3. THE BIDIAGONALIZATION PROCEDURES

The preceding use of the Lanczos process results in two forms of a bidiagonalization procedure due to Golub and Kahan [9]. We state these forms as procedures Bidiag 1 and 2, and then give some unexpected relationships between them.

Bidiag 1 (starting vector b; reduction to lower bidiagonal form):

$$\left.\begin{aligned} \beta_1 u_1 &= b, \qquad \alpha_1 v_1 = A^T u_1. \\ \beta_{i+1} u_{i+1} &= A v_i - \alpha_i u_i \\ \alpha_{i+1} v_{i+1} &= A^T u_{i+1} - \beta_{i+1} v_i \end{aligned}\right\}, \qquad i = 1, 2, \ldots. \tag{3.1}$$

The scalars $\alpha_i \geq 0$ and $\beta_i \geq 0$ are chosen so that $\|u_i\| = \|v_i\| = 1$. With the definitions

$$U_k \equiv [u_1, u_2, \ldots, u_k], \qquad V_k \equiv [v_1, v_2, \ldots, v_k],$$

$$B_k \equiv \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \ddots & \\ & & \ddots & \alpha_k \\ & & & \beta_{k+1} \end{bmatrix},$$

(where B_k is the rectangular matrix introduced in Section 2), the recurrence relations (3.1) may be rewritten as

$$U_{k+1}(\beta_1 e_1) = b, \tag{3.2}$$

$$A V_k = U_{k+1} B_k, \tag{3.3}$$

$$A^T U_{k+1} = V_k B_k^T + \alpha_{k+1} v_{k+1} e_{k+1}^T. \tag{3.4}$$

If exact arithmetic were used, then we would also have $U_{k+1}^T U_{k+1} = I$ and $V_k^T V_k = I$, but, in any event, the previous equations hold to within machine precision.

Bidiag 2 (starting vector $A^T b$; reduction to upper bidiagonal form):

$$\left.\begin{aligned} \theta_1 v_1 &= A^T b, \qquad \rho_1 p_1 = A v_1. \\ \theta_{i+1} v_{i+1} &= A^T p_i - \rho_i v_i \\ \rho_{i+1} p_{i+1} &= A v_{i+1} - \theta_{i+1} p_i \end{aligned}\right\}, \qquad i = 1, 2, \ldots. \tag{3.5}$$

Again, $\rho_i \geq 0$ and $\theta_i \geq 0$ are chosen so that $\| p_i \| = \| v_i \| = 1$. In this case, if

$$P_k \equiv [p_1, p_2, \ldots, p_k], \qquad V_k \equiv [v_1, v_2, \ldots, v_k],$$

$$R_k \equiv \begin{bmatrix} \rho_1 & \theta_2 & & & & \\ & \rho_2 & \theta_3 & & & \\ & & \ddots & \ddots & & \\ & & & & \rho_{k-1} & \theta_k \\ & & & & & \rho_k \end{bmatrix},$$

we may rewrite (3.5) as

$$V_k(\theta_1 e_1) = A^T b, \tag{3.6}$$

$$AV_k = P_k R_k, \tag{3.7}$$

$$A^T P_k = V_k R_k^T + \theta_{k+1} v_{k+1} e_k^T, \tag{3.8}$$

and with exact arithmetic we would also have $P_k^T P_k = V_k^T V_k = I$.

Bidiag 2 is the procedure originally given by Golub and Kahan (with the particular starting vector $A^T b$). Either procedure may be derived from the other by choosing the appropriate starting vector and interchanging A and A^T.

3.1 Relationship Between the Bidiagonalizations

The principal connection between the two bidiagonalization procedures is that the matrices V_k are the same for each, and that the identity

$$B_k^T B_k = R_k^T R_k \tag{3.9}$$

holds. This follows from the fact that v_1 is the same in both cases, and V_k is mathematically the result of applying k steps of the Lanczos process (2.2) with $B = A^T A$. The rather surprising conclusion is that R_k must be identical to the matrix that would be obtained from the conventional QR factorization of B_k. Thus

$$Q_k B_k = \begin{bmatrix} R_k \\ 0 \end{bmatrix} \tag{3.10}$$

where Q_k is orthogonal. In the presence of rounding errors, these identities will, of course, cease to hold. However, they throw light on the advantages of algorithm LSQR over two earlier methods, LSCG and LSLQ, as discussed in Section 7.4.

The relationship between the orthonormal matrices U_k and P_k can be shown to be

$$U_{k+1} = [U_k u_{k+1}] = \left[P_k \frac{r_k}{\| r_k \|} \right] Q_k \tag{3.11}$$

for some vector r_k. We also have the identities

$$\begin{aligned} \alpha_1^2 + \beta_2^2 &= \rho_1^2, & \alpha_1 \beta_1 &= \theta_1, \\ \alpha_i^2 + \beta_{i+1}^2 &= \rho_i^2 + \theta_i^2, & \alpha_i \beta_i &= \rho_{i-1} \theta_i, \qquad \text{for } i > 1. \end{aligned} \tag{3.12}$$

4. ALGORITHM LSQR

The quantities generated from A and b by Bidiag 1 will now be used to solve the least-squares problem, $\min \| b - Ax \|$.

Let the quantities

$$x_k = V_k y_k, \tag{4.1}$$

$$r_k = b - Ax_k, \tag{4.2}$$

$$t_{k+1} = \beta_1 e_1 - B_k y_k, \tag{4.3}$$

be defined in terms of some vector y_k. It readily follows from (3.2) and (3.3) that the equation

$$r_k = U_{k+1} t_{k+1} \tag{4.4}$$

holds to working accuracy. Since we want $\| r_k \|$ to be small, and since U_{k+1} is bounded and theoretically orthonormal, this immediately suggests choosing y_k to minimize $\| t_{k+1} \|$. Hence we are led naturally to the least-squares problem

$$\min \| \beta_1 e_1 - B_k y_k \| \tag{4.5}$$

which forms the basis for LSQR.

Computationally, it is advantageous to solve (4.5) using the standard QR factorization of B_k [8], that is, the same factorization (3.10) that links the two bidiagonalizations. This takes the form

$$Q_k \begin{bmatrix} B_k & \beta_1 e_1 \end{bmatrix} = \begin{bmatrix} R_k & f_k \\ & \bar{\phi}_{k+1} \end{bmatrix} \equiv \left[\begin{array}{ccccccc|c} \rho_1 & \theta_2 & & & & & & \phi_1 \\ & \rho_2 & \theta_3 & & & & & \phi_2 \\ & & \ddots & \ddots & & & & \vdots \\ & & & & \rho_{k-1} & \theta_k & & \phi_{k-1} \\ & & & & & \rho_k & & \phi_k \\ \hline & & & & & & & \bar{\phi}_{k+1} \end{array}\right] \tag{4.6}$$

where $Q_k \equiv Q_{k,k+1} \ldots Q_{2,3} Q_{1,2}$ is a product of plane rotations (e.g., [25]) designed to eliminate the subdiagonals $\beta_2, \beta_3, \ldots$ of B_k. The vectors y_k and t_{k+1} could then be found from

$$R_k y_k = f_k, \tag{4.7}$$

$$t_{k+1} = Q_k^{\mathrm{T}} \begin{bmatrix} 0 \\ \bar{\phi}_{k+1} \end{bmatrix}. \tag{4.8}$$

However, y_k in (4.7) will normally have no elements in common with y_{k-1}. Instead we note that $[R_k \quad f_k]$ is the same as $[R_{k-1} \quad f_{k-1}]$ with a new row and column added. Hence, one way of combining (4.1) and (4.7) efficiently is according to

$$x_k = V_k R_k^{-1} f_k \equiv D_k f_k, \tag{4.9}$$

where the columns of $D_k \equiv [d_1 \quad d_2 \cdots d_k]$ can be found successively from the

system $R_k^T D_k^T = V_k^T$ by forward substitution. With $d_0 = x_0 = 0$, this gives

$$d_k = \frac{1}{\rho_k}(v_k - \theta_k d_{k-1}), \tag{4.10}$$

$$x_k = x_{k-1} + \phi_k d_k, \tag{4.11}$$

and only the most recent iterates need be saved. The broad outline of algorithm LSQR is now complete.

4.1 Recurrence Relations

The QR factorization (4.6) is determined by constructing the kth plane rotation $Q_{k,k+1}$ to operate on rows k and $k+1$ of the transformed $[B_k \quad \beta_1 e_1]$ to annihilate β_{k+1}. This gives the following simple recurrence relation:

$$\begin{bmatrix} c_k & s_k \\ s_k & -c_k \end{bmatrix}\begin{bmatrix} \bar{\rho}_k & 0 & \bar{\phi}_k \\ \beta_{k+1} & \alpha_{k+1} & 0 \end{bmatrix} = \begin{bmatrix} \rho_k & \theta_{k+1} & \phi_k \\ 0 & \bar{\rho}_{k+1} & \bar{\phi}_{k+1} \end{bmatrix}, \tag{4.12}$$

where $\bar{\rho}_1 \equiv \alpha_1$, $\bar{\phi}_1 \equiv \beta_1$, and the scalars c_k and s_k are the nontrivial elements of $Q_{k,k+1}$. The quantities $\bar{\rho}_k$, $\bar{\phi}_k$ are intermediate scalars that are subsequently replaced by ρ_k, ϕ_k.

The rotations $Q_{k,k+1}$ are discarded as soon as they have been used in (4.12), since Q_k itself is not required. We see that negligible work is involved in computing the QR factorization to obtain R_k, f_k, and $\bar{\phi}_{k+1}$.

Some of the work in (4.10) can be eliminated by using vectors $w_k \equiv \rho_k d_k$ in place of d_k. The main steps of LSQR can now be summarized as follows. (As usual the scalars $\alpha_i \geq 0$ and $\beta_i \geq 0$ are chosen to normalize the corresponding vectors; for example, $\alpha_1 v_1 = A^T u_1$ implies the computations $\bar{v}_1 = A^T u_1$, $\alpha_1 = \| \bar{v}_1 \|$, $v_1 = (1/\alpha_1)\bar{v}_1$.)

Algorithm LSQR

(1) (Initialize.)

$\beta_1 u_1 = b$, $\alpha_1 v_1 = A^T u_1$, $w_1 = v_1$, $x_0 = 0$,

$\bar{\phi}_1 = \beta_1$, $\bar{\rho}_1 = \alpha_1$.

(2) For $i = 1, 2, 3, \ldots$ repeat steps 3–6.

(3) (Continue the bidiagonalization.)

(a) $\beta_{i+1} u_{i+1} = A v_i - \alpha_i u_i$

(b) $\alpha_{i+1} v_{i+1} = A^T u_{i+1} - \beta_{i+1} v_i$.

(4) (Construct and apply next orthogonal transformation.)

(a) $\rho_i = (\bar{\rho}_i^2 + \beta_{i+1}^2)^{1/2}$

(b) $c_i = \bar{\rho}_i / \rho_i$

(c) $s_i = \beta_{i+1} / \rho_i$

(d) $\theta_{i+1} = s_i \alpha_{i+1}$

(e) $\bar{\rho}_{i+1} = -c_i \alpha_{i+1}$

(f) $\phi_i = c_i \bar{\phi}_i$

(g) $\bar{\phi}_{i+1} = s_i \bar{\phi}_i$.

(5) (Update x, w.)

(a) $x_t \quad = x_{t-1} + (\phi_t/\rho_t)w_t$

(b) $w_{t+1} = v_{t+1} - (\theta_{t+1}/\rho_t)w_t$.

(6) (Test for convergence.)
Exit if some stopping criteria (yet to be discussed) have been met.

4.2 Historical Note

The previous algorithm was derived by the authors in 1973. An independent derivation is included in the work of Van Heijst et al. [24].

5. ESTIMATION OF NORMS

Here we show that estimates of the quantities $\|r_k\|$, $\|A^Tr_k\|$, $\|x_k\|$, $\|A\|$, and cond(A) can be obtained at minimal cost from items already required by LSQR. All five quantities will be used later to formulate stopping rules.

Knowledge of $\|A\|$ and perhaps cond(A) can also provide useful debugging information. For example, a user must define his matrix A by providing two subroutines to compute products of the form Av and A^Tu. These subroutines will typically use data derived from earlier computations, and may employ rather complex data structures in order to take advantage of the sparsity of A. If the estimates of $\|A\|$ and/or cond(A) prove to be unexpectedly high or low, then at least one of the subroutines is likely to be incorrect. As a rule of thumb, we recommend that all columns of A be scaled to have unit length ($\|Ae_j\| = 1, j = 1, \ldots, n$), since this usually removes some unnecessary ill-conditioning from the problem. Under these circumstances, a programming error should be suspected if the estimate of $\|A\|$ differs by a significant factor from $n^{1/2}$ (since the particular norm estimated will be $\|A\|_F$).

For the purposes of estimating norms, we shall often assume that the orthogonality relations $U_k^TU_k = I$ and $V_k^TV_k = I$ hold, and that $\|U_k\|_2 = \|V_k\|_2 = 1$. In actual computations these are rarely true, but the resulting estimates have proved to be remarkably reliable.

5.1 Estimates of $\|r_k\|$ and $\|A^Tr_k\|$

From (4.4) and (4.8) we have

$$r_k = \bar{\phi}_{k+1}U_{k+1}Q_k^Te_{k+1} \tag{5.1}$$

(which explains the use of r_k in (3.11)); hence, by assuming $U_{k+1}^TU_{k+1} = I$, we obtain the estimate

$$\|r_k\| = \bar{\phi}_{k+1} = \beta_1 s_k s_{k-1} \cdots s_1, \tag{5.2}$$

where the form of $\bar{\phi}_{k+1}$ follows from (4.12). LSQR is unusual in not having the residual vector r_k explicitly present, but we see that $\|r_k\|$ is available essentially free. Clearly the product of sines in (5.2) decreases monotonically. It should converge to zero if the system $Ax = b$ is compatible. Otherwise it will converge to a positive finite limit.

For least-squares problems a more important quantity is A^Tr_k, which would be zero at the final iteration if exact arithmetic were performed. From (5.1), (3.4),

and (4.6) we have

$$A^{\mathrm{T}}r_k = \bar{\phi}_{k+1}(V_k B_k^{\mathrm{T}} + \alpha_{k+1}v_{k+1}e_{k+1}^{\mathrm{T}})Q_k^{\mathrm{T}}e_{k+1}$$
$$= \bar{\phi}_{k+1}V_k[R_k^{\mathrm{T}} \quad 0]e_{k+1} + \bar{\phi}_{k+1}\alpha_{k+1}(e_{k+1}^{\mathrm{T}}Q_k^{\mathrm{T}}e_{k+1})v_{k+1}.$$

The first term vanishes, and it is easily seen that the $(k+1)$st diagonal of Q_k is $-c_k$. Hence we have

$$A^{\mathrm{T}}r_k = -(\bar{\phi}_{k+1}\alpha_{k+1}c_k)v_{k+1} \tag{5.3}$$

and

$$\|A^{\mathrm{T}}r_k\| = \bar{\phi}_{k+1}\alpha_{k+1}|c_k| \tag{5.4}$$

to working accuracy. No orthogonality assumptions are needed here.

5.2 An Estimate of $\|x_k\|$

The upper bidiagonal matrix R_k may be reduced to lower bidiagonal form by the orthogonal factorization

$$R_k\bar{Q}_k^{\mathrm{T}} = \bar{L}_k, \tag{5.5}$$

where $\bar{Q}_k$ is a suitable product of plane rotations. Defining $\bar{z}_k$ by the system

$$\bar{L}_k\bar{z}_k = f_k, \tag{5.6}$$

it follows that $x_k = (V_kR_k^{-1})f_k = (V_k\bar{Q}_k^{\mathrm{T}})\bar{z}_k \equiv \bar{W}_k\bar{z}_k$. Hence, under the assumption that $V_k^{\mathrm{T}}V_k = I$, we can obtain the estimate

$$\|x_k\| = \|\bar{z}_k\|. \tag{5.7}$$

Note that the leading parts of $\bar{L}_k$, $\bar{Q}_k$, $\bar{W}_k$, and $\bar{z}_k$ do not change after iteration k. Hence we find that estimating $\|x_k\|$ via (5.5)–(5.7) costs only 13 multiplications per iteration, which is negligible for large n.

5.3 Estimation of $\|A\|_{\mathrm{F}}$ and cond(A)

It is clear from (3.1) that all the v_i lie in the range of A^{T} and are therefore orthogonal to the null space of A and $A^{\mathrm{T}}A$. With appropriate orthogonality assumptions we have from (3.3) that

$$B_k^{\mathrm{T}}B_k = V_k^{\mathrm{T}}A^{\mathrm{T}}AV_k,$$

and so from the Courant-Fischer minimax theorem the eigenvalues of $B_k^{\mathrm{T}}B_k$ are interlaced by those of $A^{\mathrm{T}}A$ and are bounded above and below by the largest and smallest nonzero eigenvalues at $A^{\mathrm{T}}A$. The same can therefore be said of the singular values of B_k compared with those of A. It follows that for the 2- and F-norms,

$$\|B_k\| \leq \|A\|, \tag{5.8}$$

where equality will be obtained in the 2-norm for some $k \leq \mathrm{rank}(A)$ if b is not orthogonal to the left-hand singular vector of A corresponding to its largest singular value. Equality will only be obtained for the F-norm if b contains components of all left-hand singular vectors of A corresponding to nonzero

singular values. Nevertheless, we will use $\|B_k\|_F$ as a monotonically increasing estimate of the size of A.

The foregoing also implies that $B_k^T B_k = R_k^T R_k$ is nonsingular and for the 2- and F-norms

$$\|R_k^{-1}\| = \|B_k^+\| \leq \|A^+\|. \tag{5.9}$$

The remarks on equality are the same, except now "largest singular value" is replaced by "smallest nonzero singular value."

Combining these results with the definition $D_k = V_k R_k^{-1}$ in (4.9) now gives

$$1 \leq \|B_k\| \, \|D_k\| \leq \|A\| \, \|A^+\| = \text{cond}(A) \tag{5.10}$$

for the 2- and F-norms. Hence we take $\|B_k\|_F \|D_k\|_F$ as a monotonically increasing estimate of cond(A), which starts at the optimistic estimate $\|B_1\|_F \|D_1\|_F = 1$.

Use of Frobenius norms allows the estimates to be accumulated cheaply, since $\|B_k\|_F^2 = \|B_{k-1}\|_F^2 + \alpha_k^2 + \beta_{k+1}^2$ and $\|D_k\|_F^2 = \|D_{k-1}\|_F^2 + \|d_k\|^2$. The individual terms in the sum $\|d_k\|^2 \equiv \sum_{i=1}^{n} \delta_{ik}^2$ can be used further for estimating standard errors, as we show next.

5.4 Standard Errors

In regression problems with $m > n = \text{rank}(A)$, the standard error in the ith component of the true solution x is taken to be s_i where

$$s_i^2 \equiv \frac{\|b - Ax\|^2}{m - n} \sigma_{ii} \tag{5.11}$$

and $\sigma_{ii} \equiv e_i^T (A^T A)^{-1} e_i$ is the ith diagonal element of $(A^T A)^{-1}$. Now from (3.3) and (3.10) we have $V_k^T A^T A V_k = R_k^T R_k$, which with (4.9) gives

$$D_k^T A^T A D_k = I.$$

Assuming that premature termination does not occur, it follows that with exact arithmetic $D_n D_n^T = (A^T A)^{-1}$, and we can approximate the σ_{ii} by $\sigma_{ii}^{(k)} \equiv e_i^T D_k D_k^T e_i$. Since $D_k D_k^T = D_{k-1} D_{k-1}^T + d_k d_k^T$, we have

$$\sigma_{ii}^{(k)} = \sigma_{ii}^{(k-1)} + \delta_{ik}^2, \qquad \sigma_{ii}^{(0)} \equiv 0,$$

and the $\sigma_{ii}^{(k)}$ are monotonically increasing estimates of the σ_{ii}.

In the implementation of LSQR we accumulate $\sigma_{ii}^{(k)}$ for each i, and upon termination at iteration k we set $l = \max(m - n, 1)$ and output the square roots of

$$s_i^{(k)2} \equiv \frac{\|b - Ax_k\|^2}{l} \sigma_{ii}^{(k)}$$

as estimates of the s_i in (5.11). The accuracy of these estimates cannot be guaranteed, especially if termination occurs early for some reason. However, we have obtained one reassuring comparison with the statistical package GLIM [16].

On a moderately ill-conditioned problem of dimensions 171 by 38 (cond(A) $\simeq 10^3$, relative machine precision $\simeq 10^{-11}$), an accurate solution x_k was obtained after 69 iterations, and at this stage all $s_i^{(k)}$ agreed to at least one digit with the s_i output by GLIM, and many components agreed more closely.

A further comparison was obtained from the 1033 by 434 gravity-meter problem discussed in [21]. For this problem a sparse QR factorization was constructed, $QA = \begin{bmatrix} R \\ 0 \end{bmatrix}$, and the quantities σ_{ii} were computed accurately using $R^T v_i = e_i$, $\sigma_{ii} = \| v_i \|^2$. Again, the estimates of $s_i^{(k)}$ from LSQR proved to be accurate to at least one significant figure, and the larger values were accurate to three or more digits.

Note that s_i^2 estimates the variance of the ith component of x, and that $s_i^{(k)^2}$ approximates this variance estimate. In an analogous manner, we could approximate certain covariance estimates by accumulating

$$\sigma_{ij}^{(k)} = \sigma_{ij}^{(k-1)} + \delta_{ik}\delta_{jk}, \qquad \sigma_{ij}^{(0)} \equiv 0,$$

for any specific pairs (i, j), and then computing

$$\frac{\| b - Ax_k \|^2}{l} \sigma_{ij}^{(k)}$$

on termination. This facility has not been implemented in LSQR and we have not investigated the accuracy of such approximations. Clearly only a limited number of pairs (i, j) could be dealt with efficiently on large problems.

6. STOPPING CRITERIA

An iterative algorithm must include rules for deciding whether the current iterate x_k is an acceptable approximation to the true solution x. Here we formulate stopping rules in terms of three dimensionless quantities ATOL, BTOL, and CONLIM, which the user will be required to specify. The first two rules apply to compatible and incompatible systems, respectively. The third rule applies to both. They are

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

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

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

We can implement these rules efficiently using the estimates of $\| r_k \|$, $\| A \|_F$, and so forth, already described.

The criteria S1 and S2 are based on allowable perturbations in the data. The user may therefore set ATOL and BTOL according to the accuracy of the data. For example, if (A, b) are the given data and $(\tilde{A}, \tilde{b})$ represents the (unknown) true values, then

$$\text{ATOL} = \frac{\| A - \tilde{A} \|}{\| A \|}$$

should be used if an estimate of this is available; similarly for BTOL.

Criterion S3 represents an attempt to regularize ill-conditioned systems.

6.1 Compatible Systems

To justify S1, let $r_k = b - Ax_k$ as usual, and define the quantities

$$\delta_k \equiv \text{BTOL}\|b\| + \text{ATOL}\|A\|\,\|x_k\|,$$

$$g_k \equiv \text{BTOL}\|b\|\frac{r_k}{\delta_k},$$

$$h_k \equiv \text{ATOL}\|A\|\,\|x_k\|\frac{r_k}{\delta_k}.$$

Then $r_k = g_k + h_k$, and

$$\left(A + \frac{h_k x_k^T}{x_k^T x_k}\right) x_k = b - g_k$$

so that x_k is the exact solution for a system with both A and b perturbed. It can be seen that these perturbations are within their allowable bounds when the inequality of S1 holds. Hence, criterion S1 is consistent with the ideas of backward rounding error analysis and with knowledge of data accuracy. Since this argument does not depend on orthogonality, S1 can be used in any method for solving compatible linear systems.

6.2 Incompatible Systems

Stewart [23] has observed that if

$$r_k = b - Ax_k$$

and

$$\tilde{r}_k = b - (A + E_k)x_k$$

where

$$E_k = -\frac{r_k r_k^T A}{\|r_k\|^2},$$

then $(A + E_k)^T \tilde{r}_k = 0$, so that x_k and $\tilde{r}_k$ are the exact solution and residual for a system with A perturbed. Since $\|E_k\|_2 = \|A^T r_k\|/\|r_k\|$, the perturbation to A will be negligible if the test in S2 is satisfied.

In our particular method it happens that $E_k x_k = 0$, since (5.3) shows that $x_k = V_k y_k$ is theoretically orthogonal to $A^T r_k$. Hence $\tilde{r}_k = r_k$, so both x_k and r_k are exact for the perturbed system. This strengthens the case for using rule S2.

In practice we find that $\|A^T r_k\|/\|r_k\|$ can vary rather dramatically with k, but it does tend to stabilize for large k, and the stability is more apparent for LSQR than for the standard method of conjugate gradients (see $\|A^T r_k\|$ in Figures 3 and 4, Section 8). Criterion S2 is sufficient to ensure that x_k is an acceptable solution to the least-squares problem, but the existence of an easily computable test that is both sufficient and necessary remains an open question [23].

6.3 Ill-Conditioned Systems

Stopping rule S3 is a heuristic based on the following arguments. Suppose that A has singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n > 0$. It has been observed in some problems

that as k increases the estimate $\|B_k\|_F \|D_k\|_F \simeq \text{cond}(A)$ in (5.10) temporarily levels off near some of the values of the ordered sequence σ_1/σ_1, σ_1/σ_2, $\cdots$, σ_1/σ_n, with varying numbers of iterations near each level. This tends to happen when the smaller σ_i are very close together, and therefore suggests criterion S3 as a means of regularizing such problems when they are very ill-conditioned, as in the discretization of ill-posed problems (e.g., [15]).

For example, if the singular values of A were known to be of order 1, 0.9, 10^{-3}, 10^{-6}, 10^{-7}, the effect of the two smallest singular values could probably be suppressed by setting CONLIM = 10^4.

A more direct interpretation of rule S3 can be obtained from the fact that $x_k = D_k f_k$. First, suppose that the singular value decomposition of A is $A = U\Sigma V^T$ where $U^TU = V^TV = VV^T = I$, $\Sigma = \text{diag}(\sigma_1, \sigma_2, \cdots, \sigma_n)$, and let

$$A^{(r)} = U\Sigma^{(r)}V^T$$

be defined by setting $\sigma_{r+1} = \cdots = \sigma_n = 0$. A common method for regularizing the least-squares problem is to compute $x^{(r)} \equiv V\Sigma^{(r)+}U^Tb$ for some $r \leq n$, since it can readily be shown that the size of $x^{(r)}$ is bounded according to

$$\frac{\|A\|_2\|x^{(r)}\|}{\|b\|} \leq \frac{\sigma_1}{\sigma_r} \equiv \text{cond}(A^{(r)}).$$

In the case of LSQR, we have $\|x_k\| \leq \|D_k\|_F \|b\|$, and so if rule S3 has not yet caused termination, we know that $\|B_k\|_F\|x_k\|/\|b\| \leq \|B_k\|_F\|D_k\|_F < \text{CONLIM}$. Since $\|B_k\|_F$ usually increases to order $\|A\|_F$ quite early, we effectively have

$$\frac{\|A\|_F\|x_k\|}{\|b\|} < \text{CONLIM},$$

which is exactly analogous to the bound above.

6.4 Singular Systems

It is sometimes the case that $\text{rank}(A) < n$. Known dependencies can often be eliminated in advance, but others may remain if only through errors in the data or in formulation of the problem.

With conventional (direct) methods it is usually possible to detect rank deficiency and to advise the user that dependencies exist. In the present context it is more difficult to provide such useful information, but we recognize the need for a method that at least does not fail when applied (perhaps unknowingly) to a singular system. In such cases we again suggest the parameter CONLIM as a device for controlling the computation. Our experience with LSQR on singular problems is that convergence to an acceptable solution occurs normally, but if iterations are allowed to continue, the computed x_k will begin to change again and then grow quite rapidly until

$$\frac{\|A\|\|x_k\|}{\|b\|} \simeq \frac{1}{\epsilon} \tag{6.1}$$

(while $\|r_k\|$ remains of reasonable size). The estimate of $\text{cond}(A)$ typically grows large *before* the growth in x_k. A moderate value of CONLIM (say $1/\epsilon^{1/2}$) may therefore cause termination at a useful solution.

In some cases it can be useful to set CONLIM as large as $1/\epsilon$ and allow x_k to diverge. In this context we note that the algorithm SYMMLQ [20] can be applied to singular symmetric systems, and that extreme growth in the resulting $\|x_k\|$ forms an essential part of a practical method for computing eigenvectors of large symmetric matrices [14]. By analogy, in the presence of rounding errors LSQR will usually produce an approximate singular vector of the matrix A. In fact, using (6.1) and $\|r_k\| \leq \|b\|$, we see that the normalized vector $\bar{x}_k \equiv x_k/\|x_k\|$ will usually satisfy

$$\begin{aligned} A\bar{x}_k &= \frac{1}{\|x_k\|}(b - r_k) \\ &\simeq \epsilon \frac{\|A\|}{\|b\|}(b - r_k) \\ &= 0(\epsilon)\|A\| \end{aligned}$$

for large enough k, and hence will lie very nearly in the null space of A. The vector $\bar{x}_k$ may reveal to the user where certain unexpected dependencies exist. Suitable new rows could then be added to A.

7. OTHER METHODS

Several other conjugate-gradient methods are discussed here. All except the first (CGLS) are stated using notation consistent with Sections 3–6 in order to illustrate certain analytic identities.

7.1 CGLS

If the conjugate-gradient method for symmetric positive definite systems is applied naively to the normal equations $A^TAx = A^Tb$, the method does not perform well on ill-conditioned systems. To a large extent this is due to the explicit use of vectors of the form A^TAp_t. An algorithm with better numerical properties is easily derived by a slight algebraic rearrangement, making use of the intermediate vector Ap_t [10]. It is usually stated in notation similar to the following.

Algorithm CGLS

(1) Set $r_0 = b$, $s_0 = A^Tb$, $p_1 = s_0$, $\gamma_0 = \|s_0\|^2$, $x_0 = 0$.

(2) For $t = 1, 2, 3, \ldots$ repeat the following:

(a) $q_t = Ap_t$

(b) $\alpha_t = \gamma_{t-1}/\|q_t\|^2$

(c) $x_t = x_{t-1} + \alpha_t p_t$

(d) $r_t = r_{t-1} - \alpha_t q_t$

(e) $s_t = A^Tr_t$

(f) $\gamma_t = \|s_t\|^2$

(g) $\beta_t = \gamma_t/\gamma_{t-1}$

(h) $p_{t+1} = s_t + \beta_t p_t$.

A practical implementation of the method would also need to compute $\|r_i\|$, $\|x_i\|$, and an estimate of $\|A\|$ in order to use the stopping criteria developed in Section 6. Otherwise the method is clearly simple and economical. Analytically it generates the same points x_i as LSQR. The vectors v_{i+1} and d_i in LSQR are proportional to s_i and p_i, respectively.

Note that q_i and s_i just given can share the same workspace. A FORTRAN implementation of CGLS has been given by Björck and Elfving [3]. This incorporates an acceleration (preconditioning) technique in a way that requires minimal additional storage.

7.2 Craig's Method for $Ax = b$

A very simple method is known for solving compatible systems $Ax = b$. This is Craig's method, as described in [6]. It is derivable from Bidiag 1, as shown by Paige [17], and differs from all other methods discussed here by minimizing the error norm $\|x_k - x\|$ at each step, rather than the residual norm $\|b - Ax_k\| = \|A(x_k - x)\|$. We review the derivation briefly.

If L_k is the first k rows of B_k,

$$L_k = \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_k & \alpha_k \end{bmatrix},$$

then eqs. (3.3)–(3.4) describing Bidiag 1 may be restated as

$$\begin{aligned} AV_k &= U_k L_k + \beta_{k+1} u_{k+1} e_k^T, \\ A^T U_k &= V_k L_k^T. \end{aligned} \tag{7.1}$$

Craig's method is defined by the equations

$$L_k z_k = \beta_1 e_1, \qquad x_k = V_k z_k, \tag{7.2}$$

and we can show from (7.1) that the residual vector satisfies $r_k \equiv b - AV_k z_k = -\zeta_k \beta_{k+1} u_{k+1}$ and hence $U_k^T r_k = 0$. We can therefore expect r_k to vanish (analytically) for some $k \leq n$.

The vectors z_k and x_k are readily computed from

$$\zeta_k = -\frac{\beta_k}{\alpha_k}\zeta_{k-1}, \qquad x_k = x_{k-1} + \zeta_k v_k,$$

where $\zeta_0 \equiv -1$. Since the increments v_k form an orthogonal set, there is no danger of cancellation, and the step lengths ζ_k are bounded by $|\zeta_k| \leq \|z_k\| = \|x_k\| \leq \|x\|$. We can therefore expect the method to possess good numerical properties. This is confirmed by the comparison in Section 8.

7.3 Extension of Craig's Method

A scheme for extending Craig's method to least-squares problems was suggested by Paige in [17]. The vectors in (7.2) were retained and an additional vector of

the form $V_k w_k$ was computed in parallel. On termination, a suitable scalar γ_k was computed and the final solution taken to be

$$x_k = (V_k z_k) - \gamma_k (V_k w_k) \equiv V_k y_k.$$

In the present context this method may be interpreted as a means of solving the least-squares system (2.7), namely,

$$\begin{bmatrix} I & B_k \\ B_k^T & \end{bmatrix} \begin{bmatrix} t_{k+1} \\ y_k \end{bmatrix} = \begin{bmatrix} \beta_1 e_1 \\ 0 \end{bmatrix},$$

using the fact that the underdetermined system $B_k^T t_{k+1} = 0$ has a unique solution apart from a scalar multiple.

In practice we have found that the method is stable only when b lies in the range of A. Further details are given in [21].

7.4 LSCG and LSLQ

A second algorithm for least-squares problems was given by Paige [17]. This is algorithm LSCG, based on Bidiag 2. In the notation of Section 3 this algorithm is defined by the equations

$$R_k^T R_k y_k = \theta_1 e_1, \qquad x_k = V_k y_k \tag{7.3}$$

and implemented in the form $R_k^T f_k = \theta_1 e_1$, $x_k = (V_k R_k^{-1}) f_k$. Given the relations between the two bidiagonalizations, we now recognize that this is analytically equivalent to LSQR, but numerically inferior, since it is effectively solving the least-squares problem $\min \| B_k y_k - \beta_1 e_1 \|$ by using the corresponding normal equations. (The latter are $B_k^T B_k y_k = B_k^T \beta_1 e_1 = \alpha_1 \beta_1 e_1$ and by (3.9) and (3.12) this is equivalent to the first equation in (7.3).)

Algorithm LSLQ [19] is a refinement of LSCG, but again it is based on Bidiag 2 and the normal equations just given, and is therefore inferior to LSQR on ill-conditioned problems. The refinement has been described in Section 5.2, giving $x_k = \bar{W}_k \bar{z}_k$, where $\bar{W}_k$ is theoretically orthonormal, the intention being to avoid any possible cancellation that could occur in accumulating $x_k = D_k f_k \equiv (V_k R_k^{-1}) f_k$. The same refinement can easily be made to LSQR, and it was implemented in an earlier version of the algorithm for the same reason. However, we have not been able to detect any numerical difference between $x_k = \bar{W}_k \bar{z}_k$ and $x_k = D_k f_k$ in the two versions of LSQR, so the fear of cancellation appears to have been unfounded. We have therefore retained the slightly more economical $x_k = D_k f_k$, which also allows cond(A) to be estimated from $\| D_k \|_F$, as already described.

Algorithms LSCG and LSLQ need not be considered further.

7.5 Chen's Algorithm RRLS

Another algorithm based on Bidiag 2 has been described by Chen [4]. This is algorithm RRLS, and it combines Bidiag 2 with the so-called residual-reducing method of Householder [11]. In the notation of Section 3 it may be described as follows. The residual-reducing property is implicit in steps 2(b) and (c).

Algorithm RRLS

(1) Set $r_0 = b$, $\theta_1 v_1 = A^T b$, $w_1 = v_1$, $x_0 = 0$.

(2) For $i = 1, 2, 3, \ldots$ repeat the following:

(a) $\rho_i p_i = Aw_i$

(b) $\lambda_i = p_i^T r_i$

(c) $r_i = r_{i-1} - \lambda_i p_i$

(d) $\theta_{i+1} v_{i+1} = A^T p_i - \rho_i v_i$

(e) $x_i = x_{i-1} + (\lambda_i/\rho_i) w_i$

(f) $w_{i+1} = v_{i+1} - (\theta_{i+1}/\rho_i) w_i,$

where the scalars ρ_i and θ_i are chosen so that $\|p_i\| = \|v_i\| = 1$.

As with CGLS, a practical implementation would also require $\|r_i\|$ and $\|x_i\|$. The square root of the sum $\sum_{i=1}^{k} (\rho_i^2 + \theta_{i+1}^2) = \|R_k\|_F^2 = \|B_k\|_F^2$ could be used to estimate $\|A\|_F$, and $\|A^T r_i\|$ could also be estimated cheaply.

Note that the vectors v_i are generated as in Bidiag 2, but the vectors p_i come instead from step 2(a). Substituting the latter into step 2(d) shows that RRLS requires explicit computation of the vectors $A^T A w_i$ (ignoring normalization by ρ_i). Unfortunately this must cast doubt on the numerical properties of the method, particularly when applied to compatible systems. Indeed we find that for some systems $Ax = b$, the final norms $\|r_i\|$ and $\|x_i - x\|$ are larger, by a factor approaching cond(A), than those obtained by CGLS and LSQR. This is illustrated in Section 8.3.

A second algorithm called RRLSL has been described by Chen [4], in which the residual-reducing method is combined with Bidiag 1. However, the starting vector used is $AA^T b$ (rather than b), and products of the form $A^T A w_i$ are again required, so that improved performance seems unlikely. Chen reports that RRLS and RRLSL behaved similarly in all test cases tried.

In spite of the above comments, we have also observed ill-conditioned least-squares problems for which RRLS obtains far *greater* accuracy than would normally be expected of any method (see Section 8.4 for a possible explanation). Because of this unusual behavior, we have investigated a residual-reducing version of LSQR as now described.

7.6 RRLSQR

If the residual vector r_i is explicitly introduced, algorithm LSQR as summarized in Section 4.1 can be modified slightly. First, the residual-reducing approach requires step 5(a) to be replaced by the two steps

$$r_i = r_{i-1} - \lambda_i p_i, \qquad x_i = x_{i-1} + \lambda_i w_i,$$

where $p_i = Aw_i$ and $\lambda_i = p_i^T r_{i-1} / \|p_i\|^2$. (In this case p_i is unnormalized.) Second, the product Aw_i can be used to eliminate Av_i from Bidiag 1, leading to an alternative method,

$$\beta_{i+1} u_{i+1} = Aw_i - \frac{\bar{\rho}_i}{\|r_{i-1}\|} r_{i-1}, \tag{7.4}$$

for generating each β_i and u_i. (This result is difficult to derive, but the key relation is $p_i/\|p_i\| = c_i r_{i-1}/\|r_{i-1}\| + s_i u_{i+1}$, which may be deduced from (3.11).)

The remainder of LSQR is retained, including the QR factorization of B_k. The

coefficient of r_{i-1} in (7.4) can be expressed in several ways; for example,

$$\frac{\bar{\rho}_i}{\|r_{i-1}\|} = \frac{\bar{\rho}_i}{\bar{\phi}_i} = \frac{1}{\zeta_i} = (-1)^{i-1}\frac{\alpha_1\alpha_2 \cdots \alpha_i}{\beta_1\beta_2 \cdots \beta_i},$$

where ζ_i comes from the system $L_k z_k = \beta_1 e_1$ of Craig's method. Different formulas lead to different iteration paths, but no variation appears to be consistently better than the rest.

A summary of the resulting algorithm follows.

Algorithm RRLSQR

(1) $r_0 = b$, $\beta_1 u_1 = b$, $\alpha_1 v_1 = A^T u_1$, $w_1 = v_1$, $x_0 = 0$,
$\bar{\phi}_1 = \beta_1$, $\bar{\rho}_1 = \alpha_1$.

(2) For $i = 1, 2, 3, \ldots$ repeat steps 3–6.

(3) (a) $p_i = Aw_i$
(b) $\lambda_i = p_i^T r_{i-1}/\|p_i\|^2$
(c) $r_i = r_{i-1} - \lambda_i p_i$
(d) $\beta_{i+1}u_{i+1} = p_i - (\bar{\rho}_i/\bar{\phi}_i)r_{i-1}$
(e) $\alpha_{i+1}v_{i+1} = A^T u_{i+1} - \beta_{i+1}v_i$

(4) Compute ρ_i, c_i, s_i, θ_{i+1}, $\bar{\rho}_{i+1}$, $\bar{\phi}_{i+1}$ as in Section 4.1, step 4.

(5) (a) $x_i = x_{i-1} + \lambda_i w_i$
(b) $w_{i+1} = v_{i+1} - (\theta_{i+1}/\rho_i)w_i$

(6) Exit if appropriate.

This adaption of Bidiag 1 to obtain RRLSQR is analogous to (and was motivated by) Chen's adaption of Bidiag 2 to obtain RRLS. Note, however, that there are no products of the form $A^T A w_i$. In practice we find that RRLSQR typically performs at least as well as LSQR, as measured by the limiting $\|x_i - x\|$ attainable. Furthermore, it attains the same unusually high accuracy achieved by RRLS on certain ill-conditioned least-squares problems. On these grounds RRLSQR could sometimes be the preferred method. However, its work and storage requirements are significantly higher than for the other methods considered.

7.7 Storage and Work

The storage and work requirements for the most promising algorithms are summarized in Table I. Recall that A is m by n and that for least-squares problems m may be considerably larger than n. Craig's method is applicable only to compatible systems $Ax = b$, which usually means $m = n$.

All methods require the starting vector b. If necessary this may be overwritten by the first m-vector shown (r or u). The m-vector Av shown for Craig and LSQR represents working storage to hold products of the form Av and $A^T u$. (An n-vector would be needed if $m < n$.) In some applications this could be dispensed with if the bidiagonalization operations $Av - \alpha u$ and $A^T u - \beta v$ were implemented to overwrite u and v, respectively. Similarly, the n-vector $A^T p$ for RRLS could in some cases be computed without extra storage.

Table I

	Storage		Work	
	m	n	m	n
Craig ($Ax = b$ only)	u, Av	x, v	3	4
CGLS	r, q	x, p	2	3
LSQR	u, Av	x, v, w	3	5
RRLS	r, p	x, v, w, A^Tp	4	5
RRLSQR	r, u, p	x, v, w	6	5

The work shown for each method is the number of multiplications per iteration. For example, LSQR requires $3m + 5n$ multiplications. (A further $2n$ multiplications are needed to accumulate estimates of $\text{cond}(A)$ and standard errors for x.) Practical implementations of CGLS and RRLS would require a further $m + n$ multiplications to compute $\|r_i\|$ and $\|x_i\|$ for use in stopping rules, although this could be limited to every tenth iteration, say, without serious consequence.

All methods require one product Av and one product A^Tv each iteration. This could dominate the work requirements in some applications.

8. NUMERICAL COMPARISONS

Here we compare LSQR numerically with four of the methods discussed in Section 7, denoted by CRAIG, CGLS, RRLS, and RRLSQR. The machine used was a Burroughs B6700 with relative precision $\epsilon = 0.5 \times 8^{-12} \simeq 0.7 \times 10^{-11}$.

The results given here are complementary to those given by Elfving [5], who compares CGLS with several other conjugate-gradient algorithms and also investigates their performance on problems where A is singular.

8.1 Generation of Test Problems

The following steps may be used to generate a test problem $\min \|b - Ax\|$ with known solution x.

(1) Choose vectors x, y, z, c and diagonal matrix D arbitrarily, with $\|y\| = \|z\| = 1$. (For any chosen $m \geq n$, the vectors should be of dimensions n, m, n, and $m - n$, respectively.)

(2) Define

$$Y = I - 2yy^T, \qquad Z = I - 2zz^T, \qquad A = Y \begin{bmatrix} D \\ 0 \end{bmatrix} Z.$$

(3) Compute

$$r = Y \begin{bmatrix} 0 \\ c \end{bmatrix}, \qquad b = Ax + r.$$

The minimal residual norm is then $\|r\| = \|c\|$. Since A and D have the same singular values, the condition of the problem is easily specified.

The particular problems used here will be called

$$P(m, n, d, p)$$

Table II

Case ($m = n = 10$, cond$(A) = 10^8$)	$\log_{10}\|x_k - x\|$ $k = 60$	80	100	120
1 $A = YDZ$ (as above)	−0.3	−3.3	−3.3	−3 3
2 $A = YD$	−0.5	−3.9	−3.9	−4.1
3 $A = DZ$	−2.1	−5.9	−5.9	−9.2
4 $A = D$	−9.4	−9.4	−9.4	−9.4

to indicate dependence on four integer parameters, where d represents duplication of singular values and p is a power factor. The matrix D is of the form diag(σ_i^p), with each σ_i duplicated d times. Specific values for x, y, z, c, and D were generated as follows:

(1) $x = (n - 1, n - 2, n - 3, \ldots, 2, 1, 0)^T$.
(2) $y_i = \sin(4\pi i/m)$, $z_i = \cos(4\pi i/n)$, followed by normalization so that $\|y\| = \|z\| = 1$.
(3) $c = (1/m, -2/m, 3/m, \ldots, \pm(m - n)/m)^T$.
(4) $\sigma_i = [(i - 1 + d)/d]d/n$, where integer division is used for the term in square brackets. Choosing $n = qd$ to be a multiple of d leads to d copies of each singular value:

$$(\sigma_i) = \left(\frac{1}{q}, \ldots, \frac{1}{q}, \frac{2}{q}, \ldots, \frac{2}{q}, \ldots, \frac{q}{q}, \ldots, \frac{q}{q}\right).$$

[For reference, this gives $\|x\| \simeq n(n/3)^{1/2}$, $\|r\| = \|c\|' \simeq [(m - n)/m]\ ((m - n)/3)^{1/2}$, $\|A\|_F = \|D\|_F \simeq (n/3)^{1/2}$, cond$(A)$ = cond$(D) \simeq (\sigma_n/\sigma_1)^p = q^p$.]

The orthogonal matrices Y and Z are intended to reduce the possibility of anomalous numerical behavior. For example, when LSQR was applied to four cases of the problem $P(10, 10, 1, 8)$, the following error norms resulted (Table II). Since each case was a compatible system $Ax = b$, we normally would expect an error norm approaching $\|x\| \cdot \text{cond}(A) \cdot \epsilon \simeq 10^{-2}$, so that case 1 is the most realistic. In case 2 the error was concentrated in the first and second components of x_k (with the remaining components accurate almost to working precision), whereas in cases 3 and 4 the final x_k was virtually exact in spite of the high condition number of A.

Although cases 2–4 represent less expensive test problems, it is clear that results obtained from them could be very misleading. In the following subsections we use only case 1. Even these test problems may be less than completely general. This possibility is discussed in Section 8.4.

8.2 $Ax = b$: Deterioration of CGLS in Adverse Circumstances

Figure 1 illustrates the performance of five methods on the ill-conditioned system $P(10, 10, 1, 8)$, that is, $m = n = 10$, one copy of each singular value, cond$(A) = 10^8$. The quantities $\log_{10}\|r_k\|$ and $\log_{10}\|x_k - x\|$ are plotted against iteration number k.

This example distinguishes the standard conjugate-gradient method CGLS from the remaining methods. All except CGLS reduced $\|r_k\|$ and $\|x_k - x\|$ to a satisfactory level before $k = 120$.

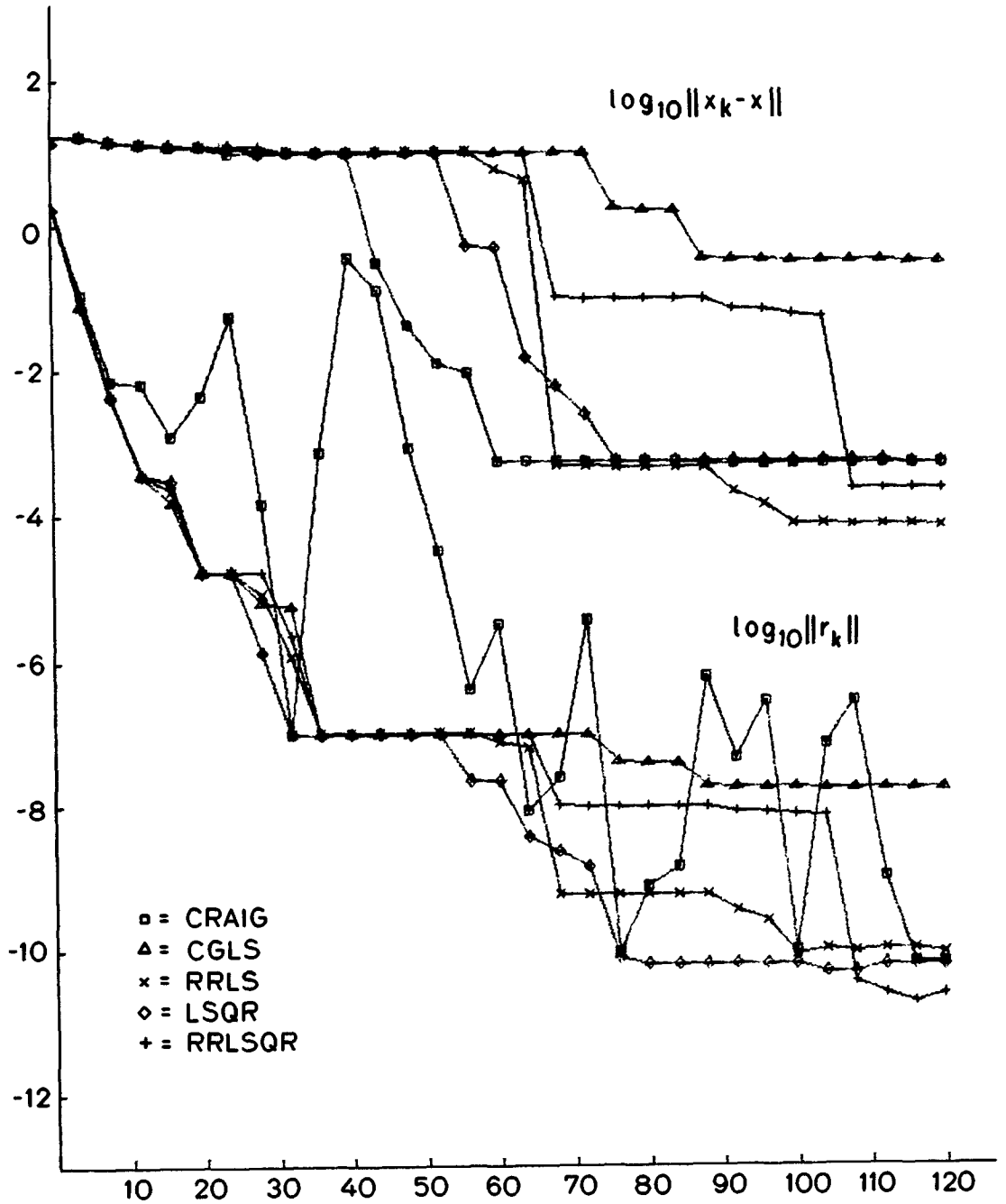

Fig. 1. All ill-conditioned systems $Ax = b$, $n = 10$, cond$(A) = 10^8$. CGLS is unable to reduce $\|r_k\|$ or $\|x_k - x\|$ satisfactorily CRAIG exhibits severe fluctuations in $\|r_k\|$

Also apparent is the erratic behavior of $\| r_k \|$ for method CRAIG, a potential penalty for minimizing $\| x_k - x \|$ at each step without regard to $\| r_k \|$. In theory all other methods minimize $\| r_k \|$ at each step and also reduce $\| x_k - x \|$ monotonically.

If any method is to be preferred in this example it would be LSQR, since it reached limiting accuracy at iteration 76 and stayed at essentially the same point thereafter. With values ATOL = BTOL = ϵ, the stopping rule S1 as implemented in LSQR would have caused termination at $k = 76$ as desired.

8.3 $Ax = b$: Failure of RRLS

Figure 2 illustrates the same five methods applied to a larger problem $P(40, 40, 4, 7)$, in which each singular value is repeated four times and $\text{cond}(A) = 10^7$. In this case all methods except RRLS reduced $\| r_k \|$ satisfactorily to about 10^{-9} for $k \geq 105$. For method RRLS, $\| r_k \|$ remained of order 10^{-5} for $k \geq 30$ up to $k = 250$, and zero digits of accuracy were obtained in x_k.

A similar disparity between RRLS and the remaining methods was observed on the problems $P(40, 40, 4, p)$, $p = 5, 6$, $\text{cond}(A) = 10^p$. In fairness, Chen [4] did not intend RRLS to be applied to compatible systems. However, the success of the other least-squares methods suggests that this is not an unreasonable demand.

8.4 $\min \| Ax - b \|$: High Accuracy by RRLS and RRLSQR

Figure 3 shows the performance of four least-squares methods on the ill-conditioned problem $P(20, 10, 1, 6)$. Since $\text{cond}(A)^2 = 10^{12} \simeq 1/\epsilon$, we would normally expect at most one digit of accuracy in the final x_k. This is achieved by LSQR and CGLS, with LSQR showing a smoother decrease of $\| A^T r_k \|$.

In contrast, the residual-reducing methods achieved at least six digits of accuracy in x_k. Similarly, three or four digits of accuracy were obtained on the problem $P(20, 10, 1, 8)$, for which $\text{cond}(A) = 10^8$ is so high that no digits could be expected. At first sight it may appear that the residual-reducing methods possess some advantage on least-squares problems. However, this anomalous behavior cannot be guaranteed; for example, it did not occur on $P(80, 40, 4, 6)$, as shown in Figure 4. Also, the final value of $\| A^T r_k \|$ is no smaller than for LSQR and this is really the more important quantity.

Part of the explanation for these occasional anomalies may lie in the following. Suppose the original data (A, b) have solution and residual $(\hat{x}, \hat{r})$, while perturbed data $(A + \delta A, b + \delta b)$ have $(\hat{x} + \delta x, \hat{r} + \delta r)$. If $A + \delta A$ has full column rank, then it is straightforward to show that

$$\delta x = (A + \delta A)^+ (\delta b - \delta A \hat{x}) + ((A + \delta A)^T (A + \delta A))^{-1} \delta A^T \hat{r}.$$

In the present example $\epsilon \simeq 0.7 \times 10^{-11}$, $\| A \|_2 = 1$, $\text{cond}(A) = 10^6$, $\| b \| \simeq 2.4$, $\| \hat{x} \| \simeq 17$, $\| \hat{r} \| \simeq 1$. If the perturbations were caused by rounding errors in the initial data, then $\| \delta A \| \simeq \epsilon$, $\| \delta b \| \simeq \epsilon$, and the first term in the expression for δx could be about as large as 10^{-4} in norm, and the second could be of order 7. Figure 3 suggests the second term is unusually small for the RRLS and RRLSQR computations. Looking at the particular form of the test problem, if we write

$$Y \equiv [Y_1, Y_2], \qquad A = Y_1 D Z, \qquad \hat{r} = Y_2 c,$$

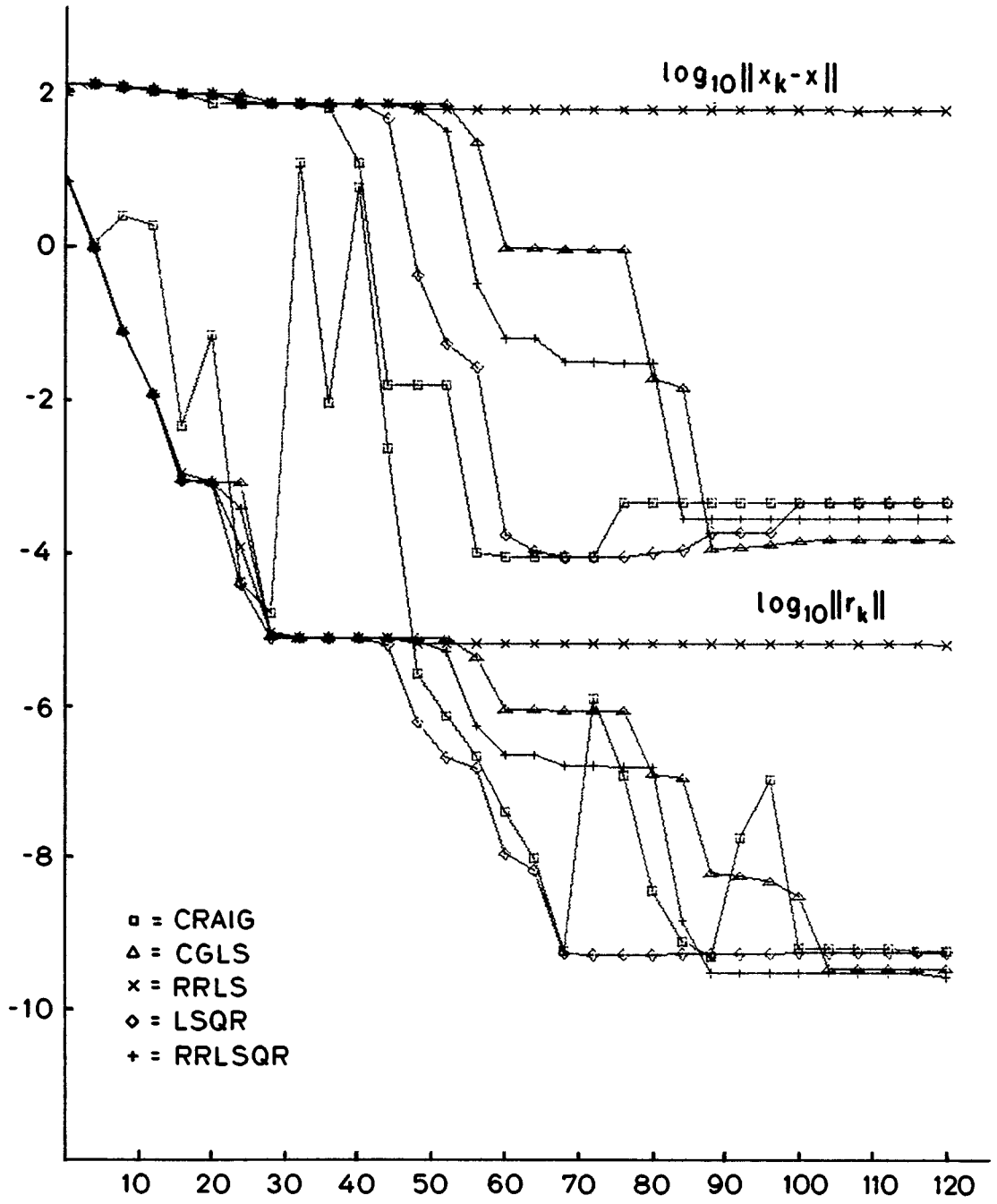

Fig. 2. An ill-conditioned system $Ax = b$, $n = 40$, cond$(A) = 10^7$. RRLS is unable to reduce $\| r_k \|$ or $\| x_k - x \|$ satisfactorily.

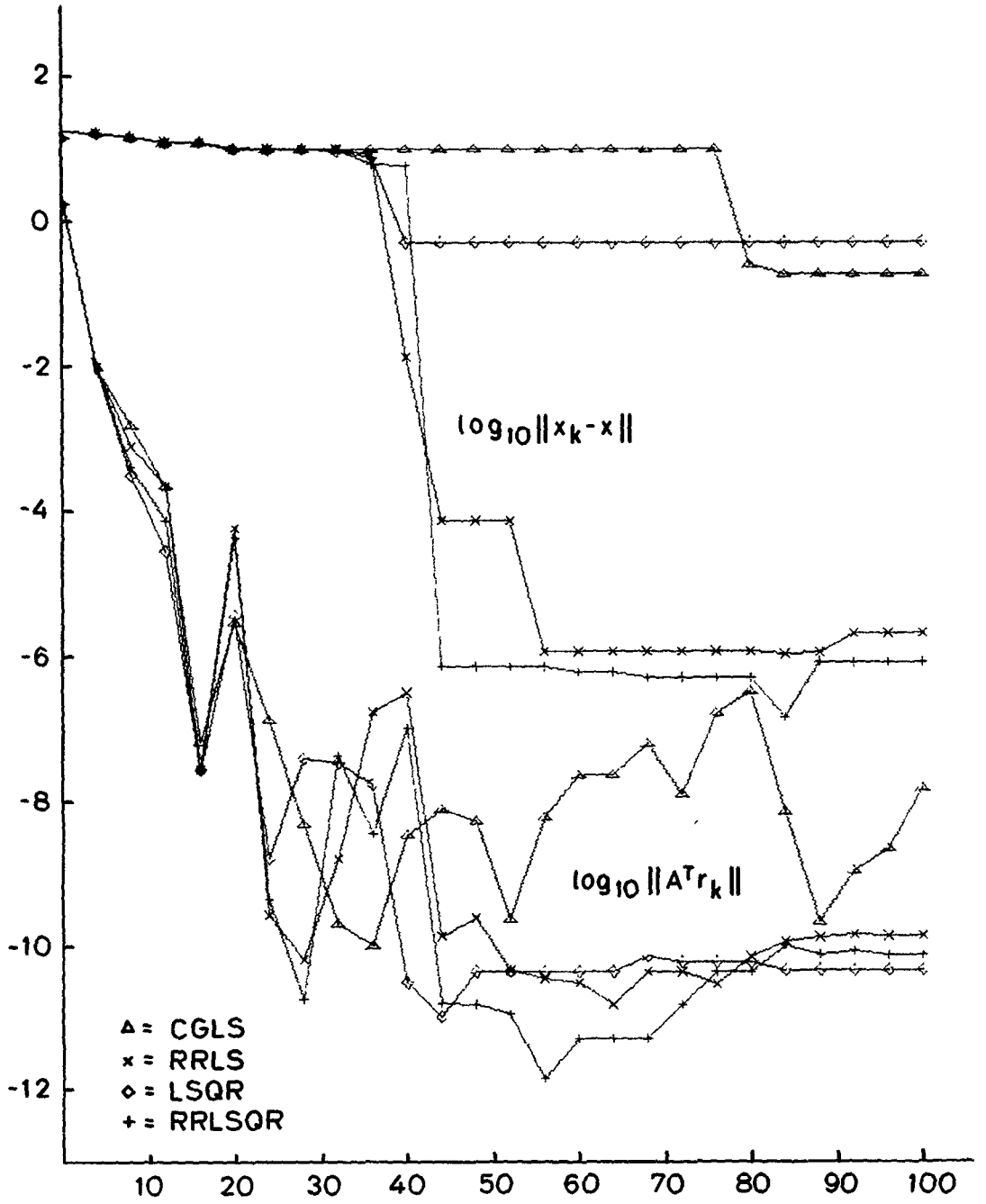

Fig. 3. An ill-conditioned problem, $\min\|Ax - b\|$, $m = 20$, $n = 10$, $\text{cond}(A) = 10^6$. RRLS and RRLSQR achieve anomalously high accuracy in x_k.

we have

$$(A + \delta A)^+ \simeq A^+ = Z^T D^{-1} Y_1^T,$$

and the second term in δx is effectively

$$Z^T D^{-2} Z \delta A^T Y_2 c.$$

Now suppose δA is simply an equivalent perturbation in A caused when A multiplies a vector v in our test case. Using the results of rounding error analyses given by Wilkinson [25],

$$(A + \delta A)v \equiv \mathrm{fl}(Y_1 \mathrm{fl}(D\mathrm{fl}(Zv))) = (Y_1 + \delta Y_1)(D + \delta D)(Z + \delta Z)v,$$

where $\| \delta Y_1 \| \simeq \epsilon \| Y_1 \|$, and so forth; hence

$$\delta A \equiv \delta Y_1 (D + \delta D)(Z + \delta Z) + Y_1 (D\delta Z + \delta D(Z + \delta Z)).$$

Using this δA in the second term for δx effectively gives

$$Z^T D^{-1}(I + D^{-1} Z \delta Z^T D)(I + D^{-1}\delta D)\delta Y_1^T Y_2 c,$$

which is bounded above by about 7×10^{-6} in norm, rather than 7 as expected. This gives a hint of what might be happening above, since a more realistic problem would not admit such a relation between rounding errors and residual. This does not invalidate the other numerical comparisons, but it does emphasize the care needed when constructing artificial test problems.

8.5 $\min \| Ax - b \|$: Normal Behavior

Figure 4 illustrates more typical performance of the four methods, using the least-squares problem $P(80, 40, 4, 6)$ for which $\mathrm{cond}(A) = 10^6$. All methods reduced $\| A^T r_k \|$ to a satisfactory level, and the final error norm is consistent with a conventional sensitivity analysis of the least-squares problem; in this case, no more than one significant digit can be expected. Note that CGLS converged more slowly than the other methods. It also displayed rather undesirable fluctuations in $\| A^T r_k \|$ considerably beyond the point at which the other methods reached limiting accuracy.

8.6 Some Results Using Greater Precision

Algorithm LSQR was also applied to the previous four test problems on an IBM 370 computer using double-precision arithmetic, for which $\epsilon \simeq 2.2 \times 10^{-16}$. With increased precision, LSQR gave higher accuracy and also required fewer steps to attain this accuracy. This is best seen by referring to the figures. In Figure 1 the log of the residual reached -14.4 at the forty-eighth step and stayed there; the log of the error was then -8.6, but decreased 20 steps later to -9.3 and stayed there. In Figure 2 the logs of the residual and error were -13.8 and -8.0 at step 44 and differed negligibly from these values thereafter. In Figure 3, $\log_{10} \| A^T r_k \| = -14.6$ and $\log_{10} \| x_k - x \| = -6.0$ at $k = 32$ and thereafter, while in Figure 4, $\log_{10} \| A^T r_k \| = -13.9$ and $\log_{10} \| x_k - x \| = -4.6$ at $k = 36$, with little change for larger k.

8.7 Other Results

Algorithms CGLS and LSQR have been compared independently by Björck [1], confirming that on both compatible and incompatible systems LSQR is likely to

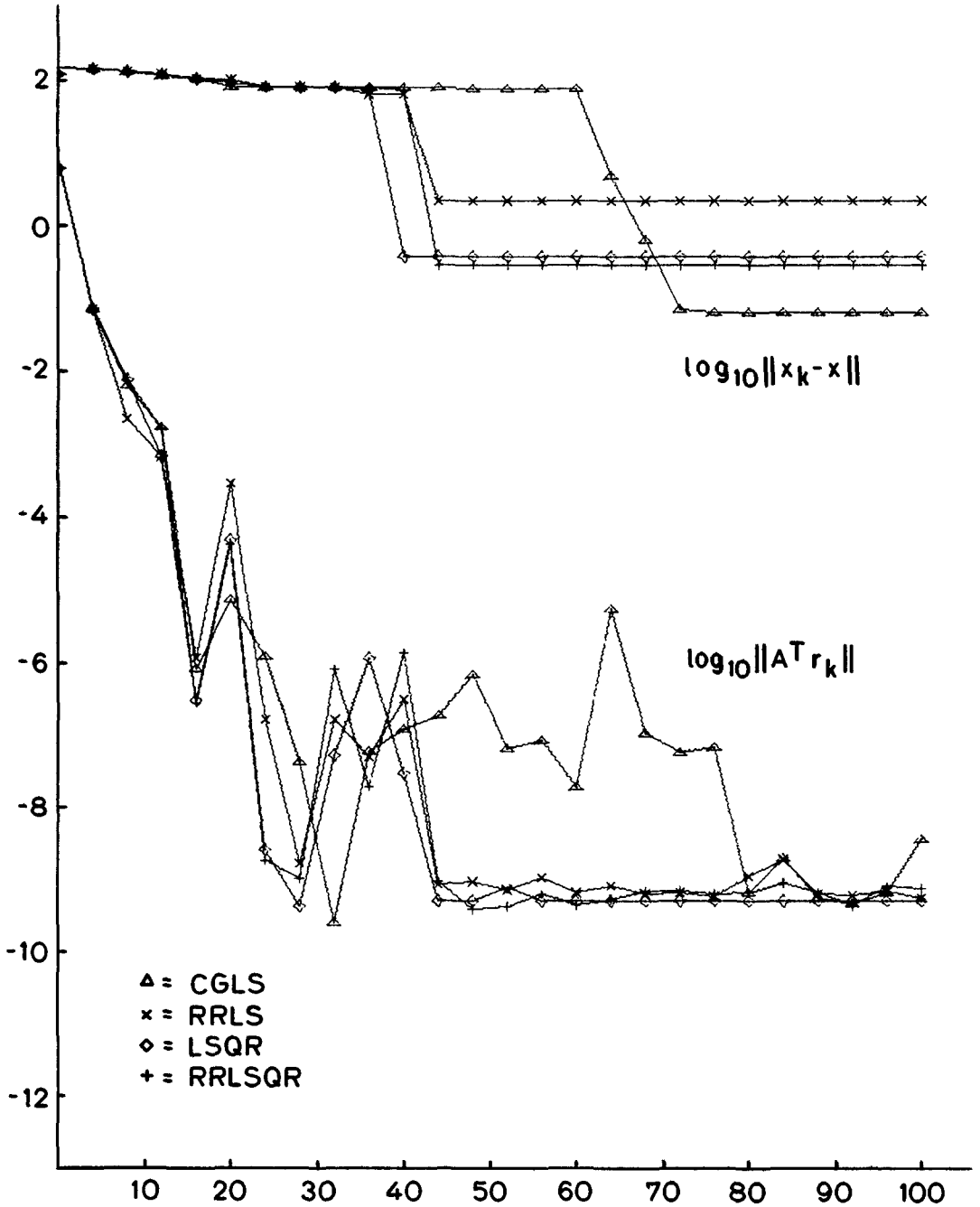

Fig. 4. An ill-conditioned problem, $\min\|Ax - b\|$, $m = 80$, $n = 40$, $\text{cond}(A) = 10^6$. All methods obtain a satisfactory solution, although CGLS exhibits slower convergence and undesirable fluctuations in $\|A^T r_k\|$.

obtain more accurate solutions in fewer iterations. The difference is not significant when A is reasonably well conditioned, but in extreme cases CGLS may need up to twice as many iterations to obtain comparable precision (Figures 1-4).

9. SUMMARY

A direct method may often be preferable to the iterative methods discussed here; for instance, the methods given by Björck and Duff [2] and George and Heath [7] show great promise for sparse least squares. Nevertheless, iterative methods will always retain advantages for certain applications. For example, conjugate-gradient methods converge extremely quickly if A is of the form $M - N$ where $M^TM = I$ and N has low rank. They also have low storage requirements (for both code and workspace).

Our aim has been to present the derivation of a new conjugate-gradient algorithm, along with details of its implementation and sufficient experimental evidence to suggest that it compares favorably with other similar methods and that it can be relied upon to give satisfactory numerical solutions to problems of practical importance (see also [21]).

Reliable stopping criteria were regarded here as being essential to any iterative method for solving the problems $Ax = b$ and $\min \| Ax - b \|$. The criteria developed for LSQR may be useful for other solution methods. Estimates of $\| A \|$, $\text{cond}(A)$, and standard errors for x have also been developed to provide useful information to the user at minimal cost.

In closing, we make the following recommendations:

(1) The symmetric conjugate-gradient algorithm should be applied to the normal equations $A^TAx = A^Tb$ only if there is reason to believe that very few iterations will produce a satisfactory estimate of x.
(2) The least-squares adaptation of symmetric CG will always be more reliable, at the expense of slightly more storage and work per iteration. (This is the algorithm of Hestenes and Stiefel, described in Section 7.1 of this paper as algorithm CGLS.) The additional expense is negligible unless $m \gg n$.
(3) If A is at all ill-conditioned, algorithm LSQR should be more reliable than CGLS, again at the expense of more storage and work per iteration.

ACKNOWLEDGMENTS

We wish to thank Dr. Robert Davies of the New Zealand DSIR for his assistance and advice on many aspects of this work. We are also grateful to Dr. Derek Woodward of the DSIR for the feedback obtained during his application of LSQR to the analysis of gravity-meter observations, to Professor Åke Bjorck for his helpful comments on the manuscript, and to Professor Gene Golub and the referee for some additional useful suggestions.

REFERENCES

1. BJORCK, Å. Use of conjugate gradients for solving linear least squares problems. In Duff, I.S. (Ed.), *Conjugate-Gradient Methods and Similar Techniques*, Rep. AERE R-9636, Computer Science and Systems Division, AERE Harwell, England, 1979, 48-71.
2. BJORCK, Å., AND DUFF, I.S. A direct method for the solution of sparse linear least squares problems. *Linear Algebra Appl. 34* (1980), 43-67.

3. BJORCK, Å, AND ELFVING, T. Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations. Res Rep. LiTH-MAT-R-1978-5, Dep. Mathematics, Linkoping Univ., Linkoping, Sweden, 1978.
4. CHEN, Y.T. Iterative methods for linear least-squares problems. Res. Rep. CS-75-04, Dep. of Computer Science, Univ. Waterloo, Waterloo, Ont., Canada, 1975.
5. ELFVING, T. On the conjugate gradient method for solving linear least-squares problems. Res. Rep. LiTH-MAT-R-1978-3, Dep. Mathematics, Linkoping Univ., Linköping, Sweden, 1978.
6. FADDEEV, D.K., AND FADDEEVA, V.N. *Computational Methods of Linear Algebra*, Freeman, London, 1963.
7. GEORGE, A , AND HEATH, M T. Solution of sparse linear least squares problems using Givens rotations. *Linear Algebra Appl. 34* (1980), 69–83.
8. GOLUB, G.H. Numerical methods for solving linear least-squares problems. *Numer. Math. 7* (1965), 206–216.
9. GOLUB, G.H., AND KAHAN, W. Calculating the singular values and pseudoinverse of a matrix. *SIAM J. Numer. Anal. 2* (1965), 205–224.
10. HESTENES, M.R., AND STIEFEL, E. Methods of conjugate gradients for solving linear systems *J. Res. N.B.S. 49* (1952), 409–436.
11. HOUSEHOLDER, A.S. Terminating and non-terminating iterations for solving linear systems. *SIAM J. Appl. Math. 3* (1955), 67–72.
12. KENNEDY, W.J., AND GENTLE, J.E. *Statistical Computing*. Marcel Dekker, Inc., New York and Basel, 1980.
13. LANCZOS, C. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J Res. N.B.S. 45* (1950), 255–282.
14. LEWIS, J.G. Algorithms for sparse matrix eigenvalue problems. Res. Rep. STAN-CS-77-595, Stanford Univ., Stanford, CA, 1977.
15. NASHED, M.Z. Aspects of generalized inverses in analysis and regularization. In Nashed, M.A. (Ed.), *Generalized Inverses and Applications*, Academic Press, New York, 1976, 193–244.
16. NELDER, J.A. *GLIM Manual*. Numerical Algorithms Group, 13 Banbury Road, Oxford, England, 1975
17. PAIGE, C.C. Bidiagonalization of matrices and solution of linear equations. *SIAM J. Numer. Anal. 11* (1974), 197–209.
18. PAIGE, C.C. Error analysis of the Lanczos algorithm for tridiagonalizing a symmetric matrix. *J. Inst. Maths. Appl 18* (1976), 341–349.
19 PAIGE, C.C., AND SAUNDERS, M.A. Solution of sparse indefinite systems of equations and least-squares problems Res. Rep. STAN-CS-73-399, Stanford Univ., Stanford, CA, 1973.
20 PAIGE, C.C., AND SAUNDERS, M.A. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal. 12* (1975), 617–629.
21 PAIGE, C.C., AND SAUNDERS, M.A. A bidiagonalization algorithm for sparse linear equations and least-squares problems. Rep. SOL 78–19, Dep. Operations Research, Stanford Univ., Stanford, CA, 1978.
22. PAIGE, C.C., AND SAUNDERS, M.A. LSQR· Sparse linear equations and least-squares problems. *ACM Trans. Math. Softw.*, to appear.
23. STEWART, G.W. Research, development and LINPACK. In Rice, J.R. (Ed.), *Mathematical Software III*, Academic Press, New York, 1977, pp. 1–14.
24. VAN HEIJST, J., JACOBS, J., AND SCHERDERS, J. Kleinste-kwadraten problemen. Dep. Mathematics Rep., Eindhoven University of Technology, Eindhoven, The Netherlands, August 1976
25. WILKINSON, J.H. *The Algebraic Eigenvalue Problem*. Oxford University Press (Clarendon), New York, 1965.

Received June 1980; revised September 1981, accepted November 1981