SIAM J. MATRIX ANAL. APPL.
Vol. 19, No. 3, pp. 720–736, July 1998

008

STABILITY OF CONJUGATE GRADIENT AND LANCZOS METHODS FOR LINEAR LEAST SQUARES PROBLEMS*

ÅKE BJÖRCK†, TOMMY ELFVING†, AND ZDENĚK STRAKOŠ‡

Abstract. The conjugate gradient method applied to the normal equations $A^TAx = A^Tb$ (CGLS) is often used for solving large sparse linear least squares problems. The mathematically equivalent algorithm LSQR based on the Lanczos bidiagonalization process is an often recommended alternative. In this paper, the achievable accuracy of different conjugate gradient and Lanczos methods in finite precision is studied. It is shown that an implementation of algorithm CGLS in which the residual $s_k = A^T(b - Ax_k)$ of the normal equations is recurred will not in general achieve accurate solutions. The same conclusion holds for the method based on Lanczos bidiagonalization with starting vector A^Tb. For the preferred implementation of CGLS we bound the error $\|r - r_k\|$ of the computed residual r_k. Numerical tests are given that confirm a conjecture of backward stability. The achievable accuracy of LSQR is shown to be similar. The analysis essentially also covers the preconditioned case.

Key words. conjugate gradient method, least squares, numerical stability

AMS subject classification. 65F20

PII. S089547989631202X

1. Introduction.

Iterative methods are useful alternatives to direct methods for several classes of large sparse least squares problems, see [13, 3]. In this paper we compare different implementations of Krylov subspace methods for solving the linear least squares problem

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

where $A \in \mathbf{R}^{m\times n}$, $m \geq n$, is a given matrix. We will assume that $\operatorname{rank}(A) = n$, although some of the conclusions also hold for $\operatorname{rank}(A) < n$.

It is well known that x is a least squares solution if and only if the residual vector $r = b - Ax \perp \mathcal{R}(A)$, or equivalently when $A^T(b - Ax) = 0$. The resulting system of normal equations

$$A^TAx = A^Tb \tag{1.2}$$

is always consistent. The implementations include conjugate gradient methods and methods based on two different versions of Lanczos bidiagonalization.

The conjugate gradient method (CG), developed in the early 1950s, has become a basic tool for solving large sparse linear systems and linear least squares problems. In the original paper by Hestenes and Stiefel [14], and in the subsequent paper [23], a version of the conjugate gradient method for the solution of the normal equations (1.2) was given. Läuchli [17] discussed a preconditioned conjugate gradient method

*Received by the editors November 13, 1996; accepted for publication (in revised form) by A. Greenbaum May 4, 1997; published electronically March 18, 1998.
http://www.siam.org/journals/simax/19-3/31202.html

†Department of Mathematics, Linköping University, S-581 83, Linköping, Sweden (akbjo@math.liu.se, toelf@math.liu.se). The work of T. Elfving was supported by the Swedish Research Council for Engineering Sciences (TFR) under contract 222-95-401.

‡Institute of Computer Science, Academy of Sciences of the Czech Republic, Pod vodárenskou věží 2, 182 07 Praha 8, Czech Republic (strakos@uivt.cas.cz). The work of this author was supported by AS CR grant A2030706.

for solving least squares geodetic network problems. The application of CG methods to linear least squares problems has also been discussed by Lawson [18] and Chen [5]. Paige [19] derived a method LSCG based on the Lanczos bidiagonalization process of Golub and Kahan in [9]. This method was later shown to be numerically unstable, and a stable version called LSQR was given by Paige and Saunders in [20].

Reid [21] gave an excellent discussion of different computational variants of the conjugate gradient method for solving symmetric positive definite systems. In theory, applying the conjugate gradient method to the normal equations is a straightforward extension of the standard conjugate gradient method. However, the actual implementation is critical and numerically unstable variants are still seen in the literature! Although no comprehensive comparison of different implementations has been published, several conclusions can be found in Elfving [7] and in Paige and Saunders [20]. The aim of this paper is to compare the achievable accuracy in finite precision of different implementations of Krylov subspace methods for solving (1.2). We will make heavy use of recent important results on the behavior of Lanczos and conjugate gradient methods in finite precision, see Greenbaum [10, 11], Greenbaum and Strakoš [12], and Strakoš [24]. The finite precision behavior of stationary iterative methods is studied in [16].

In section 2 we give a survey of Krylov subspace methods for solving the normal equations and their properties in exact arithmetic. In section 3 different implementations of conjugate gradient or Lanczos type methods are considered and their computational complexity compared. The performance of the algorithms in finite precision is discussed in sections 4 and 5. In section 4 it is shown that some implementations of conjugate gradient and Lanczos type methods fail to give accurate solutions. The recommended implementations of conjugate gradient and LSQR are analyzed in section 5. An upper bound for the residual error is derived, which shows these implementations to be backward stable. This extends the analysis of Greenbaum [11] to inconsistent least squares problems. Numerical test results are given in section 6 that confirm the theoretical analysis.

2. Krylov space methods for least squares. When A has full rank the system of normal equations (1.2) has a unique solution

$$x = A^{\dagger}b, \qquad A^{\dagger} = (A^TA)^{-1}A^T.$$

We denote by $r = b - Ax$ the corresponding residual. For a given starting vector x_0 the conjugate gradient algorithm generates approximations x_k in the affine subspace

$$x_k \in x_0 + \mathcal{K}_k(A^TA, s_0), \tag{2.1}$$

$s_0 = A^T(b - Ax_0)$, where

$$\mathcal{K}_k(A^TA, s_0) = \operatorname{span}\left\{s_0, (A^TA)s_0, \ldots, (A^TA)^{k-1}s_0\right\} \tag{2.2}$$

is a Krylov subspace. The iterates generated are optimal in the sense that for each k, x_k minimizes the error functional

$$E_\mu(y) = (x-y)^T(A^TA)^\mu(x-y), \qquad y \in x_0 + \mathcal{K}_k(A^TA, s_0). \tag{2.3}$$

Only the values $\mu = 0, 1, 2$ are of practical interest. By (2.3) and using

$$A(x - x_k) = b - r - Ax_k = r_k - r,$$

where $r_k = b - Ax_k$, we obtain

$$E_\mu(x_k) = \begin{cases} \|x - x_k\|^2, & \mu = 0; \\ \|r - r_k\|^2 = \|r_k\|^2 - \|r\|^2, & \mu = 1; \\ \|A^T(r - r_k)\|^2 = \|A^T r_k\|^2, & \mu = 2. \end{cases} \tag{2.4}$$

Here and in the following, $\|\cdot\|$ denotes the l_2-norm. The second expression for $E_1(x_k)$ in (2.4) follows from the fact that $r \perp r - r_k$.

Using the inner product and norm

$$(x, y)_\mu = x^T (A^T A)^\mu y, \qquad \|x\|_\mu^2 = (x, x)_\mu$$

the conjugate gradient method can be formulated as follows.

ALGORITHM 2.1 (CGLS (μ)).

Let x_0 be an intial approximation, put

$$r_0 = b - Ax_0, \quad s_0 = p_1 = A^T r_0, \quad \gamma_0 = \|s_0\|_{\mu-1}^2, \tag{2.5}$$

and for $k = 1, 2, \ldots$ compute

$$\begin{aligned} q_k &= Ap_k, \\ \alpha_k &= \gamma_{k-1}/\|p_k\|_\mu^2, \\ x_k &= x_{k-1} + \alpha_k p_k, \\ r_k &= r_{k-1} - \alpha_k q_k, \\ s_k &= A^T r_k, \\ \gamma_k &= \|s_k\|_{\mu-1}^2, \\ \beta_k &= \gamma_k/\gamma_{k-1}, \\ p_{k+1} &= s_k + \beta_k p_k. \end{aligned} \tag{2.6}$$

For $\mu = 0$ the method is equivalent to Craig's method [6], which is called CGNE in [8]. Note that $\mu = 0$ is feasible only when $b \in \mathcal{R}(A)$, since we need to be able to evaluate $s_k^T (A^T A)^{-1} s_k$, where $s_k = A^T r_k$ is the current residual of the normal equations. If $b \in \mathcal{R}(A)$, then we have

$$s_k^T (A^T A)^{-1} s_k = (b - Ax_k)^T A (A^T A)^{-1} A^T (b - Ax_k) = r_k^T r_k,$$

where $A(A^T A)^{-1} A^T$ is the orthogonal projection onto $\mathcal{R}(A)$, see also [19].

For $\mu > 0$ we use $\|p_k\|_\mu^2 = \|q_k\|_{\mu-1}^2$. For $\mu = 1$ the method is called CGLS in [20] and CGNR in [8].

The variational property of the conjugate gradient method implies that in exact arithmetic the error functional $E_\mu(x_k)$ decreases monotonically as a function of k. By (2.4), $\|r_k\|$ will also decrease monotonically for $\mu = 1$. Further, we have the following result.

LEMMA 2.1. *Let $\{x_k\}$ be the sequence of conjugate gradient approximations, which minimize $E_\mu(y)$ subject to $y \in x_0 + \mathcal{K}_k(A^T A, s_0)$. Then for $\mu = 1, 2$ the sequences $E_\nu(x_k)$, $0 \le \nu \le \mu$ all decrease monotonically as functions of k.*

Proof. For $\mu = 1$ see [14, p. 416]. For $\mu = 2$ see [14, Section 7]. ∎

It follows that for $\mu = 1$ both $\|r - r_k\|$ and $\|x - x_k\|$ decrease monotonically. However, $\|A^T r_k\|$ will often exhibit large oscillations when $\kappa(A)$ is large. We stress that this behavior is *not* a result of rounding errors. For $\mu = 2$ $\|A^T r_k\|$ will also

decrease monotonically, but this choice gives slower convergence in $\|r - r_k\|$ and $\|x - x_k\|$. It also gives lower final accuracy in these quantities, see [1], and requires more operations or storage. Therefore we consider here only the case $\mu = 1$, which is of most practical interest.

From the optimality property follows the upper bound on the rate of convergence

$$E_\mu(x_k) \le 2\left(\frac{\kappa - 1}{\kappa + 1}\right)^k E_\mu(x_0), \tag{2.7}$$

where $\kappa = \kappa(A) = \sqrt{\kappa(A^T A)}$, see [3, Chapter 7.4]. However, the convergence of the conjugate gradient method also depends on the distribution of the singular values of A. In particular, if A has only $t \le n$ distinct singular values, then in exact arithmetic the solution is obtained in at most t steps.

Often a right preconditioner S is used, which corresponds to the transformation

$$\min_y \|(AS^{-1})y - b\|_2, \qquad Sx = y. \tag{2.8}$$

Typically S is chosen as an approximation of the Cholesky factor R of $A^T A$. If $S = R$, the matrix AS^{-1} is orthogonal and the conjugate gradient method converges in one iteration.

Although (2.7) essentially continues to also hold in finite precision, this is not true of the finite termination property, and conjugate gradient methods should be regarded as iterative methods. In many practical applications, however, one is satisfied with approximations that are obtained in far less than n iterations.

3. Implementation. There are many ways, all mathematically equivalent, in which to implement the conjugate gradient method. In *exact* arithmetic they will all generate the same sequence of approximations, but in finite precision the achieved accuracy may differ substantially. It is important to notice that an implementation of the conjugate gradient method for symmetric positive definite systems should not be applied directly to the normal equations. In particular the explicit formation of the matrix $A^T A$ should be avoided. All the algorithms below require two matrix vector multiplications of the form Ap and $A^T q$ per iteration step.

3.1. The conjugate gradient method CGLS. The algorithm originally given by Hestenes and Stiefel [14, p. 424] and Stiefel [23] is as follows.

ALGORITHM 3.1 (CGLS1).

Set

$$r_0 = b - Ax_0, \quad s_0 = p_1 = A^T r_0, \quad \gamma_0 = \|s_0\|^2, \tag{3.1}$$

and for $k = 1, 2, \ldots$ *compute*

$$\begin{aligned} q_k &= Ap_k, \\ \alpha_k &= \gamma_{k-1}/\|q_k\|^2, \\ x_k &= x_{k-1} + \alpha_k p_k, \\ r_k &= r_{k-1} - \alpha_k q_k, \\ s_k &= A^T r_k, \\ \gamma_k &= \|s_k\|^2, \\ \beta_k &= \gamma_k/\gamma_{k-1}, \\ p_{k+1} &= s_k + \beta_k p_k. \end{aligned} \tag{3.2}$$

Elfving [7] compared CGLS1 with several other implementations of the conjugate gradient method and found this to be the most accurate. CGLS1 requires storage of two n-vectors x, p and two m vectors r, q. (Note that s can share storage with q.) Each iteration requires about $2\mathrm{nz}(A) + 2m + 3n$ multiplications, where $\mathrm{nz}(A)$ is the number of nonzero elements in A.

A small variation of Algorithm CGLS1 is obtained if instead of r_k the residual of the normal equations $s = A^T(b - Ax)$ is recurred.

ALGORITHM 3.2 (CGLS2).

Let x_0 be an initial approximation, set

$$s_0 = p_1 = A^T(b - Ax_0), \quad \gamma_0 = \|s_0\|^2, \tag{3.3}$$

and for $k = 1, 2, \ldots$ compute

$$\begin{aligned} q_k &= Ap_k, \\ \alpha_k &= \gamma_{k-1}/\|q_k\|^2, \\ x_k &= x_{k-1} + \alpha_k p_k, \\ s_k &= s_{k-1} - \alpha_k(A^T q_k), \\ \gamma_k &= \|s_k\|^2, \\ \beta_k &= \gamma_k/\gamma_{k-1}, \\ p_{k+1} &= s_k + \beta_k p_k. \end{aligned} \tag{3.4}$$

CGLS2 requires the storage of three n-vectors x, p, s and one m vector q and $2\mathrm{nz}(A) + 4n + m$ multiplications.

3.2. Methods based on Lanczos bidiagonalization. Paige and Saunders [20] developed algorithms based on the Lanczos bidiagonalization process of Golub and Kahan [9]. There are two forms of this bidiagonalization procedure, Bidiag1 and Bidiag2, that produce two algorithms that differ in their numerical properties.

In Bidiag1 we start the recursion with $\beta_1 u_1 = b - Ax_0$, $\beta_1 = \|b - Ax_0\|$. After k steps we have computed

$$V_k = (v_1, \ldots, v_k), \qquad U_{k+1} = (u_1, \ldots, u_{k+1}),$$

$$B_k = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \ddots & \\ & & \ddots & \alpha_k \\ & & & \beta_{k+1} \end{pmatrix} \in \mathbf{R}^{(k+1)\times k}. \tag{3.5}$$

In exact arithmetic it holds that $AV_k = U_{k+1}B_k$, $V_k^T V_k = I_k$, $U_{k+1}^T U_{k+1} = I_{k+1}$. The columns of V_k form an orthonormal basis for the Krylov subspace $\mathcal{K}_k(A^T A, s_0)$. If we set $x_k - x_0 = V_k y_k$, then $\|b - Ax_k\|$ is minimized when y_k solves the linear least squares problem

$$\min_{y_k} \|B_k y_k - \beta_1 e_1\|.$$

This problem can easily be solved by reducing B_k to upper bidiagonal form by a sequence of Givens rotations. This leads to the LSQR algorithm in [20].

ALGORITHM 3.3 (LSQR (Bidiag1)).
Initialize

$$\beta_1 u_1 = b - Ax_0, \quad \alpha_1 v_1 = A^T u_1, \quad w_1 = v_1,$$
$$\bar{\phi}_1 = \beta_1, \quad \bar{\rho}_1 = \alpha_1,$$

and for $k = 1, 2, \ldots$ *repeat*

$$\begin{aligned}
&\beta_{k+1} u_{k+1} = Av_k - \alpha_k u_k,\\
&\alpha_{k+1} v_{k+1} = A^T u_{k+1} - \beta_{k+1} v_k,\\
&[c_k, s_k, \rho_k] = \text{givrot}(\bar{\rho}_k, \beta_{k+1}),\\
&\theta_{k+1} = s_k \alpha_{k+1}, \quad \bar{\rho}_{k+1} = c_k \alpha_{k+1},\\
&\phi_k = c_k \bar{\phi}_k, \quad \bar{\phi}_{k+1} = -s_k \bar{\phi}_k,\\
&x_k = x_{k-1} + (\phi_k/\rho_k) w_k,\\
&w_{k+1} = v_{k+1} - (\theta_{k+1}/\rho_k) w_k.
\end{aligned}$$

Here the scalars $\alpha_i \geq 0$ and $\beta_i \geq 0$ are chosen to normalize the vectors v_i and u_i, respectively, and $[c, s, \sigma] = \text{givrot}(\alpha, \beta)$ is a subroutine that computes c, s, σ in a Givens rotations such that $-s\alpha + c\beta = 0$ and $\sigma = (\alpha^2 + \beta^2)^{1/2}$.

In addition to the $2\text{nz}(A)$ multiplications required by all versions, LSQR requires $3m+5n$ multiplications and storage of two m-vectors u, Av and three n-vectors x, v, w. A basic relation between LSQR and CGLS1 is that v_i and w_i are proportional to s_{i-1} and p_i. Paige and Saunders showed in [20] by numerical examples that LSQR tends to converge slightly faster than CGLS1 when A is ill-conditioned.

Paige derived in [19] an analytically equivalent algorithm LSCG using a variant of the Lanczos bidiagonalization called Bidiag2. In Bidiag2 the Lanczos recursion is started with $\theta_1 v_1 = A^T(b - Ax_0)$, $\theta_1 = \|A^T(b - Ax_0)\|$. After k steps we have computed

$$V_k = (v_1, \ldots, v_k), \qquad P_k = (p_1, \ldots, p_k),$$

$$R_k = \begin{pmatrix} \rho_1 & \theta_2 & & & \\ & \rho_2 & \theta_3 & & \\ & & \ddots & \ddots & \\ & & & \rho_{k-1} & \theta_k \\ & & & & \rho_k \end{pmatrix} \in \mathbf{R}^{k \times k}, \tag{3.6}$$

and in exact arithmetic it holds that $AV_k = P_k R_k$, $V_k^T V_k = P_k^T P_k = I_k$. Here R_k is already in upper triangular form. Hence $x_k - x_0 = V_k y_k$, where y_k is obtained from $R_k^T R_k y_k = \theta_1 e_1$, and the method is implemented in the form

$$R_k^T f_k = \theta_1 e_1, \qquad x_k = (V_k R_k^{-1}) f_k.$$

ALGORITHM 3.4 (LSCG (Bidiag2)).

$$\theta_1 v_1 = A^T(b - Ax_0), \quad \rho_1 p_1 = Av_1, \quad w_1 = v_1/\rho_1, \quad \zeta_0 = -1,$$

TABLE 3.1
Comparison of storage and operations.

	μ	Storage	mults/step
CGLS1:	1	$2n+2m$	$3n+2m$
CGLS2:	1	$3n+m$	$4n+m$
LSQR:	1	$3n+2m$	$5n+3m$
LSCG:	1	$3n+m$	$5n+3m$

and for $k = 1, 2 \ldots$ *compute*

$$\begin{aligned}
&\zeta_k = -(\theta_k/\rho_k)\zeta_{k-1},\\
&x_k = x_{k-1} + \zeta_k w_k,\\
&\theta_{k+1} v_{k+1} = A^T p_k - \rho_k v_k,\\
&\rho_{k+1} p_{k+1} = A v_{k+1} - \theta_{k+1} p_k,\\
&w_{k+1} = (v_{k+1} - \theta_{k+1} w_k)/\rho_{k+1}.
\end{aligned}$$

Here the matrix $V_k = (v_1, \ldots, v_k)$ is the same as in LSQR. LSCG requires $3m + 5n$ multiplications and storage of one m-vector p and three n-vectors x, v, w.

3.3. Storage and operations. All the Krylov methods described above require two matrix vector multiplications costing $2\mathrm{nz}(A)$ multiplications at each iteration step. In the preconditioned case two linear systems of the form $St = p$ and $S^T s = r$ must also be solved at each step. Storage may be needed for A and S. These costs often dominate the total storage and work.

A comparison of additional storage and operations needed for the methods considered is given in Table 3.1. Here CGLS1 shows an advantage over LSQR. Note however that this may be partly offset by the fact that viable rules for stopping the iterations are more costly for CGLS1 than for LSQR.

For LSQR Paige and Saunders [20] consider several stopping rules which require (estimates of) $\|r_k\|, \|x_k\|, \|s_k\|, \|A\|$, and $\|A^\dagger\|$. They show that all these quantities can be obtained at minimal cost in LSQR. For CGLS1 $\|s_k\|$ is available but $\|r_k\|$ has to be separately computed, if needed, at an extra cost of m multiplications.

4. CGLS2 and LSCG in finite precision.

4.1. Floating point arithmetic. In the following analysis we assume that the standard model for floating point computation holds; i.e., if x and y are floating point numbers, then

$$fl(x \; op \; y) = (x \; op \; y)(1+\delta), \qquad |\delta| \le u,$$

where u is the unit roundoff. This leads to a bound for the roundoff error in the product of two matrices $A \in \mathbf{R}^{m\times n}$ and $B \in \mathbf{R}^{n\times p}$,

$$|fl(AB) - AB| \le \gamma_n |A|\,|B|, \qquad \gamma_n = nu/(1-nu), \tag{4.1}$$

where the inequalities are to be interpreted componentwise, see Higham [15]. It is always assumed when this result is used that $nu \ll 1$.

It is well known, see [3, Section, 1.4.3], that an approximate solution $\tilde{x}$ to (1.2) computed by a normwise backward stable method for solving (1.1) will satisfy an

error bound of the form ($x \neq 0$)

$$\|x - \tilde{x}\| \leq \gamma_m \kappa_{LS}(A, b)\|x\|, \tag{4.2}$$

$$\kappa_{LS}(A, b) = \kappa(A)\left(1 + \frac{\|A^\dagger\|\,\|r\|}{\|x\|}\right). \tag{4.3}$$

Here $\kappa(A) = \sigma_1(A)/\sigma_n(A)$ is the spectral condition number of A, and κ_{LS} is the condition number for computing x. Note that if $\|r\| < \|x\|/\|A^\dagger\|$, then $\kappa_{LS} < 2\kappa(A)$, although in general a term proportional to $\kappa^2(A)$ is present.

A first-order perturbation bound corresponding to componentwise relative errors satisfying $|\delta A| \leq \omega|A|$, $|\delta b| \leq \omega|b|$ is given by (see [2])

$$|x - \tilde{x}| \leq \omega|A^\dagger|(|A|\,|x| + |b|) + \omega|(A^TA)^{-1}|\,|A^T|\,|r| + O(\omega^2). \tag{4.4}$$

4.2. Failure of CGLS2 and LSCG. It was pointed out by Paige and Saunders in [20, Section, 7.1] that the explicit use of vectors of the form $A^T(Ap)$ as in CGLS2 can lead to poor performance on ill-conditioned systems. Below we give a new simple explanation for the failure of both CGLS2 and LSCG to obtain good accuracy.

The two algorithms CGLS2 and LSCG share the following feature. Assuming that $x_0 = 0$, the only information about the right-hand side b is in the initialization of $p_0 = s_0 = A^Tb$ and $\theta_1 v_1 = A^Tb$, respectively. Note that *no reference to b is made in the iterative phase.* It follows that roundoff errors that occur in computing the vector A^Tb cannot be compensated for later. By (4.1) we have

$$|fl(A^Tb) - A^Tb| \leq \gamma_m|A^T|\,|b|, \qquad \gamma_m = mu/(1 - mu), \tag{4.5}$$

and this is almost sharp. The perturbed solution $x + \delta x$ corresponding to $c = fl(A^Tb)$ satisfies

$$A^TA(x + \delta x) = A^Tb + \delta c, \qquad |\delta c| \leq \gamma_m|A^T|\,|b|.$$

Subtracting $A^TAx = A^Tb$ and solving for δx we obtain $\delta x = (A^TA)^{-1}\delta c$, and from this we get the componentwise bound

$$|\delta x| \leq \gamma_m|(A^TA)^{-1}|\,|A^T|\,|b|. \tag{4.6}$$

Hence if $|b| \gg |r|$, which is the case for a nearly consistent system, this error bound is much larger than the second term in the perturbation bound (4.4).

Using norms we obtain

$$\|\delta x\| \leq \gamma_m\|(A^TA)^{-1}\|\,\|A^T\|\,\|b\| = \gamma_m\kappa(A)\|A^\dagger\|\,\|b\|. \tag{4.7}$$

Since b is not used by CGLS2/LSCG it follows that *this initial error cannot be canceled*, and the best error bound we can hope for will include the term given above.

Comparing with the error bound (4.2)–(4.3) we conclude that if

$$\max\left\{\|x\|/\|A^\dagger\|, \|r\|\right\} \ll \|b\|,$$

CGLS2/LSCG can be expected to produce much less than optimal accuracy. Note that since $x = A^\dagger b$ we could theoretically have $\|x\| = \|A^\dagger\|\,\|b\|$. However, this situation very seldomly reflects the properties of least squares solutions. In particular, ill-conditioned problems usually have solutions with much smaller norms: $\|x\| \ll \|A^\dagger\|\,\|b\|$.

Similarly the residual of the perturbed solution may differ from the true residual by as much as

$$\|b - A(x + \delta x) - (b - Ax)\| = \|A\delta x\| \leq \gamma_m \kappa(A)\|b\|.$$

Again, this cannot be canceled in the iterations, and the iterated residual will (in the best case) converge to the perturbed residual. The numerical experiments in section 6 demonstrate this nicely.

The loss of accuracy will occur even if a preconditioner is used. In CGLS2 we then initialize $s_0 = p_1 = S^{-T}(A^T(b - Ax_0))$, where $S^{-T}A^T$ is never explicitly formed. Therefore the same roundoff errors will occur in computing $A^T(b - Ax_0)$, and by the same reasoning as above these error will never be canceled.

We remark that avoiding the loss of information from explicitly multiplying b by A^T is also important when solving discrete ill-posed linear systems. Using a Krylov subspace method, regularization can be achieved by working with the Krylov subspaces span $\{(AA^T)^p b, (AA^T)^{p+1}b, \ldots, (AA^T)^{p+k}b\}$, where p is some positive integer. In [4] an implicit shift restarted Lanczos method (see [22]) is used to implement such a method in a numerically stable way.

5. Error analysis of CGLS1 and LSQR. Greenbaum [11] studies the finite precision implementation of the class of iterative methods in which each step updates the approximate solution x_k and residual r_k using the formulas

$$\begin{aligned} x_{k+1} &= x_k + \alpha_k p_k, \\ r_{k+1} &= r_k - \alpha_k A p_k. \end{aligned} \tag{5.1}$$

Though in LSQR the residual is not recursively updated, we can formally add the update

$$r_k = r_{k-1} - (\phi_k/\rho_k)Aw_k \tag{5.2}$$

to Algorithm 3.3 without making any other changes in it. It follows then that the analysis based on (5.1) can be applied also to the LSQR method. In our experiments we have used LSQR with this additional recursion.

In [11], the matrix A is assumed square nonsingular. It is easy to see that the analysis will also apply, with a simple modification, to the least squares problem with A rectangular and $r = b - Ax$ different from zero. Let x_k, r_k, α_k, and p_k denote from now on the computed values. Then, following [11], we may easily derive the following result (here and in the rest of the paper we assume for simplicity that $x_0 = 0$).

THEOREM 5.1. *The difference between the true residual $b - Ax_k$ and the recursively computed vector r_k satisfies*

$$\begin{aligned} \frac{\|b - Ax_k - r_k\|}{\|A\|\|x\|} \leq u\,[k + 1 + (1 + c + k(10 + 2c))\Theta_k] + u(k + 1)\frac{\|r\|}{\|A\|\|x\|} \\ + O(u^2), \end{aligned} \tag{5.3}$$

where

$$\Theta_k = \max_{j \leq k} \|x_j\|/\|x\|,$$

u is the machine precision, and c depends on the accuracy of the matrix vector multiply (4.5). *If the matrix-vector product is computed in the standard way, then $c = \gamma_m$.*

By using the relation

$$b - Ax_k - r_k = A(x - x_k) + (r - r_k)$$

we obtain the following slight modification of (5.3):

$$\frac{\|A(x-x_k)\|}{\|A\|\|x\|} \le u\,[k+1+(1+c+k(10+2c))\Theta_k] + u(k+1)\frac{\|r\|}{\|A\|\|x\|} + \\ + \frac{\|r-r_k\|}{\|A\|\|x\|} + O(u^2). \tag{5.4}$$

If it can be shown that there is a constant c_1 such that the computed recursive residual r_k satisfies

$$\frac{\|r-r_k\|}{\|A\|\,\|x\|} \le c_1 u + O(u^2), \quad k \ge S, \tag{5.5}$$

then (5.4) gives an upper bound for the accuracy of the ultimately attainable true residual of the computed approximation. Here S denotes the number of iterations needed to reach a steady-state.

Though we offer no formal proof of (5.5) (to our knowledge, no rigorous formal proofs exist even for the practically used variants of conjugate gradient-type methods for solving linear systems), there is overwhelming experimental evidence that justifies our further considerations. In the following we *assume* that (5.5) holds for both CGLS1 and LSQR. From (5.4) and (5.5) we then get

$$\frac{\|A(x-x_S)\|}{\|A\|\|x\|} \le u\,[S+1+c_1+(1+c+S(10+2c))\Theta] + u(S+1)\frac{\|r\|}{\|A\|\|x\|} \\ + O(u^2), \tag{5.6}$$

where $\Theta = \Theta_S$.

We wish to bound the value Θ. If $\|A^\dagger(r_{k-1} - r_k)\|$ reaches the level $O(u)\|x_{k-1}\|$, then by (5.1) the approximate solution x_k remains essentially unchanged (we assume this situation will take place on or before the step S). In exact arithmetic, both CGLS1 and LSQR are equivalent to the conjugate gradient method applied to the normal equations. By Lemma 2.1, the l_2-norm of the error will therefore decrease monotonically in both cases. Using the analogy developed in [10, 12], and arguing similarly as in [11, Section 3.3], one may therefore expect that the relation $\|x - x_k\| \le \|x - x_0\|$, and thus

$$\|x_k\| \le 2\|x\| + \|x_0\|$$

will hold to a close approximation for the computed quantities. Consequently, we obtain the estimate $\Theta \approx 2 + \Theta_0$. Hence, considering $x_0 = 0$,

$$\frac{\|A(x-x_S)\|}{\|A\|\|x\|} \le u\,(3+c_1+21S+2c+4Sc) + u(S+1)\frac{\|r\|}{\|A\|\|x\|} + O(u^2). \tag{5.7}$$

It follows that the ultimately attainable error of the computed approximation x_S for both CGLS1 and LSQR is bounded by

$$\|x - x_S\| \le u\kappa(A)\left[3+c_1+21S+2c+4Sc+(S+1)\frac{\|r\|}{\|A\|\|x\|}\right]\|x\| + O(u^2). \tag{5.8}$$

In the consistent case, (5.7) shows that the residuals corresponding to the computed approximate solution is of the order of the unit roundoff u, and in this sense CGLS1 and LSQR are normwise backward stable. In the inconsistent case, (5.8) shows that CGLS1 and LSQR may compute *more accurate* solutions than a backward stable method. Note, however, that the bound S for the number of iterations needed depends on $\kappa(A)$.

We note that essentially the same formulas hold for the preconditioned case. Then a vector t_k is computed by solving $St_k = p_k$, but this does not alter the form of the recursion formulas (5.1), which become

$$\begin{aligned} x_{k+1} &= x_k + \alpha_k t_k, \\ r_{k+1} &= r_k - \alpha_k A t_k. \end{aligned} \tag{5.9}$$

In general, the preconditioner will accelerate convergence, and hence the steady-state will be reached for a smaller number of steps S.

It should be mentioned that though both CGLS1 and LSQR were shown here to behave in essentially the same way, some differences may occur for very ill-conditioned problems. In fact in our analysis, we use some implicit restrictions on $\kappa(A)$ allowing us to apply the results [10, 11, 12]. Therefore our results do not apply to highly ill-conditioned problems.

6. Numerical results. Numerical tests were performed in MATLAB on a SUN SPARCstation 10 using double precision with unit roundoff $u = 2.2 \cdot 10^{-16}$. The tests are similar to tests run on an IBM 370 using double precision by the first author during a visit to Stanford University in 1979 [1].

The test problems denoted $P(m, n, d, p)$ were taken from Paige and Saunders [20]. The matrix A was constructed by

$$A = Y \begin{pmatrix} D \\ 0 \end{pmatrix} Z^T \in \mathbf{R}^{m \times n}, \quad Y = I - 2yy^T, \quad Z = I - 2zz^T.$$

Here y and z are Householder vectors of appropriate dimension, with elements

$$\begin{aligned} y_i &= \sin(4\pi i/m), \quad i = 1, \ldots, m, \\ z_i &= \cos(4\pi i/n), \quad i = 1, \ldots, n, \end{aligned}$$

followed by normalization so that $\|y\| = \|z\| = 1$. For $n = q$ $(d = 1)$ the singular values are chosen to be

$$D = q^{-p} \mathrm{diag}\,(q^p, \ldots, 3^p, 2^p, 1);$$

i.e., p is a power factor. Taking $n = qd$ leads to d copies of each singular value. The solution is taken to be $x = (n-1, \ldots, 2, 1, 0)^T$, and the right-hand side is constructed as

$$\begin{aligned} b &= Ax + r, \quad r = \rho Y \begin{pmatrix} 0 \\ c \end{pmatrix}, \\ c &= m^{-1}(1, -2, 3, \ldots, \pm(m-n))^T. \end{aligned}$$

Thus if $m > n$ and $\rho > 0$ the system is incompatible and $\|r\| \approx \rho$. (In [20] $\rho = 1$ was used throughout.)

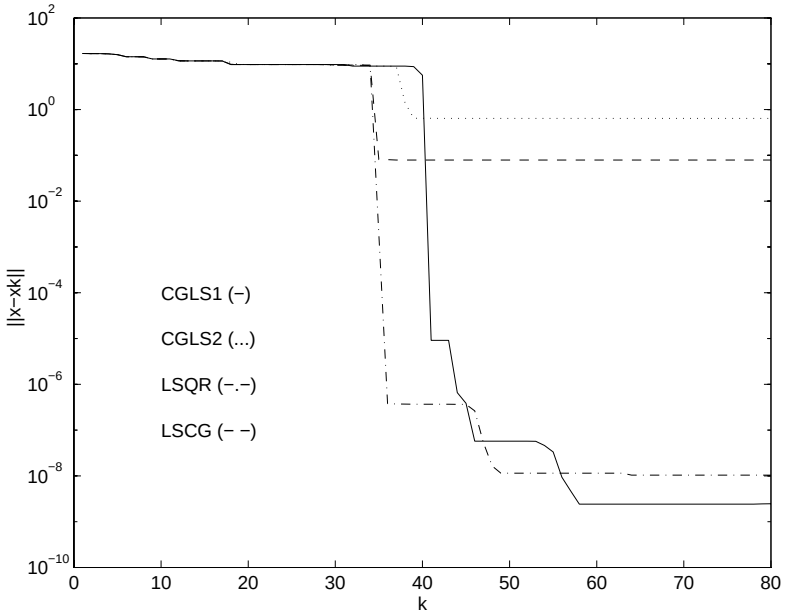

FIG. 6.1. $\|x - x_k\|$ *for problem* $PS(10, 10, 1, 8)$, $\kappa = 10^8$.

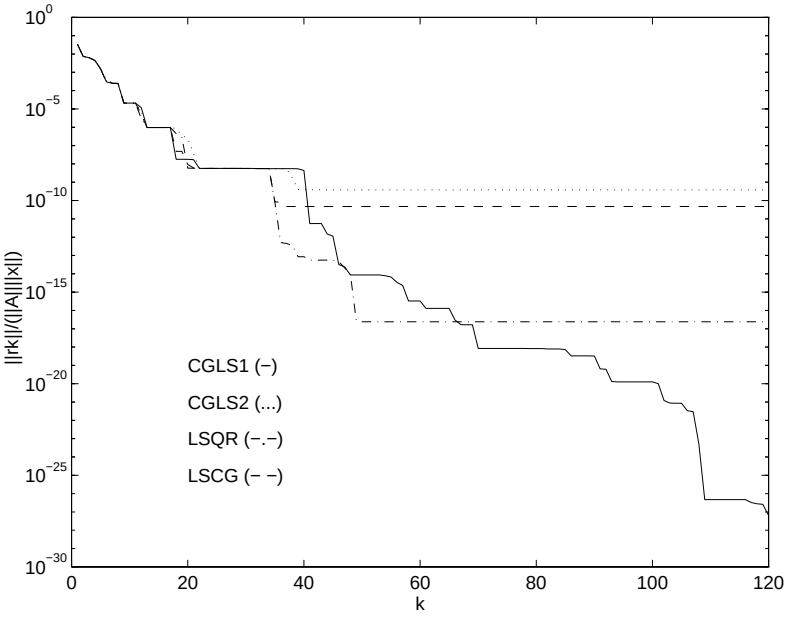

FIG. 6.2. $\|r_k\|/(\|A\|\|x\|)$ *for problem* $PS(10, 10, 1, 8)$, $\kappa = 10^8$.

The first tests were run on the consistent problem $PS(10, 10, 1, 8)$, with $\kappa(A) = 10^8$. Results are shown for CGLS1, CGLS2, and the two Lanczos methods LSQR and LSCG in Figures 6.1–6.3. Here x and r denote the exact solution and the exact residual. Hence $r = 0$ in the consistent case.

Figure 6.1 shows that CGLS1 and LSQR both achieve a relative accuracy of about 10^{-9}, whereas the error in the unstable versions CGLS2 and LSCG are about a factor

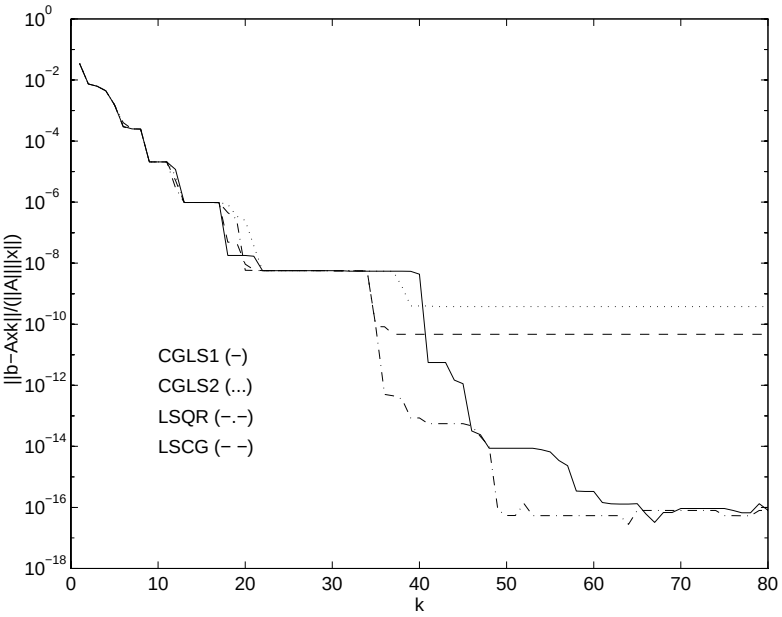

FIG. 6.3. $\|b - Ax_k\|/(\|A\|\|x\|)$ *for problem* $PS(10, 10, 1, 8)$, $\kappa = 10^8$.

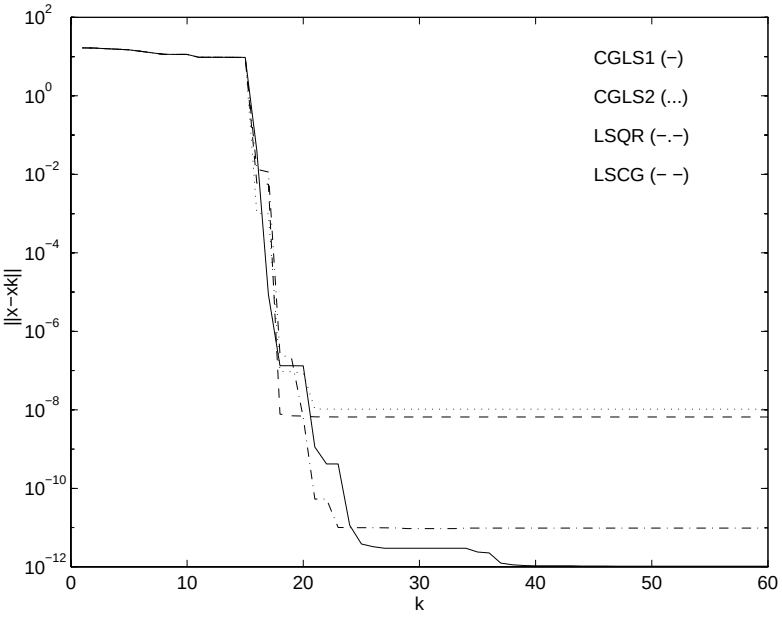

FIG. 6.4. $\|x - x_k\|$ *for problem* $PS(20, 10, 1, 4)$, $\kappa = 10^4$, $\kappa_{LS} = 6.81 \cdot 10^4$.

of $\kappa(A)$ worse. In Figure 6.2 the norm of the *recursive* residual is plotted. For CGLS1 this norm still decreased after 200 iterations, when it had reached 10^{-35}. Figure 6.3 shows the relative norm of the *true* residual, which for both stable versions reaches the level of machine precision after 50–70 iterations.

The second test problem was the inconsistent problem $PS(20, 10, 1, 4)$ with $\rho = 0.01$, for which $\kappa(A) = 10^4$, and $\kappa_{LS} = 6.81 \cdot 10^4$ (see (4.2)–(4.3)). Figure 6.4 shows

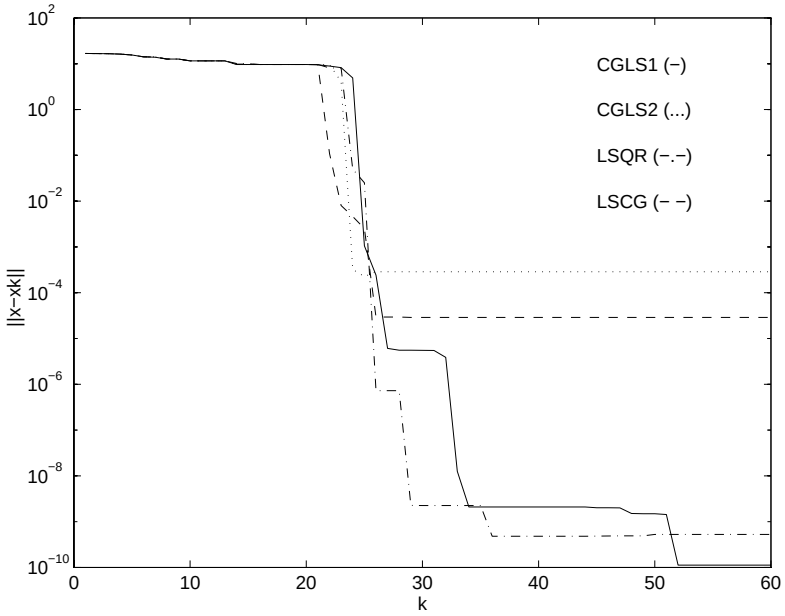

FIG. 6.5. $\|x - x_k\|$ *for problem* $PS(20, 10, 1, 6)$, $\rho = 0.001$, $\kappa = 10^6$, $\kappa_{LS} = 5.91 \cdot 10^7$.

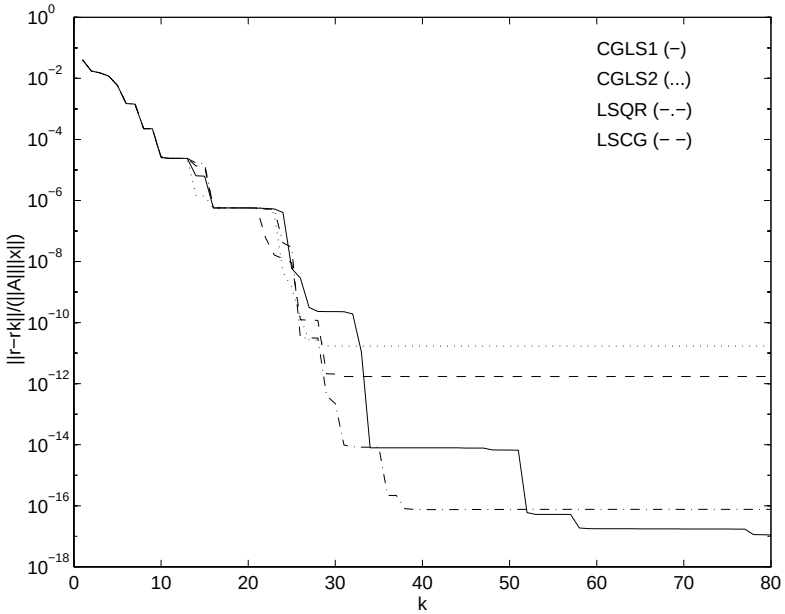

FIG. 6.6. $\|r - r_k\|/(\|A\|\|x\|)$ *for problem* $PS(20, 10, 1, 6)$, $\rho = 0.001$, $\kappa = 10^6$, $\kappa_{LS} = 5.91 \cdot 10^7$.

that the limiting relative accuracy in the solution is better than 10^{-11} for the stable versions. Again the accuracy for the unstable versions is worse by a factor of $\kappa(A)$.

In Figures 6.5–6.6 we show the results for the more ill-conditioned inconsistent problem $PS(20, 10, 1, 6)$ with $\rho = 0.001$, $\kappa(A) = 10^6$, and $\kappa_{LS} = 5.91 \cdot 10^7$. The relative accuracy in the computed solution, shown in Figure 6.5, is here around 10^{-9}

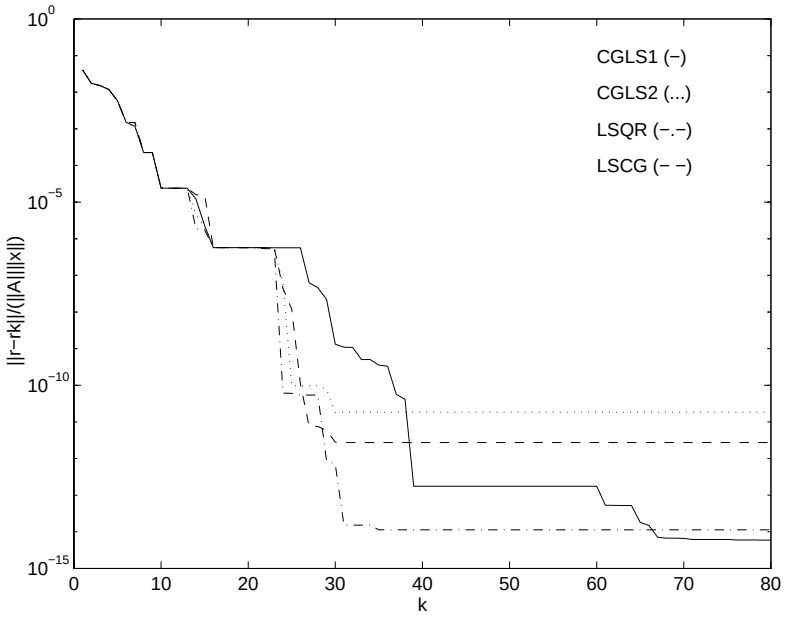

FIG. 6.7. $\|r - r_k\|/(\|A\|\|x\|)$ *for problem* $PS(20, 10, 1, 6)$, $\rho = 0.1$, $\kappa = 10^6$, $\kappa_{LS} = 5.81 \cdot 10^9$.

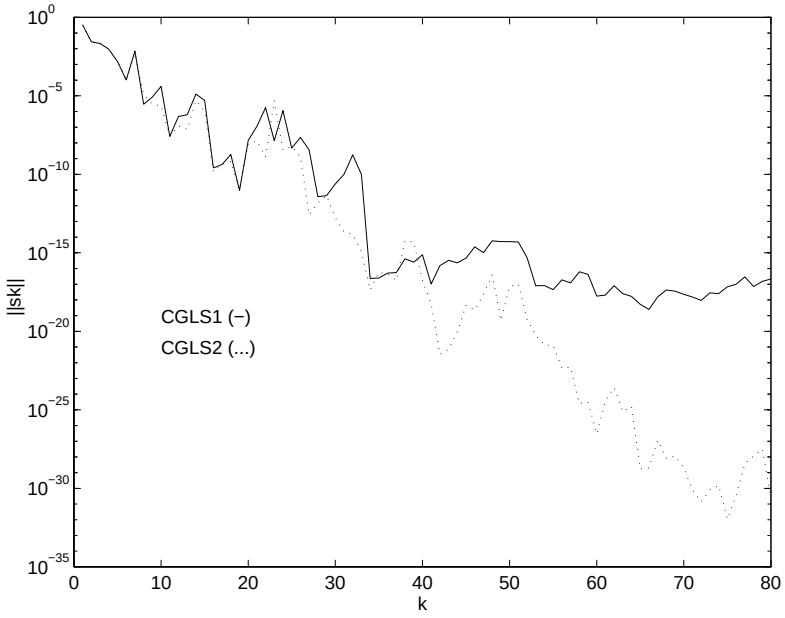

FIG. 6.8. $\|s_k\|$ *for problem* $PS(20, 10, 1, 6)$, $\rho = 0.001$, $\kappa = 10^6$, $\kappa_{LS} = 5.91 \cdot 10^7$.

for both stable versions. Figure 6.6 shows that in the limit $\|r - r_k\|/(\|A\|\,\|x\|)$ is less than 10^{-15}. In Figure 6.7 $\rho = 0.1$, $\kappa_{LS} = 5.81 \cdot 10^9$, and the limit is about 10^{-14}. Considering the ill-conditioning of the problem these results are much better than can be expected from a normwise backward stable method and support the assumption in (5.5).

Finally, in Figure 6.8 the norm of the recursive residual $s_k = A^T r_k$ of the normal

equations is plotted for the case $\rho = 0.001$. For CGLS1 this decreases to about machine precision, whereas for the unstable version CGLS2 it decreases to zero in the limit. Hence it is important *not* to base the comparison of the two algorithms on this quantity!

The results suggests some cheap stopping rules for compatible systems. As shown by Figures 6.2 and 6.3, in this case $\|r_k\|$ becomes excessively small and so will cause termination at a sensible point, even though $\|b - Ax_k\|$ will not be as small as predicted. For incompatible systems, as Figure 6.8 shows, $\|s_k\|$ levels off at $O(u)$. On the other hand, for the unstable version CGLS2, $\|s_k\|$ does eventually become very small. However, this does not mean that $\|r - (b - Ax_k)\|$ will be small and should not be used as a reason for using CGLS2.

7. Conclusions. We have studied two different implementations, CGLS1 and CGLS2, of the conjugate gradient method applied to the normal equations and two algorithms, LSQR and LSCG, based on Lanczos bidiagonalization. Although these four algorithms for solving the linear least squares problem $\min_x \|Ax - b\|$ are mathematically equivalent, their performance in finite precision differs significantly. The achievable accuracy in finite precision of CGLS2 and LSCG can be lower by a factor of

$$\frac{\|b\|}{\max\{\|r\|, \|x\|/\|A^\dagger\|\}}$$

than that of a backward stable method. For the preferred implementation CGLS1 as well as for LSQR a bound is derived for the error $\|r - r_k\|$ of the computed residual r_k. This bound shows that for the consistent case ($r = 0$) these two methods achieve an accuracy similar to a normwise backward stable method. For the inconsistent case CGLS1 and LSQR achieve even better accuracy than a normwise backward stable method. Numerical tests confirming this behavior have been given.

Acknowledgments. The authors would like to thank Tianruo Yang for carrying out most of the numerical tests. We also thank Michael Saunders for giving valuable comments on the error analysis and on stopping rules. An anonymous referee report also helped to improve the paper.

REFERENCES

[1] Å. Björck, *Conjugate Gradient Methods for Sparse Least Squares Problems*, Unpublished notes, Computer Science Department, Stanford University, Stanford, CA, 1979.
[2] Å. Björck, *Component-wise perturbation analysis and errors bounds for linear least squares solutions*, BIT, 31 (1991), pp. 238–244.
[3] Å. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, PA, 1996.
[4] Å. Björck, E. Grimme, and P. Van Dooren, *An implicit bidiagonalization algorithm for ill-posed systems*, BIT, 34 (1994), pp. 510–534.
[5] Y. T. Chen, *Iterative Methods for Linear Least Squares Problems*, Technical report CS-75-04, Department of Computer Science, University of Waterloo, Waterloo, ON, Canada, 1975.
[6] E. J. Craig, *The N-step iteration procedures*, J. Math. Phys., 34 (1955), pp. 64–73.
[7] T. Elfving, *On the Conjugate Gradient Method for Solving Linear Least Squares Problems*, Report LiTH-MAT-R-78-3, Department of Math., Linköping University, Linköping, Sweden, 1978.
[8] R. W. Freund, G. H. Golub, and N. Nachtigal, *Iterative solution of linear systems*, Acta Numerica, (1991), pp. 57–100.
[9] G. H. Golub and W. Kahan, *Calculating the singular values and pseudo-inverse of a matrix*, SIAM J. Numer. Anal. Ser. B, 2 (1965), pp. 205–224.
[10] A. Greenbaum, *Behavior of slightly perturbed Lanczos and conjugate-gradient recurrences*, Linear Algebra Appl., 113 (1989), pp. 7–63.

[11] A. Greenbaum, *Estimating the attainable accuracy of recursively computed residual methods*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 535–551.
[12] A. Greenbaum and Z. Strakoš, *Predicting the behavior of finite precision Lanczos and conjugate gradient computations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 121–137.
[13] M. T. Heath, *Numerical methods for large sparse linear least squares problems*, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 497–513.
[14] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Stds., B49 (1952), pp. 409–436.
[15] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, PA, 1996.
[16] N. J. Higham and P. A. Knight, *Finite precision behavoir of stationary iteration for solving singular systems*, Linear Algebra Appl., 192 (1993), pp. 165–186.
[17] P. Läuchli, *Iterative Lösung und Fehlerabschätzung in der Ausgleichrechnung*, ZAMP, 10 (1959), pp. 245–280.
[18] C. L. Lawson, *Sparse Matrix Methods based on Orthogonality and Conjugacy*, Technical mem. 33-627, Jet Propulsion Lab., California Institute of Technology, 1973.
[19] C. C. Paige, *Bidiagonalization of matrices and solution of linear equations*, SIAM J. Numer. Anal., 11 (1974), pp. 197–209.
[20] C. C. Paige and M. A. Saunders, *LSQR: An algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Software, 8 (1982), pp. 43–71.
[21] J. K. Reid, *On the method of conjugate gradients for the solution of large systems of linear equations*, in Large Sparse Sets of Linear Equations, J. K. Reid, ed., Academic Press, New York, 1971, pp. 231–254.
[22] D. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385.
[23] E. Stiefel, *Ausgleichung ohne Aufstellung der Gausschen Normalgleichungen*, Wiss. Z. Technische Hochschule Dresden, 2 (1952/53), pp. 441–442.
[24] Z. Strakoš, *On the real convergence of the conjugate gradient method*, Linear Algebra Appl., 154/156 (1991), pp. 535–549.