A ROBUST PRECONDITIONER WITH LOW MEMORY REQUIREMENTS FOR LARGE SPARSE LEAST SQUARES PROBLEMS*

MICHELE BENZI† AND MIROSLAV TŮMA‡

Abstract. We describe a technique for constructing robust preconditioners for the CGNR method applied to the solution of large and sparse least squares problems. Our algorithm computes an incomplete LDL^T factorization of the normal equations matrix without the need to form the normal matrix itself. The preconditioner is reliable (pivot breakdowns cannot occur) and has low intermediate storage requirements. Numerical experiments illustrating the performance of the preconditioner are presented.

Key words. Large sparse least squares problems, preconditioned CGNR, robust incomplete factorization, incomplete C-orthogonalization

AMS subject classifications. Primary 65F10, 65F20, 65F25, 65F35, 65F50. Secondary 15A06.

1. Introduction. In this paper we consider the solution of linear least squares problems of the form

$$||b - Ax||_2 = \min \tag{1.1}$$

where the coefficient matrix A is large and sparse. We assume that A is $m \times n$ with $m \geq n$ and A has full column rank. Although the techniques considered in this paper are applicable to the square case ($m = n$), we are mostly interested in the overdetermined case ($m > n$).

Björck [5] gives a comprehensive treatment of available solution algorithms for problem (1.1). In the large and sparse case, there are two main approaches to solve (1.1), namely, sparse direct methods based on orthogonalization and iterative methods based on the conjugate gradient algorithm implicitly applied to the normal equations. For overdetermined systems, the iterative method of choice is the CGNR (or CGLS) algorithm. Recall that the normal equations are

$$Cx = f, \quad C = A^T A, \quad f = A^T b. \tag{1.2}$$

Sparse direct solvers are very reliable, but they may be prohibitively expensive in terms of storage and operation count for very large problems. Iterative methods generally require much less storage and have the potential to be faster in terms of execution time, but only if their convergence is sufficiently rapid. For the CGNR method, this means that a good preconditioner is needed.

In this paper we present a new approach to construct reliable preconditioners for the CGNR method. Our method is based on C-orthogonalization, i.e., orthogonalization with respect to the inner product

$$\langle x, y \rangle_C := x^T C y \quad \text{for all } x, y \in \mathbb{R}^n. \tag{1.3}$$

*This version dated 26 April 2002.

†Department of Mathematics and Computer Science, Emory University, Atlanta, Georgia 30322, USA (benzi@mathcs.emory.edu).

‡Institute of Computer Science, Academy of Sciences of the Czech Republic, Pod vodárenskou věží 2, 182 07 Prague 8, Czech Republic (tuma@cs.cas.cz). The work of this author was supported by Grant Agency of the Academy of Sciences of the Czech Republic grants 1030103 and 2030801.

We show how C-orthogonalization can be used to compute a root-free incomplete factorization of the normal equations matrix

$$C \approx LDL^T$$

where L is a $n \times n$ unit lower triangular matrix and D is diagonal and positive definite. Our algorithm enjoys the following desirable properties:

1. No entry of $C = A^T A$ needs to be explicitly computed (the algorithm works entirely with A);
2. The incomplete factorization process cannot break down;
3. Intermediate storage requirements are negligible.

Of course, properties 1–3 alone are not enough unless the preconditioner also significantly reduces the solution time compared to the unpreconditioned iteration. We will show experimentally that our preconditioner results in good convergence rates when applied to large and sparse least squares problems.

The remainder of the paper is organized as follows. In section 2 we briefly review some of the previous work on preconditioners for the CGNR method. In section 3 we present the basic C-orthogonalization scheme and explain how it can be used to compute a root-free Cholesky factorization of the normal equations matrix. The preconditioner based on this scheme is described in section 4. Numerical experiments are presented in section 5, and some conclusions are given in section 6.

2. Previous work. General-purpose preconditioners for solving least squares problems have been proposed by several authors. An early paper by Läuchli [16] considers a preconditioner for (1.1) based on the LU factorization of a $n \times n$ nonsingular submatrix of A. Partition A as

$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$$

where the $n \times n$ matrix A_1 is nonsingular. This can be ensured by suitable pivoting strategies. Then the matrix $A_1 = LU$ can be used as a left preconditioner for (1.1). The normal equations matrix corresponding to the preconditioned least squares problems is readily seen to be

$$(AA_1^{-1})^T (AA_1^{-1}) = I_n + B^T B, \quad \text{where } B = A_2 A_1^{-1}.$$

Since B has at most $p = \min\{m - n, n\}$ distinct singular values, rapid convergence can be expected when $p \ll n$. Matrix-vector products with this matrix require solving linear systems with coefficient matrices A_1 and A_1^T, which is done using the LU factorization of A_1. Variations on this basic idea have been explored by several authors; see Ch. 7.5.3 in [5] and the paper [6] and the references therein.

In [22], Saunders proposed to use Gaussian elimination with partial pivoting to compute a stable, sparse factorization $PA = LU$ where the $m \times n$ matrix L is unit lower trapezoidal and U is upper triangular (P is a $m \times m$ permutation matrix). The factor U is then used as a left preconditioner for (1.1). The matrix L is not saved, and applying the preconditioner only requires backsubstitution with U. This approach is based on the observation that L is frequently well-conditioned, and that U tends to reflect most of the ill-conditioning of the original matrix A. Furthermore, the U factor can be computed with little fill-in in many cases.

Most of the recent activity in preconditioning sparse least squares problems, however, has been based on incomplete variants of the QR, rather than LU, factorization. The basic idea is the following. Let $A = QR$ be the "thin" QR factorization of A, where Q is $m \times n$ with orthonormal columns, and R is $n \times n$ upper triangular. The factorization is not unique, but it can be made unique by requiring that the diagonal entries r_{ii} of R are all positive; see [8], p. 230. In this case, $A^T A = R^T R$ is the Cholesky factorization of the normal equations matrix $C = A^T A$. If we could compute this factorization exactly, we could use the R factor as a left preconditioner for CGNR applied to problem (1.1) to get convergence in a single iteration. Indeed, the preconditioned normal equations matrix is

$$(AR^{-1})^T(AR^{-1}) = Q^T Q = I_n$$

and convergence takes place in one step. To obtain a feasible preconditioner, an approximate factorization $A \approx \bar{Q}\bar{R}$ is computed; here $\bar{R}$ is still lower triangular with positive diagonal entries, but the columns of $\bar{Q}$ may no longer be mutually orthogonal in general. The approximate $\bar{R}$ factor is usually considerably more sparse than the exact Cholesky factor R of $C = A^T A$. This can be interpreted as an incomplete Cholesky factorization of C. The closer is $\bar{R}$ to the exact Cholesky factor, the closer the preconditioned normal equations matrix is to the identity matrix:

$$(A\bar{R}^{-1})^T(A\bar{R}^{-1}) = \bar{R}^{-T}A^T A\bar{R}^{-1} \approx I_n,$$

and the faster the CGNR iteration converges. Note that there is no need for the $\bar{Q}$ factor in the iterative phase of the algorithm, therefore $\bar{Q}$ does not need to be saved.

The first paper proposing to compute an incomplete orthogonal factorization of A for use as a preconditioner for the CGNR method is [10]. In it, the authors consider both methods based on Givens rotations, and methods based on the Gram–Schmidt process. Sparsity in $\bar{R}$ (and possibly in $\bar{Q}$) is preserved by applying a (relative) drop tolerance: fill elements are dropped if they are "small" according to some criterion. The possibility of breakdowns (singular $\bar{R}$) is considered; in some cases, diagonal corrections may be needed to preserve positivity of the diagonal entries of $\bar{R}$. Some breakdown-free variations of the algorithms are proposed. All of these algorithms work with A only, and there is no need to form $C = A^T A$ explicitly.

Subsequent papers include [20] and [28] with focus on algorithms based on incomplete Gram–Schmidt and Givens rotations, respectively; see also [27]. Other references include [3] and [26] for preconditioners based on incomplete Gram–Schmidt, and [1] and [18] for Givens-based methods. In [21], shifted incomplete orthogonalization methods are studied. The focus of the paper is to develop heuristics for the automatic selection of global diagonal shifts aimed at increasing the stability of the preconditioner.

While incomplete QR methods can be very reliable (at least for sufficiently accurate approximations to the full QR decomposition) and often result in fast convergence of the preconditioned CGNR iteration, they suffer from high set-up costs. Moreover, intermediate storage requirements are typically very high. As an example, we report some results obtained with the Gram–Schmidt-based incomplete QR method of Jennings and Ajiz [10]. We use a rather small square matrix, WEST0655, from the Matrix Market [17]. This matrix has dimension $m = n = 655$ and contains $nnz(A) = 2854$ nonzero entries. Its condition number is estimated to be of the order of 10^{12}, thus the normal equation matrix C has condition number of the order of 10^{24}. The matrix C, if explicitly formed, would contain $nnz(C) = 10,672$ nonzeros. Prior to applying the

Jennings–Ajiz incomplete QR preconditioner, we permute the columns of A so that C is reordered according to a minimum degree algorithm. This greatly reduces the amount of fill-in generated during the incomplete orthogonalization process. We use a drop tolerance resulting in an incomplete factor with $nnz(\bar{R}) = 10,997$ nonzeros. The preconditioned CGNR algorithm is initialized with a zero initial guess and is stopped when the initial residual is reduced by eight orders of magnitude. The right-hand side vector b is chosen so that the solution is the vector of all ones. Convergence takes place in 47 iterations, which is reasonable for such an ill-conditioned problem. However, the amount of intermediate fill-in incurred by the Jennings–Ajiz algorithm is very high: the maximum number of nonzero elements that have to be kept in storage at any given time during the course of the incomplete Gram–Schmidt process is $44,620$, almost sixteen times the number of nonzeros in the original matrix A. Clearly, this is a severe drawback of the algorithm, especially in applications involving large matrices.

Algorithms based on Givens rotations are also limited in their usefulness by their high intermediate costs as well as by the often large amounts of intermediate fill-in generated in the course of the incomplete orthogonalization process; see [18].

Clearly, an incomplete Cholesky factor of C can always be obtained by computing $C = A^T A$ explicitly and then applying a standard incomplete Cholesky factorization algorithm to C. As noted in [5], there is actually no need to form all of C explicitly; rather, its rows can be computed one at a time, used to perform the corresponding step of the incomplete Cholesky algorithm, and then discarded. Nevertheless, forming the normal equations, even piecemeal, entails some overhead and may lead to severe loss of information in very ill-conditioned cases. Also, for a general symmetric positive definite (SPD) matrix, standard incomplete Cholesky factorization algorithms may fail due to pivot breakdowns (that is, negative or zero pivots). There exist reliable incomplete factorization algorithms that can be applied to a general SPD matrix without breakdowns; see [23, 24, 25, 13, 4] and the references therein. (Note that the first four of these papers consider different variations of the same idea.) However, these techniques either require access to the entries of C, or have high intermediate storage requirements, or both. The only exception is the Robust Incomplete Factorization (RIF) method introduced in [4]. This is the method that we propose to use to compute reliable preconditioners for problem (1.1).

3. C-orthogonalization and the normal equations. We start by recalling that since A has full column rank, the $n \times n$ matrix $C = A^T A$ is SPD and therefore it defines an inner product on $\mathbb{R}^n$ via (1.3). Given a set of n linearly independent vectors $v_1, v_2, \ldots, v_n \in \mathbb{R}^n$, we can build a C-orthogonal (or C-conjugate) set of vectors $z_1, z_2, \ldots, z_n \in \mathbb{R}^n$ by a *conjugate Gram–Schmidt process*, i.e., a Gram–Schmidt process with respect to the inner product (1.3). Written as a *modified* Gram–Schmidt process, the (right-looking) algorithm starts by setting $z_i = v_i$ and then performs the following nested loop:

$$z_i \leftarrow z_i - \frac{\langle z_j, z_i \rangle_C}{\langle z_j, z_j \rangle_C} z_j \tag{3.4}$$

where $j = 1, 2, \ldots, n-1$ and $i = j+1, \ldots, n$. Letting

$$Z = [z_1, z_2, \ldots, z_n],$$

we have

$$Z^T C Z = D = \text{diag}\,(d_1, d_2, \ldots, d_n) \tag{3.5}$$

where

$$d_j = \langle z_j, z_j \rangle_C = z_j^T C z_j = (Az_j)^T (Az_j) = ||Az_j||_2^2 > 0, \quad 1 \le j \le n.$$

If we set $v_i = e_i$ (the i-th unit basis vector) for $1 \le i \le n$, then $Z^T = L^{-1}$ where L is the unit lower triangular factor in the root-free Cholesky factorization $C = LDL^T$; the matrix D is exactly the same here and in (3.5). Indeed, it is clear from (3.4) that the vector z_i is modified only above position i (for $2 \le i \le n$), therefore Z is unit upper triangular and by virtue of (3.5) and the uniqueness of the LDL^T factorization, it must be $Z^T = L^{-1}$.

Now, it was observed in [4] that the conjugate Gram–Schmidt process (3.4) produces not just the Z factor but also, at the same time, the L factor itself. To see this, observe that L in the LDL^T factorization of C and the inverse factor Z satisfy

$$CZ = LD, \quad \text{or} \quad L = CZD^{-1}.$$

This easily follows from (3.5) and the fact that $Z^T = L^{-1}$. By equating corresponding entries of CZD^{-1} and $L = [l_{ij}]$ we find that

$$l_{ij} = \frac{\langle z_j, z_i \rangle_C}{\langle z_j, z_j \rangle_C}, \quad i \ge j. \tag{3.6}$$

Thus, the L factor of C can be obtained as a by-product of the C-orthogonalization process (3.4), *at no extra cost.* This observation is the basis for the RIF preconditioner developed in [4] for solving general SPD systems. In the next section we show how this technique can be used to compute reliable preconditioners for problem (1.1).

4. Robust incomplete factorization. Two different types of preconditioners can be obtained by carrying out the C-orthogonalization process (3.4) incompletely. Given a drop tolerance $0 < \tau \ll 1$, the entries of z_i are scanned after each update and entries that are smaller than τ in absolute value are discarded. We denote by $\bar{z}_i$ the sparsified vector, and we set $\bar{Z} = [\bar{z}_1, \bar{z}_2, \ldots, \bar{z}_n]$. In alternative, a *relative* drop tolerance can be used; for example, τ can be replaced by $\tau ||a_i||_2$, where a_i is the i-th column of A. It is often advantageous to scale A so that $||a_i||_2 = 1$, where a_i is the i-th column of A (i.e., C has unit diagonal); this tends to improve the conditioning of the normal equations and it allows for the use of an absolute drop tolerance τ. Whatever the scaling or the drop strategy used, the incomplete C-orthogonalization process results in a sparse matrix $\bar{Z} \approx L^{-T}$, that is, we have an incomplete inverse factorization of C of the form

$$C^{-1} \approx \bar{Z} \bar{D}^{-1} \bar{Z}^T$$

where $\bar{D}$ is diagonal with entries $\bar{d}_j = \bar{z}_j^T C \bar{z}_j > 0$. This is a factored sparse approximate inverse that can be used as a preconditioner for the conjugate gradient algorithm applied to $Cx = f$; see [2, 14]. The preconditioner is guaranteed to be positive definite (since $\bar{d}_j > 0$ for all j) and is easily applied in parallel, since its application only requires matrix–vector products. It is generally known as the SAINV (for Stabilized Approximate Inverse) preconditioner.

Note that the construction of the preconditioner does not require forming $C = A^T A$ explicitly. Indeed, the main loop (3.4) involves the computation of the inner products

$$\langle \bar{z}_j, \bar{z}_i \rangle_C = \bar{z}_j^T C \bar{z}_i = (A\bar{z}_j)^T A \bar{z}_i, \quad i = j, \ldots n, \ j = 1, \ldots n-1.$$

Hence, computing the multipliers in (3.4) only involves matrix–vector products of the form $A\bar{z}_i$, to be computed as a linear combination of the columns of A corresponding to nonzero entries in $\bar{z}_i$, and inner products of two sparse vectors $A\bar{z}_i$ and $A\bar{z}_j$. Typically, most of these inner products will be structurally zero (that is, $A\bar{z}_i$ and $A\bar{z}_j$ have no nonzero entries in the same position) and the corresponding update in (3.4) can be skipped. It is important to mention that our implementation makes use of structural information on the incomplete inverse factor $\bar{Z}$ so as to avoid checking which of the inner products are structurally zero, which would be an $\mathcal{O}(n^2)$ operations; see [2]. Note that the $\bar{z}_i$ vectors are stored (they form the columns of the approximate inverse factor $\bar{Z}$), whereas the multipliers $\langle \bar{z}_j, \bar{z}_i\rangle_C / \langle \bar{z}_j, \bar{z}_j\rangle_C$ are discarded after they are used to perform an update step.

The second preconditioner that can be obtained is, in a sense, the *dual* of the previous one. In this algorithm we save the multipliers

$$\bar{l}_{ij} = \frac{\langle \bar{z}_j, \bar{z}_i\rangle_C}{\langle \bar{z}_j, \bar{z}_j\rangle_C} = \frac{(A\bar{z}_j)^T A\bar{z}_i}{(A\bar{z}_j)^T A\bar{z}_j}, \quad i \geq j \tag{4.7}$$

and we discard the vector $\bar{z}_i$ as soon as it has been used to form the corresponding parts of the incomplete factor $\bar{L} = [\bar{l}_{ij}]$ of C. Hence, we have an algorithm to compute an incomplete root-free Cholesky factorization

$$C = A^T A \approx \bar{L}\bar{D}\bar{L}^T$$

of the normal equations matrix.

Note that this incomplete triangular factorization of C does not require forming the matrix $C = A^T A$ itself (not even one row at a time): the incomplete conjugate Gram–Schmidt process on which it is based works exclusively with A. Moreover, the preconditioner is guaranteed to be positive definite and no breakdown in the course of the incomplete factorization is possible. This follows from the fact that the pivots $\bar{d}_j$ are given by $\bar{d}_j = (A\bar{z}_j)^T A\bar{z}_j$, an inherently positive quantity. Note that from well-known extremal properties of the Rayleigh quotient, the following lower bound for the generic pivot $\bar{d}_j$ holds:

$$\bar{d}_j = \langle \bar{z}_j, \bar{z}_j\rangle_C = \bar{z}_j^T C\bar{z}_j \geq \lambda_{\min}(C)||\bar{z}_j||_2^2 = \sigma_{\min}^2(A)||\bar{z}_j||_2^2 > 0,$$

where $\lambda_{\min}(C)$ and $\sigma_{\min}(A)$ denote the smallest eigenvalue of C and the smallest singular value of A, respectively. Also note that $||\bar{z}_j||_2^2 \geq 1$ since the j-th entry of $\bar{z}_j$ is equal to one. Because of the breakdown-free property, we denote this preconditioner by RIF (for Robust Incomplete Factorization). While it is possible in principle that a pivot is so small to be numerically zero, we have not encountered a single case where this happened (even for very ill-conditioned A). We could in principle push any such tiny pivot away from zero simply by multiplying A by a sufficiently large number, but this could cause overflow. Again, we have never had to resort to any trick of this sort in actual computations, although our codes can perform local pivot modifications in such a situation.

We stress the very important fact that the construction of the RIF preconditioner incurs only modest intermediate storage costs; the total storage is dominated by the storage for A and for the final incomplete factor $\bar{L}$. The algorithm requires some temporary storage for the sparse $\bar{z}_i$ vectors while they are needed to compute the multipliers (4.7). At step j of the algorithm $(1 \leq j \leq n-1)$ we need to store, besides the j computed columns of the incomplete factor $\bar{L}$, the remaining $n - j$

columns $\bar{z}_{j+1}, \ldots, \bar{z}_n$ of $\bar{Z}$. Notice that these sparse vectors can have nonzero entries only within the first j positions, besides the 1 in position k of $\bar{z}_k$. As j increases, there are fewer and fewer such vectors that need to be kept in temporary storage, but they tend to fill in. Assuming a uniform distribution of nonzeros in both $\bar{L}$ and $\bar{Z}$, a back-of-the-envelope calculation shows that in terms of storage, the "high-water mark" is reached for $j \approx n/2$, that is, mid-way through the C-orthogonalization process, after which it begins to decrease. The total storage can be estimated to be approximately 25% more than the storage required by the final incomplete factor $\bar{L}$. In practice, we found that this is often an overestimate, unless a very small drop tolerance is used (which is not practical anyway). With a careful implementation and using suitable dynamic data structures, the proposed algorithm incurs negligible intermediate storage requirements.

At first sight, the RIF preconditioner needs two drop tolerances: one for the incomplete C-orthogonalization process, to be applied to the $\bar{z}_i$ vectors, and a second one to be applied to the entries of $\bar{L}$. Note that the latter is simply a post-filtration: once a column of $\bar{L}$ has been computed, it does not enter the computation of the remaining ones. In practice we found that this post-filtration is usually not necessary, and indeed it can be harmful to the quality of the preconditioner; see [4]. Therefore, post-filtration in $\bar{L}$ is not recommended, and dropping is applied only to the $\bar{z}_i$ vectors.

The RIF preconditioner is generally more effective at reducing the number of CGNR iterations than the SAINV one (for a comparable density of the incomplete factors). The main advantage of SAINV is that it can be easily applied in parallel. In the remainder of the paper, we consider the RIF preconditioner only.

5. Numerical experiments. In this section we present the results of some numerical experiments aimed at assessing the performance of the RIF preconditioner for least squares problems.

In Table 1 we provide some basic information about the test problems used in the numerical experiments. We used eight rectangular matrices of widely different size. The first four problems (WELL1033, ILLC1033, WELL1850, ILLC1850) are from the Matrix Market repository [17]. The remaining four (SMALL, MEDIUM, LARGE, VERYL) arise in animal breeding analysis [11, 12][1]. For each matrix we report the number m of rows, the number n of columns, and the number nnz of nonzeros.

TABLE 1
Test problem information.

Matrix	m	n	nnz
WELL1033	1033	320	4732
ILLC1033	1033	320	4732
WELL1850	1850	712	8758
ILLC1850	1850	712	8758
SMALL	3140	1988	8510
MEDIUM	9397	6119	25013
LARGE	28524	17264	75018
VERYL	174193	105882	463303

Tables 2-3 contain results for the four smaller matrices using incomplete factors of variable densities obtained by adjusting the drop tolerance τ. For RIF, we typ-

[1]These matrices can be downloaded from `ftp://ftp.cerfacs.fr/pub/algo`.

ically take τ between 0.05 and 0.1. We also present results for a standard, drop tolerance-based incomplete Cholesky factorization preconditioner applied to the (explicitly formed) normal equations matrix $C = A^T A$. We denote this preconditioner by ICNE. The drop tolerance is chosen so as to obtain preconditioners of similar density to those computed with RIF. For each method we report the number of nonzeros in the incomplete factor (under "size"), the time to construct the preconditioner (under "P-time"), the number of preconditioned CGNR iterations needed to achieve

$$||A^T(b - Ax^{(k)})||_2 < 10^{-8}||A^T(b - Ax^{(0)})||_2$$

(under "Its") and the time for the iterative solution phase (under "It-time"). The initial guess was $x^{(0)} = 0$, and the right-hand side b was chosen so that the solution was the vector of all ones. For these experiments, no scalings or sparse matrix reorderings were applied. Our codes are written in Fortran90. All runs were performed on an SGI Origin 200 computer. For completeness, we show in Table 4 the results for the unpreconditioned CGNR method. It can be seen that solution times are significantly reduced by preconditioning, except for the well-conditioned problem WELL1033.

TABLE 2
Test results for RIF, small problems.

	RIF			
Matrix	size	P-time	Its	It-time
WELL1033	911	0.03	72	0.04
ILLC1033	825	0.03	256	0.13
WELL1850	2835	0.05	89	0.08
ILLC1850	2904	0.04	248	0.23

TABLE 3
Test results for ICNE, small problems.

	ICNE			
Matrix	size	P-time	Its	It-time
WELL1033	866	0.005	108	0.05
ILLC1033	816	0.005	286	0.14
WELL1850	2595	0.01	182	0.17
ILLC1850	2691	0.01	774	0.72

TABLE 4
Test results for CGNR, small problems.

	CGNR	
Matrix	Its	It-time
WELL1033	167	0.07
ILLC1033	3212	1.34
WELL1850	424	0.32
ILLC1850	2047	1.49

For these problems, we see that RIF tends to result in better convergence rates than ICNE for preconditioners of comparable density. The time to compute the preconditioner is smaller for ICNE than for RIF; note that these problems are so small that for both algorithms the preconditioner construction time is dominated by non-floating point operations. A major difference between the two methods is that ICNE is much more sensitive than RIF to the choice of the drop tolerance, and the dependency is non-monotonic. Indeed, if we try to improve the convergence rate for ICNE by reducing the drop tolerance (thereby allowing extra fill-in in the incomplete factor), the number of iterations may actually increase. For example, consider problem ILLC1850. If we increase the number of nonzeros in the ICNE factor to 3229, the number of iterations goes up to 2515; if we further increase it to 3461, it takes 5393 iterations to converge. On the other hand, reducing the size of the preconditioner to 1868 nonzeros results in 1013 iterations. For this matrix, 774 iterations is the best result we could get with ICNE using reasonably sparse preconditioners. We think that this non-monotonic behavior of ICNE with respect to the drop tolerance reflects the instability of applying a standard incomplete Cholesky factorization to the explicitly formed normal equations matrix. In contrast, RIF behaves in a stable and predictable manner: decreasing the drop tolerance leads to faster convergence of the preconditioned CGNR iteration, although not necessarily to reduced CPU times since each iteration is now more costly.

Table 5
Test results for RIF, animal breeding problems.

	RIF			
Matrix	size	P-time	Its	It-time
SMALL	**12199**	**0.22**	**32**	**0.05**
MEDIUM	**28420**	**0.20**	**53**	**0.31**
	37248	0.70	34	0.21
LARGE	**76094**	**0.54**	**63**	**0.96**
	82287	0.70	58	0.91
	107007	3.62	39	0.70
VERYL	**426830**	**5.18**	**199**	**26.0**
	272207	3.10	225	34.9
	249327	2.17	206	31.4

Table 6
Test results for ICNE, animal breeding problems.

	ICNE			
Matrix	size	P-time	Its	It-time
SMALL	**17377**	**0.05**	**33**	**0.06**
	13443	0.03	134	0.24
MEDIUM	**55950**	**0.14**	**246**	**1.50**
	47714	0.11	†	†
LARGE	**224430**	**1.38**	**38**	**1.00**
	180309	0.85	†	†
	163157	0.69	†	†
VERYL	1185773	7.71	127	31.9
	962079	**5.72**	**135**	**29.2**
	730622	4.02	507	104.

TABLE 7
Test results for CGNR, animal breeding problems.

Matrix	CGNR	
	Its	It-time
SMALL	308	0.30
MEDIUM	390	1.12
LARGE	573	5.29
VERYL	1303	142.9

Tables 5-7 present results with the four larger problems. For each of the two preconditioners and for each of the four problems, we highlight the best results with boldface. Again, we see that RIF is generally better than ICNE, sometimes by a large margin. Generally speaking, for these problems ICNE tends to require a lot more fill-in than RIF in order to achieve convergence. Indeed, for certain choices of the drop tolerance ICNE preconditioning failed to produce convergence in less than $10,000$ iterations (denoted by "†" in Table 5). This never happened with RIF.

In all cases, the amount of additional temporary storage needed to set up the RIF preconditioner was only a small fraction of the space needed to store the coefficient matrix and the final incomplete factor $\bar{L}$. As an example, for the VERYL test problem the additional overhead in temporary storage needed for constructing the preconditioner was only about 4% of the space needed to store the incomplete factor.

6. Conclusions. In this paper we have presented a reliable preconditioner with low storage requirements for large sparse least squares problems. The preconditioner is an incomplete LDL^T factorization of the normal equations matrix $C = A^TA$ based on C-orthogonalization. The preconditioner construction cannot break down and does not require forming the matrix product $C = A^TA$ explicitly, not even one row at a time as required by some of the previously developed techniques. While somewhat more expensive to compute than more standard incomplete Cholesky factorization algorithms, numerical experiments indicate that our method results in better convergence rates and behaves in a stable and predictable manner with respect to the drop tolerance used.

We think that this approach has considerable advantages over preconditioners based on incomplete orthogonalization of A, which tend to require large amounts of temporary storage and are often very expensive. In particular, the size of the data structures required to implement incomplete QR preconditioners is generally much larger than that required by RIF [3, 18]. For this reason, we expect RIF to be preferrable whenever storage is at a premium. A direct comparison of RIF and incomplete orthogonalization preconditioners, however, is outside the scope of the present paper.

Besides as a preconditioner for CGNR applied to the normal equations, there is at least one other situation where RIF should prove useful. Consider a saddle point problem of the form

$$\begin{bmatrix} F & A \\ A^T & O \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ c \end{bmatrix} \tag{6.8}$$

where F is $n \times n$ and SPD, A is $n \times m$ with $n > m$ and has full column rank, $b \in \mathbb{R}^n$ and $c \in \mathbb{R}^m$ are given, and $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ are the unknowns. One of the most effective techniques for solving problem (6.8) is to use a Krylov subspace method (such as symmetric QMR, or GMRES) with the following preconditioner:

$$M = \begin{bmatrix} I_n & A \\ A^T & O \end{bmatrix}$$

(the so-called *constraint preconditioner*; see, e.g., [15]). Application of this preconditioner within a Krylov subspace method requires solving a least squares problem of the form (1.1) at each iteration. In principle this could be done by a sparse QR factorization, or by forming and then factoring the normal equations. In alternative, a sparse direct indefinite factorization of M could be used [7]. However, for large-scale problems such as three-dimensional mixed finite element formulations, these direct solvers become prohibitively expensive; see, e.g., [19]. As explained in [19], a sparse incomplete factorization of A^TA can be used to obtain a cheap approximation of the action of M^{-1} without too much of a negative effect on the rate of convergence that would be observed with the "exact" preconditioner. The experiments in [19] show that dramatic savings in both storage and CPU time are achieved. The results in [19] were obtained with an incomplete Cholesky factorization of the normal equations matrix similar to the ICNE algorithm of the previous section; we expect that using RIF instead would result in improved results in many cases. Some preliminary experiments have been performed by John Haws in his Ph.D. thesis [9], but this is a topic that deserves further research.

REFERENCES

[1] Z. Z. Bai, I.S. Duff, and A. J. Wathen, *A class of incomplete orthogonal factorization methods. I: Methods and theories*, BIT, 41 (2001), pp. 53–70.

[2] M. Benzi, J. K. Cullum, and M. Tůma, *Robust approximate inverse preconditioning for the conjugate gradient method*, SIAM J. Sci.Comput., 22 (2000), pp. 1318-1332.

[3] M. Benzi and M. Tůma, *A comparison of some preconditioning techniques for general sparse matrices*, in S. D. Margenov and P. S. Vassilevski, editors, *Iterative Methods in Linear Algebra, II*, IMACS Series in Computational and Applied Mathematics, vol. 3, IMACS, New Brunswick, NJ (1996), pp. 191–203.

[4] M. Benzi and M. Tůma, *A robust incomplete factorization preconditioner for positive definite matrices*, Numer. Linear Algebra Appl., to appear.

[5] Å. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.

[6] Å. Björck and J. Y. Yuan, *Preconditioners for least squares problems by LU factorization*, Electr. Trans. Numer. Anal., 8 (1997), pp. 26–35.

[7] I. S. Duff and J. K. Reid, *The multifrontal solution of indefinite sparse symmetric linear equations*, ACM Trans. Math. Software, 9 (1983), p. 302–325.

[8] G. H. Golub and C. F. van Loan, *Matrix Computations. Third Edition*, Johns Hopkins University Press, Baltimore and London, 1996.

[9] J. C. Haws, *Preconditioning KKT Systems*, Ph.D. thesis, Department of Mathematics, North Carolina State University, Raleigh, NC, 2002.

[10] A. Jennings and M. A. Ajiz, *Incomplete methods for solving $A^TAx = b$*, SIAM J. Sci. Stat. Comput., 5 (1984), pp. 978–987.

[11] M. Hegland, *On the computation of breeding values*, in H. Burkhart, ed., *Proceedings of CONPAR 90-VAPP IV Joint International Conference on Vector and Parallel Processing*, Lecture Notes in Computer Science, Springer-Verlag, Berlin (1990), pp. 232–242.

[12] M. Hegland, *Description and Use of Animal Breeding Data for Large Least Squares Problems*, Technical Report TR/PA/93/50, CERFACS, Toulouse, France, 1993.

[13] I. E. Kaporin, *High quality preconditioning of a general symmetric positive definite matrix based on its $U^TU + U^TR + R^TU$-decomposition*, Numer. Linear Algebra Appl., 5 (1998), pp. 483–509.

[14] S. A. Kharchenko, L. Yu. Kolotilina, A. A. Nikishin, and A. Yu. Yeremin. *A robust AINV-type method for constructing sparse approximate inverse preconditioners in factored form*, Numer. Linear Algebra Appl., 8 (2001), pp. 165–179.

[15] C. Keller, N. I. M. Gould, and A. J. Wathen, *Constraint preconditioning for indefinite linear systems*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1300–1317.

[16] P. Läuchli, *Jordan-Elimination und Ausgleichung nach kleinsten Quadraten*, Numer. Math., 3 (1961), pp. 226–240.

[17] National Institute of Standards, *Matrix Market*, available online at `http://math.nist.gov/MatrixMarket`.

[18] A. T. Papadopoulos, I. S. Duff, and A. J. Wathen, *Incomplete Orthogonal Factorization Methods Using Givens Rotations II: Implementation and Results*, Report no. 02/07, Oxford University Computing Laboratory, Oxford, England, 2002.

[19] I. Perugia and V. Simoncini, *Block-diagonal and indefinite symmetric preconditioners for mixed finite element formulations*, Numer. Linear Algebra Appl., 7 (2000), pp. 585–616.

[20] Y. Saad, *Preconditioning techniques for nonsymmetric and indefinite linear systems*, J. Comput. Appl. Math., 24 (1988), pp. 89–105.

[21] Y. Saad and M. Sosonkina, *Enhanced Preconditioners for Large Sparse Least Squares Problems*, Report UMSI-2001-1, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, MN, 2001.

[22] M. A. Saunders, *Sparse least squares problems by conjugate gradients: a comparison of preconditioning methods*, in *Proceedings of Computer Science and Statistics: Twelfth Annual Conference on the Interface*, Waterloo, Canada, 1979.

[23] M. Tismenetsky, *A new preconditioning technique for solving large sparse linear systems*, Linear Algebra Appl., 154-156 (1991), pp. 331–353.

[24] X. Wang, *Incomplete Factorization Preconditioning for Linear Least Squares Problems*, Ph.D. thesis, Department of Computer Science, University of Illinois at Urbana–Champaign, 1994.

[25] X. Wang, K. A. Gallivan, and R. Bramley, *CIMGS: An incomplete orthogonal factorization preconditioner*, SIAM J. Sci. Comput., 18 (1997), pp. 516–536.

[26] S.-L. Zhang, K. Nakata, and M. Kojima, *Incomplete orthogonalization preconditioners for solving large and dense linear systems which arise from semidefinite programming*, Appl. Numer. Math., 41 (2002), pp. 235–245.

[27] Z. Zlatev, *Computational Methods for General Sparse Matrices*, Kluwer Academic Publishers, Dordrecht/Boston/London, 1991.

[28] Z. Zlatev and H. B. Nielsen, *Solving large and sparse linear least-squares problems by conjugate gradient algorithms*, Comput. Math. Appl., 15 (1988), pp. 185–202.