NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
Numer. Linear Algebra Appl. 2001; **99**:1–20 *Prepared using* nlaauth.cls *[Version: 2000/03/22 v1.0]*

A robust incomplete factorization preconditioner for positive definite matrices

Michele Benzi[1,*] and Miroslav Tůma[2]

[1] *Department of Mathematics and Computer Science, Emory University, Atlanta, GA 30322, U.S.A.*
[2] *Institute of Computer Science, Academy of Sciences of the Czech Republic, 182 07 Prague 8, CZ*

SUMMARY

We describe a novel technique for computing a sparse incomplete factorization of a general symmetric positive definite matrix A. The factorization is not based on the Cholesky algorithm (or Gaussian elimination), but on A–orthogonalization. Thus, the incomplete factorization always exists and can be computed without any diagonal modification. When used in conjunction with the conjugate gradient algorithm, the new preconditioner results in a reliable solver for highly ill-conditioned linear systems. Comparisons with other incomplete factorization techniques using challenging linear systems from structural analysis and solid mechanics problems are presented.

KEY WORDS: sparse linear systems, positive definite matrices, preconditioned conjugate gradients, incomplete factorization, A–orthogonalization, SAINV

1. INTRODUCTION

We consider the solution of linear systems of the form

$$Ax = b \tag{1}$$

where A is a large, sparse symmetric positive definite (SPD) matrix and b a given right-hand side, using preconditioned conjugate gradients (PCG). We are particularly interested in finding reliable preconditioners for the poorly conditioned linear systems that arise from the finite element method (FEM) applied to problems in structural and solid mechanics.

We focus our attention on incomplete factorization preconditioners. These methods are quite effective but special care must be taken in order to avoid breakdowns, due to the occurrence of nonpositive pivots, during the incomplete factorization process. After reviewing previous work in section 2, we describe in section 3 a safe and reliable way of computing an approximate factorization $A \approx LDL^T$ with L unit lower triangular and D diagonal. Unlike the usual Cholesky–based approaches, this technique is based on an inherently stable

*Correspondence to: Michele Benzi, Department of Mathematics and Computer Science, Emory University, NDB Suite 100, Atlanta, GA 30322, USA (e-mail: benzi@mathcs.emory.edu).

Received ***DRAFT***
Revised ***DRAFT***
Accepted ***DRAFT***

A–orthogonalization process and is therefore applicable to general SPD matrices. This A–orthogonalization process has been previously used in [1] and [2] to construct robust sparse approximate inverse preconditioners in factored form; here we show that the same algorithm can be used to reliably compute an approximate factorization of A, with only a slight increase in terms of storage.

In section 4 we present the results of numerical experiments involving difficult problems from solid and structural mechanics. We compare our algorithm with several existing methods and conclude that the new technique is indeed reliable and has certain advantages relative to other robust incomplete factorization methods. Finally, section 5 concludes the paper with a summary of our findings and indications for future work.

2. THE SEARCH FOR A ROBUST INCOMPLETE FACTORIZATION

Preconditioning techniques based on incomplete factorization of the coefficient matrix A are among the best known and most popular methods for solving (1). However, the existence of an incomplete factorization for a general SPD matrix is a delicate issue which must be addressed if one wishes to design reliable preconditioners. As is well known, the existence of an incomplete factorization $A \approx LL^T$ (or $A \approx LDL^T$ in the square root-free case) has been established for certain classes of matrices. Already in [3] the existence of the incomplete Cholesky factorization was proved, for arbitrary choices of the sparsity pattern, for the class of M-matrices. This existence result was extended shortly thereafter to a somewhat larger class (that of H-matrices with positive diagonal entries) by several authors [4, 5, 6]. See [7] and [8] for additional information and references. On the other hand, an incomplete factorization can fail for a general SPD matrix due to the occurrence of nonpositive pivots. This is commonly referred to as *pivot breakdown*. Matrices arising from FEM modelling of solids and structures are usually not H-matrices, and failures due to pivot breakdowns are common.

Researchers have been aware of this problem since the early days of incomplete factorization preconditioning, and various remedies have been proposed to deal with it. Broadly speaking, and in increasing order of complexity, these remedies are:

- Increase the diagonal dominance of A by diagonal shifts (either locally or globally, before or during the factorization);
- Replace A with a "nearby" M-matrix $\hat{A}$, compute an incomplete factorization of $\hat{A} \approx LL^T$ and use that as a preconditioner for $Ax = b$;
- Do not modify A but recast the factorization in a form that avoids breakdowns.

We note in passing that some of these techniques are also applicable if A is nonsymmetric but positive definite (that is, $x^T Ax > 0$ if $x \neq 0$) or if A is indefinite with a few negative eigenvalues and an approximate factorization of A of the form $A \approx LL^T$ is sought.

The simplest fix is to replace a negative or zero pivot at step i with some (positive) quantity d_i. This dynamic, local diagonal shift strategy was suggested by Kershaw [9]. It was motivated by the hope that if only a few of the pivots are unstable (i.e., nonpositive), the resulting factorization might still yield a satisfactory preconditioner. The choice of a replacement value for a nonpositive pivot is no simple matter. If d_i is chosen too large, the preconditioner will be inaccurate (i.e., $||A - LL^T||_F$ is large); if d_i is chosen too small, the triangular solves with L

and L^T can become unstable (i.e., $||I - L^{-1}AL^{-T}||_F$ is large). Some heuristics for choosing d_i can be found in [9, 10] and in the optimization literature [11, 12]; see also [13]. Unfortunately, it is frequently the case that even a handful of pivot shifts will result in a poor preconditioner, regardless of the shift strategy used. The problem is that if a nonpositive pivot occurs, the loss of information about the true factors due to dropping has already been so great that no "local" trick can succeed in recovering a good preconditioner—it's too late.†

A more effective strategy was suggested by Manteuffel in [4]. If an incomplete factorization of A fails due to pivot breakdown, a *global* diagonal shift is applied to A prior to reattempting the incomplete factorization. That is, the incomplete factorization is carried out on $\hat{A} = A + \alpha\,\mathrm{diag}(A)$ where $\alpha > 0$ and $\mathrm{diag}(A)$ denotes the diagonal part of A. If this fails, α is increased and the process is repeated until an incomplete factorization is successfully computed.

Clearly, there exists a value α^* for which the incomplete factorization of $\hat{A}$ is guaranteed to exist: for instance, one can take the smallest α that makes $\hat{A} = A + \alpha\,\mathrm{diag}(A)$ diagonally dominant. Because diagonally dominant matrices are H-matrices, the incomplete factorization of $\hat{A}$ exists. However, the best results are usually obtained for values of α that are much smaller than α^*, and larger than the smallest α for which $\hat{A}$ admits an incomplete factorization. The obvious disadvantage of this approach is the difficulty in picking a "good" value of α. Because α is chosen on the basis of a trial-and-error strategy, this approach can be quite expensive.

A popular technique in the structural engineering community is the robust incomplete Cholesky factorization developed by Ajiz and Jennings in [14]; see also [6]. This method is based on the idea of *stabilized cancellation*. Whenever an element l_{ij} of the incomplete Cholesky factor L is dropped, $|l_{ij}|$ is added to the corresponding diagonal entries a_{ii} and a_{jj} of A. In some cases it is desirable to multiply $|l_{ij}|$ by a scaling factor before adding it to the diagonal, so as to ensure that the diagonal modification is proportional to the size of the diagonal entry being modified; see [14, 15].

In this method, the incomplete Cholesky factor L is the *exact* Cholesky factor of a perturbed matrix $\hat{A} = A + C$ where C, the *cancellation matrix*, is positive semidefinite. Hence, no breakdown can occur. Moreover, the splitting $A = LL^T - C = \hat{A} - C$ is convergent, in the sense that $\rho(I - \hat{A}^{-1}A) < 1$, where $\rho(\cdot)$ denotes the spectral radius of a matrix. Thus, the triangular solves with the incomplete factor L and its transpose cannot become unstable.

Although reliable, this strategy may result in preconditioners of low quality, generally inferior to a standard incomplete Cholesky factorization (with comparable amount of fill-in) whenever the latter exists without breakdowns. Indeed, if many fill-ins are dropped from the factors, large diagonal modifications are performed, reducing the accuracy of the preconditioner.

A method close to the Ajiz–Jennings technique has been proposed by Eijkhout [16]. His method is guaranteed to produce positive pivots for any matrix with positive diagonal entries. Numerical tests in [16] indicate this is a robust method; however, like the Ajiz–Jennings technique, it can yield poor convergence rates compared to other incomplete factorization methods, especially on hard problems.

In the Ajiz–Jennings and Eijkhout approaches the modification of diagonal entries is done *dynamically*, as dropping occurs in the course of the incomplete factorization process. Another approach, known as *diagonally compensated reduction* of positive off-diagonal entries, consists

†This point is convincingly argued by Lin and Moré in [13].

of modifying the matrix A *before* computing an incomplete factorization. In the simplest variant of this method [17], positive off-diagonal entries are set to zero and their original values are added to the corresponding diagonal entries. The resulting matrix satisfies $\hat{A} = A + C$ with C positive semidefinite, and has non-positive off-diagonal entries. Hence, it is a Stieltjes matrix (symmetric M-matrix) and an incomplete factorization of $\hat{A}$ can be computed without breakdowns; the resulting preconditioner $M = LL^T$ is then applied to the original linear system $Ax = b$.

Preconditioners based on diagonally compensated reduction give fairly good results on moderately ill-conditioned problems, such as solid elasticity problems [18] and diffusion problems posed on highly distorted meshes [1], but they are ineffective for more difficult problems arising in the finite element analysis of thin shells and other structures; see [15, 19, 1].

A more sophisticated approach has been proposed by Tismenetsky [20], with significant improvements and theoretical foundations supplied by Kaporin in an important paper [21]; see also [22]. Tismenetsky's method was originally described in terms of rank-one updates to the active submatrix and is applicable to nonsymmetric matrices. However, to show that in the SPD case the method cannot suffer from pivot breakdowns, it is preferable to interpret it as a bordering scheme for the (incomplete) Cholesky factorization. In this framework, the essence of Tismenetsky's method is the following. Let A_{k+1} denote the leading principal submatrix of A of order $k+1$. At step $k+1$ the exact decomposition is:

$$A_{k+1} = \begin{bmatrix} A_k & v_k \\ v_k^T & \alpha_{k+1} \end{bmatrix} = \begin{bmatrix} L_k & 0 \\ y_k^T & \delta_{k+1} \end{bmatrix} \begin{bmatrix} L_k^T & y_k \\ 0 & \delta_{k+1} \end{bmatrix},$$

where

$$A_k = L_k L_k^T, \quad y_k = L_k^{-1} v_k, \quad \delta_{k+1} = \sqrt{\alpha_{k+1} - y_k^T y_k}.$$

When dropping is applied to the y_k vector, an incomplete Cholesky factorization is obtained (a special case of Chow and Saad's ILUS preconditioner [23]). A breakdown occurs if $y_k^T y_k \geq \alpha_{k+1}$. When Tismenetsky's strategy is applied to SPD matrices, the pivot δ_{k+1} can be shown to be given by the expression

$$\delta_{k+1} = \sqrt{\alpha_{k+1} - 2y_k^T y_k + y_k^T L_k^{-1} A_k L_k^{-T} y_k}, \tag{2}$$

which reduces to $\delta_{k+1} = \sqrt{\alpha_{k+1} - y_k^T y_k}$ when there is no dropping. However, breakdowns are now impossible even in the presence of dropping, since the expression under square root in (2) is

$$\begin{bmatrix} y_k^T L_k^{-1} & -1 \end{bmatrix} \begin{bmatrix} A_k & v_k \\ v_k^T & \alpha_{k+1} \end{bmatrix} \begin{bmatrix} L_k^{-T} y_k \\ -1 \end{bmatrix} > 0.$$

After the pivot δ_{k+1} has been computed, a drop tolerance can be applied to remove unwanted fill-ins from y_k. This strategy produces high-quality preconditioners which frequently outperform all other known incomplete factorization techniques in terms of convergence rates; see the results in [20, 21]. Its main disadvantage is that it is expensive in terms of preconditioner construction costs and intermediate storage requirements. While high set-up costs in terms of CPU time may be acceptable in situations where the preconditioner can be reused for several solves, the high intermediate storage requirement could be a serious drawback in applications involving large matrices.

Without going into details (for which the reader is referred to the original papers), these high costs stem from the need to retain a high fraction of fill-ins in the course of the factorization.

These fill-ins are eventually dropped and they are not used in the preconditioned iteration phase, but they must be temporarily stored as they are needed in order to preserve positive definiteness, i.e., to avoid pivot breakdowns. Intermediate storage can be reduced to more acceptable levels by combining Tismenetsky's idea with the use of two drop tolerances. Briefly, some of the (smaller) intermediate fill-ins that would have been retained are dropped, for example by applying a second drop tolerance test to the intermediate fill-ins. The larger this second tolerance, the closer the method becomes to the Ajiz–Jennings technique. Some form of diagonal compensation (in the style of Ajiz–Jennings) is then needed in order to avoid pivot breakdowns. Variants of this approach have been tested in [20, 22, 21]. In particular, [21] uses a second drop tolerance that is a simple function of the first one, so that the user only needs to specify one tolerance. Generally speaking, these methods display convergence rates which are comparable with those of the "full" Tismenetsky method while at the same time requiring much less storage and set-up time. However, these techniques are still quite expensive in time and space.

In summary, it can be said that existing solutions to the robustness (existence) problem for incomplete factorization preconditioners for general SPD matrices tend to fall into one of two camps: simple and inexpensive fixes that result in low quality preconditioners in terms of rates of convergence, or sophisticated, expensive strategies that yield high-quality preconditioners. In this paper we introduce a preconditioner that strikes a compromise between these two extremes.

3. INCOMPLETE FACTORIZATION BASED ON A–ORTHOGONALIZATION

All the incomplete factorizations surveyed in the previous section were based on the well known Cholesky algorithm or, possibly, on its root-free variant (LDL^T factorization). However, this is not the only way to compute a triangular factorization of A. Here we show how the factorization $A = LDL^T$ can be obtained by means of an A–orthogonalization process applied to the unit basis vectors $e_1, e_2, \dots, e_n$. This is simply the Gram–Schmidt process with respect to the inner product generated by the SPD matrix A. This algorithm was first described in [24].

The observation that A–orthogonalization of the unit basis vectors is closely related to (symmetric) Gaussian elimination is not new: indeed, it can be found already in the celebrated paper [25] by Hestenes and Stiefel on the conjugate gradient method (see pp. 425–427).[‡] Because this algorithm costs twice as much as the Cholesky factorization in the dense case, A–orthogonalization is never used to factor matrices. However, as noted in [26], A–orthogonalization also produces the inverse factorization $A^{-1} = ZD^{-1}Z^T$ (with Z unit upper triangular and D diagonal) and this fact has been exploited to construct factored sparse approximate inverse preconditioners; see [26, 1, 2]. Because A–orthogonalization, even when performed incompletely, is not subject to pivot breakdowns, these preconditioners are reliable [1, 2]. However, they are often less effective than incomplete Cholesky preconditioning at reducing the number of PCG iterations and their main interest stems from the fact that they are easily implemented on parallel computers.

[‡]Curiously, this fact was not noticed in [24].

Here, for the first time, we investigate the use of A–orthogonalization as a way to compute an incomplete factorization of A (rather than A^{-1}). The resulting incomplete factorization algorithm can be expected to be computationally more expensive than standard incomplete Cholesky; however, it cannot break down and we obtain a reliable algorithm.

We begin by recalling the A–orthogonalization process. Initially, $z_j = e_j$ $(1 \le j \le n)$. Then the (right-looking) algorithm consists of the following nested loop:

$$z_i \leftarrow z_i - \frac{\langle Az_j, z_i\rangle}{\langle Az_j, z_j\rangle} z_j \tag{3}$$

where $j = 1, 2, \ldots, n$ and $i = j+1, \ldots, n$. Upon completion, letting $Z = [z_1, z_2, \ldots, z_n]$ and $D = \mathrm{diag}(d_1, d_2, \ldots, d_n)$ with $d_j = \langle Az_j, z_j\rangle$, we obtain the inverse upper-lower factorization $A^{-1} = ZD^{-1}Z^T$. A left-looking variant of this scheme also exists.

To obtain a sparse preconditioner, a dropping rule is applied to the z_i vectors after each update step; see [1]. Of course, in order to have an efficient algorithm, the loop has to be carefully implemented by exploiting sparsity in A and in the z_i vectors. In particular, most of the inner products $\langle Az_j, z_i\rangle$ are structurally zero and they need not be computed, and the corresponding update (3) can be skipped. In this way, it is generally possible to compute a sparse approximate inverse preconditioner $A^{-1} \approx ZD^{-1}Z^T$ quite cheaply [1]. Because the pivots d_j are computed as $d_j = \langle Az_j, z_j\rangle$ with $z_j \neq 0$ (since the j-th entry of z_j is equal to 1), this preconditioner, which is usually referred to as SAINV, is well-defined for a general SPD matrix. Variants of SAINV have been shown to be reliable in solving highly ill-conditioned linear systems [1, 2, 27].

Consider now the exact algorithm (with no dropping) and write $A = LDL^T$ with L unit lower triangular and D diagonal. Observe that L in the LDL^T factorization of A and the inverse factor Z satisfy

$$AZ = LD, \quad \text{or} \quad L = AZD^{-1}$$

where D is the diagonal matrix containing the pivots. This easily follows from

$$Z^TAZ = D \quad \text{and} \quad Z^T = L^{-1}.$$

Recall that $d_j = \langle Az_j, z_j\rangle$; by equating corresponding entries of AZD^{-1} and $L = [l_{ij}]$ we find that

$$l_{ij} = \frac{\langle Az_j, z_i\rangle}{\langle Az_j, z_j\rangle}, \quad i \ge j. \tag{4}$$

Hence, the L factor of A can be obtained as a by-product of the A–orthogonalization process, *at no extra cost*. We mention that the relationship (4), which can be found in [25], has also been independently noted in [28].

In the implementation of SAINV, the quantities l_{ij} in (4) are the *multipliers* that are used in updating the columns of Z (see (3)). Once the update is computed, they are no longer needed and are discarded. To obtain an incomplete factorization of A, we do just the opposite; we save the multipliers l_{ij}, and discard the column vectors z_j as soon as they have been computed and operated with. Hence, the incomplete L factor is computed by columns; these columns can be stored in place of the z_j vectors, with minimal modifications to the code. Here, we are assuming that the right-looking form of SAINV is being used. If the left-looking one is being used, then L would be computed by rows.

The right-looking algorithm requires some intermediate storage for the sparse z_i vectors while they are needed to compute the multipliers (4). At step j of the algorithm $(1 \le j \le n)$

we need to store, besides the j computed columns of the incomplete factor L, the remaining $n-j$ columns $z_{j+1}, \ldots, z_n$ of Z. Notice that these sparse vectors can have nonzero entries only within the first j positions, besides the 1 in position k of 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 L and 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 A–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 L. In practice, we found that this is often an overestimate, unless a very small drop tolerance is used (which is not practical anyway). This is due to the fact that the bandwidth of A and Z can be reduced by suitable permutations, and therefore the nonzeros are not uniformly distributed. Hence, with a careful implementation and using suitable dynamic data structures, the proposed algorithm incurs negligible intermediate storage requirements. We note that the overhead would be much higher for the left-looking variant.

The incomplete factorization based on $A-$orthogonalization needs two drop tolerances; one for the SAINV (incomplete A–orthogonalization) process, to be applied to the z vectors, and a second one to be applied to the entries of L. Note that the latter is simply post-filtration: once a column of L has been computed, it does not enter the computation of the remaining ones. In practice we found that this post-filtration can be helpful. Removing small entries from the columns of the incomplete L factor leads to some degradation of the convergence rate, but this is often compensated by the savings in computational work obtained from having a more sparse L factor. Moreover, since the post-filtration is performed on columns of L as they are computed, considerable reductions in storage requirements can be expected. The drawback is the need to introduce a second drop tolerance; we circumvent this problem by using the same value of the drop tolerance for both the SAINV process and the entries of the incomplete factor L. If necessary, it is easy to incorporate a dual threshold strategy (analogous to the one used in ILUT [29]) in order to impose an *a priori* limit on the storage needed for L.

4. NUMERICAL EXPERIMENTS

In this section we present the results of numerical experiments on difficult linear systems resulting from finite element modelling of problems in solid and structural mechanics. The primary goal of these experiments is to assess the reliability of our new preconditioner when it is used to solve highly ill-conditioned problems from realistic applications. In addition, we would also like to make comparisons with some other robust preconditioning methods. However, the reader should be cautioned against inferring too much from such a (necessarily) limited empirical study. Note that no tuning of the drop tolerance for optimal performance was attempted.

We begin with a fairly small problem which, while not being an H-matrix, does not cause difficulties for standard incomplete Cholesky techniques. This is a linear system arising in the finite element analysis of shells; the matrix is S2RMQ4M1 from the CYLSHELL collection in the Matrix Market [30]. It corresponds to a cylindrical shell of characteristic thickness $t/R = 10^{-2}$ discretized with a 30×30 uniform quadrilateral mesh. The stiffness matrix A has order $n = 5,489$ and the lower triangular part of A contains $nnz = 143,300$ nonzero entries. The matrix has a condition number $\kappa(A) \approx 1.2 \cdot 10^8$, and is one of the easier ones to solve in

Table I. Comparison of preconditioners on problem S2RMQ4M1.

Precond.	Density	Storage	Its.	P-time	It-time	Tot-time
JCG	–	–	1609	–	11.2	11.2
ICT	1.08	1.08	89	0.41	2.40	2.81
RIC1	1.04	1.04	241	0.35	5.90	6.25
RIC2S	1.05	4.06	95	2.05	2.37	4.42
RIF	1.05	1.06	166	1.11	4.27	5.38
RIF_p	0.35	0.36	291	1.11	3.50	4.61
CHOL	4.70	4.70	1	7.80	0.10	7.90

the CYLSHELL set. For the right-hand side b, a physically realistic load vector was used. See [31] for further details.

As a baseline method, we report results for Jacobi (diagonal scaling) preconditioning, denoted "JCG" in the table. The other preconditioners tested are:

- a standard, dual threshold incomplete Cholesky factorization (denoted "ICT" in the table, which happens to be well-defined in this case;
- the Ajiz–Jennings robust incomplete Cholesky factorization (denoted "RIC1");
- Kaporin's second-order stabilized robust incomplete Cholesky factorization (denoted "RIC2S");[§]
- two variants of our own A–orthogonalization based robust incomplete factorization, one without post-filtration (denoted "RIF"), the other with it (denoted "RIF_p"). The post-filtration parameter was the same as the drop tolerance for the underlying SAINV process.

For each preconditioner we report the number of nonzeros in the incomplete L factor divided by the number of nonzeros in the lower triangular part of A (under "Density"); the total amount of storage needed, which consists of the intermediate storage plus the final storage for the L factor (but not for A), normalized by the number of nonzeros in the lower triangular part of A (under "Storage"); the number of PCG iterations needed to achieve $||b - Ax_k||_2 < 10^{-8}||b||_2$, the initial guess being $x_0 = 0$ ("Its."); the time for constructing the preconditioner, in seconds ("P-time"); the time for the iterative solution phase ("It-time"); and the total solution time ("Tot-time"). The computations have been performed using one processor of a four-processor SUN Enterprise 450 (400MHz, 1.2Gb RAM). Prior to computing the incomplete factorization preconditioners, the coefficient matrix was symmetrically scaled by its diagonal and a reverse Cuthill–McKee ordering [32] was applied. The parameters were chosen so as to obtain preconditioners with similar density (≈ 1), except for RIF_p which, due to post-filtration, is much sparser. Finally, we include results for a simple complete Cholesky factorization (without symbolic analyse phase or supernode use) with multiple minimum degree ordering of A [33].

[§]The description of this algorithm in [21] should be amended in two points on page 500. On line 6 from top, $\sqrt{d}_j$ should be replaced by $\sqrt{d_i d_j}$. On line 22, the *else* statement should be replaced by *else if* $v_j \neq 0$.

For this problem, standard incomplete Cholesky preconditioning gives the best results overall; however, this method is unreliable as it fails (due to pivot breakdowns) on numerous problems from the CYLSHELL set [31, 27]. The robust Ajiz–Jennings preconditioner has low set-up and storage costs, but it is clearly inferior to standard incomplete Cholesky in terms of convergence rate. Kaporin's robust method displays rapid convergence (resulting in the smallest It-time of all tested methods), but it is the most expensive to compute and incurs high intermediate storage costs. It can be seen that for this matrix, the total storage needed by RIC2S comes close to that of the *complete* Cholesky factorization; note, however, the different orderings used. (RIC2S gave poor results with minimum degree, and we ran out of memory when we tried to compute a complete Cholesky factorization with the reverse Cuthill–McKee ordering.) RIF preconditioning has reasonably low set-up cost, negligible intermediate storage requirements, and results in faster convergence than the Ajiz–Jennings preconditioner; however, it requires more iterations than ICT and RIC2S. Applying post-filtration results in a preconditioner RIF_p with very low storage needs, as well as smaller solution time. The number of iterations is now higher than for RIC1, but each iteration is so much cheaper than the total solution time is smaller. This method is competitive with Kaporin's RIC2S in terms of computational effort, and it needs a fraction (less than a tenth) of the space to generate the preconditioner.

Because post-filtration is so effective with RIF, it is tempting to apply it to the other preconditioners as well. This turned out to be quite useful in improving the performance of Ajiz and Jenning's RIC preconditioner. Using $\tau_2 = 10 \cdot \tau_1$ where τ_1 denotes the drop tolerance for the incomplete Cholesky process and τ_2 the post-filtration tolerance, the number of iterations for RIC increases to 358 but the total solution times are decreased to 5.08 seconds, with a more than threefold reduction in the size of the incomplete factor.

On the other hand, for this problem post-filtration had a strong negative impact on both ICT and RIC2S. Using the same post-filtration tolerance as was used for RIC resulted again in an incomplete factor more than three times smaller (density ≈ 0.33). However, the resulting loss in convergence rate was so pronounced that the total solution time was almost doubled for ICT (387 iterations, 5.53 seconds) and considerably higher for RIC2S (380 its., 7.35 seconds). At the same time, the savings in space for generating the preconditioner are negligible, particularly for RIC2S. Thus, it appears that at least for this problem, post-filtration is harmful to high-quality preconditioners, but helpful for preconditioners of lesser quality.

In the following we show results of numerical experiments involving larger problems. In Table II we provide some basic information on the test problems used. None of these matrices is an H-matrix, and all of them are quite ill-conditioned. For each matrix we provide the problem size n, the number nnz of nonzeros in the lower triangular part, the application from which the matrix originates, and the source.

In Tables III–X we present, for each test problem, the results obtained with various preconditioners. For each method we provide preconditioner size and storage information, and the number of preconditioned conjugate gradient iterations (and relative timings, in seconds) to solve the problem to a relative residual precision of 10^{-8}, allowing a maximum of 10,000 iterations. A physical right-hand side (load vector) was used whenever available; otherwise we used an artificial right-hand side computed as $b = Ae$, where e is the vector of all ones. The initial guess was always the zero vector. The matrices were diagonally scaled and reordered using reverse Cuthill–McKee. This ordering was found to give the best results for the incomplete Cholesky-type preconditioners. For the preconditioners based on

Table II. Test problem information.

Matrix	n	nnz	Application	Source
BCSSTK17	10,974	219,812	Pressure vessel	Matrix Market [30]
BCSSTK18	11,948	80,519	Power plant	”
BCSSTK25	15,439	133,840	Skyscraper	”
CT20STIF	52,329	1,375,396	Engine block	U. of Florida [34]
NASASRB	54,870	1,366,097	Shuttle rocket booster	”
TUBE1-2	21,498	459,277	Thin shell	R. Kouhia [35]
SMT	25,710	1,889,447	Surface mounted transistor	”
ENGINE	143,571	2,424,822	Engine head	”

Table III. Test results for matrix BCSSTK17.

Precond.	Density	Storage	Its.	P-time	It-time	Tot-time
JCG	–	–	2608	–	47.7	47.7
RIC1	1.20	1.20	245	0.65	13.3	13.9
RIC2S	1.10	4.49	179	3.41	9.52	12.9
RIF	1.29	1.30	262	2.79	15.3	18.1
RIF$_p$	0.50	0.51	348	2.79	12.9	15.7
CHOL	4.58	4.58	1	9.17	0.14	9.31

Table IV. Test results for matrix BCSSTK18.

Precond.	Density	Storage	Its.	P-time	It-time	Tot-time
JCG	–	–	1008	–	6.48	6.48
RIC1	1.20	1.20	101	0.13	2.05	2.18
RIC2S	1.17	4.00	43	0.95	0.86	1.81
RIF	1.18	1.23	78	0.81	1.42	2.23
RIF$_p$	0.61	0.66	123	0.81	1.65	2.46
CHOL	8.23	8.23	1	8.92	0.08	9.00

A–orthogonalization, RIF and RIF$_p$, minimum degree ordering also gave good results, in agreement with the findings in [36, 37, 1]. The differences were almost always small, and the results in the tables are for reverse Cuthill–McKee ordering. The only exception is matrix NASASRB, for which minimum degree resulted in a 40% reduction in total solution times for RIF. Standard drop tolerance preconditioners fail for some combination of parameters on all the problems considered here, except for ENGINE. They can be used with diagonally compensated reduction [17], but the resulting approach is generally not competitive with the other methods considered here; hence, we do not report any of the corresponding results.

On the basis of these results, we can make the following general observations. The reliability of the preconditioners RIC1, RIC2S and RIF was confirmed in these experiments. None of

Table V. Test results for matrix BCSSTK25.

Precond.	Density	Storage	Its.	P-time	It-time	Tot-time
JCG	–	–	9297	–	92.0	92.0
RIC1	1.39	1.39	900	0.34	34.0	34.3
RIC2S	1.28	7.39	526	2.29	18.7	21.6
RIF	1.37	1.38	589	1.83	23.9	25.5
RIF_p	0.58	0.59	861	1.83	21.4	23.2
CHOL	10.6	10.6	1	18.4	0.22	18.6

Table VI. Test results for matrix CT20STIF.

Precond.	Density	Storage	Its.	P-time	It-time	Tot-time
JCG	–	–	>10000	–	>1635.	>1635.
RIC1	1.06	1.06	2731	4.40	1008.	1013.
RIC2S	1.04	5.67	2036	43.0	744.	787.
RIF	1.10	1.11	2284	13.4	861.	874.
RIF_p	0.37	0.38	6066	13.4	1545.	1559.
CHOL	8.06	8.06	1	647.	1.67	649.

Table VII. Test results for matrix NASASRB.

Precond.	Density	Storage	Its.	P-time	It-time	Tot-time
JCG	–	–	>10000	–	>1790.	>1790.
RIC1	1.00	1.00	1930	3.80	719.	723.
RIC2S	1.07	5.38	560	38.3	218.	256.
RIF	1.02	1.26	1829	15.0	707.	722.
RIF_p	0.30	0.54	3246	15.0	818.	833.
CHOL	8.71	8.71	1	347.	1.82	349.

Table VIII. Test results for matrix TUBE1-2.

Precond.	Density	Storage	Its.	P-time	It-time	Tot-time
JCG	–	–	>10000	–	>608.	>608.
RIC1	1.03	1.03	1573	1.19	194.	195.
RIC2S	1.11	7.93	1058	16.6	136.	153.
RIF	1.05	1.05	1053	5.01	126.	131.
RIF_p	0.36	0.36	1619	5.01	128.	133.
CHOL	5.00	5.00	1	33.7	0.37	34.1

Table IX. Test results for matrix SMT.

Precond.	Density	Storage	Its.	P-time	It-time	Tot-time
JCG	–	–	2031	–	442.	442.
RIC1	1.01	1.01	325	12.8	150.	163.
RIC2S	1.03	6.65	63	144.	30.0	174.
RIF	0.99	1.00	159	25.3	76.6	102.
RIF_p	0.25	0.26	227	25.3	65.1	90.4
CHOL	7.68	7.68	1	1135.	2.20	1137.

Table X. Test results for matrix ENGINE.

Precond.	Density	Storage	Its.	P-time	It-time	Tot-time
JCG	–	–	3325	–	1137.	1137.
RIC1	1.10	1.10	645	5.55	469.	475.
RIC2S	1.10	4.16	210	37.0	153.	190.
RIF	1.11	1.17	515	10.3	373.	392.
RIF_p	0.27	0.33	958	10.3	452.	462.
CHOL	10.4	10.4	1	1042.	3.97	1046.

the methods failed for any of the choices of input parameters that we tried, and it was easy to tune the drop tolerance so as to have incomplete factors L with density close to 1. It should be emphasized that this is not an optimal choice, in general; it was chosen because it allows a fair comparison of the quality of the various preconditioners. Among all methods tested, Kaporin's RIC2S was clearly the best in terms of convergence rates. This powerful preconditioner consistently resulted in the smallest number of iterations, and nearly always in the best timings. Ajiz and Jennings' RIC1 was the least effective preconditioner overall (excluding JCG); our own RIF method falls somewhere in between these two. On the other hand, RIC1 was generally the least expensive method in terms of set-up, while RIC2S was the most expensive one; again, RIF is in between the two. In one case (problem SMT), high set-up costs for RIC2S caused it to be slower than the other preconditioners in terms of total timings. However, if the cost of the preconditioner set-up can be amortized over repeated solves, the superior convergence properties of RIC2S make it the best choice.

In terms of totale storage needs, RIC2S is by far the most demanding method. It needs an amount of storage often comparable to that required by a *complete* Cholesky decomposition with a suitable ordering (minimum degree). This is the only drawback of RIC2S, which is an otherwise excellent preconditioner. Storage demands are low for RIC1 and RIF; in particular, the overhead for the latter method is negligible. We mention that intermediate storage requirements were higher for RIF with minimum degree orderings, often close to the theoretical estimate of 25% of the final size of L; see the results for NASASRB in Table VII.

Post-filtration results in a RIF_p preconditioner with very low memory requirements. However, the deterioration in terms of convergence rate is noticeable, and in terms of solution times the advantages of post-filtration are usually small or non-existent. Note the especially poor performance of RIF_p on problem CT20STIF. Likewise, the beneficial effects of post-filtration on RIC1 that were observed for problem S2RMQ4M1 were not replicated in the experiments with the larger problems; indeed, in a majority of cases we found that post-filtration had an adverse effect on solution times. Hence, the use of post-filtration cannot be recommended in general.

Finally, we observe that in several cases the fastest solution method was actually the direct solver. See in particular the results for problem TUBE1-2, one of the most ill-conditioned ones in our test set. For such problems it is difficult to find a preconditioner that can compete with a good sparse direct solver.

5. CONCLUSIONS AND FUTURE WORK

We have introduced a new incomplete factorization preconditioner based on A–orthogonalization. Numerical experiments on difficult linear systems from solid and structural mechanics suggest that the new technique is reliable and can be competitive with existing robust incomplete factorization methods. In particular, the new method appears to be generally superior to the popular Ajiz–Jennings incomplete Cholesky preconditioner. Owing to its negligible intermediate storage requirements, it should be especially useful in situations where only limited storage is available. On the other hand, if storage is not an issue, then Kaporin's robust incomplete factorization method (RIC2S) is a better choice, due to its impressive convergence properties.

Future work should include the extension of the method to the case of nonsymmetric, positive definite matrices and to preconditioning the CGNR (or CGNE) method for solving large, sparse least-squares problems.

Concerning the first class of problems, it is clear that the preconditioner is breakdown-free, since $\langle Az_i, z_i\rangle > 0$ for all $z_i \neq 0$. The loop (3) can be used to compute an incomplete L factor; to compute an incomplete U factor, the same algorithm must be run with A^T in place of A; see [38]. Alternatively, a transpose-free, hybrid incomplete factorization–SAINV preconditioner could be obtained by executing (3) incompletely and keeping both L and Z; the corresponding preconditioner would then be $M^{-1} = ZD^{-1}L^{-1} \approx A^{-1}$, where $Z \approx U^{-1}$. Application of this hybrid preconditioner requires a forward substitution involving the unit lower triangular matrix L, a diagonal scaling operation, and a matrix–vector multiply with Z.

The solution of linear least-squares problems

$$||b - Ax||_2 = \min$$

where A is a large, sparse $m \times n$ matrix with full column rank ($n \leq m$; notice that A could be square) can be performed by implicitly applying the method of conjugate gradients to the normal equations:

$$A^T Ax = A^T b \quad \text{or} \quad AA^T y = b, \quad x = A^T y;$$

see [39]. In either case the coefficient matrix is SPD. The problem is then to design a reliable preconditioner for the normal equations. It is important that the preconditioner be computable without the need to explicitly form the product $A^T A$ (or AA^T), working with A only. Approaches based on variants of incomplete QR or LQ factorizations of A (or A^T) have been proposed by various authors; see [40, 41, 42, 43, 39, 44]. However, these methods either suffer from high intermediate costs, or they are not always guaranteed to produce a positive definite preconditioner. A reliable preconditioner with low intermediate storage requirement can be constructed by A^TA-orthogonalization. Note that the main loop (3) becomes

$$z_i \leftarrow z_i - \frac{\langle Az_j, Az_i\rangle}{\langle Az_j, Az_j\rangle} z_j \tag{5}$$

where $j = 1, 2, \ldots, n$ and $i = j+1, \ldots, n$. Hence, there is no need to form A^TA explicitly; all is needed is the sparse–sparse matrix–vector products Az_j, Az_i and the corresponding sparse–sparse inner products, which can be computed efficiently. The corresponding multipliers

$$l_{ij} = \frac{\langle Az_j, Az_i\rangle}{\langle Az_j, Az_j\rangle}, \quad i \geq j$$

are the entries of the Cholesky factor of A^TA; when algorithm (5) is performed incompletely, one obtains an incomplete factorization of A^TA, which can be used as a preconditioner for the CGNR method. Pivot breakdowns cannot occur, since

$$\langle Az_j, Az_j\rangle = \langle A^TAz_j, Az_j\rangle > 0 \quad \text{for} \quad z_j \neq 0.$$

Of course, an analogous approach for CGNE based on AA^T–orthogonalization is also possible. We plan to study the effectiveness of such preconditioning techniques for sparse least-squares problems in a future paper.

ACKNOWLEDGEMENTS

The work of the second author was supported in part by Grant Agency of the Academy of Sciences of the Czech Republic grants No. 2030801 and 1030103.

REFERENCES

1. Benzi M, Cullum JK, Tůma M. Robust approximate inverse preconditioning for the conjugate gradient method. *SIAM Journal on Scientific Computing* 2000; **22**: 1318–1332.
2. Kharchenko SA, Kolotilina LYu, Nikishin AA, Yeremin AYu. A robust AINV-type method for constructing sparse approximate inverse preconditioners in factored form. *Numerical Linear Algebra with Applications* 2001; **8**:165–179.
3. Meijerink JA, van der Vorst HA. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Mathematics of Computation* 1977; **31**:148–162.
4. Manteuffel T. An incomplete factorization technique for positive definite linear systems. *Mathematics of Computation* 1980; **34**:473–497.
5. Varga RS, Saff EB, Mehrmann V. Incomplete factorizations of matrices and connections with H-matrices. *SIAM Journal on Numerical Analysis* 1980; **17**:787–793.
6. Robert Y. Regular incomplete factorizations of real positive definite matrices. *Linear Algebra and its Applications*, 1982; **48**:105–117.
7. Axelsson O. *Iterative Solution Methods.* Cambridge University Press: Cambridge, 1994.
8. Eijkhout V. On the existence problem of incomplete factorisation methods. *LAPACK Working Note 144, UT-CS-99-435*, Computer Science Department, University of Tennessee, Knoxville, TN, 1999.
9. Kershaw DS. The incomplete Cholesky–conjugate gradient method for the iterative solution of systems of linear equations. *Journal of Computational Physics* 1978; **26**:43–65.
10. van der Vorst HA. Iterative solution methods for certain sparse linear systems with a non-symmetric matrix arising from PDE-problems. *Journal of Computational Physics* 1981; **44**:1–19.
11. Gill PE, Murray W, Wright MH. *Practical Optimization.* Academic Press: London, 1981.
12. Schnabel RB, Eskow E. A new modified Cholesky factorization. *SIAM Journal on Scientific and Statistical Computing*, 1990; **11**:1136–1158.
13. Lin CJ, Moré JJ. Incomplete Cholesky factorizations with limited memory. *SIAM Journal on Scientific Computing* 1999; **21**:24–45.

14. Ajiz MA, Jennings A. A robust incomplete Choleski–conjugate gradient algorithm. *International Journal for Numerical Methods in Engineering* 1984; **20**:949–966.
15. Hladík I, Reed MB, Swoboda G. Robust preconditioners for linear elasticity FEM analyses. *International Journal for Numerical Methods in Engineering* 1997; **40**: 2109–2127.
16. Eijkhout V. The 'weighted modification' incomplete factorisation method. *LAPACK Working Note 145, UT-CS-99-436*, Computer Science Department, University of Tennessee, Knoxville, TN, 1999.
17. Axelsson O, Kolotilina LYu. Diagonally compensated reduction and related preconditioning methods. *Numerical Linear Algebra with Applications* 1994; **1**: 155–177.
18. Saint-Georges P, Warzee G, Notay Y, Beauwens R. High-performance PCG solvers for FEM structural analyses. *International Journal for Numerical Methods in Engineering* 1996; **39**:1133–1160.
19. Saint-Georges P, Warzee G, Notay Y, Beauwens R. Problem-dependent preconditioners for iterative solvers in FE elastostatics. *Computers and Structures* 1999; **73**:33–43.
20. Tismenetsky M. A new preconditioning technique for solving large sparse linear systems. *Linear Algebra and its Applications* 1991; **154-156**:331–353.
21. Kaporin IE. High quality preconditioning of a general symmetric positive definite matrix based on its $U^TU + U^TR + R^TU$-decomposition. *Numerical Linear Algebra with Applications* 1998; **5**:483–509.
22. Suarjana M, Law KH. A robust incomplete factorization based on value and space constraints. *International Journal for Numerical Methods in Engineering* 1995; **38**:1703–1719.
23. Chow E, Saad Y. ILUS: An incomplete ILU preconditioner in sparse skyline format. *International Journal for Numerical Methods in Fluids* 1997; **25**:739–748.
24. Fox L, Huskey HD, Wilkinson JH. Notes on the solution of algebraic linear simultaneous equations. *Quarterly Journal of Mechanics and Applied Mathematics* 1948; **1**:149–173.
25. Hestenes MR, Stiefel E. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards* 1952; **49**: 409–436.
26. Benzi M, Meyer CD, Tůma M. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM Journal on Scientific Computing* 1996; **17**:1135–1149.
27. Benzi M, Kouhia R, Tůma M. Stabilized and block approximate inverse preconditioners for problems in solid and structural mechanics. *Computer Methods in Applied Mechanics and Engineering* 2001; to appear.
28. Bollhöfer M, Saad Y. On the relations between ILUs and factored approximate inverses. *Technical Report UMSI-2001-67*, University of Minnesota Supercomputing Institute, Minneapolis, MN, 2001.
29. Saad Y. ILUT: A dual threshold incomplete LU factorization. *Numerical Linear Algebra with Applications* 1994; **1**:387–402.
30. Matrix Market page. http://math.nist.gov/MatrixMarket [31 August 2001].
31. Benzi M, Kouhia R, Tůma M. An assessment of some preconditioning techniques in shell problems. *Communications in Numerical Methods in Engineering* 1998; **14**:897–906.
32. George A, Liu JWH. *Computer Solution of Large Sparse Positive Definite Systems.* Prentice-Hall: Englewood Cliffs, NJ, 1981.
33. Liu JWH. Modification of the minimum degree algorithm by multiple elimination. *ACM Transactions on Mathematical Software* 1985; **11**:141–153.
34. University of Florida Sparse Matrix Collection web page. http://www.cise.ufl.edu/research/sparse/matrices/ [31 August 2001].
35. Kouhia R. Sparse matrices web page. http://www.hut.fi/~kouhia/sparse.html [31 August 2001].
36. Bridson R, Tang WP. Ordering, anisotropy and factored sparse approximate inverses. *SIAM Journal on Scientific Computing*, 1999; **21**:867–882.
37. Benzi M, Tůma M. Orderings for factorized approximate inverse preconditioners. *SIAM Journal on Scientific Computing* 2000; **21**:1851–1868.
38. Benzi M, Tůma M. A sparse approximate inverse preconditioner for nonsymmetric linear systems. *SIAM Journal on Scientific Computing* 1998; **19**:968–994.
39. Björck Å. *Numerical Methods for Least Squares Problems.* SIAM: Philadelphia, 1996.
40. Jennings A, Ajiz MA. Incomplete methods for solving $A^TAx = b$. *SIAM Journal on Scientific and Statistical Computing* 1984; **5**:978–987.
41. Saad Y. Preconditioning techniques for nonsymmetric and indefinite linear systems. *Journal of Computational and Applied Mathematics* 1988; **24**:89–105.
42. Wang X. *Incomplete Factorization Preconditioning for Linear Least Squares Problems.* Ph.D. thesis, Department of Computer Science, University of Illinois at Urbana–Champaign, 1994.
43. Benzi M, Tůma M. A comparison of some preconditioning techniques for general sparse matrices. In *Iterative Methods in Linear Algebra, II*, Margenov SD, Vassilevski PS (eds). IMACS Series in Computational and Applied Mathematics, vol. 3. IMACS: New Brunswick, NJ, 1996; 191–203.
44. Bai ZZ, Duff IS, Wathen AJ. A class of incomplete orthogonal factorization methods. I: Methods and theories. *BIT* 2001; **41**:53–70.