Approximate Inverse Preconditioners for General Sparse Matrices*

Edmond Chow and Yousef Saad

Department of Computer Science, and

Minnesota Supercomputer Institute

University of Minnesota

Minneapolis, MN 55455

May 1994

Abstract

The standard Incomplete LU (ILU) preconditioners often fail for general sparse indefinite matrices because they give rise to 'unstable' factors L and U. In such cases, it may be attractive to approximate the inverse of the matrix directly. This paper focuses on approximate inverse preconditioners based on minimizing $\|I - AM\|_F$, where AM is the preconditioned matrix. An iterative descent-type method is used to approximate each column of the inverse. For this approach to be efficient, the iteration must be done in sparse mode, i.e., with 'sparse-matrix by sparse-vector' operations. Numerical dropping is applied to each column to maintain sparsity in the approximate inverse. Compared to previous methods, this is a natural way to determine the sparsity pattern of the approximate inverse. This paper discusses options such as Newton and 'global' iteration, self-preconditioning, dropping strategies, and factorized forms. The performance of the options are compared on standard problems from the Harwell-Boeing collection. Theoretical results on general approximate inverses and the convergence behavior of the algorithms are derived. Finally, some ideas and experiments with practical variations and applications are presented.

Key words. approximate inverse, preconditioning, Krylov subspace methods, threshold dropping strategies

AMS(MOS) subject classifications. 65F10, 65F35, 65F50, 65Y05

*Work supported in part by the National Science Foundation under grant NSF/CCR-9214116 and in part by a grant from NASA.

1 Introduction

The incomplete LU factorization preconditioners have been originally developed for M-matrices which arise from the discretization of very simple partial differential equations of elliptic type, usually in one variable. For the rather common situation where the matrix A is indefinite, standard ILU factorizations may face several difficulties. The best known of these difficulties arises when the algorithm to compute the ILU factorization encounters a zero pivot leading to a fatal breakdown. However, there are other problems that are just as serious. Consider an incomplete factorization of the form

$$A = LU + E \tag{1}$$

where E is the error. The preconditioned matrices associated with the different forms of preconditioning are similar to

$$L^{-1}AU^{-1} = I + L^{-1}EU^{-1}. \tag{2}$$

What is sometimes missed is the fact that the error matrix E in (1) is not as important as the 'preconditioned' error matrix $L^{-1}EU^{-1}$ shown in (2) above. When the matrix A is diagonally dominant then L and U are typically well conditioned, and the size of $L^{-1}EU^{-1}$ remains confined within reasonable limits, typically with a nice clustering of its eigenvalues around the origin. On the other hand, when the original matrix is not diagonally dominant, L^{-1} or U^{-1} may have very large norms, causing the error $L^{-1}EU^{-1}$ to be very large and thus adding large perturbations to the identity matrix. This form of instability has been studied by Elman [12] who proposed a detailed analysis of ILU and MILU preconditioners. It can be observed experimentally that ILU preconditioners can be very poor when L^{-1} or U^{-1} are large, and that this situation often occurs for indefinite problems, or problems with large nonsymmetric parts.

A possible remedy is to try to find a preconditioner that does not require solving a linear system. For example, we can precondition the original system with a matrix M which is a direct approximation to the inverse of A. Compared to ILU techniques, approximate inverse preconditioners have the additional advantage that they are easily constructed and applied in parallel. Approximate inverses are also useful in two-level preconditioners, for example, various incomplete block factorizations, and have application to improving preconditioners.

In this paper, we focus on methods of finding approximate inverses based on minimizing the Frobenius norm of the residual matrix $I - AM$, first suggested by Benson and Frederickson [5, 6]. We look for a right approximate inverse so that practical variations that use flexible right preconditioning may be used, although left approximate inverses may be better for certain matrices depending on the structure of the inverse. Considering right approximate inverses, we seek to minimize

$$F(M) = \|I - AM\|_F^2. \tag{3}$$

An important feature of this objective function is that it can be decoupled as the sum of the squares of the 2-norms of the individual columns of the residual matrix $I - AM$

$$F(M) = \|I - AM\|_F^2 = \sum_{j=1}^{n} \|e_j - Am_j\|_2^2 \tag{4}$$

in which e_j and m_j are the j-th columns of the identity matrix and of the matrix M, respectively. Thus, minimizing (4) is equivalent to minimizing the individual functions

$$f_j(m) = \|e_j - Am\|_2^2, \quad j = 1, 2, \ldots, n. \tag{5}$$

This is clearly useful for parallel implementations. It also gives rise to a number of different options to choose from.

Up to now, the minimization in (5) has been performed directly by prescribing a sparsity pattern for M and solving the resulting least squares problems. Grote and Simon [15] choose M to be a banded matrix with $2p+1$ diagonals, $p \geq 0$, emphasizing the importance of the fast application of the preconditioner in a CM-2 implementation. This choice of structure is particularly suitable for banded matrices.

Cosgrove, Díaz and Griewank [8] select the initial structure of M to be diagonal and then use a procedure to improve the minimum by updating the sparsity pattern of M. New fill-in elements are chosen so that the fill-in contributes a certain improvement while minimizing the number of new rows in the least squares subproblem. In similar and recent work by Huckle and Grote [11], the reduction in the residual norm is tested for each candidate fill-in element, but fill-in may be introduced more than one at a time.

In other related work, Kolotilina and Yeremin [18] consider symmetric, positive definite systems and construct factorized sparse approximate inverse preconditioners which are also symmetric, positive definite. The preconditioner has the form $M = G_L^T G_L$ where G_L is lower triangular, and thus A is preconditioned to become $G_L A G_L^T$. A weighted Frobenius norm is used in this case, and results in a small linear system for each row for a given sparsity pattern. The structure of G_L is chosen to be the same as the structure of the lower triangular part of A. In their more recent work [19], fill-in elements may be added, but are chosen in a way so that constructing G_L is not much more expensive, with the simple hope that the additional degrees of freedom will improve the preconditioner quality. Preconditioners for general systems may be constructed by approximating the left and right factors separately.

This paper is organized as follows. In Section 2, we present our approximate inverse algorithms, describing some implementation details and various options. We derive some simple theoretical results for approximate inverses and the convergence behavior of the algorithms in Section 3. In Section 4, we compare and comment on the performance of the various options on matrices mostly from the Harwell-Boeing collection. Finally in Section 5, we present some ideas and experiments with practical variations and applications of approximate inverses.

2 Construction of the approximate inverse

In contrast to the previous work described above, to minimize the objective function (3) we do not prescribe a sparsity pattern for M beforehand. The location and value of the nonzero elements are determined naturally by using an iterative procedure to either minimize (3) or (5), the latter as a collection of n individual columns. In addition, elements in the approximate inverse may be removed by a numerical dropping strategy if they contribute little to the inverse. These features are clearly necessary for general sparse matrices. In Sections 2.1 and 2.2 we briefly describe two approaches where M is treated as a matrix in its entirety, rather than as individual columns. We found, however, that these methods converge more slowly than if the columns are treated separately. In the remaining sections, we consider this latter approach and the various options that are available.

2.1 Newton iteration

As an alternative to directly minimizing the objective function (3), an approximate inverse may also be computed using an iterative process known as the method of Hotelling and Bodewig [16]. This method, which is modeled after Newton's method for solving $f(x) \equiv 1/x - a = 0$, has many similarities to our descent methods which we describe later. The iteration takes the form

$$M_{i+1} = M_i(2I - AM_i), \qquad i = 0, 1, \ldots$$

For convergence, we require that the spectral radius of $I - AM_0$ be less than one, and if we choose an initial guess of the form $M_0 = \alpha A^T$ then convergence is achieved if

$$0 < \alpha < \frac{2}{\rho(AA^T)}.$$

In practice, we can follow Pan and Reif [22] and use

$$\alpha = \frac{1}{\|AA^T\|_1}$$

for the right approximate inverse. As the iterations progress, M becomes denser and denser, and a natural idea here is to perform the above iteration in sparse mode [21], i.e., drop some elements in M or else the iterations become too expensive. In this case, however, the convergence properties of the Newton iteration are lost. We will show the results of some numerical experiments in Section 4.

2.2 Global iteration

In this section we describe a 'global' approach to minimizing (3), where we use a descent-type method, treating M as an unknown sparse matrix. The objective function (3) is a quadratic function on the space of $n \times n$ matrices, viewed as objects in $\mathbb{R}^{n^2}$. The actual inner product on the space of matrices with which the function (4) is associated is

$$\langle X, Y \rangle = \mathrm{tr}(Y^T X). \tag{6}$$

One possible descent-type method we may use is steepest descent which we will describe later. In the following, we will call the array representation of an n^2 vector X the $n \times n$ matrix whose column vectors are the successive n-vectors of X.

In descent algorithms a new iterate M_{new} is defined by taking a step along a selected direction G, i.e.,

$$M_{new} = M + \alpha G$$

in which α is selected to minimize the objective function associated with M_{new}. This is achieved by taking

$$\alpha = \frac{\langle R, AG \rangle}{\langle AG, AG \rangle} = \frac{\mathrm{tr}(R^T AG)}{\mathrm{tr}\,((AG)^T AG)}. \tag{7}$$

Note that the denominator may be computed as $\|AG\|_F^2$. After each of these descent steps is taken, the resulting matrix M will tend to become denser. It is therefore essential to apply numerical dropping to the new M. However, the descent nature of the step is now lost, i.e., it is no longer guaranteed that $F(M_{new}) \leq F(M)$. An alternative would be to apply numerical dropping to the direction of search G before taking the descent step. However, in this case, the fill-in in M cannot be controlled.

The simplest choice for the descent direction G is to take it to be the residual matrix $R = I - AM$, where M is the new iterate. The corresponding descent algorithm is referred to as the Minimal Residual (MR) algorithm. It is easy to see that R is indeed a descent direction for F. Our global Minimal Residual algorithm will therefore have the following form:

Algorithm **2.1 Global Minimal Residual descent algorithm**

```
1.  Select an initial $M$
2.  Until convergence do
3.      Compute $G := I - AM$
4.      Compute $\alpha$ by (7)
5.      Compute $M := M + \alpha G$
6.      Apply numerical dropping to $M$
7.  End do
```

Another popular choice is to take G to be the direction of steepest descent, i.e., the direction opposite to the gradient. Thinking in terms of n^2 vectors, the gradient of F can be viewed as an n^2 vector g such that

$$F(x + e) = F(x) + (g, e) + O(\|e\|^2)$$

where $(\cdot,\cdot)$ is the usual Euclidean inner product. If we represent all vectors as 2-dimensional $n \times n$ arrays, then the above relation is equivalent to

$$F(X+E) = F(X) + \langle G, E\rangle + O(\|E\|^2).$$

This allows us to determine the gradient as an operator on arrays, rather than n^2 vectors, as is done in the next proposition.

Proposition 2.1 *The array representation of the gradient of F with respect to M is the matrix*

$$G = -2A^T R$$

in which R is the residual matrix $R = I - AM$.

Proof. For any matrix E we have

$$\begin{aligned} F(M+E) - F(M) &= \text{tr}(I - A(M+E))^T(I - A(M+E)) \\ &\qquad\qquad -\text{tr}(I - A(M))^T(I - A(M)) \\ &= \text{tr}\left[(R - AE)^T(R - AE) - R^T R\right] \\ &= -\text{tr}\left[(AE)^T R + R^T AE - (AE)^T(AE)\right] \\ &= -2\text{tr}(R^T AE) + \text{tr}(AE)^T(AE) \\ &= -2\langle A^T R, E\rangle + \langle AE, AE\rangle. \end{aligned}$$

Thus, the differential of F applied to E is the inner product of $-2A^T R$ with E plus a second order term. The gradient is therefore simply $-2A^T R$. □

The steepest descent algorithm consists of simply replacing G in line 3 of the MR algorithm described above by $G = A^T R$. This algorithm can be a very slow in some cases, since it is essentially a steepest descent-type algorithm applied to the normal equations.

In either global steepest descent or minimal residual, we need to form and store the G matrix explicitly. The scalars $\|AG\|_F^2$ and $\text{tr}(G^T AG)$ can be computed from the successive columns of AG, which can be generated, used, and discarded. Therefore, we need not store the matrix AG.

We will show the results of some numerical experiments with this global iteration and compare them with other methods in Section 4.

2.3 Implementation of sparse MR and GMRES

We now describe column-oriented algorithms which consist of minimizing the individual objective functions (5). We perform this minimization by taking a sparse initial guess and solving approximately the n parallel linear subproblems

$$Am_j = e_j, \quad j = 1, 2, \ldots, n \tag{8}$$

with a few steps of a nonsymmetric descent-type method, such as MR or untruncated GMRES. For this method to be efficient, the iterative method must work in sparse mode, i.e., m_j is stored and operated on as a sparse vector, and the Arnoldi basis in GMRES is kept in sparse format.

In the following MR algorithm, n_i iterations are used to solve (8) approximately for each column, giving an approximation to the j-th column of the inverse of A. Each initial m_j is taken from the columns of an initial guess, M_0. In the GMRES version of the algorithm, we never use restarting since since n_i is typically very small.

Algorithm **2.2 Minimal Residual iteration**

```
1.  Start: set $M = M_0$
2.  For each column $j = 1, \ldots, n$ do
3.      Define $m_j = Me_j$
4.      For $i = 1, \ldots, n_i$ do
5.          $r_j := e_j - Am_j$
6.          $\alpha_j := \frac{(r_j, Ar_j)}{(Ar_j, Ar_j)}$
7.          $m_j := m_j + \alpha_j r_j$
8.          Apply numerical dropping to $m_j$
9.      End do
10. End do
```

Thus, the algorithm computes the current residual r_j and then minimizes the residual norm $e_j - Am_{j,new}$ in the set $m_j + \alpha r_j$.

In the sparse implementation of MR and GMRES, the matrix-vector product, SAXPY, and dot product kernels now all entirely involve sparse vectors. The matrix-vector product is much more efficient if the sparse matrix is stored by columns since all the entries do not need to be traversed. Efficient codes for all these kernels may be constructed which utilize a full n-length work vector [9].

Columns from an initial guess M_0 for the approximate inverse are used as the initial guesses for the iterative solution of the linear subproblems. There are two obvious choices: $M_0 = \alpha I$ and $M_0 = \alpha A^T$. The scale factor α is chosen to minimize the spectral radius $\rho(I - \alpha AM)$. Denoting the initial guess as $M_0 = \alpha M$ and writing

$$\frac{\partial}{\partial \alpha}\|I - \alpha AM\|_F^2 = \frac{\partial}{\partial \alpha}\mathrm{tr}[(I - \alpha AM)^T(I - \alpha AM)] = 0$$

leads to

$$\alpha = \frac{\mathrm{tr}(AM)}{\mathrm{tr}(AM(AM)^T)}.$$

The identity initial guess is cheaper to use because it is sparser than the transpose of A. However, $M_0 = \alpha A^T$ is often a much better initial guess. The quality of the initial guess may determine whether or not the approximate inverse is a good preconditioner. If the identity is a poor initial guess, the approximate inverse may take a long time to converge toward the inverse. If the transpose is a poor initial guess and a poor dropping strategy is used, nonzeros in the transpose may never be dropped to allow the necessary fill-in in other locations. Note that the transpose of A as an initial guess makes the initial preconditioned system AM_0 symmetric, positive definite.

2.4 Self-preconditioning

The approximate solution of the linear subproblems (8) using an iterative method suffers from the same problems as solving the original problem if A is poorly conditioned. However, the linear systems may be preconditioned with the columns that have already been computed. More precisely, each system (8) for approximating column j may be preconditioned with M'_0 where the first $j-1$ columns of M'_0 are the m_k that already have been computed, $1 \le k < j$, and the remaining columns are the initial guesses for the m_k, $j \le k \le n$.

This suggests that it is possible to define *outer* iterations that sweep over the matrix, as well as *inner* iterations which compute each column. On each subsequent outer iteration, the initial guess for each column is the previous result for that column.

With this approach, the parallelism of constructing the columns of the approximate inverse simultaneously is lost, but the quality of the preconditioner is often much improved for the same amount of serial work. We note however that self-preconditioning does not always help, producing worse performance in many examples.

There is another variant of self-preconditioning that is easier to implement and more easily parallelizable. Quite simply, all the inner iterations are computed simultaneously and the results of all the columns are used as the self-preconditioner for the next outer iteration. Thus, the preconditioner for the inner iterations changes only after each outer iteration. For general matrices, the performance of this variant lies between full self-preconditioning and no self-preconditioning. A more reasonable compromise is to compute blocks of columns in parallel, and some (inner) self-preconditioning may be used.

Algorithm 2.3 implements the Minimal Residual iteration with self-preconditioning. In the algorithm, n_o outer iterations and n_i inner iterations are used. Again, $M = M_0$ initially. In the GMRES version of the algorithm, a variant called FGMRES [25] that allows a flexible preconditioner is actually used.

ALGORITHM **2.3 Self-preconditioned Minimal Residual iteration**

```
1.  Start: $M = M_0$
2.  For outer $= 1, 2, \ldots, n_o$ do
3.      For each column $j = 1, \ldots, n$ do
4.          Define $s := Me_j$
5.          For inner $= 1, \ldots, n_i$ do
```

```
6.                  $r := e_j - As$, $z = Mr$
7.                  $q := Az$
8.                  $\alpha := \frac{(r,q)}{(q,q)}$
9.                  $s := s + \alpha z$
10.                 Apply numerical dropping to $s$
11.             End do
12.             Update $j$-th column of $M$: $m_j := s$
13.         End do
14.     End do
```

In implementing self-preconditioning in FORTRAN 77, two sparse matrices are used to avoid copying of the columns of the matrix. The columns are appended to one matrix or the other depending on the parity of the outer iteration number. The matrix-vector multiplication with the self-preconditioner is done with the first $j-1$ columns of one matrix, and the remaining columns of the second matrix.

In the presence of self-preconditioning, it is not clear in what order the columns should be computed. It may be possible to choose an ordering of the matrix that will make effective use of the fill-in. We have conducted some brief experiments with Cuthill-McKee reordering for this purpose, but found the results inconclusive.

2.5 Numerical dropping strategies

There are many possible alternatives for numerical dropping. There are three issues:

1. which vectors/matrices have dropping applied,
2. when dropping is performed,
3. which elements are dropped.

In the previous algorithms, we have made issues 1 and 2 precise. However, other alternatives are possible. It was mentioned that dropping may be performed on the search direction G in Algorithm 2.1, or equivalently in r_j and z in Algorithms 2.2 and 2.3 respectively. In this case, the descent property of the algorithms is maintained, but the fill-in in M cannot be controlled, and ineffective nonzeros in M cannot be removed.

In all the algorithms, there is the possibility that dropping is only performed after M or each column of M is computed. Typically this option is too expensive. In the self-preconditioned case, a compromising option is available: dropping may be performed at the end of the inner iterations, before M is updated, namely before step 12 in Algorithm 2.3. This makes the algorithm computationally more expensive, since the vectors which are multiplied with A and M are denser. Interestingly, we found experimentally that this option is not always better.

In GMRES, the Krylov basis vectors are kept sparse by dropping elements just before the multiplication by A, or just after the self-preconditioning step.

To address issue 3, we utilize a dual threshold strategy based a drop tolerance, *droptol*, and the maximum number of elements per column, *lfil*, similar to that used in the ILUT preconditioner [24]. By limiting the maximum number of elements per column, the maximum storage for the preconditioner is known beforehand.

The drop tolerance may be applied directly to the elements to be dropped: i.e., elements are dropped if their magnitude is smaller than *droptol*. However, we found that this strategy could cause *spoiling* of the minimization and thus a deterioration of the quality of the preconditioner.

If dropping small elements in m_j is sub-optimal, one may ask the question whether or not dropping can be performed more optimally. A simple perturbation analysis will help understand the issues. We denote by m_j the current column and by d the sparse column that is added to m_j in the process of numerical dropping. The new column and corresponding residual are therefore

$$\hat{m}_j = m_j + d, \quad \hat{r}_j = r_j - Ad.$$

The square of the residual norm of the perturbed m_j is given by

$$\|\hat{r}_j\|_2^2 = \|r_j\|_2^2 - 2(d, A^T r_j) + \|Ad\|_2^2. \tag{9}$$

Recall that $-2A^T r_j$ is the gradient of the function (5). As is expected from standard results in optimization, if d is in the direction opposite to the gradient, and if it is small enough, we can achieve a decrease of the residual norm. Spoiling occurs when $(d, A^T r_j)$ is close to zero so that for practical sizes of $\|d\|_2$, $\|Ad\|_2^2$ becomes dominant causing an increase in the residual norm.

Consider specifically the situation where only one element is dropped, and assume that all the columns Ae_i of A have been pre-scaled so that $\|Ae_i\|_2 = 1$. In this case, $d = m_{ij}e_i$ and the above equation becomes

$$\|\hat{r}_j\|_2^2 = \|r_j\|_2^2 - 2m_{ij}(e_i, A^T r_j) + m_{ij}^2. \tag{10}$$

A strategy would therefore be based on attempting to make the function

$$\|\hat{r}_j\|_2^2 - \|r_j\|_2^2 = -2m_{ij}(e_i, A^T r_j) + m_{ij}^2 \tag{11}$$

nonpositive, a condition which is easy to verify. This suggests selecting elements to drop in m_j only at indices i where the selection function (11) is zero or negative. However, note that this is not entirely rigorous since in practice a few elements are dropped at the same time. Thus we do not entirely perform dropping via numerical values alone. In a two-stage process, we first select a number of candidate elements to be dropped based only on the numerical size as determined by a certain tolerance. Among these, we drop all those that satisfy the condition

$$\rho_{ij} = -2m_{ij}(e_i, A^T r_j) + m_{ij}^2 < tol_2$$

or we can keep those *lfil* elements that have the largest ρ_{ij}.

An alternative which we tried was based on attempting to achieve maximum reduction in the function (11). Ideally, we wish to have

$$m_{ij} = (e_i, A^T r_j)$$

since this will achieve the 'optimal' reduction in (11)

$$\|\hat{r}_j\|_2^2 - \|r_j\|_2^2 = -m_{ij}^2.$$

This leads to the alternative strategy of dropping elements in positions i of m_j where $m_{ij} - (e_i, A^T r_j)$ are the smallest. We found, however, that this strategy produces poorer results than the previous one.

2.6 Factorized forms of the approximate inverse

Probabilistically, the product of two sparse matrices has more nonzeros than the total number of nonzeros in the individual factors, given that the factors have at least a certain small density depending on their size. This suggests that factorized forms of sparse matrices contain more information for the same storage than if a single product was stored. If multiple outer iterations are used in computing an approximate inverse, each outer iteration can be considered to contribute a factorized update to it. Suppose that one outer iteration has produced the approximate inverse M_1. Then a second outer iteration tries to find M_2, an approximate inverse to AM_1. In general, after i outer iterations, we are looking for the update M_{i+1} which minimizes

$$\min_{M_{i+1}} \|I - AM_1M_2\cdots M_iM_{i+1}\|_F^2. \tag{12}$$

It is also possible to construct factorized approximate inverses of the form

$$\min_{M_{2i+1}} \|I - M_{2i}\cdots M_4M_2AM_1M_3\cdots M_{2i+1}\|_F^2 \tag{13}$$

which alternate from left to right factors. This latter form is reminiscent of the symmetric form of Kolotilina and Yeremin [18]. However, we only did experiments with the first factorized form (12), which we consider from here on.

Since the product $M_1M_2\cdots M_i$ is never formed explicitly, the factorized approach effectively uses less memory for the preconditioner at the cost of multiplying with each factor for each matrix-vector multiplication. This approach may be suitable for very large problems, where memory rather than solution time is the limiting factor. The implementation, however, is much more complex, since a sequence of matrices needs to be maintained.

The initial guess for each M_i is also an issue, and we choose between a scaled identity matrix and a scalar multiple of $(AM_1M_2\cdots M_{i-1})^T$. Of course, the latter is much more expensive since it must be computed and has immediately more nonzeros. The comments in Section 2.3 also apply.

2.7 Cost of constructing the approximate inverse

The cost of computing the approximate inverse is relatively high. Let n be the dimension of the linear system, n_o be the number of outer iterations, and n_i be the number of inner iterations.

Let us approximate the cost by counting the number of matrix-vector multiplications in the sparse implementation of MR and GMRES. Profiling for a few problems shows that this operation accounts for about 75 percent of the time when self-preconditioning is used. The remaining time is used by the sparse dot product and sparse SAXPY operations, and in the case of GMRES, the additional work within this algorithm.

The cost is simply nn_on_i matrix-vector multiplications, or twice this amount if self-preconditioning is used. If we assume that dropping is performed on each column at each inner iteration, then the cost of the matrix-vector multiplication with A is approximately $n_c\mathit{lfil}$ mult-add flops, where n_c is the average number of nonzeros per column, typically a small constant, and *lfil* is the maximum number of nonzeros per column in M allowed by the dropping strategy. The cost of matrix-vector multiplication by M is more expensive since M is often allowed to be denser: lfil^2 flops are required in this case. Thus self-preconditioning is expensive, especially if *lfil* is large. Self-preconditioning may also make the columns denser much more quickly.

In summary, the number of flops for all the matrix-vector multiplications is

$$nn_on_i(n_c\mathit{lfil} + \mathit{lfil}^2)$$

with the second term present for self-preconditioning. For $n_o = 5, n_i = 2, n_c = 5, \mathit{lfil} = 20$, this is $1000n$ flops in the nonself-preconditioned case, and $5000n$ flops in the self-preconditioned case.

In both Newton and global iterations, two sparse matrix-matrix products are required in each iteration, although the convergence rate of Newton iteration may be doubled with a form of Chebyshev acceleration [23]. The cost is thus comparable to the cost of the column-oriented, self-preconditioned MR algorithm, where $2n$ sparse matrix-sparse vector products are used per iteration, or simply n products if self-preconditioning is not used.

3 Theoretical considerations

Theoretical results regarding the quality of approximate inverse preconditioners are difficult to establish. However, we can prove a few rather simple results for general approximate inverses and on the convergence behavior of the algorithms.

3.1 Nonsingularity of M

An important question we wish to address is whether or not an approximate inverse obtained by the approximations described earlier can be singular. It cannot be proved that M is nonsingular unless the approximation is accurate enough. More specifically, the following general result is easy to show.

Proposition 3.1 *Assume that A is nonsingular and that the residual of the approximate inverse M satisfies the relation*

$$\|I - AM\| < 1 \tag{14}$$

where $\|\cdot\|$ is any consistent matrix norm. Then M is nonsingular.

Proof. The result follows immediately from the equality

$$AM = I - (I - AM) \equiv I - N \tag{15}$$

and the well-known fact that if $\|N\| < 1$, then $I - N$ is nonsingular. □

We note that the result is true in particular for the Frobenius norm, which, although not an induced matrix norm, is consistent.

It may sometimes be the case that AM is poorly balanced and as a result $I - AM$ can be large. Then balancing AM can yield a smaller norm and possibly a less restrictive condition for the nonsingularity of M. It is easy to extend the previous result as follows.

Corollary 3.1 *Assume that A is nonsingular and that there exist two nonsingular diagonal matrices D_1, D_2 such that*

$$\|I - D_1 AMD_2\| < 1 \tag{16}$$

where $\|\cdot\|$ is any consistent matrix norm. Then M is nonsingular.

Proof. Applying the previous result to $A' = D_1 A$ and $M' = MD_2$, implies that $M' = MD_2$ will be nonsingular from which the result follows. □

Of particular interest is the 1-norm. Each column is obtained independently by requiring a condition on the residual norm of the form

$$\|e_j - Am_j\| \le \tau. \tag{17}$$

We typically use the 2-norm since we measure the magnitude of the residual $I - AM$ using the Frobenius norm. However, using the 1-norm for a stopping criterion allows us to prove a number of simple results. We will assume in the following that we require a condition of the form

$$\|e_j - Am_j\|_1 \le \tau_j \tag{18}$$

for each column. Then we can prove the following result.

Proposition 3.2 *Assume that the condition (18) is imposed on each computed column of the approximate inverse and let $\tau = \max_j \ \tau_j$, $j = 1, \ldots, n$. Then,*

1. *Any eigenvalue λ of the preconditioned matrix AM is located in the disc*

$$|\lambda - 1| < \tau. \tag{19}$$

2. *If $\tau < 1$, then M is nonsingular.*
3. *If any k columns of M, with $k \le n$, are linearly dependent then at least one residual $e_j - Am_j$ associated with one of these columns has a 1-norm ≥ 1.*

Proof. To prove the first property we invoke Gershgorin's theorem on the matrix

$$AM = I - R$$

each column of R is the residual vector $r_{:,j} = e_j - Am_j$. The column version of Gershgorin's theorem, see e.g., [26, 14], asserts that all the eigenvalues of the matrix $I - R$ are located in the union of the disks centered at the diagonal elements $1 - r_{jj}$ and with radius

$$\sum_{i=1, i\ne j}^{n} |r_{ij}|.$$

In other words, each eigenvalue λ must satisfy at least one inequality of the form

$$|\lambda - (1 - r_{jj})| \le \sum_{i=1, i\ne j}^{n} |r_{ij}|$$

from which we get

$$|\lambda - 1| \le \sum_{i=1}^{n} |r_{ij}| \le \tau_j.$$

Therefore, each eigenvalue is located in the disk of center 1, and radius τ. The second property is a restatement of the previous proposition and follows also from the first property.

To prove the last point we assume without loss of generality that the first k columns are linearly dependent. Then there are k scalars α_i, not all zero such that

$$\sum_{i=1}^{k} \alpha_i m_i = 0. \tag{20}$$

We can assume also without loss of generality that the 1-norm of the vector of α's is equal to one (this can be achieved by rescaling the α's). Multiplying through (20) by A yields

$$0 = \sum_{i=1}^{k} \alpha_i A m_i = \sum_{i=1}^{k} \alpha_i (e_i - r_i)$$

which gives

$$\sum_{i=1}^{k} \alpha_i e_i = \sum_{i=1}^{k} \alpha_i r_i.$$

Taking the 1-norms of each side, we get

$$1 = \sum_{i=1}^{k} \|\alpha_i r_i\|_1 \leq \sum_{i=1}^{k} |\alpha_i| \|r_i\|_1 \leq \sum_{i=1}^{k} |\alpha_i| \max_{i=1,\ldots,k} \|r_i\|_1 = \max_{i=1,\ldots,k} \|r_i\|_1.$$

Thus at least one of the 1-norms of the residuals $r_i, i = 1, \ldots, k$ must be ≥ 1. □

We may ask the question as to whether similar results can be shown with other norms. Since the other norms are equivalent we can clearly adapt the above results in an easy way. For example,

$$\|x\|_1 \leq \sqrt{n}\|x\|_2 \quad \text{and} \quad \|x\|_1 \leq n\|x\|_\infty. \tag{21}$$

However, the resulting statements would be too weak to be of any practical value. We can exploit the fact that since we are computing a sparse approximation, the number p of nonzero elements in each column is small, and thus we replace the scalar n in the above inequalities by p [11].

We should point out that the result does not tell us anything about the degree of sparsity of the resulting approximate inverse M. It may well be the case that in order to guarantee nonsingularity, we must have an M that is dense, or nearly dense. In fact, in the particular case where the norm in the proposition is the 1-norm, it has been proved by Cosgrove, Díaz and Griewank [8] that the approximate inverse may be *structurally dense*, in that it is always possible to find a sparse matrix A for which M will be dense if $\|I - AM\|_1 < 1$.

Next we examine the sparsity of M and prove a simple result for the case where an assumption of the form (18) is made.

Proposition 3.3 *Let $B = A^{-1}$ and assume that a given element b_{ij} of B satisfies the inequality*

$$|b_{ij}| > \tau_j \max_{k=1,n} |b_{ik}|, \tag{22}$$

then the element m_{ij} is nonzero.

Proof. From the equality $AM = I - R$ we get

$$M = A^{-1} - A^{-1}R.$$

Thus,

$$m_{ij} = b_{ij} - \sum_{k=1}^{n} b_{ik} r_{kj}$$

and

$$\begin{aligned} |m_{ij}| &= |b_{ij} - \sum_{k=1}^{n} b_{ik} r_{kj}| \\ &\geq |b_{ij}| - \sum_{k=1}^{n} |b_{ik} r_{kj}| \\ &\geq |b_{ij}| - \max_{k=1,n} |b_{ik}| \|r_j\|_1 \\ &\geq |b_{ij}| - \max_{k=1,n} |b_{ik}| \tau_j . \end{aligned}$$

Thus, if the condition (22) is satisfied, we must have $|m_{ij}|>0$. □

This tells us that if R is small enough, then the nonzero elements of M are located in positions corresponding to the larger elements in the inverse of A. The following negative result is an immediate corollary.

Corollary 3.2 *Let τ de defined as in Proposition 3.2. If the nonzero elements of $B = A^{-1}$ are τ-equimodular in that*

$$|b_{ij}| > \tau \max_{k=1,n,\ l=1,n} |b_{lk}|$$

, then the nonzero sparsity pattern of M includes the nonzero sparsity pattern of A^{-1}. In particular, if A^{-1} is dense and its elements are τ-equimodular, then M is also dense.

The smaller the value of τ, the more likely the condition of the corollary will be satisfied. Another way of stating the corollary is that we will be able to compute *accurate* and *sparse* approximate inverses only if the elements of the actual inverse have variations in size. Unfortunately, this is difficult to verify in advance.

3.2 Case of a nearly singular A

Consider first a singular matrix A, with a singularity of rank one, i.e., the eigenvalue 0 is single. Let z be an eigenvector associated with this eigenvalue. Then, each subsystem (8) that is being solved by MR or GMRES will provide an approximation to the system, except that it cannot resolve the component of the initial residual associated with the eigenvector z. In other words, the iteration may stagnate after a few steps. Let us denote by P the spectral projector associated with the zero eigenvalue, by m_0 the initial guess to the system (8), and by $r_0 = e_j - Am_0$ the initial residual. For each column j, we would have at the end of the iteration an approximate solution of the form $m = m_0 + \delta$, whose residual is

$$\begin{aligned} e_j - Am &= (e_j - Am_0) - A\delta \\ &= r_0 - A\delta \\ &= \alpha Pr_0 + (I-P)r_0 - A\delta. \end{aligned}$$

The term Pr_0 cannot be reduced by any further iterations. Only the norm of $(I-P)r_0 - A\delta$ can be reduced by selecting a more accurate δ. The MR algorithm can also break down when Ar_j vanishes, causing a division by zero in the computation of the scalar α_j in step 6 of Algorithm 2.2, although this is not a problem with GMRES.

An interesting observation is that in case A is singular, M is not too well defined. Adding a rank-one matrix zv^T to M will indeed yield the same residual

$$I - A\left[M + zv^T\right] = I - AM$$

$$R = R_0 - AD.$$

Assume now that A is nearly singular, in that there is one eigenvalue ϵ close to zero with an associated eigenvector z. Note that for any vector v we have

$$I - A\left[M + zv^T\right] = I - AM - \epsilon zv^T.$$

If z and v are of norm one, then the residual is perturbed by a magnitude of ϵ. Viewed from another angle, we can say that for a perturbation of order ϵ in the residual, the approximate inverse can be perturbed by a matrix of norm close to one.

3.3 Eigenvalue clustering around zero

We observed in many of our experiments that often the matrix M obtained in a self-preconditioned iteration would admit a cluster of eigenvalues around the origin. More precisely, it seems that if at some point an eigenvalue of AM moves very close to zero, then this singularity tends to persist in the later stages in that the zero eigenvalue will move away from zero only very slowly. These eigenvalues seem to slow-down or even prevent convergence. In this section, we attempt to analyze this phenomenon. We examine the case where at a given intermediate iteration the matrix M becomes exactly singular. We start by assuming that a global MR iteration is taken, and that the preconditioned matrix AM is singular, i.e., there exists a nonzero vector z such that

$$AMz = 0.$$

In our algorithms, the initial guess for the next (outer) iteration is the current M, so the initial residual is $R = I - AM$. The matrix M' resulting from the next self-preconditioned iteration, either by a global MR or GMRES, will have a residual of the form

$$R' = I - AM' = \rho(AM)R = \rho(AM)(I - AM) \tag{23}$$

in which $\rho(t) = 1 - ts(t)$ is the residual polynomial. Multiplying (23) to the right by the eigenvector z yields

$$\begin{aligned}(I - AM')z &= \rho(AM)(I - AM)z\\ &= \rho(AM)z\\ &= [I - AMs(AM)]z\\ &= z.\end{aligned}$$

As a result we have

$$AM'z = 0$$

showing that z is an eigenvector of AM' associated with the eigenvalue zero.

This result can be extended to column-based iterations. First, we assume that the preconditioning M used in self-preconditioning all n inner iterations in a given outer loop is fixed. In this case, we need to exploit a left eigenvector w of AM associated with the eigenvalue zero. Proceeding as above, let m'_j be the new j-th column of the approximate inverse. We have

$$e_j - Am'_j = \rho_j(AM)(e_j - Am_j) \tag{24}$$

where ρ_j is the residual polynomial associated with the MR or GMRES algorithm for the j-th column, and is of the form $\rho_j(t) = 1 - ts_j(t)$.

Multiplying (24) to the left by the eigenvector w^T yields

$$\begin{aligned}w^T(e_j - Am'_j) &= w^T\rho_j(AM)(e_j - Am_j)\\ &= w^T[I - AMs_j(AM)](e_j - Am_j)\\ &= w^T(e_j - Am_j).\end{aligned}$$

As a result $w^TAm'_i = w^TAm_i$ for $i = 1, 2, \ldots, n$ which can be rewritten as $w^TAM' = w^TAM$. This gives

$$w^TAM' = 0,$$

establishing the same result on the persistence of a zero eigenvalue as for the global iteration.

We finally consider the general column-based MR or GMRES iterations, in which the self-preconditioner is updated from one inner iteration to the next. We can still write

$$e_j - Am'_j = \rho_j(AM)(e_j - Am_j).$$

Let M' be the new approximate inverse resulting from updating only column j. The residual associated with M' has the same columns as those of the residual associated with M except for the j-th column which is given above. Therefore

$$\begin{aligned} I - AM' &= (I - AM)(I - e_j e_j^T) + \rho_j(AM)(e_j - Am_j)e_j^T \\ &= (I - AM)(I - e_j e_j^T) + \rho_j(AM)(I - AM)e_j e_j^T \\ &= I - AM - (I - \rho_j(AM))(I - AM)e_j e_j^T \\ &= I - AM - AMs_j(AM)(I - AM)e_j e_j^T . \end{aligned}$$

If w is again a left eigenvector of AM associated with the eigenvalue zero, then multiplying the above equality to the left by w^T yields

$$w^T AM' = w^T AM = 0,$$

showing once more that the zero eigenvalue will persist.

3.4 Convergence behavior of self preconditioned MR

Next we wish to consider the convergence behavior of the algorithms for constructing an approximate inverse. We are particularly interested in the situation where self-preconditioning is used, but no numerical dropping is applied.

We consider first the case where each column is updated individually by exactly one step of the MR algorithm. Let M be the current approximate inverse at a given substep. The self-preconditioned MR iteration for computing the j-th column of the next approximate inverse is obtained by the following sequence of operations:

```
1. $r_j := e_j - Am_j = e_j - AMe_j$
2. $t_j := Mr_j$
3. $\alpha_j := \frac{(r_j, At_j)}{(At_j, At_j)}$
4. $m_j := m_j + \alpha_j t_j$
```

Note that α_j can be written as

$$\alpha_j = \frac{(r_j, AMr_j)}{(AMr_j, AMr_j)} = \frac{(r_j, Br_j)}{(Br_j, Br_j)}$$

where we define

$$B = AM$$

to be the preconditioned matrix at the given substep. We now drop the index j to simplify the notation. The new residual associated with the current column is given by

$$\begin{aligned} r^{new} &= r - \alpha t \\ &= r - \alpha AMr \\ &\equiv r - \alpha Br. \end{aligned}$$

We use the orthogonality of the new residual against AMr to obtain

$$\|r^{new}\|_2^2 = \|r\|_2^2 - \alpha^2 \|Br\|^2.$$

Replacing α by its value defined above we get

$$\|r^{new}\|_2^2 = \|r\|_2^2 \left[1 - \left(\frac{(Br,r)}{\|Br\|_2 \|r\|_2} \right)^2 \right].$$

Thus, at each inner iteration, the residual norm for the j-th column is reduced according to the formula

$$\|r^{new}\|_2 \;=\; \|r\|_2 \sin \angle(r, Br) \tag{25}$$

in which $\angle(u,v)$ denotes the acute angle between the vectors u and v. Assuming that each column converges, the preconditioned matrix B will converge to the identity. As a result of this, the angle $\angle(r, Br)$ will tend to $\angle(r,r) = 0$ and therefore the convergence ratio $\sin \angle(r, Br)$ will also tend to zero, showing superlinear convergence.

We now consider equation (25) more carefully in order to analyze more explicitly the convergence behavior. We will denote by R the residual matrix $R = I - AM$. We observe that

$$\begin{aligned} \sin \angle(r, Br) &= \min_{\alpha} \frac{\|r - \alpha\ Br\|_2}{\|r\|_2} \\ &\le \frac{\|r -\ Br\|_2}{\|r\|_2} \equiv \frac{\|Rr\|_2}{\|r\|_2} \\ &\le \|R\|_2 . \end{aligned}$$

This results in the following statement.

Proposition 3.4 *Assume that the self-preconditioned MR algorithm is employed with one inner step per iteration and no numerical dropping. Then the 2-norm of each residual $e_j - Am_j$ of the j-th column is reduced by a factor of at least $\|I - AM\|_2$, where M is the approximate inverse before the current step, i.e.,*

$$\|r_j^{new}\|_2 \;\le\; \|I - AM\|_2 \; \|r_j\|_2 \tag{26}$$

In addition, the Frobenius norm of the residual matrices $R_k = I - AM_k$ obtained after each outer iteration, satisfies

$$\|R_{k+1}\|_F \;\le\; \|R_k\|_F^2 . \tag{27}$$

As a result, when the algorithm converges, it does so quadratically.

Proof. Inequality (26) was proved above. To prove quadratic convergence, we first transform this inequality by using the fact that $\|X\|_2 \le \|X\|_F$ to obtain

$$\|r_j^{new}\|_2 \;\le\; \|R_{k,j}\|_F \; \|r_j\|_2$$

Here the k index corresponds to the outer iteration and the j-index to the column. We note that the Frobenius norm is reduced for each of the inner steps corresponding to the columns, and therefore

$$\|R_{k,j}\|_F \leq \|R_k\|_F.$$

This yields

$$\|r_j^{new}\|_2^2 \;\leq\; \|R_k\|_F^2 \; \|r_j\|_2^2$$

which, upon summation over j gives

$$\|R_{k+1}\|_F \leq \|R_k\|_F^2.$$

This completes the proof. □

It is also easy to show a similar result for the following variations:

1. MR with an arbitrary number of inner steps,
2. GMRES(m) for an arbitrary m,
3. The global MR algorithm.

The first two cases follow from the fact that they deliver an approximate column which has a smaller residual than what we obtain with one inner step MR. The third case requires a proof which is very similar to the one above.

We emphasize that quadratic convergence is guaranteed only at the limit and that the above theorem does not prove convergence. In the presence of numerical dropping, the proposition does not hold.

4 Numerical experiments and observations

Experiments with the algorithms and options described in Section 2 were performed with matrices from the Harwell-Boeing sparse matrix collection [10], and a few matrices from computational fluid dynamics extracted from solving the Navier-Stokes equations with the FIDAP [13] package. The matrices were pre-scaled so that the 2-norm of each column is unity, as described in Section 2.5. In each experiment, we report the number of GMRES(20) steps to reduce the initial residual of the right-preconditioned linear system by 10^{-5}. A zero initial guess was used, and the right-hand-side was constructed so that the solution is a vector of all ones. A dagger (†) in the tables below indicates that there was no convergence in 500 iterations. In some tables we also show the value of the Frobenius norm (3). Even though this is the function that we minimize, we see that it fails to be a reliable measure of GMRES convergence in many cases. Most of the results are shown as the outer iterations progress from 1 to 5. One inner iteration was used unless otherwise indicated. The codes were written in both FORTRAN 77 and Matlab, and were run on a Sun 4 workstation and a CRAY-XMP supercomputer.

We begin with a comparison of Newton, 'global' and column-oriented iterations. Our early numerical experiments showed that in practice, Newton iteration converges very slowly initially and is much more adversely affected by numerical dropping. Global iteration was also worse than column-oriented iterations, perhaps because a single α defined by (7) is used, as opposed to one for each column in the column-oriented case. Table 1 gives some numerical results for the WEST0067 matrix from the Harwell-Boeing collection; the number of GMRES iterations is given as the number of outer iterations increases. Dropping based on numerical values was performed on a column-by-column basis, although in Newton and global iteration this restriction is not necessary. In the presence of dropping ($lfil = 10$), we did not find much larger matrices where Newton iteration gave convergent GMRES iterations. Scaling each iterate M_i by $1/\|AM_i\|_1$ did not alleviate the effects of dropping. The superior behavior of global iterations in the presence of dropping in Table 1 was not typical.

No dropping					
	1	2	3	4	5
Newton	†	414	158	100	41
Global	228	102	25	16	11
MR	130	35	13	10	6
Dropping: $lfil = 10$, $droptol = 0.001$					
	1	2	3	4	5
Newton	463	†	435	†	457
Global	241	87	46	35	26
MR	281	120	86	61	43

Table 1: WEST0067: Newton, global, and column MR iterations.

Next we show some results on matrices from computational fluid dynamics extracted using the FIDAP package. These are relatively dense matrices that are nonsymmetric, but have symmetric structure. We chose matrices that we could not solve using an incomplete LU preconditioner called ILUT [24] using an equivalent *lfil* of about 100. Our experience also showed that ILU preconditioners on these matrices produce unstable L and U factors in (2). The matrices are from 2-dimensional finite element discretizations using quadrilateral elements with 9 nodes. We show the order and number of nonzeros of these matrices in Table 2. In Table 3, numerical dropping based on (11) was used, with *lfil* of 20. Convergent GMRES iterations could be achieved even with *lfil* as small as 10, showing that an approximate inverse preconditioner much sparser than the original matrix is possible. No self-preconditioning was used; the initial guess was a scaled identity matrix.

Matrix	n	nnz	
EX3	1821	52685	Flow past a circular cylinder
EX7	1633	54543	Natural convection in a square cavity
EX9	3363	99471	Jet impingement in a narrow channel

Table 2: FIDAP example matrices.

Matrix	1	2	3	4	5
EX3	144	135	146	146	146
EX7	63	50	43	35	30
EX9	406	297	280	321	298

Table 3: Number of GMRES iterations vs. n_o.

We now show our main results in Table 4 for many standard matrices in the Harwell-Boeing collection. Again, we show the number of GMRES iterations to convergence against the number of outer iterations used to compute the approximate inverse. Dropping was based on numerical values.

The results are poorer than those for the ILUT preconditioner, but as seen with the FIDAP matrices, approximate inverses can be used to solve some problems that cannot be solved with ILU-type preconditioners. Note that many of the above results used a much smaller *lfil* than ILUT, or used very few inner iterations. The convergence rates can be improved in many cases with additional cost.

We have shown the best results after a few trials with different parameters. The method is sensitive to the widely differing characteristics of general matrices, and there is no general strategy that works best for constructing the approximate inverse. The following two tables illustrate widely different behavior for three very different matrices. LAPL0324 is a standard Laplacian matrix of order 324. The initial guess M_0 was also varied in these experiments. Table 5 shows the number of GMRES(20) iterations and Table 6 shows the Frobenius norm of the residual matrix against the number of outer iterations that were used to compute the approximate inverse.

For problems SHERMAN2, WEST0989 and GRE1107, the results become worse as the outer iterations progress. This *spoiling* effect is due to numerical dropping. In Figures 1 to 3, we show the trends in the number of iterations to convergence as a function of *lfil*, n_i and n_o for the PORES2 matrix. Dropping based on numerical values was used. These figures show the effect of spoiling with increasing number of inner and outer iterations. Increasing *lfil* is not entirely robust either. Note that the top edge of the graphs indicates no GMRES convergence.

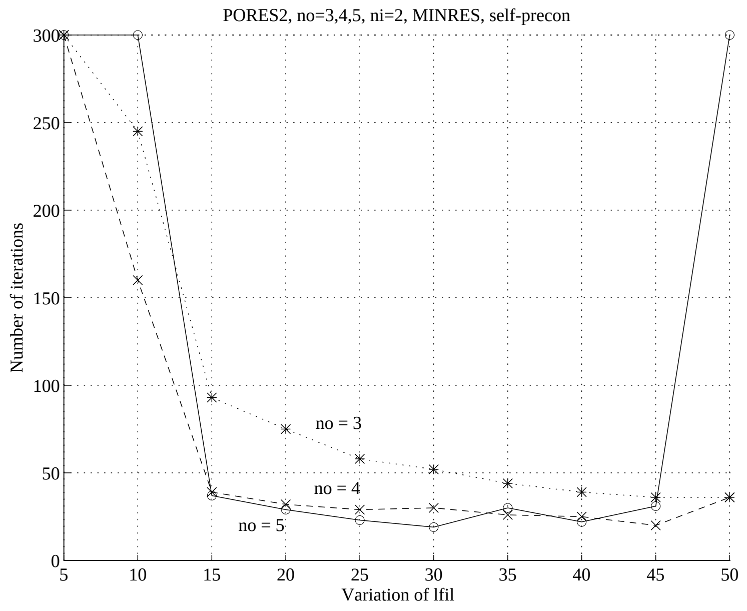


Figure 1: Variation of *lfil*.

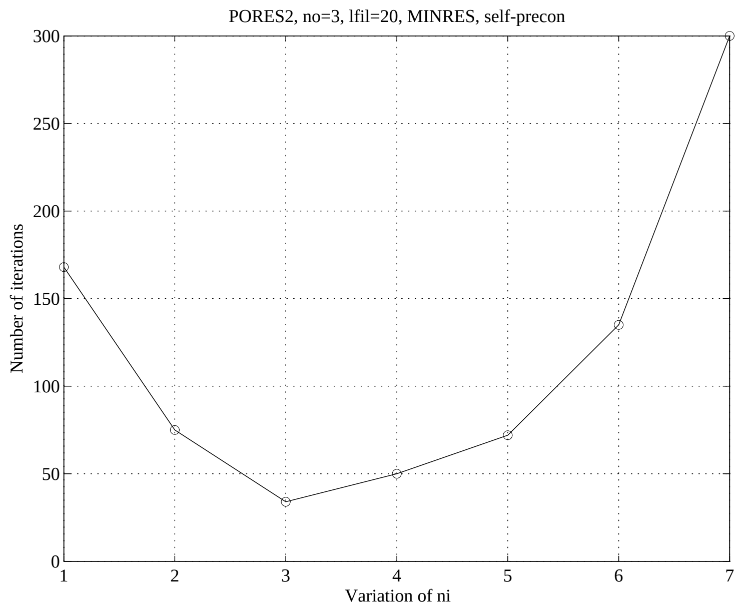


Figure 2: Variation of n_i.

Matrix	n	ilut	g/m	p/u	*lfil*	n_i	1	2	3	4	5
PORES2	1244	11	m	p	30	2	†	†	52	30	19
PORES3	532	7	m	u	10	1	†	†	421	150	112
SHERMAN1	1000	5	m	u	10	1	224	187	96	74	60
SHERMAN2	1080	6	m	p	50	1	†	†	147	46	136
SHERMAN3	5005	16	m	u	10	1	499	363	239	192	148
SHERMAN4	1104	11	m	u	10	1	87	69	43	42	41
SHERMAN5	3312	11	g	p	20	2	†	148	107	70	60
WEST0497	497	†	g	p	50	5	†	†	†	80	20
WEST0989	989	†	m	p	50	2	†	†	303	†	†
GRE1107	1107	†	m	p	50	2	421	†	†	†	†
GRE216B	216	†	m	p	5	1	3	3	3	3	3
NNC261	261	†	m	p	20	2	†	39	20	17	14

Notes:
n = order of matrix
ilut = number of ILUT iterations to convergence, using equivalent *lfil* of about 25
g/m = FGMRES or MR
p/u = self-preconditioned or unself-preconditioned
SHERMAN2 was reordered with RCM

Table 4: Number of GMRES(20) iterations vs. n_o.

The eigenvalues of the preconditioned WEST0067 matrix are plotted in Figure 4, both with and without dropping, when the number of outer iterations was 2 and 4.

As the iterations proceed, the eigenvalues of the preconditioned system become closer to 1. Numerical dropping has the effect of spreading out the eigenvalues. When dropping is severe and spoiling occurs, we have observed two phenomena: either dropping causes some eigenvalues to become negative, or some eigenvalues stay clustered around the origin.

5 Practical variations and applications

Approximate inverses can be expensive to compute for very large and difficult problems. However, the best potential is to combine them with other techniques. In essence, we would like to apply these techniques to problems that are either small, or for which we start close to a good solution in a certain sense. We saw in Table 4 that approximate inverse preconditioners work well with small matrices, for example $n = 200$. We first comment on a popular application, incomplete block tridiagonal factorizations, as applied to general sparse matrices.

Matrix	*lfil*	init	p/u	1	2	3	4	5
WEST0067	none	A^T	p	130	35	13	10	6
	none	A^T	u	484	481	†	472	†
	none	I	p	†	†	†	†	†
	10	A^T	p	281	120	86	61	43
LAPL0324	none	A^T	p	466	200	50	21	12
	none	A^T	u	21	17	12	12	10
	none	I	p	16	15	11	11	9
	10	A^T	u	30	22	17	17	17
PORES3	none	A^T	p	†	†	†	†	†
	none	A^T	u	†	†	274	174	116
	10	A^T	u	†	†	421	150	112

Table 5: Number of GMRES(20) iterations vs. n_o.

Matrix	*lfil*	init	p/u	1	2	3	4	5
WEST0067	none	A^T	p	4.43	3.21	2.40	1.87	0.95
	none	A^T	u	6.07	6.07	6.07	6.07	6.07
	none	I	p	8.17	8.17	8.17	8.17	8.17
	10	A^T	p	4.77	4.26	4.42	4.92	6.07
LAPL0324	none	A^T	p	7.91	5.69	4.25	3.12	2.23
	none	A^T	u	6.62	4.93	4.00	3.41	3.00
	none	I	p	5.34	4.21	3.53	3.08	2.75
	10	A^T	u	6.54	4.81	4.07	3.82	3.92
PORES3	none	A^T	p	10.78	9.30	8.25	7.66	7.16
	none	A^T	u	12.95	12.02	11.48	10.82	10.20
	10	A^T	u	12.94	12.02	11.48	10.82	10.23

Table 6: $\|I - AM\|_F$ vs. n_o.

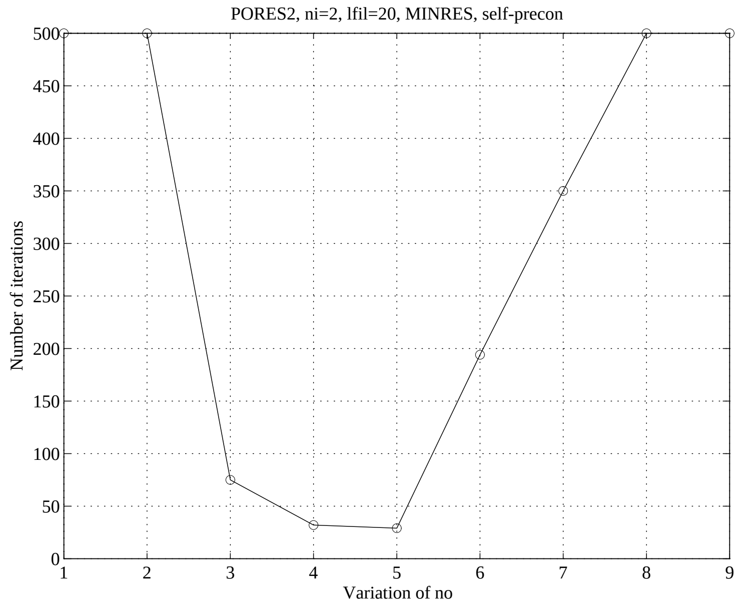


Figure 3: Variation of n_o.

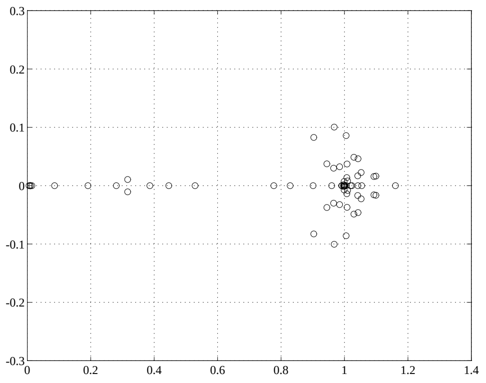
5.1 Incomplete block tridiagonal factorizations

There has been much work in the area of two-level implicit-explicit preconditioners, especially incomplete block factorizations [1, 2, 3, 4, 7, 17, 20] for matrices arising from the discretization of PDE's. These methods utilize approximate inverses when diagonal blocks need to be inverted.

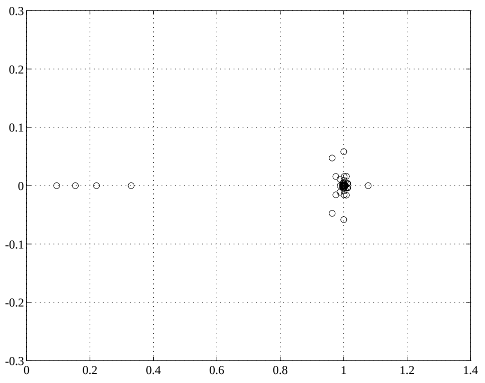
This application of approximate inverses, however, is much more difficult for general sparse matrices. Tools are required to extract the block tridiagonal structure of a matrix. Cuthill-McKee ordering may be used to initially reduce the profile of the matrix. A constant block-size partitioning may be found using the maximum half-bandwidth as the block-size. We can generally do better with a greedy approach, where a symmetrized profile is scanned and the smallest possible block-size is taken at each step. Simple heuristics, such as a minimum block-size, may be used to improve the greedy approach especially on pathologically bad cases. Finding a good block tridiagonal partitioning which has a small ratio of total block area to nonzeros is a challenging problem in itself.

The blocks themselves are stored in sparse format and a hyper-matrix data structure is required. The blocks may be variable sized, and the off-diagonal blocks may not be square. The implementation of this factorization for sparse matrices requires sparse-matrix by sparse-matrix subtraction and sparse-matrix by sparse-matrix multiplication, the latter with possibly nonsquare matrices.

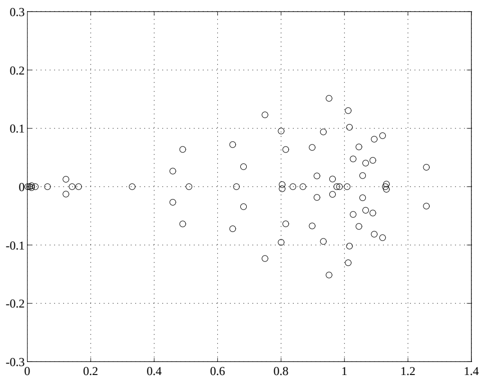
A major difficulty with applying these ideas to general sparse matrices is that the



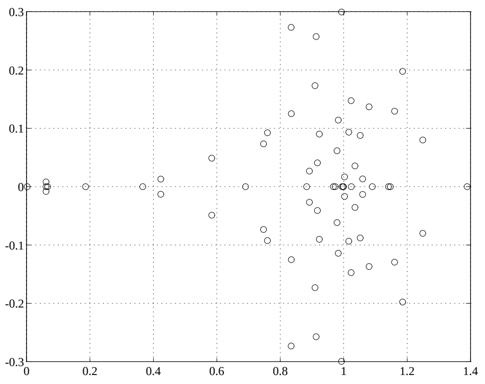
(a) no dropping, $n_o = 2$



(b) no dropping, $n_o = 4$



(c) *lfil* $= 10, n_o = 2$



(d) *lfil* $= 10, n_o = 4$

Figure 4: Eigenvalues of preconditioned system, WEST0067

diagonal blocks may be singular. In fact, it is very common for blocks to contain many rows or columns that are entirely zero. We have added arbitrary, small diagonal elements to remove the singularity. However, the choice of the size of this element is difficult: adding large elements reduces the accuracy of the factorization; adding small elements may make the factorization unstable. These drawbacks show that this popular application of approximate inverses is infeasible for general sparse matrices.

5.2 Improving a preconditioner

In all of our previous algorithms, we sought a matrix M to make AM close to the identity matrix. To be more general, we can seek instead an approximation to some matrix B. Thus, we consider the objective function

$$F(M) = \|B - AM\|_F^2 \tag{28}$$

in which B is some matrix to be defined. Once we find a matrix M whose objective function (28) is small enough, then the preconditioner for the matrix A is defined by

$$P = MB^{-1}.$$

This implies that B is a matrix which is easy to invert, or rather, that solving systems with B should be inexpensive. At one extreme when $B = A$, the best M is the identity matrix, but solves with B are expensive. At the other extreme, we find our standard situation which corresponds to $B = I$, and which is characterized by trivial B-solves but expensive to obtain M matrices. In between these two extremes there are a number of appealing compromises, perhaps the simplest being the block diagonal of A.

Another way of viewing the concept of approximately minimizing (28) is that of improving a preconditioner. Here B is an existing preconditioner, for example, an LU factorization. If the factorization gives an unsatisfactory convergence rate, it is difficult to improve it by attempting to modify the L and U factors. One solution would be to discard this factorization and attempt to recompute a fresh one, possibly with more fill-in. Clearly, this may be wasteful especially in the case when this process must be iterated a few times due to persistent failures.

For a numerical example of improving a preconditioner, we use approximate inverses to improve the block-diagonal preconditioners for the ORSREG1, ORSIRR1 and ORSIRR2 matrices. The experiments used dropping on numerical values with *lfil* = 10, and *droptol* = 0.001. In Table 7, *block size* is the effective size of the blocks in the inverse of the preconditioner, and *block precon* is the number of GMRES iterations to convergence when the block-diagonal preconditioner is used alone. The number of GMRES iterations is shown against the number of outer iterations used to improve the preconditioner.

	n	block size	block precon	1	2	3	4	5
ORSREG1	2205	21	117	49	59	95	73	71
ORSIRR1	1030	8	253	98	104	108	85	88
ORSIRR2	833	8	253	136	99	98	92	81

Table 7: Improving a preconditioner.

5.3 Use of approximate pseudo-inverses

There are instances where A is extremely ill-conditioned and possibly very indefinite. In such cases, one is tempted to get an approximate pseudo-inverse matrix, for example,

$$M \approx (A^T A)^{-1} A^T.$$

One could obtain the approximate inverse of $A^T A$ using one of the approaches described earlier. However, this approach will fail because as for A, we are likely to find a cluster of eigenvalues of $A^T A$ around the origin. The simplest remedy is simply to shift them away from the origin. Thus, we consider the algorithm:

Algorithm **5.1 Approximate pseudo-inverse**

```
1.  Select a positive $\alpha$ and compute $B = A^T A + \alpha I$
2.  Apply numerical dropping to $B$
3.  Compute an approximate inverse $C$ of $B$
4.  Compute $M = C A^T$
```

The choice of the scalar α impacts the performance in the following way. A large α will make step 3 very easy to perform, not only because the descent algorithms will converge quickly but also because for larger α, the more there are small elements in the actual inverse. On the other hand a large α will lead to a poor approximate inverse. Note that without dropping, and if the inversion in Step 3 is done exactly, then the eigenvalues of MA are given by

$$\frac{\sigma_i}{\sigma_i + \alpha}$$

where the σ_i's are the squares of the singular values of A. Thus, the largest singular values are mapped close to unity whereas the smallest ones are roughly divided by α. For example, we can say that all σ_i's that are greater than or equal to α will be mapped into the interval $[1/2, 1]$. Similarly, all σ_i's satisfying

$$\sigma_i \geq \frac{\alpha^2}{1-\alpha}$$

will be mapped into the interval $[\alpha, 1]$.

We show the results of a few numerical experiments with the WEST0067 matrix. One inner iteration was used, with a scaled identity initial guess to approximate the inverse of $A^TA + \alpha I$. Table 8 shows the results with no dropping. As clearly seen, the ultimate quality of the preconditioner decreases as α increases. However, the approximate inversion of $A^TA + \alpha I$ is much easier, as seen by the faster convergence of the residual norm. In Table 9, dropping is applied using *lfil* = 10.

6 Conclusion

Approximate inverse preconditioners can provide a good alternative to incomplete LU preconditioners for indefinite and highly nonsymmetric matrices. They do not suffer from instability and offer a high degree of parallelism. Another significant advantage is that, unlike ILU factorizations, previously computed preconditioners can be easily improved upon. On the negative side, approximate inverse preconditioners often demand more memory than standard ILU preconditioners. They are also more sensitive to numerical dropping and it is practically difficult to guarantee that will they produce a nonsingular preconditioner. It is possible that the best potential for approximate inverses is in combining them with other preconditioning techniques, e.g., to produce an improvement of a given block-diagonal preconditioner.

There are a few issues which need further investigation. First, is the issue of providing a good practical dropping strategy. Although we cannot avoid spoiling altogether if we want to preserve sparsity, we can define more robust and economical strategies that dynamically change in the outer iterations. Another issue which we have not fully investigated here is the factorized form of the approximate inverse preconditioner briefly discussed in Section 2.6. This form of the approximate inverse is more complex to implement but may offer an alternative that is less demanding in memory usage.

Acknowledgements The authors wish to acknowledge the support of the Minnesota Supercomputer Institute which provided the computer facilities and an excellent environment to conduct this research.

Number of GMRES iterations					
α	1	2	3	4	5
0.00	252	75	18	12	8
0.01	267	67	19	14	11
0.05	259	65	26	19	19
0.10	254	65	30	25	25

$\|I-(A^TA+\alpha I)C\|_F$					
α	1	2	3	4	5
0.00	4.53	3.34	2.50	1.92	1.06
0.01	4.47	3.12	1.83	0.64	0.06
0.05	4.22	2.37	0.75	0.06	0.00
0.10	3.94	1.76	0.35	0.01	0.00

Table 8: WEST0067, Approximate pseudo-inverse, no dropping.

Number of GMRES iterations					
α	1	2	3	4	5
0.00	302	261	340	†	†
0.05	305	320	436	†	†
0.10	280	209	377	481	†
0.25	225	148	†	387	477

$\|I-(A^TA+\alpha I)C\|_F$					
α	1	2	3	4	5
0.00	4.68	6.30	7.59	8.20	8.71
0.05	4.43	6.01	7.05	7.76	8.12
0.10	4.18	5.68	6.57	7.34	7.91
0.25	3.48	4.21	4.93	5.31	5.57

Table 9: WEST0067, Approximate pseudo-inverse, *lfil* = 10.

References

[1] O. Axelsson. Incomplete block matrix factorization preconditioning methods. The ultimate answer? *J. Comput. Appl. Math.,* 12 & 13 (1985), pp. 3-18.

[2] O. Axelsson. *Iterative Solution Methods.* Cambridge, Cambridge, 1994.

[3] O. Axelsson, S. Brinkkemper and V. P. Il'in. On some versions of incomplete block-matrix factorization iterative methods. *Lin. Alg. Appl.,* 58 (1984), pp. 3-15.

[4] O. Axelsson and B. Polman. On approximate factorization methods for block matrices suitable for vector and parallel processors. *Lin. Alg. Appl.,* 77 (1986), pp. 3-26.

[5] M. W. Benson. Iterative solution of large scale linear systems. M.Sc. Thesis (1973), Lakehead University, Thunder Bay, Ontario.

[6] M. W. Benson and P. O. Frederickson. Iterative solution of large sparse linear systems arising in certain multidimensional approximation problems. *Utilitas Math.,* 22 (1982), pp. 127-140.

[7] P. Concus, G. H. Golub and G. Meurant. Block preconditioning for the conjugate gradient method. *SIAM J. Sci. Stat. Comput.,* 6 (1985), pp. 309-332.

[8] J. D. F. Cosgrove, J. C. Díaz and A. Griewank. Approximate inverse preconditioning for sparse linear systems. *Intl. J. Comp. Math.*, 44 (1992), pp. 91-110.

[9] I. S. Duff, A. M. Erisman and J. K. Reid. *Direct Methods for Sparse Matrices.* Oxford, London, 1989.

[10] I. S. Duff, R. G. Grimes and J. G. Lewis. Users' guide for the Harwell-Boeing sparse matrix collection. TR/PA/92/86, CERFACS, Toulouse, 1992.

[11] T. Huckle and M. Grote. A new approach to parallel preconditioning with sparse approximate inverses. Manuscript SCCM-94-03, Scientific Computing and Computational Mathematics Program, Stanford University, Stanford, California, 1994.

[12] H. C. Elman. A stability analysis of incomplete LU factorizations. *Math. Comp.,* 47 (1986), pp. 191-217.

[13] M. Engleman. *FIDAP: Examples Manual*, Revision 6.0. Fluid Dynamics International, Evanston, Illinois, 1991.

[14] G. H. Golub and C. F. Van Loan. *Matrix Computations,* 2nd edition. John Hopkins, Baltimore, 1989.

[15] M. Grote and H. D. Simon. Parallel preconditioning and approximate inverses on the Connection Machine. In R. F. Sincovec, D. E. Keyes, L. R. Petzold and D. A. Reed, eds., *Parallel Processing for Scientific Computing,* vol. 2, pp. 519-523. SIAM, Philadelphia, Pennsylvania, 1993.

[16] A. S. Householder. *The Theory of Matrices in Numerical Analysis.* Dover, New York, 1964.

[17] L. Yu. Kolotilina and A. Yu. Yeremin. On a family of two-level preconditionings of the incomplete block factorization type, *Soviet J. Numer. Anal. Math. Model.,* 1 (1986), pp. 293-320.

[18] L. Yu. Kolotilina and A. Yu. Yeremin. Factorized sparse approximate inverse preconditionings I. Theory. *SIAM J. Matrix Anal. Appl.*, 14 (1993), pp. 45-58.

[19] L. Yu. Kolotilina and A. Yu. Yeremin. Factorized sparse approximate inverse preconditionings II. Solution of 3D FE systems on massively parallel computers. Research Report EM-RR 3/92, Elegant Mathematics, Inc. Bothell, Washington, 1992.

[20] L. Yu. Kolotilina and A. Yu. Yeremin. Incomplete block factorizations as preconditioners for sparse SPD matrices. Research Report EM-RR 6/92, Elegant Mathematics, Inc. Bothell, Washington, 1992.

[21] H. Manouzi. Personal communication, University of Laval, Quebec, 1993.

[22] V. Pan and J. Reif. Efficient parallel solution of linear systems. *Proc. 17th Annual ACM Symposium on Theory of Computing* (1985), pp. 143-152.

[23] V. Pan and R. Schreiber. An improved Newton iteration for the generalized inverse of a matrix, with applications. *SIAM J. Sci. Stat. Comput.,* 12 (1991), pp. 1109-1130.

[24] Y. Saad. ILUT: A dual threshold incomplete LU factorization. *SIAM J. Sci. Comput.,* to appear. Also, Technical Report UMSI 92/38, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, Minnesota, 1992.

[25] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM J. Sci. Comput.,* 14 (1993), pp. 461-469.

[26] Y. Saad. *Numerical Methods for Large Eigenvalue Problems.* Halstead Press, New York, 1992.