APPROXIMATE INVERSE PRECONDITIONERS VIA SPARSE-SPARSE ITERATIONS*

EDMOND CHOW† AND YOUSEF SAAD†

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 maintain sparsity; compared to previous methods, this is a natural way to determine the sparsity pattern of the approximate inverse. This paper describes Newton, 'global' and column-oriented algorithms, and discusses options for initial guesses, self-preconditioning, and dropping strategies. Some limited theoretical results on the properties and convergence of approximate inverses are derived. Numerical tests on problems from the Harwell-Boeing collection and the FIDAP fluid dynamics analysis package show the strengths and limitations of approximate inverses. Finally, some ideas and experiments with practical variations and applications are presented.

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

AMS subject classifications. 65F10, 65F35, 65F50, 65Y05

1. Introduction. The incomplete LU factorization preconditioners were originally developed for M-matrices that 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 which is the encounter of a zero pivot. However, there are other problems that are just as serious. Consider an incomplete factorization of the form

$$A = LU + E \tag{1.1}$$

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

$$L^{-1}AU^{-1} = I + L^{-1}EU^{-1}. \tag{1.2}$$

What is sometimes missed is the fact that the error matrix E in (1.1) is not as important as the *preconditioned* error matrix $L^{-1}EU^{-1}$ shown in (1.2) above. When the matrix A is diagonally dominant, L and U are typically well conditioned, and the size of $L^{-1}EU^{-1}$ remains confined within reasonable limits, typically with a 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 was studied by Elman [14] in a detailed analysis of ILU and MILU preconditioners for finite difference matrices. It can be observed experimentally

* This work was supported in part by the National Science Foundation under grant NSF/CCR-9214116 and in part by NASA under grant NAG2-904.

† Department of Computer Science and Minnesota Supercomputer Institute, University of Minnesota, 4-192 EE/CSci Bldg., 200 Union St., S.E., Minneapolis, Minnesota, 55455-0154 (chow@cs.umn.edu and saad@cs.umn.edu).

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.

One possible remedy that has been proposed is *stabilized* or *perturbed* incomplete factorizations, for example [15] and the references in [25]. A numerical comparison with these preconditioners will be given later. In this paper, we consider trying to find a preconditioner that does not require solving a linear system. For example, we can precondition the original system with a sparse matrix M that is a direct approximation to the inverse of A. Sparse approximate inverses are also necessary for incomplete block factorizations with large sparse blocks, as well as several other applications, also described later.

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]. Consider the minimization of

$$F(M) = \|I - AM\|_F^2. \tag{1.3}$$

to seek a right approximate inverse. 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 $I - AM$

$$F(M) = \|I - AM\|_F^2 = \sum_{j=1}^{n} \|e_j - Am_j\|_2^2 \tag{1.4}$$

in which e_j and m_j are the j-th columns of the identity matrix and of the matrix M, respectively. Thus, minimizing (1.4) is equivalent to minimizing the individual functions

$$f_j(m) = \|e_j - Am\|_2^2, \qquad j = 1, 2, \ldots, n. \tag{1.5}$$

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

The minimization in (1.5) is most often performed directly by prescribing a sparsity pattern for M and solving the resulting least squares problems. Grote and Simon [19] 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 [10] 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 work by Grote and Huckle [18], 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 [23] consider symmetric, positive definite systems and construct factorized sparse approximate inverse preconditioners which are also symmetric, positive definite. Each factor implicitly approximates the inverse of the lower triangular Cholesky factor of A. The structure of each factor is chosen to be the same as the structure of the lower triangular part of A. In their more recent work [24], fill-in elements may be added, and their locations are chosen such that the construction and application of the approximate inverse is not much more

expensive on a model hypercube computer. Preconditioners for general systems may be constructed by approximating the left and right factors separately.

This paper is organized as follows. In §2, we present several approximate inverse algorithms based on iterative procedures, as well as describe sparse-sparse implementation and various options. We derive some simple theoretical results for approximate inverses and the convergence behavior of the algorithms in §3. In §4, we show the strengths and limitations of approximate inverse preconditioners through numerical tests with problems from the Harwell-Boeing collection and the FIDAP fluid dynamics analysis package. Finally in §5, we present some ideas and experiments with practical variations and applications of approximate inverses.

2. Construction of the approximate inverse via iteration. The sparsity pattern of an approximate inverse of a general matrix should not be prescribed, since an appropriate pattern is usually not known beforehand. In contrast to the previous work described above, the locations and values of the nonzero elements are determined naturally as a side-effect of utilizing an iterative procedure to minimize (1.3) or (1.5). 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 §§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 (1.3), an approximate inverse may also be computed using an iterative process known as the method of Hotelling and Bodewig [20]. 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 [27] 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 [26], 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 §4.

2.2. Global iteration. In this section we describe a 'global' approach to minimizing (1.3), where we use a descent-type method, treating M as an unknown sparse matrix. The objective function (1.3) 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 (1.4) is associated is

$$\langle X, Y \rangle = \mathrm{tr}(Y^T X). \tag{2.1}$$

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

$$\alpha = \frac{\langle R, AG \rangle}{\langle AG, AG \rangle} = \frac{\mathrm{tr}(R^T AG)}{\mathrm{tr}\,((AG)^T AG)} \tag{2.2}$$

where $R = I - AM$ is the residual matrix. 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 some kind of numerical dropping, either to the new M or to the search direction G before taking the descent step. In the first case, the descent nature of the step is lost, i.e., it is no longer guaranteed that $F(M_{new}) \leq F(M)$, while in the second case, the fill-in in M is more difficult to control. We will discuss both these alternatives in §2.5.

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. In the simpler case where numerical dropping is applied to M, our global Minimal Residual algorithm will 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 α by (2.2)*
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 2.2. *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

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

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 $\mathrm{tr}(R^T AG)$ 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 §4.

2.3. Implementation of sparse mode MR and GMRES. We now describe column-oriented algorithms which consist of minimizing the individual objective functions (1.5). We perform this minimization by taking a sparse initial guess and solving approximately the n linear subproblems

$$Am_j = e_j, \qquad j = 1, 2, \ldots, n \tag{2.3}$$

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 (2.3) 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. Again, we assume numerical dropping is applied to M. In the GMRES version of the algorithm, we never use restarting since since n_i is typically very small. Also, a variant called FGMRES [31] which allows an arbitrary Arnoldi basis, is actually used in this case.

ALGORITHM 2.3. (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.            α_j := (r_j, Ar_j) / (Ar_j, Ar_j)
7.            m_j := m_j + α_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 [11].

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 transpose initial guess is more expensive to use because it is denser than the identity initial guess. However, for very indefinite systems, this guess immediately produces a symmetric positive definite preconditioned system, corresponding to the normal error equations. Depending on the structure of the inverse, a denser initial guess is often required to involve more of the matrix A in the computation. Interestingly, the cheaper the computation, the more it uses only 'local' information, and the less able it may be to produce a good approximate inverse.

The choice of initial guess also depends to some degree on 'self-preconditioning' which we describe next. Additional comments on the choice of initial guess will be presented there.

2.4. Self-preconditioning. The approximate solution of the linear subproblems (2.3) using an iterative method suffers from the same problems as solving the original problem if A is indefinite or poorly conditioned. However, the linear systems may be preconditioned with the columns that have already been computed. More precisely, each system (2.3) 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 that compute each column. On each subsequent outer iteration, the initial guess for each column is the previous result for that column. This technique usually results in much faster convergence of the approximate inverse.

Unfortunately with this approach, the parallelism of constructing the columns of the approximate inverse simultaneously is lost. However, 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. The performance of this variant usually 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.

Self-preconditioning is particularly valuable for very indefinite problems when combined with a scaled transpose initial guess; the initial preconditioned system AM_0 is positive definite, and the subsequent preconditioned systems somewhat maintain this property, even in the presence of numerical dropping. Self-preconditioning with a transpose initial guess, however, may produce worse results if the matrix A is very ill-conditioned. In this case, the initial worsening of the conditioning of the system is too severe, and the alternative scaled identity initial guess should be used instead. We have also found cases where self-preconditioning produces worse results, usually for positive definite problems; this is not surprising, since the minimizations would progress very well, only to be hindered by self-preconditioning with a poor approximate inverse in the early stages. Numerical evidence of these phenomena will be provided in §4.

Algorithm 2.4 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. We have also indicated where numerical dropping might be applied.

ALGORITHM 2.4. (Self-preconditioned Minimal Residual iteration)

Start: $M = M_0$
For outer $= 1, 2, \ldots, n_o$ *do*
    For each column $j = 1, \ldots, n$ *do*
        Define $s := m_j = Me_j$
        For inner $= 1, \ldots, n_i$ *do*
            $r := e_j - As$
            $z := Mr$
            $q := Az$
            $\alpha := \frac{(r,q)}{(q,q)}$
            $s := s + \alpha z$
            Apply numerical dropping to s
        End do
        Update j*-th column of* M: $m_j := s$
    End do
End do

In a FORTRAN 77 implementation, M is stored as n sparse vectors, each holding up to *lfil* entries. M is thus constructed in place.

The multiple outer iterations used in constructing the approximate inverse suggests the use of factorized updates. Factorized matrices can express denser matrices than the sum of their numbers of elements alone. 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

$$\min_{M_{i+1}} \|I - AM_1M_2 \cdots M_iM_{i+1}\|_F^2. \tag{2.4}$$

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

$$\min_{M_{2i+1}} \|I - M_{2i} \cdots M_4 M_2 A M_1 M_3 \cdots M_{2i+1}\|_F^2 \tag{2.5}$$

which alternate from left to right factors. This latter form is reminiscent of the symmetric form of Kolotilina and Yeremin [23].

Since the product $M_1 M_2 \cdots M_i$ 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.

2.5. Numerical dropping strategies. There are many options for numerical dropping. So far, to ease the presentation, we have only discussed the case where dropping is performed on the solution vectors or matrices. Section 2.5.1 discusses this case in more detail, while §2.5.2 discusses the case where dropping is applied to the search directions. In the latter case, the descent property of the algorithms is maintained.

2.5.1. Dropping in the solution. When dropping is performed on the solution, we have options for

1. when dropping is performed, and
2. which elements are dropped.

In the previous algorithms, we have made the first point precise; however, there are other alternatives. For example, dropping may be performed only after M or each column of M is computed. Typically this option is too expensive, but as a compromise, dropping may be performed at the end of a few inner iterations, before M is updated, namely before step 13 in Algorithm 2.4. Interestingly, we found experimentally that this option is not always better.

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

To address which elements are dropped, we can utilize a dual threshold strategy based on a drop tolerance, *droptol*, and the maximum number of elements per column, *lfil*. 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, i.e., the residual norm may increase after several steps, along with 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 $\hat{m}_j$ the perturbed column formed by adding the sparse column d 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

$$\|\hat{r}_j\|_2^2 = \|r_j\|_2^2 - 2(d, A^T r_j) + \|Ad\|_2^2. \tag{2.6}$$

Recall that $-2A^T r_j$ is the gradient of the function (1.5). 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

$$\|\hat{r}_j\|_2^2 = \|r_j\|_2^2 - 2m_{ij}(e_i, A^T r_j) + m_{ij}^2. \tag{2.7}$$

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

$$\|\hat{r}_j\|_2^2 - \|r_j\|_2^2 = -2m_{ij}(e_i, A^T r_j) + m_{ij}^2 \tag{2.8}$$

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 (2.8) 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}.

Another alternative is based on attempting to achieve maximum reduction in the function (2.8). Ideally, we wish to have

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

since this will achieve the 'optimal' reduction in (2.8)

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

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, and neither of these strategies completely eliminate spoiling.

2.5.2. Dropping in the search direction. Dropping may be performed on the search direction G in Algorithm 2.1, or equivalently in r_j and z in Algorithms 2.3 and 2.4 respectively. In these cases, the descent property of the algorithms is maintained, and the problem of spoiling is avoided.

Starting with a sparse initial guess, the allowed number of fill-ins is gradually increased at each iteration. For an MR-like algorithm, the search direction d is derived by dropping entries from the residual direction r. So that the sparsity pattern of the solution x is controlled, d is chosen to have the same sparsity pattern as x, plus one new entry, the largest entry in absolute value. No drop tolerance is used. Minimization is performed by choosing the step-length as

$$\alpha = \frac{(r, Ad)}{(Ad, Ad)}$$

and thus the residual norm for the new solution is guaranteed to be not more than the previous residual norm. In contrast to Algorithm 2.3, the residual may be updated with very little cost. The iterations may continue as long as the residual norm is larger than some threshold, or a set number of iterations may be used.

If A is indefinite, the normal equations residual direction $A^T r$ may be used as the search direction, or simply to determine the location of the new fill-in. It is interesting to note that the largest entry in $A^T r$ gives the greatest residual norm reduction in a one-dimensional minimization. When fill-in is allowed to increase gradually using this search direction, this technique becomes very similar to the adaptive selection scheme of [18]. The effect is also similar to self-preconditioning with a transpose initial guess.

At the end of each iteration, it is possible to use a second stage that exchanges entries in the solution with new entries if this causes a reduction in the residual norm. This is required if the sparsity pattern in the approximate inverse needs to change as the approximations progress. We have found this to be necessary, particularly for very unstructured matrices, but have not yet found a strategy that is genuinely effective [7]. As a result, approximations using numerical dropping in the solution are often better, even though the scheme just described has a stronger theoretical justification, similar to that of [18]. This also shows that the adaptive scheme of [18] may benefit from such an exchange strategy.

Algorithm 2.5 implements a Minimal Residual-like algorithm with this numerical dropping strategy. The number of inner iterations is usually chosen to be *lfil* or somewhat larger.

ALGORITHM 2.5. (Self-preconditioned MR algorithm with dropping in search direction)

1. *Start:* $M = M_0$
2. *For each column* $j = 1, \ldots, n$ *do*
3. *Define* $m_j = Me_j$
4. $r_j := e_j - Am_j$
5. *For* $inner = 1, 2, \ldots, n_i$ *do*
6. $t := Mr_j$
7. *Choose* d *to be* t *with the same pattern as* m_j*;*
   If $nnz(m_j) <$ lfil *then add one entry which is the largest remaining entry in absolute value*
8. $q := Ad$
9. $\alpha := \frac{(r_j, q)}{(q, q)}$
10. $m_j := m_j + \alpha d$
11. $r_j := r_j - \alpha q$
12. *End do*
13. *End do*

If dropping is applied to the unpreconditioned residual, then economical use of this approximate inverse technique is not limited to approximating the solution to linear systems with sparse coefficient matrices or sparse right-hand sides. An approximation may be found, for example, to a factorized matrix, or a dense operator which may only be accessed with a matrix-vector product. Such a need may arise, for instance, when preconditioning row projection systems. These approximations are not possible with other existing approximate inverse techniques.

We must mention here that any adaptive strategy such as this one for choosing the sparsity pattern makes massive parallelization of the algorithm more difficult. If, for

instance, each processor has the task of computing a few columns of the approximate inverse, it is not known beforehand which columns of A must be fetched into each processor.

2.6. 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 ($n_o = 1$ in Algorithm 2.5).

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

If Algorithm 2.4 is used, two sparse mode matrix-vector products are used, the first one for computing the residual; three are required if self-preconditioning is used. In Algorithm 2.5 the residual may be updated easily and stored, or recomputed as in Algorithm 2.4. Again, an additional product is required for self-preconditioning. The cost is simply nn_on_i times the number of these sparse mode matrix-vector multiplications. Each multiplication is cheap, depending on the sparseness of the columns in M. Dropping in the search directions, however, is slightly more expensive because, although the vectors are sparser at the beginning, it typically requires much more inner iterations (e.g., one for each fill-in).

In Newton iteration, two sparse matrix-sparse matrix products are required, although the convergence rate may be doubled with form of Chebyshev acceleration [28]. Global iterations without self-preconditioning require three matrix-matrix products. These costs are comparable to the column-oriented algorithms above.

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 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, typically to a level that is impractical to attain. This is a difficulty for all approximate inverse preconditioners, except for triangular factorized forms described in [23].

The drawback of using M that is possibly singular is the need to check the solution, or the actual residual norm at the end of the linear iterations. In practice, we have not noticed premature terminations due to a singular preconditioned system, and this is likely a very rare event.

We begin this section with an easy proposition.

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

$$\|I - AM\| < 1 \tag{3.1}$$

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

Proof. The result follows immediately from the equality

$$AM = I - (I - AM) \equiv I - N \tag{3.2}$$

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

$$\|I - D_1 A M D_2\| < 1 \tag{3.3}$$

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

Proof. Applying the previous result to $A' = D_1 A$ and $M' = M D_2$, implies that $M' = M D_2$ 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

$$\|e_j - A m_j\| \leq \tau. \tag{3.4}$$

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

$$\|e_j - A m_j\|_1 \leq \tau_j \tag{3.5}$$

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

PROPOSITION 3.3. *Assume that the condition (3.5) 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*

$$|\lambda - 1| < \tau. \tag{3.6}$$

2. *If $\tau < 1$, then M is nonsingular.*

3. *If any k columns of M, with $k \leq n$, are linearly dependent then at least one residual $e_j - A m_j$ 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 $r_{:,j} = e_j - A m_j$. The column version of Gershgorin's theorem, see e.g., [30, 17], 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

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

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

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

from which we get

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

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

$$\sum_{i=1}^{k} \alpha_i m_i = 0. \tag{3.7}$$

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 (3.7) 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,

$$\|x\|_1 \leq \sqrt{n}\|x\|_2 \quad \text{and} \quad \|x\|_1 \leq n\|x\|_\infty. \tag{3.8}$$

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 [18].

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 [10] 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 (3.5) is made.

PROPOSITION 3.4. *Let $B = A^{-1}$ and assume that a given element b_{ij} of B satisfies the inequality*

$$|b_{ij}| > \tau_j \max_{k=1,n} |b_{ik}|, \tag{3.9}$$

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 (3.9) 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 3.5. *Let τ de defined as in Proposition 3.3. 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 (2.3) 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 (2.3), 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

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

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 2.3, 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 step, will have a residual of the form

$$R' = I - AM' = \rho(AM)R = \rho(AM)(I - AM) \tag{3.10}$$

in which $\rho(t) = 1 - ts(t)$ is the residual polynomial. Multiplying (3.10) 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-oriented 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

$$e_j - Am'_j = \rho_j(AM)(e_j - Am_j) \tag{3.11}$$

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 (3.11) 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-oriented 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

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

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

$$w^TAM' = w^TAM = 0,$$

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.

3.4.1. Global MR iterations. When self-preconditioning is used in the global MR iteration, the matrix which defines the search direction is $Z_k = M_k R_k$, where R_k is the current residual. Therefore, the algorithm (without dropping) is as follows.

1. $R_k := I - AM_k$
2. $Z_k := MR_k$
3. $\alpha_k := \frac{\langle R_k, AZ_k \rangle}{\langle AZ_k, AZ_k \rangle}$
4. $M_{k+1} := M_k + \alpha_k Z_k$

At each step the new residual matrix R_{k+1} satisfies the relation

$$\begin{aligned} R_{k+1} &= I - AM_{k+1} \\ &= I - A(M_k + \alpha_k Z_k) \\ &= R_k - \alpha_k AZ_k. \end{aligned}$$

Our first observation is that R_k is a polynomial in R_0. This is because, from the above relation,

$$\begin{aligned} R_{k+1} &= R_k - \alpha_k AM_k R_k \\ &= R_k - \alpha_k (I - R_k) R_k \\ &= (1 - \alpha_k) R_k + \alpha_k R_k^2. \end{aligned} \tag{3.12}$$

Thus, by induction,

$$R_{k+1} = p_{2^k}(R_0)$$

in which p_j is a certain polynomial of degree j. Throughout this section we use the notation

$$B_k \equiv AM_k = I - R_k. \tag{3.13}$$

The following recurrence is easy to infer from (3.12),

$$B_{k+1} = B_k + \alpha_k B_k (I - B_k). \tag{3.14}$$

Note that B_{k+1} is also a polynomial of degree 2^k in B_0. In particular, if the initial B_0 (equivalently R_0) is symmetric, then all subsequent R_k's and B_k's are also symmetric. This is achieved when the initial M is a multiple of A^T, i.e., when

$$M_0 = \alpha_0 A^T .$$

We are now ready to prove a number of simple results.

PROPOSITION 3.6. *If the self-preconditioned MR iteration converges, then it does so quadratically.*

Proof. Define for any α,

$$R(\alpha) = (1 - \alpha) R_k + \alpha R_k^2.$$

Recall that α_k achieves the minimum of $\|R(\alpha)\|_F$ over all α's. In particular,

$$\begin{aligned} \|R_{k+1}\|_F &= \min_{\alpha} \|R(\alpha)\|_F \\ &\leq \|R(1)\|_F = \|R_k^2\|_F \\ &\leq \|R_k\|_F^2. \end{aligned} \tag{3.15}$$

This proves quadratic convergence at the limit. □

The following proposition is a straightforward generalization to the matrix case of a well-known result [13] concerning the convergence of the vector Minimal Residual iteration.

PROPOSITION 3.7. *Assume that at a given step k, the matrix B_k is positive definite. Then, the following relation holds,*

$$\|R_{k+1}\|_F \leq \|R_k\|_F \ \sin \angle(R_k, B_k R_k) \tag{3.16}$$

with

$$\cos \angle(R, BR) \equiv \frac{\langle R, BR\rangle}{\|R\|_F \|BR\|_F} \geq \frac{\mu_{min}(B)}{\sigma_{max}(B)} \tag{3.17}$$

in which $\mu_{min}(B)$ is the smallest eigenvalue of $\frac{1}{2}(B+B^T)$ and $\sigma_{max}(B)$ is the largest singular value of B.

Proof. Start with

$$\|R_{k+1}\|_F^2 = \langle R_{k+1}, R_k - \alpha_k A Z_k\rangle = \langle R_{k+1}, R_k\rangle - \alpha_k \langle R_{k+1}, AZ_k\rangle .$$

By construction, the new residual R_{k+1} is orthogonal to AZ_k, in the sense of the $\langle\cdot,\cdot\rangle$ inner product, and as a result, the second term in the right-hand side of the above equation vanishes. Noting that $AZ_k = B_k R_k$, we thus obtain

$$\begin{aligned}
\|R_{k+1}\|_F^2 &= \langle R_k - \alpha_k B_k R_k, R_k\rangle \\
&= \langle R_k, R_k\rangle - \alpha_k \langle B_k R_k, R_k\rangle \\
&= \|R_k\|_F^2 \left(1 - \frac{\langle B_k R_k, R_k\rangle}{\langle B_k R_k, B_k R_k\rangle} \frac{\langle B_k R_k, R_k\rangle}{\langle R_k, R_k\rangle}\right) \\
&= \|R_k\|_F^2 \left(1 - \left(\frac{\langle B_k R_k, R_k\rangle}{\|R_k\|_F \|B_k R_k\|_F}\right)^2\right).
\end{aligned} \tag{3.18}$$

The result (3.16) follows immediately.

To derive (3.17), note that

$$\langle R, BR\rangle = \sum_{i=1}^{n} (Br_i, r_i) \tag{3.19}$$

in which r_i is the i-th column of R, and similarly

$$\langle BR, BR\rangle = \sum_{i=1}^{n} (Br_i, Br_i). \tag{3.20}$$

For each i we have

$$(Br_i, r_i) \geq \lambda_{min}\left(\frac{B+B^T}{2}\right) |r_i\|^2$$

and

$$\|Br_i\| \leq \sigma_{max}(B) \|r_i\|.$$

The result follows after substituting these relations in the ratio (3.17). □

Note that because of (3.16) the Frobenius norm of R_{k+1} is bounded from above for all k, specifically, $\|R_{k+1}\|_F \leq \|R_0\|_F$ for all k. A consequence is that the largest singular value of $B_{k+1} = I - R_{k+1}$ is also bounded from above. Specifically, we have

$$\sigma_{max}(B_k) = \sigma_{max}(I - R_k) \leq 1 + \sigma_{max}(R_k) \leq 1 + \|R_k\|_F \leq 1 + \|R_0\|_F \ .$$

Assume now that $M_0 = \alpha_0 A^T$ so that all matrices B_k are symmetric. If in addition, each B_k is positive definite with its smallest eigenvalue bounded from below by a positive number, then B_k will converge to the identity matrix. Further, the convergence will be quadratic at the limit.

3.4.2. Column-oriented MR iterations. The convergence result may be extended to 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

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

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

$$\|r^{new}\|_2 = \|r\|_2 \sin \angle(r, Br) \tag{3.21}$$

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 (3.21) 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

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

This results in the following statement.

PROPOSITION 3.8. *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.,*

$$\|r_j^{new}\|_2 \ \leq \ \|I - AM\|_2 \ \|r_j\|_2 \tag{3.22}$$

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

$$\|R_{k+1}\|_F \ \leq \ \|R_k\|_F^2. \tag{3.23}$$

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

Proof. Inequality (3.22) was proved above. To prove quadratic convergence, we first transform this inequality by using the fact that $\|X\|_2 \leq \|X\|_F$ to obtain

$$\|r_j^{new}\|_2 \ \leq \ \|R_{k,j}\|_F \ \|r_j\|_2.$$

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.

These follow from the fact that the algorithms deliver an approximate column which has a smaller residual than what we obtain with one inner step MR.

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 §2 were performed with matrices from the Harwell-Boeing sparse matrix collection [12], and matrices extracted from example problems in the FIDAP fluid dynamics analysis package [16]. The matrices were scaled so that the 2-norm of each column is unity. 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 (1.3). Even though this is the function that we minimize, we see that it is not always a reliable measure of GMRES convergence. All the results are shown as the outer iterations progress. In Algorithm 2.4 (dropping in solution vectors) one inner iteration was used unless otherwise indicated; in algorithm 2.5 (dropping in residual vectors) one additional fill-in was allowed per iteration. Various codes in FORTRAN 77, C++, and Matlab were used, and run in 64-bit precision on Sun workstations and a Cray C90 supercomputer.

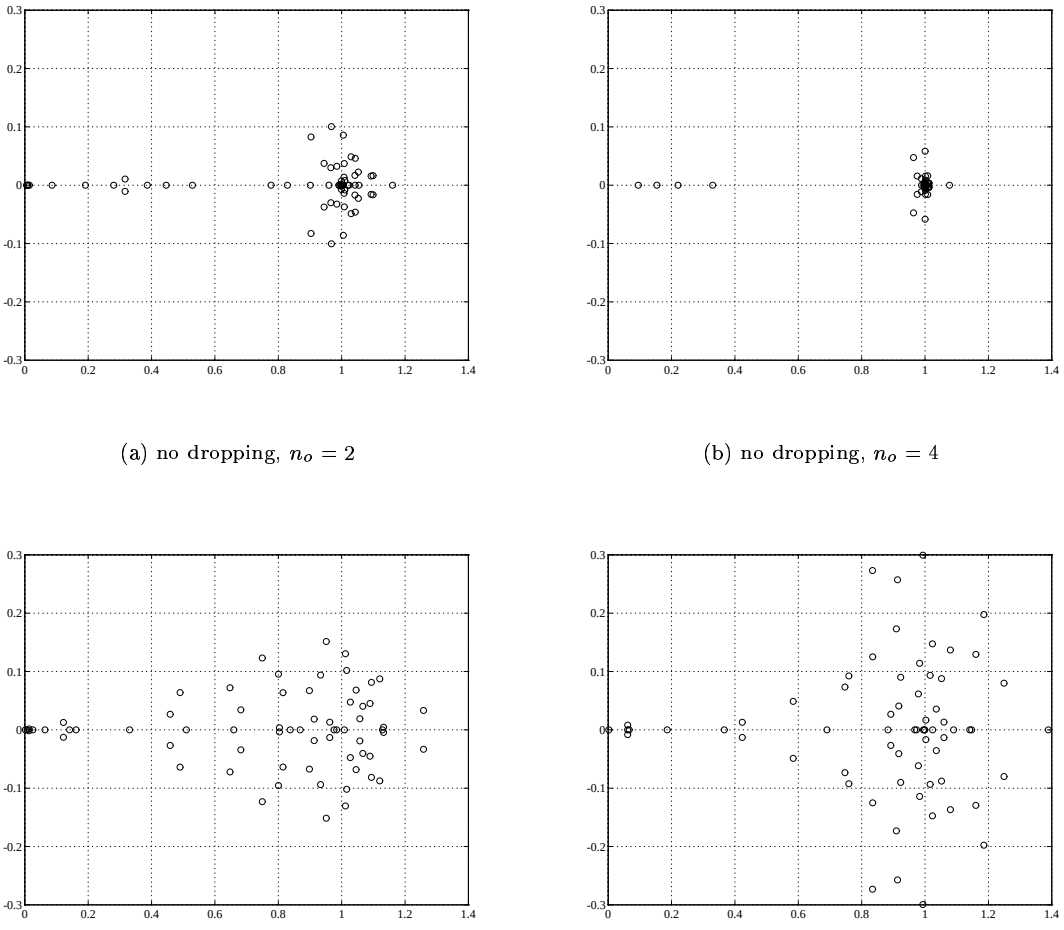
We begin with a comparison of Newton, 'global' and column-oriented iterations. Our early numerical experiments showed that in practice, Newton iteration converges very slowly initially and is more adversely affected by numerical dropping. Global iterations were also worse than column-oriented iterations, perhaps because a single α defined by (2.2) is used, as opposed to one for each column in the column-oriented case. Table 4.1 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. The MR iteration used self-preconditioning with a scaled transpose initial guess. Dropping based on numerical values in the intermediate solutions was performed on a column-by-column basis, although in the Newton and global iterations 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 4.1 was not typical.

TABLE 4.1
WEST0067: Newton, global, and column MR iterations.

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

The eigenvalues of the preconditioned WEST0067 matrix are plotted in Fig. 4.1, both with and without dropping, using column-oriented MR iterations. 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.



(a) no dropping, $n_o = 2$

(b) no dropping, $n_o = 4$

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

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

FIG. 4.1. *Eigenvalues of preconditioned system, WEST0067*

Next we show some results on matrices that arise from solving the fully-coupled Navier-Stokes equations. The matrices were extracted from the FIDAP package at the final nonlinear iteration of each problem in their Examples collection. The matrices are from 2-dimensional finite element discretizations using 9-node quadrilateral elements for velocity and temperature, and linear discontinuous elements for pressure.

Table 4.2 lists some statistics about all the positive definite matrices from the collection. The combination of ill-conditioning and indefiniteness of the other matrices was too difficult for our methods, and their results are not shown here.

All the matrices are also symmetric, except for Example 7. None of the matrices could be solved with ILU(0) or ILUT [32], a threshold incomplete LU factorization,

TABLE 4.2
FIDAP Example matrices.

Example	n	nnz	
3	1821	52685	Flow past a circular cylinder
7	1633	54543	Natural convection in a square cavity
9	3363	99471	Jet impingement in a narrow channel
10	2410	54840	2D flow over multiple steps in a channel
13	2568	75628	Axisymmetric flow through a poppet valve
15	6867	98671	2D spin up of a liquid in an annulus
33	1733	22189	2D radiation heat transfer in a cavity

even with large amounts of fill-in. Our experience with these matrices is that they produce unstable L and U factors in (1.2).

Table 4.3 shows the results of preconditioning with the approximate inverse, using dropping in the residual search direction. Since the problems are very ill-conditioned but positive definite, a scaled identity initial guess with no self-preconditioning was used. The columns show the results as the iterations and fill-in progress. 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.

TABLE 4.3
Number of GMRES iterations vs. lfil.

Example	10	20	30	40	50	60	70
3	159	133	47	42	40	38	38
7	33	23	18	17	14	14	14
9	203	117	67	51	47	41	41
10	438	191	107	107	81	66	63
13	56	39	34	28	26	24	24
15	103	83	62	54	53	52	52
33	249	105	86	44	40	39	39

For comparison, we solve the same problems using perturbed ILU factorizations. Perturbations are added to the inverse of diagonal elements to avoid small pivots, and thus control the size of the elements in the L and U factors. We use a two-level block ILU strategy called BILU(0)–SVD(α), that uses a modified singular value decomposition to invert the blocks. When a block $A = U\Sigma V^T$ needs to be inverted, it is replaced by the perturbed inverse $M = V\bar{\Sigma}^{-1}U^T$, where $\bar{\Sigma}$ is Σ with its singular values thresholded by $\alpha\sigma_1$, a factor of the largest singular value.

Table 4.4 shows the results, using a block size of 4. The method is very successful for this set of problems, showing results comparable to approximate inverse preconditioning, but with less work to compute the preconditioner. None of the problems converged, however, for $\alpha = 0.1$, and there was not one α that gave the best result for all problems.

We now show our main results in Table 4.5 for several standard matrices in the Harwell-Boeing collection. All the problems are nonsymmetric and indefinite, except for SHERMAN1 which is symmetric, negative definite. In addition, SAYLR3 is singular. SHERMAN2 was reordered with reverse Cuthill-McKee to attempt to change the sparsity pattern of the inverse. Again, we show the number of GMRES iterations to convergence against the number of outer iterations used to compute the approximate inverse. A scaled transpose initial guess was used. When columns in the initial guess contained more than *lfil* nonzeros, dropping was applied to the guess.

TABLE 4.4
BILU(0)–SVD(α) preconditioner.

Example	α= 0.3	α= 1.0
3	†	170
7	19	39
9	28	72
10	66	140
13	20	40
15	†	119
33	†	149

Numerical dropping was applied to the intermediate vectors in the solution, retaining *lfil* nonzeros and using no drop tolerance.

TABLE 4.5
Number of GMRES(20) iterations vs. n_o.

Matrix	n	ILU(0)	g/m	p/u	*lfil*	n_i	1	2	3	4	5
PORES2	1244	44	m	p	30	2	†	†	52	30	19
PORES3	532	38	m	u	10	1	†	†	421	150	112
SHERMAN1	1000	32	m	u	10	1	224	187	96	74	60
SHERMAN2	1080	8	m	p	50	1	†	†	147	46	136
SHERMAN3	5005	56	m	u	10	1	499	363	239	192	148
SHERMAN4	1104	22	m	u	10	1	87	69	43	42	41
SHERMAN5	3312	22	g	p	20	2	†	148	107	70	60
SAYLR3	1000	†	m	u	10	1	223	188	96	74	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
NNC666	666	†	m	p	50	2	†	†	427	147	173

g/m = GMRES or MR
p/u = self-preconditioned or unself-preconditioned

For problems SHERMAN2, WEST0989, GRE1107 and NNC666, the results become worse as the outer iterations progress. This spoiling effect is due to the fact that the descent property is not maintained when dropping is applied to the intermediate solutions. This is not the case when dropping is applied to the search direction, as seen in Table 4.3.

Except for SAYLR3, the problems that could not be solved with ILU(0) also could not be solved with BILU(0)–SVD(α), nor with ILUTP, a variant of ILUT more suited to indefinite problems since it uses partial pivoting to avoid small pivots [29]. ILUTP also substitutes $(10^{-4} + \delta)$ times the norm of the row when it is forced to take a zero pivot, where δ is the drop tolerance. ILU factorization strategies simply do not apply in these cases.

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 apart from the comments we have already made for selecting an initial guess and whether or not to use self-preconditioning, there is no general set of parameters that works best for constructing the approximate inverse.

The following two tables illustrate some different behaviors that can be seen for three very different matrices. LAPL0324 is a standard symmetric positive definite 2-D Laplacian matrix of order 324. WEST0067 and PORES3 are both indefinite;

WEST0067 has very little structure, while PORES3 has a symmetric pattern. Table 4.6 shows the number of GMRES(20) iterations and Table 4.7 shows the Frobenius norm of the residual matrix against the number of outer iterations that were used to compute the approximate inverse.

TABLE 4.6
Number of GMRES(20) iterations vs. n_o.

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 4.7
$\|I - AM\|_F$ *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

5. Practical variations and applications. Approximate inverses can be expensive to compute for very large and difficult problems. However, their best potential is in combinations 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 4.5 that approximate inverses work well with small matrices, most likely because of their local nature. In the next section, we show how smaller approximate inverses may be used effectively in incomplete block tridiagonal factorizations.

5.1. Incomplete block tridiagonal factorizations. Incomplete factorization of block tridiagonal matrices has been studied extensively in the past decade [1, 2, 3, 4, 9, 21, 22], but there have been very few numerical results reported for general sparse systems. Banded or polynomial approximations to the pivot blocks have been primarily used in the past, for systems arising from finite difference discretizations of partial differential equations. There are currently very few options for incomplete

factorizations of block matrices that require approximate inversion of general large, sparse blocks.

The inverse-free form of block tridiagonal factorization is

$$M = (D^{-1} - L_A)D(D^{-1} - U_A) \tag{5.1}$$

where L_A is the strictly lower block tridiagonal part of the coefficient matrix A, U_A is the corresponding upper part, and D is a block diagonal matrix whose blocks D_i are defined by the recurrence

$$D_i = (A_{i,i} - A_{i,i-1}D_{i-1}A_{i-1,i})^{-1} \tag{5.2}$$

starting with $D_0 = 0$. The factorization is made incomplete by using approximate inverses rather than the exact inverse in (5.2). This inverse-free form only requires matrix-vector multiplications in the preconditioning operation.

We illustrate the use of approximate inverses in these factorizations with Example 19 from FIDAP, the largest nonsymmetric matrix in the collection ($n = 12005, nnz = 259879$). The problem is an axisymmetric 2D developing pipe flow, using the two-equation k-ϵ model for turbulence. A constant block size of 161 was used, the smallest block size that would yield a block tridiagonal system (the last block has size 91). Since the matrix arises from a finite element problem, a more careful selection of the partitioning may yield better results. In the worse case, a pivot block may be singular; this would cause difficulties for several approximate inverse techniques such as [23] if the sparsity pattern is not augmented. In our case, a minimal residual solution in the null space would be returned.

Since the matrix contains different equations and variables, the rows of the system were scaled by their 2-norms, and then their columns were scaled similarly. A Krylov subspace size for GMRES of 50 was used. Table 5.1 first illustrates the solution with BILU(0)–SVD(α) with a block size of 5 for comparison. The infinity-norm condition of the inverse of the block LU factors is estimated with $\|(LU)^{-1}e\|_\infty$, where e is the vector of all ones. This condition estimate decreases dramatically as the perturbation is increased.

TABLE 5.1
Example 19, BILU(0)–SVD(α).

α	condition estimate	GMRES steps
0.000	1.e46	†
0.001	7.e47	†
0.010	8.e26	†
0.050	3.e08	†
0.100	3.e05	†
0.500	129.	87
1.000	96.	337

Table 5.2 shows the condition estimate, number of GMRES steps to convergence, timings for setting up the preconditioner and the iterations, and the number of nonzeros in the preconditioner. The method BTIF denotes the inverse-free factorization (5.1), and may be used with several approximate inverse techniques. MR-s(*lfil*) and MR-r(*lfil*) denote the minimal residual algorithm using dropping in the solution and residual vectors, respectively, and LS is the least squares solution using the sparsity pattern of the pivot block as the sparsity pattern of the approximate inverse. The

MR methods used *lfil* of 10, and specifically, 3 outer and 1 inner iteration for MR-s, and *lfil* iterations for MR-r. Self-preconditioning and transpose initial guesses were used. LS used the `DGELS` routine in LAPACK to compute the least squares solution. The experiments were carried out on one processor of a Sun Sparcstation 10. The code for constructing the incomplete block factorization is somewhat inefficient in two ways: it transposes the data structure of the pivot block and the inverse (to use column-oriented algorithms), and it counts the number of nonzeros in the sparse matrix-matrix multiplication before performing the actual multiplication.

TABLE 5.2
Example 19, block tridiagonal incomplete factorization.

	cond.	GMRES	CPU time (s)			nonzeros
	est.	steps	precon	solve	total	precon
BILU(0)–SVD(0.5)	129.	87	15.98	143.18	159.16	983 875
BTIF–MR-s(10)	119.	186	56.20	113.41	169.61	120 050
BTIF–MR-r(10)	92.	239	77.85	142.80	220.65	120 050
BTIF–MR-s(5)	382.	328	44.58	186.34	230.92	60 025
BTIF–MR-r(5)	93.	527	34.86	295.51	330.37	60 025
BTIF–LS	5.e95	†	290.02	†	†	453 605

The timings show that BTIF–MR-s(10) is comparable to BILU(0)–SVD(0.5) but uses much less memory. Although the actual number of nonzeros in the matrix is 259 879, there were 39 355 block nonzeros required in BILU(0), and therefore almost a million entries that needed to be stored. BILU(0) required more time in the iterations because the preconditioner was denser, and needed to operate with much smaller blocks. The MR methods produced approximate inverses that were sparser than the original pivot blocks. The LS method produces approximate inverses with the same number of nonzeros as the pivot blocks, and thus required greater storage and computation time. The solution was poor, however, possibly because the second, third, and fourth pivot blocks were poorly approximated. In these cases, at least one local least squares problem had linearly independent columns. No pivot blocks were singular.

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

$$F(M) = \|B - AM\|_F^2 \tag{5.3}$$

in which B is some matrix to be defined. Once we find a matrix M whose objective function (5.3) 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 (5.3) 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 5.3, *block size* is the block size of the block-diagonal preconditioner, and *block precon* is the number of GMRES iterations required for 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.

TABLE 5.3
Improving a preconditioner.

Matrix	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

Besides these applications, we have used approximate inverse techniques for several other purposes. Like in (5.3), we can generalize our problem to minimize

$$f(x) = \|b - Ax\|_F^2 \tag{5.4}$$

where b is a right-hand side and x is an approximate sparse solution. The right-hand side b does not need to be sparse if dropping is used in the search direction. Sparse approximate solutions to linear systems may be used in forming preconditioners, for example, to form a sparse approximation to a Schur complement or its inverse. See [7] and [8] for more details.

6. Conclusion. This paper has described an approach for constructing approximate inverses via sparse-sparse iterations. The sparse mode iterations are designed to be economical, however, their cost is still not competitive with ILU factorizations. Other approximate inverse techniques that use adaptive sparsity selection schemes also suffer from the same drawback. However, several examples show that these preconditioners may be applied to cases where other existing options, such as perturbed ILU factorizations, fail.

More importantly, our conclusion is that the greatest value of sparse approximate inverses may be their use in conjunction with other preconditioners. We demonstrated this with incomplete block factorizations and improving block diagonal preconditioners. They have also been used successfully for computing sparse solutions when constructing preconditioners, and one variant has the promise of computing approximations to operators that may be effectively dense.

Two limitations of approximate inverses in general are their local nature, and the question of whether or not an inverse can be approximated by a sparse matrix. Their local nature suggests that their use is more effective on small problems, for

example the pivot blocks in incomplete factorizations, or else large amounts of fill-in must be allowed. In current work, Tang [33] couples local inverses over a domain in a Schur complement approach. Preliminary results are consistently better than when the approximate inverse is applied directly to the matrix, and its effect has similarities to [7].

In trying to ensure that there is enough variation in the entries of the inverse for a sparse approximation to be effective, we have tried reordering to reduce the profile of a matrix. In a very different technique, Wan *et. al.* [34] compute the approximate inverse in a wavelet space, where there may be greater variations in the entries of the inverse, and thus permit a better sparse approximation.

Acknowledgments. The authors are grateful to the referees for their comments which substantially improved the quality of this paper. The authors also wish to acknowledge the support of the Minnesota Supercomputer Institute which provided the computer facilities and an excellent environment to conduct this research.

REFERENCES

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

[2] ———, *Iterative Solution Methods*, Cambridge University Press, 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*, Master's Thesis, Lakehead University, Thunder Bay, ON, 1973.

[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] E. Chow and Y. Saad, *Approximate inverse techniques for block-partitioned matrices*, Tech. Report UMSI 95/13, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, MN, 1995.

[8] ———, ILUS: *an incomplete* LU *factorization for matrices in sparse skyline format*, Tech. Report UMSI 95/78, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, MN, 1995.

[9] 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.

[10] 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.

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

[12] I. S. Duff, R. G. Grimes, and J. G. Lewis, *Sparse matrix test problems*, ACM Trans. Math. Softw., 15 (1989), pp. 1–14.

[13] S. C. Eisenstat, H. C. Elman, and M. H. Schultz, *Variational iterative methods for nonsymmetric systems of linear equations*, SIAM J. Num. Anal., 20 (1983), pp. 345–357.

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

[15] ———, *Relaxed and stabilized incomplete factorizations for non-self-adjoint linear systems*, BIT, 29 (1989), pp. 890–915.

[16] M. Engelman, *FIDAP: Examples Manual, Revision 6.0*, Fluid Dynamics International, Evanston, IL, 1991.

[17] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The John Hopkins University Press, Baltimore, MD, second ed., 1989.

[18] M. Grote and T. Huckle, *Parallel preconditioning with sparse approximate inverses*, SIAM J. Sci. Comput., (to appear).

[19] M. Grote and H. D. Simon, *Parallel preconditioning and approximate inverses on the Connection Machine*, in Parallel Processing for Scientific Computing, R. F. Sincovec, D. E.

Keyes, L. R. Petzold, and D. A. Reed, eds., SIAM, Philadelphia, PA, 1993, pp. 519–523. Vol. 2.

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

[21] 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.

[22] ———, *Incomplete block factorizations as preconditioners for sparse* SPD *matrices*, Tech. Report EM-RR 6/92, Elegant Mathematics, Inc., Bothell, WA, 1992.

[23] ———, *Factorized sparse approximate inverse preconditionings* I. *Theory*, SIAM J. Mat. Anal., 14 (1993), pp. 45–58.

[24] ———, *Factorized sparse approximate inverse preconditionings* II. *Solution of* 3D FE *systems on massively parallel computers*, Intl. J. High Speed Computing, 7 (1995), pp. 191–215.

[25] M. MAGOLU, *Modified block-approximate factorization strategies*, Numer. Math., 61 (1992), pp. 91–110.

[26] H. MANOUZI. Private communication, 1993.

[27] V. PAN AND J. REIF, *Efficient parallel solution of linear systems*, in Proc. 17th Annual ACM Symposium on Theory of Computing, 1985, pp. 143–152.

[28] 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.

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

[30] ———, *Numerical Methods for Large Eigenvalue Problems*, Halstead Press, New York, 1992.

[31] ———, *A flexible inner-outer preconditioned* GMRES *algorithm*, SIAM J. Sci. Stat. Comput., 14 (1993), pp. 461–469.

[32] ———, ILUT: *a dual threshold incomplete* ILU *factorization*, Num. Lin. Alg. Appl., 1 (1994), pp. 387–402.

[33] W.-P. TANG, *Effective sparse approximate inverse preconditioners.* In preparation.

[34] W. L. WAN, T. F. CHAN, B. SMITH, AND W.-P. TANG, *Fast wavelet-based sparse approximate inverse preconditioners.* In preparation.