SIAM J. SCI. COMPUT.
Vol. 22, No. 4, pp. 1318–1332

ROBUST APPROXIMATE INVERSE PRECONDITIONING FOR THE CONJUGATE GRADIENT METHOD*

MICHELE BENZI†, JANE K. CULLUM‡, AND MIROSLAV TŮMA§

Abstract. We present a variant of the AINV factorized sparse approximate inverse algorithm which is applicable to any symmetric positive definite matrix. The new preconditioner is breakdown-free and, when used in conjunction with the conjugate gradient method, results in a reliable solver for highly ill-conditioned linear systems. We also investigate an alternative approach to a stable approximate inverse algorithm, based on the idea of diagonally compensated reduction of matrix entries. The results of numerical tests on challenging linear systems arising from finite element modeling of elasticity and diffusion problems are presented.

Key words. sparse linear systems, finite element matrices, preconditioned conjugate gradients, factorized sparse approximate inverses, incomplete conjugation, stabilized AINV, diagonally compensated reduction

AMS subject classifications. Primary, 65F10, 65N22, 65F50; Secondary, 15A06

PII. S1064827599356900

1. Introduction.

We consider the solution of sparse linear systems $Ax = b$, where A is a symmetric and positive definite (SPD) matrix, by the preconditioned conjugate gradient method. In the last few years there has been considerable interest in explicit preconditioning techniques based on directly approximating A^{-1} with a sparse matrix M; see, e.g., [7], [8], [16], [18], [23], [24], [27], [31], and the recent survey [10]. Sparse approximate inverses have been shown to result in good rates of convergence of the preconditioned iteration (comparable to those obtained with incomplete factorization methods) while being well suited for implementation on vector and parallel architectures; see, e.g., [6], [9], [12], [21].

Although the main motivation for the development of sparse approximate inverse preconditioners comes from parallel processing, it is becoming clear that these techniques are also of interest because of their robustness. Sparse approximate inverses are often applicable to difficult problems where other preconditioners may break down [4]. For instance, incomplete factorization preconditioners, while widely popular and fairly robust, are not always reliable, in that the incomplete factorization process may

*Received by the editors June 4, 1999; accepted for publication (in revised form) June 19, 2000; published electronically October 25, 2000.

http://www.siam.org/journals/sisc/22-4/35690.html

†Computer Research and Applications Group (CIC-3), MS B256, Los Alamos National Laboratory, Los Alamos, New Mexico 87545. Current address: Department of Mathematics and Computer Science, Emory University, Atlanta, GA 30322 (benzi@mathcs.emory.edu). The work of the first author was supported in part by Department of Energy grant W-7405-ENG-36 with Los Alamos National Laboratory.

‡Computer Research and Applications Group (CIC-3), MS B256, Los Alamos National Laboratory, Los Alamos, New Mexico 87545 (cullum@lanl.gov). The work of the second author was supported by Department of Energy grant W-7405-ENG-36 and by Department of Energy Applied Mathematical Sciences Program grant KC-07-01-01.

§Institute of Computer Science, Academy of Sciences of the Czech Republic, Pod vodárenskou věží 2, 182 07 Prague 8, Czech Republic (tuma@cs.cas.cz). The work of the third author was supported by Grant Agency of the Czech Academy of Sciences grants 2030706 and 2030801.

suffer from various types of instability [17], [20]. Even in the SPD case, existence of the standard incomplete Cholesky (IC) factorization [34] is guaranteed only for special classes of matrices, such as H-matrices [33]. For general SPD matrices, breakdown in the IC process may occur due to exceedingly small or negative pivots. For this reason, variants of IC have been developed which are applicable to any SPD matrix without breakdowns; see [1], [25], [28], [37], [38]. However, some of these modifications are expensive, while others require diagonal perturbations which introduce additional parameters in the algorithm.

We are interested in parallel preconditioners that are widely applicable, reliable, and effective at reducing the number of iterations. The factorized sparse approximate inverse (FSAI) method developed by Kolotilina and Yeremin in [31] is one of the very few naturally parallel techniques that are also quite effective and highly stable for arbitrary SPD matrices. In this method, a lower triangular matrix G is computed as a sparse approximation to L^{-1}, where $A = LL^T$ is the Cholesky factorization of A. The algorithm does not require any information about L and works exclusively with A. Entries of G are computed as solutions of small "local" linear systems having principal submatrices of A as coefficient matrices. These are SPD and can be solved in a stable way by standard direct methods. Hence, no breakdowns are possible in computing G. Also, G is necessarily nonsingular, and the preconditioner $M := G^T G \approx A^{-1}$ is SPD and can be used with the conjugate gradient method. The standard form of this algorithm requires a prescribed sparsity pattern for G. Some ideas for determining sparsity patterns can be found in [16] and [26], but the effectiveness of these heuristics for factorized approximate inverses remains to be investigated.

A different factorized sparse approximate inverse preconditioner, based on incomplete conjugation (A-orthogonalization) of the unit basis vectors, is the AINV preconditioner [7], [8]. This method does not require the sparsity pattern of the approximate inverse factors to be specified in advance; rather, a good sparsity pattern is determined *dynamically* as the preconditioner is being computed. This is done by applying a drop tolerance to the computed entries of the inverse factors. As shown in [6], the calculation of the preconditioner can be parallelized using graph partitioning. However, the preconditioner may not be well defined for a general SPD matrix, due to breakdowns—see the next section for the definition of breakdown in the context of AINV. A sufficient condition for AINV to be breakdown-free is that A be an H-matrix [7]. We recall that M-matrices and diagonally dominant matrices are examples of H-matrices. If A is far from being an M-matrix or diagonally dominant, for example, if A has large positive off-diagonal entries, the preconditioner may not be defined or may fail to be positive definite. This is indeed the case for many problems arising from finite element modeling of structures and thin shells [5] and for certain diffusion problems involving highly distorted meshes [35].

In this paper we present a variant of the AINV preconditioner which is well defined (in exact arithmetic) for an arbitrary SPD matrix. We refer to the resulting preconditioner as the stabilized AINV, or SAINV, preconditioner. It is mathematically equivalent to the standard AINV algorithm when no dropping is applied.

The price to pay for the added robustness is the slightly higher cost of computing the preconditioner with respect to the standard AINV algorithm. However, for most problems the cost of forming the preconditioner is still reasonable. As we will see, matrix reorderings and scalings can be used to reduce costs.

We mention that breakdown-free variants of AINV have also been developed, independently, by Bridson [14] and by Kharchenko et al. [29]. Bridson's scheme is

different from ours. The variant of Kharchenko et al., on the other hand, is identical to ours. Their paper offers independent further evidence of the reliability of this approach.

Together with SAINV, we explore another approach to preventing breakdowns in the AINV process, based on the notion of *diagonally compensated reduction of positive off-diagonal entries*. This technique, introduced and analyzed by Axelsson and Kolotilina in [3], associates with any SPD matrix a Stieltjes matrix (that is, a symmetric and necessarily positive definite M-matrix) in a natural way. The standard AINV process can be applied to this new matrix, without breakdowns, resulting in an approximate inverse that can be used as a preconditioner for the original system.

The remainder of the paper is organized as follows. In section 2 we recall the standard AINV algorithm and review the techniques that have been used so far to handle breakdowns. In section 3 we introduce the stabilized AINV algorithm for SPD matrices, and in section 4 we describe the alternative approach based on diagonally compensated reduction. Section 5 is devoted to numerical experiments using challenging matrices from finite element modeling, including some experiments with other methods. Finally, in section 6 we make some concluding remarks and suggestions for further work.

2. The AINV algorithm. From now on, $A \in \mathbb{R}^{n\times n}$ is assumed to be SPD. The AINV algorithm [7] builds a factorized sparse approximate inverse of the form

$$M = ZD^{-1}Z^T \approx A^{-1}, \tag{2.1}$$

where Z is a unit upper triangular matrix and D is diagonal. The approximate inverse factor Z is a sparse approximation of the inverse of the L^T factor in the LDL^T decomposition of A. When the inverse factorization is performed exactly, the diagonal matrix D is the same in the two decompositions and contains the *pivots* down the main diagonal.

The AINV algorithm computes Z and D directly from A by means of an incomplete A-orthogonalization process applied to the unit basis vectors. In this process, small elements are dropped to preserve sparsity. The underlying assumption is that most entries in L^{-1} are small in magnitude. This is true for many problems of practical interest, particularly for discretizations of partial differential equations of elliptic type. Sparsity can also be achieved by combining the dropping of small entries with suitable sparse matrix orderings, such as nested dissection and minimum degree. These orderings are beneficial in that they result in smaller time and space requirements for forming and storing the preconditioner, while at the same time improving the quality of the preconditioner in a significant number of cases; see [11], [15], [22].

In order to describe the procedure, let a_i^T denote the ith row of A. Also, let e_i denote the ith unit basis vector. The basic A-orthogonalization procedure can be written as follows.

ALGORITHM 2.1.

(1) *Let* $z_i^{(0)} = e_i \quad (1 \le i \le n)$
(2) *For* $i = 1, 2, \ldots, n$ *do*
(3) *For* $j = i, i+1, \ldots, n$ *do*
(4) $p_j^{(i-1)} := a_i^T z_j^{(i-1)}$
(5) *End do*
(6) *if* $i = n$ *go to* (11)
(7) *For* $j = i+1, \ldots, n$ *do*

(8) $z_j^{(i)} := z_j^{(i-1)} - \left(\frac{p_j^{(i-1)}}{p_i^{(i-1)}}\right) z_i^{(i-1)}$

(9) *End do*

(10) *End do*

(11) *Let* $z_i := z_i^{(i-1)}$ *and* $p_i := p_i^{(i-1)}$, *for* $1 \le i \le n$. *Return* $Z = [z_1, z_2, \ldots, z_n]$ *and* $D = \mathrm{diag}(p_1, p_2, \ldots, p_n)$.

Sparsity is preserved by dropping off-diagonal entries in the z-vectors after the updates at step (8). The resulting (incomplete) algorithm is sometimes referred to as the *right-looking* AINV process. A left-looking variant also exists [9], which is sometimes advantageous. These algorithms can be extended in a straightforward manner to the nonsymmetric case [8].

A *breakdown* is defined as a negative or zero value of a pivot p_i. When no dropping is applied, $p_i = z_i^T A z_i > 0$. The incomplete procedure is well defined, i.e., no breakdown can occur, if A is an H-matrix (in the absence of round-off). When no breakdown occurs, the resulting sparse approximate inverse preconditioner is usually quite effective; see [9], [10] for comparisons between AINV and other preconditioners. In the general case, breakdowns can occur. Breakdowns have a crippling effect on the quality of the preconditioner. A negative p_i would result in an approximate inverse which is not positive definite; a zero pivot would force termination of the procedure, since step (8) cannot be carried out. In practice, exactly zero pivots are very unlikely to occur, but exceedingly small pivots can happen, resulting in uncontrolled growth of the entries of Z and extremely high fill-in.

In [7], a dynamic strategy to shift negative or numerically small pivots away from zero was proposed. Unfortunately, in the vast majority of cases the resulting sparse approximate inverse is not a good preconditioner. A somewhat better strategy is to adapt to AINV the a priori diagonal shift technique introduced by Manteuffel [33] for the IC factorization. Whenever a negative or exceedingly small pivot is encountered, the AINV process is terminated and reattempted on a new matrix $A' := A + \alpha \mathrm{diag}(A)$. Here $\mathrm{diag}(A)$ denotes the main diagonal of A, and $\alpha > 0$ denotes a parameter such that A' has a stable approximate inverse factorization, to be determined by trial and error. That such an α must exist is clear, since the AINV process cannot break down on a diagonally dominant matrix, and α can be chosen so large as to make A' diagonally dominant. However, the *optimal* value of α is usually much smaller than the one that makes A' diagonally dominant. Although this shifting strategy has proved successful in handling some difficult problems, it is expensive and frequently produces preconditioners of poor quality.

In the next section we propose a reformulation of the AINV algorithm that is applicable, without breakdowns, to any SPD matrix.

3. Stabilized AINV. First we need to take a close look at the mechanism of breakdown. We begin by writing down the explicit formula for the p_j's:

$$p_j^{(i-1)} = a_i^T z_j^{(i-1)} = \sum_{l=1}^{i-1} a_{il} z_{lj}^{(i-1)} + a_{ij} \qquad (i \le j \le n). \tag{3.1}$$

Here a_{ij} is the (i,j) entry of A, and $z_{lj}^{(i-1)}$ denotes the lth entry of vector $z_j^{(i-1)}$. Suppose now that a dropping rule is applied in the calculation of the z-vectors. The modified z-vectors will be denoted by $\bar{z}_j^{(i-1)}$, and the corresponding pivots are given

by

$$\bar{p}_i^{(i-1)} = \sum_{l=1}^{i-1} a_{il}\bar{z}_{li}^{(i-1)} + a_{ii} \qquad (1 \le i \le n). \tag{3.2}$$

(We explicitly stipulate that no dropping is applied to the diagonal entries of Z or to the diagonal pivots.) When A is an M-matrix, it is easily seen by induction that all the $\bar{z}_{lj}^{(i-1)}$ are nonnegative [7]. Because the off-diagonal entries of A are nonpositive, the action of dropping one entry in the z-vector cannot cause the (inexact) pivot $\bar{p}_i^{(i-1)}$ to decrease; it either remains unchanged, or it increases, depending on whether the corresponding coefficient a_{il} is zero or not. Because the exact pivots are positive, the AINV process cannot break down. Indeed, dropping has the effect of stabilizing the AINV process even further. This is in perfect analogy with the IC factorization for M-matrices [34].

Dropping an entry at step $(i-1)$ of the AINV process amounts to setting $\bar{z}_{lj}^{(i-1)} = 0$ for some l. Therefore, if either

$$a_{il} > 0 \qquad \text{and} \qquad z_{li}^{(i-1)} > 0$$

or

$$a_{il} < 0 \qquad \text{and} \qquad z_{li}^{(i-1)} < 0,$$

then the inexact pivot will be smaller than the exact pivot. In particular, if either one of a_{il} or $z_{li}^{(i-1)}$ is positive and large and the other is positive and not small, the inexact pivot can be significantly smaller than the exact one and can even become negative. In reality, the dropping rule will cause many of the $\bar{z}_{lj}^{(i-1)}$ to be zero, and the net change in the ith pivot will be the result of cumulative effects, some of which tend to increase and others to decrease the value of $\bar{p}_i^{(i-1)}$. Such effects may cancel each other out. Nevertheless, we have observed in numerical experiments performed on matrices with relatively large positive off-diagonal entries that the inexact pivots can become negative and, in fact, rather large in absolute value. The matrices for which this behavior has been observed were not artificial examples but came from real applications in structural analysis. It is no surprise that just setting these negative pivots equal to some arbitrary positive quantity resulted in preconditioners of very poor quality.

The way to avoid nonpositive pivots is simply to recall that in the exact A-orthogonalization process, the p_i's are the diagonal entries of matrix D which satisfies the matrix equation

$$Z^T A Z = D;$$

hence for $1 \le i \le n$

$$p_i = z_i^T A z_i > 0$$

since A is SPD and $z_i \neq 0$. (Recall that the ith entry of z_i is equal to 1.) In the exact process, the following equality holds:

$$p_i = z_i^T A z_i = a_i^T z_i. \tag{3.3}$$

This identity follows immediately from the fact that Z is unit upper triangular and AZ is lower triangular. Clearly, it is more economical to compute the pivots using the expression on the right-hand side of (3.3) rather than that in the middle. The same goes for the p_j's:

$$p_j = z_i^T A z_j = a_i^T z_j. \tag{3.4}$$

(For brevity, we have omitted the $(i-1)$ superscripts in the above formulas.) However, because of dropping and the resulting loss of A-orthogonality in the $\bar{z}$-vectors, such identities no longer hold in the inexact process, and for some matrices one can have

$$a_i^T \bar{z}_i \ll \bar{z}_i^T A \bar{z}_i$$

with the concomitant possibility of breakdowns.

These simple observations point to several reformulations of the AINV algorithm that are breakdown-free in exact arithmetic. The simplest thing to do is to use the standard AINV process (Algorithm 2.1 with dropping) as far as possible, and to switch to formula $\bar{p}_i = \bar{z}_i^T A \bar{z}_i$ whenever a negative or exceedingly small pivot shows up. Unfortunately, this simple fix is not good enough. It typically results in slow convergence of the PCG iteration, and for really hard problems there may be no convergence within a reasonable number of steps. The reason is that by the time a negative pivot occurs, the loss of information about the true inverse due to dropping has already been so great that no "local" trick can succeed in recovering a good preconditioner. A more global strategy is needed.

We found that a robust algorithm requires that the $\bar{p}_i$'s be computed using the quadratic form $\bar{z}_i^T A \bar{z}_i$ throughout the entire AINV process, for $i = 1, \ldots, n$. This still leaves the question of how to compute the $\bar{p}_j$'s with $i+1 \le j \le n$. One could either use the inexpensive formula (3.1) or the more expensive bilinear form (3.4). In the latter case, we have

$$\bar{p}_j = \bar{v}_i^T \bar{z}_j, \qquad \text{where} \qquad \bar{v}_i^T := \bar{z}_i^T A.$$

Because the vector $\bar{v}_i$ has already been computed as part of the calculation of $\bar{p}_i$, this approach is only slightly more expensive than using formula (3.1). The difference depends on how much more dense $\bar{v}_i^T$ is compared to a_i^T. It turns out that using the bilinear form (3.4) results in a preconditioner of much higher quality for nearly the same cost, so it definitely pays off to use the more expensive approach.

Hence, we obtain a reliable version of the AINV algorithm based on the following reformulation of Algorithm 2.1.

ALGORITHM 3.1.

(1) *Let* $z_i^{(0)} = e_i \quad (1 \le i \le n)$
(2) *For* $i = 1, 2, \ldots, n$ *do*
(3) $\quad v_i := A z_i^{(i-1)}$
(4) $\quad$ *For* $j = i, i+1, \ldots, n$ *do*
(5) $\quad\quad p_j^{(i-1)} := v_i^T z_j^{(i-1)}$
(6) $\quad$ *End do*
(7) $\quad$ *if* $i = n$ *go to* (12)
(8) $\quad$ *For* $j = i+1, \ldots, n$ *do*
(9) $\quad\quad z_j^{(i)} := z_j^{(i-1)} - \left(\frac{p_j^{(i-1)}}{p_i^{(i-1)}} \right) z_i^{(i-1)}$

(10) *End do*
(11) *End do*
(12) *Let* $z_i := z_i^{(i-1)}$ *and* $p_i := p_i^{(i-1)}$, *for* $1 \le i \le n$. *Return* $Z = [z_1, z_2, \ldots, z_n]$ *and* $D = \mathrm{diag}(p_1, p_2, \ldots, p_n)$.

Obviously, Algorithms 2.1 and 3.1 are mathematically equivalent. However, the incomplete process obtained by dropping in the z-vectors after step (9) of Algorithm 3.1 leads to a reliable approximate inverse procedure. This algorithm, in exact arithmetic, is applicable to any SPD matrix without breakdowns. Of course, instabilities due to positive but extremely small pivots may occur in finite precision, and a thresholding technique may still be necessary to guard against such possibility. However, we have not met this situation in our tests, and the resulting preconditioner appears to be reliable in practice. This new preconditioner will be hereafter referred to as the SAINV (for stabilized AINV) preconditioner.

What about the cost of SAINV? At first sight it might look very expensive to compute the preconditioner on the basis of Algorithm 3.1, which requires the computation of n matrix-vector products $v_i = Az_i$. To see that this is not nearly as expensive as it looks, it should be noticed that in the incomplete process the $\bar{z}_i$ vectors are kept sparse through the use of dropping (and possibly through reorderings). Hence, the n matrix-vector products can be performed in *sparse-sparse mode*, to borrow the term used in [18]. Hence, computing $\bar{v}_i$ amounts to forming a linear combination of a few columns of A, namely, those that correspond to nonzero entries in $\bar{z}_i$. Also, because $\bar{z}_i$ is the ith column of an upper triangular matrix, it is clear that at step i only the first i columns of A enter the matrix-vector product. Assuming that the drop tolerance is chosen so that the final Z contains $\mathcal{O}(n)$ nonzeros, and assuming an even distribution of nonzeros across the columns of Z, the cost of computing all the $\bar{p}_j$'s in the form of sparse bilinear expressions involving A is linear in the dimension n of the problem. As we shall see in the section on numerical experiments, the run time for computing the SAINV preconditioner is only slightly higher than that for the standard AINV algorithm when the latter does not break down, and the run time tends to be dominated by the time required to perform the iteration phase.

In the next section we briefly describe an alternative, inexpensive approach to systematically avoid breakdowns in the AINV process, based on the diagonally compensated reduction of positive off-diagonal matrix entries.

4. Diagonally compensated reduction approximate inverse. The method of diagonally compensated reduction of positive off-diagonal entries, due to Axelsson [2] and Axelsson and Kolotilina [3], associates with any SPD matrix A an SPD M-matrix $\hat{A}$. In the simplest variant of this technique, $\hat{A}$ is obtained by setting to zero the positive off-diagonal entries a_{ij} of A (reduction), which are subsequently added to the corresponding diagonal entry a_{ii} (diagonal compensation). Formally, the idea is to split A as

$$A = B + R,$$

where R contains the off-diagonal positive entries of A, and to let

$$\hat{A} = B + \Delta,$$

where Δ is the diagonal matrix satisfying

$$\Delta e = Re,$$

where e denotes the vector of all ones. Thus,

$$\hat{A} = A + (\Delta - R).$$

Notice that the symmetric matrix $\Delta - R$ is a singular M-matrix, since it has nonpositive off-diagonal entries and zero row sums; see [13], page 147. In particular, $\Delta - R$ is symmetric positive semidefinite. In turn, this shows that $\hat{A}$ is SPD, because it is the sum of an SPD matrix and a positive semidefinite one. Because $\hat{A}$ has nonpositive off-diagonal entries, it must be an M-matrix [13].

The standard AINV process can be applied to $\hat{A}$ without breakdowns, and the corresponding approximate inverse $M \approx \hat{A}^{-1}$ can be used as a preconditioner for the original linear system $Ax = b$. It is hoped that M will be a good preconditioner for A, provided that A is not too far from being an M-matrix. A rough way to estimate how much a given A deviates from being an M-matrix is to compute the Frobenius norm $\|R\|_F$ of the matrix R which contains the positive off-diagonal entries of A. To make this quantity invariant under global scalings, we divide it by $\|A\|_F$. That is, we let

$$\eta = \frac{\|R\|_F}{\|A\|_F}.$$

Notice that $0 \leq \eta < 1$, with $\eta = 0$ if and only if A is an M-matrix. In the next section we will try to ascertain whether there is a correlation between the size of η and the quality of the preconditioner M obtained by applying the standard AINV approximate inverse algorithm to $\hat{A}$. See [25] for results on the use of diagonally compensated reduction in the context of IC factorizations.

5. Numerical experiments. The standard AINV preconditioner is often applicable, with good results, to matrices that do not satisfy the H-matrix condition [7]. There are, however, applications that lead to matrices for which the standard AINV approach is unstable and produces unreliable preconditioners. This is typically the case for matrices from structural engineering, including finite element modeling of elasticity and thin shell problems [5].

Another class of problems for which the standard AINV method fails due to breakdowns consists of diffusion equations discretized on highly distorted finite element meshes (e.g., Kershaw meshes). Such meshes arise frequently in codes developed at Los Alamos and elsewhere for the modeling of phenomena with complex physics, e.g., radiation diffusion [35].

In this section we present the results of a number of numerical tests performed on a selection of 16 matrices from the 2 areas mentioned above. Of these, 14 are from structural analysis and the remaining 2 are diffusion problems. Most of the matrices from structural analysis can be downloaded from the Matrix Market website [36]. The exceptions are the NASA examples, extracted from the University of Florida Sparse Matrix Collection [19], and the SMT matrix, which was provided by R. Kouhia of the Helsinki University of Technology. The two diffusion problems were extracted from a Los Alamos diffusion package, AUGUSTUS, developed by Mike Hall. These are three-dimensional, steady-state problems defined on the unit cube discretized on a highly skewed Kershaw mesh using the scheme described in [35].

Some basic information about the tests problems is provided in Table 1. For each matrix we provide the problem size n, the number of nonzeros in the lower triangular part nnz, the value of η as defined in the previous section, and the application area. In the last column we give the number of iterations and time (in seconds) required to solve

TABLE 1
Test problem information.

Matrix	n	nnz	η	Application	JCG (Its/Time)
BCSSTK13	2003	42943	0.474	Fluid flow	1406/20.4
BCSSTK14	1806	32630	0.325	Roof of Coliseum	409/4.79
BCSSTK15	3948	60882	0.096	Offshore platform	518/16.4
BCSSTK16	4884	147631	0.221	Dam	191/11.4
BCSSTK17	10974	219812	0.472	Pressure vessel	2522/287.
BCSSTK18	11948	80519	0.202	Power plant	1120/47.0
BCSSTK21	3600	15100	0.248	Clamped plate	559/4.81
BCSSTK25	15439	133840	0.003	Skyscraper	>10000/–
S1RMQ4M1	5489	143300	0.203	Cylindrical shell	692/37.1
S2RMQ4M1	5489	143300	0.293	Cylindrical shell	1529/80.0
S3RMQ4M1	5489	143300	0.302	Cylindrical shell	6884/359.
NASA2910	2910	88603	0.213	NASA structure	1350/51.1
NASA4704	4704	54730	0.248	NASA structure	4866/145.3
SMT	25710	1889447	0.423	Mounted transistor	1984/492.
AUGUSTUS5	134144	645028	0.235	Diffusion	842/136.
AUGUSTUS7	1060864	5187320	0.233	Diffusion	1540/2970.

the linear system using the conjugate gradient method with Jacobi preconditioning (JCG). The iteration was terminated when the 2-norm of the initial residual was reduced by at least 8 orders of magnitude, or when a maximum of 10,000 iterations was reached. The initial guess was the zero vector, and the right-hand side was constructed as $b = Ax$, where x is a vector with random entries, uniformly distributed in $(0, 1)$. Similar results were obtained for other choices of the right-hand side. All the runs were performed on a SUN Ultra 5 workstation, except for those with SMT and the AUGUSTUS diffusion problems, for which one processor of an SGI Origin 2000 was used. The codes were written in standard Fortran77 and compiled with the -O3 optimization option. As can be seen, several of these problems are rather difficult to solve and have substantial positive off-diagonal part R. The standard AINV algorithm is unstable on all these problems, except BCSSTK16, which is the easiest in our data set. The most challenging problems are BCSSTK25 and the thin shell S3RMQ4M1.

In the following tables we present a number of results obtained with various preconditioners. We provide the time for computing the preconditioner (P-time), the number of PCG iterations (Its), the time to perform the PCG iterations (It-time), the total time (Tot-time), and the density ρ of the preconditioner. This is defined as the ratio between the number of nonzeros in the approximate inverse factor and the number of nonzeros in the lower triangular part of A. The preconditioners considered are FSAI [31] and variants of the standard AINV algorithm and of SAINV. The variants correspond to various preliminary transformations operated on the coefficient matrix. These are symmetric diagonal scaling (J), reordering with the multiple minimum degree (MMD) algorithm [32], diagonally compensated reduction of positive off-diagonal entries (DCR), and combinations of these. Thus, for instance, J-DCR-MMD-AINV stands for the standard AINV algorithm applied to the matrix obtained from the diagonally compensated reduction of the original matrix after diagonal scaling and MMD reordering. For AINV and SAINV, the value $\tau = 10^{-1}$ of the drop tolerance is used throughout.

In Table 2 we present results obtained with the FSAI preconditioner [31]. This algorithm already incorporates a sophisticated scaling strategy; therefore, no diagonal scaling was applied. We found that MMD reordering had a beneficial effect on the

TABLE 2
Test results for MMD-FSAI preconditioner.

Matrix	P-time	Its	It-time	Tot-time
BCSSTK13	0.71	440	15.2	15.9
BCSSTK14	0.35	80	2.35	2.70
BCSSTK15	0.54	201	10.6	11.1
BCSSTK16	2.67	72	7.78	10.5
BCSSTK17	2.70	472	82.4	85.1
BCSSTK18	0.63	322	27.0	27.6
BCSSTK21	0.09	260	5.39	5.48
BCSSTK25	1.25	1136	145.	146.
S1RMQ4M1	1.99	233	24.1	26.1
S2RMQ4M1	2.01	293	30.5	32.5
S3RMQ4M1	2.03	446	48.0	50.0
NASA2910	2.51	230	15.6	18.1
NASA4704	0.42	1068	52.6	56.0
SMT	53.4	403	193.	246.
AUGUSTUS5	3.60	363	95.2	98.8
AUGUSTUS7	30.6	705	2042.	2073.

TABLE 3
Test results for BCSSTK16.

Preconditioner	P-time	Its	It-time	Tot-time	ρ
JCG	–	191	11.4	11.4	0.00
MMD-FSAI	2.67	72	7.78	10.5	1.00
AINV	0.79	105	6.33	7.12	0.12
MMD-AINV	1.01	99	5.93	6.94	0.13
J-AINV	0.89	101	6.09	6.98	0.12
J-MMD-AINV	0.99	102	6.02	7.01	0.12
DCR-AINV	0.49	150	8.76	9.25	0.05
DCR-MMD-AINV	0.54	147	8.39	8.93	0.06
J-DCR-AINV	0.51	149	8.63	9.14	0.05
J-DCR-MMD-AINV	0.54	149	8.46	9.00	0.05
SAINV	1.00	101	6.12	7.12	0.12
MMD-SAINV	1.36	96	5.68	7.04	0.13
J-SAINV	0.88	98	5.82	**6.70**	0.12
J-MMD-SAINV	1.05	98	5.72	6.77	0.12

quality of FSAI preconditioning for most problems, especially the larger ones, and this ordering was used in the experiments. The sparsity pattern for the approximate inverse factor G was the same as that of the lower triangular part of P^TAP, where P is the permutation matrix corresponding to the MMD ordering.

These results clearly demonstrate the robustness of FSAI. This algorithm is generally more reliable and efficient than JCG. Note, however, the relatively poor performance on BCSSTK21. We mention that the performance of FSAI can be significantly improved using a better choice for the sparsity pattern and by postprocessing the preconditioner; see, e.g., the suggestions in [30].

In Table 3 we present results for BCSSTK16 using a variety of preconditioners. We show these results because the standard AINV is stable for this matrix and we wish to compare it with the new variants. The fastest total timing is in boldface.

There are several observations worth making. We note that AINV and SAINV both produce very sparse and yet quite effective preconditioners. Because of such sparsity, the cost of SAINV is not much higher than that of AINV, particularly with

TABLE 4
Test results for BCSSTK18.

Preconditioner	P-time	Its	It-time	Tot-time	ρ
JCG	–	1120	47.0	47.0	0.00
MMD-FSAI	0.63	322	27.0	27.6	1.00
AINV	unst.	–	–	–	–
MMD-AINV	1.78	7068	501.	503.	1.15
J-AINV	0.47	743	46.6	47.1	0.73
J-MMD-AINV	0.96	1925	118.	119.	0.69
DCR-AINV	0.50	924	49.5	50.0	0.34
DCR-MMD-AINV	0.58	1006	53.3	53.9	0.34
J-DCR-AINV	0.49	663	36.3	36.8	0.35
J-DCR-MMD-AINV	0.56	613	32.2	32.8	0.34
SAINV	3.37	257	18.9	22.3	1.32
MMD-SAINV	2.83	354	24.6	27.4	0.99
J-SAINV	0.89	278	17.1	18.0	0.65
J-MMD-SAINV	1.05	261	15.5	**16.6**	0.58

symmetric scaling. On the other hand, diagonally compensated reduction (which is not needed here since AINV is stable), which also produces very sparse preconditioners, results in slower convergence and higher overall timings. These experiments suggest that SAINV can be superior to AINV even when AINV is stable, provided that the preconditioner is sufficiently sparse.

In Table 4 we show the results for a more difficult problem, BCSSTK18. These results are fairly typical, and this is why we discuss them in some detail. For this example, AINV is unstable. Applying symmetric scaling and reordering for sparsity improves the situation only slightly: convergence is extremely slow. Using diagonally compensated reduction with AINV is somewhat more effective, but worse than MMD-FSAI. We performed some experiments with a smaller value of the drop tolerance to see if a denser preconditioner would help, but this was not the case. The number of iterations was somewhat reduced but not the timings. Better results are obtained with SAINV, particularly in combination with scaling and MMD reordering. Notice that SAINV results in preconditioners which are fairly sparse, although not as much as for the previous example. The time for computing the approximate inverse with SAINV and its variants is still quite small.

In Table 5 we show the results of using AINV with diagonally compensated reduction. We see that this method is reliable but generally not very effective. On difficult matrices, such as BCSSTK18 and S3RMQ4M1, the performance is poor. In some cases, improved results can be obtained in combination with symmetric scalings and MMD reordering, but no clear trend emerges and it is difficult to make specific recommendations.

Table 6 contains results for the SAINV preconditioner. For most problems, the performance is better than that of DCR-AINV and comparable with that of MMD-FSAI. Note that MMD-FSAI performs better on the shell problems.

Finally, in Table 7 we show the results for SAINV used in conjunction with symmetric scaling and MMD reordering. While this combination is not always optimal, it was the best or nearly so in a majority of cases. Therefore we feel comfortable recommending to use this combination in practice. Note in particular that J-MMD-SAINV is on average a factor of three faster than Jacobi preconditioning. Also notice that this approach is better than using AINV with diagonally compensated reduc-

TABLE 5
Test results for DCR-AINV preconditioner.

Matrix	P-time	Its	It-time	Tot-time	ρ
BCSSTK13	0.11	1185	19.8	19.9	0.13
BCSSTK14	0.06	353	4.69	4.75	0.12
BCSSTK15	0.11	676	18.6	18.7	0.15
BCSSTK16	0.49	150	8.76	9.25	0.05
BCSSTK17	0.41	1531	150.	150.	0.12
BCSSTK18	0.50	924	49.5	50.0	0.34
BCSSTK21	0.03	285	3.39	3.42	0.66
BCSSTK25	33.8	1853	256.	290.	2.10
S1RMQ4M1	0.30	539	30.5	30.8	0.10
S2RMQ4M1	0.30	1619	92.4	92.7	0.10
S3RMQ4M1	0.30	7675	440.	440.	0.10
NASA2910	0.18	689	22.6	22.8	0.08
NASA4704	0.11	2935	79.4	79.5	0.26
SMT	5.55	1380	353.	358.	0.02
AUGUSTUS5	3.11	331	82.7	85.8	0.82
AUGUSTUS7	35.2	605	2733.	2761.	0.81

TABLE 6
Test results for SAINV preconditioner.

Matrix	P-time	Its	It-time	Tot-time	ρ
BCSSTK13	5.27	368	11.2	16.5	1.41
BCSSTK14	1.05	78	1.41	2.46	0.73
BCSSTK15	1.66	219	7.79	9.45	0.61
BCSSTK16	1.00	101	6.12	7.12	0.12
BCSSTK17	4.87	919	112.	117.	0.54
BCSSTK18	3.37	257	18.9	22.3	1.32
BCSSTK21	0.24	169	2.70	2.94	1.83
BCSSTK25	15.4	1848	286.	302.	2.20
S1RMQ4M1	8.04	267	23.8	31.8	1.06
S2RMQ4M1	13.8	521	60.4	74.2	1.91
S3RMQ4M1	36.4	4239	679.	715.	3.24
NASA2910	1.80	372	15.4	17.4	0.49
NASA4704	1.98	831	30.0	32.0	0.91
SMT	25.2	597	162.	188.	0.12
AUGUSTUS5	19.0	273	77.8	96.8	1.28
AUGUSTUS7	222.	515	1706.	1928.	1.28

tion in all cases, except AUGUSTUS5, where the results are comparable. We should

TABLE 7
Test results for J-MMD-SAINV preconditioner.

Matrix	P-time	Its	It-time	Tot-time	ρ
BCSSTK13	0.82	349	6.53	7.35	0.39
BCSSTK14	0.46	73	1.07	1.53	0.27
BCSSTK15	0.81	167	5.05	5.86	0.33
BCSSTK16	1.05	98	5.72	6.77	0.12
BCSSTK17	3.13	711	79.8	82.9	0.40
BCSSTK18	1.05	261	15.5	16.5	0.58
BCSSTK21	0.39	191	2.88	3.27	1.51
BCSSTK25	1.99	1512	151.	153.	0.57
S1RMQ4M1	1.26	248	15.0	16.3	0.20
S2RMQ4M1	1.42	528	32.7	34.1	0.25
S3RMQ4M1	1.43	1140	70.3	71.7	0.24
NASA2910	0.95	341	12.7	13.6	0.28
NASA4704	0.91	1176	38.3	39.2	0.62
SMT	10.3	546	148.	159.	0.11
AUGUSTUS5	12.0	281	75.6	87.6	1.10
AUGUSTUS7	220.	516	1551.	1771.	1.06

mention that for the matrices from structural analysis, which are fairly dense, the right-looking version of the SAINV code was found to be faster and it was the one used in the experiments. For the diffusion problems, which are comparatively sparse, the left-looking version is faster and was used for the runs presented here.

The experiments in this section suggest that SAINV is a robust and effective general purpose preconditioner for the conjugate gradient method, especially when used in combination with symmetric scalings and MMD ordering. The standard AINV preconditioner can be used in connection with diagonally compensated reduction, resulting in a preconditioner which is also reliable but not nearly as effective, with the possible exception of diffusion problems. Concerning this approach, we note that it is difficult to predict the performance on the basis of η. Some correlation is present, e.g., in the three shell problems, for which η increases with the difficulty of the problem. However, η is too coarse an indicator of performance to be useful in deciding when to use diagonally compensated reduction.

The SAINV preconditioner is also easy to use. As already mentioned, we have used the standard value $\tau = 10^{-1}$ for the drop tolerance in all the tests with AINV and SAINV. Much better results can be obtained in some cases with a different value of τ, but we deliberately avoided fine-tuning because we wanted to show that this is a good choice in general. Also, the performance of the algorithm is only moderately affected by the choice of τ, provided that it is not chosen too small or too large; see [12].

6. Conclusions. We have developed SAINV, a reliable version of the AINV factorized approximate inverse preconditioner for the conjugate gradient method. The algorithm is applicable to general SPD matrices, is easily parallelized, and performs well on challenging linear systems arising in finite element modeling of elasticity and diffusion problems. The new preconditioner is easy to use, as it does not require the user to specify a sparsity pattern for the approximate inverse but only the value of the drop tolerance τ. Typically, setting $\tau = 10^{-1}$ gives good results. While the setup phase is slightly more expensive than for the standard AINV preconditioner, the cost of computing the preconditioner is still reasonable and the overall algorithm is cost effective.

Our experiments indicate that the performance of SAINV can be significantly improved by applying some preliminary transformations to the matrix A, namely, symmetric diagonal scaling and MMD reordering. With these transformations the storage required for the SAINV preconditioner is often significantly less than that for the coefficient matrix A itself. We stress that the cost of these preprocessings is negligible compared to that of solving the linear system, and we recommend using them as the default options in practice.

We have also investigated the use of diagonally compensated reduction of positive off-diagonal matrix entries as an alternative, inexpensive means to stabilize the AINV preconditioner. While this approach gave good results on some of our test problems, it proved to be generally less effective than SAINV applied to the original matrix.

Acknowledgments. The first author would like to thank Sofia Benzi for postponing her birth until after the bulk of this work was completed. Parts of this paper were written while the third author was a visitor at Los Alamos National Laboratory; the hospitality and support of LANL are greatly appreciated. Thanks also to four anonymous referees for helpful comments.

REFERENCES

[1] M. A. Ajiz and A. Jennings, *A robust incomplete Choleski-conjugate gradient algorithm*, Internat. J. Numer. Methods Engrg., 20 (1984), pp. 949–966.
[2] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, Cambridge, 1994.
[3] O. Axelsson and L. Yu. Kolotilina, *Diagonally compensated reduction and related preconditioning methods*, Numer. Linear Algebra Appl., 1 (1994), pp. 155–177.
[4] S. T. Barnard, L. M. Bernardo, and H. D. Simon, *An MPI implementation of the SPAI preconditioner on the T3E*, Internat. J. High Perf. Comput. Applic., 13 (1999), pp. 107–123.
[5] M. Benzi, R. Kouhia, and M. Tůma, *An assessment of some preconditioning techniques in shell problems*, Comm. Numer. Methods Engrg., 14 (1998), pp. 897–906.
[6] M. Benzi, J. Marín, and M. Tůma, *A two-level parallel preconditioner based on sparse approximate inverses*, in Iterative Methods in Scientific Computation IV, D. R. Kincaid and A. C. Elster, eds., IMACS Series in Computational and Applied Mathematics, 5, IMACS, New Brunswick, NJ, 1999, pp. 167–178.
[7] M. Benzi, C. D. Meyer, and M. Tůma, *A sparse approximate inverse preconditioner for the conjugate gradient method*, SIAM J. Sci. Comput., 17 (1996), pp. 1135–1149.
[8] M. Benzi and M. Tůma, *A sparse approximate inverse preconditioner for nonsymmetric linear systems,* SIAM J. Sci. Comput., 19 (1998), pp. 968–994.
[9] M. Benzi and M. Tůma, *Numerical experiments with two approximate inverse preconditioners,* BIT, 38 (1998), pp. 234–241.
[10] M. Benzi and M. Tůma, *A comparative study of sparse approximate inverse preconditioners*, Appl. Numer. Math., 30 (1999), pp. 305–340.
[11] M. Benzi and M. Tůma, *Orderings for factorized sparse approximate inverse preconditioners*, SIAM J. Sci. Comput., 21 (2000), pp. 1851–1868.
[12] L. Bergamaschi, G. Pini, and F. Sartoretto, *Approximate inverse preconditioning in the parallel solution of sparse eigenproblems*, Numer. Linear Algebra Appl., 7 (2000), pp. 99–116.
[13] A. Berman and R. J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, Academic Press, New York, 1979.
[14] R. Bridson, *Multi-Resolution Approximate Inverses*, M.Sc. thesis, Computer Science Department, Waterloo University, Waterloo, Ontario, Canada, 1999.
[15] R. Bridson and W.-P. Tang, *Ordering, anisotropy, and factored sparse approximate inverses*, SIAM J. Sci. Comput., 21 (1999), pp. 867–882.
[16] E. Chow, *A priori sparsity patterns for parallel sparse approximate inverse preconditioners*, SIAM J. Sci. Comput., 21 (2000), pp. 1804–1822.
[17] E. Chow and Y. Saad, *Experimental study of ILU preconditioners for indefinite matrices*, J. Comput. Appl. Math., 86 (1997), pp. 387–414.
[18] E. Chow and Y. Saad, *Approximate inverse preconditioners via sparse-sparse iterations*, SIAM J. Sci. Comput., 19 (1998), pp. 995–1023.

[19] T. Davis, *University of Florida Sparse Matrix Collection*, http://www.cise.ufl.edu/∼davis/sparse/ (1999).
[20] H. C. Elman, *A stability analysis of incomplete LU factorizations*, Math. Comp., 47 (1986), pp. 191–217.
[21] M. R. Field, *An Efficient Parallel Preconditioner for the Conjugate Gradient Algorithm*, Hitachi Dublin Laboratory Technical Report HDL-TR-97-175, Dublin, Ireland, 1997.
[22] M. R. Field, *Improving the Performance of Factorised Sparse Approximate Inverse Preconditioner*, Hitachi Dublin Laboratory Technical Report HDL-TR-98-199, Dublin, Ireland, 1998.
[23] M. J. Grote and T. Huckle, *Parallel preconditioning with sparse approximate inverses*, SIAM J. Sci. Comput., 18 (1997), pp. 838–853.
[24] N. I. M. Gould and J. A. Scott, *Sparse approximate-inverse preconditioners using norm-minimization techniques*, SIAM J. Sci. Comput., 19 (1998), pp. 605–625.
[25] I. Hladík, M. B. Reed, and G. Swoboda, *Robust preconditioners for linear elasticity FEM analyses*, Internat. J. Numer. Methods Engrg., 40 (1997), pp. 2109–2127.
[26] T. Huckle, *Approximate sparsity patterns for the inverse of a matrix and preconditioning*, Appl. Numer. Math., 30 (1999), pp. 291–303.
[27] I. E. Kaporin, *New convergence results and preconditioning strategies for the conjugate gradient method*, Numer. Linear Algebra Appl., 1 (1994), pp. 179–210.
[28] I. E. Kaporin, *High quality preconditioning of a general symmetric positive definite matrix based on its* $U^TU + U^TR + R^TU$ *decomposition*, Numer. Linear Algebra Appl., 5 (1998), pp. 483–509.
[29] S. A. Kharchenko, L. Yu. Kolotilina, A. A. Nikishin, and A. Yu. Yeremin, *A Reliable AINV-type Preconditioning Method for Constructing Sparse Approximate Inverse Preconditioners in Factored Form*, Moscow University, Moscow, Russia, 1999, preprint.
[30] L. Yu. Kolotilina, A. A. Nikishin, and A. Yu. Yeremin, *Factorized sparse approximate inverse preconditionings.* IV*: Simple approaches to rising efficiency*, Numer. Linear Algebra Appl., 6 (1999), pp. 515–531.
[31] L. Yu. Kolotilina and A. Yu. Yeremin, *Factorized sparse approximate inverse preconditioning.* I. *Theory*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 45–58.
[32] J. W. H. Liu, *Modification of the minimum degree algorithm by multiple elimination*, ACM Trans. Math. Software, 11 (1985), pp. 141–153.
[33] T. A. Manteuffel, *An incomplete factorization technique for positive definite linear systems*, Math. Comp., 34 (1980), pp. 473–497.
[34] J. A. Meijerink and H. A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric* M*-matrix*, Math. Comp., 31 (1977), pp. 148–162.
[35] J. E. Morel, M. L. Hall, and M. J. Shashkov, *A Local Support-Operators Diffusion Discretization Scheme for Hexahedral Meshes*, Report LA-UR-99-4358, Los Alamos National Laboratory, Los Alamos, NM, 1999.
[36] National Institute of Standards, *Matrix Market*, http://math.nist.gov/MatrixMarket (1999).
[37] M. Suariana and K. H. Law, *A robust incomplete factorization based on value and space constraints*, Internat. J. Numer. Methods Engrg., 38 (1995), pp. 1703–1719.
[38] M. Tismenetsky, *A new preconditioning technique for solving large sparse linear systems*, Linear Algebra Appl., 154–156 (1991), pp. 331–353.