A BLOCK APPROXIMATE INVERSE PRECONDITIONER FOR SPARSE NONSYMMETRIC LINEAR SYSTEMS *

J. CERDÁN†, T. FARAJ‡, J. MARíN†, AND J. MAS†

Abstract. In this paper a block approximate inverse preconditioner to solve sparse nonsymmetric linear systems with iterative Krylov subspace methods is introduced. The computation of the preconditioner involves the Sherman-Morrison-Woodbury formula which shows how to update the inverse of a nonsingular $n \times n$ matrix A subject to a modification of the type $A + XY^T$, where $X, Y \in \mathbb{R}^{n\times m}$. Therefore, it is a generalization of the preconditioner presented in [12]. The stability of the preconditioner is studied and it is shown that the computation of the preconditioner is breakdown-free for M-matrices and H-matrices. To test its performance numerical experiments obtained for a representative set of matrices are presented. Finally, the effect of symmetric reorderings on the quality of the preconditioner is evaluated.

Key words. Approximate inverse preconditioners, block algorithms, Krylov iterative methods, Sherman-Morrison-Woodbury formula

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

1. Introduction.

In this paper we consider the solution of linear systems

$$Ax = b\,, \tag{1.1}$$

by preconditioned iterations. We assume the matrix $A \in \mathbb{R}^{n\times n}$ to be sparse and nonsymmetric. For large values of n an approximate solution for (1.1) is frequently obtained by means of iterative Krylov subspace methods. In practice, to accelerate the convergence of these methods, the (left) preconditioned linear system is solved instead

$$MAx = Mb,$$

where the matrix M is the preconditioner [28].

Usually the matrix M is chosen in such a way that the preconditioned matrix MA is close to the identity I_n in some sense. For instance, the condition number is small and/or the eigenvalues are clustered away from the origin. In general, the more clustered the eigenvalues are, the faster the convergence rate is. Another desired situation is that the preconditioner should be easy to compute and the cost of the preconditioning step should be of the same order of a matrix-vector product with the coefficient matrix A.

In the last years several preconditioning techniques have been proposed. Roughly speaking they can be grouped in two classes: *implicit preconditioners* and *explicit preconditioners*. Preconditioners of the first class typically compute incomplete factorizations of A, as incomplete LU, and therefore the preconditioning step is done by solving two triangular linear systems (see for example [24, 25, 27, 28]). By contrast the second class of preconditioners compute and store a sparse approximation of the inverse of A and the preconditioning step is done by a matrix-vector product (see [8, 14, 22, 23]). Since this operation is easy to implement in parallel and vector

*Supported by Spanish Grants MTM 2004 02998 and Generalitat Valenciana Grupos 03/062.

†Institut de Matemàtica Aplicada, Universitat Politècnica de València, 46022 València, Spain (`jcerdan,jmarinma,jmasm@mat.upv.es`).

‡I.E.S Salvador Gadea, Valencia.

computers, approximate inverse preconditioners are attractive for parallel computations. In addition, some authors argue that approximate inverse preconditioners are more robust than implicit ones [3]. For a comparative study of some of these techniques we refer to [9]. We focus in this paper on sparse approximate inverse preconditioners.

A new framework for computing factorized sparse approximate inverse preconditioners has recently been presented in [12]. Given two sets of vectors $\{x_k\}_{k=1}^n$ and $\{y_k\}_{k=1}^n$ in $\mathbb{R}^n$, and a nonsingular matrix A_0 such that $A = A_0 + \sum_{k=1}^n x_k y_k^T$, the AISM algorithm computes an approximate factorization of the matrix $A_0^{-1} - A^{-1}$ of the form $U\Omega V^T$ using the Sherman-Morrison formula. The particular case $A_0 = sI_n$, where I_n is the identity matrix and s is a positive scalar factor was studied and it was shown that the computation of the preconditioner is breakdown-free when A is an M-matrix. We will show that this is also true for the wider class of H-matrices. Compared to AINV [8] the authors observed that AISM performed similarly for a representative set of matrices.

On the other hand, exploiting faster speeds of level 3 BLAS block algorithms is becoming increasingly popular in matrix computations. Since they operate on blocks or submatrices of the original matrix, they are well suited for modern high performance computers. Furthermore, certain problems have a natural block structure that should be exploited to gain robustness. There are a number of papers on block preconditioners, see for instance [7, 4, 11, 13] and the references therein. In all cases an improved efficiency with respect to non-blocking algorithms was observed.

These considerations motivate the present study. We present a block preconditioner which generalizes AISM. The new preconditioner is based on the Sherman-Morrison-Woodbury formula [32] which states that the inverse of the matrix $A+XY^T$ is given by

$$A^{-1} - A^{-1}X(I + Y^T A^{-1}X)^{-1}Y^T A^{-1}. \tag{1.2}$$

provided that the matrices $A \in \mathbb{R}^{n\times n}$ and $I + Y^T A^{-1}X$ are nonsingular and $X, Y \in \mathbb{R}^{n\times m}$.

This paper is organized as follows. In section 2 an expression for the inverse of a matrix A using the Sherman-Morrison-Woodbury formula (1.2) is obtained. Then, in section 3 this expression is used to obtain a block approximate inverse preconditioner. Our findings indicate that this preconditioner can be computed without breakdowns for M-matrices and H-matrices. Therefore, since the preconditioner proposed in [12] is a particular case of the preconditioner proposed here, we also prove that the computation of point AISM is stable for H-matrices. In order to evaluate the performance of the preconditioner, the results of the numerical experiments for a representative set of matrices are presented in section 4. In addition, the effect of symmetric reorderings on the quality of the preconditioner is specified. Finally, in section 5 the main conclusions are drawn.

Throughout the paper the following notation will be used. Given two matrices $A = [a_{ij}]$ and $B = [b_{ij}]$ we denote $A \geq B$ when $a_{ij} \geq b_{ij}$, similar convention follows for $\leq$. Likewise $|A| = [|a_{ij}|]$. A matrix A is a nonsingular M-matrix if $a_{ij} \leq 0$ for all $i \neq j$ and it is monotone, i.e., $A^{-1} \geq O$. For a given matrix A one can associate its comparison matrix $\mathcal{M}(A) = [\alpha_{ij}]$, where

$$\alpha_{ii} = |a_{ii}|, \text{ and } \alpha_{ij} = -|a_{ij}| \text{ for } i \neq j.$$

The matrix A is an H-matrix if its comparison matrix $\mathcal{M}(A)$ is an M-matrix. An H-matrix is always nonsingular.

We conclude this section with some well-known properties of M- and H-matrices. If $B \geq A$ with $b_{ij} \leq 0$ for $i \neq j$ and A is an M-matrix, then B is also an M-matrix. Moreover, $A^{-1} \geq B^{-1}$ [31]. If A is an H-matrix, then $|A^{-1}| \leq \mathcal{M}(A)^{-1}$ [26].

2. An expression for the inverse of A. Let $A_0 \in \mathbb{R}^{n\times n}$ be a nonsingular matrix, and let $X_k, Y_k \in \mathbb{R}^{n\times m_k}$, $k = 1,\dots,p$ be two sets of rectangular matrices such that,

$$A = A_0 + \sum_{k=1}^{p} X_k Y_k^T. \tag{2.1}$$

Assume that the matrices $T_k = I_{m_k} + Y_k^T A_{k-1}^{-1} X_k$, $k = 1,\dots,p$ are nonsingular, where $A_{k-1} = A_0 + \sum_{i=1}^{k-1} X_i Y_i^T$, and I_{m_k} denotes the identity matrix of size $m_k \times m_k$. From (1.2) the inverse of A_k is given by

$$A_k^{-1} = A_{k-1}^{-1} - A_{k-1}^{-1} X_k T_k^{-1} Y_k^T A_{k-1}^{-1}, \quad k = 1,\dots,p. \tag{2.2}$$

Since $A_p^{-1} = A^{-1}$, applying (2.2) recursively one has

$$A^{-1} = A_0^{-1} - \sum_{k=1}^{p} A_{k-1}^{-1} X_k T_k^{-1} Y_k^T A_{k-1}^{-1}, \tag{2.3}$$

which can be written in matrix notation as

$$A^{-1} = A_0^{-1} - \Phi T^{-1} \Psi^T, \tag{2.4}$$

where

$$\Phi = \begin{bmatrix} A_0^{-1}X_1 & A_1^{-1}X_2 & \cdots & A_{p-1}^{-1}X_p \end{bmatrix},$$

$$T^{-1} = \begin{bmatrix} T_1^{-1} & & & \\ & T_2^{-1} & & \\ & & \ddots & \\ & & & T_p^{-1} \end{bmatrix} \quad \text{and} \quad \Psi^T = \begin{bmatrix} Y_1^T A_0^{-1} \\ Y_2^T A_1^{-1} \\ \vdots \\ Y_p^T A_{p-1}^{-1} \end{bmatrix}.$$

To avoid having to compute the matrices A_k, we can define from X_k and Y_k, for $k = 1,\dots,p$, two new sets of matrices U_k and V_k as in (2.5) and (2.6). The following result is a generalization of Theorem 2.1 in [12].

THEOREM 2.1. *Let A and A_0 be two nonsingular matrices, and let $\{X_k\}_{k=1}^p$ and $\{Y_k\}_{k=1}^p$ be two sets of matrices such that condition (2.1) is satisfied. In addition suppose that the matrices $T_k = I_{m_k} + Y_k^T A_{k-1}^{-1} X_k$ for $k = 1,\dots,p$ are nonsingular. Then*

$$U_k = X_k - \sum_{i=1}^{k-1} U_i T_i^{-1} V_i^T A_0^{-1} X_k, \tag{2.5}$$

$$V_k = Y_k - \sum_{i=1}^{k-1} V_i T_i^{-T} U_i^T A_0^{-T} Y_k \tag{2.6}$$

are well defined for $k = 1, \ldots, p$. *Moreover,*

$$\begin{aligned} A_{k-1}^{-1} X_k &= A_0^{-1} U_k \\ Y_k^T A_{k-1}^{-1} &= V_k^T A_0^{-1} \end{aligned} \tag{2.7}$$

and

$$T_k = I_{m_k} + Y_k^T A_0^{-1} U_k = I_{m_k} + V_k^T A_0^{-1} X_k. \tag{2.8}$$

Proof. Similar to the proof of Theorem 2.1 in [12]. □

Denoting by $U = [U_1\ U_2\ \cdots\ U_p]$ and $V = [V_1\ V_2\ \cdots\ V_p]$ the matrices whose block columns are the matrices U_k and V_k, respectively, equation (2.4) can be rewritten as

$$A^{-1} = A_0^{-1} - A_0^{-1} U T^{-1} V^T A_0^{-1}, \tag{2.9}$$

and hence the computation of the matrices A_k may be avoided.

3. A block approximate inverse preconditioner. Different choices of the matrices X_k, Y_k and A_0 allow different ways of computing the inverse of the matrix A. Nevertheless, it is convenient that A_0 be a matrix whose inverse is either known or easy to obtain. As in [12], we take $A_0 = sI_n$ where s is a positive scalar and I_n is the identity matrix of size n. Consider the matrix A partitioned in block form:

$$A = \begin{bmatrix} A_{11} & A_{12} & \ldots & A_{1p} \\ A_{21} & A_{22} & \ldots & A_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ A_{p1} & A_{p2} & \ldots & A_{pp} \end{bmatrix} \tag{3.1}$$

where $A_{ij} \in \mathbb{R}^{m_i \times m_j}$, $\sum_{k=1}^{p} m_k = n$. Let $X = I_n$ and $Y = (A - sI_n)^T$ be matrices partitioned consistently with (3.1). Explicitly, the kth block column of the matrices Y and X are given by

$$Y_k = \begin{bmatrix} A_{k1} & \cdots & A_{kk-1} & A_{kk} - sI_{m_k} & \cdots & A_{kp} \end{bmatrix}^T \tag{3.2}$$

and

$$X_k = \begin{bmatrix} 0 & \cdots & I_{m_k} & \cdots & 0 \end{bmatrix}^T. \tag{3.3}$$

With this choice the expressions (2.5), (2.6) and (2.8) simplify to

$$U_k = X_k - \sum_{i=1}^{k-1} s^{-1} U_i T_i^{-1} V_i^T X_k, \tag{3.4}$$

$$V_k = Y_k - \sum_{i=1}^{k-1} s^{-1} V_i T_i^{-T} U_i^T Y_k \tag{3.5}$$

and

$$T_k = I_{m_k} + s^{-1} V_k^T X_k = I_{m_k} + s^{-1} V_{kk}^T. \tag{3.6}$$

To obtain the preconditioner, incomplete factors $\bar{U}_k$ and $\bar{V}_k$ are obtained by applying a drop strategy to the off-diagonal elements during the computation of U_k and V_k. Once the factors $\bar{U}_k$ and $\bar{V}_k$ have been computed, two different preconditioning strategies can be used

$$M_1 := s^{-1}I_n - s^{-2}\bar{U}\bar{T}^{-1}\bar{V}^T \tag{3.7}$$

and

$$M_2 := s^{-2}\bar{U}\bar{T}^{-1}\bar{V}^T\,. \tag{3.8}$$

In [12] both preconditioners are studied and compared showing similar performances in number of iterations. But, since M_2 requires fewer computations by iteration usually is preferred. The following algorithm computes the approximate factors for building the preconditioners.

Algorithm 1 Computing the incomplete factors in (3.7) and (3.8)

(1) **let** $U_k = X_k$, $V_k = Y_k$ $(k = 1, \ldots, p)$
(2) **for** $k = 1, \ldots, p$
(3) **for** $i = 1, \ldots, k-1$
$C_i = \bar{U}_i\bar{T}_i^{-1}\bar{V}_i^T$
$U_k = U_k - C_iA_0^{-1}X_k$
$V_k = V_k - C_i^TA_0^{-T}Y_k$
end for
compute $\bar{V}_k, \bar{U}_k$ **dropping elements in** V_k, U_k
$\bar{T}_k = I_{m_k} + \bar{V}_k^TA_0^{-1}X_k$
end for
(4) **return** $\bar{U} = [\bar{U}_1\,\bar{U}_2\,\cdots\,\bar{U}_p]$, $\bar{V} = [\bar{V}_1\,\bar{V}_2\,\cdots\,\bar{V}_p]$ and $\bar{T} = \mathrm{diag}(\bar{T}_1\,\bar{T}_2\,\cdots\,\bar{T}_p)$

Algorithm 1 runs to completion if the pivot matrices T_k are nonsingular. It will be shown that this condition holds for M-matrices and H-matrices. To prove these results we first show that the matrices T_k are closely related with the pivots of the block LU factorization applied to A.

LEMMA 3.1. *Let A be a matrix partitioned as in (3.1). If the block LU factorization of A can be done without pivoting, then*

$$T_k = s^{-1}A_{kk}^{(k-1)}$$

where $A_{kk}^{(k-1)}$ is the kth pivot in the block LU factorization of A.

Proof. Observe that the block (i,j) of the matrix $A^{(k)}$ obtained from the kth step of LU factorization is given by

$$\begin{aligned} A_{ij}^{(k)} &= A_{ij}^{(k-1)} - A_{ik}^{(k-1)}\left(A_{kk}^{(k-1)}\right)^{-1}A_{kj}^{(k-1)} \\ &= A_{ij} - \begin{bmatrix} A_{i1} & A_{i2} & \ldots & A_{ik} \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} & \ldots & A_{1k} \\ A_{21} & A_{22} & \ldots & A_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ A_{k1} & A_{k2} & \ldots & A_{kk} \end{bmatrix}^{-1} \begin{bmatrix} A_{1j} \\ A_{2j} \\ \vdots \\ A_{kj} \end{bmatrix} \end{aligned}$$

with $i,j > k$. Consider the matrix T_k given by (see section 2):

$$T_k = I_{m_k} + Y_k^TA_{k-1}^{-1}X_k. \tag{3.9}$$

We have

$$A_{k-1} = \left[\begin{array}{ccc|c|c|c} A_{11} & \dots & A_{1,k-1} & A_{1k} & \dots & A_{1p} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ A_{k-1,1} & \dots & A_{k-1,k-1} & A_{k-1,k} & \dots & A_{k-1,p} \\ \hline 0 & \dots & 0 & sI_{m_k} & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & 0 & \dots & sI_{m_p} \end{array}\right] = \left[\begin{array}{c|c|c|c} C_{11} & C_{12} & \dots & C_{1,p-k} \\ \hline 0 & sI_{m_k} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & sI_{m_p} \end{array}\right], \tag{3.10}$$

whose inverse is

$$A_{k-1}^{-1} = \begin{bmatrix} C_{11}^{-1} & -s^{-1}C_{11}^{-1}C_{12} & \dots & -s^{-1}C_{11}^{-1}C_{1,p-k} \\ 0 & s^{-1}I_{m_k} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & s^{-1}I_{m_p} \end{bmatrix}. \tag{3.11}$$

Then, by substituting (3.11) in (3.9), and bearing in mind (3.2) and (3.3), we have

$$\begin{aligned} T_k &= I_{m_k} - s^{-1} \begin{bmatrix} A_{k1} & \dots & A_{k,k-1} \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1,k-1} \\ A_{21} & A_{22} & \dots & A_{2,k-1} \\ \vdots & \vdots & \ddots & \vdots \\ A_{k-1,1} & A_{k-1,2} & \dots & A_{k-1,k-1} \end{bmatrix}^{-1} \begin{bmatrix} A_{1k} \\ A_{2k} \\ \vdots \\ A_{k-1,k} \end{bmatrix} + \\ &\quad s^{-1}(A_{kk} - sI_{m_k}) \\ &= s^{-1} \left(A_{kk} - \begin{bmatrix} A_{k1} & \dots & A_{k,k-1} \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1,k-1} \\ A_{21} & A_{22} & \dots & A_{2,k-1} \\ \vdots & \vdots & \ddots & \vdots \\ A_{k-1,1} & A_{k-1,2} & \dots & A_{k-1,k-1} \end{bmatrix}^{-1} \begin{bmatrix} A_{1k} \\ A_{2k} \\ \vdots \\ A_{k-1,k} \end{bmatrix} \right) \\ &= s^{-1} A_{kk}^{(k-1)}. \end{aligned}$$

□

This result generalizes Lemma 3.2 in [12] to the case of a matrix A partitioned in blocks. If A is either an M-matrix or an H-matrix, the block LU factorization can be done without pivoting [2]. Thus, Algorithm 1 runs to completion for these matrices when no dropping strategy is used. The following results show that this situation is also true in the incomplete case.

THEOREM 3.2. *Let A be an M-matrix partitioned as in (3.1). The matrices $\bar{T}_k$ computed by Algorithm 1 are nonsingular M-matrices.*

Proof. The proof proceeds by induction over k, $k = 1, \dots, p$. We will show that $\bar{T}_k$ are M-matrices, observing that

$$U_{ik} \geq \bar{U}_{ik} \geq 0, \qquad i \leq k, \tag{3.12}$$

$$V_{ik} \leq \bar{V}_{ik} \leq 0, \qquad i > k, \tag{3.13}$$

$$T_k^{-1} \geq \bar{T}_k^{-1} \geq 0. \tag{3.14}$$

1. For $k = 1$, we have $U_{11} = I_{m_1} \geq 0$. Moreover, $V_{i1} = A_{1i}^T \leq 0$ for $i > 1$ since A is an M-matrix. Therefore, after zeroing several elements, it follows that $V_{i1} \leq \bar{V}_{i1} \leq 0$. Observe now that $T_1 = s^{-1}A_{11}$. Then, after dropping off-diagonal elements, it follows that the incomplete block pivot $\bar{T}_1$ is an M-matrix since $\bar{T}_1 \geq T_1$. Moreover, $T_1^{-1} \geq \bar{T}_1^{-1} \geq 0$.
2. Now, assume that (3.12), (3.13) and (3.14) hold for $k-1$. For k, we have $U_{kk} = I_{m_k} \geq 0$. For $i < k$, it follows

$$U_{ik} = -\sum_{j=i}^{k-1} U_{ij} T_j^{-1} V_{kj}^T \geq -\sum_{j=i}^{k-1} \bar{U}_{ij} \bar{T}_j^{-1} \bar{V}_{kj}^T = \bar{U}_{ik} \geq 0.$$

For $i \neq k$

$$V_{ik} = A_{ki}^T - s^{-1} \sum_{j=1}^{k-1} V_{ij} T_j^{-T} \sum_{l=1}^{j} U_{lj}^T A_{kl}^T = A_{ki}^T - s^{-1} \sum_{j=1}^{k-1} \sum_{l=1}^{j} V_{ij} T_j^{-T} U_{lj}^T A_{kl}^T.$$

Since A is an M-matrix $A_{kl}^T \leq 0$ for $l \neq k$. Then, for $i > k$

$$-V_{ij} T_j^{-T} U_{lj}^T A_{kl}^T \leq -\bar{V}_{ij} \bar{T}_j^{-T} \bar{U}_{lj}^T A_{kl}^T \leq 0 \tag{3.15}$$

and

$$V_{ik} \leq A_{ki}^T - s^{-1} \sum_{j=1}^{k-1} \sum_{l=1}^{j} \bar{V}_{ij} \bar{T}_j^{-T} \bar{U}_{lj}^T A_{kl}^T = \bar{V}_{ik} \leq 0.$$

Now dropping elements in $\bar{U}_{ik}$ and $\bar{V}_{ik}$ and mantaining the same notation for the incomplete factors, the inequalities (3.12) and (3.13) hold for k.
Similarly, from (3.6)

$$\begin{aligned} T_k &= s^{-1} \left(A_{kk} - s^{-1} \sum_{j=1}^{k-1} \sum_{l=1}^{j} (V_{kj} T_j^{-T} U_{lj}^T A_{kl}^T)^T \right) \\ &\leq s^{-1} \left(A_{kk} - s^{-1} \sum_{j=1}^{k-1} \sum_{l=1}^{j} (\bar{V}_{kj} \bar{T}_j^{-T} \bar{U}_{lj}^T A_{kl}^T)^T \right) = \bar{T}_k. \end{aligned}$$

By equation (3.15) the matrix $\bar{T}_k$ has non-positive off-diagonal entries. Since T_k is an M-matrix, it follows that $\bar{T}_k$ is also an M-matrix and hence $T_k^{-1} \geq \bar{T}_k^{-1} \geq 0$.

□

THEOREM 3.3. *Let A be an H-matrix partitioned as in (3.1). The matrices $\bar{T}_k$ computed by Algorithm 1 are nonsingular H-matrices.*

Proof. To simplify the notation, let us denote by B the comparison matrix of A, $\mathcal{M}(A)$, and by (U_j^B, V_j^B, T_j^B) the matrices obtained by applying Algorithm 1 to matrix B.

As before, the proof proceeds by induction over k, $k = 1, \ldots, p$. We will show that $\bar{T}_k$ are H-matrices observing that

$$U_{ik}^B \geq \left|\bar{U}_{ik}\right| \geq 0 \quad i \leq k, \tag{3.16}$$

$$V_{ik}^B \leq -\left|\bar{V}_{ik}\right| \leq 0 \quad i > k, \tag{3.17}$$

$$(T_k^B)^{-1} \geq \left|\bar{T}_k^{-1}\right| \geq 0. \tag{3.18}$$

1. For $k = 1$, we have $\bar{U}_{11} = U_{11} = I_{m_1} \geq 0$. Thus, $\left|\bar{U}_{11}\right| = U_{11}^B$. On the other hand, $V_{i1} = A_{1i}^T$ for $i > 1$ and therefore $-\left|V_{i1}\right| = V_{i1}^B$. After dropping several elements, it follows that $-\left|\bar{V}_{i1}\right| \geq -\left|V_{i1}\right| = V_{i1}^B$. Observe now that, $T_1 = s^{-1}A_{11}$ and $\mathcal{M}(T_1) = T_1^B$ is an M-matrix. After dropping off-diagonal elements in T_1, it follows that $\mathcal{M}(\bar{T}_1) \geq T_1^B$ is an M-matrix too. Therefore, $\bar{T}_1$ is a nonsingular H-matrix and it follows that
$$\left|\bar{T}_1^{-1}\right| \leq \mathcal{M}\left(\bar{T}_1\right)^{-1} \leq \left(T_1^B\right)^{-1}.$$
2. Now, assume that (3.16), (3.17) and (3.18) hold for $k-1$. For k, we have $\bar{U}_{kk} = I_{m_k} = U_{kk}^B \geq 0$. For $i < k$, it follows
$$\begin{aligned}\left|\bar{U}_{ik}\right| &= \left|\frac{1}{s}\sum_{j=1}^{k-1}\bar{U}_{ij}\bar{T}_j^{-1}\bar{V}_{kj}^T\right| \leq \frac{1}{s}\sum_{j=1}^{k-1}\left|\bar{U}_{ij}\right|\left|\bar{T}_j^{-1}\right|\left|\bar{V}_{kj}^T\right| \\ &\leq -\frac{1}{s}\sum_{j=1}^{k-1}U_{ij}^B\left(T_j^B\right)^{-1}\left(V_{kj}^B\right)^T = U_{ik}^B.\end{aligned}$$
In addition, for $i > k$ one has:
$$\begin{aligned}-\left|\bar{V}_{ik}\right| &= -\left|A_{ki}^T - \frac{1}{s}\sum_{j=1}^{k-1}\bar{V}_{ij}\bar{T}_j^{-T}\left(\sum_{l=1}^{j}\bar{U}_{lj}^T A_{kl}^T\right)\right| \\ &\geq -\left|A_{ki}^T\right| - \frac{1}{s}\sum_{j=1}^{k-1}\sum_{l=1}^{j}\left|\bar{V}_{ij}\bar{T}_j^{-T}\bar{U}_{lj}^T A_{kl}^T\right| \\ &\geq -\left|A_{ki}^T\right| - \frac{1}{s}\sum_{j=1}^{k-1}\sum_{l=1}^{j}\left|\bar{V}_{ij}\right|\left|\bar{T}_j^{-T}\right|\left|\bar{U}_{lj}^T\right|\left|A_{kl}^T\right|\end{aligned}$$
By applying (3.16), (3.17) and (3.18) it follows
$$-\left|\bar{V}_{ik}\right| \geq -\left|A_{ki}^T\right| + \frac{1}{s}\sum_{j=1}^{k-1}\sum_{l=1}^{j}V_{ij}^B\left(T_j^B\right)^{-T}\left(U_{lj}^B\right)^T\left|A_{kl}^T\right| = V_{ij}^B$$
Now dropping elements in $\bar{U}_{ik}$ and $\bar{V}_{ik}$ and mantaining the same notation for the incomplete factors, the inequalities (3.16) and (3.17) hold for k.
In addition,
$$\bar{T}_k = I + \frac{1}{s}\bar{V}_{kk}^T = \frac{1}{s}\left(A_{kk} - \frac{1}{s}\sum_{j=1}^{k-1}\sum_{l=1}^{j}\left(\bar{V}_{kj}\bar{T}_j^{-T}\bar{U}_{lj}^T A_{kl}^T\right)^T\right)$$

We now compare the matrices $\mathcal{M}(\bar{T}_k)$ and T_k^B element by element. We denote by $R_m(\cdot)$ and $C_m(\cdot)$ the mth row and column of a matrix. Considering the diagonal elements, we have

$$\bar{T}_k(m,m) = \frac{1}{s} A_{kk}(m,m) - \frac{1}{s^2} \sum_{j=1}^{k-1} \sum_{l=1}^{j} R_m\left(\bar{A}_{kl}^T\right) \bar{U}_{lj} \bar{T}_j^{-1} C_m\left(\bar{V}_{kj}^T\right).$$

Then,

$$\begin{aligned}\left|\bar{T}_k(m,m)\right| &\geq \frac{1}{s} \left|A_{kk}(m,m)\right| - \frac{1}{s^2} \sum_{j=1}^{k-1} \sum_{l=1}^{j} \left|R_m\left(A_{kl}^T\right) \bar{U}_{lj} \bar{T}_j^{-1} C_m\left(\bar{V}_{kj}^T\right)\right| \\ &\geq \frac{1}{s} \left|A_{kk}(m,m)\right| - \frac{1}{s^2} \sum_{j=1}^{k-1} \sum_{l=1}^{j} R_m\left(\left|A_{kl}^T\right|\right) \left|\bar{U}_{lj}\right| \left|\bar{T}_j^{-1}\right| C_m\left(\left|\bar{V}_{kj}^T\right|\right).\end{aligned}$$

By applying (3.16), (3.17) and (3.18) it follows,

$$\begin{aligned}\left|\bar{T}_k(m,m)\right| &\geq \frac{1}{s} \left|A_{kk}(m,m)\right| + \frac{1}{s^2} \sum_{j=1}^{k-1} \sum_{l=1}^{j} R_m\left(\left|A_{kl}^T\right|\right) U_{lj}^B \left(T_j^B\right)^{-1} C_m\left(\left(V_{kj}^B\right)^T\right) \\ &= T_k^B(m,m).\end{aligned}$$

Similarly, one has $-\left|\bar{T}_k(m,n)\right| \geq T_k^B(m,n)$ for all $m \neq n$. Then,

$$\mathcal{M}\left(\bar{T}_k\right) \geq T_k^B.$$

By Theorem 3.2 it follows that T_k^B is an M-matrix and hence $\bar{T}_k$ is an H-matrix which implies that $\left|\bar{T}_k^{-1}\right| \leq \mathcal{M}\left(\bar{T}_k\right)^{-1} \leq \left(T_k^B\right)^{-1}$.

$\square$

4. Numerical experiments. In this section, the results of numerical experiments performed with the block approximate inverse preconditioner based on the Sherman-Morrison-Woodbury are presented. We will refer to it as AISMB preconditioner. The test matrices come from the Harwell Boeing [18] collection and Tim Davis' collection [16]. Table 4.1 provides data on the size, the number of nonzeros and the problem type. The most complex problems correspond to the matrices SHERMAN2 and UTM*. Matrix SHERMAN2 has been reported as a difficult problem for approximate inverse preconditioners by different authors [3, 9, 21, 22]. In [6] the matrices SHERMAN2 and UTM5940 are solved by applying permutation and scalings in order to place large entries in the diagonal of the matrix.

Algorithm 1 was implemented in Matlab with small modifications. Noting that equations (3.4) and (3.5) can be rewritten as

$$U_k = X_k - \sum_{i=1}^{k-1} s^{-1} U_i T_i^{-1} V_i^T X_k = \left(I_n - \sum_{i=1}^{k-1} s^{-1} U_i T_i^{-1} V_i^T\right) X_k\,,$$

$$V_k = Y_k - \sum_{i=1}^{k-1} s^{-1} V_i T_i^{-T} U_i^T Y_k = \left(I_n - \sum_{i=1}^{k-1} s^{-1} U_i T_i^{-1} V_i^T\right)^T Y_k\,,$$

Table 4.1
Size (n) and number of nonzero entries (nnz) of tested matrices.

Matrix	n	nnz	Description
ADD20	2395	17319	Circuit analysis
FS5414	541	4285	Chemical kinetics
HOR131	434	4710	Network flow
ORSIRR1	1030	6858	Oil reservoir simulation
ORSIRR2	886	5970	Oil reservoir simulation
ORSREG1	2205	14133	Oil reservoir simulation
SAYLR3	1000	3750	Oil reservoir simulation
SAYLR4	3564	22316	Oil reservoir simulation
SHERMAN2	1080	23094	Oil reservoir simulation
UTM1700B	1700	21509	Plasma physics
UTM3060	3060	42211	Plasma physics
UTM5940	5940	83842	Plasma physics

then the inner loop (3) in Algorithm 1 was avoided by performing rank-m updates and using vectorized operations. Thus, a faster code was obtained.

An artificial right hand side vector was generated such that the solution is the vector of all ones. No significative differences were observed for other choices of the right hand side vector. The iterative method employed was the BiCGSTAB method [30] with left preconditioning as described in [5]. All the tests where performed in Matlab running on a node of a cluster IBM 1350. The initial guess was always $x_0 = 0$, and the iterative solver was stopped when the initial residual was reduced at least by a factor of 10^{-8}, or after 2000 iterations.

To preserve sparsity, we follow the strategy recommended in [12]. Fill-in on U is reduced by removing the entries whose absolute value is less than a given drop tolerance namely *tol*. For the factor V a threshold relative to the maximum element of the matrix $|A|$ is used instead, i.e, an element is annihilated if its absolute value is less than *tol* times $\max |A|$. Usually the choice $tol = 0.1$ gives good results, but other values are used for specific matrices as indicated in the tables.

To invert the block pivots T_k, we have used exact and sparse incomplete LU factorization with different thresholds. In all cases the Matlab function `luinc` was used. For the matrices SHERMAN2 and UTM* the inversion with the incomplete LU factorization produced the breakdown of the algorithm due to almost singular block pivots. Only the exact factorization was used in these cases, but in order to preserve sparsity on the U and V factors, an additional drop strategy consisting in keeping only a maximum of $lfil$ elements by column was adopted.

The results for several block sizes are given in Tables 4.2 and 4.3. For simplicity the block size was chosen to be an exact factor of the matrix size, but variable block size can be employed instead. In Tables 4.2 and 4.3, m represents the block size, *lfil* the maximum number of nonzero elements in each column of the factors $\bar{U}$ and $\bar{V}$, ρ is the sparsity index, computed as the ratio between the number of nonzero elements of the preconditioner factors and the number of nonzero elements of the matrix, the number of iterations is represented by Its. As recommended by the authors in [12], we always took $s = 1.5$. The considered value of *tolU* and *tolV* is indicated in the caption of Tables 4.2 and 4.3, in general it is equal to 0.1 except for matrices ADD20 and UTM*.

TABLE 4.2
Results with different strategies for inverting T_k.

		Exact			*Tol ILU* $= 0.1$			*Tol ILU* $= 0.01$		
Matrix	m	ρ	*Its* M_1	*Its* M_2	ρ	*Its* M_1	*Its* M_2	ρ	*Its* M_1	*Its* M_2
	1	1.13	7	8	1.13	7	8	1.13	7	8
	2	1.14	7	8	1.13	7	8	1.13	7	8
ADD20	5	1.42	6	7	1.11	7	8	1.25	7	8
$tol = 0.01$	10	1.49	6	7	1.10	7	8	1.25	7	8
	20	1.62	6	7	1.09	7	8	1.27	7	8
	30	1.76	5	7	1.09	7	8	1.28	7	7
	40	1.84	5	7	1.01	7	8	1.30	7	7
	1	1.13	46	47	1.13	46	47	1.13	46	47
	2	1.20	40	43	1.16	41	43	1.19	40	42
HOR131	5	1.40	32	36	1.20	37	38	1.36	35	35
$tol = 0.1$	10	2.02	28	31	1.24	35	33	1.83	27	30
	20	2.96	24	26	1.32	34	32	2.5	26	26
	30	4.13	25	29	1.65	32	34	3.35	26	28
	40	5.31	24	24	1.75	34	33	3.97	23	27
	1	1.70	33	33	1.70	33	33	1.70	33	33
	2	2.08	34	34	1.63	36	36	1.69	35	35
ORSIRR1	5	2.62	33	33	1.61	37	37	1.81	34	33
$tol = 0.01$	10	3.61	33	33	1.55	38	38	2.17	33	35
	20	5.24	29	27	1.44	37	38	2.69	29	29
	30	6.28	22	24	1.39	35	36	2.81	27	27
	50	7.94	20	21	1.36	39	39	3.04	26	26
	1	1.70	34	36	1.70	34	36	1.70	34	36
	2	2.06	34	34	1.65	35	33	1.69	37	37
ORSIRR2	5	2.55	34	33	1.63	35	35	1.79	36	35
$tol = 0.01$	10	3.50	33	33	1.58	36	36	2.13	36	34
	20	5.18	28	29	1.46	33	35	2.68	29	28
	30	6.24	23	23	1.44	39	35	2.78	26	24
	50	7.91	20	21	1.45	38	36	3.13	25	26
	1	1.21	38	37	1.21	38	37	1.21	38	37
	5	1.38	37	43	1.19	43	44	1.33	40	40
ORSREG1	15	1.43	36	40	1.19	42	38	1.37	41	43
$tol = 0.1$	35	1.82	44	36	1.16	43	39	1.77	40	35
	45	1.97	36	37	1.15	43	40	1.91	35	36
	63	2.39	29	27	1.13	48	41	2.06	31	31
	1	1.84	37	40	1.84	37	40	1.84	37	40
	2	1.84	34	41	1.73	36	37	1.80	37	41
SAYLR3	5	2.13	34	41	1.77	39	42	1.96	39	44
$tol = 0.1$	10	2.40	34	40	1.76	38	42	2.02	37	44
	20	3.21	26	30	1.58	36	41	2.46	29	33
	25	3.74	26	28	1.61	41	38	2.82	28	30
	50	5.64	24	26	1.72	38	42	3.84	27	31
	1	3.19	47	46	3.19	47	46	3.19	47	46
	2	2.63	47	43	2.64	47	43	2.64	47	43
SAYLR4	5	2.45	47	47	2.46	42	42	2.46	42	42
$tol = 0.1$	10	2.52	45	45	2.52	42	47	2.52	47	47
	20	2.72	42	42	2.68	45	45	2.69	42	41
	50	6.91	37	38	2.77	43	42	2.78	42	42

Considering the results in Table 4.2, one can observe that the number of iterations tends to decrease when increasing the block size. On the other hand, fill-in increases for the exact LU inversion of the pivots, which is controlled using inexact LU. In fact, we can observe for ILU(0.1) that in most cases fill-in decreases without affecting negatively the performance.

As observed in Table 4.3, for the matrices SHERMAN2, UTM3060 and UTM5940, the iterative method only converged when preconditioning with the block strategy presented was used, i.e, with $m > 1$. These matrices have been reported in the bibliography as difficult to solve for both approximate inverse and ILU type preconditioners. In practice, reordering techniques and diagonal scaling have been used to achieve convergence (see [6]). For the SHERMAN2 matrix using a block size multiple

TABLE 4.3
Exact inversion and additional dropping strategy.

Matrix	m	ρ	*Its* M_1	*Its* M_2	ρ	*Its* M_1	*Its* M_2
		lfil $= n$			*lfil* $= 20$		
SHERMAN2 $tol = 0.1$	1	5.18	†	†	0.78	†	†
	6	0.9	38	38	0.59	40	41
	12	1.21	38	37	0.80	40	41
	18	1.45	38	35	0.95	40	39
	24	1.50	38	35	0.99	43	43
	54	2.55	29	29	1.30	34	34
	72	3.38	14	14	1.43	16	16
		lfil $= n$			*lfil* $= 40$		
UTM3060 $tol = 0.5$	1	5.95	†	†	2.21	†	†
	5	1.89	†	†	1.47	†	†
	10	2.15	537	614	1.56	1148	928
	20	2.85	525	515	1.93	623	588
	30	3.48	446	501	2.28	518	667
	45	4.10	405	475	2.63	742	770
		lfil $= n$			*lfil* $= 40$		
UTM5940 $tol = 0.5$	1	13.44	†	†	2.40	†	†
	5	1.83	†	†	1.45	†	†
	10	2.01	1555	1435	1.49	1700	1531
	20	2.61	940	982	1.81	1254	1204
	30	3.28	932	972	2.20	1596	1616
	45	4.20	751	778	2.60	1364	1140
		lfil $= n$			*lfil* $= 50$		
UTM1700B $tol = 0.2$	1	7.83	545	383	2.64	†	†
	2	7.45	369	457	3.14	1879	1818
	5	11.83	375	399	3.58	797	789
	10	9.85	174	198	3.67	491	433
	20	12.5	165	175	4.05	352	390

of 6, the problem was solved quite easily. Fill-in was controlled with *lfil*=20 without degrading the performance. For the UTM* matrices, the results for *lfil* different from n were not so satisfactory. Although the size of the preconditioner was reduced, the number of iterations was almost double in some cases. By applying reorderings to these matrices better results can be obtained as indicated in Table 4.6.

Although preconditioning computation times are not listed, we found that they decrease for larger block sizes until a maximum value of m is reached which depends on the matrix size. This can be expected since block algorithms can exploit more efficiently the memory hierarchy of the computer. Moreover, the use of block algorithms facilitates the use of vectorized operations, improving the speed of Matlab calculations. As result, the overall performance for the block preconditioner AISMB was better than for point AISM ($m = 1$) for the set of problems tested.

Concerning the preconditioners M_1 and M_2, both performed similarly. But since M_2 is slightly cheaper and easier to be applied, the results in the next section will be presented only with this preconditioner.

Effect of reorderings. Symmetric reorderings applied to the coefficient matrix can be very beneficial for factorized approximate inverse preconditioners as reported by several authors [10, 11, 19]. Usually this benefit consists of a reduction of a reduction in the preconditioner fill-in and a faster convergence of the Krylov iterative method employed to solve the linear system. In this subsection, we evaluate the effect of several symmetric reorderings on the AISMB and point AISM preconditioners. We considered the minimum degree (mmd) [29], generalized nested dissection (gnd) [20] and reverse Cuthill-McKee (rcm) [15] reorderings. A detailed description of these algorithms can also be found in [17]. The permutation matrix P is computed using the structure of the matrix $A + A^T$. Once it is computed, we solve the linear system $P^T APy = P^T b$ and we obtain the solution as $x = Py$.

Table 4.4
Results for point AISM. The tol parameter is the same as in Table 4.2.

	original		mmd		gnd		rcm	
Matrix	ρ	*Its* M_2	ρ	*Its* M_2	ρ	*Its* M_2	ρ	*Its* M_2
ADD20	1.13	8	0.73	7	0.73	10	0.80	7
HOR131	1.13	46	0.67	44	0.79	45	1.13	43
ORSIRR1	1.70	33	1.57	33	1.41	47	1.82	33
ORSIRR2	1.70	36	1.60	33	1.50	45	1.79	46
ORSREG1	1.21	37	1.12	39	1.12	38	1.45	35
SAYLR3	1.84	40	1.08	36	1.08	36	1.52	43
SAYLR4	3.19	46	2.02	42	1.78	43	2.79	44

In Table 4.4 we observe that point AISM benefits from reorderings since with sparser preconditioners the number of iterations remains the same or even improves. In addition, as with other sparse approximate inverse preconditioners [10], the best reorderings seem to be mmd and gnd.

Concerning the block preconditioner presented, Table 4.5 shows the results for the SAYLR4 which is representative for the set of matrices tested in Table 4.2. The results correspond to the inexact inversion of the block pivots T_k with ILU. As for the point case we found that with $tol = 0.01$ a sparser AISMB preconditioner can be computed using the mmd and gnd reorderings while the convergence was acquired with the same or even fewer iterations. Moreover, when the block size increases both the number of iterations and fill-in of the preconditioner tend to be smaller. Therefore, we can also conclude that AISMB is a faster algorithm with the reorderings tested.

However, for several structured matrices, it is essential to exploit the matrix structure to achieve convergence. In these cases the use of reorderings can produce poor results since the structure is destroyed. This is what is also noted in the case of SHERMAN2, which after reordering the matrix, the convergence was not reached for any block size. In summary, we found for this matrix that the best strategy corresponds to the use of blocks with original ordering.

The results for the UTM* matrices are listed in Table 4.6. The use of reorderings allowed smaller drop tolerances than those used for the experimental results in Table 4.3 with a small penalty on the density of the factors. We can see how for the original matrix with drop tolerance equal to 0.1 the fill-in of the preconditioner increases. Therefore, with the mmd and gnd reorderings a better preconditioner was computed as observed from the number of iterations needed to converge and the density of the preconditioner. We note that this is a quite challenging matrix set as we

TABLE 4.5
Results of different reordering strategies for the matrix SAYLR4 and inverting the blocks T_k with ILU(0.1).

	original($tol = 0.1$)		mmd($tol = 0.01$)		gnd($tol = 0.01$)		rcm($tol = 0.1$)	
m	ρ	*Its* M_2	ρ	*Its* M_2	ρ	*Its* M_2	ρ	*Its* M_2
1	3.19	46	2.02	43	1.78	42	2.8	45
2	2.64	43	1.92	41	1.73	42	2.81	43
5	2.46	42	1.81	40	1.66	41	2.82	44
10	2.52	47	1.74	43	1.60	41	2.84	44
20	2.68	45	1.69	43	1.55	41	2.84	44
50	2.77	42	1.64	40	1.50	42	2.84	43

observed an erratic behaviour on the convergence rate since the number of iterations does not decrease uniformly when increasing the block size. Nevertheless, we found again that with AISMB the number of iterations tends to be smaller than with point AISM.

TABLE 4.6
Results for the UTM matrices for different orderings.*

		original		mmd		gnd		rcm	
Matrix	m	ρ	*Its* M_2	ρ	*Its* M_2	ρ	*Its* M_2	ρ	*Its* M_2
UTM3060 $tol = 0.1$	1	13.37	333	3.34	199	3.07	192	9.81	201
	5	20.30	305	4.38	186	4.37	196	12.3	191
	10	23.16	282	5.02	137	5.36	229	14.6	128
	20	34.70	334	5.92	156	6.41	218	18.6	134
	45	41.96	186	6.23	147	8.15	120	28.5	218
UTM1700B $tol = 0.1$	1	11.49	152	3.01	126	2.80	153	7.43	109
	5	21.23	336	5.45	188	4.28	216	10.01	112
	10	18.06	103	6.21	209	4.99	130	11.99	110
	20	19.90	72	7.68	259	5.78	98	14.36	150
	34	24.16	63	8.01	120	6.61	112	15.93	154
UTM5940 $tol = 0.1$	1	19.43	688	3.59	833	3.2	627	14.37	914
	5	38.14	1015	4.60	571	4.52	825	20.84	1057
	10	32.64	471	5.23	603	5.5	844	25.4	1279
	20	34.53	299	5.83	740	6.88	808	36.9	1132
	45	48.79	402	6.48	655	8.62	584	48.1	†

5. Conclusions. In this paper we have presented a block approximate inverse preconditioner based on the Sherman-Morrison-Woodbury formula. It is a generalization of the AISM preconditioner presented in [12] for rank-m updates. We have been able to prove the convergence of the algorithm for M-matrices and H-matrices which also extends results already presented for AISM.

In order to evaluate the performance of the block preconditioner, it has been compared with point AISM for a set of matrices arising from different applications. The main conclusion is that the block version, AISMB, outperformed point AISM for the tested problems. Even in some cases, as for the matrix SHERMAN2, convergence was only attained with AISMB. In general, the larger block size, the smaller number of iterations needed to converge. Moreover, except for the UTM* matrices, fill-in was successfully limited by inexact inversion of the block pivots with incomplete factorizations as ILU. For the UTM* matrices, breakdown problems were encountered during the inexact inversion of pivots, and therefore, performing exact inversion together with a second drop strategy based on a maximum number of elements by column was used to solve these problems.

In addition, the effect on the quality of the preconditioner after applying symmetric reorderings to the coefficient matrix was evaluated. In particular, the minimum degree (mmd), generalized nested dissection (gnd) and reverse Cuthill-McKee (rcm) reorderings were used. We observed that for both AISMB and point AISM, mmd and gnd performed similarly and always better than rcm and the original ordering. This conclusion is in concordance with the results observed for others approximate inverse preconditioners [10]. Nevertheless, we should point out that for some highly-structured matrices, reordering can produce poor results as those observed for the SHERMAN2 matrix.

Finally, except for SHERMAN2, the block size was choosen without taking into account any particular structure of the matrix. This is important because it allows the choice of an optimal block size based on cache memory efficiency considerations. In this sense, we may expect even better results for structured matrices or when using graph compression algorithms to identify dense blocks in order to exploit high-level BLAS with dense linear algebra computations [1].

REFERENCES

[1] C. Ashcraft. Compressed graphs and the minimum degree algorithm. *SIAM J. Sci. Comput.*, 16:1404–1411, 1995.

[2] Owe Axelsson. *Iterative Solution Methods.* Cambridge University Press, 1996.

[3] 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.*, 13:107–123, 1999.

[4] S. T. Barnard and M. J. Grote. A block version of the SPAI preconditioner. In *Proc. 9th SIAM Conf. on Parall. Process. for Sci. Comp.*, March 1999.

[5] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* SIAM, Philadelphia, 1994.

[6] M. Benzi, J.C. Haws, and M. Tůma. Preconditioning highly indefinite and nonsymmetric matrices. *SIAM J. Sci. Comput.*, 22(4):1333–1353, 2000.

[7] M. Benzi, R. Kouhia, and M. Tůma. Stabilized and block approximate inverse preconditioners for problems in solid and structural mechanics. *Comput. Methods Appl. Mech. Engrg.*, 190:6533–6554, 2001.

[8] M. Benzi and M. Tůma. A sparse approximate inverse preconditioner for nonsymmetric linear systems. *SIAM J. Sci. Comput.*, 19:968–994, 1998.

[9] M. Benzi and M. Tůma. A comparative study of sparse approximate inverse preconditioners. *Applied Numerical Mathematics*, 30(2–3):305–340, 1999.

[10] M. Benzi and M. Tůma. Orderings for factorized sparse approximate inverse preconditioners. *SIAM J. Sci. Comput.*, 21(5):1851–1868, 2000.

[11] R. Bridson and W.-P. Tang. Ordering, anisotropy and factored sparse approximate inverses. *SIAM J. Sci. Comput.*, 21(867–882), 1999.

[12] R. Bru, J. Cerdán, J. Marín, and J. Mas. Preconditioning sparse nonsymmetric linear systems with the Sherman-Morrison formula. *SIAM J. Sci. Comput.*, 25(2):701–715, 2003.

[13] E. Chow and Y. Saad. Approximate inverse techniques for block-partitioned matrices. *SIAM J. Sci. Comput.*, 18:1657–1675, 1997.

[14] E. Chow and Y. Saad. Approximate inverse preconditioners via sparse-sparse iterations. *SIAM J. Sci. Comput.*, 19:995–1023, 1998.

[15] E. Cuthill and J. McKee. Reducing the bandwith of sparse symmetric matrices. In Brandon Press, editor, *Proc. 24th National Conference of the Association for Computing Machinery*, pages 157–172, New Jersey, 1969.

[16] T. Davis. University of Florida sparse matrix collection. *NA Digest*, 92(42), October 16, 1994. http://www.cise.ufl.edu/research/sparse/matrices. Retrieved April, 2002.

[17] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices*. Monograph on Numerical Analysis. Oxford Science Publications, New York, 1986.

[18] I. S. Duff, R. G. Grimes, and J. G. Lewis. User's guide for the Harwell-Boeing sparse matrix collection. Technical Report RAL-92-886, Rutherford Appleton Laboratory, Chilton, England, 1992.

[19] M. R. Field. Improving the performance of factorised approximate inverse preconditioner. Technical Report HDL-TR-98-199, Hitachi Dublin Laboratory, Dublin, Ireland, 1998.

[20] A. George. Nested dissection of a regular finite-element mesh. *SIAM J. Numer. Anal.*, 10:345–363, 1973.

[21] N. I. M. Gould and J. A. Scott. Sparse approximate inverse preconditioners using norm minimization techniques. *SIAM J. Sci. Comput.*, 19:605–625, 1998.

[22] M. Grote and T. Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM J. Sci. Comput.*, 18, 1997.

[23] L. Yu. Kolotilina and A. Yu. Yeremin. Factorized sparse appoximate inverse preconditionings I. Theory. *SIAM J. Matrix Anal. Applic.*, 14:45–58, 1993.

[24] T. A. Manteuffel. An incomplete factorization technique for positive definite linear systems. *Math. Comp.*, 34:473–497, 1980.

[25] 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.*, 31:148–162, 1977.

[26] A. M. Ostrowski. Über die Determinanten mit überwiegender Hauptdiagonale. *Comentarii Mathematici Helvetici*, 10:69–96, 1937.

[27] Y. Saad. ILUT: a dual threshold incomplete ILU factorization. *Numerical Linear Algebra with Applications*, 1:387–402, 1994.

[28] Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company, Boston, 1996.

[29] W. F. Tinney and J. W. Walker. Direct solutions of sparse network equation by optimally ordered triangular factorization. In *Proc. IEEE*, number 55, pages 1801–1809, 1967.

[30] H. A. van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 12(6):631–644, 1992.

[31] R. S. Varga. *Matrix Iterative Analysis*. Prentice–Hall, Englewood Cliffs, NJ, 1962.

[32] M. Woodbury. Inverting modified matrices. Technical report, Statistical Research Group, Princeton University, Princeton, NJ, 1950.