SIAM J. SCI. COMPUT.
Vol. 25, No. 2, pp. 701–715

PRECONDITIONING SPARSE NONSYMMETRIC LINEAR SYSTEMS WITH THE SHERMAN–MORRISON FORMULA*

R. BRU†, J. CERDÁN†, J. MARÍN†, AND J. MAS†

Abstract. Let $Ax = b$ be a large, sparse, nonsymmetric system of linear equations. A new sparse approximate inverse preconditioning technique for such a class of systems is proposed. We show how the matrix $A_0^{-1} - A^{-1}$, where A_0 is a nonsingular matrix whose inverse is known or easy to compute, can be factorized in the form $U\Omega V^T$ using the Sherman–Morrison formula. When this factorization process is done incompletely, an approximate factorization may be obtained and used as a preconditioner for Krylov iterative methods. For $A_0 = sI_n$, where I_n is the identity matrix and s is a positive scalar, the existence of the preconditioner for M-matrices is proved. In addition, some numerical experiments obtained for a representative set of matrices are presented. Results show that our approach is comparable with other existing approximate inverse techniques.

Key words. nonsymmetric linear systems, factorized sparse approximate inverses, Sherman–Morrison formula, preconditioned iterative methods

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

DOI. 10.1137/S1064827502407524

1. Introduction.

Let

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

be a linear system of n equations with n unknowns, where A is a large, sparse, nonsymmetric matrix. An approximate solution of (1.1) is usually obtained by using a preconditioned iterative Krylov-type method, such as the GMRES [22] and BiCGSTAB [24] methods.

In general, to obtain good convergence rates, or even to converge, these methods are applied to the right preconditioned linear system

$$AMy = b, \qquad x = My,$$

where the matrix M is the preconditioner. Left preconditioning and two-side preconditioning are also possible [21]. The matrix M should be chosen in such a way that the preconditioned matrix AM has better spectral properties. Typically, since $AM \approx I_n$ is desired, the eigenvalues should be clustered about 1. But, for most problems, other distributions of the eigenvalues may be satisfactory. If the eigenvalues of AM are clustered away from the origin, one can expect a good performance of the iterative solver. Moreover, the preconditioner should be easily and inexpensively computed and applied. Clearly, both requirements are difficult to fulfill for a general purpose preconditioner. A considerable effort is being made to develop suitable methods which perform well for a wide range of problems.

Several preconditioning techniques have been proposed. Among them, we are interested in the study of approximate inverse preconditioners. With such preconditioners an approximation of the inverse of A is computed and stored explicitly. As

*Received by the editors May 14, 2002; accepted for publication (in revised form) February 6, 2003; published electronically November 11, 2003. This research was supported by Spanish DGI grant BFM2001-2641.

http://www.siam.org/journals/sisc/25-2/40752.html

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

a result, the preconditioning step can be made by computing sparse matrix-vector products. Since this operation is easy to parallelize, this approach is particularly convenient when a parallel computer is to be used. This observation is motivation enough, yet recent studies show that sparse approximate inverse preconditioners are also of interest because of their robustness [2].

There are a number of approximate inverse preconditioners which appear in the literature. Among the most successful techniques, Frobenius norm minimization methods and factorized sparse approximate inverse preconditioners must be considered. Briefly, Frobenius norm methods compute a preconditioner by minimizing the norm $\|I_n - AM\|_F$, subject to some sparsity constraints on M. This problem reduces to solving independent linear least square problems. This feature makes them attractive for parallel environments. Examples of this class are the SPAI [17] and MR [12] preconditioners.

Factorized sparse approximate inverse preconditioners compute an approximation of the inverse of A in a different manner. If A has an LDU decomposition, $A = LDU$, these methods compute a sparse approximate inverse of the form $M = \bar{Z}\bar{D}^{-1}\bar{W}^T \approx A^{-1}$, where $\bar{Z} \approx U^{-1}$ and $\bar{W} \approx L^{-T}$ are upper triangular matrices, and $\bar{D} \approx D$ is a nonsingular diagonal matrix. For instance, the AINV method [7, 8] computes these sparse factors by means of an A-orthogonalization process on the unit vectors. This preconditioner, or its variant SAINV [3], has been successfully applied to a variety of challenging problems [4, 5], showing that it is one of the most effective approximate inverse techniques.

For a comparative study of these and other techniques we refer to [9]. In this work the authors conclude that factorized forms tend to deliver better convergence rates for the same amount of nonzero elements in the preconditioner. In addition, they can benefit from reorderings applied to the coefficient matrix A [4, 10].

This paper presents a new framework for computing factorized sparse approximate inverse preconditioners based on the Sherman–Morrison formula [23, 1], which establishes that, given a nonsingular $n \times n$ matrix B and two vectors x and y in $\mathbf{R}^n$ such that $r = 1 + y^T B^{-1} x \neq 0$, the matrix $A = B + xy^T$ is nonsingular, and its inverse is

$$A^{-1} = B^{-1} - r^{-1} B^{-1} x y^T B^{-1}. \tag{1.2}$$

The Sherman–Morrison formula (1.2), or its variants, have been applied in many contexts: to update linear models by using least squares in statistics, in networks and structures analysis to compute a new solution when the system is modified, to update a factorization of a matrix, etc. For a complete survey of the use of these formulas we refer to [18]. In particular, the formula can be used to invert a matrix, for example, by using the reinforcement method [16].

We will show how the matrix $A_0^{-1} - A^{-1}$ is factored in the form $U\Omega V^T$ using the Sherman–Morrison formula. If incomplete factors are computed by applying some dropping strategy, a sparse approximate factorization of the matrix $A_0^{-1} - A^{-1}$ is obtained which can be used as a preconditioner. In this paper we shall analyze the specific choice $A_0 = sI_n$, where I_n is the identity matrix and s is a positive scalar factor. It will be shown that the computation of the preconditioner is breakdown-free when A is an M-matrix.

Throughout the paper, we consider the kth column of A as a column vector denoted by a_k and the kth row of A as a row vector denoted by a^k. For the identity matrix I_n, e_k and e^k will be used.

The paper is organized as follows. In section 2 the factorization of the matrix $A_0^{-1} - A^{-1}$ is obtained. In section 3 this factorization is used as a preconditioner for $A_0 = sI_n$. Some theoretical properties are also shown. In section 4 the results of the numerical experiments for a representative set of matrices are presented. Finally, in section 5 the main conclusions are drawn.

2. Factorization of $A_0^{-1} - A^{-1}$ using the Sherman–Morrison formula. Let $\{x_k\}_{k=1}^n$ and $\{y_k\}_{k=1}^n$ be two sets of vectors in $\mathbf{R}^n$, and let A_0 be a nonsingular $n \times n$ matrix whose inverse is either known or easy to obtain, such that

$$A = A_0 + \sum_{k=1}^{n} x_k y_k^T. \tag{2.1}$$

If we define $A_k := A_0 + \sum_{i=1}^{k} x_i y_i^T$, $k = 1, \ldots, n$, then $A_{k+1} = A_k + x_{k+1} y_{k+1}^T$ and $A_n = A$.

If x_k and y_k are vectors such that $r_k = 1 + y_k^T A_{k-1}^{-1} x_k \neq 0$, then by applying (1.2) the inverse of the matrix A_k is given by

$$A_k^{-1} = A_{k-1}^{-1} - \frac{1}{r_k} A_{k-1}^{-1} x_k y_k^T A_{k-1}^{-1}, \qquad k = 1, \ldots, n.$$

Since $A^{-1} = A_n^{-1}$, it follows that

$$A^{-1} = A_0^{-1} - \sum_{k=1}^{n} \frac{1}{r_k} A_{k-1}^{-1} x_k y_k^T A_{k-1}^{-1}$$

or

$$A_0^{-1} - A^{-1} = \sum_{k=1}^{n} \frac{1}{r_k} A_{k-1}^{-1} x_k y_k^T A_{k-1}^{-1},$$

which can be written in matrix notation as

$$A_0^{-1} - A^{-1} = \Phi \Omega^{-1} \Psi^T, \tag{2.2}$$

where

$$\Phi = \begin{bmatrix} A_0^{-1} x_1 & A_1^{-1} x_2 & \cdots & A_{n-1}^{-1} x_n \end{bmatrix},$$

$$\Omega^{-1} = \begin{bmatrix} r_1^{-1} & & & \\ & r_2^{-1} & & \\ & & \ddots & \\ & & & r_n^{-1} \end{bmatrix}, \quad \text{and} \quad \Psi^T = \begin{bmatrix} y_1^T A_0^{-1} \\ y_2^T A_1^{-1} \\ \vdots \\ y_n^T A_{n-1}^{-1} \end{bmatrix}.$$

This process depends explicitly on the matrices $\{A_k\}_{k=1}^n$. The following result shows how the factorization (2.2) can be computed without these matrices explicitly appearing.

THEOREM 2.1. *Let A and A_0 be two nonsingular matrices, and let $\{x_k\}_{k=1}^n$ and $\{y_k\}_{k=1}^n$ be two sets of vectors such that condition (2.1) is satisfied. Moreover, suppose that $r_k = 1 + y_k^T A_{k-1}^{-1} x_k \neq 0$ for $k = 1, \ldots, n$, where $A_{k-1} = A_0 + \sum_{i=1}^{k-1} x_i y_i^T$. Then*

$$u_k := x_k - \sum_{i=1}^{k-1} \frac{v_i^T A_0^{-1} x_k}{r_i} u_i \tag{2.3}$$

and

$$v_k := y_k - \sum_{i=1}^{k-1} \frac{y_k^T A_0^{-1} u_i}{r_i} v_i \tag{2.4}$$

are well defined for $k = 1, \ldots, n$. *In addition, the relations*

$$\begin{array}{rcl} A_{k-1}^{-1} x_k & = & A_0^{-1} u_k, \\ y_k^T A_{k-1}^{-1} & = & v_k^T A_0^{-1}, \end{array} \tag{2.5}$$

and

$$r_k = 1 + y_k^T A_0^{-1} u_k = 1 + v_k^T A_0^{-1} x_k \tag{2.6}$$

are satisfied.

Proof. The proof proceeds by induction. For $k = 1$, $u_1 := x_1$ and $v_1 := y_1$ and the relations (2.5) and (2.6) are trivially satisfied. Suppose that for $i = 1, \ldots, k-1$ the relations are true. Applying the inductive process, the vectors u_k and v_k are well defined. Moreover,

$$\begin{array}{rcl} A_{k-1}^{-1} x_k & = & (A_0^{-1} - \sum_{i=1}^{k-1} \frac{1}{r_i} A_{i-1}^{-1} x_i y_i^T A_{i-1}^{-1}) x_k \\ & = & A_0^{-1} x_k - \sum_{i=1}^{k-1} \frac{y_i^T A_{i-1}^{-1} x_k}{r_i} A_{i-1}^{-1} x_i \\ & = & A_0^{-1} x_k - \sum_{i=1}^{k-1} \frac{v_i^T A_0^{-1} x_k}{r_i} A_0^{-1} u_i \\ & = & A_0^{-1} (x_k - \sum_{i=1}^{k-1} \frac{v_i^T A_0^{-1} x_k}{r_i} u_i) \\ & = & A_0^{-1} u_k \end{array}$$

and

$$\begin{array}{rcl} y_k^T A_{k-1}^{-1} & = & y_k^T (A_0^{-1} - \sum_{i=1}^{k-1} \frac{1}{r_i} A_{i-1}^{-1} x_i y_i^T A_{i-1}^{-1}) \\ & = & y_k^T A_0^{-1} - \sum_{i=1}^{k-1} \frac{y_k^T A_{i-1}^{-1} x_i}{r_i} y_i^T A_{i-1}^{-1} \\ & = & y_k^T A_0^{-1} - \sum_{i=1}^{k-1} \frac{y_k^T A_0^{-1} u_i}{r_i} v_i^T A_0^{-1} \\ & = & (y_k^T - \sum_{i=1}^{k-1} \frac{y_k^T A_0^{-1} u_i}{r_i} v_i^T) A_0^{-1} \\ & = & v_k^T A_0^{-1}. \end{array}$$

Finally, (2.6) is obtained by substituting (2.5) into $r_k = 1 + y_k^T A_{k-1}^{-1} x_k$. ∎

COROLLARY 2.2. *Under the assumptions of Theorem* 2.1, *let* U *and* V *be the matrices whose columns are the vectors* u_k *and* v_k *as defined in* (2.3) *and* (2.4), *respectively. Then*

$$A_0^{-1} - A^{-1} = A_0^{-1} U \Omega^{-1} V^T A_0^{-1}, \tag{2.7}$$

where $\Omega = \operatorname{diag}(r_1, r_2, \ldots, r_n)$. *Moreover, there exist unique unit upper triangular matrices* T_X *and* T_Y *such that*

$$X = U T_X, \quad Y = V T_Y, \tag{2.8}$$

where X *and* Y *are matrices whose columns are the vectors* x_i *and* y_i, *respectively,*

$$T_X = \begin{bmatrix} 1 & t_{12} & \cdots & & t_{1n} \\ 0 & 1 & & & t_{2n} \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & & 1 & t_{n-1,n} \\ 0 & 0 & \cdots & & 1 \end{bmatrix}, \qquad t_{ij} = \frac{v_i^T A_0^{-1} x_j}{r_i}, \quad i < j,$$

and

$$T_Y = \begin{bmatrix} 1 & \hat{t}_{12} & \cdots & & \hat{t}_{1n} \\ 0 & 1 & & & \hat{t}_{2n} \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & & 1 & \hat{t}_{n-1,n} \\ 0 & 0 & \cdots & & 1 \end{bmatrix}, \qquad \hat{t}_{ij} = \frac{y_j^T A_0^{-1} u_i}{r_i}, \quad i < j.$$

Thus, the matrix U (V) is nonsingular if and only if the matrix X (Y) is nonsingular.

Proof. Equation (2.7) follows from (2.2) and (2.5). Equation (2.8) corresponds to (2.3) and (2.4) written in matrix notation. ∎

We note that this factorization process resembles the rank reducing process used in [13] to characterize different factorizations of a matrix as a biconjugation process. In [13] the Wedderburn formula, which resembles the Sherman–Morrison formula, is used. But the sets of vectors $\{u_i\}$ and $\{v_i\}$ defined in Theorem 2.1 are not biconjugated.

3. Approximate inverse preconditioning based on the Sherman–Morrison formula. Different choices of the vectors $\{x_k\}$, $\{y_k\}$ and the matrix A_0 may lead to different factorizations of the matrix $A_0^{-1} - A^{-1}$. In this work we take

$$\begin{aligned} &A_0 = sI_n, \quad s > 0, \\ &x_k = e_k, \\ &y_k = (a^k - a_0^k)^T, \end{aligned} \tag{3.1}$$

where a^k and a_0^k denote the kth row of the matrices A and A_0, respectively, and e_k is the kth column of the identity matrix. In this case, the factorization (2.7) simplifies to

$$s^{-2} U \Omega^{-1} V^T = s^{-1} I_n - A^{-1}. \tag{3.2}$$

Denoting by $(x)_k$ the kth entry of a vector x, we rewrite the expressions for the vectors u and v (equations (2.3) and (2.4)) as

$$u_k := x_k - \sum_{i=1}^{k-1} \frac{(v_i)_k}{s r_i} u_i \tag{3.3}$$

and

$$v_k := y_k - \sum_{i=1}^{k-1} \frac{y_k^T u_i}{s r_i} v_i. \tag{3.4}$$

We note that since $x_k = e_k$, the matrix X in Corollary 2.2 is the identity matrix. Thus $U = T_X^{-1}$ is a nonsingular unit upper triangular matrix. Moreover, $V = Y T_Y^{-1}$ is nonsingular if and only if $s \notin \sigma(A)$, where $\sigma(A)$ is the spectrum of A. To compute the factorization (3.2) Algorithm 1 can be used.

Even if A is a sparse matrix, in general the matrices U and V in (3.2) tend to be dense. In order to preserve sparsity, one can apply different dropping strategies. Fill-in is reduced by removing suitable off-diagonal entries in the computation of the u and v vectors. A common rule consists of dropping an element if it is less in absolute value

ALGORITHM 1 (computation of the factorization (3.2)).

(1) **Set** $x_k = e_k$, $y_k = (a^k - se^k)^T$, $(k = 1, \ldots, n)$
(2) **for** $k = 1, \ldots, n$
  $u_k = x_k$
  $v_k = y_k$
  for $i = 1, \ldots, k-1$
    $u_k = u_k - \frac{(v_i)_k}{sr_i} u_i$
    $v_k = v_k - \frac{y_k^T u_i}{sr_i} v_i$
  end for
  $r_k = 1 + (v_k)_k / s$
end for
(3) **Return** $U = [u_1, u_2, \cdots, u_n]$, $V = [v_1, v_2, \cdots, v_n]$, and $\Omega = \mathrm{diag}(r_1, r_2, \ldots, r_n)$.

than a relative or absolute threshold. In this work the best results have been obtained using an absolute threshold for the factor U and relative for V, as is explained later.

After applying a dropping strategy the incomplete factorization algorithm computes sparse matrices $\bar{U}$ and $\bar{V}$ and a diagonal matrix $\bar{\Omega}$ such that

$$s^{-2}\bar{U}\bar{\Omega}^{-1}\bar{V}^T \approx s^{-1}I_n - A^{-1}. \tag{3.5}$$

If A is a nonsingular M-matrix, then the incomplete factorization runs to completion, and the pivots are strictly positive. Moreover, the pivots in the incomplete factorization are not smaller than the pivots in the exact factorization, as established below. We need the following results.

LEMMA 3.1. *Let u_k, v_k, and r_k be the vectors and pivots computed by the complete factorization algorithm, and let u_k^*, v_k^*, and r_k^* be the vectors and pivots computed by the complete factorization algorithm for $s^* = 1$, respectively. Then,*

$$u_k = u_k^*, \tag{3.6}$$

$$v_k = v_k^* - (s-1)w_k \quad \textit{where} \quad w_k = x_k - \sum_{i=1}^{k-1} \frac{(y_i^*)^T u_k^*}{r_i^*} w_i, \tag{3.7}$$

$$r_k = r_k^*/s. \tag{3.8}$$

Proof. The proof follows by induction over k. For $k = 1$, we have from (3.3) $u_1 = x_1 = u_1^*$. From (3.4) and (3.1), it follows that

$$v_1 = y_1 = (a^1 - se^1)^T = (a^1 - e^1)^T - (s-1)e_1 = y_1^* - (s-1)e_1 = v_1^* - (s-1)w_1.$$

Moreover,

$$\begin{aligned} r_1 &= 1 + \frac{1}{s} y_1^T u_1 = 1 + \frac{1}{s}(y_1^* - (s-1)e_1)^T u_1^* \\ &= 1 + \frac{1}{s}((y_1^*)^T u_1^* - (s-1)e_1^T u_1^*) = 1 + \frac{1}{s}((y_1^*)^T u_1^* - (s-1)) \\ &= \frac{1}{s}(1 + (y_1^*)^T u_1^*) = \frac{1}{s} r_1^*. \end{aligned} \tag{3.9}$$

Suppose that for $i = 1, \ldots, k-1$ the relations are true. Note that in (3.7) $(w_k)_i = 0$ for $i > k$ and $(v_k)_i = (v_k^*)_i$ for $i > k$. Then, from (3.3) $u_k = u_k^*$. Now observe that $y_k^T u_i = (y_k^*)^T u_i^*$ for $i < k$. From (3.4)

$$\begin{aligned} v_k &= (y_k^* - (s-1)e_k) - \sum_{i=1}^{k-1} \frac{(y_k^*)^T u_i^*}{r_i^*}(v_i^* - (s-1)w_i) \\ &= \left(y_k^* - \sum_{i=1}^{k-1} \frac{(y_k^*)^T u_i^*}{r_i^*} v_i^* \right) - (s-1)\left(e_k - \sum_{i=1}^{k-1} \frac{(y_k^*)^T u_i^*}{r_i^*} w_i \right) \\ &= v_k^* - (s-1)w_k. \end{aligned}$$

Finally, $r_k = r_k^*/s$ follows as in (3.9). ∎

LEMMA 3.2. *Let A be a nonsingular M-matrix. Then, the pivots r_k computed by the complete factorization (Algorithm 1) are positive.*

Proof. The pivots r_k^* as defined in Theorem 3.1 are the same pivots computed by the Gauss–Jordan algorithm without pivoting [11]. Since these pivots coincide with those appearing during the Gaussian elimination process, which are positive for an M-matrix [20], then $r_k = r_k^*/s > 0$ by (3.8). ∎

THEOREM 3.3. *Let A be a nonsingular M-matrix. The pivots r_k computed by the complete factorization (Algorithm 1) and the pivots $\bar{r}_k$ computed by the incomplete factorization algorithm verify $\bar{r}_k \geq r_k > 0$.*

Proof. We proceed by induction over k. We show that $\bar{r_k} \geq r_k > 0$ for $1 \leq k \leq n$. We show that the columns of the matrices U and V of the complete factorization algorithm and the columns of the matrices $\bar{U}$ and $\bar{V}$ of the incomplete one satisfy

$$\begin{aligned} &u_k \geq \bar{u}_k \geq 0 \qquad \text{(componentwise)}, \\ &0 \geq (\bar{v}_k)_j \geq (v_k)_j, \qquad j = k+1, \ldots, n. \end{aligned}$$

It is clear that $u_1 = x_1 = e_1 \geq 0$. Since $\bar{u}_1$ is obtained from u_1 zeroing some off-diagonal entries, then $u_1 \geq \bar{u}_1 \geq 0$. On the other hand, $y_1 = (a^1 - se^1)^T$ coincides with the first row of A, except in the first component. From the equality $v_1 = y_1$, since $\bar{v}_1$ is obtained annihilating some off-diagonal entries of v_1, one has $(v_1)_j \leq (\bar{v}_1)_j \leq 0$ for $j > 1$. Thus, $\bar{r}_1 = 1 + \frac{1}{s} y_1^T \bar{u}_1 \geq 1 + \frac{1}{s} y_1^T u_1 = r_1$. By Lemma 3.2 $\bar{r_1} \geq r_1 > 0$.

Now assume that for $i \leq k-1$

$$u_i \geq \bar{u_i} \geq 0, \qquad 0 \geq (\bar{v}_i)_j \geq (v_i)_j, \qquad j = i+1, \ldots, n,$$

holds, and $\bar{r}_i \geq r_i > 0$. Clearly,

$$0 \geq \frac{(\bar{v}_i)_k}{s\bar{r}_i} \bar{u}_i \geq \frac{(v_i)_k}{sr_i} u_i, \qquad i = 1, \ldots, k-1\,.$$

Then, $u_k = x_k - \sum_{i=1}^{k-1} \frac{(v_i)_k}{sr_i} u_i \geq x_k - \sum_{i=1}^{k-1} \frac{(\bar{v}_i)_k}{s\bar{r}_i} \bar{u}_i = \bar{u}_k \geq 0$. Since U is an unit upper triangular matrix, then

$$y_k^T u_i = (y_k)_i + \sum_{j=1}^{i-1} (y_k)_j (u_i)_j \leq (y_k)_i + \sum_{j=1}^{i-1} (y_k)_j (\bar{u}_i)_j = y_k^T \bar{u}_i \leq 0, \quad i = 1, \ldots, k-1,$$

and then

$$\frac{y_k^T u_i}{sr_i} \leq \frac{y_k^T \bar{u}_i}{s\bar{r}_i} \leq 0, \quad i = 1, \ldots, k-1.$$

By the induction hypothesis over the entries of the lower triangular part of V,

$$0 \leq \frac{y_k^T \bar{u}_i}{s\bar{r}_i}(\bar{v}_i)_j \leq \frac{y_k^T u_i}{s r_i}(v_i)_j, \quad j = k+1, \ldots, n, \quad i = 1, \ldots, k-1.$$

Then

$$0 \geq (\bar{v}_k)_j = (y_k)_j - \sum_{i=1}^{k-1} \frac{y_k^T \bar{u}_i}{s\bar{r}_i}(\bar{v}_i)_j \geq (y_k)_j - \sum_{i=1}^{k-1} \frac{y_k^T u_i}{s r_i}(v_i)_j = (v_k)_j, \quad j = k+1, \ldots, n.$$

Finally, $\bar{r}_k = 1 + \frac{y_k^T \bar{u}_k}{s} \geq 1 + \frac{y_k^T u_k}{s} = r_k > 0.$ ∎

Remark 3.4. In Theorem 3.3 it has been proved for a given M-matrix A that

$$\begin{aligned} U \geq \bar{U} \geq 0, & \\ \bar{\Omega}_{ii} \geq \Omega_{ii} > 0, & \qquad i = 1, \ldots, n, \\ 0 \geq \bar{V}_{ij} \geq V_{ij}, & \qquad i > j. \end{aligned}$$

In addition, if $s > \max_{i=1,\ldots,n}\{a_{ii}\}$, then the last relation extends to $0 \geq \bar{V} \geq V$.

Remark 3.5. Observe that a column oriented version of Algorithm 1 can be obtained with the choice $x_k = a_k - se_k$ and $y_k = e_k$, which is equivalent to applying the algorithm to A^T. Since the M-matrix property is invariant under transposition, the theoretical study remains true. Therefore, the same study made for the $\bar{U}$ and $\bar{V}$ factors applies to the $\bar{V}$ and $\bar{U}$ factors of the column oriented version.

The incomplete factorization (3.5) can be used to define two different preconditioner matrices:

$$M_1 := s^{-1} I_n - s^{-2} \bar{U} \bar{\Omega}^{-1} \bar{V}^T \approx A^{-1} \tag{3.10}$$

and

$$M_2 := s^{-2} \bar{U} \bar{\Omega}^{-1} \bar{V}^T \approx s^{-1} I_n - A^{-1}. \tag{3.11}$$

The relation between these preconditioners is given by

$$M_2 = s^{-1} I_n - M_1. \tag{3.12}$$

The spectrum of the preconditioned matrix for these preconditioners is

$$\begin{aligned} \sigma(AM_1) &\approx \sigma(I_n), \\ \sigma(AM_2) &\approx \frac{\sigma(A)}{s} - 1. \end{aligned} \tag{3.13}$$

If $s > \rho(A)$, the spectrum of the matrix AM_2 is contained in the left-half complex plane which is good for Krylov iterative methods. Moreover, compared with M_1, the application of the preconditioner M_2 over a vector requires less arithmetic operations per iteration.

In the next section the results of the numerical experiments with these strategies for a representative set of matrices are presented.

TABLE 4.1
Size (n) and number of nonzero elements (nnz) of the test matrices.

Matrix	n	nnz	Description
ADD20	2395	17319	Circuit simulation
FS5414	541	4285	Chemical kinetics
HOR131	434	4710	Network flow
ORSIRR1	1030	6858	Reservoir simulation
ORSIRR2	886	5970	Reservoir simulation
ORSREG1	2205	14133	Reservoir simulation
SAYLR3	1000	3750	Reservoir simulation
SAYLR4	3564	22316	Reservoir simulation
SHERMAN1	1000	3750	Reservoir simulation
WATT1	1856	11360	Petroleum engineering
WATT2	1856	11550	Petroleum engineering

4. Numerical experiments. In this section, the row and column oriented versions of the two different Sherman–Morrison-based preconditioning strategies (equations (3.10) and (3.11)) are compared. The experiments show that both preconditioners exhibit similar performance for the tested problems. In addition, they are compared with the AINV preconditioner [7, 8], which has been widely compared with other preconditioning techniques showing good performance [9].

The test matrices come from the Harwell Boeing [15] collection and Tim Davis's collection [14]. Table 4.1 shows the size, number of nonzeros, and the problem where each matrix arises. A solution vector with random entries uniformly distributed in $(0, 1)$ was used to construct the right-hand side of each linear system. No significant differences were observed for other choices of the right-hand side vector. An approximate solution was computed with the BiCGSTAB method [24] with right preconditioning. The iterative solver and the preconditioner AINV are coded in Fortran 77 and compiled with -O3 optimization using the IFC Intel Fortran compiler. Fortran codes were kindly provided by M. Benzi. The results were obtained by running the codes on a 450 MHz Intel Pentium III computer. 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 simplify the application of the preconditioner, the $\bar{V}$ factor used was the product of the factors $\bar{\Omega}^{-1}$ and $\bar{V}$ of the approximate factorization (3.5).

To preserve sparsity, for the row (column) oriented version (see Algorithm 1) fill-in on the factor U (V in the column version) is reduced by removing the elements whose absolute value is less than an absolute drop tolerance, typically $tolU = 0.1$. However, for the factor V (U in the column version) a relative threshold was found to be a better strategy. In this case, an entry of V is zeroed if its absolute value is less than a drop tolerance $tolV$ multiplied by the maximum of the matrix $|A|$. The motivation for this strategy comes from the observation that scaling A produces the same scaling in V and leaves invariant the factor U. In the numerical experiments the same drop tolerance was used for both factors U and V, i.e., $tolU = tolV$. From now on, for the sake of clarity, we mainly refer to the row oriented version and we point out the main differences with the column oriented one.

4.1. Comparative study of the Sherman–Morrison preconditioners. In this subsection we study the performance of the Sherman–Morrison based preconditioners M_1 and M_2 (equations (3.10) and (3.11)) for different dropping tolerances and with different values of the parameter s (see (3.1)). To ensure $s > \rho(A)$, this

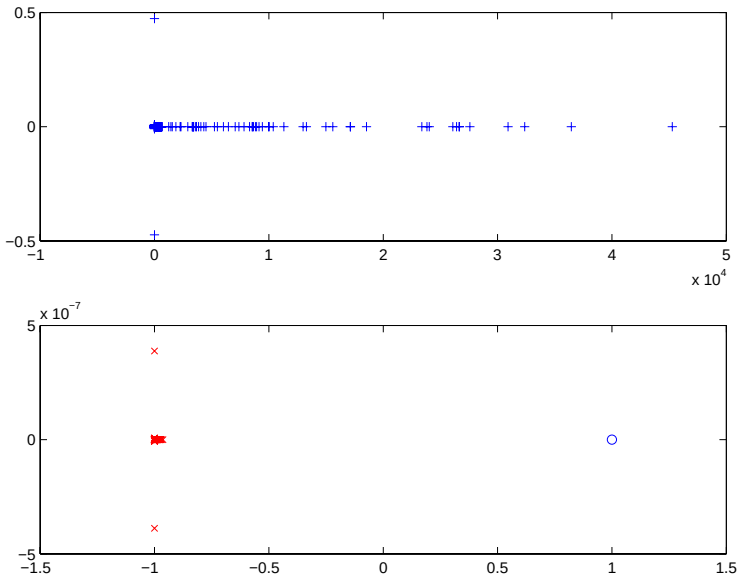

FIG. 4.1. *Spectrum of the matrix FS* 5414 *(top) and the preconditioned matrix* AM_1 *(bottom* $\circ$*) and* AM_2 *(bottom* $\times$*)) with* $s = 1.5\|A\|_\infty$ *and* $tol = 0$.

TABLE 4.2
Experimental results for the matrix FS 5414 *with the row oriented version.*

tol	$s/\|A\|_\infty$	$nnz(\bar{U})$	$nnz(\bar{V})$	nnz	iter. M_1	iter. M_2
0.1	1	547	708	1255	55	63
0.1	1.5	547	751	1298	51	53
0.1	5	547	871	1418	44	44
0.1	10	547	1047	1594	47	47
0.1	100	547	1589	2136	43	43
0.01	1	592	1092	1684	44	44
0.01	1.5	592	1188	1780	47	48
0.001	1	778	1794	2572	46	46
0.001	1.5	778	1852	2630	45	45
0.0001	1	1418	3425	4843	32	32
0.0001	1.5	1418	3584	5002	28	28
0.00001	1	2598	5532	8130	6	6
0.00001	1.5	2598	5707	8305	6	6

parameter is taken as a multiple greater than or equal to one of the ∞-norm of the matrix A. Thus, the factor V is nonsingular, and the spectrum of the preconditioned matrix is on the left-half complex plane when the preconditioner M_2 is used. This is illustrated in Figure 4.1 where the factorization (3.2) was computed with a drop tolerance equal to zero, i.e., without dropping any element, and $s = 1.5\|A\|_\infty$. Observe that, according to the equations (3.13), the spectrum of the preconditioned matrix AM_1 is clustered around 1, and the eigenvalues of the matrix AM_2 are on the left-half complex plane clustered approximately around -1. A similar situation is illustrated in Figure 4.2 for the matrix SAYLR3 but with a drop tolerance equal to 0.1.

Tables 4.2 and 4.4 show the results of the numerical experiments for the matrices FS5414 and SAYLR3 with Algorithm 1, which are representative of the set of the

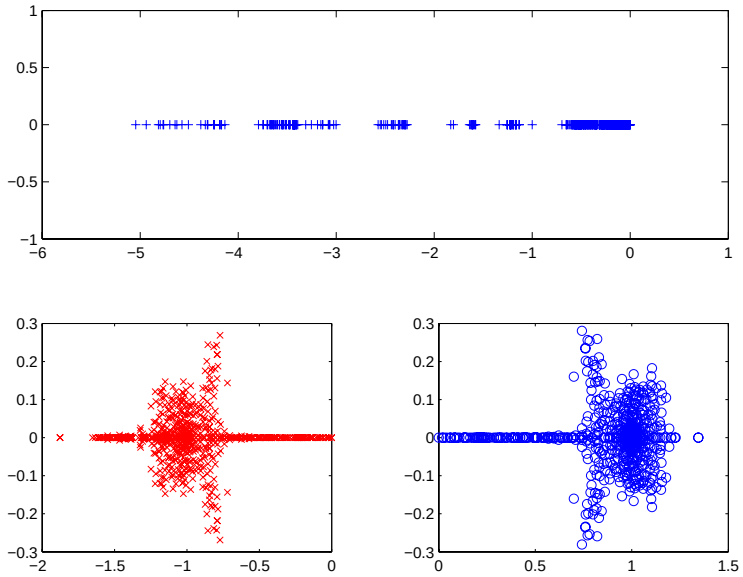

FIG. 4.2. *Spectrum of the matrix SAYLR3 (top) and the preconditioned matrix AM_1 (bottom right) and AM_2 (bottom left) with $s = 1.5\|A\|_\infty$ and tol $= 0.1$.*

TABLE 4.3
Experimental results for the matrix FS 5414 with the column oriented version.

tol	$s/\|A\|_\infty$	$nnz(\bar{U})$	$nnz(\bar{V})$	nnz	iter. M_1	iter. M_2
0.1	1	1914	541	2455	62	62
0.1	1.5	1967	541	2508	61	62
0.1	5	2427	541	2968	58	58
0.1	10	2901	541	3442	58	58
0.1	100	4013	541	4554	47	47
0.01	1	2926	560	3486	59	59
0.01	1.5	3132	560	3692	51	51
0.001	1	4197	631	4828	31	31
0.001	1.5	4355	631	4986	30	30
0.0001	1	6365	725	7090	13	13
0.0001	1.5	6657	725	7382	13	12
0.00001	1	7932	756	8688	6	6
0.00001	1.5	8167	756	8923	5	5

tested problems. The results for the column oriented algorithm are listed in Tables 4.3 and 4.5. Different values of the drop tolerance *tol* and the parameter s relative to the ∞-norm of A were employed. The number of nonzero elements of the incomplete factors is denoted by $nnz(\bar{V})$ and $nnz(\bar{U})$, and $nnz = nnz(\bar{U}) + nnz(\bar{V})$ is the total number of nonzero elements of the preconditioner. The number of iterations of preconditioning performed with M_i is denoted by iter. M_i.

None of the tested matrices is an M-matrix, and therefore breakdowns may occur. However, in the reported experiments the preconditioners were computed without breakdowns except for the matrix SAYLR3 (Table 4.4) with values of $s/\|A\|_\infty \geq 10$. Note that from (3.8) a large value of s produces small pivots which can degenerate in breakdown problems, as occurs in this case. In our code if the absolute value of

TABLE 4.4
Experimental results for the matrix SAYLR3 with the row oriented version.

tol	$s/\|A\|_\infty$	$nnz(\bar{U})$	$nnz(\bar{V})$	nnz	iter. M_1	iter. M_2
0.1	1	1314	4750	6064	43	44
0.1	1.5	1314	5606	6920	41	41
0.1	5	1314	8840	10154	35	35
0.1	10	1314	11041	12355	34	34
0.1	100	1314	21408	22722	33	33

TABLE 4.5
Experimental results for the matrix SAYLR3 with the column oriented version.

tol	$s/\|A\|_\infty$	$nnz(\bar{U})$	$nnz(\bar{V})$	nnz	iter. M_1	iter. M_2
0.1	1	1314	4750	6064	40	40
0.1	1.5	1314	5606	6920	37	37
0.1	5	1314	8840	10154	34	34
0.1	10	1314	11041	12355	33	33
0.1	100	1314	21408	22722	31	31

a pivot is less than the machine precision, ϵ_M, it is replaced by $\sqrt{\epsilon_M}$. Further study on this breakdown cure deserves to be investigated but it is out of the scope of this paper.

The number of nonzeros of the factor $\bar{U}$ for the row oriented version remains constant for a fixed drop tolerance when s changes. In contrast, the number of nonzero elements of $\bar{V}$ increases with s and, in general, the convergence is attained in fewer iterations. This can be explained from Theorem 3.1. It can be shown that (3.6) and (3.7) also hold for the incomplete factors. In fact, the matrix $\bar{U}$ does not depend on the value of s, and the elements on the upper triangular part of $\bar{V}$ are modified by a linear combination of vectors depending on s. In particular, for M-matrices it can be seen that the absolute value of these elements increases, augmenting the fill-in on $\bar{V}$. Therefore, recomputing the preconditioner for a different value of s can be done with some savings because only the upper triangular part of $\bar{V}$ must be updated. Regarding Table 4.3 the situation is reversed as it is expected from Remark 3.5.

It is also observed that the preconditioner M_2 converges in a number of iterations similar to M_1. However, since M_1 requires n multiplications per iteration more than M_2, the last one may perform slightly better in time. Thus, M_2 is recommended to be used as the default. Moreover, both row and column versions performed similarly.

Finally, concerning the choice of the parameter s, it was observed in all the cases that the balance between the number of iterations and fill-in of the factors was satisfactory taking $s/\|A\|_\infty = 1.5$. Moreover, with this choice of s the risk of appearing small pivots decrease, as was argued above. Therefore, only this value will be considered in the next subsection.

4.2. Comparative study with AINV. In this subsection we compare the row and column versions of the preconditioner M_2, which will be called AISM, with the AINV preconditioner and without preconditioner. The iterative solver employed was BiCGSTAB. For AINV, as recommended by the authors in [9], a drop tolerance of 0.1 was always used. For AISM, a fixed drop tolerance of 0.1 was used as well, except for the matrices ORSIRR* and ADD20. For these matrices the value 0.1 produced very sparse factors with poor convergence results, and therefore the value 0.01 was

TABLE 4.6
Comparison between the AISM with $s/\|A\|_\infty = 1.5$ and AINV preconditioners.

Matrix	No Prec.		AISM (row/column)				AINV(0.1)		
	Iter.	Time	*tol*	*nnz*	Iter.	Time	*nnz*	Iter.	Time
ADD20	297	0.64	0.01	14880/14863	7/8	0.06/0.06	9990	7	0.05
FS5414	783	0.34	0.1	1298/2508	53/62	0.03/0.04	4104	87	0.06
HOR131	†		0.1	4593/8642	40/38	0.03/0.03	8129	27	0.025
ORSIRR1	903	0.64	0.01	11637/11579	24/27	0.05/0.05	6300	26	0.04
ORSIRR2	749	0.46	0.01	10121/10140	26/27	0.05/0.06	5488	26	0.04
ORSREG1	273	0.56	0.1	17099/16583	33/32	0.16/0.13	13728	34	0.16
SAYLR3	366	0.23	0.1	6920/6920	41/37	0.06/0.05	6690	32	0.05
SAYLR4	†		0.1	71106/71106	38/48	0.55/0.6	51926	38	0.49
SHERMAN1	364	0.22	0.1	6848/6848	39/39	0.045/0.05	6690	32	0.04
WATT1	14	0.04	0.1	8829/8894	3/3	0.02/0.02	14807	2	0.03
WATT2	26	0.06	0.1	12075/14460	5/50	0.03/0.19	15488	10	0.05

found to be quite better. The results are reported in Table 4.6 and correspond to $s/\|A\|_\infty = 1.5$. The symbol † indicates no convergence of the iterative method.

The time, measured with the function `dtime()` and given in seconds, corresponds to the iterative solution phase only and does not include the time for computing the preconditioner. For AINV, the time for computing the preconditioner is usually very small (smaller than the time for the iterative phase), and the same is expected to be true for AISM since both algorithms have similar recurrence formulas.

It can be observed that the performance of the row oriented version of AISM is similar and comparable to AINV for most of the tested problems. There are two cases (matrices FS5414 and WATT2) where AISM is better than AINV, and others (like HOR131, SAYLR3, and SHERMAN1) where the situation is reversed. Concerning the column oriented version the performance observed was similar for most of the problems with slight differences, except for the matrix WATT2 for which the results where rather poor.

It is worth mentioning that AINV appears to be easier to tune since for a fixed value of the drop tolerance equal to 0.1 we always observed good performance. We conclude that AISM was as robust as AINV for the tested problems since it solved problems that BiCGSTAB without preconditioning was unable to solve.

5. Conclusions and future work. In this paper we have presented a new technique for computing factorized sparse approximate inverse preconditioners for nonsymmetric linear systems. It is based on the Sherman–Morrison formula to update the inverse of a matrix. Using this formula we compute incompletely two sets of vectors which are the columns of the sparse factors $\bar{U}$ and $\bar{V}$ of the preconditioner, such that $A_0^{-1} - A_0^{-1}\bar{U}\bar{\Omega}\bar{V}A_0^{-1} \approx A^{-1}$, where A_0^{-1} is either known or easy to compute. For the particular choice $A_0 = sI_n$, where s is a positive scalar, it has been shown that the preconditioner is guaranteed to exist if A is an M-matrix. The spectrum of the preconditioned matrix tends to be shifted to the left-half complex plane, which is a good situation for Krylov-type solvers.

Although for the computation of the preconditioner, referred to as AISM, we need to assess two parameters (s and the drop tolerance) from the numerical experiments, we recommend the choice $s = 1.5\|A\|_\infty$ as well as a value of 0.1 or 0.01 for the drop tolerance.

We have tested row and column oriented versions of AISM, and no significant differences between them were observed for most of the problems, except for matrix WATT2 for which the column version peformed worst. Therefore, the row oriented version of AISM could be recommended.

In addition, AISM has been compared with the well-known preconditioner AINV showing that both preconditioners are comparable for the tested problems. Like AINV, AISM works well even for matrices that are not M-matrices, and breakdowns are rare in practice. In the general case, nonsymmetric permutations like those used in [4] can be used to further improve robustness. Moreover, since the expressions used to compute AISM resemble those used by AINV, their computation complexity is similar. The relation between the two preconditioners is still under study.

Concerning future work, a full parallel version of the preconditioner is to be developed. We think that the use of graph partitioning strategies can help, as done in [6] for the AINV preconditioner, in addition to different choices of the matrix A_0 and the initial sets of vectors, which can lead to strategies with higher inherent parallelism. Moreover, the study of the symmetric case remains open. Concerning the last two questions, some interesting ideas found in [19, 25] may be exploited.

Another situation to be studied corresponds to the case of performing rank-k updates in the computation of the preconditioner, which can be of interested for matrices with a natural block structure, such as those arising in solid and structural mechanics problems.

Finally, it is worth noting that the proposed technique can be the basis for examining new strategies to update or improve an existing preconditioner, which may be of interest when the system matrix is modified as found in some networks or structures applications.

Acknowledgments. The authors wish to express their thanks to Michele Benzi for providing us his code for computing the AINV preconditioner and the referees for their helpful suggestions which greatly improved the paper.

REFERENCES

[1] M. S. Barlett, *An inverse matrix adjustment arising in discriminant analysis*, Ann. Math. Statist., 22 (1951), pp. 107–111.

[2] S. T. Barnard, L. M. Bernardo, and H. D. Simon, *An MPI implementation of the SPAI preconditioner on the T 3E*, Internat. J. High Perform. Comput. Appl., 13 (1999), pp. 107–123.

[3] M. Benzi, J. K. Cullum, and M. Tůma, *Robust approximate inverse preconditioning for the conjugate gradient method*, SIAM J. Sci. Comput., 22 (2000), pp. 1318–1332.

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

[5] 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 (2001), pp. 6533–6554.

[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, IMACS Ser. Comput. Appl. Math., D. R. Kincaid and A. C. Elster, eds., 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, *A comparative study of sparse approximate inverse preconditioners*, Appl. Numer. Math., 30 (1999), pp. 305–340.

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

[11] R. Bru, J. Mas, and A. Urbano, *Analysis of the reinforcement method for inverting a matrix*, in Proceedings of the International Conference on Numerical Algorithms, 2001, p. 58.

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

[13] M. T. Chu, R. E. Funderlic, and G. H. Golub, *A rank-one reduction formula and its applications to matrix factorizations*, SIAM Rev., 37 (1995), pp. 512–530.
[14] T. Davis, *University of Florida sparse matrix collection*, NA Digest, 92 (1994), http://www.cise.ufl.edu/research/sparse/matrices (April 2002).
[15] I. S. Duff, R. G. Grimes, and J. G. Lewis, *User's Guide for the Harwell-Boeing Sparse Matrix Collection*, Tech. report RAL-92-886, Rutherford Appleton Laboratory, Chilton, England, 1992.
[16] D. K. Fadeev and V. N. Faddeeva, *Computational Methods of Linear Algebra*, W. H. Freeman, San Francisco, CA, 1963.
[17] M. J. Grote and T. Huckle, *Parallel preconditioning with sparse approximate inverses*, SIAM J. Sci. Comput., 18 (1997), pp. 838–853.
[18] W. W. Hager, *Updating the inverse of a matrix*, SIAM Rev., 31 (1989), pp. 221–239.
[19] A. S. Householder, *Theory of Matrices in Numerical Analysis*, Blaisdell Publishing, Johnson, CO, 1964.
[20] 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.
[21] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, Boston, 1996.
[22] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.
[23] J. Sherman and W. Morrison, *Adjustment of an inverse matrix, corresponding to a change in one element of a given matrix*, Ann. Math. Statist., 21 (1950), pp. 124–127.
[24] H. A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 631–644.
[25] H. S. Wilf, *Matrix inversion by the annihilation of rank*, J. Soc. Indust. Appl. Math., 7 (1959), pp. 149–151.