DOI: 10.2478/auom-2024-0006
An. Şt. Univ. Ovidius Constanţa **Vol. 32(1),2024, 105–126**

An inverse LU preconditioner based on the Sherman–Morrison formula *

R. Bru, J. Cerdán, J. Marín and J. Mas

Abstract

An approximate inverse LU preconditioner is constructed based on the Sherman–Morrison formula. Applying recursively that inversion formula a multiplicative decomposition of the inverse of a matrix is obtained. This recursion in compact form is the base to build the proposed preconditioner that we call V–AISM. For nonsingular M-matrices and H-matrices of the invertible class the stability of the preconditioner is proved. Numerical results show that V–AISM is robust and competitive compared with other preconditioners.

1 Introduction

Let $Ax = b$ be a large, sparse nonsymmetric linear system where $A \in \mathbb{R}^{n\times n}$ is nonsingular and $x,\ b \in \mathbb{R}^n$. Developing preconditioners for solving linear systems by iterative methods is an important problem in Numerical Linear Algebra. The right preconditioning technique consists of finding a matrix M for which the solution via an iterative method of the equivalent linear system $AM^{-1}y = b$, $y = Mx$ is obtained more efficiently. The preconditioner M should approximate the matrix A in some sense. There are mainly two preconditioning techniques. One that computes the matrix M and another that

Key Words: Nonsymmetric linear systems, iterative methods, aproximate inverse LU preconditioning, Sherman-Morrison formula.
2010 Mathematics Subject Classification: Primary 65F08, 65F10; Secondary 65F50.
Received: 01.12.2022
Accepted: 20.04.2023
*The work was supported by Conselleria de Innovación, Universidades, Ciencia y Sociedad Digital, Generalitat Valenciana (CIAICO/2021/162)

computes its inverse. In this work we study factorized approximate inverse preconditioners that compute explicitly an approximation of A^{-1}. Then the preconditioners are applied by matrix-vector products in each iteration of the Krylov method that is important for efficient parallel computations. Among this class of preconditioners we can mention the AINV (Approximate Inverse) preconditioner [1] and some variants of it, see [4, 15]. A comparative study of this kind of preconditioners can be seen in [2].

The idea of building preconditioners by the Sherman-Morrison formula [14, 16] gives rise to preconditioners of both classes mentioned above, the AISM (Approximate Inverse Sherman-Morrison) [6, 10] that computes an approximate inverse and the BIF (Balanced Incomplete Factorization) [7, 8] that computes an incomplete LU factorization.

In this work, we use the Sherman–Morrison formula to obtain an approximate inverse LU preconditioner. The main difference with respect to the AISM is the way of applying recursively the inversion formula to obtain a new decomposition of A^{-1}. Then we use a compact representation of this decomposition to build our proposed preconditioner denoted by V–AISM.

The structure of the paper is the following. In section 2 we present a new decomposition of A^{-1}. In section 3, we show that the inverse of the LU factors are in this decomposition. In addition, we also prove that the computation of our algorithm is breakdown-free for nonsingular M–matrices and H–matrices of invertible class. In section 4 some numerical results of a set of matrices from Harwell–Boeing [12] and SuiteSparse Matrix [11] collections are given. Then the results are compared with those of the approximate inverse preconditioners AISM, AINV and with the ICI (Incomplete Cholesky Inverse) preconditioner [17] adapted for nonsymmetric matrices. Section 5 gives the main conclusions of our work.

2 A new decomposition of A^{-1}

Consider two nonsingular $n \times n$ matrices A and A_0, and two sets of vectors $\{x_k\}_{k=1}^n$ and $\{y_k\}_{k=1}^n$ such that

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

where $X = [x_1\ x_2\ \cdots\ x_n]$ and $Y = [y_1\ y_2\ \cdots\ y_n]$.

Defining $A_k = A_0 + \sum_{i=1}^{k} x_i y_i^T$ with $k = 1, \ldots, n$ we have

$$\begin{cases} A_k = A_{k-1} + x_k y_k^T \\ A_n = A. \end{cases}$$

Suppose that x_1 and y_1 are vectors such that $r_1 = 1 + y_1^T A_0^{-1} x_1 \neq 0$. By the Sherman–Morrison formula [14, Eq. (2)] and [16], the matrix $A_1 = A_0 + x_1 y_1^T$ is nonsingular and its inverse is given by

$$\begin{aligned} A_1^{-1} &= A_0^{-1} - \frac{1}{r_1} A_0^{-1} x_1 y_1^T A_0^{-1} \\ &= A_0^{-1} \left(I - \frac{1}{r_1} x_1 y_1^T A_0^{-1} \right) \\ &= A_0^{-1} \left(I - \frac{1}{r_1} x_1 w_1^T \right), \end{aligned}$$

where $w_1^T = y_1^T A_0^{-1}$.

Let $V_1 = I - \frac{1}{r_1} x_1 w_1^T$, then

$$A_1^{-1} = A_0^{-1} V_1.$$

Following this process, assuming that $r_k = 1 + y_k^T A_{k-1}^{-1} x_k \neq 0$ for x_k and y_k, then

$$\begin{aligned} A_k^{-1} &= A_{k-1}^{-1} - \frac{1}{r_k} A_{k-1}^{-1} x_k y_k^T A_{k-1}^{-1} \\ &= A_{k-1}^{-1} \left(I - \frac{1}{r_k} x_k y_k^T A_{k-1}^{-1} \right) \\ &= A_{k-1}^{-1} \left(I - \frac{1}{r_k} x_k w_k^T \right) \\ &= A_{k-1}^{-1} V_k, \end{aligned}$$

where $w_k^T = y_k^T A_{k-1}^{-1}$ and $V_k = I - \frac{1}{r_k} x_k w_k^T$.

The matrix A_k^{-1} can be written in a factorized way in terms of the matrices V_k as

$$\begin{aligned} A_k^{-1} &= A_{k-1}^{-1} V_k \\ &= A_{k-2}^{-1} V_{k-1} V_k \\ &= A_0^{-1} V_1 \cdots V_k. \end{aligned}$$

Then we obtain the following factorization of A^{-1}

$$A^{-1} = A_n^{-1} = A_0^{-1} V_1 \cdots V_n. \tag{2}$$

The coefficients r_k will be called pivots of the Sherman–Morrison formula when it is applied recursively. In section 3, they will be used and related with the pivots of the LU factorization of A.

It is worth to say that there is a main difference between the expressions of A^{-1}, the one in (2) and that obtained in [6]. Actually, to build the preconditioner AISM given in [6] the expression of A^{-1} is an additive decomposition obtained applying also the Sherman–Morrison formula. Here to construct the new preconditioner V–AISM we have a multiplicative representation of A. In fact, This decomposition depends explicitly on the matrices V_k.

Focusing on the construction of these matrices, we can simplify the above inverse decomposition with the following compact representation.

Theorem 1. *Consider two nonsingular $n \times n$ matrices A, A_0 and two sets of vectors $\{x_k\}_{k=1}^n$, $\{y_k\}_{k=1}^n$ satisfying (1) and $r_k = 1 + y_k^T A_{k-1}^{-1} x_k \neq 0$. Then*

(i)

$$V_1 \cdots V_k = I - X_k R_k W_k^T, \qquad k = 1, \ldots, n, \tag{3}$$

where $X_k = [x_1 \ x_2 \ \ldots \ x_k]$, $W_k = [w_1 \ w_2 \ \ldots \ w_k]$, $w_i^T = y_i^T A_{i-1}^{-1}$ and R_k is the $k \times k$ upper triangular nonsingular matrix

$$R_k = \begin{bmatrix} R_{k-1} & -\dfrac{1}{r_k} R_{k-1} W_{k-1}^T x_k \\ 0 & \dfrac{1}{r_k} \end{bmatrix}, \tag{4}$$

with $R_1 = \left[r_1^{-1} \right]$.

(ii) The compact representation of A^{-1} is

$$A^{-1} = A_0^{-1}(I - X_n R_n W_n^T). \tag{5}$$

Proof. (i) The proof is done by induction over k and similar to the one used in [9, Lemma 2.1].

Initially we have $V_1 = I - x_1 \dfrac{1}{r_1} w_1^T = I - X_1 R_1 W_1^T$. Then,

$$\begin{aligned} V_1 V_2 &= \left(I - \frac{1}{r_1} x_1 w_1^T \right) \left(I - \frac{1}{r_2} x_2 w_2^T \right) \\ &= I - \frac{1}{r_1} x_1 w_1^T - \frac{1}{r_2} x_2 w_2^T + \frac{1}{r_1 r_2} x_1 w_1^T x_2 w_2^T \\ &= I - \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} \dfrac{1}{r_1} w_1^T - \dfrac{1}{r_1 r_2} w_1^T x_1 w_2^T \\ \dfrac{1}{r_2} w_2^T \end{bmatrix} \\ &= I - \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} \dfrac{1}{r_1} & -\dfrac{1}{r_1 r_2} w_1^T x_1 \\ 0 & \dfrac{1}{r_2} \end{bmatrix} \begin{bmatrix} w_1^T \\ w_2^T \end{bmatrix} = I - X_2 R_2 W_2^T, \end{aligned}$$

where $R_2 = \begin{bmatrix} R_1 & -\dfrac{1}{r_2} R_1 W_1^T x_2 \\ 0 & \dfrac{1}{r_2} \end{bmatrix}$ according to (4).

Now, assume that the relations (3) and (4) hold for $k-1$, that is,

$$V_1 \cdots V_{k-1} = I - X_{k-1} R_{k-1} W_{k-1}^T.$$

Let us see that they are also valid for k. We have

$$\begin{aligned}
V_1 \cdots V_k &= (V_1 \cdots V_{k-1}) V_k \\
&= (I - X_{k-1} R_{k-1} W_{k-1}^T)(I - \frac{1}{r_k} x_k w_k^T) \\
&= I - X_{k-1} R_{k-1} W_{k-1}^T - \frac{1}{r_k} x_k w_k^T + \frac{1}{r_k} X_{k-1} R_{k-1} W_{k-1}^T x_k w_k^T \\
&= I - \begin{bmatrix} X_{k-1} & x_k \end{bmatrix} \begin{bmatrix} R_{k-1} W_{k-1}^T - \dfrac{1}{r_k} R_{k-1} W_{k-1}^T x_k w_k^T \\ \dfrac{1}{r_k} w_k^T \end{bmatrix} \\
&= I - \begin{bmatrix} X_{k-1} & x_k \end{bmatrix} \begin{bmatrix} R_{k-1} & -\dfrac{1}{r_k} R_{k-1} W_{k-1}^T x_k \\ 0 & \dfrac{1}{r_k} \end{bmatrix} \begin{bmatrix} W_{k-1}^T \\ w_k^T \end{bmatrix} \\
&= I - X_k R_k W_k^T,
\end{aligned}$$

where

$$R_k = \begin{bmatrix} R_{k-1} & -\dfrac{1}{r_k} R_{k-1} W_{k-1}^T x_k \\ 0 & \dfrac{1}{r_k} \end{bmatrix}.$$

(ii) Applying (3) to the factorization (2) we have

$$A^{-1} = A_0^{-1}(I - X_n R_n W_n^T).$$ □

3 An approximate inverse LU preconditioner

Now we obtain an approximate inverse preconditioner using the decomposition of A^{-1} given in Theorem 1. In what follows, we choose $x_k = a_k - e_k$, $y_k = e_k$

and $A_0 = I$, where a_k and e_k represent the kth column of the matrix A and the identity matrix I, respectively. Then, from (5) we have

$$A^{-1} = I - (A - I)R_n W_n^T. \tag{6}$$

We will prove that the matrices R_n and W_n^T are related with the inverse factors of the LU decomposition of A. To give the result it is worth to analyze the structure of matrices $W_k^T = [w_1\ w_2\ \dots\ w_k]^T$, $k = 1, \dots, n$. In particular, we will show that the leading principal $k \times k$ submatrix of W_k^T is unit lower triangular. We have $w_1^T = y_1^T A_0^{-1} = e_1^T$ and for $k \geq 1$

$$\begin{aligned} w_{k+1}^T &= y_{k+1}^T A_k^{-1} = y_{k+1}^T A_0^{-1} V_1 \cdots V_k \\ &= y_{k+1}^T V_1 \cdots V_k \\ &= e_{k+1}^T (I - X_k R_k W_k^T) \\ &= e_{k+1}^T - e_{k+1}^T X_k R_k W_k^T \\ &= e_{k+1}^T - e_{k+1}^T \left[\begin{array}{cccc} a_1 - e_1 & a_2 - e_2 & \cdots & a_k - e_k \end{array} \right] R_k W_k^T \\ &= e_{k+1}^T - \left[\begin{array}{cccc} a_{k+1,1} & a_{k+1,2} & \cdots & a_{k+1,k} \end{array} \right] R_k W_k^T. \end{aligned} \tag{7}$$

Then,

$$\begin{array}{rcl} w_1^T &=& e_1^T = \left[\begin{array}{cccccc} 1 & 0 & 0 & 0 & \cdots & 0 \end{array} \right] \\ w_2^T &=& e_2^T - \left[\begin{array}{c} a_{2,1} \end{array} \right] R_1 W_1^T = \left[\begin{array}{cccccc} * & 1 & 0 & 0 & \cdots & 0 \end{array} \right] \\ w_3^T &=& e_3^T - \left[\begin{array}{cc} a_{3,1} & a_{3,2} \end{array} \right] R_2 W_2^T = \left[\begin{array}{cccccc} * & * & 1 & 0 & \cdots & 0 \end{array} \right] \end{array}$$

that is,

$$W_3^T = \left[\begin{array}{c} w_1^T \\ w_2^T \\ w_3^T \end{array} \right] = \left[\begin{array}{cccccc} 1 & 0 & 0 & 0 & \cdots & 0 \\ * & 1 & 0 & 0 & \cdots & 0 \\ * & * & 1 & 0 & \cdots & 0 \end{array} \right] = [T_3 \ \ O],$$

where the block T_3 is a unit lower matrix of size 3, the block O is a null matrix with appropriate size and $*$ denotes an unspecified number.

Then $W_k^T = [T_k \ \ O]$, where T_k is a unit lower matrix of size k. Moreover

$$\begin{aligned} W_{k+1}^T = \left[\begin{array}{cc} T_k & O \\ \multicolumn{2}{c}{w_{k+1}^T} \end{array} \right] &= \left[\begin{array}{ccc} T_k & 0 & O \\ - \left[\begin{array}{cccc} a_{k+1,1} & a_{k+1,2} & \cdots & a_{k+1,k} \end{array} \right] R_k T_k & 1 & O \end{array} \right] \\ &= [T_{k+1} \ \ O] \end{aligned} \tag{8}$$

and $W_n^T = T_n$.

Theorem 2. *Let $x_k = a_k - e_k$, $y_k = e_k$ and $A_0 = I$, where a_k and e_k represent the kth column of A and I, respectively. Assuming the conditions of Theorem 1, A has LU factorization, $W_n^T = L^{-1}$ and $R_n = U^{-1}$.*

Proof. We are going to show that $A = W_n^{-T} R_n^{-1}$. For that, we will prove by induction over k that $T_k^{-1} R_k^{-1} = A_{kk}$ where A_{kk} represents the leading principal submatrix of A of size k, with $k = 1, \dots, n$.

For $k = 1$, it is trivially satisfied that $T_1^{-1} R_1^{-1} = [1][r_1] = [a_{11}] = A_{11}$ since $r_1 = 1 + w_1^T x_1$.

Assuming that $T_k^{-1} R_k^{-1} = A_{kk}$, let us see that the equality holds for $k+1$. That is, T_k^{-1} and R_k^{-1} are the LU factors of the leading principal submatrix A_{kk}. From (8) we have

$$T_{k+1} = \begin{bmatrix} T_k & 0 \\ -\begin{bmatrix} a_{k+1,1} & a_{k+1,2} & \dots & a_{k+1,k} \end{bmatrix} R_k T_k & 1 \end{bmatrix}$$

and then its inverse is

$$T_{k+1}^{-1} = \begin{bmatrix} T_k^{-1} & 0 \\ \begin{bmatrix} a_{k+1,1} & a_{k+1,2} & \dots & a_{k+1,k} \end{bmatrix} R_k & 1 \end{bmatrix}. \tag{9}$$

Then, from (4) and (9) it follows

$$T_{k+1}^{-1} R_{k+1}^{-1} = \begin{bmatrix} T_k^{-1} & 0 \\ \begin{bmatrix} a_{k+1,1} & a_{k+1,2} & \dots & a_{k+1,k} \end{bmatrix} R_k & 1 \end{bmatrix} \begin{bmatrix} R_k^{-1} & W_k^T x_{k+1} \\ 0 & r_{k+1} \end{bmatrix}$$

$$= \begin{bmatrix} A_{kk} & \begin{bmatrix} a_{1,k+1} \\ a_{2,k+1} \\ \vdots \\ a_{k,k+1} \end{bmatrix} \\ \begin{bmatrix} a_{k+1,1} & a_{k+1,2} & \dots & a_{k+1,k} \end{bmatrix} & a_{k+1,k+1} \end{bmatrix} = A_{k+1,k+1}.$$

Then, for $k = n$, $T_n^{-1} R_n^{-1} = W_n^{-T} R_n^{-1} = A$. That is, A is factorized as a product of a unit lower triangular matrix W_n^{-T} and an upper triangular matrix R_n. Since the LU factorization of a nonsingular matrix is unique, then $W_n^T = L^{-1}$ and $R_n = U^{-1}$. □

Note that under the conditions of the above theorem the pivots of the Sherman-Morrison formula r_k are equal to those of the *LU* factorization. Then we give an algorithm to compute the approximate factors denoted by $\bar{R}$ and $\bar{W}^T$. The columns of these matrices are obtained according to (4) and (7). In the algorithm a MATLAB notation is used to indicate the indices of vectors. We recall that even a sparse matrix can have a dense inverse and therefore the number of nonzero elements of these factors (fill-in) grows up during its computation in exact arithmetic. Therefore, selected new entries must be nullified in order to keep the preconditioner sparse. This is fundamental for an efficient preconditioning of the iterative method.

Algorithm 1 V–AISM algorithm

Inputs: A, $X = A - I$, $Y = I$, $W^T = I$,

(1.1) $\bar{w}_1^T = Y_{[1,:]}$, $\bar{W}_{[1,:]}^T = \bar{w}_1^T$ and $\bar{R}_{[1,1]} = a_{11}^{-1}$

For $k = 2, \ldots, n$

(2.1) $\bar{w}_k^T = Y_{[k,:]} - A_{[k,1:k-1]} \bar{R}_{[1:k-1,1:k-1]} \bar{W}_{[1:k-1,:]}^T$

(2.2) $\bar{W}_{[k,:]}^T = \bar{w}_k^T$

(2.3) $\bar{r}_k = 1 + \bar{w}_k^T X_{[:,k]}$

(2.4) $\bar{c}_k = -\dfrac{1}{\bar{r}_k} [\bar{R}_{[1:k-1,1:k-1]} (\bar{W}_{[1:k-1,:]}^T X_{[:,k]})]$

(2.5) $\bar{R}_{[1:k-1,k]} = \bar{c}_k$

(2.6) $\bar{R}_{[k,k]} = \dfrac{1}{\bar{r}_k}$

EndFor

Outputs: Approximate factors $\bar{R} \approx U^{-1}$ and $\bar{W}^T \approx L^{-1}$

The computation of the new row $\bar{w}_k^T$ of $\bar{W}_k^T$ is done in the step (2.1) of the algorithm. Then $\bar{r}_k$ can be computed in step (2.3). In addition, the step (2.4) gives the off–diagonal elements of the kth column of $\bar{R}_k$. Finally, the matrix $\bar{R}_k$ is completed in steps (2.5) and (2.6). We note that the inexact factors are obtained by applying a dropping strategy after steps (2.1) and (2.4) to off–diagonal elements of these matrices.

The algorithm gives us the new preconditioner V–AISM which can be considered as a variant of the AISM preconditioner [6]. The approximate preconditioner is based on the two approximate factors $\bar{R} \approx U^{-1}$ and $\bar{W}^T \approx L^{-1}$. Therefore, the preconditioning step is done by performing two matrix-by-vector products as

$$\bar{R}(\bar{W}^T x) \ .$$

The algorithm runs to the end if all the pivots $\bar{r}_k$, $k = 1, 2, \ldots, n$, are nonzero. The following theorems prove that this is the case when the matrix A is a nonsingular M–matrix or H–matrix of the invertible class.

Remark 1. *Given two matrices $M = [m_{ij}]$ and $N = [n_{ij}]$, we denote $M \geq N$ when $m_{ij} \geq n_{ij}$. Likewise, $|M| = [|m_{ij}|]$. A matrix M is a nonsingular M–matrix if $m_{ij} \leq 0$ for all $i \neq j$ and $M^{-1} \geq O$ [3, Cond. N_{38} of Th. (2.3)]. Recall that a nonsingular M–matrix has positive diagonal entries, moreover the pivots of the LU factorization without pivoting are positive [3, Cond. E_{18} of Th. (2.3)]. Indeed, the LU factors are also M–matrices [3, Ex. (5.16)].*

Theorem 3. *Let A be a nonsingular M–matrix. Then the matrix $\bar{R}$ computed by Algorithm 1 is nonsingular. Moreover, the pivots satisfy $\bar{r}_k > 0$, $k = 1, 2, \ldots, n$.*

Proof. To prove that the matrix $\bar{R}$ is nonsingular we will show by induction over k that

$$O \leq \bar{R}_k \leq R_k, \tag{10}$$

with the help of

$$O \leq \bar{W}_k^T \leq W_k^T. \tag{11}$$

For $k = 1$, we have $W_1^T = \left[w_1^T\right] = \left[e_1^T\right]$ and $R_1 = \left[r_1^{-1}\right]$ where $r_1 = 1 + w_1^T x_1 = a_{11} > 0$. Note that no dropping can be done for the first vector, and (10) and (11) trivially hold.

Now, assume that (10) and (11) hold for k. Let us see for $k+1$. First we prove (11).

Recall that the matrix W_{k+1}^T is built adding a row, the vector w_{k+1}^T, to the matrix W_k^T (see (8)). Since A is an M–matrix, $a_{ij} \leq 0$ for $i \neq j$, and $0 \leq \bar{R}_k \leq R_k$ holds for k by the induction hypothesis, then

$$\begin{aligned} 0 &\leq \bar{w}_{k+1}^T \\ &= e_{k+1}^T - \left[\begin{array}{cccc} a_{k+1,1} & a_{k+1,2} & \cdots & a_{k+1,k} \end{array}\right] \bar{R}_k \bar{W}_k^T \\ &\leq e_{k+1}^T - \left[\begin{array}{cccc} a_{k+1,1} & a_{k+1,2} & \cdots & a_{k+1,k} \end{array}\right] R_k W_k^T \\ &= w_{k+1}^T, \end{aligned}$$

now some off-diagonal entries of $\bar{w}_{k+1}^T$ are nullified and (11) holds for $k+1$. We use the same notation.

Now, let us prove (10). From (4)

$$R_{k+1} = \left[\begin{array}{cc} R_k & -\frac{1}{r_{k+1}} R_k W_k^T x_{k+1} \\ 0 & \frac{1}{r_{k+1}} \end{array}\right].$$

Then, it will be enough to check that it is fulfilled for the last column.

First, we have

$$
\begin{aligned}
\bar{r}_{k+1} &= 1+\bar{w}_{k+1}^T x_{k+1} = 1+\bar{w}_{k+1}^T(a_{k+1}-e_{k+1}) \\
&= 1+\left[\ \bar{w}_{k+1}^T[1:k] \ \middle|\ 1 \ \middle|\ \mathrm{O}\ \right]\begin{bmatrix} a_{k+1}[1:k] \\ a_{k+1,k+1}-1 \\ a_{k+1}[k+2:n] \end{bmatrix} \\
&= a_{k+1,k+1}+\bar{w}_{k+1}^T[1:k]\begin{bmatrix} a_{1,k+1} \\ a_{2,k+1} \\ \vdots \\ a_{k,k+1} \end{bmatrix} \\
&\geq a_{k+1,k+1}+w_{k+1}^T[1:k]\begin{bmatrix} a_{1,k+1} \\ a_{2,k+1} \\ \vdots \\ a_{k,k+1} \end{bmatrix} \\
&= r_{k+1} > 0,
\end{aligned}
$$

which is positive since it is the pivot of the gaussian elimination of a nonsingular M–matrix (see Remark 1). Here $w_{k+1}^T[1:k]$ and $a_{k+1}[1:k]$ denote the k first entries of the vector w_{k+1}^T and a_{k+1}, respectively. Moreover $a_{k+1}[k+2:n]$ denotes the $n-k+2$ last components of the vector a_{k+1}.

Second,

$$0 \geq \bar{W}_k^T x_{k+1} = \bar{W}_k^T(a_{k+1}-e_{k+1}) = \bar{W}_k^T a_{k+1} \geq W_k^T a_{k+1} = W_k^T x_{k+1},$$

since A is an M–matrix and (11) holds for k. Therefore,

$$0 \leq -\frac{1}{\bar{r}_{k+1}}\bar{R}_k\bar{W}_k^T x_{k+1} \leq -\frac{1}{r_{k+1}}R_k W_k^T x_{k+1}$$

and (10) holds for $k+1$. Thus, $\bar{R}=\bar{R}_n$ is nonsingular and the pivots are positive. □

In what follows we need to denote by $\bar{R}_k(C)$ and $\bar{W}_k^T(C)$ the two matrices obtained by the algorithm when is applied to a given matrix C. A similar notation is used for vectors or its components.

Remark 2. *The comparison matrix of M is $\mathcal{M}(M)=[\alpha_{ij}]$, where $\alpha_{ii}=|m_{ij}|$ and $\alpha_{ij}=-|m_{ij}|$ for $i\neq j$. The matrix M is an H–matrix of the invertible class if its comparison matrix $\mathcal{M}(M)$ is a nonsingular M–matrix (see [5, Table 1]).*

Theorem 4. *Let A be an H–matrix of the invertible class. Then the matrix $\bar{R}$ computed by Algorithm 1 is nonsingular.*

Proof. To prove that the matrix $\bar{R}$ is nonsingular we will show by induction over k that

$$0 \leq \left|\bar{R}_k(A)\right| \leq \bar{R}_k(\mathcal{M}(A)), \tag{12}$$

showing

$$0 \leq \left|\bar{W}_k^T(A)\right| \leq \bar{W}_k^T(\mathcal{M}(A)). \tag{13}$$

For $k = 1$, we have

$$|\bar{W}_1^T(A)| = |W_1^T(A)| = \left|\left[w_1^T(A)\right]\right| = \left[e_1^T\right] = W_1^T(\mathcal{M}(A)) = \bar{W}_1^T(\mathcal{M}(A))$$

and

$$\begin{aligned} |r_1(A)| &= \left|1 + w_1^T(A)x_1(A)\right| = |a_{11}| \\ &= 1 + w_1^T((\mathcal{M}(A))x_1(\mathcal{M}(A)) = r_1(\mathcal{M}(A)) > 0. \end{aligned}$$

Then $\left|\bar{R}_1(A)\right| = |R_1(A)| = |R_1(\mathcal{M}(A))| = \left|\bar{R}_1(\mathcal{M}(A))\right|$, because no dropping can be done, (12) and (13) trivially hold.

Now, assuming that (12) and (13) hold for k, let us prove them for $k+1$. By (8) the matrix $\bar{W}_{k+1}^T(A)$ is built adding the row vector $\bar{w}_{k+1}^T(A)$ to the matrix $\bar{W}_k^T(A)$. Then, we have only to prove that $\left|\bar{w}_{k+1}^T(A)\right| \leq \bar{w}_{k+1}^T(\mathcal{M}(A))$. That is,

$$\begin{aligned} \left|\bar{w}_{k+1}^T(A)\right| &= \left|e_{k+1}^T - \begin{bmatrix} a_{k+1,1} & a_{k+1,2} & \cdots & a_{k+1,k} \end{bmatrix} \bar{R}_k(A)\bar{W}_k^T(A)\right| \\ &\leq e_{k+1}^T + \left|\begin{bmatrix} a_{k+1,1} & a_{k+1,2} & \cdots & a_{k+1,k} \end{bmatrix} \bar{R}_k(A)\bar{W}_k^T(A)\right| \\ &\leq e_{k+1}^T + \left|\begin{bmatrix} a_{k+1,1} & a_{k+1,2} & \cdots & a_{k+1,k} \end{bmatrix}\right|\left|\bar{R}_k(A)\right|\left|\bar{W}_k^T(A)\right| \\ &\leq e_{k+1}^T + \left|\begin{bmatrix} a_{k+1,1} & a_{k+1,2} & \cdots & a_{k+1,k} \end{bmatrix}\right|\bar{R}_k(\mathcal{M}(A))\bar{W}_k^T(\mathcal{M}(A)) \\ &= e_{k+1}^T \\ &\quad - \left(-\left|\begin{bmatrix} a_{k+1,1} & a_{k+1,2} & \cdots & a_{k+1,k} \end{bmatrix}\right|\right) \bar{R}_k(\mathcal{M}(A))\bar{W}_k^T(\mathcal{M}(A)) \\ &= w_{k+1}^T(\mathcal{M}(A)), \end{aligned}$$

Now, let us prove the (12). From (4) applied to $\mathcal{M}(A)$, we have

$$R_{k+1}(\mathcal{M}(A)) = \begin{bmatrix} R_k(\mathcal{M}(A)) & \dfrac{-1}{r_{k+1}(\mathcal{M}(A))} R_k(\mathcal{M}(A)) W_k^T(\mathcal{M}(A)) x_{k+1}(\mathcal{M}(A)) \\ 0 & \dfrac{1}{r_{k+1}(\mathcal{M}(A))} \end{bmatrix}$$

It will be enough to prove for the last column.

First, we have

$$\begin{aligned}
|\bar{r}_{k+1}(A)| &= \left|a_{k+1,k+1} + \bar{w}_{k+1}^T(A)[1:k]\,A[1:k,k+1]\right| \\
&\geq |a_{k+1,k+1}| - \left|\bar{w}_{k+1}^T(A)[1:k]\right| \left|A[1:k,k+1]\right| \\
&\geq |a_{k+1,k+1}| - w_{k+1}^T(\mathcal{M}(A))[1:k]\,|A[1:k,k+1]| \\
&= |a_{k+1,k+1}| + w_{k+1}^T(\mathcal{M}(A))[1:k]\,(-|A[1:k,k+1]|) \\
&= |a_{k+1,k+1}| + w_{k+1}^T(\mathcal{M}(A))[1:k]\,\mathcal{M}(A)[1:k,k+1] \\
&= \bar{r}_{k+1}(\mathcal{M}(A)) > 0.
\end{aligned}$$

Thus $\left|\dfrac{1}{\bar{r}_{k+1}(A)}\right| \leq \dfrac{1}{\bar{r}_{k+1}(\mathcal{M}(A))}$.

Second,

$$\begin{aligned}
\left|-\frac{1}{\bar{r}_{k+1}(A)}\bar{R}_k(A)\bar{W}_k^T(A)x_{k+1}(A)\right| &\leq \left|-\frac{1}{\bar{r}_{k+1}(A)}\right| \left|\bar{R}_k(A)\right| \left|\bar{W}_k^T(A)x_{k+1}(A)\right| \\
&\leq \frac{1}{\bar{r}_{k+1}(\mathcal{M}(A))}\bar{R}_k(\mathcal{M}(A))\bar{W}_k^T(\mathcal{M}(A))\,|a_{k+1}(A)| \\
&= -\frac{1}{\bar{r}_{k+1}(\mathcal{M}(A))}\bar{R}_k(\mathcal{M}(A))\bar{W}_k^T(\mathcal{M}(A))\,(-|a_{k+1}(A)|) \\
&= -\frac{1}{\bar{r}_{k+1}(\mathcal{M}(A))}\bar{R}_k(\mathcal{M}(A))\bar{W}_k^T(\mathcal{M}(A))a_{k+1}(\mathcal{M}(A)) \\
&= -\frac{1}{\bar{r}_{k+1}(\mathcal{M}(A))}\bar{R}_k(\mathcal{M}(A))\bar{W}_k^T(\mathcal{M}(A))x_{k+1}(\mathcal{M}(A))
\end{aligned}$$

Then (12) is satisfied for $k+1$.

□

4 Numerical results

In this section we study the numerical performance of the proposed preconditioner V–AISM. We have compared V–AISM with the AISM [6], the AINV [1] and ICI [17] preconditioners. The comparison with AISM is a natural choice since V–AISM is related to it. The AINV preconditioner is an important and well known approximate inverse preconditioner, also used in [6] to assess the AISM performance. It computes approximate LU factors by a biconjugation process. The ICI preconditioner was used in [17] to obtain an approximate inverse of the Cholesky factor of SPD matrices. The algorithm first computes an approximate Cholesky factor and then obtain an approximation of its inverse. We modified the algorithm for nonsymmetric matrices. An ILU(0) was first computed and after that, approximations of the inverse of the incomplete LU factors were obtained applying Algorithm 2 of [17] with the same procedure used for SPD matrices.

The goal is to show that the new version V–AISM allows for the computation of robust preconditioners for solving sparse nonsymmetric linear systems. In addition, we think that the way its computation is formulated permits more efficient implementations than AISM. In fact, the algorithm shows that the main operations performed are two sparse matrix-vector products that opens the possibility for using sparse BLAS level 2 routines [13]. Efficient implementations of these routines are available for modern computer architectures.

All the experiments were done in MATLAB. The AINV, AISM and ICI preconditioners were coded as described in [1], [6] and [17], respectively. Moreover, for AINV the results with an optimized FORTRAN code kindly provided by Michele Benzi are also reported. The iterative method used is the BiCGSTAB with right inverse preconditioning using the MATLAB function `bicgstab()`. The BiCGSTAB method was stopped when the relative initial residual was reduced to 10^{-8} and allowing up to 2000 iterations. The number of iterations reported in the tables corresponds to the rounded value returned by the function mentioned above. The initial guess was the corresponding zero vector in all computations. To preserve the sparsity of the preconditioner small entries were dropped by value. More precisely, the new off-diagonal entries in vectors $\bar{w}_k$ and $\bar{c}_k$ (steps (2.1) and (2.4) in Algorithm 1) were dropped if their relative value with respect to the maximum value of $|A|$ was less than a given threshold. The same dropping strategy and drop tolerances were used for both factors $\bar{R}$ and $\bar{W}^T$. This threshold is the only parameter needed to build the preconditioner. In some cases better results can be obtained with different drop tolerances for these factors, but we avoided fine tuning in order to simplify the results. The value of the threshold was choosen so that the number of nonzero elements of the preconditioner was approximately the same for all the preconditioners tested.

Prior computations with V–AISM the matrices were rescaled in two different ways. The first one consists in dividing all the elements of a matrix by the absolute value of its largest entry. For the second one each column was rescaled with its maximum column entry in absolute value. In Tables 2 and 4 an asterisk symbol $*$ indicates the second situation. We found some differences in the results with these two different scaling strategies and therefore, we only show the best result obtained. For AINV the matrices were also rescaled in the first way as it was done in [1]. For AISM and ICI the matrices were not rescaled as it was not done in [6] and [17].

The matrices used for the test can be downloaded from the Harwell–Boeing [12] and SuiteSparse Matrix collections [11], see Table 1 where n and nnz represents the size and number of nonzero elements. We have selected most of the matrices used in [6] to compare the AISM and AINV preconditioners. In addition we include the result with larger matrices, CHEM_MASTER1, EPB3 and POISSON3Db.

Table 1: Size (n) and number of nonzero elements (nnz) of the test matrices.

Matrix	n	nnz	Description
ADD20	2395	17319	Circuit simulation
CHEM_MASTER1	40401	201201	Chemical reaction simulation
EPB3	84617	463625	Thermal Problem
FS_541_4	541	4285	Chemical kinetics
HOR_131	434	4710	Network flow
JPWH_991	991	6027	Circuit physics modeling
MEMPLUS	17758	99147	Circuit simulation
ORSIRR_1	1030	6858	Reservoir simulation
ORSIRR_2	886	5970	Reservoir simulation
ORSREG_1	2205	14133	Reservoir simulation
POISSON3Db	85623	2374949	Computational fluid Dynamics
PORES_2	1224	9613	Reservoir simulation
RAEFSKY1	3242	293409	Computational fluid dynamics
RAEFSKY5	6316	168658	Computational fluid dynamics
SHERMAN2	1080	23094	Computational fluid dynamics
WATT_1	1856	11360	Computational fluid dynamics
WATT_2	1856	11550	Computational fluid dynamics

In the tables, *tol* indicates the value for the drop tolerance for the different preconditioners used. The density of the preconditioner, ρ, is computed as the ratio between the number of the nonzero elements of the factors and the number of elements of the initial matrix. For instance, for V–AISM it is

$$\rho = \frac{\mathrm{nnz}(\bar{R}) + \mathrm{nnz}(\bar{W^T})}{\mathrm{nnz}(A)}.$$

The symbol † indicates that no convergence was attained by the iterative method. Moreover, the CPU times for computing the preconditioner (T_p) and solving the system (T_s) are reported. The symbol § means that the time for computing the preconditioner was larger than $2,000$ seconds. For AINV these times are detailed for both, the MATLAB and the FORTRAN codes. We think that comparing a basic MATLAB implementation of AINV with its FORTRAN version gives an idea of the performance improvement that can be achieved with a fully optimized code. In other way, comparing the times obtained with highly optimized FORTRAN implementations against MATLAB scripts could be unfair. We believe that, due to the recursion nature of all these algorithms, a FORTRAN implementation of V–AISM can be as efficient as the AINV one. In fact, in [7] the authors experiment with an ILU preconditioner derived from the AISM formulas that competes against RIF, a robust version of AINV for SPD matrices, and other ILU-type preconditioners implemented in FORTRAN.

Table 2: Comparison between V–AISM and AISM preconditioners.

Matrix	V–AISM					AISM				
	tol	ρ	T_p	T_s	Iter.	*tol*	ρ	T_p	T_s	Iter.
ADD20	0.1	0.7	0.37	0.001	8	0.01	1.1	20.2	0.003	7
FS_541_4*	0.01	1.1	0.04	0.001	5	0.0001	1.2	1.0	0.002	23
HOR_131*	0.1	2.3	0.04	0.002	39	0.1	1.1	0.67	0.002	39
JPWH_991*	1.0	0.4	0.08	0.006	26	0.1	1.2		†	
MEMPLUS*	0.2	0.8	13.8	0.02	53	0.01	0.7	1110	0.04	137
ORSIRR_1*	0.1	0.9	0.1	0.002	29	0.01	1.7	3.8	0.004	35
ORSIRR_2*	0.03	1.4	0.08	0.002	27	0.01	1.7	2.8	0.004	34
ORSREG_1*	0.3	0.9	0.31	0.002	28	0.1	1.2	17.2	0.004	38
PORES_2*	0.05	2.6	0.18	0.004	55	0.0001	5.2	5.5	0.01	86
RAEFSKY1	0.06	0.3	1.6	0.02	42	0.1	0.7	40	0.03	50
RAEFSKY5	0.02	0.5	4.1	0.003	2	0.1	0.2	142	0.005	6
SHERMAN2*	0.01	1.9	0.24	0.002	13	0.1	5.1		†	
WATT_1	0.5	0.4	0.2	0.001	4	0.1	0.8	12.1	0.001	2
WATT_2	0.5	0.4	0.2	0.001	13	0.5	0.5	11.7	0.002	7

Table 2 shows the results with the V–AISM and AISM preconditioners. The first to observe is that the proposed preconditioner solved all the problems, while AISM failed to solve the problems SHERMAN2 and JPWH_991. Also, V–AISM had a clearly advantage in number of iterations for the matrices

MEMPLUS and PORES_2. Note that AISM needed a denser preconditioner to converge for PORES_2. For the rest, it seems that in general the balance between density and number of iterations was better for V–AISM. Concerning the computational time, V–AISM was extremely faster compared with AISM. The reason is that V–AISM avoids the inner loop present in the AISM algorithm and uses matrix-by-vector multiplications instead.

Table 3 reports the results for the ICI preconditioner compared with V–AISM. For ICI we mostly used a drop tolerance equal to 0.01 as it is done in [17]. We do not further sparsified the factors of the preconditioner wich is equivalent to apply a value of the parameter $F = \infty$ in [17]. We observe that ICI fails to converge for the matrix JPWH_991. For the matrices ADD20, MEMPLUS, ORSIRR_1, ORSIRR_2, ORSREG_1 and WATT_2, V–AISM has an advantage. By contrast, ICI obtains better results for PORES_2. For the rest of the matrices the results are quite similar.

Table 3: Comparison between V–AISM and ICI preconditioners.

Matrix	V–AISM					ICI				
	tol	ρ	T_p	T_s	Iter.	*tol*	ρ	T_p	T_s	Iter.
ADD20	0.1	0.7	0.37	0.001	8	0.01	0.6	0.29	0.01	136
CHEM_MASTER1	0.1	3.0	223	0.2	130	0.1	2.6	307	0.2	107
EPB3	0.1	2.1	619.0	0.53	196	0.1	3.7	1977	†	
FS_541_4*	0.01	1.1	0.04	0.001	5	0.01	1.1	0.06	0.03	5
HOR_131*	0.1	2.3	0.04	0.002	39	0.01	3.7	0.22	0.003	34
JPWH_991*	1.0	0.4	0.08	0.006	26	0.01	4.6		†	
MEMPLUS*	0.2	0.8	13.8	0.02	53	0.01	0.6	13.9	0.11	221
ORSIRR_1*	0.1	0.9	0.1	0.002	29	0.01	1.5	0.25	0.002	33
ORSIRR_2*	0.03	1.4	0.08	0.002	27	0.01	1.6	0.22	0.03	31
ORSREG_1*	0.3	0.9	0.31	0.002	28	0.01	1.6	0.7	0.003	34
POISSON3Db	0.1	0.4	973.0	1.04	210	0.1	0.4		†	
PORES_2*	0.05	2.6	0.18	0.004	55	0.01	2.5	0.6	0.003	30
RAEFSKY1	0.06	0.3	1.6	0.02	42	0.01	0.9	24.6	0.02	29
RAEFSKY5	0.02	0.5	4.1	0.003	2	0.01	0.7	13.4	0.002	2
SHERMAN2*	0.01	1.9	0.24	0.002	13	0.01	1.4	0.95	0.002	11
WATT_1	0.5	0.4	0.2	0.001	4	0.5	0.4	0.15	0.001	4
WATT_2	0.5	0.4	0.2	0.001	13	0.5	0.4	0.15	0.02	266

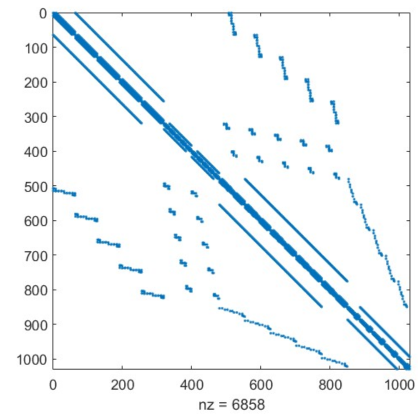
Table 4 shows the results comparing V–AISM and AINV. For AINV in the columns T_p the preconditioner computation time for MATLAB is written first and then FORTRAN. The solution time corresponds to the FORTRAN code since it was almost equal to the one obtained in MATLAB. One can see the big improvement that can be achieved with an optimized FORTRAN code. Due to the recursion nature of these algorithms the computational time may grow significantly with the matrix size, and therefore special coding techniques are needed to avoid it (see [1]). Comparing only the MATLAB times one can see that the computation of V–AISM is much faster than AINV. Thus, we believe that a fully optimized version of V–AISM can be at least as fast as AINV. We observe that V–AISM performs better for FS_541_4, MEMPLUS and WATT_2. By contrast, AINV was better for ADD20, and HOR_131. Note that AINV failed to solve the JPWH_991 and SHERMAN2 problems. For the rest of the matrices V–AISM and AINV performed similarly.

Table 4: Comparison between V–AISM and AINV preconditioners.

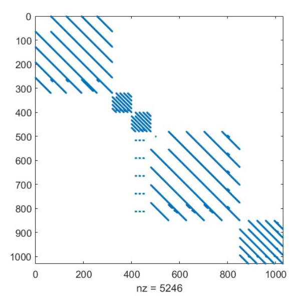
Matrix	V–AISM					AINV				
	tol	ρ	T_p	T_s	Iter.	*tol*	ρ	T_p	T_s	Iter.
ADD20	0.1	0.7	0.37	0.001	8	0.1	0.6	407/0.02	0.001	7
CHEM_MASTER1	0.1	3.0	223	0.2	130	0.1	2.6	§/0.1	0.4	156
EPB3	0.1	2.1	619.0	0.53	196	0.1	3.7	§/0.1	1.4	244
FS_541_4*	0.01	1.1	0.04	0.001	5	0.02	1.2	1.42/0.01	0.001	10
HOR_131*	0.1	2.3	0.04	0.002	39	0.1	1.8	1.0/0.01	0.002	26
JPWH_991*	0.1	1.4	0.11	0.001	13	0.1	1.2		†	
MEMPLUS*	0.2	0.8	13.9	0.02	53	0.1	0.6	§/0.05	0.1	155
ORSIRR_1*	0.1	0.9	0.1	0.002	29	0.1	0.9	4.35/0.01	0.002	32
ORSIRR_2*	0.03	1.4	0.08	0.002	27	0.1	0.9	3.24/0.01	0.002	36
ORSREG_1*	0.3	0.9	0.31	0.002	28	0.2	0.9	49.1/0.02	0.002	33
POISSON3Db	0.1	0.4	973.0	1.0	210	0.1	0.4	§/2.2	11.1	1312
PORES_2*	0.05	2.6	0.18	0.004	55	0.05	2.1	6.84/0.02	0.006	80
RAEFSKY1	0.06	0.3	1.6	0.02	42	0.05	0.2	76.9/0.07	0.03	53
RAEFSKY5	0.02	0.5	4.1	0.003	2	0.01	0.7	208.1/0.1	0.006	2
SHERMAN2*	0.01	1.9	0.24	0.002	13	0.1	4.0		†	
WATT_1	0.5	0.4	0.2	0.001	4	0.5	0.4	0.34/0.01	0.002	3
WATT_2	0.5	0.4	0.28	0.001	13	0.5	0.4	0.39/0.01	0.002	23

Finally, concerning the largest matrices of the set we see that the ICI preconditioner performed the best in density and number of iterations for CHEM_MASTER1 but failed to converge for the other two matrices EPB3 and POISSON3Db. V–AISM and AINV were able to solve the three problems, but V–AISM spent considerably less iterations and time for solving the POISSON3Db matrix.

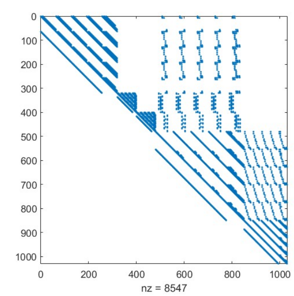
Figure 1 shows the nonzero patterns of the matrix ORSIRR_1 and the patterns of the sum of the factors of the different preconditioners analyzed. The patterns of V–AISM, AINV and ICI look similar showing that they capture more or less de same information. Note that the pattern of AISM is quite different. The reason is that the factors of this preconditioner does not approximate the inverse LU factors as the other preconditioners do. For all the preconditioners the density is similar and also the number of iterations needed to converge. Figure 2 shows the patterns for the matrix FS_541_4. We can



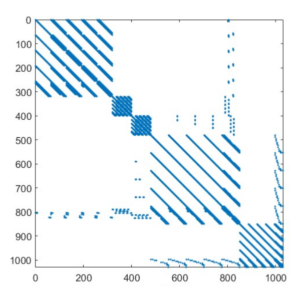
(a) ORSIRR_1



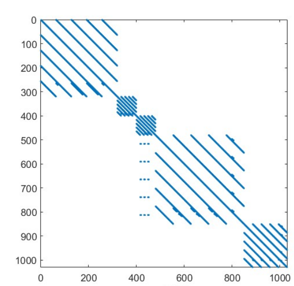
(b) AINV



(c) AISM

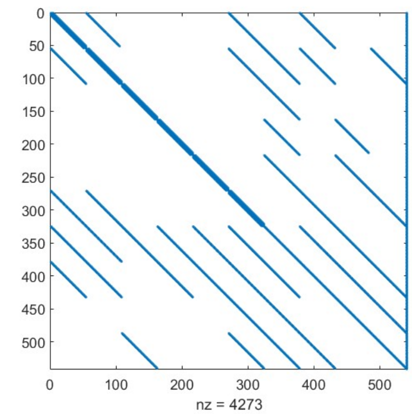


(d) ICI

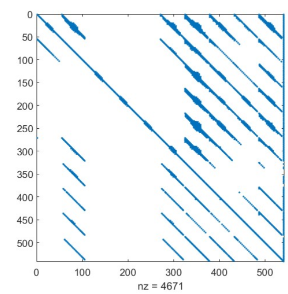


(e) V-AISM

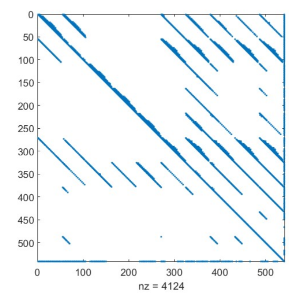
Figure 1: Nonzero patterns of the matrix ORSIRR_1 and the sum of the factors of the different preconditioners analyzed.



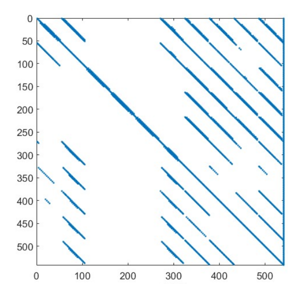
(a) FS_541_4



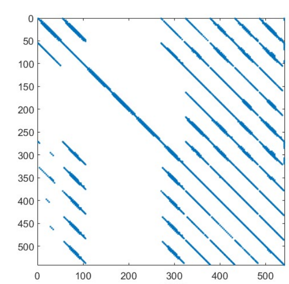
(b) AINV



(c) AISM



(d) ICI



(e) V-AISM

Figure 2: Nonzero patterns of the matrix FS_541_4 and the sum of the factors of the different preconditioners analyzed.

see that differences in the nonzero pattern in AINV with respect ICI and V–AISM, even with almost identical number of nonzero elements, could lead to different number of iterations, 10 iterations for AINV and 5 iterations for ICI and V–AISM.

5 Conclusions

We have introduced a new way to build an approximate factorization of the inverse of a nonsingular matrix applying recursively the Sherman–Morrison inversion formula. Then, with a compact representation of that decomposition a new preconditioner is built which is an approximate inverse LU preconditioner, referred to as V–AISM. These kind of preconditioners perform matrix-vector products in each iteration of the Krylov subspace method. Then, they are attractive for the parallel execution of the preconditioning step in Krylov subspace methods. An advantage of the new algorithm is that the main operations for computing the preconditioner are sparse matrix-vector products that opens the door for efficient implementations that it can be investigated in the future. Moreover this algorithm is stable for nonsingular M–matrices and H–matrices of the invertible class. The numerical computations done with a set of matrices from the Harwell–Boeing and SuiteSparse Matrix collections show that the proposed approximate inverse preconditioner V–AISM is robust and competitive with respect to AISM, AINV and ICI preconditioners.

References

[1] M. Benzi and M. Tuma. A sparse approximate inverse preconditioner for nonsymmetric linear systems. SIAM Journal on Scientific Computing, 19, 968–994, 1998.

[2] M. Benzi and M. Tuma. A comparative study of sparse approximate inverse preconditioners. Appl. Numer. Math. 30, 305–340, 1999.

[3] A. Berman and R. J. Plemmons. Nonnegative Matrices in the Mathematical Sciences. SIAM, Philadelphia, PA, USA, 1979.

[4] M. Bollhöfer and Y. Saad. On the relations between ILUs and factored approximate inverses. SIAM J. Matrix Anal. Appl., 24(1), 219–237, 2002.

[5] R. Bru, C. Corral, M. I. Gimenez and J. Mas. Classes of general H–matrices. Linear Algebra and its Applications. 429(10), 2358–2366, 2008.

[6] R. Bru, J. Cerdán, J. Marín and J. Mas. Preconditioning sparse nonsymmetric linear systems with the Sherman–Morrison formula. SIAM Journal on Scientific Computing, 25, 701–715, 2003.

[7] R. Bru, J. Marín, J. Mas and M. Tŭma. Balanced incomplete factorization. SIAM J. Sci. Comput., 30, 2302–2318, 2008.

[8] R. Bru, J. Marín, J. Mas and M. Tŭma. Improved balanced incomplete factorization. SIAM J. Matrix Anal. Appl., 31, 2431–2452, 2010.

[9] R. Byrd, J. Nocedal and R. Schnabel. Representations of Quasi-Newton matrices and their use in limited memory methods. Mathematical Programming, 63(13), 129–156, 1994.

[10] J. Cerdán, T. Faraj, N. Malla, J. Marín and J. Mas. Block approximate inverse preconditioners for sparse nonsymmetric linear systems. Electron. Trans. Numer. Anal., 37, 23–40, 2010.

[11] T. A. Davis and Y. Hu. The University of Florida Sparse Matrix Collection. ACM Transactions on Mathematical Software 38, 1, Article 1 (December 2011), 25 pages.

[12] I. S. Duff, R.G. Grimes and J. G. Lewis. Users' Guide for the Harwell-Boeing Sparse Matrix Collection. Tech. Report RAL 92–886, RutherfordAppleton Laboratory, Chilton, England, 1992.

[13] I. Duff, M. Heroux, R. Pozo. An Overview of the Sparse Basic Linear Algebra Subprograms: The New Standard from the BLAS Technical Forum. ACM Trans. Math. Softw. 28, 2, 239–267, 2002.

[14] W. W. Hager. Updating the inverse of matrix. SIAM Rev., 31(2), 221–239, 1989.

[15] A. Rafiei, M. Bollhöfer and F. Benkhaldoun. A block version of left-looking AINV preconditioner with one by one or two by two block pivots. Applied Mathematics and Computation, 350, 36–385, 2019.

[16] J. Sherman and W. J. Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. Ann. Math. Statist., 21, 124–127, 1950.

[17] C. N. van Duin. Scalable parallel preconditioning with the sparse approximate inverse of triangular matrices. SIAM J. Matrix Anal. Appl. 20(4), 987–1006, 1999. MR1699790.

Rafael Bru,
Institut de Matemàtica Multidisciplinar,
Universitat Politècnica de València,
Camí de Vera s/n, València, Spain.
Email: rbru@imm.upv.es

Juana Cerdán,
Institut de Matemàtica Multidisciplinar,
Universitat Politècnica de València,
Camí de Vera s/n, València, Spain.
Email: jcerdan@imm.upv.es

José Marín,
Institut de Matemàtica Multidisciplinar,
Universitat Politècnica de València,
Camí de Vera s/n, València, Spain.
Email: jmarinma@imm.upv.es

José Mas,
Institut de Matemàtica Multidisciplinar,
Universitat Politècnica de València,
Camí de Vera s/n, València, Spain.
Email: jmasm@imm.upv.es