Rev. Real Acad. Cienc. Exactas Fis. Nat. Ser. A-Mat. (2023) 117:5
https://doi.org/10.1007/s13398-022-01334-1

ORIGINAL PAPER

Balanced incomplete factorization preconditioner with pivoting

J. Marín[1] · **J. Mas**[1]

Received: 28 July 2022 / Accepted: 29 September 2022 / Published online: 12 October 2022

Abstract
In this work we study pivoting strategies for the preconditioner presented in Bru (SIAM J Sci Comput 30(5):2302–2318, 2008) which computes the LU factorization of a matrix A. This preconditioner is based on the Inverse Sherman Morrison (ISM) decomposition [Preconditioning sparse nonsymmetric linear systems with the Sherman–Morrison formula. Bru (SIAM J Sci Comput 25(2):701–715, 2003), that using recursion formulas derived from the Sherman-Morrison formula, obtains the direct and inverse LU factors of a matrix. We present a modification of the ISM decomposition that allows for pivoting, and so the computation of preconditioners for any nonsingular matrix. While the ISM algorithm at a given step computes only a new pair of vectors, the new pivoting algorithm in the k-th step also modifies all the remaining vectors from $k+1$ to n. Thus, it can be seen as a right looking version of the ISM decomposition. The results of numerical experiments with ill-conditioned and highly indefinite matrices arising from different applications show the robustness of the new algorithm, since it is able to solve problems that are not possible to solve otherwise.

Keywords Incomplete LU preconditioners · Iterative methods · Pivoting · Ill-conditioned problems · Sparse linear systems

Mathematics Subject Classification 65F08 · 65F10 · 65F50 · 65F05

1 Introduction

This paper is concerned with the computation of robust preconditioners by using standard pivoting techniques for solving ill-conditioned sparse nonsingular linear systems of equations of the form

$$Ax = b, \quad A \in \mathbb{R}^{n\times n}, \quad b \in \mathbb{R}^n, \tag{1}$$

✉ J. Mas
jmasm@imm.upv.es

J. Marín
jmarinma@imm.upv.es

[1] Instituto de Matemática Multidisciplinar, Universitat Politècnica de València, Valencia, Spain

using Krylov subspace methods. Devising robust preconditioning algorithms for (1) such that it can be solved efficiently by means of iterative methods still remains one of the most active research areas in numerical linear algebra. Preconditioned Krylov methods have been traditionally linked to the solution of large and sparse linear systems due to the relative small amount of memory and computation time needed to obtain an approximate solution compared with direct methods. There exist different techniques that can be used successfully to compute preconditioners, as incomplete LU factorizations, approximate inverses, algebraic methods, etc. (see [2] and the references therein). But in recent years they have also been employed in the context of mixed precision techniques for solving dense linear systems. The accuracy of an initial solution obtained with an LU factorization computed in single precision is improved by iterative refinement using the LU factorization as preconditioner [10, 15].

Ill-conditioned nonsingular linear systems arise in many areas of scientific and engineering applications, and computing numerically stable LU factorizations for these linear systems becomes a challenge [1, 13]. Pivoting has been originally used to compute good LU factorizations for such problems [14], but there have been also some work for incomplete factorizations [6, 18, 21]. Also MATLAB has incorporated this possibility into his function `ilu`, that computes the Incomplete LU factorization of a matrix, [17].

There are different pivoting techniques being partial, complete and rook pivoting the more important ones [14, 19]. Basically, at a given step of Gaussian elimination pivoting looks for an element sufficiently large in magnitude in the remaining submatrix, the Schur complement, to use it as the next pivot. These techniques involve row and possibly column permutations of the matrix that supposes a computational overhead. In this sense, partial pivoting is the cheapest pivoting technique, since it looks only in the first column of the Schur complement. Close behind is rook pivoting [20], it selects a pivot with maximum absolute value in its row and column, moving first to the biggest entry in magnitude in the first column, then it moves in the corresponding row, and then again in the column, and so on until the requirement is fulfilled. Finally, complete pivoting is the most expensive one, but guarantees the largest pivot at any stage however because the pivot is the entry of biggest magnitude in all the Schur complement.

In this work we study pivoting techniques for the balanced incomplete factorization preconditioner, BIF. BIF preconditioning is based on the incomplete Sherman-Morrison decomposition, ISM. The ISM decomposition uses recursion formulas derived from the Sherman-Morrison formula and was introduced in [7] as a method for computing approximate inverse preconditioners. In [8] the authors show that, applying the ISM algorithm to a symmetric and positive definite matrix A it is possible to compute an incomplete Cholesky factorization, later in [9] they showed that applying the ISM algorithm to A and A^T, it is possible to compute an incomplete LDU factorization. Moreover, in both cases the inverse factors are also available and they influence the computation of the Cholesky or LDU factorization, and vice versa. In addition, the availability of the direct and inverse factors is exploited to implement norm based dropping rules [5]. The numerical results show that BIF is a robust algorithm comparable to other techniques as ILU(τ) [3], ILUT (Threshold Incomplete LU) [22] and RIF (Robust Incomplete Factorization) [4]. Nevertheless, as mentioned above, computing stable (incomplete) factorizations for general ill-conditioned problems still require the application of pivoting techniques, except for some kind of matrices that can be solved with high accuracy, [16]. In this paper we show that with a slight modification of the ISM recursion formulas it is possible to incorporate pivoting to BIF, and obtain an algorithm which can be efficiently implemented and with similar efficiency that the well known incomplete LU with pivoting (ILUP). So, this study completes our previous work.

The paper is organized as follows. In Sect. 2 an overview of the ISM decomposition is presented. In Sect. 3 we introduce and analyze the right looking ISM decomposition. It is shown that the Schur complement computed with Gaussian elimination is available at each step of the modified algorithm and therefore, it is possible to incorporate any standard pivoting technique. In Sect. 4 the BIF algorithm with pivoting is presented and the results for several ill-conditioned matrices are reported. Our experiments show that BIF with pivoting is able to solve such a challenging problems and it is comparable to ILUT with partial pivoting. Finally, the main conclusions are presented in Sect. 5.

2 The ISM decomposition

The ISM decomposition was introduced in [7] as an algorithm to compute approximate inverse preconditioners since it obtains a factorization of the (shifted) inverse matrix of A, as

$$s^{-1}I - A^{-1} = s^{-2}ZD_s^{-1}V_s^T, \tag{2}$$

where $s > 0$ is a given scalar and the columns of the matrices Z and V_s are computed using the recursion formulas

$$z_k = e_k - \sum_{i=1}^{k-1} \frac{v_i^T e_k}{sr_i} z_i \quad \text{and} \quad v_k = y_k - \sum_{i=1}^{k-1} \frac{y_k^T z_i}{sr_i} v_i, \tag{3}$$

for $k = 1, 2, \ldots, n$. In (3) the vector e_k (e^k) denotes the $k-$th column (row) of the identity matrix, $y_k = (a^k - se^k)^T$ where a^k denotes the k-th row of A, and

$$r_k = 1 + y_k^T z_k/s = 1 + v_k^T e_k/s, \tag{4}$$

are the entries of the diagonal matrix D_s.

It was proved in [8] that for symmetric matrices the factorization $A = LDL^T$ and the decomposition (2) satisfy

$$D = sD_s, \quad Z = L^{-T}, \quad V_s = LD - sL^{-T}.$$

The algorithm to get the decomposition of A uses explicitly the computed factors of A^{-1}, that is, A^{-1} is implicitly factorized at the same time. In [9] it is proved the following result

Theorem 1 (Theorem 2.1 of [9]) *Let $A = LDU$ be the LDU decomposition of A, and let $s^{-1}I - A^{-1} = s^{-2}ZD_s^{-1}V_s^T$ be the ISM decomposition (2). Then*

$$Z = U^{-1}, \quad \textit{and} \quad V_s = U^T D - sL^{-T}. \tag{5}$$

Observe that L does not appear in (5). Therefore, to get the LU factorization for general matrices it is necessary to compute also the ISM decomposition of A^T that gives as result

$$\tilde{Z} = L^{-T}, \quad \text{and} \quad \tilde{V}_s = LD - sU^{-1},$$

where we have denoted with tilde the factors of the ISM decomposition of A^T.

It is well known that a nonsingular matrix A has an LU factorization if there exists a lower unit triangular matrix L and an upper triangular matrix U, such that $A = LU$. The LDU factorization is obtained from the LU factorization by taking D as the diagonal matrix whose entries are the diagonal entries of U, and applying its inverse to U as $D^{-1}U$. Both factorizations are closely related with Gaussian elimination. Note that not all the nonsingular

matrices have LU factorization since a zero pivot can be found during the Gaussian elimination process. However it is always possible to permute some rows, and maybe some columns of the matrix in such a way that the permuted matrix PAQ has LU factorization. Here P and Q are permutation matrices acting on rows and columns of A, respectively.

The idea is that it is possible to find permutation matrices P and Q such that at the k-th step of the Gaussian elimination process one obtains the matrix

$$(PAQ)^{(k)} = \begin{bmatrix} L_{11} & O \\ L_{21} & I \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ O & S^{(k)} \end{bmatrix},$$

where the Schur complement $S^{(k)} = A_{22} - A_{21}A_{22}^{-1}A_{12}$ is nonsingular and its first diagonal element is nonzero. Then, the permuted matrix PAQ is factorized as

$$PAQ = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix}. \tag{6}$$

Here, A_{11}, A_{12}, A_{21} and A_{22} represent the submatrices of the reordered matrix PAQ, and the size of A_{11} is $k \times k$.

Note that in practice, the permutation matrices P and Q are not known in advance and therefore LU factorization algorithms determine which rows and columns must be interchanged during the elimination process. In the next section we show that it is possible to obtain the factorization (6) from the ISM decomposition by modifying its recursion formulas.

3 Right looking ISM algorithm

To implement pivoting in the ISM decomposition it is necessary to know the Schur complement of the LU factorization. To accomplish that, the vectors z_k and v_k must be computed in a different way. Instead of computing only one pair of vectors in the k-th step of the algorithm according to equations (3), the modification consist in updating also the remaining vectors, from $k+1$ to n. That is, the right part of the matrices Z and V are updated in each step. The following MATLAB code implements the new right looking version of ISM.

Algorithm 1 The ISM right looking algorithm (ISMRL)

```
function [Z, V, D] = ismrl(A)
n = size(A,1);
Y = (A-eye(n))';
Z = eye(n);
V = A'-eye(n);
D = zeros(n,1);
for k=1:n-1
  D(k) = 1+V(k,k);
  for l = k+1:n
    Z(:,l) = Z(:,l) - V(l,k)/D(k)*Z(:,k);
    V(:,l) = V(:,l) - (Y(:,l)'*Z(:,k))/D(k)*V(:,k);
  end
end
D(n)=1+V(n,n);
```

Next, we will show that the Schur complement $S^{(k)}$ is available from the matrix V. As usual, we denote by u_{kj} the entry in the row k and column j of the matrix U, and by l_{ik} the entry in the row i and column k of the matrix L. Observe that after the k-th step the first k columns of Z and V are computed. From $A = LDU$ and Theorem 1, $AZ = LD$, and after the k-th step

$$\mathbf{a}^i \mathbf{z}_k = l_{ik} d_k, \quad \text{if } i > k\,, \tag{7}$$

and the j-th element of $\mathbf{v}_k$ is

$$v_{jk} = u_{kj}, \quad j \neq k\,. \tag{8}$$

From (7) and considering that $\mathbf{z}_k$ has zero entries bellow the row k, an important equality for the proof of our main result is

$$\mathbf{y}_i^T \mathbf{z}_k = \mathbf{a}^i \mathbf{z}_k - \mathbf{e}^i \mathbf{z}_k = \mathbf{a}^i \mathbf{z}_k = l_{ik} d_k, \quad i > k. \tag{9}$$

We denote by $V_{22}^{(k)}$ the $(n-k) \times (n-k)$ submatrix of V in Algorithm 1 after step k, with rows and columns with indexes in $\{k+1, \ldots, n\}$. Then we have the following result. To prove it we denote the entry in the row i and column j of a matrix M as m_{ij}.

Theorem 2 *If A is a nonsingular matrix, then at the k-th step of the right looking ISM Algorithm 1*

$$V_{22}^{(k)} = S^{(k)^T} - I. \tag{10}$$

Proof We are going to prove equation (10) by induction on the steps.

Clearly the initialization of V is $V = A^T - I$, so we can write $V^{(0)} = S^{(0)^T} - I$.

For $k = 1$ let us consider the element $a_{ij}^{(1)}$ of the Schur complement $S^{(1)}$, that is entries with $i, j > 1$. It is well known

$$a_{ij}^{(1)} = a_{ij}^{(0)} - \frac{a_{i1}^{(0)} a_{1j}^{(0)}}{a_{11}^{(0)}} = a_{ij} - l_{i1} u_{1j}$$

In the right looking ISM the (j, i) entry of the matrix $V^{(1)}$, for $i, j > 1, i \neq j$ is

$$v_{ji}^{(1)} = v_{ji}^{(0)} - \frac{\mathbf{y}_i^T \mathbf{z}_1^{(0)}}{r_1} \mathbf{v}_1^{(0)}(j) = a_{ij} - \frac{a_{i1}}{a_{11}} a_{1j} = a_{ij} - l_{i1} u_{1j}.$$

where we have used Eqs. (8) and (9).

Working in the same way when $i = j$, we have

$$v_{ii}^{(1)} = v_{ii}^{(0)} - \frac{\mathbf{y}_i^T \mathbf{z}_1^{(0)}}{r_1} \mathbf{v}_1^{(0)}(i) = (a_{ii} - 1) - \frac{a_{i1}}{a_{11}} a_{1i} = a_{ii} - l_{i1} u_{1i} - 1.$$

Then

$$V_{22}^{(1)} = S^{(1)^T} - I.$$

Assume now that

$$V_{22}^{(k-1)} = S^{(k-1)^T} - I.$$

Let us prove the equality for the k-th step. Consider the entries of the Schur complement $S^{(k)}$, that is $a_{ij}^{(k)}$ for $i, j > k$

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - \frac{a_{ik}^{(k-1)} a_{kj}^{(k-1)}}{a_{kk}^{(k-1)}} = a_{ij}^{(k-1)} - l_{ik} u_{kj}.$$

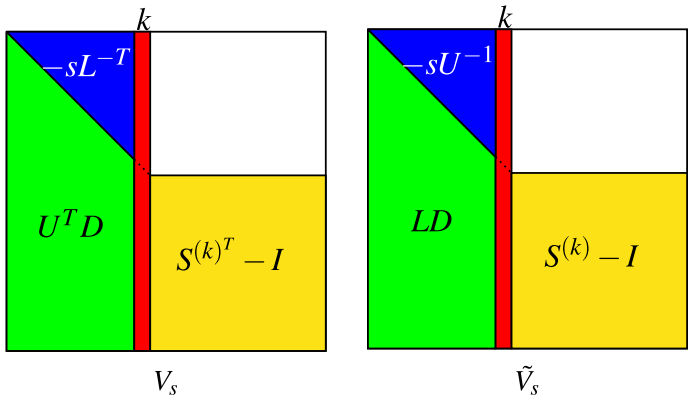

Fig. 1 Matrices V_s and $\tilde{V}_s$ at the k-th step of the right looking ISM algorithm

Again, in the right looking ISM the (j, i) entry of the matrix $V^{(k-1)}$, for $i, j > 1, i \neq j$ is

$$v_{ji}^{(k)} = v_{ji}^{(k-1)} - \frac{\mathbf{y}_i^T \mathbf{z}_k^{(k-1)}}{r_k} \mathbf{v}_k^{(k-1)}(j) = a_{ij}^{(k-1)} - \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}} a_{kj}^{(k-1)} = a_{ij}^{(k-1)} - l_{ik} u_{kj},$$

where we have used Eqs. (8) and (9).

Working in the same way, when $i = j$ we have

$$v_{ii}^{(k)} = v_{ii}^{(k-1)} - \frac{\mathbf{y}_i^T \mathbf{z}_k^{(k-1)}}{r_k} \mathbf{v}_k^{(k-1)}(i) = \left(a_{ii}^{(k-1)} - 1\right) - \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}} a_{ki}^{(k-1)} = a_{ii}^{(k-1)} - l_{ik} u_{ki} - 1.$$

Then

$$V_{22}^{(k)} = S^{(k)^T} - I.$$

□

Since we need to compute also the factorization of A^T, observe that.

Corollary 1 *If the right looking algorithm is applied to A^T then*

$$\tilde{V}_{22}^{(k)} = S^{(k)} - I.$$

Figure 1 shows a graphical representation of the above results. To introduce pivoting strategies the relation

$$V_{22}^{(k)} = S^{(k)^T} - I,$$

should be taken in account. The new pivot is looked for into the submatrix $V_{22}^{(k)} + I$ that corresponds to the transpose of the same submatrix in $A^{(k)}$ in Gaussian elimination. Thus, in partial pivoting if two columns k and $p > k$ are permuted at step k in matrix $V^{(k)}$, the rows k and p should be permuted in A.

Also note that the pivoting strategy should be decided by looking into the Schur complement contained in V_s, or that in $\tilde{V}_s$, but not both. In contrast, for complete pivoting it is clear that V_s or $\tilde{V}_s$ produce the same pivot in exact arithmetic so any of them or both may be used.

Table 1 Test problems

Matrix	n	nz	$cond(A)$	Application
adder_dcop_06	1813	11,224	$1.1 \cdot 10^{12}$	Circuit simulation matrix
adder_dcop_19	1813	11,224	$5.9 \cdot 10^{11}$	Circuit simulation matrix
adder_dcop_26	1813	11,224	$5.6 \cdot 10^{11}$	Circuit simulation matrix
adder_dcop_57	1813	11,224	$5.6 \cdot 10^{11}$	Circuit simulation matrix
oscil_dcop_01	1813	11,224	$5.9 \cdot 10^{12}$	Circuit simulation problem
oscil_dcop_57	1813	11,224	$1.4 \cdot 10^{21}$	Circuit simulation problem
radfr1	1048	13,299	$5.9 \cdot 10^{10}$	Chemical process separation
west0989	989	3518	$9.9 \cdot 10^{11}$	Chemical process separation
mahindas	1258	7682	$1.0 \cdot 10^{13}$	Economic model
orani678	2529	90,158	$1.0 \cdot 10^{7}$	Economic model
str_600	363	3279	$1.8 \cdot 10^{6}$	Simplex method
shl_400	663	1712	$1.9 \cdot 10^{7}$	Simplex method

4 Numerical experiments

In this section we report the results of some numerical experiments with a set of matrices from The SuiteSparse Matrix Collection [11] and the Harwell-Boeing collection [12]. The matrices are listed in Table 1 where their size, number of nonzeros, condition number and application are indicated. They correspond to very ill-conditioned and highly indefinite problems for which Gaussian elimination without pivoting fails to compute good quality L and U factors, so the same is expected to be the case for incomplete LU factorizations (see [5, 10]). Partial, rook and complete pivoting techniques have been tested. The experiments have been implemented and run in MATLAB R2022a. As iterative solvers the MATLAB implementation of full GMRES [23] and BiCGStab [24] were used. The right hand side vector was computed such that the solution was the vector of all ones, and the initial guess was the zero vector. The iterations were stopped when the initial residual was reduced by 8 orders of magnitude with a maximum number of $1,000$ iterations. To compare the results obtained with BIF the problems were also solved with the MATLAB's incomplete LU preconditioner with partial pivoting, ILUTP.

The implementation of the BIF preconditioner is based on the algorithm described in [9] but with the right looking modification described in Sect. 3. In [9] the authors show that in the ISM factorization the computation of the direct and inverse LU factors is interleaved and they mutually influence each other. These characteristics allows for the use of advanced dropping rules, see [5]. We will not discuss in detail these rules but we recall that their application requires the estimation of the norm of the columns of the LU factors and their inverses. Since approximations of these factors are explicitely available the application of this kind of dropping rules is straightforward.

For simplicity, all the experiments have been done with the the parameter s of the ISM decomposition equal to one. The algorithm is implemented such that the ISM decompositions of A and A^T are computed at the same time. Therefore, accessing A and A^T simultaneously is needed. The pivot is chosen from the Schur complement contained in V_s rather than $\tilde{V}_s$. We note that for complete pivoting the same pivot could be obtained working either with V_s

or $\tilde{V}_s$ but we choose working with V_s for simplicity. Thus, if in the k-th step of the algorithm the pivot strategy determines that the new pivot is in column p and row q, since V_s stores the transpose of the Schur complement, then the columns p and k in V_s and A^T must be interchanged. Observe that in $\tilde{V}_s$, $\tilde{Z}$ and A the rows p and k must be pivoted instead. By the same reason the rows q and k must be interchanged in V_s, Z and A^T, and the corresponding columns in $\tilde{V}_s$ and A. Also, other vectors whose elements depend on the column and row ordering, for instance vectors storing the norms of the columns needed for the dropping rule, must be reordered accordingly. Algorithm 2 sketches the pivoted version of the BIF algorithm described.

Algorithm 2 The BIF algorithm with pivoting (BIFP)

```
function [L, U, P, Q] = bifp(A, tolZ, tolV, pivoting)
%
% Input parameters
% A % input matrix,
% tolZ, tolV % drop tolerances for Z and V, respectively
% pivoting strategy
%
% Output parameters
% L, U % LU factors of A
% P, Q (permutation vectors PAQ = LU)
      % Initializations
n = size(A,1);
At = A';
Z = eye(n);
Zt = eye(n);
V = At - Z; % Matrix V
Vt = A - Zt; % Matrix Vt
nrm_l = zeros(n,1); % Norms of rows of L
nrm_il = zeros(n,1); % Norms of rows of inv(L)
nrm_u = zeros(n,1); % Norms of rows of U
nrm_iu = zeros(n,1); % Norms of rows of inv(U)
D = zeros(n,1); % Matrix D
Dt = zeros(n,1); % Matrix D of A'
      % Main loop
for k = 1:n-1
  if strcmp(pivoting, 'c') % Complete pivoting
    [indrow,indcol] = FindPivotComplete(V(k:n,k:n)+eye(n-
k+1), k);
  elseif strcmp(pivoting, 'r') % Rook pivoting
    [indrow,indcol] = FindPivotRook(V(k:n,k:n)+eye(n-k+1),k);
  elseif strcmp(pivoting, 'p') % Partial pivoting
    [indrow,indcol] = FindPivotPartial(V(k:n,k:n)+eye(n-
k+1),k);
  else % No pivoting
    indcol = k; indrow = k;
  end
      % Permutations
  P( [k, indcol] ) = P( [indcol, k] ); % Permute row
permutation vector
  [At,V] = pivotcols(At,V,k,indcol);
```

```
   [A,Vt,Zt] = pivotrows(A,Vt,Zt,k,indcol);
   nrm_l( [k, indcol] ) = nrm_l( [indcol, k] );
   Q( [k, indrow] ) = Q( [indrow, k] );
   [At,V] = pivotrows(At,V,k,indrow);
   [A,Vt,Z] = pivotcols(A,Vt,Z,k,indrow);
   nrm_u( [k, indrow] ) = nrm_u( [indrow, k] );
       % Apply dropping
   D(k) = 1 + V(k,k);
   temp = 1.0/(D(k)*D(k));
   nrm_il(k) = sqrt(1.0+norm(V(1:k-1,k)));
   nrm_iu(k) = sqrt(1.0+norm(Vt(1:k-1,k)));
   nrm_u(k+1:n) = nrm_u(k+1:n)+temp*V(k+1:n,k).*V(k+1:n,k);
   nrm_l(k+1:n) = nrm_l(k+1:n)+temp*Vt(k+1:n,k).*Vt(k+1:n,k);
   nrm_l(k) = sqrt(nrm_l(k) +1.0);
   nrm_u(k) = sqrt(nrm_u(k) +1.0);
   V(1:k-1,k)=V(1:k-1,k).*((abs(V(1:k-1,k))> tolV.
/nrm_l(1: k-1)));
   V(k+1:n,k)=V(k+1:n,k).*((abs(V(k+1:n,k))>tolV*abs(D(k))
/nrm_iu(k)));
   Vt(1:k-1,k)=Vt(1:k-1,k).*((abs(Vt(1:k-1,k))
>tolV. /nrm_u(1:k-1)));
   Vt(k+1:n,k)=Vt(k+1:n,k).*((abs(Vt
(k+1:n,k))>tolV*abs(D(k)) /nrm_il(k)));
   Z(1:k-1,k)=Z(1:k-1,
k).*((abs(Z(1:k-1,k))> tolZ./nrm_l(1:k-1)));
   Zt(1:k-1,k)=Zt(1:k-1,k).*((abs(Zt(1:k-1,k))>tolZ.
/nrm_u(1:k-1)));
       % Update matrices V, Z, Vt, Zt
   for l = k+1:n
     Z(:,l) = Z(:,l) - V(l,k)/D(k)*Z(:,k);
     V(:,l) = V(:,l) - (At(:,l)'*Z(:,k))/D(k)*V(:,k);
     Zt(:,l) = Zt(:,l) - Vt(l,k)/Dt(k)*Zt(:,k);
     Vt(:,l) = Vt(:,l) - (A(:,l)'*Zt(:,k))/Dt(k)*Vt(:,k);
   end end
D(n)=1 + V(n,n);
       % Results
L=tril(Vt)+eye(n);
U=((tril(V)+eye(n))/diag(D))';
```

The algorithm first determines the pivot position according to the pivoting strategy applied, complete, rook or partial. Then, permutation of the matrices and vectors are performed. After that, norms of the columns of the LU factors and their inverses are updated and the dropping rule is applied. The recursion formula of the factorization is applied. Finally, after n steps the LU factors are extracted from matrices V and $\tilde{V}$.

In Tables 2 and 3 the pivoting strategy is indicated with C, P and R for the complete, partial and rook pivoting strategies, respectively. $Density$ is the ratio between the number of nonzeros of the preconditioner and the number of nonzeros of the matrix, that is $\frac{\text{nnz}(L)+\text{nnz}(U)}{\text{nnz}(A)}$. Column $iter$ shows the number of iterations of the solver and $droptol$ is the tolerance used to drop elements in BIFP and ILUTP. The other columns are self explanatory. To reduce the numbers in the tables, a blank space means that the value is the same appearing in previous

Table 2 Test results for the University of Florida matrices

Matrix	*precond*	*solver*	*droptol*	*piv*	*density*	*iter*
	BIFP	GMRES	10^{-6}	C	1.76	4
				P	1.81	3
				R	2.92	4
adder_dcop_06		BiCGStab		C		1
				P		1
				R		1
	ILUTP	GMRES	10^{-7}	P	1.67	3
		BiCGStab		P		1
	BIFP	GMRES	10^{-6}	C	0.69	3
				P	0.94	3
				R	0.72	3
adder_dcop_19		BiCGStab		C		2
				P		1
				R		2
	ILUTP	GMRES	10^{-2}	P	0.70	6
		BiCGStab		P		2
	BIFP	GMRES	10^{-6}	C	0.61	3
				P	0.73	4
				R	0.64	4
adder_dcop_26		BiCGStab		C		2
				P		1
				R		2
	ILUTP	GMRES	10^{-2}	P	0.60	6
		BiCGStab		P		2
	BIFP	GMRES	10^{-1}	C	0.52	2
				P	0.51	3
				R	0.51	2
adder_dcop_57		BiCGStab		C		4
				P		3
				R		4
	ILUTP	GMRES	10^{-1}	P	0.56	8
		BiCGStab		P		3
	BIFP	GMRES	10^{-7}	C	2.06	10
				P	2.64	19
				R	2.08	11
oscil_dcop_01		BiCGStab		C		1

Table 2 continued

Matrix	*precond*	*solver*	*droptol*	*piv*	*density*	*iter*
				P		4
				R		1
	ILUTP	GMRES	10^{-9}	P	2.70	3
		BiCGStab		P		1
	BIFP	GMRES	10^{-16}	C	2.31	11
			10^{-11}	P	2.28	28
			10^{-16}	R	2.57	11
oscil_dcop_57		BiCGStab		C		1
				P		2
				R		1
	ILUTP	GMRES	10^{-16}	P	2.74	11
		BiCGStab		P		1
	BIFP	GMRES	10^{-2}	C	2.25	3
			10^{-5}	P	3.86	1
			10^{-2}	R	2.70	2
radfr1		BiCGStab		C		9
				P		3
				R		8
	ILUTP	GMRES	10^{-3}	P	2.69	2
		BiCGStab		P		10

rows. For instance, in Table 2 the *droptol* value for BIFP was always 10^{-6} and therefore it appears only in the first row. The same holds for the preconditioner densities which are the same for GMRES and BiCSTAB and therefore only indicated once.

Next, we will comment on the results. We note that the matrices tested can not be solved without pivoting with both BIFP and ILUTP preconditioners. Thus, pivoting is an essential tool to gain robustness for these factorizations. Starting with the University of Florida test matrices, Table 2, we observe for the adder group that there are not big differences between the different pivoting strategies for BIFP. Density is small, except for adder_dcop_06. The same can be said for the number of iterations spent by both iterative solvers. For the rest of matrices one can see that BIFP with complete pivoting computes sparser preconditioners than partial and rook pivoting. The iteration count does not present remarkable differences except for the oscil_dcop_01 matrix for which GMRES with partial pivoting, although with larger nonzero density, doubles the number of iterations.

For the Harwell-Boeing matrices reported in Table 3 the first thing to observe is that the preconditioners are quite dense, with the exception of partial pivoting for the matrix orani678. Complete pivoting still produces less fill-in in the preconditioner and the number of iterations is similar to the other pivoting strategies. Note however that partial pivoting performs extraordinary well for the matrix orani678 since it is able to converge in the same number of iterations with a very sparse preconditioner.

Finally, comparing the performance of BIFP with ILUTP we did not observed significant differences, specially with the preconditioned BiCGStab method. We recall that ILUTP uses

Table 3 Harwell-Boeing group

Matrix	*prec*	*solver*	*droptol*	*piv*	*density*	*iter*
	BIFP	GMRES	10^{-6}	C	3.98	7
				P	4.70	6
				R	7.49	8
west0989		BiCGStab		C		2
				P		1
				R		2
	ILUTP	GMRES	10^{-6}	P	4.59	11
		BiCGStab		P		5
	BIFP	GMRES	10^{-8}	C	3.78	2
				P	6.28	3
				R	5.59	2
mahindas		BiCGStab		C		1
				P		1
				R		1
	ILUTP	GMRES	10^{-8}	P	5.49	2
		BiCGStab		P		1
	BIFP	GMRES	10^{-2}	C	3.07	11
				P	0.57	11
				R	11.33	11
orani678		BiCGStab		C		5
				P		6
				R		11
	ILUTP	GMRES	10^{-2}	P	0.38	10
		BiCGStab		P		6
	BIFP	GMRES	10^{-2}	C	1.62	9
			10^{-3}	P	1.97	9
			10^{-2}	R	1.61	9
str_600		BiCGStab		C		6
				P		8
				R		7
	ILUTP	GMRES	10^{-3}	P	1.82	6
		BiCGStab		P		5
	BIFP	GMRES	10^{-4}	C	2.38	1
				P	1.78	1
				R	2.64	1
shl_400		BiCGStab		C		1
				P		1
				R		1
	ILUTP	GMRES	10^{-4}	P	1.79	1
		BiCGStab		P		1

partial pivoting and we observe that BIFP with this pivoting strategy performed closely in most cases.

5 Conclusions

In this paper we have presented an improved version of the BIF preconditioner that incorporates pivoting. The algorithm relies on a modification of the recursion formulas such that the Schur complement of standard Gaussian elimination is available at each step of the factorization. Thus, the application of different pivoting techniques, as for instance partial, rook and complete pivoting, can be done in a straightforward manner. Incorporating pivoting turns out to be an important step in order to achieve our initial goal of obtaining a more robust preconditioner since it is able to solve very ill-conditioned and indefinite problems that it may not be possible to solve in other way. The results of the numerical experiments with several matrices arising in different applications confirm that BIF with pivoting is a robust algorithm. Partial, rook and complete pivoting has been tested. Although complete pivoting very often produces sparser preconditioners with a competitive iteration count, rook and partial pivoting perform also quite well. Taking into account that partial and rook are less expensive from a computational point of view since they need less comparisons in order to determine the pivot, these two techniques may be preferable as default. Also, a comparison with ILU with partial pivoting (ILUP) has been done and one can see that the results between them are fairly close. As a final note on future work, since with the ISM decomposition one can also compute incomplete approximate inverse preconditioners, it can be worth to explore its application for ill-conditioned problems, as it is done for instance in [15].

Acknowledgements The authors thank the anonymous referees for their helpful comments that improved the quality of the manuscript.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. The work was supported by Conselleria de Innovación, Universidades, Ciencia y Sociedad Digital, Generalitat Valenciana (CIAICO/2021/162).

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Beliakov, G., Matiyasevich, Y.: A parallel algorithm for calculation of determinants and minors using arbitrary precision arithmetic. BIT **56**, 33–50 (2015). https://doi.org/10.1007/s10543-015-0547-z

2. Benzi, M.: Preconditioning techniques for large linear systems: A survey. J. Comput. Phys. **182**, 418–477 (2002). https://doi.org/10.1006/jcph.2002.7176
3. Benzi, M., Tůma, M.: A comparative study of sparse approximate inverse preconditioners. Appl. Numer. Math. **30**(2-3), 305–340 (1999). https://doi.org/10.1016/S0168-9274(98)00118-4
4. Benzi, M., Tůma, M.: A robust incomplete factorization preconditioner for positive definite matrices. Numer. Linear Algebra Appl. **10**(5-6), 385–400 (2003). https://doi.org/10.1002/nla.320
5. Bollhöfer, M.: A robust and efficient ILU that incorporates the growth of the inverse triangular factors. SIAM J. Sci. Comput. **25**(1), 86–103 (2003). https://doi.org/10.1137/S1064827502403411
6. Bollhöfer, M., Saad, Y.: A factored approximate inverse preconditioner with pivoting. SIAM J. Matrix Anal. Appl. **23**, 692–705 (2001). https://doi.org/10.1137/S0895479800372122
7. Bru, R., Cerdán, J., Marín, J., Mas, J.: Preconditioning sparse nonsymmetric linear systems with the Sherman–Morrison formula. SIAM J. Sci. Comput. **25**(2), 701–715 (2003). https://doi.org/10.1137/S1064827502407524
8. Bru, R., Marín, J., Mas, J., Tůma, M.: Balanced incomplete factorization. SIAM J. Sci. Comput. **30**(5), 2302–2318 (2008). https://doi.org/10.1137/070696088
9. Bru, R., Marín, J., Mas, J., Tůma, M.: Improved balanced incomplete factorization. SIAM J. Matrix Anal. Appl. **31**(5), 2431–2452 (2010). https://doi.org/10.1137/090747804
10. Carson, E., Higham, N.: A new analysis of iterative refinement and its application to accurate solution of ill-conditioned sparse linear systems. SIAM J. Sci. Comput. **39**(6), A2834–A2856 (2017). https://doi.org/10.1137/17M1122918
11. Davis, T.A., Hu, Y.: The University of Florida sparse matrix collection. ACM Trans. Math. Software **38**(1), 1–25 (2011). https://doi.org/10.1145/2049662.2049663
12. Duff, I., Grimes, R., Lewis, J.: Sparse matrix test problems. ACM Trans. Math. Softw. **15**, 1–14 (1989). https://doi.org/10.1145/62038.62043
13. Ferronato, M., Janna, C., Pini, G.: Parallel solution to ill-conditioned FE geomechanical problems. Int. J. Numer. Anal. Methods Geomech. **36**, 422–437 (2012). https://doi.org/10.1002/nag.1012
14. Higham, N.J.: Accuracy and stability of numerical algorithms. SIAM, Philadelphia (2002)
15. Kobayashi, Y., Ogita, T.: A fast and efficient algorithm for solving ill-conditioned linear systems. JSIAM Lett. **7**, 1–4 (2015). https://doi.org/10.14495/jsiaml.7.1
16. Mainar, E., Peña, J., Rubio, B.: Accurate computations with Gram and Wronskian matrices of geometric and Poisson bases. Rev. Real Acad. Cienc. Exactas Fis. Nat. Ser. A-Mat. **116**(126), 1–22 (2022). https://doi.org/10.1007/s13398-022-01253-1
17. MATLAB: version 9.12.0.1927505 (R2022a) Update 1. The MathWorks Inc., Natick, Massachusetts (2022)
18. Mayer, J.: ILUCP: A Crout ILU preconditioner with pivoting. Numer. Linear Algebra Appl. **12**, 941–955 (2005). https://doi.org/10.1002/nla.456
19. Neal, L., Poole, G.: A geometric analysis of Gaussian elimination. II. Linear Algebra Appl. **173**, 239–264 (1992). https://doi.org/10.1016/0024-3795(92)90432-A
20. Poole, G., Neal, L.: The rook's pivoting strategy. Journal of Computational and Applied Mathematics **123**, 353–369 (2000). https://doi.org/10.1016/S0377-0427(00)00406-4.https://www.sciencedirect.com/science/article/pii/S0377042700004064
21. Rafiei, A.: A complete pivoting strategy for the right-looking robust incomplete factorization preconditioner. Comput. Math. Appl. **64**, 2682–2694 (2012). https://doi.org/10.1016/j.camwa.2012.08.001
22. Saad, Y.: ILUT: A dual threshold incomplete lu factorization. Numer. Linear Algebra Appl. **1**(4), 387–402 (1994). https://doi.org/10.1002/nla.1680010405
23. Saad, Y., Schulz, M.: GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. **7**(3), 856–869 (1986). https://doi.org/10.1137/0907058
24. van der Vorst, H.: Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. **12**, 631–644 (1992). https://doi.org/10.1137/0913035