Opuscula Math. 35, no. 2 (2015), 235–250
http://dx.doi.org/10.7494/OpMath.2015.35.2.235 Opuscula Mathematica

ILU PRECONDITIONING BASED ON THE FAPINV ALGORITHM

Davod Khojasteh Salkuyeh, Amin Rafiei, and Hadi Roohani

Communicated by Zdzisław Jackiewicz

Abstract. A technique for computing an ILU preconditioner based on the factored approximate inverse (FAPINV) algorithm is presented. We show that this algorithm is well-defined for H-matrices. Moreover, when used in conjunction with Krylov-subspace-based iterative solvers such as the GMRES algorithm, results in reliable solvers. Numerical experiments on some test matrices are given to show the efficiency of the new ILU preconditioner.

Keywords: system of linear equations, preconditioner, FAPINV, ILU preconditioner, H-matrix, GMRES.

Mathematics Subject Classification: 65F10, 65F50.

1. INTRODUCTION

Consider the linear system of equations

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

where the coefficient matrix $A \in \mathbb{R}^{n\times n}$ is nonsingular, large, sparse and $x, b \in \mathbb{R}^n$. Such linear systems are often solved by Krylov subspace methods such as the GMRES [28] and the BiCGSTAB [32] methods. In general, the convergence of Krylov subspace methods is not guaranteed or it may be extremely slow. Hence, the original system (1.1) is transformed into a more tractable form. More precisely, to obtain good convergence rates, or even to converge, Krylov subspace methods are applied to the left preconditioned linear system

$$MAx = Mb,$$

or to the right preconditioned linear system

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

where the matrix M is a proper preconditioner. Two-side preconditioning is also possible [1, 7, 26].

There are two general ideas for constructing a preconditioner. The first one is to find a matrix G approximating A in some sense and to set $M = G^{-1}$. In this case, M should be chosen such that AM (or MA) is a good approximation of the identity matrix. The best-known general-purpose preconditioners in this class are those based on the incomplete LU (ILU) factorization of the original matrix. Let the matrix A admits the LDU factorization $A = LDU$, where L and U^T are lower unitriangular matrices and $D = \mathrm{diag}(d_1, d_2, \ldots, d_n)$ is a diagonal matrix. Here $G = \tilde{L}\tilde{D}\tilde{U}$, where $\tilde{L}$ and $\tilde{U}^T$ are sparse lower unitriangular matrices which approximate L and U^T, respectively, and $\tilde{D}$ is a diagonal matrix which approximates D. The ILU preconditioners are very effective in increasing the rate of convergence. Their main drawback is the possibility of breakdowns during the incomplete factorization process, due to the occurrence of a zero or small pivots (or the appearance of nonpositive pivot elements for symmetric positive definite (SPD) matrices). In [21], Meijerink and van der Vorst have shown that this type of preconditioning exists for M-matrices. Then, Manteuffel in [20] has extended this result to the H-matrices. We recall that the matrix $A = (a_{ij})$ is an M-matrix if $a_{ij} \leq 0$ for all $i \neq j$, A is nonsingular and $A^{-1} \geq 0$. Moreover, A is an H-matrix if its comparison matrix $\hat{A} = (\hat{a}_{ij})$ is an M-matrix where

$$\hat{a}_{ij} = \begin{cases} -|a_{ij}|, & i \neq j, \\ |a_{ii}|, & i = j. \end{cases}$$

For general matrices, there are some ways to guard against the appearance of zero or very small pivots, see for example [8, 14, 23]. Another drawback of the ILU preconditioners is the lack of inherent parallelism. Many researchers have made effort to improve the accuracy and the degree of parallelism of the ILU preconditioners in the past [3, 5, 22, 25–27, 29, 30].

The second idea for constructing a preconditioner is to find a matrix M that directly approximates A^{-1} ($M \approx A^{-1}$). In this case, in practice we do not need to compute AM (or MA) explicitly, because when Krylov subspace methods are used to solve a preconditioned system, only the matrix-vector product is required. One drawback of many sparse approximate inverse techniques is their high construction cost, unless the computation can be done efficiently on parallel computers. One approach in this class is to compute a sparse approximate inverse in the factored form. Here from LDU factorization $A = LDU$, we have $A^{-1} = ZD^{-1}W$ where $W = L^{-1}$ and $Z = U^{-1}$. If $G = \tilde{Z}\tilde{D}\tilde{W}$, where $\tilde{W} \approx W$, $\tilde{Z} \approx Z$, and $\tilde{D} \approx D^{-1}$, in which $\tilde{W}$ and $\tilde{Z}^T$ are sparse lower unitriangular matrices and $\tilde{D}$ is diagonal matrix, then G may be used as a preconditioner for system (1.1) and is called a factored sparse approximate inverse. Here $\tilde{W}$ and $\tilde{Z}$ are called sparse approximate inverse factors of A. There are several algorithms to compute a factored sparse approximate inverse of a matrix. Among them are the factorized sparse approximate inverse (FSAI) algorithm [12, 13], the approximate inverse via bordering (AIB) algorithm [26], the approximate inverse (AINV) technique [4, 6], and the factored approximate inverse (FAPINV) algorithm [15, 17–19, 33, 34].

For SPD matrices, there exists a variant of the AINV method, denoted by SAINV (for Stabilized AINV), that is breakdown-free [2]. This algorithm is also presented, independently, by Kharchenko *et al.* in [11]. Benzi and Tuma in [8] have introduced an ILU factorization based on the SAINV algorithm. In the proposed algorithm the L factor of $\mathrm{LDL}^{\mathrm{T}}$ factorization of A can be obtained as a by-product of the A-orthogonalization process used in the SAINV algorithm, at no extra cost. Rezghi and Hosseini in [23], have shown that a similar algorithm free from breakdown can be established for nonsymmetric positive definite matrices.

The main idea of the FAPINV algorithm was introduced by Luo [17–19]. Then the algorithm was investigated by Zhang in [34]. Since in this procedure the factorization is performed in a backward direction, we call it the BFAPINV (Backward FAPINV) algorithm. In [33], Zhang proposed an alternative procedure to compute the factorization in the forward direction, which we call it the FFAPINV (Forward FAPINV) algorithm. In [15], Lee and Zhang have shown that the BFAPINV algorithm is well-defined for M-matrices. It can be easily seen that this is correct for the FFAPINV algorithm as well. In [31], Salkuyeh showed that the FFAPINV algorithm with a simple revision may be used for nonsymmetric positive definite matrices, free from breakdown.

In this paper, we show that the FFAPINV algorithm is free from breakdown for H-matrices and we propose a technique for computing an ILU preconditioner based on the FAPINV algorithm at no extra cost.

This paper is organized as follows. In Section 2, we review the FFAPINV algorithm. Section 3 is devoted to the main results. Numerical experiments are given in Section 4. Finally, we give some concluding remarks in Section 5.

2. A REVIEW OF THE FFAPINV ALGORITHM

Let W and Z^T be lower unitriangular matrices and D be a diagonal matrix. Also, suppose that

$$WAZ = D^{-1}. \tag{2.1}$$

In this case, we term W, Z and D, the inverse factors of $A = (a_{ij})$. Consider $W = (w_1^T, w_2^T, \ldots, w_n^T)^T$, $Z = (z_1, z_2, \ldots, z_n)$ and $D = \mathrm{diag}(d_1, d_2, \ldots, d_n)$, in which w_i's and z_i's are the rows and columns of W and Z, respectively. Using Eq. (2.1) we obtain

$$w_i A z_j = \begin{cases} \frac{1}{d_i}, & i = j, \\ 0, & i \neq j. \end{cases} \tag{2.2}$$

From the structure of the matrices W and Z, we have

$$z_1 = e_1, \quad z_j = e_j - \sum_{i=1}^{j-1} \alpha_i^{(j)} z_i, \quad j = 2, \ldots, n, \tag{2.3}$$

$$w_1 = e_1^T, \quad w_j = e_j^T - \sum_{i=1}^{j-1} \beta_i^{(j)} w_i, \quad j = 2, \ldots, n, \tag{2.4}$$

for some $\alpha_i^{(j)}$'s and $\beta_i^{(j)}$'s, where e_j is the j-th column of the identity matrix.

First of all, from (2.2) we see that

$$d_1 = \frac{1}{w_1^T A z_1} = \frac{1}{e_1^T A e_1} = \frac{1}{a_{11}}.$$

Now let $2 \leq j \leq n$ be fixed. Then from Eqs. (2.2) and (2.3) and for $k = 1, \ldots, j-1$, we have

$$0 = w_k A z_j = w_k A e_j - \sum_{i=1}^{j-1} \alpha_i^{(j)} w_k A z_i = w_k A_{*j} - \alpha_k^{(j)} w_k A z_k = w_k A_{*j} - \alpha_k^{(j)} \frac{1}{d_k},$$

where A_{*j} is the j-th column of A. Therefore,

$$\alpha_i^{(j)} = d_i w_i A_{*j}, \quad i = 1, \ldots, j-1.$$

In the same manner

$$\beta_i^{(j)} = d_i A_{j*} z_i, \quad i = 1, \ldots, j-1,$$

where A_{j*} is the j-th row of A. Pre-multiplying both sides of (2.3) by $w_j A$ yields

$$d_j = \frac{1}{w_j A_{*j}}.$$

Putting these results together gives the following algorithm (FFINV for Forward Factored INVerse) for computing the inverse factors of A.

Algorithm 1. FFINV algorithm (vector form)

1. $z_1 := e_1$, $w_1 := e_1^T$ and $d_1 := \frac{1}{a_{11}}$
2. For $j = 2, \ldots, n$, Do
3. $\quad z_j := e_j; \quad w_j := e_j^T$
4. $\quad$ For $i = 1, \ldots, j-1$, Do
5. $\quad\quad \alpha_i^{(j)} := d_i w_i A_{*j}; \quad \beta_i^{(j)} := d_i A_{j*} z_i$
6. $\quad\quad z_j := z_j - \alpha_i^{(j)} z_i; \quad w_j := w_j - \beta_i^{(j)} w_i$
7. $\quad$ EndDo
8. $\quad d_j := \frac{1}{w_j A_{*j}}$
9. EndDo
10. Return $W = [w_1^T, \cdots, w_n^T]^T$, $Z = [z_1, \cdots, z_n]$ and $D = diag(d_1, \cdots, d_n)$.

Algorithm 1, is the vector form of the FFINV algorithm. It can be easily verified that this algorithm is equivalent to Algorithm 2 (see [34]). Moreover, the values of $\alpha_i^{(j)}$'s and $\beta_i^{(j)}$'s are the same in both Algorithms 1 and 2. In this algorithm, we assume $w_j = (w_{j1}, w_{j2}, \ldots, w_{jn})$ and $z_j = (z_{1j}, z_{2j}, \ldots, z_{nj})^T$.

Algorithm 2. FFINV algorithm

1. $W := I_{n\times n}$, $Z := I_{n\times n}$.
2. For $j = 1, \ldots, n$, Do

3. For $i = j-1, \ldots, 1$, Do
4. $\quad l_i^{(j)} = a_{ji} + \sum_{k=1}^{i-1} a_{jk} z_{ki}$
5. $\quad \beta_i^{(j)} = l_i^{(j)} d_i$
6. EndDo
7. For $i = 1, \ldots, j-1$, Do
8. $\quad w_{ji} = -\beta_i^{(j)} - \sum_{k=i+1}^{j-1} \beta_k^{(j)} w_{ki}$
9. EndDo,
10. $d_j = 1/(a_{jj} + \sum_{k=1}^{j-1} w_{jk} a_{kj})$
11. For $i = j-1, \ldots, 1$, Do
12. $\quad u_i^{(j)} = a_{ij} + \sum_{k=1}^{i-1} w_{ik} a_{kj}$
13. $\quad \alpha_i^{(j)} = u_i^{(j)} d_i$
14. EndDo
15. For $i = 1, \ldots, j-1$ Do
16. $\quad z_{ij} = -\alpha_i^{(j)} - \sum_{k=i+1}^{j-1} \alpha_k^{(j)} z_{ik}$,
17. EndDo
18. EndDo
19. Return $W = (w_{ij})$, $Z = (z_{ij})$ and $D = diag(d_1, \cdots, d_n)$.

This algorithm computes the inverse factors W, Z and D such that Eq. (2.1) holds. Therefore, we have $A^{-1} = ZDW$. This shows that the inverse of A in the factored form can be computed by this algorithm. A sparse approximate inverse of A is computed by inserting some dropping strategies (replacing small values in absolute value by zero) in Algorithm 2. A dropping strategy can be used as follows. The dropping strategy is applied in four places in the algorithm. In step 5, if $|\beta_i^{(j)}| < \tau$, then $\beta_i^{(j)} := 0$, and in step 8, if $|w_{ji}| < \tau$, then $w_{ji} := 0$. In the same way $\alpha_i^{(j)}$ in step 13 and z_{ij} in step 16 are dropped when their absolute values are less than tolerance τ. Hereafter, the FFAPINV algorithm refers to the FFINV algorithm with this type of dropping. Here we mention that the dropping strategy proposed in [33] is slightly different from our dropping strategy. In the next section, we show that the FFAPINV algorithm is well-defined for H-matrices.

3. EXISTENCE OF FFAPINV ALGORITHM FOR H-MATRICES

First we state the following theorem.

Theorem 3.1. *Assume that A is an M-matrix. Let W and Z be the inverse factors of A computed by the FFINV algorithm, i.e. $WAZ = D^{-1}$. Also suppose that $\hat{W}$ and $\hat{Z}$ be the inverse factors of A computed by the FFAPINV algorithm, i.e. $\hat{W}A\hat{Z} \approx \hat{D}^{-1}$. Then*

$$W \geq \hat{W} \geq 0, \quad Z \geq \hat{Z} \geq 0,$$
$$\frac{1}{d_j} \geq \frac{1}{\hat{d}_j} > 0, \quad j = 1, 2, \ldots, n,$$

where $\tilde{D} = \mathrm{diag}(\tilde{d}_1, \ldots, \tilde{d}_n)$.

Proof. The proof of this theorem is quite similar to Proposition 2.2 in [15] and is omitted here. □

This theorem shows that the FFAPINV algorithm is well-defined for M-matrices. We mention that this is correct for the BFAPINV algorithm, as well (see [15]).

Theorem 3.2. *Let A be an H-matrix and $\hat{A}$ be its comparison matrix. Let also $A^{-1} = ZDW$ and $\hat{A}^{-1} = \hat{Z}\hat{D}\hat{W}$ be the computed inverse in the factored form by the FFINV algorithm for A and $\hat{A}$, respectively. Then*

$$\Big|\frac{1}{d_i}\Big| \geq \frac{1}{\hat{d}_i} > 0.$$

Proof. The elements w_{jl} and z_{lj} may be assumed as rational functions

$$w_{jl} = F_{jl}(a_{11}, \ldots, a_{j-1,n}, d_1, \ldots, d_{j-1}),$$
$$z_{lj} = G_{lj}(a_{11}, \ldots, a_{n,j-1}, d_1, \ldots, d_{j-1}).$$

In fact, w_{jl} (z_{lj}) is a rational function of the first $j-1$ columns ($j-1$ rows) of A and the first $j-1$ diagonal entries of D. In the same way, $\hat{w}_{jl}$ ($\hat{z}_{lj}$) is a rational function F_{jl} (G_{lj}) of the first $j-1$ columns ($j-1$ rows) of $\hat{A}$ and the first $j-1$ diagonal entries of $\hat{D}$. Let us also assume that

$$\tilde{w}_{jl} = F_{jl}(\hat{a}_{11}, \ldots, \hat{a}_{j-1,n}, |d_1|, \ldots, |d_{j-1}|),$$
$$\tilde{z}_{lj} = G_{lj}(\hat{a}_{11}, \ldots, \hat{a}_{n,j-1}, |d_1|, \ldots, |d_{j-1}|).$$

This means that $\tilde{w}_{jl}$ and $\tilde{z}_{lj}$ are computed similarly to $\hat{w}_{jl}$ and $\hat{z}_{lj}$, with $|d_1|, \ldots, |d_{j-1}|$ instead of $d_1, \ldots, d_{j-1}$. By induction, we prove that

$$\begin{cases} \text{a)} & |\frac{1}{d_k}| \geq \frac{1}{\hat{d}_k}, \\ \text{b)} & \hat{w}_{kt} \geq \tilde{w}_{kt} \geq 0, \\ \text{c)} & \hat{l}_t^{(k)} \leq \tilde{l}_t^{(k)} \leq 0, \\ \text{d)} & \hat{z}_{tk} \geq \tilde{z}_{tk} \geq 0, \\ \text{e)} & \hat{u}_t^{(k)} \leq \tilde{u}_t^{(k)} \leq 0, \end{cases} \tag{3.1}$$

for $k = 1, \ldots, n$ and $t \leq k-1$. Note that $\hat{l}_t^{(k)}$, $\tilde{l}_t^{(k)}$, $\tilde{u}_t^{(k)}$ and $\hat{u}_t^{(k)}$ are defined similarly to $\hat{w}_{jl}$ and $\tilde{w}_{jl}$. For $k = 1$, there is nothing to prove. Now, let all of these relations hold for every $k \leq j-1$. We show that all of them are correct for $k = j$, as well. From step 4 of Algorithm 2, for every $t \leq j-1$, we have

$$\hat{l}_t^{(j)} = \hat{a}_{jt} + \sum_{i=1}^{t-1} \hat{a}_{ji}\hat{z}_{it}.$$

From the hypothesis for every $t \leq j-1$, we have $\hat{z}_{it} \geq \tilde{z}_{it} \geq 0$. Therefore,

$$\hat{l}_t^{(j)} = \hat{a}_{jt} + \textstyle\sum_{i=1}^{t-1} \hat{a}_{ji}\hat{z}_{it} \leq \hat{a}_{jt} + \sum_{i=1}^{t-1} \hat{a}_{ji}\tilde{z}_{it} = \tilde{l}_t^{(j)} \leq 0. \tag{3.2}$$

Also, from steps 7–9 of Algorithm 2, we have

$$\hat{w}_{jt} = -\hat{l}_t^{(j)}\hat{d}_t - \sum_{i=t+1}^{j-1} \hat{l}_i^{(j)}\hat{d}_i\hat{w}_{it}, \qquad t = 1, \ldots, j-1.$$

From (3.2) and the hypothesis, for every $k \leq j-1$, we have

$$\begin{aligned} &\hat{d}_k \geq |d_k| \geq 0 \Rightarrow -\hat{l}_k^{(j)}\hat{d}_k \geq -\tilde{l}_k^{(j)}|d_k|, \\ &\hat{w}_{kt} \geq \tilde{w}_{kt} \geq 0 \Rightarrow -\hat{l}_k^{(j)}\hat{d}_k\hat{w}_{kt} \geq -\tilde{l}_k^{(j)}|d_k|\tilde{w}_{kt}, \qquad t < k. \end{aligned}$$

Hence, we conclude that

$$\hat{w}_{jt} = -\hat{l}_t^{(j)}\hat{d}_t - \sum_{i=t+1}^{j-1} \hat{l}_i^{(j)}\hat{d}_i\hat{w}_{it} \geq -\tilde{l}_t^{(j)}|d_t| - \sum_{i=t+1}^{j-1} \tilde{l}_i^{(j)}|d_i|\tilde{w}_{it} = \tilde{w}_{jt} \geq 0.$$

In the same way, it can be verified that

$$\begin{aligned} \hat{u}_t^{(j)} &= \hat{a}_{tj} + \sum_{i=1}^{t-1} \hat{a}_{ij}\hat{w}_{ti} \leq \hat{a}_{tj} + \sum_{i=1}^{t-1} \hat{a}_{ij}\tilde{w}_{tj} = \tilde{u}_t^{(j)} \leq 0, \\ \hat{z}_{tj} &= -\hat{u}_t^{(j)}\hat{d}_t - \sum_{i=t+1}^{j-1} \hat{u}_i^{(j)}\hat{d}_i\hat{z}_{ti} \geq -\tilde{u}_t^{(j)}|d_t| - \sum_{i=t+1}^{j-1} \tilde{u}_i^{(j)}|d_i|\tilde{z}_{ti} = \tilde{z}_{jt} \geq 0. \end{aligned}$$

Now, we consider two cases $a_{jj} > 0$ and $a_{jj} < 0$. If $a_{jj} > 0$, then from $\hat{w}_{jt} \geq \tilde{w}_{jt} \geq 0$ and $\hat{a}_{tj} \leq 0$, we conclude that

$$\frac{1}{\hat{d}_j} = \hat{a}_{jj} + \sum_{t=1}^{j-1} \hat{a}_{tj}\hat{w}_{jt} \leq \hat{a}_{jj} + \sum_{t=1}^{j-1} \hat{a}_{tj}\tilde{w}_{jt}.$$

Consider the polynomials of w_{jt} and $\tilde{w}_{jt}$. It is easy to see that the corresponding terms of these polynomials have the same absolute values. In other words, they may differ only by the sign. On the other hand, every term of $\sum_{t=1}^{j-1} \hat{a}_{tj}\tilde{w}_{jt}$, when considered as a polynomial in elements of $\hat{A}$, is nonpositive, since all terms of $\tilde{w}$ are nonnegative. Therefore, it is less than or equal to $\sum_{t=1}^{j-1} a_{tj}w_{jt}$. Since, it is enough that one of its terms to be nonnegative. Putting these results together indicates that

$$\frac{1}{\hat{d}_j} \leq \hat{a}_{jj} + \sum_{t=1}^{j-1} \hat{a}_{tj}\tilde{w}_{jt} \leq a_{jj} + \sum_{t=1}^{j-1} a_{tj}w_{jt} = \frac{1}{d_j}. \tag{3.3}$$

If $a_{jj} < 0$, then from $\hat{w}_{jt} \geq \tilde{w}_{jt} \geq 0$ and $\hat{a}_{tj} \leq 0$, we have

$$-\frac{1}{\hat{d}_j} = -\hat{a}_{jj} - \sum_{t=1}^{j-1} \hat{a}_{tj}\hat{w}_{jt} \geq -\hat{a}_{jj} - \sum_{t=1}^{j-1} \hat{a}_{tj}\tilde{w}_{jt}.$$

Similar to the case $a_{jj} > 0$, one can conclude that

$$-\frac{1}{\hat{d}_j} \geq -\hat{a}_{jj} - \sum_{t=1}^{j-1} \hat{a}_{tj}\tilde{w}_{jt} \geq -\hat{a}_{jj} + \sum_{t=1}^{j-1} a_{tj}w_{jt} = a_{jj} + \sum_{t=1}^{j-1} a_{tj}w_{jt} = \frac{1}{d_j}. \tag{3.4}$$

Now from Eqs. (3.3) and (3.4), we have

$$\Big|\frac{1}{d_j}\Big| \geq \frac{1}{\hat{d}_j}.$$

This together with Theorem 3.1 gives the desired result. □

Theorem 3.3. *Let A be a H-matrix and $\hat{A}$ be its comparison matrix. Let also $A^{-1} \approx ZDW$ and $\hat{A}^{-1} \approx \hat{Z}\hat{D}\hat{W}$ be the factored approximate inverse computed by the FFAPINV algorithm for A and $\hat{A}$, respectively. Then*

$$\Big|\frac{1}{d_i}\Big| \geq \frac{1}{\hat{d}_i} > 0.$$

Proof. The proof is quite similar to the proof of the previous theorem. Indeed, the proof can be done by induction and noting that all of the inequalities have been obtained by comparing both sides of the inequalities term-by-term. □

From this theorem, the following corollary can be easily concluded.

Corollary 3.4. *Let A be an H-matrix and $D = \mathrm{diag}(d_1, \ldots, d_n)$ be the diagonal matrix computed by Algorithm 1. Then, the sign of a_{jj} and d_j are the same.*

4. AN ILU PRECONDITIONER BASED ON THE FFAPINV ALGORITHM

Let W and Z be the inverse factors of A in Eq. (2.1). Also suppose that $L := W^{-1}$ and $U := Z^{-1}$. It is easy to verify that $A = LD^{-1}U$ is the LDU factorization of A and for $i \leq j$

$$L_{ji} = \beta_i^{(j)}, \qquad U_{ij} = \alpha_i^{(j)},$$

in which $\alpha_i^{(j)}$ and $\beta_i^{(j)}$ are computed in steps 5 and 13 of Algorithm 2. By the above discussion we propose the next algorithm that computes an ILU factorization of A as a by-product of the FFAPINV process. We term this algorithm ILUFF (refer to an ILU preconditioning based on the FFAPINV algorithm). The algorithm is as follows:

Algorithm 3. ILUFF algorithm

1. Set $L = U = I_{n\times n}$, $z_1 := e_1$, $w_1 := e_1^T$ and $d_1 := \frac{1}{a_{11}}$
2. For $j = 2, \ldots, n$, Do
3. $\quad z_j := e_j; \quad w_j := e_j^T$
4. $\quad$ For $i = 1, \ldots, j-1$, Do
5. $\quad\quad U_{ij} := d_i w_i A_{*j}$

6. If $|U_{ij}| > \tau$, then $z_j := z_j - U_{ij} z_i$
7. Drop entries of z_j whose absolute values are smaller than τ
8. EndDo
9. For $i = 1, \ldots, j-1$, Do
10. $L_{ji} := d_i A_{j*} z_i$
11. If $|L_{ji}| > \tau$, then $w_j := w_j - L_{ji} w_i$
12. Drop entries of w_j whose absolute values are smaller than τ
13. EndDo
14. $d_j := \frac{1}{w_j A_{*j}}$
15. EndDo
16. Return $L = (L_{ji})$, $U = (U_{ij})$ and $D = \mathrm{diag}(d_1, d_2, \ldots, d_n)$ $(A \approx LD^{-1}U)$

Some consideration can be given here. Obviously, when the matrix A is symmetric then $W = Z^T$ and the computations will be halved. In the case that the matrix A is symmetric positive definite then we have

$$d_j = \frac{1}{z_j^T A z_j} > 0,$$

and the algorithm is free from breakdown. This is true for nonsymmetric positive definite matrices, as well (see [31]). Therefore, if the matrix A is positive definite (symmetric or nonsymmetric) then we can use

$$d_j := \frac{1}{z_j^T A z_j},$$

in step 14 of Algorithm 3.

5. NUMERICAL EXPERIMENTS

In this section, we have used the ILUFF as the right preconditioner to solve the linear system of equations (1.1) with GMRES(50) method. The ILUFF code is written in Fortran 77. But the GMRES(50) code in the Sparskit package [24] has been used. In the first part of the numerical experiments we used 38 nonsymmetric test matrices. All of these matrices have been taken from the University of Florida Sparse Matrix Collection [9] and none of them are positive definite. In all the experiments whenever a zero pivot has occurred, then we have replaced the zero by the square root of the machine precision. The machine, we used for the experiments, has one quad Intel(R) CPU and 8 GB of RAM memory. The initial guess for the iterative solver was always a zero vector. The stopping criterion used was

$$\frac{\|r_k\|_2}{\|r_0\|_2} < 10^{-10}, \tag{5.1}$$

where r_k is the residual of the unpreconditioned system in the kth iterate. We have considered the exact solution as the vector $e = (1, \cdots, 1)^T$ and the vector $b = Ae$.

We have used the Multilevel Nested Dissection reordering as a preprocessing [10]. To implement theILUFF algorithm, it is clear that matrix A should be accessed row-wise and column-wise. In our implementations, we have used just the Compressed Sparse Row (CSR) format of storage [26] to traverse A row-wise. For the column-wise traverse of A, we have used the linked lists [16].

In Table 1, properties of test matrices and the convergence results of the GMRES(50) method without preconditioning have been reported. In this table, n and nnz are the dimension and the number of nonzero entries of the matrix, respectively. Its stands for the number of iterations and $Time$ is the iteration time which has been computed by the *dtime* command. The times are in seconds. In this table, a + symbol means that the stopping criterion has not been satisfied after 10,000 number of iterations.

Table 2, includes properties of the ILUFF preconditioner and the results of a right preconditioned GMRES(50). Considering Algorithm 3, τ is the tolerance parameter to drop entries of L, U and W, Z factors. In this table, the parameter τ has been set to 0.1 for all the test matrices. Suppose that $nnz(A)$, $nnz(L)$ and $nnz(U)$ are the number of nonzero entries of matrix A and L and U factors of the ILUFF preconditioner. In the implementation of the ILUFF preconditioner, we have merged the D factor into the U factor. Therefore, parameter *density* in Table 2, is defined as:

$$density = \frac{nnz(L) + nnz(U)}{nnz(A)}.$$

In this table, $Ptime$ stands for the preconditioning time and $It-Time$ is the iteration time to solve the preconditioned system. $Ttime$ is defined as the sum of $Ptime$ and $It - Time$. In this table, Its is the number of iterations of GMRES(50) to solve the right preconditioned system. Numerical results presented in Table 2, show that the proposed preconditioner greatly reduces the time and iterations for convergence.

For the second part of the numerical experiments we consider the matrix *atmosmodj*. This matrix belongs to the *Bourchtein* Group of matrix collections [9]. Dimension of this matrix is $1,270,432$ and it has $8,814,880$ number of nonzero entries. Consider system (1.1) when the coefficient matrix is *atmosmodj*. Also suppose that $b = Ae$ in which $e = (1, \cdots, 1)^T$ is the exact solution. We term this artificial system as the *atmosmodj* system. Without preconditioning, GMRES(50) method for this system converges in 1312 number of iterations in about 447.66 seconds. In Table 3, we have reported the results of thr ILUFF preconditioner and the right preconditioned GMRES(50) method for this system. In this table, *density*, $Ptime$, $It-Time$, $Ttime$ and Its have the same meaning as in Table 2. All the $Ttime$ and Its in this table are less than numbers 447.66 and 1312, respectively.

In Figure 1, we have drawn four graphs related to the *atmosmodj* system. In this figure, we take an in-depth look at the results of Table 3. The graph with solid line is devoted to the case that GMRES(50) method without preconditioning is applied to solve the *atmosmodj* system. This graph gives the logarithm of the fraction $\frac{\|r_k\|_2}{\|r_0\|_2}$ for each iterate of GMRES(50). The three other graphs, illustrated by the dashed, dotted and dashed-dotted lines, give the above mentioned logarithm for each iterate x_k of the right preconditioned GMRES(50) method.

Table 1. Test matrices properties together with results of GMRES(50) without preconditioning

Group/Matrix	n	nnz	Time	Its
HB/fs_ 183_ 1	183	998	0.00	38
HB/fs_ 183_ 6	183	1000	0.01	36
Simon/raefsky1	3242	293 409	2.76	3588
Simon/raefsky2	3242	293 551	3.65	4790
Simon/raefsky5	6316	167 178	0.24	306
Simon/raefsky6	3402	130 371	0.68	1385
Muite/Chebyshev3	4101	36 879	+	+
Oberwolfach/flowmeter5	9669	67 391	+	+
Rajat/rajat03	7602	32 653	+	+
HB/sherman3	5005	20 033	+	+
Hamm/memplus	17 758	99 147	7.22	3878
FEMLAB/poisson3Da	13 514	352 762	0.86	342
Botonakis/FEM_ 3D_ thermal1	17 880	430 740	0.88	283
FEMLAB/poisson3Db	13 514	352 762	12.98	620
Oberwolfach/chipcool0	20 082	281 150	+	+
Oberwolfach/chipcool1	20 082	281 150	+	+
Averous/epb1	14 734	95 053	2.08	1432
Averous/epb2	25 228	175 028	2.81	908
Wang/wang3	26 064	177 168	2.34	803
Wang/wang4	26 068	177 196	+	+
IBM_ Austin/coupled	11 341	97 193	+	+
Simon/venkat01	62 424	1 717 792	+	+
Sandia/ASIC_ 100ks	99 190	578 890	23.85	1887
Hamm/hcircuit	105 676	513 072	+	+
Norris/lung2	109 460	492 564	+	+
IBM_ EDA/dc1	116 835	766 396	+	+
IBM_ EDA/dc2	116 835	766 396	+	+
IBM_ EDA/dc3	116 835	766 396	+	+
IBM_ EDA/trans4	116 835	749 800	+	+
IBM_ EDA/trans5	116 835	749 800	+	+
Botonakis/FEM_ 3D_ thermal2	147 900	3 489 300	18.32	652
QLi/crashbasis	160 000	1 750 416	10.09	437
FEMLAB/stomach	213 360	3 021 648	11.64	344
Sandia/ASIC_ 320ks	321 671	1 316 085	10.39	201
Sandia/ASIC_ 680ks	682 712	1 693 767	13.72	84
Bourchtein/atmosmodd	1 270 432	8 814 880	241.92	707
Bourchtein/atmosmodl	1 489 752	10 319 760	166.14	415

Table 2. Properties of ILUFF preconditioner with $\tau = 0.1$ and right preconditioned GMRES(50) method

Matrix	density	Ptime	It-Time	Ttime	Its
fs_ 183_ 1	0.55	0.81	0.00	0.81	10
fs_ 183_ 6	0.54	0.73	0.00	0.73	10
raefsky1	0.05	0.75	1.03	1.78	1092
raefsky2	0.08	0.75	1.45	2.20	1419
raefsky5	0.20	0.75	0.01	0.76	11
raefsky6	0.15	0.75	0.01	0.76	12
Chebyshev3	0.33	0.75	0.09	0.84	245
flowmeter5	0.74	0.76	2.37	3.13	2106
rajat03	0.78	0.82	0.25	1.07	390
sherman3	0.83	0.73	0.82	1.55	1747
memplus	0.39	0.77	0.89	1.66	376
poisson3Da	0.18	0.75	0.55	1.30	180
FEM_ 3D_ thermal1	0.22	0.76	0.23	0.99	53
poisson3Db	0.18	1.06	9.40	10.46	395
chipcool0	0.35	0.78	1.22	2.00	321
chipcool1	0.35	0.75	1.70	2.45	446
epb1	0.78	0.73	1.11	1.84	547
epb2	0.57	0.75	0.83	1.58	209
wang3	0.85	0.76	0.84	1.60	201
wang4	0.55	0.80	1.13	1.93	279
coupled	0.48	0.81	0.21	1.02	149
venkat01	0.34	1.07	1.67	2.74	90
ASIC_ 100ks	0.79	1.01	0.35	1.36	23
hcircuit	0.76	0.99	8.95	9.94	513
lung2	1.03	1.00	5.67	6.67	304
dc1	0.65	35.26	4.75	40.01	234
dc2	0.64	34.06	2.97	37.03	143
dc3	0.64	33.75	8.64	42.39	451
trans4	0.62	21.95	2.36	24.31	128
trans5	0.63	21.80	7.52	29.32	397
FEM_ 3D_ thermal2	0.22	1.19	1.41	2.60	41
crashbasis	0.58	1.19	1.82	3.01	59
stomach	0.22	1.26	2.79	4.05	72
ASIC_ 320ks	0.66	1.45	4.02	5.47	71
ASIC_ 680ks	0.61	1.97	0.68	2.65	6
Bourchtein/atmosmodd	0.64	4.02	200.66	204.68	503
Bourchtein/atmosmodl	0.84	5.14	100.32	105.46	209

Table 3. Properties of ILUFF preconditioner and right preconditioned GMRES(50) for the matrix atmosmodj

τ	density	Ptime	It-Time	Ttime	Its
0.1	0.64	4.04	240.81	244.85	603
0.05	0.76	4.42	232.66	237.08	561
0.01	1.04	5.17	198.06	203.23	464

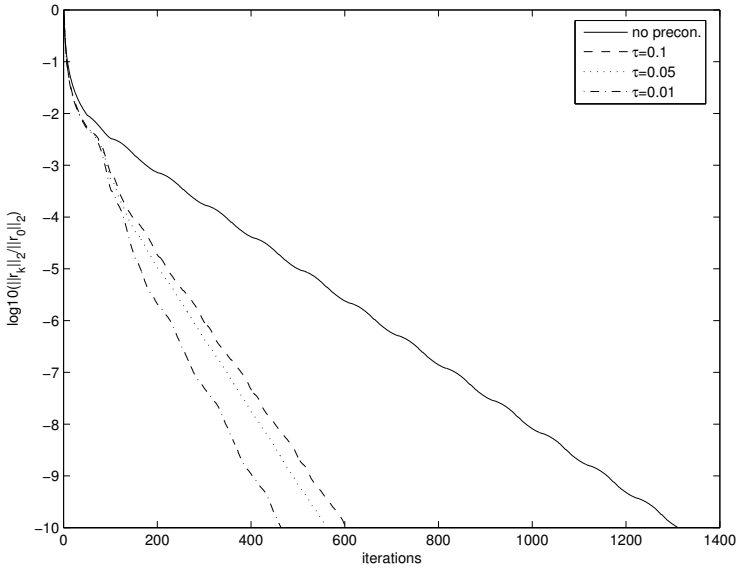

Fig. 1. *Effect of different drop parameters for the ILUFF preconditioner of the atmosmodj system*

For these three graphs, the right preconditioner is the ILUFF and has been computed with τ equal to 0.1, 0.05 and 0.01. The shape of the graphs first indicates that the ILUFF preconditioner is a good tool to decrease the number of iterations of the GMRES(50) method, applied for the *atmosmodj* system. It also shows the fact that the parameter τ gives the better quality of ILUFF preconditioner for the *atmosmodj* system.

6. CONCLUSION

In this paper, we have proposed an ILU preconditioner based on the Forward FAPINV algorithm say ILUFF which is free from breakdown for nonsymmetric positive definite and H-matrices. Numerical results presented in this paper show that the new preconditioner is very robust and effective.

Acknowledgments
The authors would like to thank the anonymous referee for his/her helpful comments and suggestions which improved the quality of the paper.

REFERENCES

[1] M. Benzi, *Preconditioning techniques for large linear systems: A survey*, J. of Computational Physics **182** (2002), 418–477.

[2] M. Benzi, J.K. Cullum, M. Tuma, C.D. Meyer, *Robust approximate inverse preconditioning for the conjugate gradient method*, SIAM J. Sci. Comput. **22** (2000), 1318–1332.

[3] M. Benzi, W.D. Joubert, G. Mateescu, *Numercal experiments with parallel orderings for ILU preconditioners*, ETNA **8** (1999), 88–114.

[4] M. Benzi, C.D. Meyer, M. Tuma, *A sparse approximate inverse preconditioner for the conjugate gradient method*, SIAM J. Sci. Comput. **17** (1996), 1135–1149.

[5] M. Benzi, D.B. Szyld, A. van Duin, *Ordering for incomplete factorization preconditioning of nonsymmetric problems*, SIAM J. Sci. Comput. **20** (1999), 1652–1670.

[6] M. Benzi, M. Tuma, *A sparse approximate inverse preconditioner for nonsymmetric linear systems*, SIAM J. Sci. Comput. **19** (1998), 968–994.

[7] M. Benzi, M. Tuma, *A comparative study of sparse approximate inverse preconditioners*, Appl. Numer. Math. **30** (1999), 305–340.

[8] M. Benzi, M. Tuma, *A robust incomplete factorization preconditioner for positive definite matrices*, Numer. Linear Algebra Appl. **10** (2003), 385–400.

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

[10] G. Karypis, V. Kumar, *Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs*, SIAM J. Sci. Comput. **20** (1998) 1, 359–392.

[11] S.A. Kharchenko, L.Yu. Kolotilina, A.A. Nikishin, A.Yu. Yeremin, *A robust AINV-type method for constructing sparse approximate inverse preconditioners in factored form*, Numer. Linear Algebra Appl. **8** (2001), 165–179.

[12] L.Y. Kolotilina, A.Y. Yeremin, *Factorized sparse approximate inverse preconditioning I. Theory*, SIAM J. Matrix Anal. Appl. **14** (1993), 45–58.

[13] L.Y. Kolotilina, A.Y. Yeremin, *Factorized sparse approximate inverse preconditioning II: Solution of 3D FE systems on massively parallel computers*, Int. J. High Speed Comput. **7** (1995), 191–215.

[14] E.-J. Lee, J. Zhang, *A two-phase preconditioning strategy of sparse approximate inverse for indefinite matrices*, Technical Report No. 476-07, Department of Computer Science, University of Kentuky, Lexington, KY, 2007.

[15] E.-J. Lee, J. Zhang, *Fatored approximate inverse preonditioners with dynamic sparsity patterns*, Technial Report No. 488-07, Department of Computer Science, University of Kentuky, Lexington, KY, 2007.

[16] N. Li, Y. Saad, E. Chow, *Crout version of ILU for general sparse matrices*, SIAM J. Sci. Comput. **25** (2003) 2, 716–728.

[17] J.-G. Luo, *A new class of decomposition for symmetric systems*, Mechanics Research Communications **19** (1992), 159–166.

[18] J.-G. Luo, *An incomplete inverse as a preconditioner for the conjugate gradient method*, Comput. Math. Appl. **25** (1993), 73–79.

[19] J.-G. Luo, *A new class of decomposition for inverting asymmetric and indefinite matrices*, Comput. Math. Appl. **25** (1993), 95–104.

[20] T. Manteuffel, *An incomplete factorization technique for positive definite linear systems*, Math. Comput. **34** (1980), 473–497.

[21] J.A. Meijerink, H.A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comput. **31** (1977), 148–162.

[22] G. Meurant, *The block preconditioned conjugate gradient method on vector computers*, BIT **24** (1984), 623–633.

[23] M. Rezghi, S.M. Hosseini, *An ILU preconditioner for nonsymmetric positive definite matrices by using the conjugate Gram-Schmidt process*, J. Comput. Appl. Math. **188** (2006), 150–164.

[24] Y. Saad, *Sparskit and sparse examples*, NA Digest (1994), http://www-users.cs.umn.edu/~saad/software, Accessed 2010.

[25] Y. Saad, *ILUT: A dual theshold incomplete LU preconditioner*, Numer. Linear Algebra Appl. **1** (1994), 387–402.

[26] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Press, New York, 1995.

[27] Y. Saad, *ILUM: A multi-elimination ILU preconditioner for general sparse matrices*, SIAM J. Sci. Comput. **174** (1996), 830–847.

[28] Y. Saad, M.H. Schultz, *GMRES: A generalized minimal residual algorithm for nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput. **7** (1986), 856–869.

[29] Y. Saad, J. Zhang, *BILUM: Block versions of multi-elimination and multilevel ILU preconditioner for general linear sparse systems*, SIAM J. Sci. Comput. **20** (1999), 2103–2121.

[30] Y. Saad, J. Zhang, *BILUTM: a domain-based multilevel block ILUT preconditioner for general sparse matrices*, SIAM J. Matrix Anal. Appl. **21** (1999), 279–299.

[31] D.K. Salkuyeh, *A sparse approximate inverse preconditioner for nonsymmetric positive definite matrices*, Journal of Applied Mathematics and Informatics **28** (2010), 1131–1141.

[32] 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. Statist. Comput. **12** (1992), 631–644.

[33] J. Zhang, *A procedure for computing factored approximate inverse*, M.Sc. Dissertation, Department of Computer Science, University of Kentucky, 1999.

[34] J. Zhang, *A sparse approximate inverse technique for parallel preconditioning of general sparse matrices*, Appl. Math. Comput. **130** (2002), 63–85.

Davod Khojasteh Salkuyeh
khojasteh@guilan.ac.ir

University of Guilan
Faculty of Mathematical Sciences
Rasht, Iran

Amin Rafiei
rafiei.am@gmail.com, a.rafiei@hsu.ac.ir

Hakim Sabzevari University
Department of Mathematics
P.O. Box. 397, Sabzevar, Iran

Hadi Roohani
hadiroohani61@gmail.com

Malek Ashtar University of Technology
Department of Mathematics
Shahin Shar, Isfahan, P.O. Box 83145-115, Iran

Received: April 25, 2014.
Revised: May 27, 2014.
Accepted: June 7, 2014.