Computers and Mathematics with Applications 64 (2012) 2682–2694

Contents lists available at SciVerse ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

A complete pivoting strategy for the right-looking Robust Incomplete Factorization preconditioner

A. Rafiei

Department of Applied Mathematics. Hakim Sabzevari University, Sabzevar, Iran

ARTICLE INFO

Article history:
Received 10 April 2012
Received in revised form 31 July 2012
Accepted 5 August 2012

Keywords:
ILU factorization
Right-looking *AINV* preconditioner
Right-looking *RIF* preconditioner
Gaussian Elimination process
Krylov subspace methods
Pivoting

ABSTRACT

In this paper, we use a complete pivoting strategy for the right-looking version of Robust Incomplete Factorization preconditioner (Benzi and Tůma, 2003) [7]. The new preconditioner has been used as the right preconditioner for several linear systems and its effectiveness has been studied.

1. Introduction

Consider the linear system of equations of the form

$$Ax = b, \tag{1}$$

where the coefficient matrix $A \in \mathbb{R}^{n\times n}$ is nonsingular, large, sparse and nonsymmetric and also $x, b \in \mathbb{R}^n$. Krylov subspace methods can be used to solve this system [1]. An implicit preconditioner M for system (1) is an approximation of matrix A, i.e., $M \approx A$. If M is a good approximation of A, then it can be used as the right preconditioner for system (1). In this case, instead of solving system (1) it is better to solve the right preconditioned system

$$AM^{-1}u = b; \quad M^{-1}u = x,$$

by the Krylov subspace methods.

Suppose that there is the factorization

$$A = LDU, \tag{2}$$

for matrix A, where L and U^T are unit lower triangular matrices and D is a diagonal matrix. Also suppose that dropping be applied on L, U and D. Then, matrix M which is

$$A \approx M = LDU,$$

is an implicit preconditioner for system (1). This preconditioner is also termed as an *ILU* preconditioner. One can use IJK version of Gaussian Elimination to compute the *ILU* preconditioner M [2]. In this case, the computation of L and U

E-mail addresses: rafiei.am@gmail.com, a.rafiei@sttu.ac.ir.

0898-1221/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2012.08.001

has been interlaced together. There are four ways to update the Schur-Complement matrix in this version of Gaussian Elimination [3].

An explicit preconditioner M for system (1) is an approximation of matrix A^{-1}, i.e., $M \approx A^{-1}$. In the same way as the implicit preconditioner, if the explicit preconditioner M is a good approximation of A^{-1}, then it is better to solve system

$$AMu = b; \quad Mu = x,$$

by the Krylov subspace methods. The most well-known explicit preconditioner is the *AINV* preconditioner [4]. This preconditioner has three factors in the form

$$A^{-1} \approx M = ZD^{-1}W^T, \tag{3}$$

where Z and W are unit upper triangular matrices and D is a diagonal matrix. There are two left and right-looking versions for this preconditioner.

There are two possibilities to compute Z and W factors in (3). The computation of Z can be done independent or dependent of W [3]. There is no need to work with the Schur-Complement matrix to compute the *AINV* preconditioner. But, instead one should do the rank-one updates.

In [3], Bollhoefer and Saad could find a relation to obtain the Schur-Complement matrix, which is computed through the IJK version of Gaussian Elimination. In this relation, the Schur-Complement matrix is gained by the computed factors of the *AINV* preconditioner. This was the essential key to extend the complete pivoting strategy to the *AINV* preconditioner [5]. In [6], Tůma has also presented another work on *AINV* with complete pivoting.

In 2003, Benzi and Tůma, introduced a new *ILU* preconditioner for symmetric positive definite matrices. This preconditioner is computed as a by-product of the *AINV* preconditioner and is termed Robust Incomplete Factorization or *RIF* [7]. There are two left and right-looking versions for this preconditioner.

In this paper, we present a complete pivoting strategy for right-looking version of the *RIF* preconditioner. To test effectiveness of such a pivoting, we have generated several linear systems. The coefficient matrices are taken from University of Florida Sparse Matrix Collection [8]. Then, we have computed the right-looking version of *RIF* with pivoting for such systems and have solved the right preconditioned linear systems by the *GMRES*(30), Bicgstab and *TFQMR* Krylov subspace methods [1].

In this paper, $A_{:,i}$ and $A_{i,:}$ refer to the i-th column and the i-th row of matrix A, respectively. Notation $A_{i:j,k}$ indicates entries of the k-th column of matrix A whose row indices are between i and j. We also use notation $A_{k,i:j}$ to indicate entries of the k-th row of matrix A whose column indices are between i and j.

In Sections 2 and 3 of this paper, we review a complete pivoting strategy for IJK version of Gaussian Elimination and for right-looking version of the *AINV* preconditioner, respectively. In Section 4, we have presented a complete pivoting strategy for the *RIF* preconditioner. Section 5 has been devoted to the numerical experiments.

2. Complete pivoting strategy for IJK version of Gaussian Elimination

Algorithm 1, is the IJK version of Gaussian Elimination. There are three nested i, j and k loops in this algorithm.

Algorithm 1 (*IJK* **version of Gaussian Elimination process with dropping)**

Input: $A = (a_{ij}) \in \mathbb{R}^{n\times n}$, τ_l and $\tau_u \in (0, 1)$ be drop tolerances for L and U matrices.
Output: $A \approx LDU$.

1. $L = U = I_n$, $S^{(0)} = A$.
2. **for** $i = 1$ to n **do**
3. $\quad p_i^{(i-1)} = (S^{(i-1)})_{ii}$, $\quad q_i^{(i-1)} = (S^{(i-1)})_{ii}$
4. $\quad$ **for** $j = i + 1$ to n **do**
5. $\quad\quad p_j^{(i-1)} = \frac{(S^{(i-1)})_{ji}}{p_i^{(i-1)}}$, $\quad q_j^{(i-1)} = \frac{(S^{(i-1)})_{ij}}{q_i^{(i-1)}}$.
6. $\quad\quad L_{ji} = p_j^{(i-1)}$, $\quad U_{ij} = q_j^{(i-1)}$.
7. $\quad\quad$ apply dropping rule to L_{ji} and to U_{ij} if their absolute values are less than τ_u and τ_l.
8. $\quad\quad$ **for** $k = i + 1$ to n **do**
9. $\quad\quad\quad (S^{(i)})_{jk} = (S^{(i-1)})_{jk} - L_{ji}d_{ii}U_{ik}$.
10. $\quad\quad$ **end for**
11. $\quad$ **end for**
12. $\quad d_{ii} = p_i^{(i-1)}$
13. **end for**
14. Return $L = (L_{ij})_{1\le i,j\le n}$, $U = (U_{ij})_{1\le i,j\le n}$ and $D = diag(d_{ii})_{1\le i\le n}$.

Suppose that no dropping be applied in Algorithm 1. Then, this algorithm computes the *LDU* decomposition of matrix *A*. At the first step of this algorithm, matrix *A* is factorized as

$$A = \begin{bmatrix} a_{11} & f \\ e & C \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ g & I \end{bmatrix} \begin{bmatrix} \delta & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} 1 & h \\ 0 & I \end{bmatrix}. \tag{4}$$

In relation (4), $\delta = a_{11}$ is the (1, 1) entry of matrix A and vectors $g, e \in \mathbb{R}^{(n-1)\times 1}$ and $f, h \in \mathbb{R}^{1\times(n-1)}$ satisfy the relations

$$g\delta = e \in \mathbb{R}^{(n-1)\times 1}, \qquad \delta h = f \in \mathbb{R}^{1\times(n-1)}.$$

Matrix *S* is called Schur-Complement matrix. At the next step of Algorithm 1, matrix *S* will be factorized as in (4) and the same process will be repeated recursively in the other steps of this algorithm. Therefore, the *LDU* factorization of matrix A in (2) will be obtained at the end of this algorithm.

Suppose that dropping be applied in Algorithm 1. Then, vectors $\tilde{h}$ and $\tilde{g}$ are computed which are the approximations of vectors *h* and *g*, respectively. In this case, there are four different versions

$$\begin{aligned} &S \approx C - \tilde{g}\delta\tilde{h}, \\ &S \approx C - \tilde{g}f, \\ &S \approx C - e\tilde{h}, \\ &S \approx C - \tilde{g}f - (e - \tilde{g}\delta)\tilde{h}, \end{aligned} \tag{5}$$

to compute the approximate Schur-Complement matrix *S*. In sparse cases, the Schur-Complement matrix is formed only very exceptionally and one should always work just with vectors. In Algorithm 1, the first version of update in (5) has been used. Therefore, at the end of step *n* of this algorithm, the *ILU* preconditioner $M = LDU \approx A$, will be computed that has the sparser *L* and *U* factors than the case we use second to fourth versions of update in (5).

Because of having the Schur-Complement matrix at any step *i* of Algorithm 1, a complete pivoting strategy can be applied to this algorithm. Algorithm 2, applies a complete pivoting strategy for IJK version of Gaussian Elimination. This algorithm has also been presented in [2] in a different way.

At the end of step $i-1$ of Algorithm 2, row permutation matrices $\Pi_1, \ldots, \Pi_{i-1}$ and column permutation matrices $\Sigma_1, \ldots, \Sigma_{i-1}$ and matrix $S^{(i-1)}$ are computed. Suppose that vectors $g_k \in \mathbb{R}^{(n-k)\times 1}$ for $1 \le k \le i-1$ are the strict lower triangular parts of the already computed columns 1 to $i-1$ of matrix *L*. Also suppose that vectors $h_k \in \mathbb{R}^{1\times(n-k)}$ for $1 \le k \le i-1$ are the strict upper triangular parts of the already computed rows 1 to $i-1$ of matrix *U*. Then, the following relation

$$\Pi_{i-1}\,\Pi_{i-2}\cdots\Pi_1\,A\,\Sigma_1\cdots\Sigma_{i-2}\,\Sigma_{i-1} \approx \underbrace{\begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & \ddots & & & & \\ & & & 1 & & & \\ g_1 & g_2 & & g_{i-1} & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} d_{11} & & & \\ & \ddots & & \\ & & d_{i-1,i-1} & \\ & & & (S^{(i-1)})_{k,l\ge i} \end{bmatrix}}_{S^{(i-1)}} \underbrace{\begin{bmatrix} 1 & & & h_1 & & \\ & 1 & & h_2 & & \\ & & \ddots & & & \\ & & & 1 & h_{i-1} & \\ & & & & 1 & \\ & & & & & \ddots \\ & & & & & 1 \end{bmatrix}}_{U},$$

holds at the end of step $i-1$ of Algorithm 2.

At the beginning of step *i* of Algorithm 2, *satisfied_p* = *satisfied_q* = *false*. Then, vector $p^{(i-1)} = (p_i^{(i-1)}, \ldots, p_n^{(i-1)})$ is computed which contains the entries of the first column of the Schur-Complement matrix $(S^{(i-1)})_{k,l\ge i}$. Then, the row pivoting criterion

$$|p_i^{(i-1)}| < \alpha |p_k^{(i-1)}| = \alpha \max_{m\ge i} |p_m^{(i-1)}|, \tag{6}$$

is applied to find the row index *k* where $\alpha \in (0, 1]$. The role of α is to control the row pivoting process, since it gives the chance to control the magnitude of the pivot entry. After finding the row index *k*, $m_i = 1$ and matrix $\pi_1^{(i-1)} = I_n$ is produced and we need to interchange submatrices $L(i, 1 : i-1)$ and $L(k, 1 : i-1)$. Next, we need to interchange *i*-th and *k*-th rows of matrix $\pi_1^{(i-1)}$ and elements $p_i^{(i-1)}$ and $p_k^{(i-1)}$ together. Finally, we set

$$S^{(i-1)} = \pi_1^{(i-1)} S^{(i-1)}, \qquad \Pi = \pi_1^{(i-1)} \Pi.$$

The row pivoting is now completed. So, *satisfied_p* is set to *true*.

To apply the complete pivoting strategy, one should also consider the column pivoting strategy and repeat the search for pivot in the first row of the permuted Schur-Complement matrix $(S^{(i-1)})_{k,l\geq i}$. Therefore, at first, the vector $q^{(i-1)} = (q_i^{(i-1)}, \ldots, q_n^{(i-1)})$ is computed which is the first row of the new Schur-Complement matrix $(S^{(i-1)})_{k,l\geq i}$. Then, the column pivoting criterion

$$|q_i^{(i-1)}| < \alpha |q_l^{(i-1)}| = \alpha \max_{m\geq i} |q_m^{(i-1)}|, \tag{7}$$

is used to find the column index l. Parameter α, also controls the column pivoting process and magnitude of the pivot entry. After finding column index l, $n_i = 1$ and matrix $\sigma_1^{(i-1)} = I_n$ is produced and we need to interchange submatrices $U(1 : i-1, i)$ and $U(1 : i-1, l)$. Next, we need to interchange i-th and l-th columns of matrix $\sigma_1^{(i-1)}$ and elements $q_i^{(i-1)}$ and $q_l^{(i-1)}$ together. Finally, we set

$$S^{(i-1)} = S^{(i-1)} \sigma_1^{(i-1)}, \qquad \Sigma = \Sigma \sigma_1^{(i-1)}. \tag{8}$$

The first relation in (8) means that $S^{(i-1)} = \pi_1^{(i-1)} S^{(i-1)} \sigma_1^{(i-1)}$. The column pivoting is now completed. So, *satisfied_ q* is set to *true*. Since the balance of the pivot element has been disturbed then, *satisfied_p* is set to false in line 23 of this algorithm and we emphasize again that for the goal of complete pivoting, one should repeat the search for pivot in the first row of the permuted Schur-Complement matrix.

The process of pivoting will alternate between row interchanges and column interchanges and usually a finite sequence of row permutation matrices $\pi_1^{(i-1)}, \ldots, \pi_{m_i}^{(i-1)}$ and column permutation matrices $\sigma_1^{(i-1)}, \ldots, \sigma_{n_i}^{(i-1)}$ are computed. This means that at the end of the internal while loop of the i-th step of Algorithm 2, matrix $S^{(i-1)}$ is

$$S^{(i-1)} = \pi_{m_i}^{(i-1)} \cdots \pi_2^{(i-1)} \pi_1^{(i-1)} S^{(i-1)} \sigma_1^{(i-1)} \sigma_2^{(i-1)} \cdots \sigma_{n_i}^{(i-1)}.$$

If we define $\Pi_i := \pi_{m_i}^{(i-1)} \cdots \pi_1^{(i-1)}$ and $\Sigma_i := \sigma_1^{(i-1)} \cdots \sigma_{n_i}^{(i-1)}$, then, in line 31 of Algorithm 2 we have $S^{(i-1)} = \Pi_i S^{(i-1)} \Sigma_i$ and we need to define the (i, i) entry of this matrix as the pivot element d_{ii}.

After computing the pivot entry d_{ii}, the other parts of Algorithm 2, is the same as Algorithm 1. This means that at first, vectors $L_{i+1:n,i}$ and $U_{i,i+1:n}$ are computed. Then, the submatrix $(S^{(i-1)})_{k,l\geq i+1}$ should be updated by using the first version in (5). After that, we set the vectors $(S^{(i-1)})_{i+1:n,i}$ and $(S^{(i-1)})_{i,i+1:n}$ equal to zero. At the end of step i of Algorithm 2, matrix $S^{(i-1)}$ is changed to a new matrix which is considered as $S^{(i)}$. Suppose that the new vectors $g_i \in \mathbb{R}^{(n-i)\times 1}$ and $h_i \in \mathbb{R}^{1\times(n-i)}$ be the strict lower triangular and strict upper triangular parts of the i-th column and i-th row of matrices L and U, respectively. This means that $g_i = L_{i+1:n,i}$ and $h_i = U_{i,i+1:n}$. Then, the row permutation matrices $\Pi_1, \ldots, \Pi_i$, column permutation matrices $\Sigma_1, \ldots, \Sigma_i$ and matrices $S^{(i)}$ and A satisfy the relation

$$\Pi_i \Pi_{i-1} \cdots \Pi_1 A \, \Sigma_1 \cdots \Sigma_{i-1} \Sigma_i \approx$$

$$\underbrace{\begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & \ddots & & & & \\ & & & 1 & & & \\ g_1 & g_2 & \cdots & g_{i-1} & 1 & & \\ & & & & g_i & 1 & \\ & & & & & & \ddots \\ & & & & & & & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} d_{11} & & & \\ & \ddots & & \\ & & d_{ii} & \\ & & & (S^{(i)})_{k,l\geq i+1} \end{bmatrix}}_{S^{(i)}} \underbrace{\begin{bmatrix} 1 & & & h_1 & & \\ & 1 & & h_2 & & \\ & & \ddots & \vdots & & \\ & & 1 & h_{i-1} & & \\ & & & 1 & h_i & \\ & & & & 1 & \\ & & & & & \ddots \\ & & & & & & 1 \end{bmatrix}}_{U},$$

at the end of step i of Algorithm 2.

Suppose that matrices Π and Σ be defined as $\Pi := \Pi_{n-1} \cdots \Pi_2 \Pi_1$ and $\Sigma := \Sigma_1 \Sigma_2 \cdots \Sigma_{n-1}$. Then, at the end of step n of this algorithm

$$\Pi A \Sigma \approx M = LDU.$$

Algorithm 2 (*IJK* **version of Gaussian Elimination with complete pivoting)**

Input: $A = (a_{ij}) \in \mathbb{R}^{n\times n}$, τ_l and $\tau_u \in (0, 1)$ be drop tolerances for L and U matrices and prescribe a tolerance $\alpha \in (0, 1]$.
Output: $\Pi\ A\ \Sigma \approx LDU$.

1. $L = U = \Pi = \Sigma = I_n$, $S^{(0)} = A$.
2. **for** $i = 1$ to n **do**
3. $m_i = n_i = 0$.
4. *satisfied_ p* = *false*, *satisfied_ q* = *false*.
5. **while** not *satisfied_ p* **do**
6. **for** $j = i$ to n **do**
7. $p_j^{(i-1)} = e_j^T\ (S^{(i-1)})\ e_i$.
8. **end for**
9. **if** $|p_i^{(i-1)}| < \alpha\ max_{m\geq i}|p_m^{(i-1)}|$ **then**
10. $m_i = m_i + 1, \quad \pi_{m_i}^{(i-1)} = I_n$.
11. *satisfied_ q* = *false*, choose k such that $|p_k^{(i-1)}| = max_{m\geq i}|p_m^{(i-1)}|$.
12. Interchange rows i and k of $L - I$ and $\pi_{m_i}^{(i-1)}$ and elements $p_i^{(i-1)}$ and $p_k^{(i-1)}$.
13. $S^{(i-1)} = \pi_{m_i}^{(i-1)}\ S^{(i-1)}$
14. $\Pi = \pi_{m_i}^{(i-1)}\ \Pi$.
15. **end if**
16. *satisfied_ p* = *true*.
17. **for** $j = i$ to n **do**
18. $q_j^{(i-1)} = e_i^T\ (S^{(i-1)})\ e_j$.
19. **end for**
20. **if** not *satisfied_ q* **then**
21. **if** $|q_i^{(i-1)}| < \alpha\ max_{m\geq i}|q_m^{(i-1)}|$ **then**
22. $n_i = n_i + 1, \quad \sigma_{n_i}^{(i-1)} = I_n$.
23. *satisfied_ p* = *false*, choose l such that $|q_l^{(i-1)}| = max_{m\geq i}|q_m^{(i-1)}|$.
24. Interchange columns i and l of $U - I$ and $\sigma_{n_i}^{(i-1)}$ and elements $q_i^{(i-1)}$ and $q_l^{(i-1)}$.
25. $S^{(i-1)} = S^{(i-1)}\ \sigma_{n_i}^{(i-1)}$.
26. $\Sigma = \Sigma\ \sigma_{n_i}^{(i-1)}$.
27. **end if**
28. **end if**
29. *satisfied_ q* = *true*.
30. **end while**
31. $d_{ii} = (S^{(i-1)})_{ii}$.
32. **for** $j = i + 1$ to n **do**
33. $L_{ji} = \frac{p_j^{(i-1)}}{p_i^{(i-1)}}, \quad U_{ij} = \frac{q_j^{(i-1)}}{q_i^{(i-1)}}$.
34. apply dropping rule to L_{ji} and to U_{ij} if their absolute values are less than τ_l and τ_u.
35. **for** $k = i + 1$ to n **do**
36. $(S^{(i)})_{jk} = (S^{(i-1)})_{jk} - L_{ji}d_{ii}U_{ik}$.
37. **end for**
38. **end for**
39. **end for**
40. Return $L = (L_{ij})_{1\leq i,j\leq n}$, $U = (U_{ij})_{1\leq i,j\leq n}$, $D = diag(d_{ii})_{1\leq i\leq n}$, Π and Σ.

3. Complete pivoting strategy for right-looking version of *AINV*

In 1998, Benzi and Tůma presented the *AINV* preconditioner for general matrices in the form of (3) [4]. Algorithm 3 computes the right-looking version of this preconditioner.

Suppose that no dropping be applied in both Algorithms 1 and 3. Then, both diagonal matrices D generated in two algorithms are equal. Also relations $W^T = L^{-1}$ and $Z = U^{-1}$ hold between matrices W^T, L and Z, U.

There is no Schur-Complement matrix in Algorithm 3 and consequently there is no need to do any update of this matrix. But instead, one should do the rank-one updates in line 6 of this algorithm. Is there any chance to have the Schur-Complement matrix, implicitly in this algorithm? Suppose that

$$W^{(i-1)} = [w_1^{(0)}, w_2^{(1)}, \ldots, w_i^{(i-1)}, \ldots, w_n^{(i-1)}], \qquad Z^{(i-1)} = [z_1^{(0)}, z_2^{(1)}, \ldots, z_i^{(i-1)}, \ldots, z_n^{(i-1)}],$$

are the computed W and Z matrices at the end of step $i-1$ of Algorithm 3. As we explained before, suppose that the submatrix $(S_{kl}^{(i-1)})_{k,l\geq i}$ is the Schur-Complement matrix at the end of step $i-1$ of Algorithm 1. In 2002, Bollhöfer and Saad could show that [5]

$$(S_{kl}^{(i-1)})_{k,l\geq i} = ((w_k^{(i-1)})^T A(z_l^{(i-1)}))_{k,l\geq i}. \tag{9}$$

Therefore, we can work with Algorithm 3 and compute W and Zexplicitly and also we can generate the Schur-Complement matrix $(S_{kl}^{(i-1)})_{k,l\geq i}$, implicitly. Relation (9) is the main key to extend the complete pivoting strategy to the right-looking version of the *AINV* preconditioner. Algorithm 4, computes the right-looking version of the *AINV* preconditioner and uses the complete pivoting strategy.

Algorithm 3 (**Right-looking AINV algorithm**)

Input: $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ and $\tau_w, \tau_z \in (0, 1)$ be drop tolerances.
Output: $A^{-1} \approx ZD^{-1}W^T$.

1. $w_i^{(0)} = e_i,\ \ z_i^{(0)} = e_i,\quad 1 \leq i \leq n.$
2. **for** $i = 1$ to n **do**
3. $\quad p_i^{(i-1)} = (w_i^{(i-1)})^T A z_i^{(i-1)},\quad q_i^{(i-1)} = (z_i^{(i-1)})^T A^T w_i^{(i-1)}$
4. $\quad$ **for** $j = i+1$ to n **do**
5. $\quad\quad p_j^{(i-1)} = \frac{(w_j^{(i-1)})^T A z_i^{(i-1)}}{p_i^{(i-1)}},\quad q_j^{(i-1)} = \frac{(z_j^{(i-1)})^T A^T w_i^{(i-1)}}{q_i^{(i-1)}}.$
6. $\quad\quad w_j^{(i)} = w_j^{(i-1)} - p_j^{(i-1)} w_i^{(i-1)},\quad z_j^{(i)} = z_j^{(i-1)} - q_j^{(i-1)} z_i^{(i-1)}.$
7. $\quad\quad$ for all $l \leq i$ apply dropping rule to $w_{lj}^{(i)}$ and to $z_{lj}^{(i)}$ if their absolute values are less than τ_w and τ_z.
8. $\quad$ **end for**
9. $\quad d_{ii} = p_i^{(i-1)}.$
10. **end for**
11. Return $Z = [z_1^{(0)}, z_2^{(1)}, \cdots, z_n^{(n-1)}]$, $W = [w_1^{(0)}, w_2^{(1)}, \cdots, w_n^{(n-1)}]$ and $D = diag(d_{ii})_{1\leq i\leq n}$.

Suppose that no dropping be applied in both Algorithms 2 and 4. Then, at the end of step $i-1$ of both algorithms, the crucial relation

$$(S_{kl}^{(i-1)})_{k,l\geq i} = ((w_k^{(i-1)})^T(\Pi A \Sigma)(z_l^{(i-1)}))_{k,l\geq i}, \tag{10}$$

holds between the Schur-Complement matrix $(S_{kl}^{(i-1)})_{k,l\geq i}$ and columns of the matrices $W^{(i-1)}$ and $Z^{(i-1)}$ [5]. In (10), matrices Π and Σ are the computed permutation matrices at the end of step $i-1$.

Details and explanations of step i of both Algorithms 2 and 4 are the same, except two main differences. The first difference is that we compute the first row and first column of the Schur-Complement matrix, implicitly in Algorithm 4. But we have this row and this column, explicitly in Algorithm 2. The second difference is in updating the Schur-Complement matrix. In Algorithm 2, this update is done explicitly but in Algorithm 4 it is done, implicitly. We explain these two differences for step i of both algorithms more in-depth.

Suppose that we are at step i of both Algorithms 2 and 4. In lines 7–9 of Algorithm 4, we should compute the first column of the Schur-Complement matrix $(S_{kl}^{(i-1)})_{k,l\geq i}$ by relation $((w_j^{(i-1)})^T(\Pi A\Sigma)z_i^{(i-1)})_{j\geq i}$. But in lines 6–8 of Algorithm 2, this column is at hand. Suppose that after surveying the row pivoting criterion $|p_i^{(i-1)}| < \alpha \max_{m\geq i} |p_m^{(i-1)}|$ in line 9 of Algorithm 2 and in line 10 of Algorithm 4, row index k has been selected. In Algorithm 2, at first, we have to interchange rows i and k of matrix $L-I$. Next, we should compute row permutation matrix $\pi_{m_i}^{(i-1)}$ which is obtained by interchanging the i-th and k-th rows of the identity matrix. After that, matrices $S^{(i-1)}$ and Π should be updated as $S^{(i-1)} = \pi_{m_i}^{(i-1)} S^{(i-1)}$ and $\Pi = \pi_{m_i}^{(i-1)}\Pi$. But in Algorithm 4, at first, we interchange columns i and k of matrix $W-I$. Next, the permutation matrix $\pi_{m_i}^{(i-1)}$ is gained as above. After that, we should only update matrix Π as above and there is no explicit need to update matrix $S^{(i-1)}$.

After the row pivoting, the column pivoting criterion should be checked in both algorithms. For this, we should access the first row of the new permuted Schur-Complement matrix. In Algorithm 2, the first row of this matrix is at hand in lines 17–19. But in Algorithm 4, we should compute this row as $((z_j^{(i-1)})^T(\Pi A\Sigma)^T w_i^{(i-1)})_{j\geq i}$. Suppose that the column pivoting criterion gives column index l in line 21 of both algorithms. In Algorithm 2, at first, we interchange columns i and l of matrix $U-I$. Next, column permutation matrix $\sigma_{n_i}^{(i-1)}$ is computed by interchanging i-th and l-th columns of the identity matrix. After that, matrices $S^{(i-1)}$ and Σ are updated as $S^{(i-1)} = S^{(i-1)}\sigma_{n_i}^{(i-1)}$ and $\Sigma = \Sigma\sigma_{n_i}^{(i-1)}$. But in Algorithm 4, at first, we interchange columns i and l of matrix $Z-I$. Next, the permutation matrix $\sigma_{n_i}^{(i-1)}$ is obtained as above. After that, we should only update matrix Σ as above and again there is no explicit need to update matrix $S^{(i-1)}$.

After the internal while loop of Algorithm 2, the submatrix $(S_{kl}^{(i-1)})_{k,l\geq i+1}$ is updated in lines 35–37. But in Algorithm 4, the columns $w_j^{(i-1)}$ and $z_j^{(i-1)}$, for $j \geq i+1$, of matrices $W^{(i-1)}$ and $Z^{(i-1)}$ are updated in lines 31–35.

Algorithm 4 (Right-looking AINV with pivoting)

Input: Let $A = (a_{ij}) \in \mathbb{R}^{n\times n}$, $\tau_w, \tau_z \in (0, 1)$ be drop tolerances and prescribe a tolerance $\alpha \in (0, 1]$.
Output: $(\Pi A \Sigma)^{-1} \approx ZD^{-1}W^T$.

1. $\Pi = \Sigma = I_n$.
2. $w_i^{(0)} = e_i,\ \ z_i^{(0)} = e_i, \quad 1 \le i \le n$.
3. **for** $i = 1$ to n **do**
4. $\quad m_i = n_i = 0$
5. $\quad$ *satisfied_ p = false*, *satisfied_ q = false*.
6. $\quad$ **while** not *satisfied_ p* **do**
7. $\quad\quad$ **for** $j = i$ to n **do**
8. $\quad\quad\quad p_j^{(i-1)} = (w_j^{(i-1)})^T(\Pi A\Sigma)z_i^{(i-1)}$.
9. $\quad\quad$ **end for**
10. $\quad\quad$ **if** $|p_i^{(i-1)}| < \alpha\ max_{m\ge i}|p_m^{(i-1)}|$ **then**
11. $\quad\quad\quad m_i = m_i + 1, \quad \pi_{m_i}^{(i-1)} = I_n$.
12. $\quad\quad\quad$ *satisfied_ q = false* , choose k such that $|p_k^{(i-1)}| = max_{m\ge i}|p_m^{(i-1)}|$.
13. $\quad\quad\quad$ Interchange columns i and k of $W - I$ and rows i and k of $\pi_{m_i}^{(i-1)}$ and elements $p_i^{(i-1)}$ and $p_k^{(i-1)}$.
14. $\quad\quad\quad \Pi = \pi_{m_i}^{(i-1)}\Pi$
15. $\quad\quad$ **end if**
16. $\quad\quad$ *satisfied_ p = true*.
17. $\quad\quad$ **for** $j = i$ to n **do**
18. $\quad\quad\quad q_j^{(i-1)} = (z_j^{(i-1)})^T(\Pi A\Sigma)^T w_i^{(i-1)}$.
19. $\quad\quad$ **end for**
20. $\quad\quad$ **if** not *satisfied_ q* **then**
21. $\quad\quad\quad$ **if** $|q_i^{(i-1)}| < \alpha\ max_{m\ge i}|q_m^{(i-1)}|$ **then**
22. $\quad\quad\quad\quad n_i = n_i + 1, \quad \sigma_{n_i}^{(i-1)} = I_n$.
23. $\quad\quad\quad\quad$ *satisfied_ p = false* , choose l such that $|q_l^{(i-1)}| = max_{m\ge i}|q_m^{(i-1)}|$.
24. $\quad\quad\quad\quad$ Interchange columns i and l of $Z - I$ and $\sigma_{n_i}^{(i-1)}$ and elements $q_i^{(i-1)}$ and $q_l^{(i-1)}$.
25. $\quad\quad\quad\quad \Sigma = \Sigma\sigma_{n_i}^{(i-1)}$
26. $\quad\quad\quad$ **end if**
27. $\quad\quad$ **end if**
28. $\quad\quad$ *satisfied_ q = true*
29. $\quad$ **end while**
30. $\quad d_{ii} = p_i^{(i-1)}$.
31. $\quad$ **for** $j = i + 1$ to n **do**
32. $\quad\quad p_j^{(i-1)} = \frac{p_j^{(i-1)}}{d_{ii}},\ \ q_j^{(i-1)} = \frac{q_j^{(i-1)}}{d_{ii}}$.
33. $\quad\quad w_j^{(i)} = w_j^{(i-1)} - p_j^{(i-1)}w_i^{(i-1)},\ \ z_j^{(i)} = z_j^{(i-1)} - q_j^{(i-1)}z_i^{(i-1)}$.
34. $\quad\quad$ for all $l \le i$ apply dropping rule to $w_{lj}^{(i)}$ and to $z_{lj}^{(i)}$, if their absolute values are less than τ_w and τ_z.
35. $\quad$ **end for**
36. **end for**
37. Return $Z = [z_1^{(0)}, z_2^{(1)}, \cdots, z_n^{(n-1)}]$, $W = [w_1^{(0)}, w_2^{(1)}, \cdots, w_n^{(n-1)}]$, $D = diag(d_{ii})_{1\le i\le n}$, Π and Σ

4. Right-looking RIF preconditioner with pivoting

In 2003, Benzi and Tůma computed an incomplete factorization of a symmetric positive definite matrix A in the form of $A \approx LDL^T$ as a by-product of the *AINV* preconditioner [7]. In this paper, we focus on the nonsymmetric version of this preconditioner [9]. Algorithm 5, computes the right-looking version of this preconditioner. One of the advantages of the *RIF* preconditioner over the other *ILU*'s that are computed by different versions of the Gaussian Elimination, is that its factors are computed independently and there is no need to work with the Schur-Complement matrix. At step i of Algorithm 5, $L_{:,i}$ and $U_{i,:}$ are computed, independently. More precisely, matrix L is computed column-wise and matrix U is computed row-wise. If no dropping be applied in both Algorithms 1 and 5, the L, U and D matrices computed in both algorithms are equal.

The main question that came to our mind was whether or not it is possible to do pivoting in Algorithm 5? Since the basis of this algorithm has been inherited from the *AINV*, the pivoting in this algorithm should be done through the pivoting in *AINV*.

Algorithm 6, computes the right-looking version of the *RIF* preconditioner and uses the complete pivoting strategy. This algorithm is so similar to Algorithm 4, except that one should compute the L and U matrices as by-products and also more interchanges should be done for the row and column pivoting. Suppose that at step i of this algorithm and in line 10, the

row index k satisfies criterion (6). Besides interchanging columns i and k of matrix $W - I$ and rows i and k of matrix $\pi_{m_i}^{(i-1)}$, one should also interchange $L(i, 1 : i-1)$ and $L(k, 1 : i-1)$ parts of matrix L. There is not the last interchange in Algorithm 4. All of these interchanges have been mentioned in line 13 of this algorithm. Also suppose that at step i of Algorithm 6 and in line 21, the column index l satisfies criterion (7). In line 24 of this algorithm, we should interchange columns i and l of matrices $Z - I$ and $\sigma_{n_i}^{(i-1)}$ and also we should interchange $U(1 : i-1, i)$ and $U(1 : i-1, l)$ parts of matrix U. Again, we emphasize that there is no need to the last interchange in Algorithm 4.

Algorithm 5 (Right-looking RIF algorithm)

Input: Let $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ and τ_w, τ_z, τ_l, $\tau_u \in (0, 1)$ be drop tolerances.
Output: $A \approx LDU$.

1. $w_i^{(0)} = e_i, \ z_i^{(0)} = e_i, \quad 1 \le i \le n.$
2. **for** $i = 1$ to n **do**
3. $\quad p_i^{(i-1)} = (w_i^{(i-1)})^T A z_i^{(i-1)}, \quad q_i^{(i-1)} = (z_i^{(i-1)})^T A^T w_i^{(i-1)}.$
4. $\quad$ **for** $j = i + 1$ to n **do**
5. $\quad\quad p_j^{(i-1)} = \frac{(w_j^{(i-1)})^T A z_i^{(i-1)}}{p_i^{(i-1)}}, \quad q_j^{(i-1)} = \frac{(z_j^{(i-1)})^T A^T w_i^{(i-1)}}{q_i^{(i-1)}}.$
6. $\quad\quad L_{ji} = p_j^{(i-1)}, \ U_{ij} = q_j^{(i-1)}.$
7. $\quad\quad$ apply dropping rule to L_{ji} and to U_{ij} if their absolute values are less than τ_l and τ_u.
8. $\quad\quad w_j^{(i)} = w_j^{(i-1)} - p_j^{(i-1)} w_i^{(i-1)}, \quad z_j^{(i)} = z_j^{(i-1)} - q_j^{(i-1)} z_i^{(i-1)}.$
9. $\quad\quad$ for all $l \le i$ apply dropping rule to $w_{lj}^{(i)}$ and to $z_{lj}^{(i)}$ if their absolute values are less than τ_w and τ_z.
10. $\quad$ **end for**
11. $\quad d_{ii} = p_i^{(i-1)}.$
12. **end for**
13. Return $L = (L_{ij})_{1\le i,j\le n}$, $U = (U_{ij})_{1\le i,j\le n}$ and $D = diag(d_{ii})_{1\le i\le n}$.

5. Numerical results and implementation details

In this section, we report results of the *GMRES*(30) method [1] to solve the original and the right preconditioned linear systems. Performance of *GMRES* is dependent on the basis size of the Krylov subspace method and this also affects the performance of the preconditioners. This is why we have also reported results of Bicgstab and *TFQMR* methods [1]. Preconditioners are right-looking *RIF* with and without pivoting. We have used the notation *RLRIF* to indicate the right-looking version of the *RIF* preconditioner in Tables 3–6. We have also considered the notation *RLRIFP*(α) in these tables to indicate the right-looking version of *RIF* with pivoting that uses parameter α for column and row pivoting. We have generated 19 artificial linear systems $Ax = b$ with the nonsymmetric coefficient matrices from the University of Florida Sparse Matrix Collection [8]. Vector b is Ae in which $e = [1, \ldots, 1]^T$. We have written the code for right-looking version of *RIF* with pivoting in Fortran 90. There are some details in the implementation of both Algorithms 5 and 6 that are presented here.

- In Algorithm 6, we do not compute matrices Π and Π^T. But instead, we work with two permutation arrays *permw* and *invpermw*. Suppose that interchanges in line 13 of this algorithm should be done. Then, we interchange *permw*(i) and *permw*(k) together and update array *invpermw* such that *invpermw*(*permw*(i)) $= i$ and *invpermw*(*permw*(k)) $= k$. Therefore, matrices Π and Π^T can be extracted from arrays *invpermw* and *permw*, respectively.
- If we work with two permutation arrays *permz* and *invpermz*, then we do not need to compute matrices Σ and Σ^T in Algorithm 6. Suppose that interchanges in line 24 of this algorithm should be applied. Then, we interchange *permz*(i) and *permz*(l) together and update array *invpermz* such that *invpermz*(*permz*(i)) $= i$ and *invpermz*(*permz*(l)) $= l$. Therefore, matrices Σ and Σ^T can be extracted from arrays *permz* and *invpermz*, respectively.
- In both algorithms, matrices A and A^T are stored in *csc* format [1]. This gives the opportunity to compute vectors $Az_i^{(i-1)}$ and $A^T w_i^{(i-1)}$ as a linear combination of columns of A and A^T. Therefore, elements $p_j^{(i-1)}$ and $q_j^{(i-1)}$, for $j \ge i$, in lines 3 and 5 of Algorithm 5, will be computed in sparse–sparse mode. The same situation will happen for Algorithm 6. This means that one can compute vectors $(\Pi A\Sigma)z_i^{(i-1)}$ and $(\Pi A\Sigma)^T w_i^{(i-1)}$ as a linear combination of columns of A and A^T. But this time, one should also work with the permutation arrays *invpermw*, *permw*, *permz* and *invpermz*. After computing these two vectors, elements $p_j^{(i-1)}$ and $q_j^{(i-1)}$, for $j \ge i$, in lines 8 and 18 of Algorithm 6, will be computed in sparse–sparse mode. Therefore, this format of storage for matrices A and A^T, will decrease the preconditioning time of *RLRIFP*(α) and *RLRIF* preconditioners.
- In both algorithms, matrices Z and W are stored in dynamic sparse column format. The graph of these two matrices should also be stored in dynamic sparse row format [4].
- When there is a need to interchange columns of matrices $W - I$ and $Z - I$ in Algorithm 6, we only interchange the entries of permutation arrays *permw* and *permz*, respectively. But we should also update the dynamic sparse row format of the graph of these two matrices, explicitly.

- In both algorithms, matrices L and U are stored in *csc* and *csr* formats [1], respectively. But in Algorithm 6, we also need to store the graph of matrix L in dynamic sparse row format and the graph of matrix U in dynamic sparse column format. This gives the chance to interchange rows of matrix $L - I$ and columns of matrix $U - I$ whenever it is needed.

Algorithm 6 (Right-looking RIF with pivoting)

Input: $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ and τ_w, $\tau_z \in (0, 1)$ be drop tolerances and prescribe a tolerance $\alpha \in (0, 1]$.
Output: $\Pi A \Sigma \approx LDU$

1. $\Pi = \Sigma = I_n$.
2. $w_i^{(0)} = e_i$, $z_i^{(0)} = e_i$, $1 \le i \le n$.
3. **for** $i = 1$ to n **do**
4. $m_i = n_i = 0$
5. *satisfied_ p* $=$ *false*, *satisfied_ q* $=$ *false*.
6. **while** not *satisfied_ p* **do**
7. **for** $j = i$ to n **do**
8. $p_j^{(i-1)} = (w_j^{(i-1)})^T (\Pi A \Sigma) z_i^{(i-1)}$.
9. **end for**
10. **if** $|p_i^{(i-1)}| < \alpha\ max_{m\ge i}|p_m^{(i-1)}|$ **then**
11. $m_i = m_i + 1$, $\pi_{m_i}^{(i-1)} = I_n$
12. *satisfied_ q* $=$ *false*, choose k such that $|p_k^{(i-1)}| = max_{m\ge i}|p_m^{(i-1)}|$.
13. Interchange columns i and k of $W - I$ and rows i and k of $L - I$ and $\pi_{m_i}^{(i-1)}$ and elements $p_i^{(i-1)}$ and $p_k^{(i-1)}$.
14. $\Pi = \pi_{m_i}^{(i-1)} \Pi$
15. **end if**
16. *satisfied_ p* $=$ *true*
17. **for** $j = i$ to n **do**
18. $q_j^{(i-1)} = (z_j^{(i-1)})^T (\Pi A \Sigma)^T w_i^{(i-1)}$.
19. **end for**
20. **if** not *satisfied_ q* **then**
21. **if** $|q_i^{(i-1)}| < \alpha\ max_{m\ge i}|q_m^{(i-1)}|$ **then**
22. $n_i = n_i + 1$, $\sigma_{n_i}^{(i-1)} = I_n$
23. *satisfied_ p* $=$ *false*, choose l such that $|q_l^{(i-1)}| = max_{m\ge i}|q_m^{(i-1)}|$.
24. Interchange columns i and l of $Z - I$, $U - I$ and $\sigma_{n_i}^{(i-1)}$ and elements $q_i^{(i-1)}$ and $q_l^{(i-1)}$.
25. $\Sigma = \Sigma \sigma_{n_i}^{(i-1)}$
26. **end if**
27. **end if**
28. *satisfied_ q* $=$ *true*.
29. **end while**
30. $d_{ii} = p_i^{(i-1)}$.
31. **for** $j = i + 1$ to n **do**
32. $p_j^{(i-1)} = \frac{p_j^{(i-1)}}{d_{ii}}$, $q_j^{(i-1)} = \frac{q_j^{(i-1)}}{d_{ii}}$.
33. $L_{ji} = p_j^{(i-1)}$, $U_{ij} = q_j^{(i-1)}$.
34. apply dropping rule to L_{ji} and to U_{ij} if their absolute values are less than τ_l and τ_u.
35. $w_j^{(i)} = w_j^{(i-1)} - p_j^{(i-1)} w_i^{(i-1)}$, $z_j^{(i)} = z_j^{(i-1)} - q_j^{(i-1)} z_i^{(i-1)}$.
36. for all $l \le i$ apply a dropping rule to $w_{lj}^{(i)}$ and to $z_{lj}^{(i)}$, if their absolute values are less than τ_w and τ_z.
37. **end for**
38. **end for**
39. Return $L = (L_{ij})_{1\le i,j\le n}$, $U = (U_{ij})_{1\le i,j\le n}$, $D = diag(d_{ii})_{1\le i\le n}$, Π and Σ

The code of right-looking version of *RIF* has been taken from *Sparslab* [10] package and codes of *GMRES*, Bicgstab and *TFQMR* methods have been taken from *Sparskit* [11] package. All the codes have been run on a machine with 3 GB of RAM memory and have been compiled with Compaq Visual Fortran 6.6a with no optimization option. All the codes have been run in double precision arithmetic. In all the experiments, we have selected τ_l, τ_u, τ_w and τ_z equal to 0.1.

Table 1, gives the matrix properties. In this table, n is the dimension and *nnz* is the number of nonzero entries of the matrix.

In Table 2, results of iterative methods with no preconditioning have been presented. In this table, *it* is the number of iterations and Itime is the iteration time. This parameter is in seconds. A + in this table means that the convergence criterion

Table 1
Matrix properties.

Matrix	n	nnz
*add*20	2 395	17 319
*hor*_131	434	4 710
Poisson3Da	13 514	352 762
*raefsky*1	3 242	294 276
*raefsky*2	3 242	294 276
*raefsky*5	6 316	168 658
*raefsky*6	3 402	137 845
*sherman*4	1 104	3 786
sme3Da	12 504	874 887
sme3Db	29 067	2 081 063
*orsirr*_1	1 030	6 858
*orsirr*_2	886	5 970
*wang*1	2 903	19 093
*wang*2	2 903	19 093
*tols*1090	1 090	3 546
*cdde*1	961	4 681
memplus	17 758	99 147
*orsreg*_1	2 205	14 133
*sherman*5	3 312	20 793

Table 2
Results of iterative methods without preconditioning.

Method	Bicgstab		*GMRES*(30)		*TFQMR*	
	it	Itime	*it*	Itime	*it*	Itime
*add*20	735	0.187	699	0.500	419	0.125
*hor*_131	+	+	+	+	+	+
Poisson3Da	221	0.796	261	1.609	189	0.718
*raefsky*1	435	1.015	+	+	731	1.734
*raefsky*2	735	1.718	+	+	1047	2.484
*raefsky*5	181	0.281	58	0.171	111	0.171
*raefsky*6	+	+	1353	2.531	575	0.703
*sherman*4	201	0.015	558	0.171	219	0.015
sme3Da	+	+	+	+	+	+
sme3Db	+	+	+	+	+	+
*orsirr*_1	+	+	+	+	+	+
*orsirr*_2	+	+	2015	0.531	1777	0.171
*wang*1	559	0.140	824	0.703	927	0.281
*wang*2	559	0.156	1131	0.968	877	0.265
*tols*1090	+	+	+	+	+	+
*cdde*1	201	0.015	801	0.218	181	0.015
memplus	2153	3.75	+	+	1473	2.984
*orsreg*_1	673	0.140	432	0.265	515	0.109
*sherman*5	+	+	+	+	2439	0.843

has not been satisfied in 5000 iterations. For all the experiments, the convergence criterion is satisfied when the relative residual is less than 10^{-8}.

In Table 3, *Ptime* is the preconditioning time which is computed by using the *dtime* command. This parameter is in seconds. Since we have merged factors D and U together, then, for all preconditioners, density in this table is defined as

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

where $nnz(L)$, $nnz(U)$ and $nnz(A)$ are the number of nonzero entries of L, U and A, respectively. In this table, we have selected parameter α equal to 0.1, 0.4, 0.75 and 1.0. Results of this table, indicate that *Ptime* of *RLRIFP*(0.1), *RLRIFP*(0.4), *RLRIFP*(0.75) and *RLRIFP*(1.0) preconditioners are more or less the same. For 10 matrices, *Ptime* of these preconditioners are greater than *Ptime* of the *RLRIF* preconditioner. But for matrix *tols*1090, this is vice versa. For matrices *hor*_131, *raefsky*6, *sherman*4, *orsirr*_1, *wang*1, *cdde*1, *orsreg*_1 and *sherman*5, *Ptime* of one of the *RLRIFP*(0.1), *RLRIFP*(0.4), *RLRIFP*(0.75) and *RLRIFP*(1.0) preconditioners is equal to *Ptime* of the *RLRIF* preconditioner. In this table, except for matrices *hor*_131, *raefsky*6 and *tols*1090, for other matrices, densities of *RLRIFP*(0.1), *RLRIFP*(0.4), *RLRIFP*(0.75) and *RLRIFP*(1.0) preconditioners are nearly the same as each other and it is also nearly the same as the density of the *RLRIF* preconditioner. This gives a proper atmosphere to compare number of iterations of *GMRES*(30), Bicgstab and *TFQMR* methods when these preconditioners are used as the right preconditioner.

In Tables 4–6, results of *GMRES*(30), Bicgstab and *TFQMR* methods are presented, respectively. In these tables, *it* is the number of iterations of the Krylov subspace method and *Ttime* is the total time which is defined as the iteration time plus

Table 3
Properties of *RLRIFP*(1.0), *RLRIFP*(0.1), *RLRIFP*(0.4), *RLRIFP*(0.75) and *RLRIF* preconditioners.

Method	*RLRIFP*(1.0)		*RLRIFP*(0.1)		*RLRIFP*(0.4)		*RLRIFP*(0.75)		*RLRIF*	
	Ptime	*Density*	*Ptime*	*Density*	*Ptime*	*Density*	*Ptime*	*Density*	*Ptime*	*Density*
*add*20	0.062	0.555	0.046	0.555	0.062	0.555	0.062	0.555	0.031	0.535
*hor*_131	0.046	1.114	0.031	0.926	0.046	1.037	0.046	1.180	0.031	0.886
*Poisson*3*Da*	1.140	0.318	1.156	0.318	1.140	0.318	1.109	0.318	0.203	0.327
*raefsky*1	0.218	0.089	0.218	0.089	0.218	0.089	0.218	0.089	0.078	0.091
*raefsky*2	0.328	0.145	0.312	0.145	0.328	0.145	0.312	0.145	0.109	0.147
*raefsky*5	0.109	0.252	0.125	0.255	0.109	0.255	0.109	0.255	0.078	0.260
*raefsky*6	0.078	0.087	0.093	0.117	0.078	0.089	0.078	0.087	0.078	0.273
*sherman*4	0.015	1.255	0.031	1.254	0.015	1.254	0.031	1.254	0.031	1.236
*sme*3*Da*	2.468	0.225	2.484	0.225	2.468	0.225	2.468	0.225	0.812	0.281
*sme*3*Db*	6.390	0.214	6.437	0.214	6.390	0.214	6.390	0.214	1.609	0.257
*orsirr*_1	0.031	0.621	0.031	0.540	0.015	0.544	0.031	0.613	0.031	0.542
*orsirr*_2	0.031	0.622	0.031	0.543	0.031	0.543	0.031	0.617	0.015	0.545
*wang*1	0.046	0.863	0.046	0.861	0.062	0.880	0.046	0.872	0.046	0.847
*wang*2	0.046	0.865	0.046	0.868	0.046	0.871	0.046	0.867	0.031	0.857
*tols*1090	0.015	0.807	0.015	0.808	0.015	0.808	0.015	0.806	0.031	1.306
*cdde*1	0.031	1.216	0.015	1.216	0.031	1.216	0.031	1.216	0.031	1.205
memplus	0.234	0.598	0.234	0.598	0.234	0.598	0.250	0.598	0.078	0.567
*orsreg*_1	0.046	0.648	0.031	0.561	0.031	0.567	0.031	0.627	0.031	0.564
*sherman*5	0.031	0.600	0.031	0.547	0.046	0.591	0.031	0.596	0.046	0.635

Table 4
Results of the *GMRES*(30) method by using *RLRIFP*(1.0), *RLRIFP*(0.1), *RLRIFP*(0.4), *RLRIFP*(0.75) and *RLRIF* preconditioners.

Method	*RLRIFP*(1.0)		*RLRIFP*(0.1)		*RLRIFP*(0.4)		*RLRIFP*(0.75)		*RLRIF*	
	it	*Ttime*	*it*	*Ttime*	*it*	*Ttime*	*it*	*Ttime*	*it*	*Ttime*
*add*20	10	0.078	10	0.046	10	0.078	10	0.078	13	0.046
*hor*_131	74	0.062	98	0.046	10	0.593	10	0.625	31	0.031
*Poisson*3*Da*	137	2.265	137	2.312	137	2.250	137	2.234	128	1.250
*raefsky*1	367	1.484	367	1.500	367	1.484	367	1.484	448	1.578
*raefsky*2	422	1.828	422	1.859	422	1.843	422	1.812	478	1.781
*raefsky*5	7	0.125	7	0.140	7	0.125	7	0.125	7	0.093
*raefsky*6	7	0.093	11	0.109	9	0.093	9	0.093	7	0.093
*sherman*4	42	0.031	39	0.046	39	0.031	39	0.046	53	0.046
*sme*3*Da*	+	+	+	+	+	+	+	+	+	+
*sme*3*Db*	+	+	+	+	+	+	+	+	+	+
*orsirr*_1	34	0.046	254	0.109	39	0.937	39	0.953	71	0.062
*orsirr*_2	34	0.046	292	0.125	292	0.125	39	0.828	71	0.031
*wang*1	60	0.109	**1983**	2.250	292	2.843	**1250**	1.437	62	0.125
*wang*2	62	0.125	972	1.125	**1766**	2.015	234	0.296	68	0.109
*tols*1090	8	0.015	6	0.015	6	0.015	6	0.015	68	0.906
*cdde*1	54	0.046	54	0.031	54	0.046	54	0.046	58	0.062
memplus	76	0.734	76	0.734	76	0.734	76	0.750	136	0.937
*orsreg*_1	37	0.078	261	0.234	76	1.953	76	1.968	73	0.078
*sherman*5	60	0.109	261	2.984	76	3.031	277	0.359	92	0.156

the preconditioning time. This parameter is also in seconds. In these tables, + indicates that the convergence criterion has not been satisfied in 2000 iterations. There are some bold numbers in columns 4, 6 and 8 of these tables which indicate that pivoting with parameters $\alpha = 0.1$, $\alpha = 0.4$ and $\alpha = 0.75$ has a bad effect on the performance of the Krylov subspace methods.

With respect to the information of *GMRES*(30), Bicgstab and *TFQMR* methods in Tables 4–6, we have categorized matrices of each of these tables to 7 groups.

Group 1. For matrices of this group, *RLRIFP*(0.1), *RLRIFP*(0.4), *RLRIFP*(0.75) and *RLRIFP*(1.0) preconditioners make the Krylov subspace method convergent in the same number of iterations. This number of iterations is less than the number of iterations of the Krylov subspace method when *RLRIF* is the preconditioner. For matrices of this group, there is not a great difference between *Ttime* of all of the preconditioners.

Group 2. For matrices of this group, the *RLRIFP*(1.0) preconditioner makes the Krylov subspace method convergent in less number of iterations than *RLRIFP*(0.1), *RLRIFP*(0.4), *RLRIFP*(0.75) and *RLRIF* preconditioners. For matrices of this group, there is not a great difference between *Ttime* of the *RLRIFP*(1.0) preconditioner with *Ttime* of other preconditioners.

Group 3. There are matrices in this group that their *RLRIFP*(0.1), *RLRIFP*(0.4) and *RLRIFP*(0.75) preconditioners make the Krylov subspace method convergent in less number of iterations than *RLRIFP*(1.0) and *RLRIF* preconditioners. For these matrices, the *RLRIFP*(1.0) preconditioner also makes the Krylov subspace method convergent in less number of iterations than the *RLRIF* preconditioner.

Table 5
Results of the Bicgstab method by using *RLRIFP*(1.0), *RLRIFP*(0.1), *RLRIFP*(0.4), *RLRIFP*(0.75) and *RLRIF* preconditioners.

Method	*RLRIFP*(1.0)		*RLRIFP*(0.1)		*RLRIFP*(0.4)		*RLRIFP*(0.75)		*RLRIF*	
	it	*Ttime*	*it*	*Ttime*	*it*	*Ttime*	*it*	*Ttime*	*it*	*Ttime*
*add*20	11	0.062	11	0.046	11	0.062	11	0.062	15	0.031
*hor*_131	115	0.062	163	0.046	395	0.093	409	0.109	43	0.031
Poisson3Da	149	1.984	**241**	2.500	**241**	2.500	**241**	2.468	183	1.203
*raefsky*1	97	0.500	97	0.484	97	0.500	97	0.484	97	0.343
*raefsky*2	143	0.750	143	0.750	143	0.750	143	0.734	151	0.546
*raefsky*5	9	0.125	9	0.140	9	0.125	9	0.140	7	0.093
*raefsky*6	9	0.093	17	0.125	13	0.093	13	0.093	7	0.093
*sherman*4	45	0.015	49	0.031	49	0.031	49	0.031	51	0.031
sme3Da	+	+	+	+	+	+	+	+	+	+
sme3Db	+	+	+	+	+	+	+	+	+	+
*orsirr*_1	43	0.046	613	0.125	+	+	+	+	73	0.046
*orsirr*_2	39	0.031	647	0.109	647	0.109	+	+	79	0.031
*wang*1	65	0.093	549	0.343	**1273**	0.750	**743**	0.453	61	0.078
*wang*2	69	0.078	399	0.265	499	0.312	221	0.171	67	0.078
*tols*1090	11	0.015	9	0.015	9	0.015	9	0.015	67	0.359
*cdde*1	61	0.031	61	0.031	61	0.031	61	0.046	61	0.046
memplus	505	1.906	505	1.906	505	1.906	505	1.921	325	1.046
*orsreg*_1	49	0.062	301	0.140	505	0.859	505	0.875	77	0.046
*sherman*5	55	0.062	561	0.328	665	0.406	173	0.125	67	0.078

Table 6
Results of the *TFQMR* method by using *RLRIFP*(1.0), *RLRIFP*(0.1), *RLRIFP*(0.4), *RLRIFP*(0.75) and *RLRIF* preconditioners.

Method	*RLRIFP*(1.0)		*RLRIFP*(0.1)		*RLRIFP*(0.4)		*RLRIFP*(0.75)		*RLRIF*	
	it	*Ttime*	*it*	*Ttime*	*it*	*Ttime*	*it*	*Ttime*	*it*	*Ttime*
*add*20	11	0.062	11	0.062	11	0.062	11	0.062	15	0.031
*hor*_131	97	0.062	101	0.046	553	0.125	351	0.093	39	0.046
Poisson3Da	123	1.843	**193**	2.328	**193**	2.265	**193**	2.234	121	0.890
*raefsky*1	111	0.531	111	0.531	111	0.531	111	0.531	111	0.375
*raefsky*2	157	0.812	157	0.812	157	0.796	157	0.781	167	0.593
*raefsky*5	9	0.140	9	0.156	9	0.125	9	0.125	9	0.093
*raefsky*6	9	0.093	15	0.109	13	0.093	13	0.093	9	0.093
*sherman*4	57	0.015	57	0.046	57	0.031	57	0.046	65	0.031
sme3Da	1703	21.468	+	+	+	+	+	+	+	+
sme3Db	+	+	+	+	+	+	+	+	+	+
*orsirr*_1	37	0.031	315	0.093	1431	0.265	57	0.500	77	0.031
*orsirr*_2	39	0.046	343	0.093	343	0.078	57	0.437	77	0.031
*wang*1	77	0.093	479	0.328	**1455**	0.921	423	0.296	79	0.078
*wang*2	75	0.093	347	0.250	459	0.312	185	0.156	73	0.062
*tols*1090	11	0.031	9	0.015	9	0.015	9	0.015	73	0.421
*cdde*1	65	0.046	65	0.031	65	0.046	65	0.031	63	0.031
memplus	77	0.515	77	0.500	77	0.515	77	0.531	113	0.437
*orsreg*_1	39	0.062	291	0.125	**1933**	0.734	77	0.968	81	0.062
*sherman*5	65	0.078	633	0.390	561	0.375	137	0.109	87	0.093

Group 4. A matrix belongs to this group if pivoting with all values of α has weakened the quality of its *RLRIF* preconditioner. In other words, for matrices of this group, *RLRIFP*(0.1), *RLRIFP*(0.4), *RLRIFP*(0.75) and *RLRIFP*(1.0) preconditioners have made the Krylov subspace method convergent in more number of iterations than the *RLRIF* preconditioner.

Group 5. A matrix of this group has the property that pivoting with all values of α has no effect on the quality of its *RLRIF* preconditioner. Therefore, for matrices of this group, all the preconditioners make the Krylov subspace method convergent in the same number of iterations.

Group 6. For matrices of this group, pivoting has improved the quality of their *RLRIF* preconditioner for some values of α and has weakened the quality of this preconditioner for some other values of α. This means that for some values of α, *RLRIFP*(α) makes the Krylov subspace method convergent in less number of iterations than the *RLRIF* preconditioner and for some other values of α this is vice versa.

Group 7. A matrix is in this group if none of its preconditioners are useful to make the Krylov subspace method convergent.

Results of the *GMRES*(30) method presented in Table 4, indicate that matrices *add*20, *raefsky*1, *raefsky*2, *cdde*1 and *memplus* are in group 1. Matrices *orsirr*_ 1, *orsirr*_ 2, *wang*1, *wang*2, *orsreg*_ 1 and *sherman*5 belong to group 2. Matrices *sherman*4 and *tols*1090 are in group 3. Matrices *Poisson*3*Da* and *raefsky*6 are in group 4. Matrix *raefsky*5 is in group 5. Matrix *hor*_131 is in group 6 and finally, matrices *sme*3*Da* and *sme*3*Db* belong to group 7. By analyzing these results, one can say that when the *GMRES*(30) method is used to solve the right preconditioned linear systems, then pivoting will be effective on

the quality of the *RLRIF* preconditioner and the *RLRIFP*(1.0) preconditioner seems to be more effective to reduce the number of iterations of this Krylov subspace method than other *RLRIFP*(α) preconditioners.

Results of the Bicgstab method in Table 5, show that matrices *add*20, *raefsky*2 and *sherman*4 are in group 1. Matrices *Poisson*3*Da*, *orsirr*_ 1, *orsirr*_2, *orsreg*_1 and *sherman*5 belong to group 2. Matrix *tols*1090 is in group 3. Matrices *hor*_131, *raefsky*5, *raefsky*6, *wang*1, *wang*2 and *memplus* are in group 4. Matrices *raefsky*1 and *cdde*1 are in group 5. There is no matrix in group 6 and finally, matrices *sme*3*Da* and *sme*3*Db* are in group 7. Thus, from these results it is clear that when the Bicgstab method is used to solve the right preconditioned linear systems, pivoting is useful on the quality of the *RLRIF* preconditioner and it is better to choose parameter α equal to 1.0.

By considering results of the *TFQMR* method in Table 6, one can say that matrices *add*20, *raefsky*2, *sherman*4 and *memplus* are in group 1. Matrices *sme*3*Da*, *orsirr*_ 1, *orsirr*_ 2, *wang*1, *orsreg*_ 1 and *sherman*5 belong to group 2. Matrix *tols*1090 is in group 3. Matrices *hor*_131, *Poisson*3*Da*, *raefsky*6, *wang*2 and *cdde*1 are in group 4. Matrices *raefsky*1 and *raefsky*5 are in group 5. There is no matrix in group 6 and at the end, matrix *sme*3*Db* is in group 7. Therefore, it can be concluded from this data that when the *TFQMR* method is exploited to solve the right preconditioned linear systems, then pivoting is effective for the *RLRIF* preconditioner and the best choice of α is 1.0.

6. Conclusion

In this paper, we presented right-looking version of the *RIF* preconditioner with pivoting. We used parameters $\alpha = 0.1$, $\alpha = 0.4$, $\alpha = 0.75$ and $\alpha = 1.0$ to compute this preconditioner. We also used this preconditioner as the right preconditioner for several linear systems. Numerical experiments indicate the effectiveness of this preconditioner to reduce the number of iterations of the Krylov subspace methods *GMRES*(30), Bicgstab and *TFQMR*. The results also emphasize the fact that the choice of $\alpha = 1.0$ improves the quality of this preconditioner more than the other choices of α.

Acknowledgment

I would like to thank Miroslav Tůma for providing me with reference six.

References

[1] Y. Saad, Iterative Methods for Sparse Linear Systems, PWS Publishing, New York, 1996.
[2] M. Bollhöfer, A robust ILU based on monitoring the growth of the inverse factors, Linear Algebra Appl. 338 (2001) 201–218.
[3] M. Bollhöfer, Y. Saad, On the relations between ILUs and factored approximate inverses, SIAM J. Matrix Anal. Appl. 24 (2002) 219–237.
[4] M. Benzi, M. Tůma, A sparse approximate inverse preconditioner for nonsymmetric linear systems, SIAM J. Sci. Comput. 19 (1998) 968–994.
[5] M. Bollhöfer, Y. Saad, A factored approximate inverse preconditioner with pivoting, SIAM J. Matrix Anal. Appl. 23 (2002) 692–705.
[6] M. Túma, Solving sparse unsymmetric sets of linear equations based on implicit gauss projection. Technical Report No. 556, Institute of Computer Science, Academy of Sciences of the Czech Republic, 1993.
[7] M. Benzi, M. Tůma, A robust incomplete factorization preconditioner for positive definite matrices, Numer. Linear Algebra Appl. 10 (2003) 385–400.
[8] T. Davis, University of Florida Sparse Matrix Collection. http://www.cise.ufl.edu/research/sparse/matrices (accessed 2012).
[9] A. Rafiei, M. Bollhöfer, Extension of inverse-based dropping techniques for ILU preconditioners. http://www.digibib.tu-bs.de/?docid=00035624.
[10] M. Benzi, M. Tůma, Sparslab software package, 2012. http://www2.cs.cas.cz/~tuma/sparslab.html (accessed March 2012).
[11] Y. Saad, Sparskit and sparse examples, NA Digest, 1994. http://www-users.cs.umn.edu/~saad/software (accessed March 2012).