Cumhuriyet Üniversitesi Fen Fakültesi
Fen Bilimleri Dergisi (CFD), Cilt 37, No. 3 (2016)
ISSN: 1300-1949

Cumhuriyet University Faculty of Science
Science Journal (CSJ), Vol. 37, No. 3 (2016)
ISSN: 1300-1949

http://dx.doi.org/10.17776/csj.18615

Incomplete LU Factorization on Projection Method

S.A. SHAHZADEH FAZELI, A. GHODRATNAMA, A. SADEGHIANZ, S.M. KARBASSIX

Faculty of Mathematics, Yazd University, Yazd, Iran.

Received: 15.09.2015; Accepted: 20.06.2016

Abstarct. The projection method allows solving sparse linear systems. Solving the sparse linear systems is a common problem which arises from many complex applications. The problems to be solved often are of very large size. Combination of the preconditioners with the projection methods continues to play an important role in solving the sparse linear system. In this paper, we propose a new technique to solve a linear system. This approach is called incomplete LU factorization on Full Orthogonalization method (ILUFOM). Here we present and examine a number of techniques for solving sparse linear systems using incomplete LU factorization. Particularly GMRES and FOM method with some preconditions are considered. The efficiency of the algorithm is demonstrated using an example. According to our experiments, ILUFOM improves the convergence of FOM.

Keywords: Preconditioning, Projection method, LU Factorization, Full Orthogonalization Method

Projeksiyon Yöntemi Üzerine Eksik LU Faktörizasyonu

Özet. Projeksiyon yöntemi seyrek doğrusal sistemleri çözümüne izin verir. Seyrek doğrusal sistemlerinin çözümü, çok karmaşık uygulamalarda ortaya çıkan yaygın bir problemdir. Çözülen problemler sıklıkla çok büyük boyutlardadır. Projeksiyon yöntemleri ile ön şartlandırıcı kombinasyonu seyrek lineer sistem çözümünde önemli bir rol oynamaya devam etmektedir. Bu yazıda, doğrusal bir sistemi çözmek için yeni bir teknik öneriyoruz. Bu yaklaşıma, Tam Ortogonalleştirme metodu (ILUFOM) üzerine eksik LU çarpanlara ayırma denir. Burada mevcut ve eksik LU çarpanlara kullanarak seyrek doğrusal sistemleri çözmek için bir takım teknikler sunduk ve inceledik. Özellikle bazı önkoşullarla GMRES ve FORM yöntemleri ele alınmıştır. Algoritmanın etkinliği bir örnekle gösterilmiştir. Deneylere göre, ILUFOM, FOM yakınsamasını iyileştirmektedir.

Anahtar Kelimeler: Ön koşullandırma, projeksiyon metodu, LU çarpanlara ayırma, Tam Ortogonalleştirme metodu

1. INTRODUCTION

Solving the linear system Ax=b is one of the most important problems in linear algebra and has important applications in automatic control theory, signal processing and telecommunications. There are two types of methods for solving linear systems:

1. Direct methods
2. Iterative methods

The direct methods like Gaussian elimination and the method based on the QR factorization consist of a finite number of steps that all must be performed for any given instance before the solution is obtained. On the other hand, iterative methods are based on computing a sequence of approximations to the solution x by choosing initial solution x and computation stops whenever a certain desired accuracy is obtained or after certain number of iterations [7] and [8]. The iterative methods are used primarily for large and sparse systems. These methods include the following: the Jacobi method, the Gauss-Seidel method, the successive over relaxation method, the conjugate gradient method with and without preconditioner, the GMRES method and the FOM method [12,13,15].

The basic idea behind an iterative method is first to write the system Ax=b in an equivalent form:

$$x = Bx + r \tag{1}$$

* Corresponding author. *Email address:* fazeli@yazd.ac.ir

then, starting with an initial approximation $x^{(1)}$ of the solution vector x, generate a sequence of approximation $\{x^{(k)}\}$ iteratively defined by

$$x^{(k+1)} = Bx^{(k)} + r \quad k = 1, 2, \cdots \tag{2}$$

With a hope that under certain mild conditions, the sequence $\{x^{(k)}\}$ converges to the solution as $k \to \infty$.

To solve the linear system $Ax = b$ by an iterative method we need to know:

1. How to convert the system $Ax = b$ in the form of $x = Bx + r$ and
2. Which choice of x makes iteration converge faster [7].

In this paper, the iterative FOM method is described and for faster convergence some preconditions are applied for this method and the results of FOM method compare with GMRES method. Section 2 reviews some fundamental concepts of the FOM method and GMRES method with basic idea of ITALU and MERLU preconditions and their algorithms. In sections 3 we use preconditioners for these methods. The results of methods with preconditioners for some matrix and comparison of the results are in section 4 and section 5 is conclusion.

2. BASIC DEFINITIONS

2.1 Notations

We will often need to extract lower and upper triangular parts of matrices. Given an $N \times N$ matrix X, we denote ^{LT} the strict lower triangular part of X, and ^{UT} the upper triangular part of X (which includes the diagonal of X). ^{LTD} is the lower triangular part of X and replacing its diagonal entries by ones. The inner product of two $N \times N$ matrices is defined as:

$$\langle X, Y \rangle = Tr(Y^T X) \tag{3}$$

$||v||$ denotes the Euclidean norm of vector v. The Frobenius norm $||.||_F$ is the 2-norm associated with this inner product, i.e.

$$\|X\|_F = [Tr(X^T X)]^{1/2} \tag{4}$$

Condition number of matrix A is defined as:

$$Cond(A) = \|A\|\|A^{-1}\| \tag{5}$$

Matrix A is a well-condition matrix if Cond(A)$^{\approx}$ 1 [15].

2.2. Approximate LU Factorization

For approximate factorization in form of:

$$A = LU + R \tag{6}$$

such that matrix R is error matrix of the approximate factorization, our goal is to minimize matrix R, i.e. finding sparse matrices $\acute{L}$ and $\acute{U}$ such that $\acute{L}\acute{U}$ are a better pair of factors than LU. By replacing $\acute{L} = L + X_L$ and $\acute{U} = L + X_U$ the new error for the new factors is:

$$A-(L+X_L)(U+X_U)=(A-LU)-X_LU-LX_U-X_LX_U. \quad (7)$$

We would like new error be equal to zero. By replacing the matrix (A-LU) by R , we have:

$$X_LU+LX_U+X_LX_U-R=0. \quad (8)$$

By neglecting the quadratic term X_LX_U, we have to solve the nonlinear system 8 in form of:

$$X_LU+LX_U-R=0. \quad (9)$$

For solving system 9, we should perform two phases

1-We will consider a few approaches in the spirit of approximate inverse-type techniques which try to and sparse triangular factors X_L, X_U that approximately minimize the Frobenius norm of the left-hand side of 9. 2-We exploit an alternating procedure at the matrix level which fixes U (i.e. we set $X_U=0$) and solves for X_L, and then fixes the resulting L and solves for X_U and repeat the process until R converges to zero [4] and [14].

2.3. Some Iterative Methods for solving linear systems

2.3.1. Generalized Minimum Residual Method

The *n*th Krylov subspace for linear system $Ax=b$ and normalized b (i.e. $\|b\|=1$) is

$$\kappa_m(A,b)=Span\{b,Ab,A^2b,\cdots,A^{m-1}b\} \quad (10)$$

GMRES approximates the exact solution of $Ax=b$ by the vector $x_n\epsilon\kappa_n$ that maximizes the Euclidean norm of the residual $r_n=Ax_n-b.$

The vectors $b,Ab,\cdots,A^{n-1}b$ might be almost linearly dependent, so instead of this basis, the Arnoldi iteration is used to find orthonormal vectors $q_1,\cdots,q_n$ which form a basis for κ_n. Hence, the vector $x_n\epsilon\kappa_n$ can be written as $x_n=Q_ny_n$ with $y_n\epsilon R^n$ where Q_n is the m-by-n matrix formed by $q_1,\cdots,q_n$ The Arnodli process also produces an m+1-by-n upper Hessenberg matrix H_m with

$$AQ_n=Q_{n+1}H_m$$

Because columns of Q_n are orthogonal, we have

$$\|Ax_m-b\|=\|H_m-\beta e_1\|$$

Where e_1 is the first vector in the standard basis of R^{n+1}, $\beta=\|b-Ax_0\|$ and x_0 is the first trial vector (usually zero). Hence, x_n can be found by minimizing the Euclidean norm of the residual [12], [10] and [15]. Algorithm 2.1 is the GMRES method algorithm.

Algorithm 2.1: Generalized Minimum Residual Method
Input: Matrix A, vectors b and x_0 , dimension m

1. Compute $r_0 = b - Ax_0$, $\beta = \|r_0\|_2$, $v_1 = r_0/\beta$
2. For j=1,2,. . . ,m Do
3. Compute $w_j = Av_j$
4. For i=1,..,j
5. $h_{ij} = (w_j, v_i)$
6. $w_j = w_j - h_{ij}v_i$
7. End Do
8. $h_{j+1,j} = \|w_j\|_2$. if $h_{j+1,j} = 0$ Set $m = j$ and goto 11
9. $v_{j+1} = w_j/h_{j+1,j}$
10. End Do
11. Define the $(m+1) \times m$ Hessenberg matrix $\bar{H}_m = \{h_{ij}\}_{1\leq i\leq m+1, 1\leq j\leq m}$
12. Compute y_m the minimize of $\|\beta e_1 - \bar{H}_m y\|_2$ and $x_m = x_0 + V_m y_m$

2.3.2. Full Orthogonalization Method

Given an initial vector x_0 to the original linear system Ax=b we now consider an orthogonal projection method, which takes $\kappa = \kappa_m(A, r_0)$, with

$$\kappa_m(A, r_0) = Span\left\{r_0, Ar_0, A^2r_0, \cdots, A^{m-1}r_0\right\} \tag{11}$$

in which $r_0 = b - Ax_0$. This method seeks an approximate solution x_m from the affine subspace $x_0 + \kappa_m$ of dimension m by imposing the Galerkin condition

$$b - Ax_m \perp \kappa_m. \tag{12}$$

If $v_1 = \frac{r_0}{\|r_0\|_2}$ in Arnoldi's method and we set $\beta = \|r_0\|_2$, then

$$V_m^T A V_m = H_m \tag{13}$$

and

$$V_m^T r_0 = V_m^T(\beta v_1) = \beta e_1. \tag{14}$$

As a result, the approximate solution using the above m-dimensional subspaces is given by

$$x_m = x_0 + V_m y_m$$

where

$$y_m = H_m^{-1}(\beta e_1).$$

A method based on this approach is called the Full Orthogonalization Method (FOM) [12] and [15]. Algorithm 2.2 is the FOM method algorithm.

Algorithm 2.2: Full Orthogonalization Method
Input: Matrix A , Vectors b and x_0, dimension m

1. Compute $r_0 = b - Ax_0$, $\beta = \|r_0\|_2$, $v_1 = r_0/\beta$
2. Define $m \times m$ matrix $H_m = \{h_{i,j}\}_{i,j=1,\cdots,m}$; Set $H_m = 0$
3. For j=1,2,...,m Do
4. Compute $w_j = Av_j$
5. $Fori = 1, \cdots, j$
6. $h_{ij} = (w_j, v_i)$
7. $w_j = w_j - h_{ij}v_i$
8. End Do
9. Compute $h_{j+1,j} = \|w_j\|_2$. if $h_{j+1,j} = 0$ Set $m = j$ and goto 12
10. Compute $v_{j+1} = w_j/h_{j+1,j}$
11. End Do
12. Compute $y_m = H_m^{-1}(\beta e_1)$ and $x_m = x_0 + V_m y_m$

2.4. Preconditioning

Preconditioning is a procedure of a transforming the problem conditions into a form that is more suitable for numerical solution and solving the problem mathematically. Preconditioning is typically related to reducing a condition number of the problem. Preconditioners are also useful in iterative methods to solve a linear system $Ax = b$ for x since the rate of convergence for most iterative linear solvers increases as the lower condition number of a matrix. Preconditioned iterative solvers are typically used for large and especially sparse matrices.

2.4.1. Approximate Inverses Techniques

Our goal is to find sparse matrix M such that minimize $\|I - AM\|_F$.

Let *P* be a preconditioner of *A*, we can obtain a matrix M such that *AM* approximates $P = LU$ by trying to approximately minimize $\|P - AM\|_F$.

Matrix *M* can be computed approximately by solving the linear systems $Am_i = p_i$ where p_i is the jth column of *P*. Becuase p_i is sparse, so we can fnd ith column of M such that it is sparse. One method for solving linear systems $Am_i = e_i$ alternatively, is to update the whole matrix in each iteration. For example, we can obtain M_{new} by

$$M_{new} = M + \alpha S \tag{15}$$

where matrix *S* is a search direction matrix and the scalar α is selected to minimize the objective function associated with M_{new} We can change the problem in the form:

$$F(M) = \|P - AM\|_F^2 \tag{16}$$

where F(M) is a quadratic function on the space of $N \times N$ matrices. The simplest choice of matrix *S* is the residual matrix $R = P - AM$ [11], [6] and [5].

2.4.2. Alternating L-U descent methods (MERLU)

In equation 8, if we choose $X_L = 0$, i.e. U updates while *L* is kept frozen, then sparse matrix X_U is obtained that minimizes

$$F(X_U) = \|A - L(U + X_U)\|_F^2 = \|R - LX_U\|_F^2 \tag{17}$$

Where $R = A - LU$ is the current factorization error.

The following section will be exploited that the optimum X_U is $L^{-1}R$. We have

$$\|R - LX_U\|_F^2 = Tr([R - LX_U]^T[R - LX_U]) = \|R\|_F^2 - 2Tr(R^T LX_U) + \|LX_U\|_F^2. \tag{18}$$

The gradient matrix of $F(X_U)$ at $X_U = 0$ is $G = -2L^T R$. This means when X_U is a small multiple of $L^T R$ then X_U will decrease $F(X_U)$ Now a steepest descent method can be devised but $G = L^T R$ is not necessarily upper triangular, hence X_U will not be upper triangular as desired. We use operation $[.]_{UT}$ for $L^T R$ and define G as

$$G = [L^T R]_{UT} \tag{19}$$

which is the upper triangular part of $L^T R$

For choosing the best in 15, we should minimize the quadratic form:

$$\|R - \alpha LG\|_F^2 = Tr\left([R - \alpha LG]^T [R - \alpha LG]\right) = \|R\|_F^2 - 2\alpha Tr(R^T LG) + \alpha^2 \|LG\|_F^2 \tag{20}$$

yielding

$$\alpha_U = \frac{\langle R, LG\rangle}{\langle LG, LG\rangle} = \frac{Tr(R^T LG)}{Tr((LG)^T LG)}. \tag{21}$$

The L part can be obtained by repeating the above argument. If we replace the objective function 17 by

$$\|A - (L + X_L)U\|_F^2 = \|R - X_L U\|_F^2 \tag{22}$$

the gradient array becomes $-2RU^T$ We obtained a correction strict lower triangular X_L to L and $G = [L^T R]_{UT}$ so we have

$$\alpha_L = \frac{\langle R, G_L U\rangle}{\langle G_L U, G_L U\rangle} = \frac{Tr(R^T G_L U)}{Tr((G_L U)^T G_L U)}. \tag{23}$$

The Algorithm 2.3 shows the steps of this method.

Algorithm 2.3: MERLU (Minimal Energy Residual descent for LU)
Input: Initial Lower Triangular matrix and Upper Triangular matrix U

1. Select an initial pair L,U (with $L = L_{LTD}, U = U_{UT}$)
2. Until convergence Do
3. Compute $R := A - LU$
4. Compute $G = [L^T R]_{UT}$
5. Apply numerical dropping to G
6. Compute $\alpha = \langle R, C\rangle / \|C\|_F^2$,Where $C = wG$
7. Compute $U := G + \alpha U$
8. Compute $R := A - LU$
9. Compute $G = [RU^T]_{UT}$
10. Apply numerical dropping to G
11. Compute $\alpha = \langle R, C\rangle / \|C\|_F^2$,where $C = GU$
12. Compute L:=L+αG
13. EndDo

2.4.3. Alternating Sparse-Sparse Iteration (Italu)

In Equation 8, we set $X_U = 0$. If U is nonsingular we have

$$X_L U = R \rightarrow X_L = RU^{-1}. \tag{24}$$

We can obtain the correction to L by solving system $U^T X_L^T = R^T$ but the updated matrix $L + X_L$ is not necessarily unit lower triangular therefore we use operation $[.]_{LT}$ for $L + X_L$ or the operation
$[.]_{LTD}$ for X_L We repeat this procedure by freezing U and updating L and vice versa alternatively. This iteration is summarized in the following two equations:

$$U_{k+1} = U_K + [L_k^{-1}(A - L_k U_k)]_{UT} \tag{25}$$

and

$$L_{k+1} = L_K + [(A - L_k U_{k+1})U_{k+1}^{-1}]_{UT} \tag{26}$$

The Algorithm 2.4 shows the steps of this method.

Algorithm 2.4: ITALU (Iterative Threshold Alternating Lower-Upper Correction)
Input: initial matrix Lower TriangularL_0 and matrix upper Triangular U_0

1. Given: A ,U_0 , L_0(with $U_0 = [U_0]_{UT}$; $L_0 = [L_0]_{LTD}$)
2. For k=0,...,Do
3. Compute $R_k = A - L_k U_k$
4. Compute $X_U = [L_k^{-1} R_k]_{UT}$
5. Apply numerical dropping to X_U
6. $U_{k+1} = U_k + X_U$
7. If $det(U_{k+1}) == 0$ Abort 'Singular U reached'
8. Compute $R_{k+1/2} = A - L_k U_{k+1}$
9. Compute $X_L = [R_{k+1/2} U_{K+1}^{-1}]_{LT}$
10. Apply numerical dropping to X_L
11. $L_{k+1} = L_k + X_L$
12. EndDo

3. USING PRECONDITIONS WITH FOM METHOD

For using methods we first compute matrix M by using preconditioner. In preconditioner ITALU(1) we run ITALU algorithm two times and the output of first running time is used for L_0 and U_0 in next iteration. The preconditioner MERLU(1) also have the same definition [2] and [1].

3.1. Left-Preconditioned FOM

The left preconditioned FOM algorithm defines as the FOM algorithm applied to the system

$$M^{-1}Ax = M^{-1}b.$$

The straightforward application of FOM to the above linear system yields the following preconditioned version of FOM. The Algorithm 3.1 is FOM method with left preconditioned algorithm.

Algorithm 3.1: FOM Method with Left Preconditioning
Input: Matrix A , Vectors b and x_0, dimension m, precondition M

1. Compute $r_0 = M^{-1}(b - Ax_0)$, $\beta = \|r_0\|_2, v_1 = r_0/\beta$
2. Define $m \times m$ matrix $H_m = \{h_{i,j}\}_{i,j=1,\cdots,m}$; Set $H_m = 0$
3. For $j = 1, 2, \cdots, m$ Do
4. Compute $w_j = M^{-1}Av_j$
5. For $i = 1, 2, \cdots, j$
6. $h_{ij} = (w_j , v_i)$
7. $w_j = w_j - h_{ij}v_i$
8. End Do
9. Compute $h_{j+1,j} = \| w_j \|_2$. if $h_{j+1,j} = 0$ Set $m = j$ and goto 12
10. Compute $v_{j+1} = w_j/h_{j+1,j}$
11. End Do
12. Compute $y_m = H_m^{-1}(\beta e_1)$ and $x_m = x_0 + V_m y_m$

The Arnoldi loop constructs an orthogonal basis of the left preconditioned Krylov subspace

$$Span\left\{r_0, M^{-1}Ar_0, , (M^{-1}A)^{m-1}r_0\right\} \tag{27}$$

All residual vectors and their norms that are computed by the algorithm correspond to the preconditioned residuals, namely $z_m = M^{-1}(b - Ax_m)$ instead or the original (unpreconditioned) residuals $b - Ax_m$

We compare the results of this algorithm with GMRES method with Left Preconditioning results. For more detail [4].

4. RESULTS

For each matrix described in table.1 we run Algorithms FOM with left preconditioner and GMRES with left preconditioner for preconditioner MERELU (1) and table.2 shows performance of this preconditioner and figure.1 shows coverage rate of FOM and GMRES method with MERELU(1) pereconditioner.

Relative tolerance is set to *droptol* = 0.2 and Matlab's estimated condition number yields $cond(A) \approx 4.03E + 05$.

The results show the FOM method with left preconditioner MERELU(1) converges faster than GMRES method with left preconditioner MERELU(1).

In next execution we run FOM and GMRES methods with left preconditioner ITALU(1) for each matrix described in table.3 and table.4 shows performance of this preconditioner and figure.2 shows coverage rate of FOM and GMRES method with ITALU(1) pereconditioner.

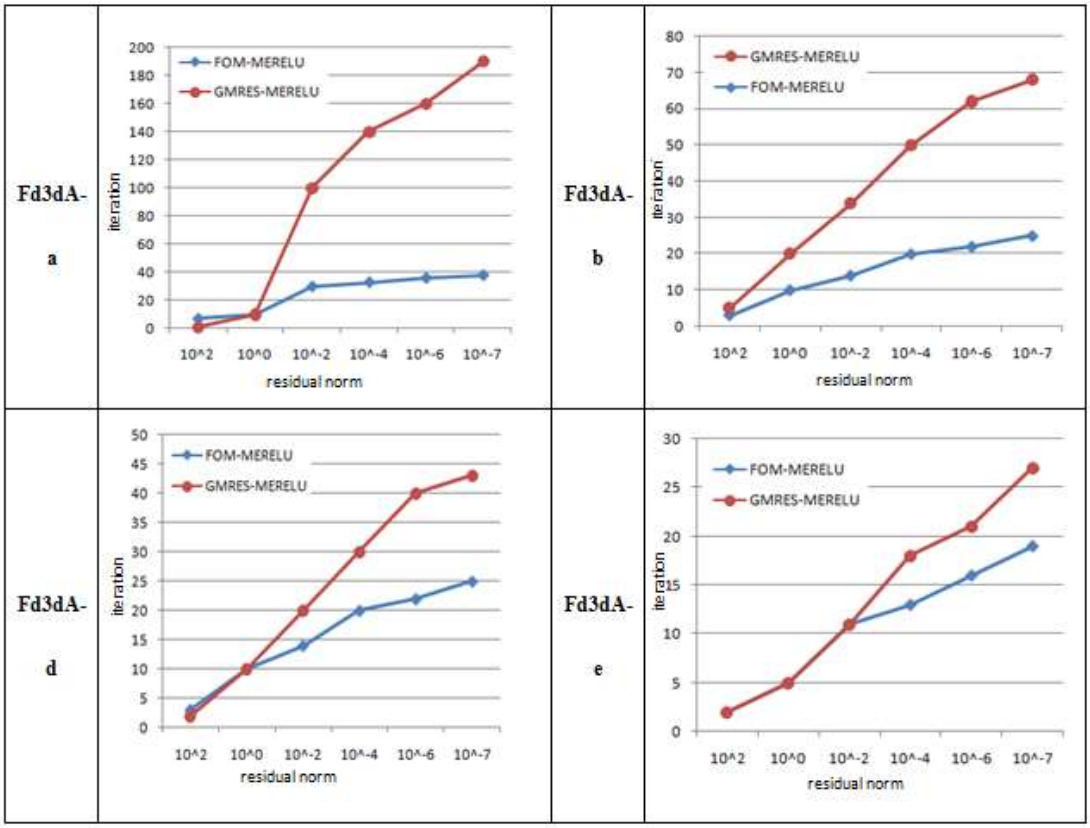

Figure 1. Converge rate of MERELU(1) preconditioner

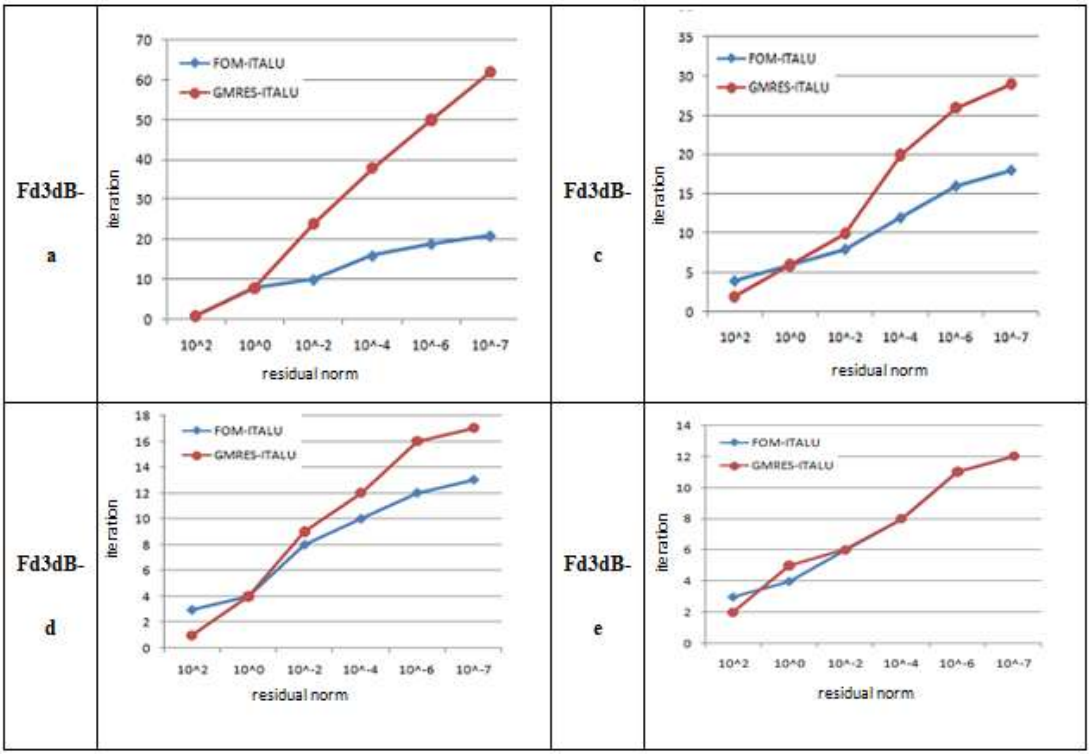

Figure 2. Converge rate of ITALU(1) preconditioner.

Table 1. Properties Matrix.

No	Name	n_x	n_y	n_z	a_x	a_y	a_z	shift	size	Condest
1	Fd3dA-a	15	10	10	0.1	0.1	0.1	0.5	1500×1500	2.2776e+003
2	Fd3dA-b	15	10	10	-0.1	0.1	0.2	0.3	1500×1500	1.4804e+003
3	Fd3dA-d	10	10	10	0.3	0.2	-0.2	0.5	1000×1000	2.4336e+003
4	Fd3dA-e	10	10	10	0.1	0.4	0.1	0.3	1000×1000	1.1814e+003

Table 2. Performance of MERELU(1) preconditioner for FOM and GMRES methods.

		MERELU(1)-FOM		MERELU(1)-GMRES	
No	Name	iter	residual	iter	residual
1	Fd3dA-a	36	2.259e-007	201	1.737e-006
2	Fd3dA-b	24	3.494e-007	42	3.761e-007
3	Fd3dA-d	23	1.734e-007	32	3.499e-007
4	Fd3dA-e	19	2.100e-007	27	2.809e-007

Table 3. Properties Matrix.

No	Name	n_x	n_y	n_z	a_x	a_y	a_z	shift	size	Condest
1	Fd3dB-a	10	15	10	0.1	0.1	0.1	0.5	1500×1500	2.2776e+003
2	Fd3dB-c	10	10	10	0.1	0.5	0.1	-0.5	1000×1000	7.2148e+003
3	Fd3dB-d	10	10	10	0.2	0.2	0.2	0.5	1000×1000	1.8625e+003
4	Fd3dB-e	10	10	10	0.1	0.0	0.0	0.3	1000×1000	586.6568

Table 4. Performance of ITALU(1) preconditioner for FOM and GMRES methods.

		ITALU(1)-FOM		ITALU(1)-GMRES	
No	Name	iter	residual	iter	residual
1	Fd3dB-a	21	1.661e-007	62	6.308e-007
2	Fd3dB-c	18	3.931e-007	29	2.410e-007
3	Fd3dB-d	13	1.971e-007	17	4.349e-007
4	Fd3dB-e	12	7.500e-008	12	1.536e-007

The results show the FOM method with left preconditioner ITALU(1) converge equal or faster than GMRES method with left preconditioner ITALU(1).

5. CONCLUSION

In this paper, we reviewed iterative GMRES and FOM methods for solving linear system Ax=b and prescribe ITALU and MERELU preconditioners for decreasing condition number of system. The results showed that the FOM method with left preconditioner MERELU(1) converges faster than GMRES method with left preconditioner MERELU(1) and FOM method with left precon-ditioner ITALU(1) converges equally or faster than GMRES method with left preconditioner ITALU(1).

REFERENCES

1. M. Benzi, Preconditioning techniques for large linear systems. Journal of Computational Physics, 182, 418-477 (2002).
2. M. Benzi, C.D. Meyer & M. Tuma, A sparse approximate inverse preconditioner for the conjugate gradient method. SIAM Journal on Scienti_c Computing, 17, 1135-1149 (1996).
3. P. Birken, J.D. Tebbens, A. Meister & M. Tuma,Preconditioner updates applied to CFD model problems. Applied Numerical Mathematics, 58, 1628-1641 (2008).
4. C. Calgaro, J.P. Chehab & Y. Saad, Incremental incomplete LU factorizations with applications. Numerical Linear Algebra with Applications, 17, 811-837 (2010).
5. E. Chow, Y. Saad, Approximate inverse preconditioners via sparse-sparse iterations. SIAM Journal on Scienti_c Computing, 19, 995-1023 (1998).
6. E. Chow, Y. Saad, Approximate inverse techniques for block-partitioned matrices.SIAM Journal on Scientific Computing, 18, 1657-1675 (1997).
7. B. Datta, Numerical Linear Algebra and Applications, Second Edition (2010). SIAM
8. T.A. Davis, Direct Methods for Sparse Linear Systems (2006). SIAM: Philadelphia, PA.
9. H.C. Elman, A stability analysis of incomplete LU factorizations. Mathematics of Com-putation, 17, 191-217 (1986).
10. Essai, Weighted FOM, GMRES for solving nonsymmetric linear systems. NumericalAlgorithms, 18, 277-292 (1998).
11. R.M. Holland, A.J. Wathen & G. Shaw, Sparse approximate inverses, target matrices. SIAM Journal on Scienti_c Computing, 26, 1000-1011 (2005).
12. Y. Saad, H. Martin, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM journal on scientific, statistical computing, 7, 856-869 (1986).
13. Y. Saad, Preconditioning techniques for nonsymmetric, inde_nite linear systems. Journal of Computational , Applied Mathematics , 24, 89-105 (1988).
14. Y. Saad, ILUT: A dual threshold incomplete LU factorization. Numerical linear algebra with applications, 1, 387-402 (1994).
15. Y. Saad, Iterative methods for sparse linear systems (2003). SIAM: Philadelpha, PA.