مجلة جامعة تشرين للبحوث والدراسات العلمية – سلسلة العلوم الأساسية المجلد (30) العدد (4) 2008
Tishreen University Journal for Research and Scientific Studies - Basic Sciences Series Vol. (30) No. (4) 2008

Comparison of some Preconditioned Krylov Methods for Solving Sparse Non-symmetric Linear Systems of Equations

Dr. Ahmad Al-Kurdi [*]

(Received 24 / 1 / 2008. Accepted 26/5/2008)

□ ABSTRACT □

Large sparse non-symmetric linear systems of equations often occur in many scientific and engineering applications. In this paper, we present a comparative study of some preconditioned Krylov iterative methods, namely CGS, Bi-CGSTAB, TFQMR and GMRES for solving such systems. To demonstrate their efficiency, we test and compare the numerical implementations of these methods on five numerical examples. The preconditioners considered here are incomplete LU-decomposition (ILU), Symmetric Successive Over Relaxation (SSOR), and Alternating Direction Implicit (ADI). The ILU preconditioner is shown to be extremely effective in achieving optimal convergence rates for the class of problems considered here. Finally, our results show that the GMRES is the best among the considered iterative methods.

Keywords: Nonsymmetric linear system, Krylov subspace methods, preconditioning.

[*]**Assistant Professor, Department of Mathematics, Faculty of Science, Al-Baath University, Homs, Syria.**

مجلة جامعة تشرين للبحوث والدراسات العلمية – سلسلة العلوم الأساسية المجلد (30) العدد (4) 2008
Tishreen University Journal for Research and Scientific Studies - Basic Sciences Series Vol. (30) No. (4) 2008

مقارنة بعض طرائق كريلوف المسرعة لحل جمل المعادلات الخطية غير المتناظرة كثيرة الأصفار

الدكتور أحمد الكردي[*]

(تاريخ الإيداع 24 / 1 / 2008. قُبِل للنشر في 2008/5/26)

□ الملخّص □

تظهر جمل المعادلات الخطية ، غير المتناظرة ، كثيرة الأصفار ، من مراتب عليا. في العديد من التطبيقات العلمية والهندسية. في هذا العمل، نجري دراسة مقارنة لبعض طرائق كريلوف التكرارية المسرعة وهي الطرائق CGS و Bi-CGSTAB و TFQMR و GMRES لحل هذا النوع من جمل المعادلات. المسرعات المدروسة هنا هي تحليل LU غير التام (ILU) و المسرع فوق الاسترخاء المتتالي التناظري (SSOR) و مسرع الاتجاهات الضمنية المتناوبة (ADI). تبين أن المسرع ILU فعال إلى حد كبير في إنجاز معدلات تقارب مثلى من أجل صف المسائل المدروسة هنا. أخيرا، تبين تجاربنا أن طريقة **GMRES(10)** كانت هي الأفضل بين جميع الطرائق التكرارية المدروسة.

الكلمات المفتاحية: جملة خطية غير متناظرة ، طرائق فضاء كريلوف ، التسريع.

[*] مدرس – قسم الرياضيات – كلية العلوم – جامعة البعث – حمص – سورية.

Introduction:

Consider the linear system of equations

$$Ax = b \qquad (1)$$

where A is a large, non-singular, sparse, non-symmetric matrix of order $n \times n$ and b is a given vector of order n. Such systems often occur in many scientific and engineering applications [1]. For such systems iterative methods are usually preferred to direct methods which are expensive both in memory and computing requirements. There are now quite a number of Krylov subspace methods available for solving (1), e.g., CGS, Bi-CGSTAB, TFQMR and GMRES; for more details see [1,2] and references therein. In order to be effective, these methods must be combined with a good preconditioner, and it is generally agreed that the choice of the preconditioner is even more crucial than the choice of the Krylov iteration method. The preconditioner application in the iteration loop is the most delicate part of the iterative process. When the iterative method is based on Krylov subspaces, there is a need to use preconditioning techniques in order to achieve convergence in a reasonable number of iteration steps.

Since using a preconditioner in an iterative method incurs some extra cost, both initially for the setup, and per iteration for applying it, there is a trade-off between the constructing cost and applying the preconditioner, and the gain in convergence speed. Certain preconditioners need little or no construction phase at all (for instance SSOR preconditioner) but for others, such as incomplete factorizations, there can be substantial work involved. Broad classes of preconditioners are based on incomplete factorization of the coefficient matrix A.

In this work, we have focused our attention to make a detail comparative study of the preconditioned CGS, Bi-CGSTAB, TFQMR and GMRES for solving (1). The considered preconditioners are ILU(m) (incomplete LU decomposition of level m) [3], SSOR and ADI [1]. The performance of each iterative method combined with one of the mentioned preconditioners is measured in terms of CPU time and number of iterations.

Importance and Parts of the Research:

Systems of linear equations given in (1) are frequently encountered in almost all the scientific and engineering applications such as physics, mechanics, signal processing and other applications of real life problems. For these reason we attempt to develop iterative methods for solving such systems of linear equations.

1. Preconditioned Krylov subspace methods

The spectrum of the coefficient matrix A governs the rate of convergence of any iterative method. By the use of preconditioning, this spectrum can be narrowed which then promises much better convergence properties for iterative methods. As will be seen, the choice of a preconditioner is crucial for the behavior of any iterative method, since it influences the speed of convergence and their stability.

Let x_0 be an arbitrary initial guess for the linear system given by (1) and let $r_0 = b - Ax_0$ be the corresponding residual vector. A Krylov subspace method is an iterative scheme, which for an arbitrary choice of x_0, seeks approximate solutions of the form $x_k \in x_0 + K_k(A, r_0)$, where $K_k(A, r_0)$ is the Krylov subspace $K_k(A, r_0) = \text{span}\{r_0, Ar_0, A^2 r_0, \ldots, A^{k-1} r_0\}$. The motivation for preconditioning is to speed up the convergence of an iterative solution of (1). Usually a preconditioner M_1 or M_2 is used to multiply (1) on the left / right respectively so that the preconditioned system is

$$M_1^{-1}AM_2^{-1}\left(M_2^{-1}x\right)=M_1^{-1}b \qquad (2)$$

where M_1 and M_2 are left and right preconditioning matrices respectively. In this work, the GMRES method is used with the right preconditioners.

2. Storage scheme

A short representation of storage technique described here is called as compressed storage scheme [3]. The storage format is the most general; it makes absolutely no assumptions about the sparsity structures of the matrix, and it does not store any unnecessary elements. Storing given matrix A with compressed storage scheme requires three one dimensional arrays **VA**, **JA**, and **IA** of length **na**, **na** and **n+1**, where **n** is the number of rows and **na** is the total number of non-zero elements in the matrix A. The array **VA** contains the non-zero elements of A stored row-by-row, **JA** contains the column indices which corresponds to the non-zero elements in the array **VA**, and **IA** contains **n+1** pointers which delimit the rows of non-zero elements in the array **VA**, as illustrated below: For example, let A be a square matrix of order **5**.

$$A=\begin{bmatrix} a_{11} & 0 & a_{13} & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 \\ 0 & 0 & a_{33} & 0 & 0 \\ a_{41} & 0 & 0 & a_{44} & a_{45} \\ 0 & a_{52} & 0 & 0 & a_{55} \end{bmatrix} \qquad (3)$$

The arrays **VA** , **JA** and **IA** are

$$\begin{array}{llllllllllll} VA= & \overbrace{a_{11}\ a_{13}}^{row1} & \overbrace{a_{21}\ a_{22}\ a_{23}}^{row2} & \overbrace{a_{33}}^{row3} & \overbrace{a_{41}\ a_{44}\ a_{45}}^{row4} & \overbrace{a_{52}\ a_{55}}^{row5} \\ JA= & 1\quad 3 & 1\quad 2\quad 3 & 3 & 1\quad 4\quad 5 & 2\quad 5 \\ IA= & 1\quad 3 & 6\quad 7 & 10 & 12 & \end{array} \qquad (4)$$

By convention, we define IA[n+1]= $na+1$. The storage savings for this approach is significant. Instead of storing n^2 elements, we need only $2na+n+1$ storage locations. The matrix (3) can not be factored, by using the above storage scheme, “in place” unless fill-ins are accounted for when storage is created. For example, when (3) is factored, non-zero numbers are assigned to a_{42} and a_{53}, but neither of these elements appears in (4) as illustrated, i.e., there is need to reallocating storage to make room for the fill-ins. In this work we present a good choice for predicting fill-ins, using powers of a Boolean matrix.

3. Preconditioning

In this section, we discuss the use of preconditioning with the aim of reducing the iteration steps required to obtain a good approximation to the solution of (1). It is clear that the preconditioned iterative method to be successful, it is important that we choose the correct preconditioner. We want to choose a preconditioner so that condition number $k\left(M^{-1}A\right)\approx 1$. The job of choosing the correct preconditioner is not easy. There are many ways to choose a preconditioner. Usually, each of these different preconditioners works best in some situation. All of these preconditioners approximate A^{-1} to some degree. If we could choose a matrix M such that $M^{-1}A=I$, we could have $k\left(M^{-1}A\right)=1$ and the solution is attained in one step. Of course, we can not do this, because amount of work involved in getting A^{-1} will be excessively high in practical circumstances. Consequently, preconditioning must be regarded as a trade off between the cost of constructing and manipulating the preconditioner, and the acceleration of the iterative process. Most preconditioners take in their application an amount of work proportional to the number of variables. This implies that they multiply the work per iteration by a constant factor. On the other hand, the number of iterations as a function of the matrix size is usually improved by a constant. Certain preconditioners are able to improve on this situation, most notably of them is the incomplete LU-decomposition.

(1). Incomplete LU Preconditioner (ILU)

Incomplete LU decomposition (ILU) is based on the LU-decomposition of the coefficient matrix A. In constructing an ILU(m) (ILU of level m), we use The Powers of a Boolean Matrix Strategy (PBS) for determining the pattern of non-zero elements of the factors LU of a given matrix A; for more details, we refer the reader to [3].

Algorithm 1. PBS (The Powers of a Boolean Matrix Strategy)

```
Step 1.
    Form the matrix $B = \lfloor b_{ij} \rfloor$ as
    $b_{ij} = \begin{cases} 1, & \text{if } a_{ij} \neq 0 \\ 0, & \text{otherwise} \end{cases}$
Step 2
    Compute $B^{2^m}, (m \geq 1)$.
    If $B^{2^m} = B^{2^{m+1}}$, then
        Form the set $P = \{ (i,j) : b_{ij}^{(2^m)} = 1 \}$.
    Else $m = m+1$, and go to Step 2.
```

From the Algorithm 1, it follows that the sparsity pattern of $M = LU$ is approximately equal to that of B^{2^m}. At any given iteration, if the calculated Boolean matrix agrees with the matrix at the previous iteration, i.e. $B^m = B^{m+1}$, then the process has converged and we have the sparsity pattern of the LU factors. If $B^m = B^{m+1}$, we get the complete LU factors. Sparse LU - decomposition method, may give the solution. However, using the complete LU factors (when $B^m = B^{m+1}$) is not desirable because it would defeat the purpose of using an iterative method. Since the preconditioner can be improved by allowing more fill-ins and its effectiveness depends on how well M^{-1} approximates A^{-1}, the Algorithm 1 can be terminated at a level m such that there is trade - off between the computational requirements (both in terms of memory and CPU time) and reducing the number of iterations. We show the efficiency of this method in section of numerical experiments by taking numerical examples.

Once the non-zero structure of L and U matrices is obtained using Algorithm 1, non-zero entries are then obtained by Doolittle's method [1], where all the diagonal entries of L are 1.

$A = LU$ gives

$$a_{ij} = \sum_{k=1}^{\min(i,j)} l_{ik} u_{kj} \qquad (5)$$

This gives the following explicit formulas for l_{ij} and u_{ij}:

$$l_{ik} = \frac{a_{ik} - \sum_{j=1}^{k-1} l_{ij} u_{jk}}{u_{kk}}, \qquad i > k \qquad (6)$$
$$u_{ik} = a_{ik} - \sum_{j=1}^{i-1} l_{ij} u_{jk}, \qquad i \le k$$

While making an incomplete **LU** – factorization, we need to store only non –zero entries of L and U. We define an extra array **Diag [1…n]** which points to the diagonal elements of U in the array **VA**. The non –zero structure P of L and U is stored in **JA, IA** and **VA** containing $a_{ij} \neq 0$ as well as fill-ins. The following algorithm calculates the incomplete decomposition. The Boolean variable **revise** is false for the standard incomplete decomposition and true for the modified version such that row sums of the rest matrix $R = A - LU$ are equal to zero. The array **Point [1…n]** is an array of integers which points to the entries in L and U of row **i.**

Algorithm 2. ILU (The incomplete LU- decomposition)

```
For i = 1 To n Do
   Point [ i ] = 0;
For i = 2 To n Do
  {
     For v = IA [ i ]+1 To IA [ i+1 ]-1 Do
        Point [ JA [ v ] ] = v;
For v = IA [ i ] To Diag [ i ]-1 Do
       {
          j = JA [ v ] ;
          VA [ v ]  = VA [ v ] / VA [ Diag [ j ] ] ;
          For w = Diag [ j ]+1 To IA [ j+1 ] -1 Do
            {
                k = Point [ JA [ w ] ] ;
                If ( k>0 ) then
                   VA [ k ] = VA [ k ] – VA [ v ] * VA [ w ] ;
                Else
                If ( revised ) then
                  VA [ Diag [ i ] ] =VA [ Diag [ i ] ] –VA [ v ]*VA [ w ] ;
            }//End For w.
       }//End For v.
     For v = IA [ i ]+1 To IA [ i+1 ]-1 Do
        Point [ JA [ v ] ] = 0 ;
  }//End For i.
```

The choice of P is extremely important. In practice, the non-zero pattern of L and U is often taken the same as that of the original matrix. This has advantage that no additional storage space is needed for the non-zero structure of the incomplete decomposition. The ILU(m) decomposition is based on the structural strategy outlined above for accepting fill-ins only to a certain level m. A level function is used in incomplete factorization to control the number of fill elements.

The Algorithm 2 for computing L and U with PBS does produce an optimal preconditioner.

(2). Symmetric Successive Over Relaxation (SSOR)

Iterative methods that can be expressed in the simple form

$$x_{l+1} = Nx_l + C \qquad (7)$$

where neither N nor C depends upon the iteration count l, are called stationary iterative methods. The stationary iterative methods, e.g., SSOR, and ADI, are rarely competitive with Krylov iterative methods. So, all these methods are often employed as preconditioners for nonstationery iterative methods. Since the SSOR scheme is a potential solver for the problem, it should be clear that the SSOR scheme would provide us with an approximation of A^{-1}. Let

$$A = L_A + D_A + U_A \qquad (8)$$

be the splitting of A into strictly lower, diagonal, and strictly upper triangular matrices of A. To find a preconditioner, we must find a matrix M as approximation to A. The residual correction method is to approximate A^{-1} and define the iteration

$$u_{l+1} = u_l + Nr_l \qquad (9)$$

where $r_l = b - Au_l$ and N is some approximation to A^{-1}.

If $N = \omega(2-\omega)(D_A + \omega U_A)^{-1} D_A (D_A + \omega L_A)^{-1}$, we get the Symmetric Successive Over Relaxation scheme (SSOR), where ω (omega) is a free parameter. Thus, when apply SSOR, we are solving the equation $N^{-1}(u_{l+1} - u_l) = \frac{1}{\omega(2-\omega)}(D_A + \omega L_A) D_A^{-1} (D_A + \omega U_A)(u_{l+1} - u_l) = r_l$. SSOR preconditioner can be used to precondition Krylov subspace methods, such as CGS, Bi-CGSTAB, TFQMR and GMRES and it requires no construction time.

(3). Alternative Direction Implicit Preconditioner (ADI)

The Alternating Direction Implicit (ADI) method is one of the stationary iterative methods used as a preconditioner for nonsymmetric systems. Let (8) be the splitting of the coefficient matrix A. If $N = \omega(D_A + \omega U_A)^{-1} D_A (D_A + \omega L_A)^{-1}$, we get the ADI scheme, where ω (omega) is a free parameter. As before, N is some approximation to A^{-1} and L_A, D_A, and U_A represent the lower triangular, diagonal, and upper triangular parts of A respectively. The matrix N is a good choice because it is lower and upper triangular matrices. Thus, when apply ADI, we are solving the equation $N^{-1}(u_{l+1} - u_l) = \frac{1}{\omega}(D_A + \omega L_A) D_A^{-1} (D_A + \omega U_A)(u_{l+1} - u_l) = r_l$. The ADI is probably best only considered as a preconditioner in the Krylov subspace methods.

(4). The Preconditioning Step

The preconditioners M considered in this work are ILU, SSOR and ADI. It is important that M^{-1} is never explicitly computed. Alternatively, we have

$$z = M^{-1}v \;\Rightarrow\; Mz = v \qquad (10)$$

The preconditioning step (10) is an important step in any preconditioned iterative method. We now describe the solution of (10) for different cases.

(I). Case 1. The Preconditioner $M = LU$.

The system (10) can be solved using CGS, Bi-CGSTAB and GMRES by the following two steps:

Step 1: Forward substitution in $LY = b$.

Step 2: Back substitution in $UX = Y$.

where the Steps 1 and 2 are given in the Algorithms 3 and 4 respectively.

Algorithm 3. Forward Substitution.

```
Y[ 1 ] = b [ 1 ] ;
For i = 2 To n Do
  {
     sum = 0 ;
     For j = IA [ i ] To  Diag [ i ]-1  Do
       sum= sum + VA [ j ] * Y[ JA [ j ] ] ;
     Y[ i ] = b [ i ] - sum ;
  }//End For i.
```

Algorithm 4. Back Substitution.

```
X [ n ] = Y [ n ] / VA [ Diag [ n ] ] ;
For i = n-1 Down To 1 Do
 {
     sum1 = 0 ;
     For j = Diag [ i ] To IA [ i+1] -1 Do
       sum1 = sum1 + VA [ j ] * X [ JA [ j ] ] ;
     X[ i ] = ( Y [ i ] – sum1 ) / VA [ Diag [ i ] ] ;
 }//End For i.
```

The system (2) can be solved using TFQMR as follows. Let M be a given nonsingular $n \times n$ matrix which approximates in some sense the coefficient matrix A of (1). Moreover, assume that M is decomposed in the form $M = M_1 M_2$. Instead of solving the original system (1), we apply the TFQMR algorithm to the equivalent linear system $A'y = b'$, where $A' = M_1^{-1} A M_2^{-1}, b' = M_1^{-1}(b - Ax_0), y = M_2 x$. Here x_0 denotes some initial guess for the solution of (1). The iterates y_k and residual vector $r'_k = b' - A' y_k$ for the preconditioned system $A'y = b'$ are transformed back into the corresponding qualities for the original system by setting $x_k = M_2^{-1} y_k$ and $r_k = M_1 r'_k$. For a case 1, the system (2) can be solved using TFQMR by taking $M_1 = L$ and $M_2 = U$ in $M = M_1 M_2$. Thus, the system (10) can be solved by the Algorithm 3 ($M_1 = L$) and the Algorithm 4 ($M_2 = U$). The preconditioning $M = LU$ involves writing A as $A = LU - R$, with R as error term. However, the size of the entries in the error matrix for ILU(m) decreases as m increases. When solving the system using the splitting $A = LU - R$, we consider the system $(LU)^{-1} Ax = (LU)^{-1} b$. The preconditioned matrix $(LU)^{-1} A$ has to resemble the identity matrix I as closely as possible. Since $(LU)^{-1} A = (LU)^{-1}[(LU) - R] = I - (LU)^{-1} R$, then the matrix $(LU)^{-1} R$ should be as small as possible in some sense. We give three Theorems which state that $(LU)^{-1}$ is a proper approximation to A^{-1} if and only if $\left\| (LU)^{-1} R \right\|$ is sufficiently small for some matrix norm $\|\cdot\|$.

Theorem 1. Suppose the product **LU** is nonsingular and $LU - R$ is a splitting of the nonsingular $n \times n$ matrix **A** and the product **LU** is nonsingular. Then

$$\frac{\left\| (LU)^{-1} R \right\|}{\operatorname{cond}(A)} \le \frac{\left\| (LU)^{-1} - A^{-1} \right\|}{\left\| A^{-1} \right\|} \le \left\| (LU)^{-1} R \right\| \qquad (11)$$

where $\operatorname{cond}(A) = \|A\|.\|A^{-1}\|$ the condition number of A, and LU is the ILU(m) factorization.

Proof.

$$(LU)^{-1}R = (LU)^{-1}(LU - A) = I - (LU)^{-1}A = \left[A^{-1} - (LU)^{-1}\right]A$$

$$\left\|(LU)^{-1}R\right\| \le \left\|A^{-1} - (LU)^{-1}\right\|\|A\| = \frac{\left\|(LU)^{-1} - A^{-1}\right\|}{\left\|A^{-1}\right\|}\|A\|\left\|A^{-1}\right\|$$

by dividing the left and the right-hand side by $\|A\|.\|A^{-1}\|$ one obtains the first inequality of (11). The second inequality follows from th following:

$$(LU)^{-1}R = \left[A^{-1} - (LU)^{-1}\right]A$$

$$(LU)^{-1}R\,A^{-1} = \left[A^{-1} - (LU)^{-1}\right]$$

$$\left\|A^{-1} - (LU)^{-1}\right\| \le \left\|(LU)^{-1}R\right\|\left\|A^{-1}\right\|$$

After division by $\left\|A^{-1}\right\|$ the desired inequality is obtained. □

Theorem 2. If x is the solution of (1) and $\tilde{x}$ satisfies $LU\tilde{x} = b$. Then

$$\frac{\|x - \tilde{x}\|}{\|x\|} \le \left\|(LU)^{-1}R\right\| \qquad (12)$$

Proof. We know that $x - \tilde{x} = A^{-1}b - (LU)^{-1}b = \left[A^{-1} - (LU)^{-1}\right]Ax$. But

$$(LU)^{-1}R = (LU)^{-1}(LU - A) = I - (LU)^{-1}A = \left[A^{-1} - (LU)^{-1}\right]A$$

Thus, we have $x - \tilde{x} = (LU)^{-1}Rx$. Taking the norm leads to the desired.

Theorem 3. Suppose $LU - R$ is a splitting of the nonsingular $n \times n$ matrix A and $\|A^{-1}R\| < 1$. Then

$$\operatorname{cond}\left[(LU)^{-1}A\right] \le \frac{1 + \|A^{-1}R\|}{1 - \|A^{-1}R\|}$$

where LU is the ILU(m) factorization.

Proof. Suppose LUx equals the null vector **0.**

$$LUx = 0 \Leftrightarrow (A + R)x = 0 \Leftrightarrow \left(I + A^{-1}R\right)x = 0 \Rightarrow \left\|A^{-1}Rx\right\| = \|x\| \Rightarrow \|x\| \le \left\|A^{-1}R\right\|\|x\|$$

Because $\|A^{-1}R\| < 1$ this implies that $\|x\|$ equals 0 so $x = 0$. This proves that **LU** is non-singular.

$$\operatorname{cond}\left[(LU)^{-1}A\right] = \operatorname{cond}\left[(A + R)^{-1}A\right] = \operatorname{cond}\left[\left(I + A^{-1}R\right)^{-1}\right]$$

$$= \left\|\left(I + A^{-1}R\right)^{-1}\right\|\left\|I + A^{-1}R\right\| \le \left\|\left(I + A^{-1}R\right)^{-1}\right\|\left(1 + \left\|A^{-1}R\right\|\right)$$

By a Theorem of Atkinson [4] $\left\| \left(I + A^{-1}R\right)^{-1} \right\| \le \dfrac{1}{1-\left\|A^{-1}R\right\|}$. This completes the proof.

The Theorem 3 states that we can make R as small as possible and this will have a positive effect on the condition of $(LU)^{-1}A$.

(II). Case 2. The Preconditioner M = SSOR and ADI.

The preconditioning step in the CGS, Bi-CGSTAB and GMRES methods involves the preconditioner SSOR is the step (10). Hence, we must solve the equation $Mz = \dfrac{1}{\omega(2-\omega)}(D_A + \omega L_A)D_A{}^{-1}(D_A + \omega U_A)z = v$. The solution to this equation can be expressed as

$$\begin{aligned}(D_A + \omega L_A)\tilde{z} &= \omega(2-\omega)v \\ (D_A + \omega U_A)z &= D_A\tilde{z}\end{aligned} \qquad (13)$$

The speed of convergence of SSOR depends critically on ω; the optimal value for ω may be estimated from the spectral radius of the Jacobi iteration matrix $R_J = -D_A^{-1}(L_A + U_A)$.

Similarly, The preconditioning step in the CGS, Bi-CGSTAB and GMRES that involves the preconditioner ADI is the step (10). Hence, we must solve the equation $Mz = \dfrac{1}{\omega}(D_A + \omega L_A)D_A{}^{-1}(D_A + \omega U_A)z = v$. The solution to this equation can be expressed as

$$\begin{aligned}(D_A + \omega L_A)\tilde{z} &= \omega v \\ (D_A + \omega U_A)z &= D_A\tilde{z}\end{aligned} \qquad (14)$$

The systems (13) and (14) can be solved by using Algorithm 3 and Algorithm 4 respectively because $D_A + \omega L_A$ and $D_A + \omega U_A$ are lower and upper triangular matrices respectively.

The optimal value for ω is given by

$$\omega = \frac{2}{1+\sqrt{2\left(1-\bar{\lambda}_J\right)}} \qquad (15)$$

where $\bar{\lambda}_J$ is the maximum eigenvalue of the matrix R_J [5]. Optimum convergence rate may be achieved for a value of relaxation factor ω selected for various computer runs, and no economical expressions for estimating ω would be suggested. In the present work, the chosen value of ω is obtained by the procedure explained in [5].

Note. The preconditioning step in the TFQMR that involves the preconditioners SSOR or ADI is the step (10). In the case SSOR is preconditioner, we have

$$M_1 = \frac{1}{\omega(2-\omega)}(D_A + \omega L_A),\ M_2 = D_A^{-1}(D_A + \omega U_A)$$

Similarly, in the case ADI is preconditioner, we have

$$M_1 = \frac{1}{\omega}(D_A + \omega L_A),\ M_2 = D_A^{-1}(D_A + \omega U_A)$$

4. Numerical Experiments

This Section compares the numerical efficiency of the ILU(m) preconditioner with SSOR and ADI. Different implementations of the CGS, Bi-CGSTAB, TFQMR and GMRES(10) will be compared in terms of CPU times and number of iterations. The iterative methods have been implemented as C++ codes using double precision accuracy. The five problems reported herein were solved on an IBM Compatible PC with Pentium IV processor (512 RAM). For our test runs, we always chose x_0 as initial guess. For all tests, the right-hand side *b* was set to Ax, where $x = (1,1,\dots,1)^T$. The iterations were stopped as soon as $\frac{\|r_1\|}{\|r_0\|} \le \varepsilon = 10^{-8}$. Finally, the convergence plots for the residual norms, in a logarithmic scale, versus the iteration number are given.

Results and Discussions:

In this Section, we introduce some examples to show the efficiency of the suggested direct and iterative methods for solving (1).

Example 1 [2]. Let us consider the matrix of size $n = 1000$

$$A = \begin{bmatrix} a & 1 & & & & & & \\ -1 & a & 1 & & & & & \\ & -1 & a & 1 & & & & \\ & & . & . & . & & & \\ & & & . & . & . & & \\ & & & & . & . & . & \\ & & & & & -1 & a & 1 \\ & & & & & & -1 & a \end{bmatrix}$$

The example is studied here to give due importance to GMRES(10) when it is used with a suitable preconditioner like ILU. Results are given in the Table (1), which lists the number of iterations, the solution CPU time in seconds and the relative residual error. It is interesting to note that the ILU preconditioned GMRES(10) and Bi-CGSTAB do not stagnate and produce very good results. It has been found that the performance of the ILU preconditioned GMRES(10) is the best, where the solution is given in two steps. However, the ILU preconditioned CGS and TFQMR algorithms stagnate and do not show any sign of convergence. Similarly when we use SSOR and ADI preconditioners, all proposed iterative methods get stagnated and do not converge. In order to show the convergence characteristics of the iterative solution methods, Fig. (1) presents the evolution of the relative residual norm for CGS, Bi-CGSTAB, TFQMR and GMRES(10) with ILU preconditioner. It is interesting to note that while the CGS and TFQMR are stagnating the GMRES(10) and Bi-CGSTAB residuals are decreasing.

Example 2 [3]. Consider the tridiagonal square matrix of size $n = 1000$.

$$A = \begin{bmatrix} 5.1 & 3 & & & & & & \\ 2 & 5.1 & 3 & & & & & \\ & 2 & 5.1 & 3 & & & & \\ & & . & . & . & & & \\ & & & . & . & . & & \\ & & & & . & . & . & \\ & & & & & 2 & 5.1 & 3 \\ & & & & & & 2 & 5.1 \end{bmatrix}$$

This example is chosen here to show that the performance of ADI preconditioner is better than that obtained by SSOR in all considered methods. Table (2) presents, under the same headings as Table (1), the relative residual error, the solution iterations and CPU time in seconds of this problem. Here, the performances of the ILU and ADI preconditioners are better than that one of SSOR in all methods considered in this work. The performance of SSOR preconditioned Bi-CGSTAB is not good in comparison to the other methods. The preconditioned GMRES(10) algorithm shows the best performance, with a speed up against the Bi-CGSTAB of about 2. Specific comparison shows for the iterative methods considered with the ILU preconditioner (or ADI) have an iteration number of the order of one-two to one–third of that with SSOR preconditioner. In Fig. (2) (for SSOR) and Fig. (3) (for ADI), we show the convergence curve for the considered methods. As the plot indicates, the convergence is faster with the GMRES(10).

Example 3 [1]. Let A is a 200×200 Toeplitz matrix of the form

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & . & . & 0 \\ -1 & 1 & 1 & 1 & 1 & 0 & . & 0 \\ & -1 & 1 & 1 & 1 & 1 & . & 0 \\ & & . & . & . & . & . & 0 \\ & & & . & . & . & . & 1 \\ & & & & -1 & 1 & 1 & 1 \\ & & & & & -1 & 1 & 1 \\ & & & & & & -1 & 1 \end{bmatrix}$$

Table (3) presents the solution CPU time in seconds, number of iterations and the relative error in getting the solution of (1). This example is given here to show that the performances of ADI and SSOR preconditioners approximately are the same in all considered methods. Again, the performance of the ILU preconditioned GMRES(10) is much better than that of the other methods where the solution is given in one step. Specific comparison shows that for the considered iterative methods with ILU preconditioner have an iteration number of the order of one-fifth to on- sixth of that with SSOR (or ADI). But the convergence is faster with the GMRES(10). Finally, Fig. (4) (for SSOR) and Fig. (5) (for ADI) show the evolution of the relative residual norms for iterative methods. It is interesting to note that while the CGS and TFQMR residuals present oscillations the GMRES(10) residuals are monotonically decreasing.

Example 4 [3]. The matrix A is given by the block tridiagonal matrix

$$A = \begin{bmatrix} E & -I & & & & & \\ -I & E & -I & & & & \\ & . & . & . & & & \\ & & . & . & . & & \\ & & & . & . & . & \\ & & & & -I & E & -I \\ & & & & & -I & E \end{bmatrix} \quad E = \begin{bmatrix} 4 & \alpha & & & & & \\ \beta & 4 & \alpha & & & & \\ & . & . & . & & & \\ & & . & . & . & & \\ & & & . & . & . & \\ & & & & \beta & 4 & \alpha \\ & & & & & \beta & 4 \end{bmatrix}$$

and $\alpha = -1 + \delta, \beta = -1 - \delta$. The matrices represent the 5- point discretization of the operator $-\frac{\partial 2}{\partial x^2} - \frac{\partial 2}{\partial y^2} + \gamma \frac{\partial}{\partial x}$ on a rectangular region. Experiments are done for the matrix A of size n, where $n = 400$, $n_0 = \dim(E) = 20$, $na = 1920$. Computations are done for $\delta = 0.5$. Table (4) and Table (5) list the non-zero entries and fill-ins in constructing L and U, the number of iterations, solution CPU time in seconds and the relative residual error. From the table, it has been seen that the performance of the ILU(2) preconditioner in the GMRES(10) method is the best. The solution is given in one step (ILU(2)) and two steps (ILU(1)). Increasing m for an ILU(m) factorization reduces the total cost of finding an accurate solution (at least for small m). The performance of SSOR and ADI preconditioned iterative methods is much better than that obtained by ILU(0). But the performance of ADI preconditioner is better than SSOR in all considered iterative methods. The ADI and SSOR preconditioned GMRES(10) algorithm is two times faster than CGS, Bi-CGSTAB and TFQMR. However, the performance of the preconditioned GMRES(10) is the best among the considered iterative methods. The number of iterations by using GMRES(10) is much less. Finally, in order to show the convergence characteristics of the proposed iterative methods. Fig. (6) (for ILU(0)), Fig. (7) (for ILU(1)), Fig. (8) (for SSOR), and Fig. (9) (for ADI) show the relative residual norms for the iterative methods, where again the good properties of GMRES(10) can be seen.

Example 5 [3]. The matrix A is given by the block tridiagonal matrix

$$A=\begin{bmatrix} E & D_1 & & & & \\ D_2 & E & D_1 & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & \cdot \\ & & & D_2 & E & D_1 \\ & & & & D_2 & E \end{bmatrix},\ D_1=\begin{bmatrix} a & & & & & \\ & a & & & & \\ & & \cdot & & & \\ & & & \cdot & & \\ & & & & \cdot & \\ & & & & a & \\ & & & & & a \end{bmatrix},\ D_2=\begin{bmatrix} b & & & & & \\ & b & & & & \\ & & \cdot & & & \\ & & & \cdot & & \\ & & & & \cdot & \\ & & & & b & \\ & & & & & b \end{bmatrix}$$

and $\alpha=-1+\delta,\beta=-1-\delta$ and $a=-1+\delta_1,b=-1-\delta_1$, **E** is same as defined in Example 4. The matrix A represents the 5- point discretization of the operator $-\frac{\partial 2}{\partial x^2}-\frac{\partial 2}{\partial y^2}+\gamma\frac{\partial}{\partial x}+\tau\frac{\partial}{\partial y}$ on a rectangular region. We have chosen the values of $\delta=2.5$ and $\delta_1=2.0$, where $n_0=\dim(E)=20, n=400, na=1920$. Table (6) and Table (7) list, under the same heading as Table (4) and Table (5), the non-zero entries and fill-ins in constructing L and U decomposition, solution CPU time in seconds, iteration number and the relative residual errors. From the tables, it has been seen that the performance of the ILU(m) (m=1,2) in the GMRES(10) method is the best. The number of iterations by GMRES(10) solver is much less. The solution is given in two steps (ILU(1)) or one step (ILU(2)). In this example, the performance of ILU(0) is better than that obtained by SSOR and ADI in CGS, Bi-CGSTAB, and TFQMR. The performance of the ADI preconditioner is better than the SSOR preconditioner in CGS, Bi-CGSTAB and TFQMR. The ADI and SSOR preconditioned GMRES(10) algorithm is 2 times faster than CGS, Bi-CGSTAB and TFQMR. Specific comparisons show that for the considered methods with ADI (or SSOR) have an iteration number of the order of one- fifth to one-sixth with GMRES(10). Finally, in order to show the convergence characteristics of the proposed iterative solution methods. Fig. (10) (for ILU(0)), Fig. (11) (for ILU(1)), Fig. (12) (for SSOR) and Fig. (13) (for ADI) show the relative residual norms for the iterative methods. Again the good properties of GMRES(10) can be seen.

Table (1). No. of Iterations, CPU time and R. Residual Norms for Example1, $\omega = 0.95$.

Method	Precond.	Non-zeros & fill-ins	No. of Iteration	CPU time	Relative Residual Norms
CGS	ILU	1980	-	-	-
	SSOR	4753	-	-	-
	ADI	6375	-	-	-
Bi-CGSTAB	ILU	1980	3	0.054945	3.286046×10^{-10}
	SSOR	4753	-	-	-
	ADI	6375	-	-	-
TFQMR	ILU	1980	-	-	-
	SSOR	4753	-	-	-
	ADI	6375	-	-	-
GMRES(10)	ILU	1980	2	0.054945	3.891019×10^{-15}
	SSOR	4753	-	-	-
	ADI	6375	-	-	-

Table (2). No. of Iterations, CPU time and R. Residual Norms for Example 2, $\omega = 0.95$.

Method	Precond.	Non-zeros & fill-ins	No. of Iteration	CPU time	Relative Residual Norms
CGS	ILU(0)	1980	3	0.054945	5.741227×10^{-14}
	SSOR	4753	6	0.054945	3.449238×10^{-11}
	ADI	6375	3	0.054945	1.479128×10^{-11}
Bi-CGSTAB	ILU(0)	1980	3	0.054945	8.178668×10^{-9}
	SSOR	4753	18	0.109890	8.762133×10^{-9}
	ADI	6375	8	0.054945	1.288786×10^{-9}
TFQMR	ILU(0)	1980	2	0.054945	9.876344×10^{-11}
	SSOR	4753	5	0.109890	1.096002×10^{-9}
	ADI	6375	3	0.054945	2.859933×10^{-9}
GMRES(10)	ILU(0)	1980	1	negligible	2.153409×10^{-15}
	SSOR	4753	4	0.054945	1.920939×10^{-9}
	ADI	6375	3	0.054945	1.166310×10^{-13}

Table (3). No. of Iterations, CPU time and R. Residual Norms for Example 3, $\omega = 1.75$.

Method	Precond.	Non-zeros & fill-ins	No. of Iteration	CPU time	Relative Residual Norms
CGS	ILU(0)	1980	3	negligible	1.431621×10^{-14}
	SSOR	4753	13	0.054945	2.873238×10^{-9}
	ADI	6375	14	0.054945	9.634591×10^{-10}
Bi-CGSTAB	ILU(0)	1980	3	negligible	6.337973×10^{-9}
	SSOR	4753	28	0.109890	7.709481×10^{-9}
	ADI	6375	30	0.109890	5.890148×10^{-9}
TFQMR	ILU(0)	1980	3	0.054945	8.346323×10^{-14}
	SSOR	4753	14	0.109890	8.4247764×10^{-9}
	ADI	6375	15	0.109890	1.867795×10^{-10}
GMRES(10)	ILU(0)	1980	1	negligible	1.945635×10^{-12}
	SSOR	4753	5	0.054945	4.010245×10^{-9}
	ADI	6375	5	0.054945	2.984196×10^{-16}

Table (4). No. of Iterations, CPU time and Relative Residual Norms for Example 4.

Method	Precond.	Non-zeros & fill-ins	No. of Iteration	CPU time	Relative Residual Norms
CGS	ILU(0)	1980	12	0.109890	4.220049×10^{-15}
	ILU(1)	4753	4	0.054945	1.100062×10^{-14}
	ILU(2)	6375	3	0.054945	1.313719×10^{-13}
Bi-CGSTAB	ILU(0)	1980	14	0.109890	8.050642×10^{-11}
	ILU(1)	4753	5	0.054945	1.465550×10^{-9}
	ILU(2)	6375	5	0.054945	8.913006×10^{-11}
TFQMR	ILU(0)	1980	12	0.109890	2.003919×10^{-12}
	ILU(1)	4753	3	0.054945	9.300247×10^{-9}
	ILU(2)	6375	2	0.054945	2.486359×10^{-9}
GMRES(10)	ILU(0)	1980	10	0.109890	7.736185×10^{-17}
	ILU(1)	4753	2	negligible	2.901990×10^{-16}
	ILU(2)	6375	1	negligible	1.426597×10^{-13}

Table (5). No. of Iterations, CPU time and R. Residual Norms for Example 4, $\omega=1.5$.

Method	Precond.	No. of Iteration	CPU time	Relative Residual Norms
CGS	SSOR	10	0.109890	4.961067×10^{-12}
	ADI	8	0.109890	6.351351×10^{-13}
Bi-CGSTAB	SSOR	14	0.109890	7.932627×10^{-9}
	ADI	11	0.109890	8.646714×10^{-10}
TFQMR	SSOR	9	0.109890	1.526661×10^{-9}
	ADI	6	0.109890	5.672042×10^{-9}
GMRES(10)	SSOR	2	0.054945	3.372888×10^{-9}
	ADI	2	0.054945	1.310233×10^{-10}

Table (6). No. of Iterations, CPU time and R. Residual Norms for Example 5.

Method	Precond.	Non-zeros & fill-ins	No. of Iteration	CPU time	Relative Residual Norms
CGS	ILU(0)	1980	11	0.109890	5.741227×10^{-10}
	ILU(1)	4753	3	0.054945	3.180915×10^{-12}
	ILU(2)	6375	2	0.054945	1.112380×10^{-21}
Bi-CGSTAB	ILU(0)	1980	12	0.109890	6.063143×10^{-9}
	ILU(1)	4753	4	0.109890	9.212743×10^{-9}
	ILU(2)	6375	3	0.054945	3.216868×10^{-17}
TFQMR	ILU(0)	1980	10	0.109890	4.608243×10^{-10}
	ILU(1)	4753	3	0.10989	3.364870×10^{-12}
	ILU(2)	6375	2	0.10989	

				0 0.10989 0	1.324277×10^{-23}

GMRES(10)	ILU(0)
	ILU(1)
	ILU(2)
		1980
		4753
		6375
			9
			3
			1
				0.10989
				0
				0.05494
				5
				0.054945
					8.736124×10^{-12}
					1.937795
					$\times10^{-17}$
					1.610924×10^{-15}

Table (7). No. of Iterations, CPU time and R. Residual Norms for Example 5, $\omega = 0.8$.

Method	Precond.	No. of Iteration	CPU time	Relative Residual Norms
CGS	SSOR ADI	16 14	0.054945 0.054945	6.083266×10^{-11} 6.639469×10^{-14}
Bi-CGSTAB	SSOR ADI	18 15	0.109890 0.054945	7.286967×10^{-9} 7.580448×10^{-9}
TFQMR	SSOR ADI	14 13	0.109890 0.109890	6.356598×10^{-9} 3.062651×10^{-10}
GMRES(10)	SSOR ADI	4 4	0.054945 0.054945	5.349289×10^{-9} 2.153155×10^{-10}

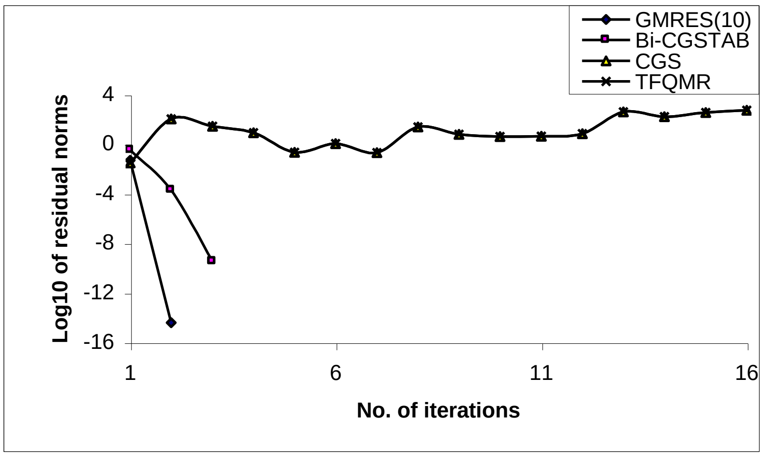

Fig.(1). Example 1, $n = 1000$, $a = 10^{-15}$, ILU preconditioner.

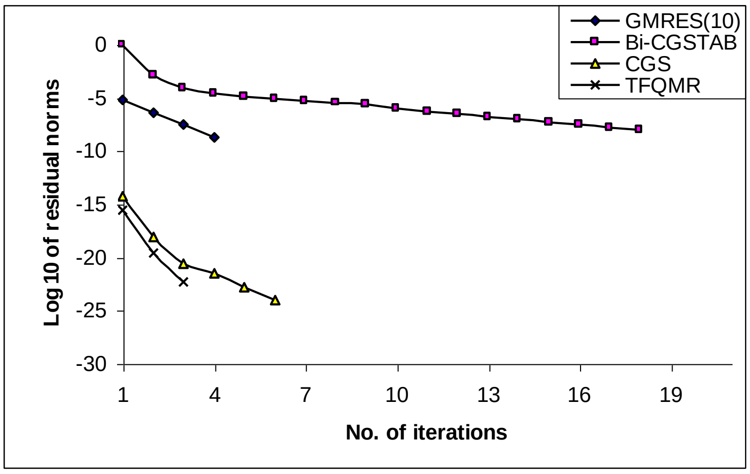

Fig.(2). Example 2, $n = 1000$, SSOR preconditioner.

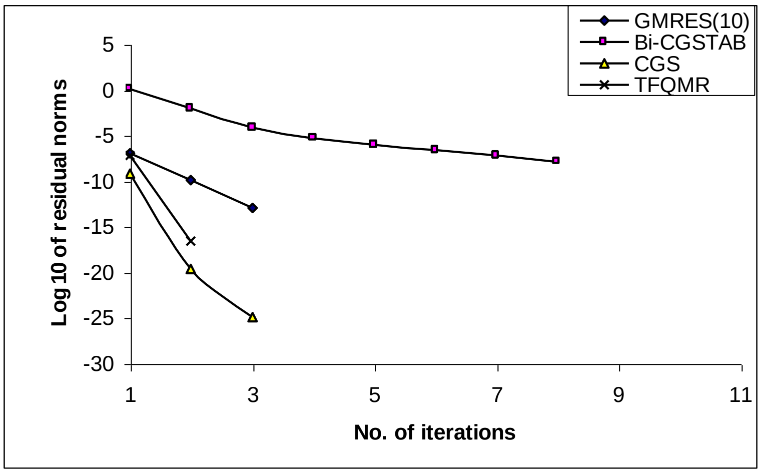

Fig.(3). Example 2, $n = 1000$, ADI preconditioner.

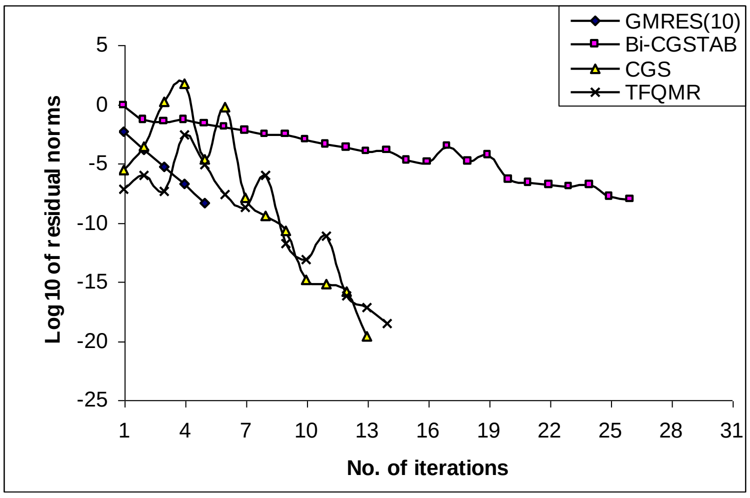

Fig.(4). Example 3, $n = 200$, SSOR preconditioner

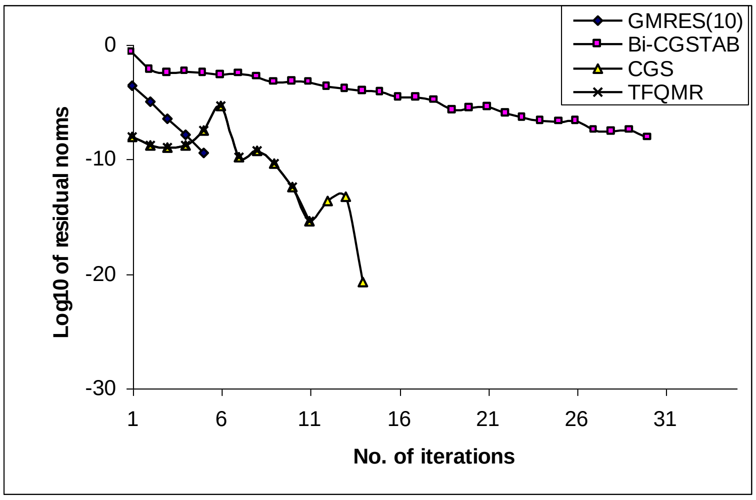

Fig.(5). Example 3, $n = 200$**, ADI preconditioner.**

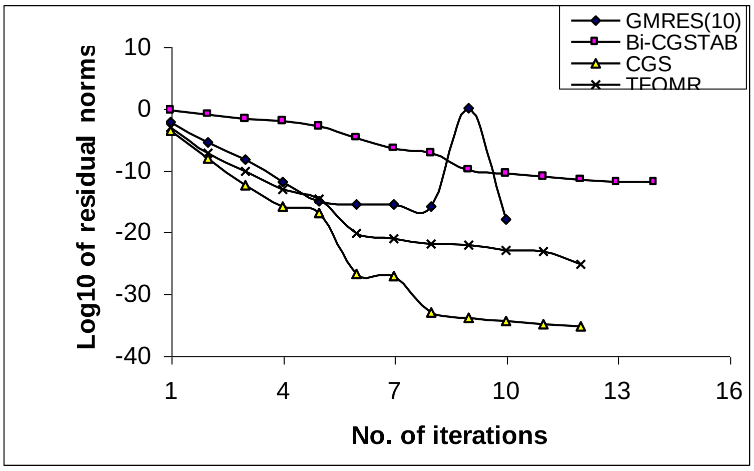

Fig.(6). Example 4, $n = 400$**, ILU(0) preconditioner.**

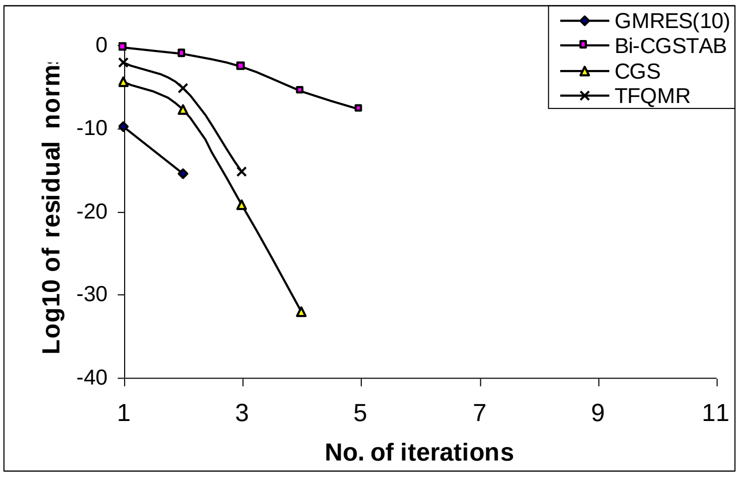

Fig.(7). Example 4, $n = 400$**, ILU(1) preconditioner.**

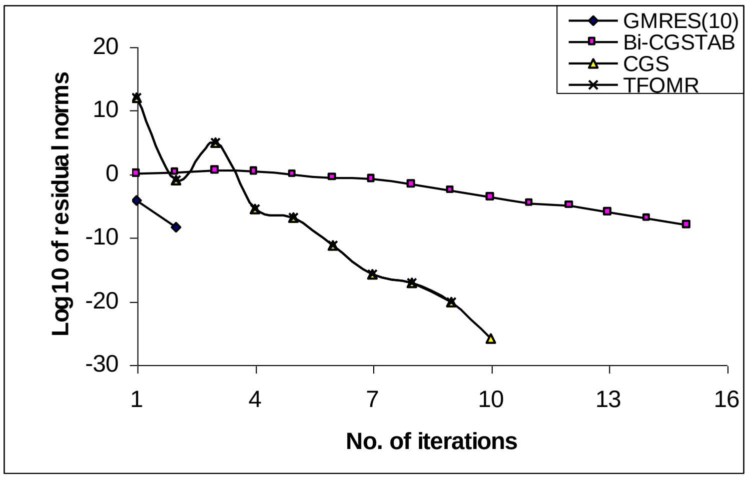

Fig.(8). Example 4, $n = 400$**, SSOR preconditioner.**

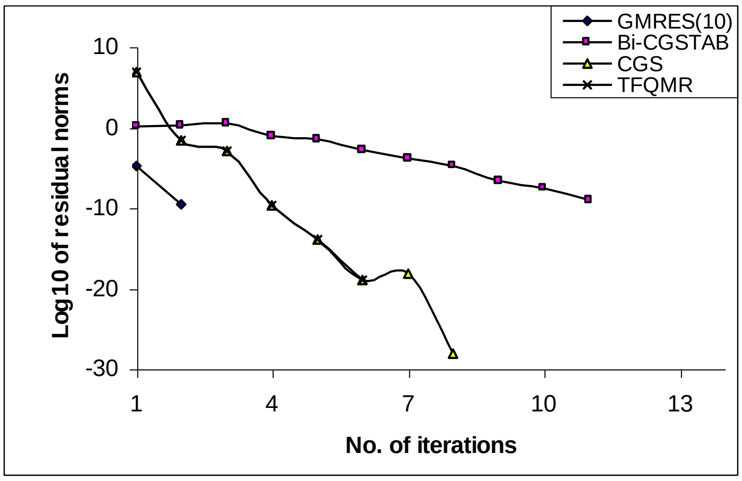

Fig.(9). Example 4, $n = 400$**, ADI preconditioner.**

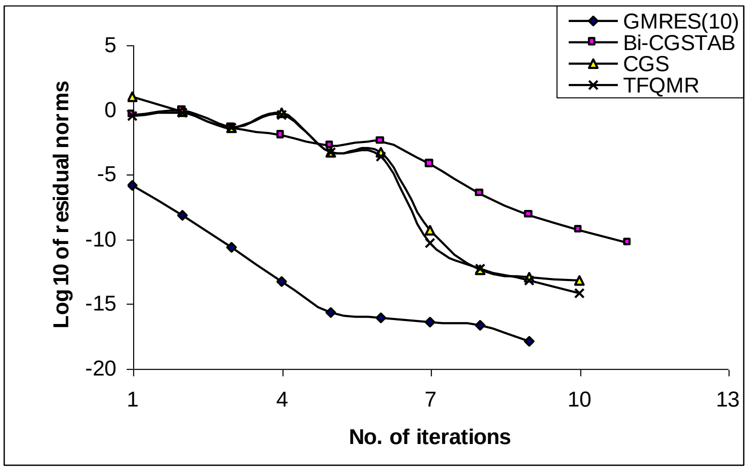

Fig.(10). Example 5, $n = 400$**, ILU(0) preconditioner.**

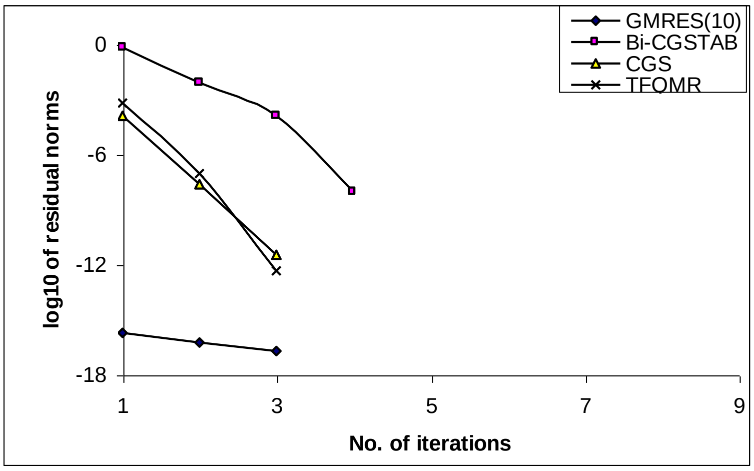

Fig.(11). Example 5, $n = 400$**, ILU(1) preconditioner.**

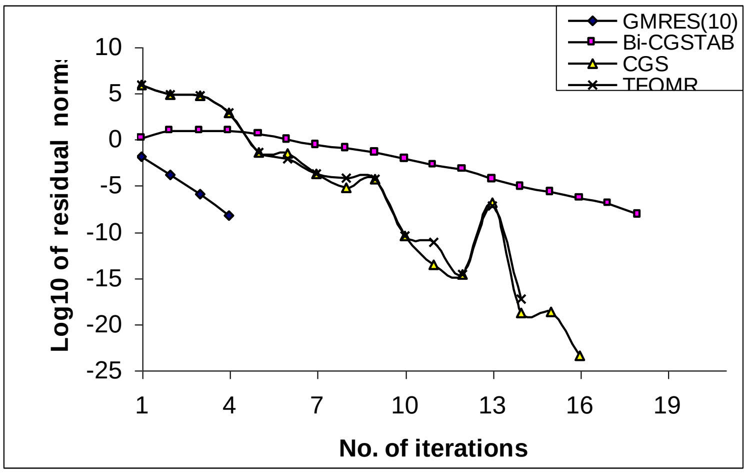

Fig.(12). Example 5, $n = 400$**, SSOR preconditioner.**

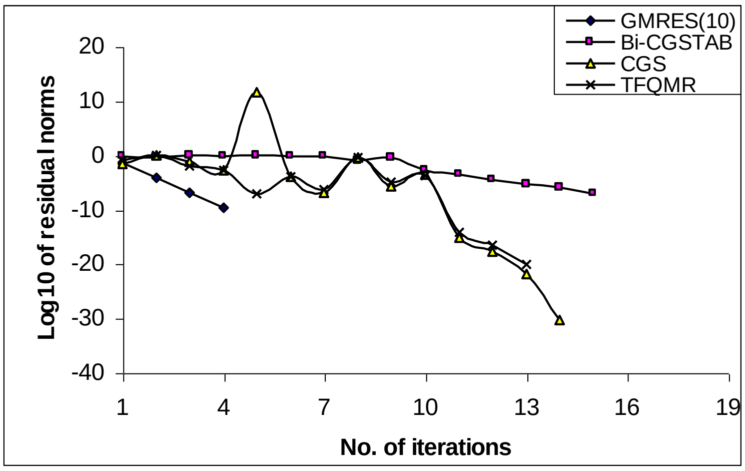

Fig.(13). Example 5, $n = 400$**, ADI preconditioner.**

Conclusions and Recommendations:

We considered some preconditioned Krylov iterative methods for solving large sparse non-symmetric linear systems of equations. The relative performance of three preconditioners, namely ILU(m), SSOR and ADI in the CGS, Bi-CGSTAB, TFQMR and GMRES(10) methods is shown in the Tables. Numerical experiments have shown that the ILU(m) (m=1,2) preconditioned GMRES(10) requires less iterations to converge. Increasing m for ILU(m) factorization reduces the total cost of finding an accurate solution (at least for small m), even though the cost of finding the approximation increases. The ILU(m) (m=1,2) preconditioner is found to be the best. That is reasonable convergence was obtained for $m \geq 2$. However, the total cost of finding the solution was reduces as m increased. Finally, the performance of preconditioned GMRES(10) was the best when it is combined with a suitable preconditioner. The GMRES(10) gives us a good performance and has fast convergence with all considered preconditioners.

REFERENCE:

[1]. SAAD, Y. *Iterative Methods for Large Sparse Linear Systems*, 1st edition, PWS Publishing Company, New York, 1995, 879.

[2]. BROWN, P. N. *A theoretical comparison of the Arnoldi and GMRES algorithm*, SIAM J. Sci. Stat. Comp. U. S. A., Vol. 12, No. 25, 1991, 58-78.

[3]. MITTAL, R. C. and AL-KURDI, A. H. *An efficient method for constructing ILU preconditioner for solving large sparse non-symmetric linear systems by GMRES method*, Computers Math. Appl. U. S. A., Vol. 45, No. 23, 2003, 1757-1772.

[4] ATKINSON, K. E. *An Introduction to Numerical Analysis*, 2nd Edition, Wiley, Chichester, New York, Brisbane, Toronto and Singapore, 1988, 570.

[5]. THOMAS, J.W. *Numerical Partial Differential Equations*, 2nd edition, Springer-Verlag, New York, 1999, 986.