مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الأساسية المجلد (30) العدد (1) 2008
Tishreen University Journal for Research and Scientific Studies - Basic Sciences Series Vol. (30) No. (1) 2008

Some Developed Direct and Iterative Methods for Solving Sparse Linear Systems of Equations

Dr. Ahmad Al-Kurdi *

(Received 12 / 9 / 2007. Accepted 26/12/2007)

□ ABSTRACT □

In this paper, efficient direct and iterative methods are described for solving a large random sparse non-symmetric linear system. Such systems of linear equations of huge order arise in several applications such as physics, mechanics, signal processing and other applications of real life problems. For this reason, we try to develop direct and iterative methods for solving such systems of linear equations. The suggested direct method is based on the sparse LU-decomposition method (DSLU). The developed iterative methods include a Semi-iterative Method (SM), a Splitting-based Iterative Method (SIM) and a preconditioned GMRES method. We consider two types of preconditioners based on Incomplete LU-decomposition (ILU). We test and compare the numerical implementations of these methods on four numerical examples to demonstrate their efficiency. Results show that the proposed ILU preconditioners in GMRES reduce largely number of iterations and give very accurate solutions.

Keywords:
GMRES , Preconditioner ILU , Sparse system , Direct method

* **Assistant Professor, Department of Mathematics, Faculty of Sciences, Al-Baath University, Homs, Syria.**

مجلة جامعة تشرين للبحوث والدراسات العلمية – سلسلة العلوم الأساسية المجلد (30) العدد (1) 2008
Tishreen University Journal for Research and Scientific Studies - Basic Sciences Series Vol. (30) No. (1) 2008

بعض الطرائق المباشرة و التكرارية المطورة لحل جمل المعادلات الخطية كثيرة الأصفار

الدكتور أحمد الكردي*

(تاريخ الإيداع 12 / 9 / 2007. قُبِل للنشر في 2007/12/26)

□ الملخّص □

في هذه المقالة، نصف طرائق مباشرة وتكرارية فعالة لحل جمل معادلات خطية، غير متناظرة ، كيفية، كثيرة الأصفار ذات مراتب عليا. تظهر هذه الجمل من المعادلات الخطية ذات المراتب العليا في تطبيقات عديدة كالفيزياء و الميكانيك والمعالجة الرقمية وتطبيقات أخرى من مسائل الحياة الحقيقية. لهذه الأسباب نحاول تطوير طرائق مباشرة وطرائق تكرارية لحل هذا النوع من جمل المعادلات. تعتمد الطريقة المباشرة المقترحة على طريقة تحليل LU كثيرة الأصفار (DSLU). تتضمن الطرائق التكرارية المطورة: طريقة نصف تكرارية (SM) و طريقة تكرارية تعتمد على التجزئة (SIM) و طريقة GMRES المسرعة. ندرس نوعين من المسرعات التي تعتمد على تحليل LU غير التام (ILU). نختبر و نقارن التنفيذات العددية لهذه الطرائق على أربعة أمثلة عددية لتوضيح فعاليتها. تبين النتائج أن المسرعات ILU المحددة في GMRES تخفض عدد التكرارات بشكل كبير و تعطي حلولا دقيقة جدا.

كلمات مفتاحية:
GMRES، المسرع ILU ، جملة كثيرة الأصفار، طريقة مباشرة.

1. Introduction:

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

Our goal is to solve a system of linear equations

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

where A is a large, random sparse and non-symmetric matrix of order $n \times n$ and b is a given column vector of order n. Such systems are frequently encountered in almost all scientific and engineering applications [1,2] such as physics, mechanics, signal processing and other applications of real life problems. For this reason, we try to develop direct and iterative methods for solving such systems of linear equations. Methods of solution may be classified as direct, involving a fixed number of arithmetic operations, and iterative, involving the repetition of certain steps until the desired accuracy is achieved. The performance of direct methods for sparse systems is largely due to that of the factorization of the matrix. The disadvantage with iterative methods is that the rate of convergence may be slow or they may even diverge and we need to find preconditioning matrix to speed up the convergence rates [3]. This suggests that direct methods should be preferred. The first approach to solve (1) is by the direct sparse LU-decomposition method (DSLU). In this approach, an upper triangular matrix U and a lower triangular matrix L are constructed such as $A = LU$ [4]. The system (1) can then be solved in two steps using the factors L and U, respectively. The second approach to solve (1) is by iterative methods, which include a Semi-iterative Method (SM), a Splitting-based Iterative Method (SIM) and preconditioned GMRES method [5]. Certain preconditioners are able to improve on this situation. A good preconditioner is necessary for the convergence of iterative methods for a large random non-symmetric coefficient matrix A. We will present two types of preconditioning in the solution of (1). The ILU preconditioners are based on an ILU-decomposition [6,7,8] which are among the most successful preconditioners. They are of interest because of the spectral condition number of the preconditioned system can be of a lower order.

In this paper, direct sparse LU-decomposition method (DSLU), SM, SIM and preconditioned GMRES method which are applicable to the solution of (1) are described. They are tested and compared on four numerical examples to demonstrate their efficiency. Two types of the preconditioners based on ILU are outlined.

2. Importance and Parts of the Research:

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

2.1. Storage Scheme:

The data structure described here [9,10] involves the use of three arrays. $VA[1...na]$ contains all the non-zero entries of A stored row by row. $JA[1...na]$ contains all the column numbers of these entries and $IA[1...n+1]$ is an array of pointers: $IA[i]$ gives the address in VA of the first non-zero entry in row number i of A when $i \le n$. $IA[n+1]-1$ points to the last non-zero entry in row n of A. For example, the matrix A:

$$A = \begin{bmatrix} a_{11} & 0 & a_{13} & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 \\ 0 & 0 & a_{33} & 0 & 0 \\ a_{41} & 0 & 0 & a_{44} & a_{45} \\ 0 & a_{52} & 0 & 0 & a_{55} \end{bmatrix} \qquad (2)$$

can be presented by the *VA*, *JA* and *IA* arrays as

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

The matrix (2) 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 (2) is factored, non-zero numbers are assigned to a_{42} and a_{53}, but neither of these elements appears in (3) as illustrated, i.e., there is a need for reallocating storage to make room for the fill-ins.

2.2. Proposed Solution Algorithms

The solution of (1) is typically found by two different types of methods-direct methods and iterative methods. In this section, we present direct sparse LU-decomposition method (DSLU) and three different iterative methods including (i) SM (ii) SIM and (iii) GMRES method with preconditioning matrices derived by incomplete LU-factorization of A in the form proposed in [11].

The determination of fill-ins of a sparse matrix is a central problem in the solution of sparse linear systems of equations using direct methods such as direct sparse LU-decomposition method (DSLU). In this subsection, we describe an efficient algorithm for determining the sparsity P of the LU factors, which have no restriction at all with respect to the sparsity pattern of A. This algorithm is based on the powers of a Boolean matrix obtained from A. The sparsity pattern is described either a priori or implicitly by some approach. However, it is desirable to know in advance the pattern of non-zeros of the LU factors because of updating the data structure to facilitate the non-zeros and fill-ins as well. Moreover, the method, as we see, provides the information needed for computing the non-zero structure of the LU factors. After determining the sparsity pattern of the LU factors, computing L and U is straightforward.

The problem is to find the set P of edges for which the factors L and U are sparse but also such as matrix LU resembles A as much as possible. In case the sparsity pattern of A is irregular, there are several possibilities to construct a good choice for the set P. Gustafsson [12] proposed the following:

First consider the standard incomplete LU- decomposition, i.e. $P = \{ (i, j) : a_{ij} \neq 0 \}$. Then extend P with positions (i, j) where the product LU has non-zero elements and eventually continue in this manner a few steps more. This technique is tested extensively by Langtangen [6]. It is costly and hence is not recommended.

Another approach determines the elements in P during the elimination process. P is described implicitly by allowing only entries which are in absolute value greater than a certain threshold value [7]. This approach is very sensitive if matrix A is ill-conditioned and thereby it does not suit such cases.

For these reasons, we describe the following best approach to construct P which avoids the above mentioned drawbacks. This approach that we take uses a Boolean matrix multiplication.

For these reasons, we use in this paper best approach [11] to construct P which avoids the above mentioned drawbacks. The approach that we take uses multiplication of Boolean matrix, which differs from regular multiplication [11].

Consider the digraph $G_A=(V,E)$ of a given matrix A. If B is the Boolean matrix representing the digraph G_A, the modified digraph $G_{B^m}=(V,E_m)$ is defined as $E_m=E\cup\{(v_i,v_k)\}$; that is a new edge (v_i,v_k) is added to G_A to form G_{B^m}, where m is some positive integer. Initially, the sparsity structures of the matrices A and B are the same, that is, A and B are exactly having the non-zero elements at the same positions. But the problem is to obtain the modified G_{B^m} at the level m. However, while finding powers of B some zero elements in B become 1 in B^m. Every initially zero element in B becomes 1 in B^m gives a new edge, say, (v_i,v_k) which is added to G_A to form G_{B^m}. These elements are precisely the positions of fill-ins in A. Now we try to find the sparsity pattern of L and U in terms of set theory.

In order to determine the non-zero structure of L and U, we define the set

$$P=\{(i,j):\ position\ \ (i,j)\ is\ \ non-zero\ \ including\ \ \ fill-in\ ,\ \ 1\le i,j\le n\}$$

Then, clearly $P=\{(i,j):b_{ij}^{(m)}=1\}$ where $B^m=\left[b_{ij}^{(m)}\right]$, $1\le m\le n-1$, and m does not exceed the longest path of G_A. Note that the both two sets P and E_m have the same elements. Thus, the set P_m gives the sparsity pattern of ILU(m) and L and U are lower and upper triangular matrices respectively at level m. The method for finding the set P is summarized in the following algorithm.

Let $B=\lfloor b_{ij}\rfloor$ is given by

$$b_{ij}=\begin{cases}1, & if\ \ a_{ij}\neq 0\\ 0, & otherwise\end{cases} \qquad (4)$$

Algorithm 1

Step 1.

Form the matrix B as defined in (4)

Step 2.

Compute $B^{2^m},(m\ge 1)$.

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 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. $B^{2^m}=B^{2^{m+1}}$, then the process has converged and we have the sparsity pattern of the LU factors. If $B^{2^m}=B^{2^{m+1}}$, we get the complete LU factors. In this case, we get the direct sparse LU decomposition method (DSLU).

Definition 1:

Suppose that $B' = \lfloor b'_{ij} \rfloor$ and $B'' = \lfloor b''_{ij} \rfloor$ are square Boolean matrices of order n and "and" be a Boolean operator defined on the entries of B' and B''. The Boolean product of B' and B'', denoted $B'B''$, is the $n \times n$ Boolean matrix $C = \lfloor c_{ij} \rfloor$ defined by

$$c_{ij} = \begin{cases} 1 \ (True) & if \ b'_{ik} = 1 \text{ and } b''_{kj} = 1 \ for \ some \ k, \ 1 \le k \le n \\ 0 \ (False) & otherwise. \end{cases} \quad (5)$$

From the definition, it follows that $c_{ij} = 1$ iff $b'_{ik} = 1$ and $b''_{kj} = 1$. Since we need to pose a more specific query, we use the Boolean operator "and", which limits results to those items that certain both (or all) of the search terms in our query. Thus, we can easily perform the comparisons and checks for each position of the Boolean product.

To find the sparsity pattern of L and U at level m, compute the powers $B^2, B^{2^2}, B^{2^3}, ..., B^{2^m}$.

Note 2.

To find the sparsity pattern of L and U such that $A = LU$, it is sufficient to find the smallest m such that $B^{2^m} = B^{2^{m+1}}$, where m is the first time the matrix power B^{2^m} becomes $B^{2^{m+1}}$.

(I). Incomplete LU-Decomposition Method (ILU Method):

Once the non-zero structure of L and U matrices is obtained, i.e., when the set P is determined, the construction of ILU decomposition based on Doolittle's method, is made where all the diagonal entries of L are 1.

$A = LU$ gives

$$a_{ij} = \sum_{k=1}^{\min(i,j)} l_{ik} u_{kj} \quad (6)$$

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

$$l_{ik} = \frac{a_{ik} - \sum_{j=1}^{k-1} l_{ij} u_{jk}}{u_{kk}} \quad i > k \quad , \quad u_{ik} = a_{ik} - \sum_{j=1}^{i-1} l_{ij} u_{jk} \quad i \le k \quad (7)$$

While making an incomplete LU–decomposition, we need to store only non–zero entries of L and U. We define extra help 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 $R = A - LU$ equal 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 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.
```

(II). Iterative Methods:

In this subsection, we describe different iterative-like methods for solving (1). These methods include the following:

(1) Semi-Iterative Method (SM):

The semi-iterative methods are in principle direct methods. The significant feature of the methods, however, is that a good approximation to the solution can be obtained after a much smaller number of iterations. The goal of reducing the solution time is achieved only if the matrix is decomposed in an efficient manner. The major drawback of direct sparse LU-decomposition method (DSLU) is that L and U are not sparse due to fill-ins, so computer storage demands are very high. Moreover, the computational work for constructing the factors L and U increases considerably with the dimensions of the problem. We outline an algorithm for increasing the accuracy of the solution vector. A Semi-iterative Method (SM) can do it. The solution procedure consists of two steps:

1.In the first step: the matrix is decomposed in an approximate manner.

2.In the second step: a semi-iterative method is used to rapidly improve the accuracy of the solution.

The basic idea behind semi-iterative method is to firstly reduce the computer storage demands required to L and U by determining the fill-ins to the level of 2 and secondly to increase the accuracy of the solution vector of (1). The semi-iterative method is given in the following algorithm:

Algorithm 3 Semi-iterative Method:

1. Call Algorithm 1 to construct P for $m = 2$.
2. Compute $M = L^*U^* = LU(2)$ by using Algorithm 2.
3. Solve $L^*U^*x_1 = b$ by using forward and back substitution algorithms respectively.

4. For $i = 1,2,...$, until the desired accuracy is achieved

i. Compute $r_i = b - Ax_i$.

ii. Solve $L^* U^* y_i = r_i$ by using forward and back substitution algorithms

iii. Compute $x_{i+1} = x_i + y_i$.

☒ The differences between DSLU and SM algorithms:

From the Algorithm 3, it is clear that the differences between DSLU and SM is that in the former method it is required to:

1. Determine the fill-ins until $B^{2^m} = B^{2^{m+1}}$ while in SM it is enough to determine the fill-ins to the level of 2.
2. The computer storage demands required in DSLU is more larger than that of SM.
3. The accuracy achieved by SM is more than that achieved by DSLU.

Note that SM begins with the factorization of the original matrix. Once the LU factors have been computed the Algorithm 3 should be used to achieve the desired accuracy.

(2) Splitting-Based Iterative Method (SIM):

In this subsection, we outline an iterative refinement method for solving large non-symmetric sparse systems of linear equations as proposed in [15]. But this method is based only on the LU-decomposition outlined in Algorithms 1 and 2. We prove that for a given non-singular matrix A having a LU-decomposition, there exists a triangular splitting of A such as the spectral radius of the iterative matrix associated with splitting can be made arbitrarily small. This method is used this as refinement process in the LU-decomposition. The efficiency of the proposed iterative refinement will be shown in the numerical experiments section.

Suppose A the coefficient matrix of (1) has splitting

$$A = M - N \qquad (8)$$

where M is nonsingular matrix. Hence we can construct a splitting-based iterative method as follows:

$$x_{u+1} = M^{-1} N x_u + M^{-1} b \qquad (9)$$

The sequence $\{x_u\}$ generated by (9) will converge to the solution $x = A^{-1} b$ of the linear system (1) if the spectral radius of the iterative matrix $M^{-1}N$ is less than one i.e. $\rho(M^{-1}N) < 1$. In general we can not guarantee that the method (9) is convergent for any choice of M and N in (8). For the sake of the efficiency of the method (9), we have to consider whether:

(i) $Mx = c$ can be easily solved.

(ii) The splitting based iterative method satisfies the convergence condition $\rho(M^{-1}N) < 1$.

We choose a special splitting (8) satisfying condition (i), then study the spectral properties of $M^{-1}N$.

Definition 2:

Splitting (8) is called triangular if M is a triangular matrix.

In general, the inverse of the matrix M may not be easily obtainable. We will show that in general M can be chosen as a triangular matrix by using the following results.

Let $A = LU$ be the decomposition of A using Algorithms 1 and 2 where L and U are lower triangular and upper triangular matrices, respectively. Then [15]:

Lemma 1:

For given U the upper triangular non-singular matrix and given distinct numbers $\lambda_i \neq 0,\ i = 1,2,...,n$, there exist invertible matrices T and diagonal D such that:

$$DU = T^{-1}(I - \Lambda)T \qquad (10)$$

where $\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_n)$.

Proof. For a given U, we can always find a diagonal matrix D such as all diagonal entries of DU are distinct and do not vanish. This means that DU is diagonalizable. However, we can find

D such that the diagonal entries of DU are prescribed numbers. Now, we are able to determine T. Let T be one of the matrices that bring the product DU into diagonal form, i.e.,

$$TDUT^{-1} = (I - \Lambda) \qquad (11)$$

which completes the proof. □

From the Lemma 1, it is clear that the product DU is diagonalizable. Thus, if U is not diagonalizable, D in (11) can not be arbitrary.

Note 2:

Analogous to the proof of Lemma 1, we have the following. Let $A = LU$ be the decomposition of A with L and U are lower triangular and upper triangular matrices, respectively. Then there exist non-singular matrices T and D, D diagonal, such that

$$U^{-1}DL^{T} = T(I - \Lambda)T^{-1} \qquad (12)$$

Theorem 2:

Let A be an $n \times n$ nonsingular matrix having an LU-decomposition outlined in Algorithms 1 and 2. Then there exists a triangular splitting of A,

$$A = M - N \qquad (13)$$

such that the spectral radius $\rho(M^{-1}N)$ can be made arbitrarily small. In particular, for a given $\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_n)$ such that $\lambda_i \neq 0 \quad (i = 1,2,...,n)$, the matrix M can be chosen such that its spectrum of $M^{-1}N$ is the same as that of Λ.

Proof.

Let $A = LU$ be the decomposition of A with L and U are lower triangular and upper triangular matrices, respectively.

From Lemma 1, there exist matrices T and D, D diagonal, such that $DU = T^{-1}(I - \Lambda)T$

Defining $M = LD^{-1}$ and $N = L(D^{-1} - U)$. Then $M - N = LD^{-1} - L(D^{-1} - U) = LU - A$

and M is lower triangular matrix. Thus, $A = M - N$ is a triangular splitting. □

Theorem 3:

Let $A = LU$ be the decomposition of A, and let $\lambda_i, i \in 1{:}n$ be real numbers. Then there exists a splitting of A of type (13) such that M is symmetric and positive definite matrix. Moreover, for any $\rho \in (0,1)$, M can be chosen such that $\rho(M^{-1}N) = \rho$.

Proof:

Let $A = LU$ be an LU-decomposition of A. Then (12) guarantees the existence of non-singular matrices T and D, D diagonal with positive elements, such that

$$U^{-1}DL^T = T(I-\Lambda)T^{-1}$$

We can define a symmetric positive definite matrix M as

$$M = LDL^T$$

and $N = M - A = LDL^T - LU = L\left(DL^T - U\right)$. It follows that

$$M^{-1}N = I - \left(L^T\right)^{-1}D^{-1}L^{-1}LU = I - \left(L^T\right)^{-1}D^{-1}U = I - \left(U^{-1}DL^T\right)^{-1}$$

$$= I - \left(T(I-\Lambda)T^{-1}\right)^{-1} = -T^{-1}(I-\Lambda)^{-1}\Lambda T$$

To have $\rho\left(M^{-1}N\right)$ arbitrary, it is enough to require

$$\max\left(\frac{\lambda_j}{1-\lambda_j}\right) \le \rho$$

Theorem4:

Let $\lambda_i = \frac{1}{1+\alpha i}$, $\alpha > 0$, $i = 1,2,...,n$ and $d_i = \frac{1-\lambda_i}{u_{ii}}$. Then $M^{-1}N = T^{-1}\Lambda T$ and $\rho\left(M^{-1}N\right) < 1$, where $\Lambda = diag\left(\lambda_1, \lambda_2, ..., \lambda_n\right)$ and T is a non-singular matrix.

Proof:

We have

$$M^{-1}N = DL^{-1}L\left(D^{-1} - U\right) = I - DU \qquad (14)$$

From Lemma 1, we have $DU = T^{-1}(I-\Lambda)T$. Then

$$M^{-1}N = T^{-1}\Lambda T$$

From (14), it has been that $I - DU$ is triangular matrix having the eigenvalues λ_i and $|\lambda_i| < 1$, then $\rho\left(M^{-1}N\right) = \rho(I - DU) = \max_{1\le i\le n}\lambda_i = \frac{1}{1+\alpha} < 1$. □

From the Theorem 4, it is evident that after choosing λ_i, the splitting (13) will be determined. However, for convergence of (9) we have to choose all $|\lambda_i| < 1$.

Corollary 1:

Let A be an $n \times n$ non-singular matrix having LU-decomposition. Then there exists a convergent splitting of A of type (13) in which M is lower triangular matrix.

Suppose that all results in this section hold for decomposition with powers of a Boolean matrix outlined briefly in the Algorithms 1 and 2 then $A = LU$. Thus, we obtain $y = L^{-1}b$ and $x = U^{-1}y$. Now we try to construct an efficient Splitting-based iterative Method (SIM). In order to guarantee the convergence of this method, we take λ_i such that $|\lambda_i| < 1$. Then we have the following algorithm.

Algorithm 4: (SIM)

1. Determine $P = \left\{(i,j) \setminus b_{ij}^{(m)} = 1\right\}$ using Algorithm 1.
2. Compute LU by using Algorithm 2.
3. Choose d_i by $d_i = \frac{1-\lambda_i}{u_{ii}}$ where $\lambda_i = \frac{1}{1+\alpha i}$, $(i = 1,2,...,n)$, $\alpha > 0$.

4. Compute $LUx_0 = b$ by forward substitution and by back substitution algorithms.

5. Define (see Theorem 2) $M = LD^{-1}$ and $N = L\left(D^{-1} - U\right)$

6. For $u = 0,1,\ldots,$until convergence achieved Do $x_{u+1} = (I - DU)x_u + DL^{-1}b$.

Theorem 5:

Suppose $A = LU$ and $d_i = \dfrac{1-\lambda_i}{u_{ii}}$ for all given $|\lambda_i| < 1$. Then x_u generated by Algorithm 4 converges to the exact solution x^* of (1). However, λ_i can be chosen sufficiently small such that Algorithm 4 has a fast convergence rate.

Proof:

Since $x_{u+1} = (I - DU)x_u + DL^{-1}b$ and $LUx^* = b$. Then $x_{u+1} - x^* = (I - DU)^{u+1}\left(x_0 - x^*\right)$

Where $Bx_u = B^{u+1}x_0$. Thus $\left\|x_{u+1} - x^*\right\|_2 \le \rho^{u+1}(I - DU)\left\|x_0 - x^*\right\|_2$.

It follows from the triangular matrix $I - DU$ having the eigenvalues $|\lambda_i| < 1$ that $\rho(I - DU) < 1$ which implies $\left\|x_{u+1} - x^*\right\|_2 \underset{u\to\infty}{\to} 0$.

☒ Advantages of Algorithm 4:

1. At each iteration step, Algorithm 4 uses one product of a triangular matrix and vector and two products of a number and vector, instead of solving two triangular systems.
2. Algorithm 4 is always convergent, which is guaranteed by Theorem 2-5 because $\rho\left(M^{-1}N\right) < \dfrac{1}{2}$.
3. All operations in Algorithm 4 can be done in the same precision. It is not necessary to use double precision for iterative refinement.
4. Algorithm 4 can start with any initial approximation x_0.
5. We can choose $\rho\left(M^{-1}N\right) < \dfrac{1}{2}$, for example $\lambda_i = \dfrac{1}{\alpha i + 1}$, where $\alpha > 0$, such that the refinement sequence converges fast to the desired solution.
6. We can use incomplete decomposition to keep the desired sparsity, and then choose λ_i and d_i for Algorithm 4.
7. Algorithm 4 is very useful for vector and parallel processing, because it just involves products of a matrix and a vector.

(III). The Proposed Preconditioners:

Convergence of Krylov subspace methods can be significantly enhanced using preconditioners. In this Section, we outline the preconditioning strategy we use.

The preconditioners considered in this work are based on ILU decomposition. The ILU decomposition given in Algorithm 2 is based on the LU-decomposition of the coefficient matrix A. In ILU(m) (ILU of level m), we use Algorithm 1 for determining non-zeros which fill- L and U to a certain level m. A level function is used in an incomplete factorization to control the number of fill-ins. Algorithm 2 together with the Algorithm 1 does produce an optimal preconditioner, when $B^{2^m} = B^{2^{m+1}}$ and we get the

solution in one step. That is LU is approximately the exact decomposition of A because there is no information left out during decomposition.

The preconditioning matrix M can be chosen to represent an incomplete LU-decomposition of A. The ILU decomposition is defined so that M has the desired sparsity pattern P. For all pairs $(i, j) \in P$ decomposition is carried out, other pairs are left out. The commonest and simplest choice is $P = \{(i, j),\ a_{ij} \neq 0\}$, which allows no fill-in during incomplete factorization. The choice $P = \{(i, j),\ a_{ij} \neq 0\}$ leads to the preconditioner ILU(0) which is not the best choice to be made. Although this is a simple and an effective way of constructing M, in some cases it can differ significantly from A^{-1}, since too much information was left out during the ILU decomposition. However, another approach to achieve a powerful preconditioner is to allow some fill-ins. Increasing the fill-in cause an increase of computational work associated with the matrix-vector operations and with the ILU procedure. Therefore, we have introduced efficient and inexpensive technique to define the sparsity pattern P.

It is important that M^{-1} is never explicitly computed. Alternatively, we have

$$z = M^{-1}v \ \Rightarrow\ Mz = v \qquad (15)$$

Case 1: Preconditioning Matrix $M = LU$

The system (15) can be solved by the following two steps:

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

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

The preconditioning $M = LU$ involves writing A as $A = LU - R$, with R as error term. When solving the system (1) 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. Because $(LU)^{-1}A = (LU)^{-1}[(LU) - R] = I - (LU)^{-1}R$, then the matrix $(LU)^{-1}R$ should be as small as possible in some sense. We give two Theorems which state that $(LU)^{-1}$ is a proper approximation to A^{-1} if and only if $\|(LU)^{-1}R\|$ is sufficiently small for some matrix norm $\|.\|$.

Theorem 6:

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

$$\frac{\|(LU)^{-1}R\|}{cond\ (A)} \leq \frac{\|(LU)^{-1} - A^{-1}\|}{\|A^{-1}\|} \leq \|(LU)^{-1}R\| \qquad (16)$$

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

Proof:

$$(LU)^{-1}R = (LU)^{-1}(LU - A) = I - (LU)^{-1}A = \left[A^{-1} - (LU)^{-1}\right]A \qquad (17)$$

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

by dividing the left and the right-hand side by $\|A\|.\left\|A^{-1}\right\|$ one obtains the first inequality of (16). The second inequality follows from equation (16) and (17).

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

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

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

Theorem 7 :

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

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

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 8:

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

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

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.

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

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

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

The Theorem 8 states that we can make R as small as possible and this will have a positive effect on the condition of $(LU)^{-1}A$.

Note 3:

If $B^{2^m} = B^{2^{m+1}}$, then the matrix A has approximately the exact factorization LU, i.e., $A = LU$ because there is no information was left out during the ILU decomposition. Consequently, $M^{-1}A = I$ and we get the solution in one step.

Case 2: Preconditioning Matrix $M = LD^{-1}$:

The system (15) can be solved by Algorithm 4. The preconditioning $M = LD^{-1}$ involves writing A as $A = M - N$, where $N = L(D^{-1} - U)$. When solving the system using the splitting $M - N = LD^{-1} - L(D^{-1} - U)$, we consider the system (15). The preconditioned matrix $M^{-1}A = (LD^{-1})^{-1}A$ has to resemble the identity matrix I as closely as possible. Because $M^{-1}A = (LD^{-1})^{-1}A = (LD^{-1})^{-1}[LD^{-1} - L(D^{-1} - U)] = DU$, then the matrix DU should be small in some sense. The next Theorem states that M^{-1} is a proper approximation to A^{-1} if and only if $\|M^{-1}N\|$ is small for some matrix norm $\|.\|$.

Theorem 9:

Suppose $M - N = LD^{-1} - L(D^{-1} - U)$ is a splitting of the nonsingular $n \times n$ matrix A having LU-decomposition. Then

$$\frac{\|(LD^{-1})^{-1}L(D^{-1} - U)\|}{cond\ (A)} \le \frac{\|A^{-1} - (LD^{-1})^{-1}\|}{\|A^{-1}\|} \le \|(LD^{-1})^{-1}L(D^{-1} - U)\| \qquad (18)$$

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

Proof:

$$(LD^{-1})^{-1}L(D^{-1} - U) = (LD^{-1})^{-1}(LD^{-1} - A) = I - (LD^{-1})^{-1}A = \left[A^{-1} - (LD^{-1})^{-1}\right]A$$

$$\|(LD^{-1})^{-1}L(D^{-1} - U)\| \le \|A^{-1} - (LD^{-1})^{-1}\|\|A\| = \frac{\|A^{-1} - (LD^{-1})^{-1}\|}{\|A^{-1}\|}\|A\|\|A^{-1}\|$$

by dividing the left and the right-hand side by $\|A\|.\|A^{-1}\|$ one obtains the first inequality of (18). The second inequality follows from equation (18).

$$(LD^{-1})^{-1}L(D^{-1} - U) = \left[A^{-1} - (LD^{-1})^{-1}\right]A \Rightarrow (LD^{-1})^{-1}L(D^{-1} - U)A^{-1} = \left[A^{-1} - (LD^{-1})^{-1}\right]$$

$$\Rightarrow \|A^{-1} - (LD^{-1})^{-1}\| \le \|(LD^{-1})^{-1}L(D^{-1} - U)\|\|A^{-1}\|$$

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

2.3. Numerical Experiments:

In the foregoing, some examples are chosen randomly to illustrate the performance of the proposed methods and the preconditioning discussed in the section 4. The direct method used for comparison of CPU times is basically the same as LU with powers of a Boolean matrix strategy as given in Algorithm 1. All experiments are performed on an IBM Compatible PC with Pentium IV processor and 512 RAM. In our test runs, the zero vector x_0 is the initial guess. The right–hand side b is taken to be $b = Ax$, where $x = (1, 1, ..., 1)^T$, such that the solution of the system is just x. The equation solvers have been implemented as C++ codes using double precision accuracy of $\varepsilon = 10^{-8}$. Finally in all considered examples we apply Algorithm 4 with $d_i = \frac{1-\lambda_i}{u_{ii}}$ with λ_i chosen, say, $\lambda_i = \frac{1}{1+105i}$, $(i = 1,2,...,n)$, which found to guarantee the spectral radius $\rho(M^{-1}N) < \frac{1}{106}$ and a fast convergence rate of the SIM. The number of iterations of GMRES method denotes the number of outer iterations.

3. 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 [4]:

Consider the system (1) whose coefficient matrix A is an $n \times n$ Hilbert matrix. The matrix A is ill- conditioned for even modest size n. The well-known Hilbert matrix, which has a large condition number, is used as a numerical example to illustrate the performances of the considered algorithms. The computational results are as follows, where x_1^{GMRES}, x_1^{SIM}, x_1^{SM}, x^{LU} and $\tilde{x}$ is the approximate solution obtained by GMRES method [5], SIM, SM, direct sparse LU-decomposition method (DSLU) and Gaussian's elimination with row pivoting [4], respectively. The obtained results are reported in the Table (1).

Table (1): The solution of Example 1 by the different methods.

Order (n)	LU	SM	SIM	GMRES	Gaussian el.
4	0.99997735	0.99999996	0.99994979	1.00000000	2.08333330
	1.00011003	1.00000022	1.00028675	1.00000000	0.24166670
	0.99992049	0.99999980	0.99967540	1.00000000	0.02119045
	0.99997437	1.00000000	1.00005815	1.00000000	-0.00023813
5	0.99999265	1.00000011	0.99996331	1.00000000	0.02283334
	1.00004302	0.99999917	1.00072368	1.00000000	0.30833328
	0.99996321	1.00000152	0.99799995	1.00000000	0.04230156
	0.99999540	0.99999919	1.00127048	1.00000000	-0.00126315
	1.0004800	1.00000001	1.00011624	1.00000000	-0.00001149
6	1.00000930	0.99999993	0.99981332	1.00000000	2.45000030
	0.99994403	1.00000031	1.00123082	1.00000000	0.36785709
	1.00007093	0.99999978	0.99829934	1.00000000	0.06545557
	0.99997872	0.99999994	1.00015364	1.00000000	-0.00332178
	1.00003600	1.00000004	1.00036251	1.00000000	-0.00007202
	0.99995953	0.99999998	1.00016002	1.00000000	-0.00000054

From the table, the proposed methods give much more accuracy than that obtained by Gaussian elimination. However, the preconditioned GMRES(10) method gives much

more accuracy among the proposed methods. The SM gives good accuracy in comparison with direct sparse LU-decomposition method (DSLU) and SIM. The SIM gives much more accuracy in comparison with Gaussian elimination. Finally, the direct sparse LU-decomposition method (DSLU) with powers of a Boolean matrix gives very good accuracy in comparison with Gaussian elimination with row pivoting.

Example 2:

Consider the system (1) whose the coefficient matrix is given in [11], where $n = 100, na = 219$.

Example 3:

Consider the system (1) whose the coefficient matrix is given in [11], where $n = 400, na = 1276$.

Example 4:

Consider the system (1) whose the coefficient matrix is given in [11], where $n = 1000, na = 2190$.

We have used the methods described in this paper to construct a ILU-decomposition for the coefficient matrices associated with the Examples 2 to 4. The ILU(m) preconditioner is based on the powers of a Boolean matrix strategy. The results were obtained with $k = 10$ and for an iteration required precision of 10^{-8} is achieved. The performance of the algorithms discussed in this paper can be considered by examination of the statistics collected in tables (2) to (4). The tables show the timing information, the error obtained in getting the solution and the number of iterations needed for the convergence of the iterative methods for examples 2 to 4. The influence of m on convergence of the GMRES method with the ILU(m) is also reported in the Tables. The second column in the Tables shows the types of the preconditioners used. The third column shows the number of non-zero entries in L and U together, which, of course, vary with the parameter m. The fourth column shows how many iterations were needed to make the convergence criterion satisfied. The fifth column gives the time needed for getting the solution. The last column shows the error (relative residual) in finding the solution. The time needed to compute the preconditioners is not included because it is found to be the same about. The influence of k, the dimension of Krylov subspace [11], on the performance of GMRES using the proposed preconditioners is tested. For all the tests carried out, the best value for k is found to be 10.

Table (2): The timing information, the number of iterations and the error in finding the solution for Example 2, where $n = 100, na = 219$.

Method	Precond.	No. of nonzero entries	No. of iterations	CPU time (s)	Error
GMRES	$LD^{-1}(0)$	219	2	negligible	5.7e-10
	ILU(0)	219	9	0.054945	6.5e-10
	ILU(1)	768	6	0.054945	3.4e-10
	ILU(2)	1224	1	0.054945	1.3e-12
SIM	-	1224	1	0.219780	2.1e-10
LU	-	1224	1	0.054945	2.1e-8
SM	-	1224	1	negligible	6.1e-12

Table (3): The timing information, the number of iterations and the error in finding the solution for Example 3, where $n = 400, na = 1276$.

Method	Precond.	No. of nonzero entries	No. of iterations	CPU time (s)	Error
GMRES	$LD^{-1}(0)$	1276	2	0.054945	5.4e-10
	ILU(0)	1276	11	0.109890	1.1e-10
	ILU(1)	4522	5	0.054945	1.2e-10
	ILU(2)	5491	1	0.109890	1.1e-12
SIM	-	5491	1	0.274725	2.1e-10
LU	-	5491	1	0.109890	2.1e-8
SM	-	5491	1	0.109890	9.1e-12

Table (4): The timing information, the number of iterations and the error in finding the solution for Example 4, where $n = 1000, na = 2190$.

Method	Precond.	No. of nonzero entries	No. of iterations	CPU time (s)	Error
GMRES	$LD^{-1}(0)$	2190	4	0.164835	5.4e-10
	ILU(0)	2190	17	0.164835	1.1e-10
	ILU(1)	9115	8	0.219780	1.2e-10
	ILU(2)	11409	1	0.274725	1.1e-12
SIM	-	11409	1	0.334628	2.1e-10
LU	-	11409	1	0.329670	2.1e-8
SM	-	11409	1	0.329670	9.1e-12

From the Tables (2) to (4), it has been seen that the GMRES method needs more iterations to converge by using ILU(0). We also have found that the ILU(0) can not benefit from k values greater than 10. From the obtained results we can observe that the GMRES(10)/ILU(2) requires a small number of outer iterations, compared to the GMRES(10)/ILU(m) ($m = 0,1$) or GMRES(10)/$LD^{-1}(0)$. That is ILU(2) is better than ILU(m) ($m = 0,1$) and $LD^{-1}(0)$. The GMRES(10)/ILU(0) takes a short time to converge but requires a large number of iterations. The $LD^{-1}(0)$ preconditioner is better than the ILU(m) ($m = 0,1$) preconditioner in the term of CPU time and number of iterations. However, the results show that the convergence of GMRES(10) improves as we increases the value m from $m = 0$ to $m = 2$. From the results we can see that the preconditioned GMRES(10) algorithm is the fast method followed by the SM and direct sparse LU-decomposition method (DSLU). Note the direct sparse LU-decomposition method (DSLU) and SM are equally effective i.e. the CPU time is the same for both SM and direct sparse LU-decomposition method (DSLU). But the accuracy achieved by SM is much more than that obtained by direct sparse LU-decomposition method (DSLU). The difference in CPU time for direct sparse LU-decomposition method (DSLU) and SM comes from increasing

the forward substitution and back substitution because of increasing the fill-ins. In the SM it is found to be that ILU(2) is enough to achieve the desired accuracy. Thus, the SM is recommendable to use. The SIM takes a small number of iterations, but requires a much more time to converge. Finally, the GMRES performs the best and SM and direct sparse LU-decomposition method (DSLU) are better than SIM.

4. Concluding Remarks and Recommendations:

In this paper, we have described efficient different methods for the solution of (1). The practical comparisons of different implementations of the proposed methods, DSLU, SM, SIM, and preconditioned GMRES have been shown in the terms of CPU time to solve the same test problems. The numerical experiments indicate that the preconditioned GMRES algorithm has been demonstrated to be superior or competitive with the other considered methods. It should be noted that the SM provides much more accuracy to the solution of (1). From the numerical results, we also see that it is advantageous to use ILU(m) preconditioner based on powers of the Boolean matrix strategy. It should be noted that ILU(2) is the best in the terms of number of iterations followed by the $LD^{-1}(0)$. The $LD^{-1}(0)$ preconditioner is very successful in the terms of CPU time and number of iterations. For all the tests carried out, the best value for k, the dimension of Krylov subspace [11], is 10. It is recommendable to use $LD^{-1}(0)$ preconditioner over ILU(2) because there is no fill-ins. We end with the concluding remark to use GMRES(10) method with ILU(2) or $LD^{-1}(0)$ as preconditioners.

References:

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

[2] GERALD, C. F.; WHEATLEY, P. O. *Applied Numerical Analysis*, Fifth Edition, Addison-Wesley Publishing Co., New York, 1994, 654.

[3] SAAD, Y. *Iterative Methods for Large Sparse Linear Systems*, 1st Edition, PWS Publishing Company, New York, 1995, 875.

[4] GOLUB, G.; VAN LOAN, C. *Matrix Computation*, 2nd ed., Johns Hopkins U. P., 1989, 618.

[5] SAAD, Y. ; SCHULTZ, M. H. "*GMRES*": *A generalized residual algorithm for solving non- symmetric linear systems.* SIAM J. Sci. Stat. Comp. U. S. A. Vol. 7, No. 42 , 1986, 856-869.

[6] LANGTANGEN, H. P. *Conjugate Gradient Methods and LU- Preconditioning of Non-symmetric Systems with Arbitrary Patterns*, Int. J. Numer. Methods Fluids U.S.A., Vol. 9 , No. 5, 1989, 213- 233.

[7] VAN DER PLOG, A. *Preconditioning Techniques for Non-Symmetric Matrices with Applications to Temperature Calculations of Cooled Concrete*, Int. J. Numer. Meth. Eng. U.S.A., Vol. 35, No. 34, 1992, 1311-1328.

[8] CHANG, X. W.; PAIGE, C. C. *On the sensitivity of the LU-factorization*, BIT U.S.A., Vol. 38, No. 24, 1998, 486-501.

[9] ROSE, D. J.; WILLOUGHBY, R. A. *Sparse Matrix and Iits Applications*, 1st Edition, Plenum Press, New York, 1972, 358.

[10] BRAMELLER, A. , ALLAN, R. N.; HAMAN, Y. M. *Sparsity*, 1st Edition, Pitman Publishing, New York, 1976, 487.

[11] MITTAL, R. C. ; 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.

[12] GUSTAFSSON, I. *A Class of First Order Factorization*, BIT U. S. A., Vol. 18, No. 5, 1978, 142- 156 .

[13] HAGER, W. W. *Applied Numerical Linear Algebra*, 2nd Edition, Prentice Hall International (UK), Ltd., London, 1988, 842.

[14]. HOROWITZ, E. ; SAHNI, S. *Fundamental of Data Structures*, 1st Edition, Computer Science Press, New York, 1982, 654.

[15] YUAN, J. Y. *Iterative Refinement Using Splitting Methods*, Linear Algebra and its Applications U. S. A., Vol. 273, No. 5, 1998, 199- 214.