

LU-Decomposition with Iterative Refinement for Solving Sparse Linear Systems

Ahmad Al-Kurdi

*Department of Mathematics, Faculty of Science, Teshreen University, Lattakia,
SYRIA*

David R. Kincaid

*Computer Sciences Department, University of Texas at Austin, Austin, Texas
78712 USA*

Abstract

In the solution of a system of linear algebraic equations $Ax = b$ with a large sparse coefficient matrix A , the LU-decomposition with iterative refinement (LUIR) is compared with the LU-decomposition with direct solution (LUDS), which is without iterative refinement. We verify by numerical experiments that the use of sparse matrix techniques with LUIR may result in a reduction of both the computing time and the storage requirements. The Powers of a Boolean Matrix Strategy (PBS) is used in an effort to achieve such a reduction and in an attempt to control the sparsity. We conclude that iterative refinement procedures may be efficiently used as an option in software for the solution of sparse linear systems of equations.

Key words: large sparse linear systems, LU-decomposition with iterative refinement (LUIR), LU-decomposition with direct solution (LUDS), powers of a Boolean matrix strategy (PBS)

1991 MSC: 65F10, 65F50

1 Introduction

Consider the following system of linear algebraic equations

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

Email addresses: ahk_1966_2002@yahoo.com.in (Ahmad Al-Kurdi),
kincaid@cs.utexas.edu (David R. Kincaid).

where A is a nonsingular, large, sparse and nonsymmetric matrix of order $n \times n$ and b is a given column vector of order n . To solve the linear system (1) one can try several different algorithms. One way is to find the inverse and multiply it on both sides, which is expensive computationally. Another way is to make a guess at the solution and iteratively refine that guess until the error is suitably small. The method proposed is an iterative refinement based on the LU-decomposition.

The technique of iterative refinement for improving the computed solution to (1) was probably first used in a computer program by Wilkinson in 1948, during the design and building of the ACE computer at the National Physical Laboratory [7]. Iterative refinement has achieved wide use ever since and is exploited by most of the linear system expert drivers. For more details see [4,6] and references therein.

The sparse techniques and the LU-decomposition method are often used in (1). Assume that the sparse LU-decomposition method is used in the solution of (1). At the end of the LU-decomposition, an approximate LU-decomposition of the matrix A is normally obtained, in which a unit lower triangular matrix L and an upper triangular matrix U are found. For example

$$A = \tilde{L}\tilde{U} + E \tag{2}$$

is obtained by taking the Powers of a Boolean Matrix Strategy (PBS) described in Appendix I, where E is an error matrix. The approximate solution of the system is computed by

$$\tilde{L}\tilde{U}x^{(0)} = b \tag{3}$$

Assume that some sparse technique such as an LU-decomposition is used in the computation of (2) in the decomposition stage and (3) in the solution stage. The decomposition stage (2) is performed by using LU-decomposition to level m when $B^{(2^m)} = B^{(2^{m+1})}$ [5]. It is well known that the factorization stage is much more expensive than the solution stage. Therefore, it may be advantageous to use small values for m to control the sparsity. If this is done, then normally the computing time needed to obtain $x^{(0)}$ and the storage needed for the nonzero entries of \tilde{L} and \tilde{U} are reduced. However, the approximation $x^{(0)}$ so computed may be crude and an attempt to regain the accuracy lost by iterative refinement has to be carried out. This means that the computations should be continued after the solution stage (3) by the following formulae:

$$r^{(i)} = b - Ax^{(i)} \tag{4}$$

$$\tilde{L}\tilde{U}d^{(i)} = r^{(i)} \tag{5}$$

$$x^{(i+1)} = x^{(i)} + d^{(i)} \tag{6}$$

for $i = 0, 1, 2, 3, \dots$

Different criteria must be used to stop the iterative process (4)–(6) if the accuracy has not been achieved or if the process has not converged. Normally single precision computations are used in (5) and (6), while the residual vectors $r^{(i)}$, for $i = 1, 2, \dots$, are accumulated in double precision and then rounded to single precision. If $x^{(i)}$ is accepted as a solution of (1), then it is said that the system is solved directly or that $x^{(i)}$ is a **LU-decomposition direct solution (LUDS)**. The solution obtained by the use of (4)–(6) is called the **LU-decomposition iteratively refined solution (LUIR)**.

If the factorization (2) is obtained by PBS with m it may be inaccurate but it preserves the sparsity of A . If the iterative process (4)–(6) is applied, then one can expect the following statements to be true.

(i) The LU-decomposition obtained by (2) is not so accurate. The same is true for the solution $x^{(0)}$ obtained by (3). However, full machine accuracy is often achieved by the iterative process (4)–(6).

(ii) Sometimes the storage requirements is greater, because a copy of the nonzero entries of A is needed in (4); compare this with 100% extra storage needed when the matrix A is dense.

(iii) Often the reduction of the factorization time is so great that the total computing time may be reduced considerably.

(iv) The computational cost per iteration is small in comparison with the computation cost of the factorization, and therefore a reduction of the total computation time may be obtained even if the rate of convergence is slow. This situation happens when the matrix A is ill-conditioned.

(v) An estimation of the accuracy achieved is easily obtainable when the iterative process (4)–(6) is used.

Our main purpose is to verify by numerical experiments the validity of the above statements. A comparison of the use of sparse LU-decomposition method (LUDS) and the use of LUIR in the solution of (1) is presented in the section on numerical experiments. Finally, some concluding remarks are given in the last section. Many of these remarks are illustrated by numerical results.

2 Sparse LUIR Technique

A detailed presentation of the sparse technique proposed is needed to better understand the results in the numerical experiments section. Consider the problem: Find an approximation \tilde{x} to the solution x of (1) for which $\|\tilde{x} - x\| \leq \varepsilon$, where ε is some error tolerance prescribed in advance. The problem

is often solved in the following way. Let A be a square nonsingular matrix of order n . Assume that M is an approximation inversion of the matrix A so that MA is approximately the identity matrix I . Define the error matrix R of M as

$$R = I - MA \quad (7)$$

which is supposed to be small. Note that

$$MA = I - R \quad (8)$$

where the size of the entries of the matrix R may be significant relative to those of M .

Consider the following formal manipulation:

$$\begin{aligned} A^{-1} &= A^{-1}(M^{-1}M) = (A^{-1}M^{-1})M = (MA)^{-1}M \\ &= (I - R)^{-1}M = (I + R + R^2 + \dots)M \end{aligned} \quad (9)$$

Assume that the sparse LU-decomposition is used in the factorization of the sparse matrix A . The details of the sparse LU-decomposition used is described in [5]. The LU-decomposition of A is obtained by taking the Powers of a Boolean Matrix Strategy (PBS) described in Appendix I. The LU-decomposition of A is obtained when $B^{(m)} = B^{(m+1)}$. Let $M^{(0)} = M$. Then we can define the m th partial sum of the expression (9) by

$$M^{(m)} = (I + R + R^2 + \dots + R^m)M^{(0)} \quad (10)$$

for large m (the limit exists since PBS terminates when $B^{(m)} = B^{(m+1)}$). Thus, we have

$$M^{(m)} \approx (\tilde{L}\tilde{U})^{-1} \approx A^{-1} \quad (11)$$

Consequently, we obtain

$$x^{(m+1)} = M^{(m)}b \approx (\tilde{L}\tilde{U})^{-1}b \approx \tilde{x} \quad (12)$$

or

$$\tilde{L}\tilde{U}\tilde{x} \approx b \quad (13)$$

The solution of (13) may differ substantially from the actual solution to the original system (1), even if the problem is well-conditioned. We now discuss a strategy designed to improve the accuracy of a computed solution \tilde{x} of (13). Because we are defining an iterate process, we change notation slightly and let $x^{(0)}$ be the computed solution to (1). To simplify, we use $M \approx (\tilde{L}\tilde{U})^{-1}$ throughout rather than $M^{(m)}$. Then let the residual vector be

$$r^{(0)} = b - Ax^{(0)} \quad (14)$$

Formally, we observe the relations

$$Mr^{(0)} = M(b - Ax^{(0)}) \approx Mb - x^{(0)} \quad (15)$$

By using (12), we have

$$Mr^{(0)} = x^{(1)} - x^{(0)} \quad \Rightarrow \quad x^{(1)} = x^{(0)} + Mr^{(0)} \quad (16)$$

Suppose $d^{(0)} = Mr^{(0)}$. Then solve

$$\tilde{L}\tilde{U}d^{(0)} = r^{(0)} \quad (17)$$

In general, define an iterate process with steps

$$r^{(i)} = b - Ax^{(i)} \quad (18)$$

$$\tilde{L}\tilde{U}d^{(i)} = r^{(i)} \quad (19)$$

$$x^{(i+1)} = x^{(i)} + d^{(i)} \quad (20)$$

for $i = 0, 1, 2, \dots$, where the starting $x^{(0)}$ is usually taken so that it is the solution of $\tilde{L}\tilde{U}x^{(0)} = b$. The iterative process described by equations (18)–(20) defines the iterative refinement (LUIR). The LUIR is in principle a direct method. The significant feature of the LUIR method is that a good approximation to the solution can be obtained, generally, after a much smaller number of iterations. The goal of reducing the solution time is achieved if the matrix is decomposed in an efficient manner [5]. Assume again that the LUIR is used in the solution of (1). The sparse LU-decomposition used in (19) is approximately the exact factorization of A (see [5]; Theorem 2). The major drawback of the LU-decomposition method is that L and U are no longer sparse due to fill-ins, so computer storage demands are high. Moreover, the computational work for constructing the factors increases considerably with the dimensions of the problem under study. Often, especially in problems arising from practical fields, matrices that produce more fill-ins (when $B^{(m)} = B^{(m+1)}$) appear and the use of the iterative refinement (LUIR) for such matrices is extremely expensive because both the time and the computer storage needed to compute and store the factors are large. The basic idea behind the LUIR method is to first reduce the computer storage requirements to store the factors L and U by determining the fill-ins to the level $\ell < m$ and secondly to increase the accuracy of the solution vector of (1).

A discussion of the PBS used in the LU-decomposition is needed in order to explain how the accuracy requirements can be relaxed in an attempt to achieve a better preservation of the sparsity of the LU factors by controlling the values of m . Assume that the PBS terminates for $\ell = 0, 1, 2, \dots, m - 1$ (when $B^{(m)} = B^{(m+1)}$), then we have

$$M^{(\ell)} = (I + R + R^2 + \dots + R^\ell)M^{(0)} \quad (21)$$

and we get

$$M^{(\ell)} \neq A^{-1} \quad (22)$$

From (22), it has been shown that the approximate inverse $M^{(\ell)}$ tends to A^{-1} when ℓ is increasing but $M^{(\ell)}$ may never be A^{-1} . By (22), it is necessary to point out that when small values of m are used, an attempt to improve the results by the iterative process (18)–(20) has to be performed. Consequently, the solution of (1) can be obtained by the following algorithm.

Algorithm LUIR

Step 1. Decomposition stage:

- i. Determine the sparsity pattern of the factors \tilde{L} and \tilde{U} to the level ℓ by using PBS, where $\ell \in \{0, 1, 2, \dots, m-1\}$ for the value of m satisfying $B^{(2^m)} = B^{(2^{m+1})}$.
- ii. Compute the factors \tilde{L} and \tilde{U} using incomplete LU-decomposition method described in [5].

Step 2. First solution:

- i. Solve $\tilde{L}\tilde{U}x^{(0)} = b$ for $x^{(0)}$ by using forward and back substitution, respectively.

Step 3. Improvement: For $i = 0, 1, 2, \dots$, until the desired accuracy is achieved (say, 10^{-16})

- i. Compute $r^{(i)} = b - Ax^{(i)}$
- ii. Solve $\tilde{L}\tilde{U}d^{(i)} = r^{(i)}$ for $d^{(i)}$
- iii. Compute $x^{(i+1)} = x^{(i)} + d^{(i)}$

From the algorithm, it can be easily seen that it consists of two steps: (i) decomposition stage and (ii) solution stage. In the first step, the factorization of the original matrix is obtained to the level $\ell < m$. In the second step, refinement is used to rapidly improve the accuracy of the solution.

Step 3 of the LUIR Algorithm is optional. One can accept $x^{(0)}$ (hoping that $\|x^{(0)} - x\| \leq \varepsilon$). Then $\tilde{x} = x^{(0)}$ is called the direct solution (LUDS). If the third step is carried out and if the process converges, then $\tilde{x} = x^{(i)}$ and $\|x^{(i)} - x\| \leq \varepsilon$. The iteratively refined solution (LUIR) is normally more accurate than $x^{(0)}$ and an estimation of $\|\tilde{x} - x\|$ is computed by $\|d^{(i)}\|$. Unfortunately, some extra storage and time are also needed when Step 3 is applied. This point is discussed in the section on numerical experiments.

3 Numerical experiments

The results obtained in the solutions of some numerical examples are discussed in this section. The computational results described here were performed on an IBM compatible PC with a Pentium II processor. The matrices used in the experiments are chosen randomly. The sparsity pattern of the matrices is given in Examples 2–6 contain nonzero entries on the diagonal and distributed sparsely elsewhere. Displays of the nonzero entries can be found in [5] and all the matrices are general and nonsymmetric. PBS, incomplete LU-decomposition, and forward and back substitutions are used in all experiments. The algorithms used to solve the systems considered are the iterative refinement (LUIR) and the sparse LU-decomposition (LUDS). Both options are used in the experiments. We shall discuss the accuracy of the computed solution, the storage needed, the computing time used, and the number of iterations performed.

Example 1. Consider the system (1) whose nonzero entries of the coefficient matrix A are given by $a_{ij} = 1/(i + j + 1)$. The matrix A is ill-conditioned for even modest size n and it has a large condition number. It is used to illustrate the performance of the algorithms. In this example, the dimensions of the matrices considered are $n = 10, 20, 40, 100$.

In Examples 2–6, we consider linear systems (1) whose coefficient matrices A are of order n with \mathbf{na} nonzero entries on the diagonal and sparsely distributed throughout as shown in [5]. For example, the largest eigenvalue λ_{\max} , the smallest eigenvalue λ_{\min} , and the condition number $\kappa(A)$ are given as follows.

Example 2. $n = 100$, $\mathbf{na} = 219$; $\lambda_{\max} = 7.213652$, $\lambda_{\min} = 0.5191232$,
 $\kappa(A) = 13.65975$

Example 3. $n = 200$, $\mathbf{na} = 438$; $\lambda_{\max} = 8.63221$, $\lambda_{\min} = 0.559632$,
 $\kappa(A) = 15.716318$

Example 4. $n = 300$, $\mathbf{na} = 657$; $\lambda_{\max} = 9.42263$, $\lambda_{\min} = 0.522163$,
 $\kappa(A) = 18.04538$

Example 5. $n = 400$, $\mathbf{na} = 876$; $\lambda_{\max} = 7.503768$, $\lambda_{\min} = 0.406214$,
 $\kappa(A) = 16.10840$

Example 6. $n = 1000$, $\mathbf{na} = 2190$; $\lambda_{\max} = 6.113192$, $\lambda_{\min} = 0.406214$,
 $\kappa(A) = 15.04919$

Example	1			
n	10	20	50	100
Accuracy	5.9E-11	1.5E-9	1.0E-10	2.4E-11

Example	2	3	4	5	6
n	100	200	300	400	1000
Accuracy	2.1E-15	1.3E-14	2.1E-15	1.4E-15	2.1E-15

Table 1

Max-norms of the error vectors for systems in Examples 1–6: direct solution (LUDS).

3.1 Accuracy

For all systems considered above, the true solution is $x = [1, 1, 1, \dots, 1]^T$. Single precision arithmetic is used and the systems are solved by iterative refinement with $\ell = 0, 1, 2, \dots, m - 1$ (where $B^{(2^m)} = B^{(2^{m+1})}$) and directly using sparse LU-decomposition method (LUDS) with m . When all the systems are solved directly by the sparse LU-decomposition method (LUDS) we have $\|\tilde{x} - x\| \in [10^{-12}, 10^{-8}]$ for matrices given in Example 1 and $\|\tilde{x} - x\| \in [10^{-15}, 10^{-13}]$ for matrices given in Examples 2–6. The results are given in Table 1. If LU-IR is used, then nearby full machine accuracy tolerance (10^{-16}) is achieved for all systems for $\ell < m$. The results obtained with $\ell = 0, 1, 2$ are given in Table 2. However, double precision arithmetic is used for ill-conditioned Example 1. From the Table 1, it is clear that if LUDS is used, then the accuracy achieved for all considered systems (ill-conditioned Example 1) and well-conditioned (Examples 2–6)) is satisfactory but the accuracy achieved for well-conditioned systems is much better. It is interesting to note that the accuracy achieved in LU-IR is zero for well-conditioned systems (Examples 2–6). In Example 1 when the accuracy of convergence in LU-IR of order 10^{-9} and a single precision arithmetic is used, the max-norms vectors $\|r^{(i)}\| \in [10^{-12}, 10^{-9}]$ and the norms of the correction vector $\|d^{(i)}\| \in [10^{-9}, 10^{-7}]$. If we use an accuracy tolerance less than 10^{-9} , the $\|d^{(i)}\|$ do not change. In both cases, the accuracy achieved with ill-conditioned systems requires more iterations to converge. This point is seen in the subsection on number of iterations. As a result, LU-IR is much better than LUDS in all considered systems (see Tables 1 and 2).

3.2 Storage requirements

The storage used in the implementations of the LU-IR and LUDS algorithms is based on the ideas proposed by Gustafson [3]. For more details on the storage of the factors L and U , see [5]. When LU-IR is used, the storage requires

Example	1				2	3	4	5	6
n	10	20	50	100	100	200	300	400	1000
Accuracy $\ell = 0$	7.8E-17	7.5E-17	1.9E-17	5.0E-17	0.0	0.0	0.0	0.0	0.0
$\ell = 1$	1.2E-17	1.8E-17	1.5E-17	3.9E-17	0.0	0.0	0.0	0.0	0.0
$\ell = 2$	0.0	1.2E-18	6.2E-18	1.3E-17	0.0	0.0	0.0	0.0	0.0

Table 2

Max-norms of the error vectors for systems in Examples 1–6: LUIR with accuracy tolerance 10^{-16} .

Example	n	LUDS		LUIR		
		na	na1	$\ell = 0$	$\ell = 1$	$\ell = 2$
1	10	33	76	33	51	67
	20	85	273	85	137	241
	40	137	453	137	281	327
	100	431	925	431	574	628
2	100	219	1224	219	478	898
3	200	438	2483	438	690	1248
4	300	657	3742	657	1014	1334
5	400	876	5091	876	1922	3103
6	1000	2190	11405	2190	2983	5154

Table 3

The values of parameters for Examples 1–6. The value of **na** is the number of nonzero entries in A and **na1** is given after the sparsity pattern of L and U is obtained.

storing the factors L and U and two extra real arrays, which are needed for residual vectors $r^{(i)}$, for $i = 0, 1, 2, \dots$, in (4), and for the correction vector $d^{(i)}$ in (5). However, the length of the arrays **VA** (a real array contains the nonzero entries of A) and **JA** (an integer array contains column indices correspond to the nonzero entries stored in **VA**) used to store L and U is known before the factorization is done. This is one of the basic features of PBS.

It is necessary to consider several different situations in order to explain the storage requirements when LUIR is used and when LUDS is acceptable.

(i) Assume that in both cases single precision is used and the same values of m is chosen (when $B^{(2^m)} = B^{(2^{m+1})}$). Assume also that the nonzero entries of L and U are stored when the system is solved directly (LUDS) and, therefore, the same must be stored with LUIR ($\ell = m$).

(ii) Let the first assumption in (i) be true again. Assume that the nonzero

entries of L and U are not stored in LUDS. Then extra storage is needed in LUIR.

(iii) Assume that the nonzero entries of L and U are not stored with the LUDS procedure. Assume that small values of m are used with LUIR. Then the storage requirements with LUIR algorithm may often be much smaller than those needed in (ii). This happens because the lengths of the arrays in row ordered list are smaller in these cases.

(iv) Denote the maximal number of nonzero entries kept in A at the level ℓ of factorization by $\mathbf{na1}$. Then the decrease of ℓ means a decreasing in $\mathbf{na1}$. That shows that the storage requirements decrease. Note that our experiments show that $\mathbf{na1}$ correspond to $\ell < m$ is a good choice for the length of \mathbf{VA} and \mathbf{JA} . In Table 3, the values of $\mathbf{na1}$ are given. When the systems are solved directly (LUDS), the values of $\mathbf{na1}$ are often bigger than those for LUIR. The extra storage requirement obtained by PBS is sufficient in LUIR where $\ell < m$ (relative to LUDS). Consider the ratio of the parameter $\mathbf{na1}$ found by the LUDS and \mathbf{na} the number of nonzero entries in the matrix A before the beginning of the factorization. Our observations show that if the matrix A produces a large s (that is, many fill-ins appear after using PBS), then the use of LUIR with small values of ℓ (with $\ell < m$) is successful (this can be seen from Table 3). There is a tendency that a considerable reduction of the storage requirements may appear just in the cases where this is needed.

(v) Assume that the nonzero entries of L and U have to be stored with LUDS. Let the other assumption made in (iii) hold. Then often the storage requirements with LUIR may be smaller; see Table 3.

(vi) Assume that the accuracy obtained by the LUDS is not efficient when single precision is used and therefore double precision must be used in order to reach the accuracy requirements. Assume that single precision with the LUIR procedure gives acceptable accuracy. In this case, the storage requirements with the LUDS algorithm may be much larger.

3.3 Computing time

The computing time may be reduced when LUIR is used for sparse systems (which may never happen in the case where the matrix is dense). Denote by t_1 , the computing time needed to solve the system by LUIR and by t_2 the computing time for the LUDS. Our experiments show that $t_1 < t_2$ for the accuracy shown in Table 1 and Table 2. The results are given in Table 4. Our experiments show that for all values of $\ell < m$, ($\ell = 0, 1, 2$), LUIR gives much better performance than LUDS. The computing time needed for the LUDS procedure is used is roughly increasing when n is increasing due to

Example	n	LUDS	LUIR		
			$\ell = 0$	$\ell = 1$	$\ell = 2$
1	10	negl.	negl.	negl.	negl.
	20	negl.	negl.	negl.	negl.
	40	negl.	negl.	negl.	negl.
	100	0.054945	negl.	negl.	negl.
2	100	0.054945	negl.	negl.	negl.
3	200	0.109891	negl.	negl.	negl.
4	300	0.164835	negl.	negl.	negl.
5	400	0.219780	negl.	0.054945	0.054945
6	1000	0.274328	0.054945	0.054945	0.109891

Table 4

The computing time (in seconds) needed to solve Examples 1–6 by using LUDS and LUIR.

more fill-ins. Thus, the efficiency of the LUIR algorithm is demonstrated.

3.4 Number of Iterations

The experiments show that the number of iterations decreases when ℓ is increased. However, this has no great influence on the total computing time. Nevertheless, the total computing time is normally reduced when the value of ℓ is small because the computational cost per iteration is much smaller than the computational cost per factorization. See the results presented in Table 5. From the tables, it is clear that the ill-conditioned systems in Example 1 need more iterations to converge. However, when the systems are well-conditioned (Examples 2–6), good values for ℓ are $\ell = 1$ or $\ell = 2$ and these values reduce both the storage and computing in the iterative process.

4 Concluding Remarks

4.1 About the Matrices Used

The matrices given in Example 1 are moderately ill-conditioned even when n is small. The condition number for these matrices are large. The spectral condition number for matrices with $n = 10, 20, 40, 100$ are in the interval $[10^4, 10^6]$.

Example	n	$\ell = 0$		$\ell = 1$		$\ell = 2$	
		Time	Iter.	Time	Iter.	Time	Iter.
1	10	negl.	10	negl.	8	negl.	6
	20	negl.	8	negl.	6	negl.	5
	40	negl.	15	negl.	13	negl.	10
	100	negl.	51	negl.	42	negl.	38
2	100	negl.	8	negl.	3	negl.	1
3	200	negl.	8	negl.	2	negl.	1
4	300	negl.	9	negl.	2	negl.	1
5	400	negl.	9	negl.	2	negl.	2
6	1000	0.054945	11	0.054945	3	0.109891	1

Table 5

The computing time (in seconds) and the number of iterations obtained by using LUIR for Examples 1–6.

Moreover, the matrices produce many fill-ins. Therefore, these matrices are a good illustration of the efficiency of LUIR (both in order to preserve the sparsity by controlling m and in order to improve the accuracy); see Subsection 3.1. The convergence of LUIR is slow. The matrices given in Examples 2–6 are well-conditioned. These matrices produce many fill-ins therefore they are a good illustration of the efficiency of LUIR. The convergence of LUIR is fast.

4.2 About the Convergence of LUIR

The LUIR algorithm may converge as long as the spectral radius of the matrix $I - (LU)^{-1}A$ is less than 1 and the error matrix R in (2) can be solved with quite large entries. The convergence is dependent on the value of this spectral radius and can be quite slow. We have illustrated the performance of this point in the numerical experiments section. If the condition number of the coefficient is large (see Example 1), then the LUIR algorithm may converge slowly with single precision arithmetic for accuracy tolerance less than 10^{-9} . It is possible to use double precision in this case and to solve the systems using the LUDS procedure. In this way, one may obtain an answer of acceptable (but unknown) accuracy. However, it is also possible to use LUIR with double precision if the accuracy is more important (but not computing time and computer storage). Moreover, the LUIR procedure converges to the true solution even though the matrix A is ill-conditioned.

4.3 On the Possibility of Achieving More Accurate Results

Assume that:

- (i) the vectors $r^{(i)}$, $x^{(i)}$ and $d^{(i)}$ are stored in double precision (where the length of the real numbers is n_2 binary digits),
- (ii) all inner products are accumulated in double precision,
- (iii) $n_2 \geq 2n_1$ (where the length of the real numbers is n_1 binary digits when single precision is used). Then about n_1 binary digits may always be gained if LUIR is used.

A version of LUIR, which exploits this idea, is tested by the matrices given in Example 1. We have found that max-norms of the errors were $\mathcal{O}(10^{-17})$ for all four systems of different dimensions. However, the number of iterations obtained were large.

4.4 On the Use of LUIR with Dense Systems and with Sparse Systems

Assume that A is dense. Consider the solution of (1) by LUDS and by LUIR. Then we have the following:

- (i) extra storage (100% when m is large) is needed when LUIR is used,
- (ii) the iterative process (18)–(20) requires extra computing time, and
- (iii) the solution obtained by LUDS is satisfactory, in general.

Therefore, LUDS should be preferred when the matrix A is dense. If the matrix is sparse and LUDS is used, we have found that the solution is satisfactory because the stability requirements in LU-decomposition are fulfilled by allowing more fill-ins by PBS. The solution is quite true (see Table 1). By the use of $\ell < m$, (when $B^{(2^m)} = B^{(2^{m+1})}$), one can reduce the storage and the computing time considerably (see Tables 2, 3 and 5). From these tables, it is clear that LUIR seems to be the better choice when sparse systems are to be solved. Thus, LUIR is much better than LUDS for sparse matrices.

4.5 When is the Use of $\ell > 0$ Most Efficient?

It is important to choose a good positive integer value of ℓ . By this, we mean such a positive value that leads to a reduction in the computing time and/or

the storage compared with the use of LUDS.

If $B^{(2^m)} = B^{(2^{m+1})}$ and the accuracy is more important (but not the storage and the computing time), then the LUIR procedure should be preferred over the LUDS procedure. All experiments show only that the use of LUIR with $\ell < m$ could be more efficient than the use of LUDS when $B^{(2^m)} = B^{(2^{m+1})}$ in the sense that the storage and/or the computing are smaller. The efficiency of the LUIR when $\ell < m$ depends on m . When m is large, then the use of LUIR with small $\ell > m$ is efficient. Especially in the problems arising from some practical fields, matrices appear, which produce $\mathbf{na1} \gg \mathbf{na}$. For such matrices, the use of LUIR with $\ell < m$ is extremely efficient, in general. Thus, by controlling the value ℓ , both computing time and the storage needed to compute and to store the factors L and U may be reduced. Our experiments support that $\ell = 1$ or $\ell = 2$ are efficient values.

5 Conclusion

If LU-decomposition with iterative refinement (LUIR) is used in the solution of systems of linear algebraic equations $Ax = b$ whose coefficient matrices are dense, then the accuracy of the results may usually be greater than the accuracy obtained by the use of the LU-decomposition with direct solution (LUDS); that is, without iterative refinement. However with LUIR, both more storage (about 100%, because a copy of matrix A is needed) and more computing time (some extra time is needed to perform the iterative process) must be used. Normally, when the matrix A is sparse the accuracy of the solution computed by some sparse matrix techniques and LUIR may still be greater. We have verified (by numerical experiments) that the use of sparse matrix techniques with LUIR may also result in a reduction of both the computing time and the storage requirements (this may never happen when LUIR is applied for dense matrices). The Powers of a Boolean Matrix Strategy (PBS) is used in the efforts to achieve such a reduction. By the use of $\ell < m$, where $B^{(2^m)} = B^{(2^{m+1})}$, an attempt is made to control the sparsity of the matrix A to level ℓ . The use of $\ell = 0, 1, 2, \dots, m - 1$ leads to an inaccurate factorization, but the accuracy lost is normally regained in the iteration process. Many examples are given in order to compare the use of LUIR with values for ℓ (with $m > \ell \geq 0$), and the use of LUDS with m in the solution of systems whose matrices are large and sparse. The main conclusion is that the iterative refinement procedure may be used efficiently as a good option in packages for the solution of sparse linear systems of equations.

Finally, it should be mentioned that if matrix A is symmetric positive definite, then one could attempt to exploit these properties. In this case: (a) the LU-decomposition can be replaced by the LL^T factorization, (b) only the nonzero

entries located on and over the main diagonal have to be stored and used in the computations, and (c) the PBS becomes simple and cheap. In such cases, the LL^T preconditioned CG method should be preferred. Otherwise, the idea given in the LUIR algorithm can be exploited for symmetric and positive definite matrices as well.

We recommend the consideration of the proposed LUIR procedure in software packages for solving sparse systems of linear equations.

Appendix I. Powers of a Boolean Matrix Strategy (PBS) [5].

The Powers of a Boolean Matrix Strategy for finding the sparsity pattern of LU factors for the matrix $A = (a_{ij})$ is given as follows:

Step 1.

Form the matrix $B^{(0)} = (b_{ij}^{(0)})$ as given by $b_{ij}^{(0)} = \begin{cases} 1, & \text{if } a_{ij} \neq 0 \\ 0, & \text{otherwise} \end{cases}$

Step 2.

Compute $B^{(m)}$ where $m \geq 2$.

If $B^{(m)} = B^{(m+1)}$, then form the sparsity pattern of LU , say P , as $P = \{(i, j) : b_{ij}^{(m)} = 1\}$
else $m = m + 1$ and go to Step 2.

We can speed up the computation and thereby reduce the execution time by computing the powers $B^{(2^m)}$, ($1 \leq 2^m \leq n - 1$). The PBS terminates when $B^{(2^m)} = B^{(2^{m+1})}$. Then Step 2 of PBS becomes:

Step 2.

Compute $B^{(2^m)}$ where $m \geq 1$.

If $B^{(2^m)} = B^{(2^{m+1})}$, then form the sparsity pattern of LU , say P , as $P = \{(i, j) : b_{ij}^{(2^m)} = 1\}$
else $m = m + 1$ and go to Step 2.

The sparsity pattern of LU at level ℓ is approximately equal to that of $B^{(2^\ell)}$. From the PBS, it follows that while finding powers of B , some zero elements in B become 1 in $B^{(2^m)}$, which causes fill-in. These elements are precisely the positions of fill-ins in A .

Suppose that $U = (u_{ij})$ and $V = (v_{ij})$ are square Boolean matrices of order n and let \wedge be the Boolean operator **and** defined on the entries of U and V . The Boolean product of U and V , denoted $U \wedge V$, is the Boolean matrix $W = (w_{ij})$ defined by

$$w_{ij} = \begin{cases} 1 \text{ (True)}, & \text{if } u_{ik} = 1 \wedge v_{kj} = 1 \text{ for some } k, \text{ for } 1 \leq k \leq n \\ 0 \text{ (False)}, & \text{otherwise} \end{cases}$$

It follows that $w_{ij} = 1$ if and only if $u_{ik} = 1$ and $v_{kj} = 1$ for some k .

References

- [1] A. H. Al-Kurdi, R. C. Mittal, and D. R. Kincaid, Performance of various preconditioned $GCR(k)$ algorithms applied to sparse nonsymmetric linear systems, *International Journal Applied Science & Computations*, **9**, 11–31 (2002).
- [2] G. Golub and C. van Loan, *Matrix Computation*, 2nd ed., Johns Hopkins University Press, 1989.
- [3] F. G. Gustavson, Some basic techniques for solving sparse systems of linear equations, in *Sparse Matrices and Their Applications*, D. J. Rose and R. A. Willoughby, eds., Plenum Press, New York, 1972, 41–52.
- [4] N. J. Higham, Iterative refinement for linear systems and LAPACK, *IMA Journal Numerical Analysis*, **17**, 495–509 (1997).
- [5] R. C. Mittal and A. H. Al-Kurdi, LU-decomposition and numerical structure for solving large sparse nonsymmetric linear systems, *Computers Mathematical Applications*, **4**, 131–155 (2002).
- [6] Z. Zlatev, Use of iterative refinement in the solution of sparse linear systems, *SIAM Journal Numerical Analysis*, **19**, 381–399 (1982).
- [7] J. M. Wilkinson, Progress report on the automatic computing engine, Report MA/17/1024, Mathematics Division, Department of Scientific and Industrial Research, National Physical Laboratory, Teddington, United Kingdom, April 1948.