Computers Math. Applic. Vol. 25, No. 4, pp. 95–104, 1993
Printed in Great Britain.

0097-4943/93 $5.00 + 0.00

A NEW CLASS OF DECOMPOSITION FOR INVERTING ASYMMETRIC AND INDEFINITE MATRICES

JENN-CHING LUO
Department of Civil Engineering and Engineering Mechanics, Columbia University
New York, NY 10027, U.S.A.

(Received April 1992)

Abstract—An innovative decomposition for inverting a nonsingular, asymmetric, and indefinite matrix $[A]$ of order $(n \times n)$ is derived in this paper. The inverse of $[A]$ is written as $[A]^{-1} = [L][D][U]$ where $[L]$ is a lower triangular matrix, $[D]$ is a diagonal matrix, and $[U]$ is an upper triangular matrix. By the method, the solution of $[A]\{X\} = \{B\}$ may be easily and efficiently computed by matrix-vector multiplications as $\{X\} = [L][D][U]\{B\}$. This technique requires a minimal amount of computer memories, and can be easily transformed into parallel procedures with high efficiencies. Performances in inverting asymmetric and indefinite matrices and in solving systems of linear equations on an Alliant/FX8 computer are reported.

1. INTRODUCTION

Inverting a matrix is one of the fundamental procedures in scientific computing. It is well-known that many scientific and engineering problems, written in differential equations or integral equations, may be approximated into an algebraic system as

$$[A]\{X\} = \{B\}, \tag{1}$$

where $[A]$ is a nonsingular matrix, $\{B\}$ is a coefficient vector, and $\{X\}$ is the unknown vector to be determined. If the inverse of $[A]$ is available, then the solution of Equation (1) may be easily and efficiently computed by a matrix-vector product. For some types of problems, matrix $[A]$ may be symmetric and positive definite, while others may result in asymmetric and indefinite matrices. This paper considers the general case with an asymmetric and indefinite matrix $[A]$.

The essential idea of this work is based upon the author's decomposition [1], which can efficiently decompose a matrix into its inverse. The original method [2] decomposes a symmetric matrix $[A]$ into $[A]^{-1}$, and $[A]^{-1}$ is written in the form

$$\begin{aligned} &[\overline{L}]\,[\overline{L}]^{\mathsf{T}}, && \text{if } [A] \text{ is positive definite;} \\ &[L]\,[D]\,[L]^{\mathsf{T}}, && \text{if } [A] \text{ is indefinite.} \end{aligned}$$

This work will extend this new class of technique to decompose an asymmetric matrix into its inverse. The inverse of an asymmetric matrix can be then written in the form of $[L][D][U]$ where $[L]$ is a lower triangular matrix, $[D]$ is a diagonal matrix, and $[U]$ is an upper triangular matrix. Based upon the method, the solution of $[A]\{X\} = \{B\}$ may be efficiently computed by matrix-vector multiplications as

$$\{X\} = [L][D][U]\{B\}.$$

The method will be discussed in the following.

The author acknowledges the Advanced Computing Research Facility, Mathematics and Computer Science Division, Argonne National Laboratory, on whose machines the computations of this research were performed.

2. DERIVATIONS

Let us consider a lower triangular matrix $[L]$ of order $(n \times n)$ with unit diagonal coefficients as

$$[L] = [\{L_1\}\ \{L_2\} \cdots \{L_j\} \cdots \{L_n\}], \tag{2}$$

where $\{L_j\}$ is the j^{th} column vector of $[L]$. Denote by L_{ij} the i^{th} coefficient of $\{L_j\}$, in which $L_{ij} = 0$ for $i < j$ and $L_{jj} = 1$. Similarly, let us consider an upper triangular matrix $[U]$ of order $(n \times n)$ with unit diagonal coefficients as

$$[U] = \begin{bmatrix} \lfloor U_1 \rfloor \\ \lfloor U_2 \rfloor \\ \vdots \\ \lfloor U_j \rfloor \\ \vdots \\ \lfloor U_n \rfloor \end{bmatrix}, \tag{3}$$

where $\lfloor U_j \rfloor$ is the j^{th} row vector of $[U]$. Denote by U_{ji} the i^{th} coefficient of $\lfloor U_j \rfloor$, in which $U_{ji} = 0$ for $j > i$ and $U_{jj} = 1$.

Let us consider a system of linear equations as

$$[A]\{X\} = \{B\}, \tag{4}$$

where matrix $[A]$ is a nonsingular square matrix which may be asymmetric and indefinite, $\{B\}$ is a coefficient vector, and $\{X\}$ is the solution to be determined. Since $[L]$ is a lower triangular matrix with unit diagonal coefficients, $\det[L] = 1$. This means $[L]$ is nonsingular, such that we may define a transformation in term of $[L]$ as

$$\{X\} = [L]\{\tilde{X}\}. \tag{5}$$

Substituting Equation (5) into Equation (4) yields

$$[A][L]\{\tilde{X}\} = \{B\}. \tag{6}$$

Premultiplying both sides in Equation (6) by $[U]$ yields

$$[U][A][L]\{\tilde{X}\} = [U]\{B\}. \tag{7}$$

If $[U][A][L]$ is a diagonal matrix, then $[U][A][L]$ may be easily inverted and may be written as

$$[U][A][L] = [D]^{-1}, \tag{8}$$

where $[D]$ is a diagonal matrix. Substitute Equation (8) into Equation (7) to yield

$$\{\tilde{X}\} = [D][U]\{B\}. \tag{9}$$

Premultiply both sides in Equation (9) by $[L]$ and apply Equation (5) to yield

$$\{X\} = [L][D][U]\{B\}. \tag{10}$$

Compare the results between Eqsuation (4) and (10) to obtain

$$[A]^{-1} = [L][D][U]. \tag{11}$$

The key to Equation (11) is the condition shown in Equation (8). If we can obtain a pair of $[L]$ and $[U]$ such that the product of $[U][A][L]$ is a diagonal matrix, then the inverse of $[A]$ can

be written in the form of Equation (11). Apparently, based upon Equation (11), the solution of Equation (4) may be easily and efficiently computed by matrix-vector multiplications as shown in Equation (10). The definitions of $[L]$, $[D]$, and $[U]$ for the inverse of $[A]$ will be derived in the following. First of all, let us discuss the conditions for decomposing $[A]$ into $[A]^{-1} = [L][D][U]$.

CONDITIONS. *The requirements for Equation (11) can be written as:*

$$\lfloor U_i \rfloor [A] \{L_j\} = 0, \qquad \text{if } i \neq j, \tag{12}$$

$$\lfloor U_i \rfloor [A] \{L_j\} \neq 0, \qquad \text{if } i = j. \tag{13}$$

PROOF. As mentioned previously, the key to Equation (11) is to find a pair of $[L]$ and $[U]$ such that $[U][A][L]$ is *diagonal* and *invertible*. This means that the off-diagonal coefficients of $[U][A][L]$ should be zero as shown in Equation (12), and the diagonal coefficients of $[U][A][L]$ cannot be zero as shown in Equation (13). ∎

Based upon Equation (12) and (13), a procedure for constructing $[L]$ and $[U]$ is arranged in backward order, i.e., $[L]$ is constructed from column n to column 1, and $[U]$ is constructed from row n to row 1. The definitions of $\{L_j\}$ and $\lfloor U_j \rfloor$ are as follows.

LEMMA 1. *A pair of $\{L_j\}$ and $\lfloor U_j \rfloor$, which satisfy the requirement of triangular configurations and may be determined by Equation (12), may be written as*

$$\lfloor U_j \rfloor = \lfloor e_j \rfloor + \sum_{i=j+1}^{n} \alpha_i \lfloor U_i \rfloor, \tag{14}$$

$$\{L_j\} = \{e_j\} + \sum_{i=j+1}^{n} \beta_i \{L_i\}, \tag{15}$$

where $\lfloor e_j \rfloor$ is an unit row vector with one non-zero coefficient in the j^{th} entry, $\{e_j\}$ is an unit column vector with one non-zero coefficient in the j^{th} entry, and α_i and β_i are coefficients to be determined.

PROOF. There are two things to be discussed. First, let us prove the requirement of triangular configurations. By Equation (14), the k^{th} coefficient of $\lfloor U_j \rfloor$ may be written as

$$U_{jk} = e_{jk} + \sum_{i=j+1}^{n} \alpha_i U_{ik}. \tag{16}$$

When $j > k$, $e_{jk} = 0$; the lower bound of i in Equation (16) is $(j+1)$, which means $i \geq j+1 > k+1 > k$ such that $U_{ik} = 0$. Then, Equation (16) becomes

$$U_{jk} = 0 \qquad \text{when } j > k. \tag{17}$$

Furthermore, when $k = j$, Equation (16) becomes

$$U_{jj} = 1 + \sum_{i=j+1}^{n} \alpha_i U_{ij}. \tag{18}$$

Since $i \geq j+1 > j$, $U_{ij} = 0$ and Equation (18) becomes

$$U_{jj} = 1. \tag{19}$$

Eqsuation (17) and (19) show that $\lfloor U_j \rfloor$ satisfies the requirement for the upper triangular configuration. Similarly, we can prove that $\{L_j\}$ satisfies the requirement for the lower triangular configuration. Second, let us prove the determinateness. The unknowns in Equation (14) are α_i $(i = j+1 \rightarrow n)$, i.e., Equation (14) contains $(n-j)$ unknowns. Similarly, Equation (15) also contains $(n-j)$ unknowns. Therefore, the pair of $\lfloor U_j \rfloor$ and $\{L_j\}$ contain $2(n-j)$ unknowns.

Basically, the $2(n-j)$ unknowns have to be determined by $2(n-j)$ conditions. By Equation (12), we may write the corresponding $2(n-j)$ conditions as

$$\lfloor U_j \rfloor [A] \{L_i\} = 0, \qquad i = j+1 \to n, \tag{20}$$

$$\lfloor U_i \rfloor [A] \{L_j\} = 0, \qquad i = j+1 \to n. \tag{21}$$

This shows that Equation (14) and (15) are 'determinate,' i.e., a pair of $\{L_j\}$ and $\lfloor U_j \rfloor$ may be written as Equations (14) and (15). This completes the proof. ∎

Equations (14) and (15) are two possible forms for $\lfloor U_j \rfloor$ and $\{L_j\}$, respectively. However, the existence of $\lfloor U_j \rfloor$ and $\{L_j\}$ depends on if the conditions, i.e., Equations (20) and (21), are solvable. The existence of $\{L_j\}$ and $\lfloor U_j \rfloor$ may be discussed by the $(n-j)\times(n-j)$ ***tailing principal submatrix*** of $[A]$, denoted by $[A^{(n-j)}]$ where the superscript indicates the order of a submatrix. The ***tailing principal submatrices***, opposed to the ***leading principal submatrices***, lie on the down right-hand corner of $[A]$; for example, if

$$[A] = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix},$$

then

$$[A^{(1)}] = [A_{33}],$$

$$[A^{(2)}] = \begin{bmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{bmatrix},$$

$$[A^{(3)}] = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}.$$

The existence of $\{L_j\}$ and $\lfloor U_j \rfloor$ will be discussed as follows.

LEMMA 2. *A pair of $\lfloor U_j \rfloor$ and $\{L_j\}$ in the form of Equations (14) and (15) exist if the tailing principal submatrix $[A^{(n-j)}]$ is nonsingular.*

PROOF. The existence of $\lfloor U_j \rfloor$ in the form of Equation (14) depends on if Equation (20) is solvable, and Equation (20) can be rewritten as

$$\lfloor U_j \rfloor [A] [L]_{(j+1)\to n} = 0, \tag{22}$$

where $[L]_{(j+1)\to n} = [\{L_{j+1}\}\ \{L_{j+2}\} \cdots \{L_n\}]$. Since all the coefficients before row $(j+1)$ in $[L]_{(j+1)\to n}$ are zero, Equation (22) may be simplified as

$$\lfloor U_j \rfloor [A]_{(j+1)\to n} [L^{(n-j)}] = 0, \tag{23}$$

where $[L^{(n-j)}]$ is the $(n-j)\times(n-j)$ ***tailing principal submatrix*** of $[L]$, and $[A]_{(j+1)\to n} = [\{A_{j+1}\}\ \{A_{j+2}\} \cdots \{A_n\}]$. Since $U_{ji} = 0$ $(j > i)$ and $U_{jj} = 1$, Equation (23) may be further simplified as

$$\lfloor U_{j(j+1)}\ U_{j(j+2)} \cdots U_{jn} \rfloor [A^{(n-j)}][L^{(n-j)}] + \lfloor A_{j(j+1)}\ A_{j(j+2)} \cdots A_{jn} \rfloor [L^{(n-j)}] = 0. \tag{24}$$

The condition for the existence of the unknowns $U_{j(j+1)}, U_{j(j+2)}, \ldots$, and U_{jn} in Equation (24) is obviously that $[A^{(n-j)}][L^{(n-j)}]$ is nonsingular. This implies that $[A^{(n-j)}]$ is nonsingular because $\det[L^{(n-j)}] = 1$. Similarly, we can prove if $[A^{(n-j)}]$ is nonsingular, then the unknowns $L_{(j+1)j}$, $L_{(j+2)j}, \ldots$, and L_{nj} in Equation (21) are solvable. This completes the proof. ∎

The existence for $[L]$ and $[U]$ can be guaranteed if the ***tailing principal submatrices*** of $[A]$, i.e., $[A^{(1)}]$, $[A^{(2)}], \ldots$, and $[A^{(n)}]$, are nonsingular. Certainly, it is possible for a ***tailing principal submatrix*** to be singular, even though $[A]$ is nonsingular; for example $[A] = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$ is

nonsingular, but $[A^{(1)}] = [0]$ is singular. Similar to the Gaussian elimination, which is a failure when dividing by a zero diagonal coefficient, exchanging rows (pivoting) is also a strategy to rearrange a matrix such that the *tailing principal submatrices* are nonsingular; for example the rows of $\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$ may be exchanged to be $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$, in which each *tailing principal submatrix* is nonsingular. Properly exchanging rows (pivoting) of a nonsingular matrix $[A]$ may guarantee the existence of $[L]$ and $[U]$, i.e., the solvability of the first requirement in Equation (12). The second requirement shown in Equation (13) will be discussed as follows.

LEMMA 3. *For a given nonsingular* $[A]$, *if matrices* $[L]$ *and* $[U]$ *exist, then the requirement in Equation (13) is automatically satisfied.*

PROOF. Since the triangular matrices $[L]$ and $[U]$ with unit diagonal coefficients exist, $\det[L] = \det[U] = 1$ and $\det([U][A][L]) = \det[A]$. If $[A]$ is nonsingular, then $\det[A] \neq 0$ which implies

$$\det([U][A][L]) \neq 0. \tag{25}$$

Furthermore, since $[L]$ and $[U]$ exist, which means that $\lfloor U_i \rfloor [A] \{L_j\} = 0$ $(i \neq j)$ and

$$[U][A][L] \text{ is a diagonal matrix.} \tag{26}$$

Equations (25) and (26) imply that the product of $[U][A][L]$ does not have zero diagonal coefficients, i.e., $\lfloor U_j \rfloor [A] \{L_j\} \neq 0$. This completes the proof. ∎

By Lemmas 2 and 3, we understand that a nonsingular matrix $[A]$ can be decomposed into $[A]^{-1}$ (with pivoting if necessary), and $[A]^{-1}$ may be written in the form of $[L][D][U]$. After discussing the existence of $[L]$, $[D]$, and $[U]$, the underlying subject is to determine them.

LEMMA 4. *For a given nonsingular* $[A]$, *if* $\{L_j\}$ *and* $\lfloor U_j \rfloor$ *exist, then the coefficients* α_i *and* β_i *in Equations (14) and (15) may be respectively written as*

$$\alpha_i = -D_{ii}(\lfloor A_j \rfloor \{L_i\}), \tag{27}$$

$$\beta_i = -D_{ii}(\lfloor U_i \rfloor \{A_j\}), \tag{28}$$

where $D_{ii} = 1/(\lfloor U_i \rfloor [A] \{L_i\})$, $\lfloor A_j \rfloor$ *is the* j^{th} *row vector of* $[A]$, *and* $\{A_j\}$ *is the* j^{th} *column vector of* $[A]$.

PROOF. Post-multiply each term in Equation (14) by $[A]\{L_k\}$ (where $k > j$) to yield

$$\lfloor U_j \rfloor [A] \{L_k\} = \lfloor A_j \rfloor \{L_k\} + \sum_{i=j+1}^{n} \alpha_i \lfloor U_i \rfloor [A] \{L_k\}. \tag{29}$$

By Equation (12), the left hand-side in Equation (29) is zero. By Equation (13), the second term in the right hand-side of Equation (29) is $\alpha_i \lfloor U_i \rfloor [A] \{L_i\}$ where $k = i$. Therefore, Equation (29) may be simplified as

$$\lfloor A_j \rfloor \{L_i\} + \alpha_i \lfloor U_i \rfloor [A] \{L_i\} = 0. \tag{30}$$

By Lemma 3, we know $\lfloor U_i \rfloor [A] \{L_i\} \neq 0$, and Equation (30) becomes

$$\alpha_i = -\frac{\lfloor A_j \rfloor \{L_i\}}{\lfloor U_i \rfloor [A] \{L_i\}} = -D_{ii}(\lfloor A_j \rfloor \{L_i\}).$$

Similarly, we can pre-multiply each term in Equation (15) by $(\lfloor U_k \rfloor [A])$ where $k > j$, and apply Equations (12) and (13) and Lemma 3 to obtain

$$\beta_i = -D_{ii}(\lfloor U_i \rfloor \{A_j\}).$$

This completes the proof. ∎

Equations (14), (15), (27), and (28) determine $\{L_j\}$ and $\lfloor U_j \rfloor$. As mentioned previously, the procedure for constructing $[L]$ and $[U]$ is from $j = n$ to $j = 1$. Once a pair of $\{L_j\}$ and $\lfloor U_j \rfloor$ have been obtained, $D_{jj} = 1/(\lfloor U_j \rfloor [A] \{L_j\})$ can be computed. However, $D_{jj} = 1/(\lfloor U_j \rfloor [A] \{L_j\})$ is an expensive expression. An alternative is discussed as follows.

LEMMA 5. *The j^{th} diagonal coefficient D_{jj} of $[D]$ may be written as*

$$D_{jj} = \frac{1}{\lfloor U_j \rfloor \{A_j\}}. \tag{31}$$

PROOF. Pre-multiply each term in Equation (15) by $\lfloor U_j \rfloor [A]$ to yield

$$\lfloor U_j \rfloor [A] \{L_j\} = \lfloor U_j \rfloor [A] \{e_j\} + \sum_{i=j+1}^{n} \beta_i \lfloor U_j \rfloor [A] \{L_i\}.$$

By Equation (12), the second term in the right-hand side of the previous equation is zero. This shows that

$$\lfloor U_j \rfloor [A] \{L_j\} = \lfloor U_j \rfloor \{A_j\},$$

where $\{A_j\} = [A]\{e_j\}$, and implies that

$$D_{jj} = \frac{1}{\lfloor U_j \rfloor \{A_j\}}.$$

This completes the proof. ∎

Equation (31) computes the j^{th} diagonal coefficient of $[D]$. Based upon Lemmas 1–5, a non-singular matrix $[A]$ is decomposed into $[A]^{-1} = [L][D][U]$. Algorithms for computing Equations (14), (15), (27), (28), and (31) will be discussed in the following section.

3. COMPUTATIONAL CONSIDERATIONS

A computational algorithm transforms mathematical equations into a procedure. Several factors have to be considered when developing an algorithm, which usually include computer memories, storage schemes, possibilities of the reduction in arithmetic operations, and parallelism. Certainly, it is not always possible to optimize these factors in an algorithm. In some situations, we can use a small amount of computer memories to save temporary results so as to avoid the repeat of certain arithmetic operations; on the other hand, if the requirement of more computer memories to save some information is impossible, we can re-compute such information when necessary. Parallel computations also require temporary computer memories to increase the degree of data independence, and to improve the parallelism. It usually happens in an algorithm that when a factor has been optimized, another factor may go worse. The most common procedure to design an algorithm begins with the consideration of computer memories, and then modifies the steps with a few computer memories so as to improve performances.

A sequential algorithm with the minimal computer memories for the computation of Equations (14), (15), (27), (28), and (31) is as follows.

ALGORITHM I. *A Sequential Procedure for Decomposing $[A]$ into $[A]^{-1} = [L][D][U]$.*

For $j = n \to 1$ with step (-1), do

(a) For $i = j + 1 \to n$ with step 1 do

$$A_{ji} \leftarrow A_{ji} + \sum_{k=i+1}^{n} A_{jk} * A_{ki};$$

(b) For $i = n \to j + 1$ with step (-1) do

$$A_{ji} \leftarrow -A_{ji} * A_{ii} - \sum_{k=j+1}^{i-1} A_{jk} * A_{kk} * A_{ki};$$

$$(c)\ A_{jj} \leftarrow \frac{1}{A_{jj} + \sum_{k=j+1}^{n} A_{jk} * A_{kj}};$$

(d) For $i = j + 1 \rightarrow n$ *with step* 1 *do*

$$A_{ij} \leftarrow A_{ij} + \sum_{k=i+1}^{n} A_{ik} * A_{kj};$$

(e) For $i = n \rightarrow j + 1$ *with step* (-1) *do*

$$A_{ij} \leftarrow -A_{ii} * A_{ij} - \sum_{k=j+1}^{i-1} A_{ik} * A_{kk} * A_{kj}.$$

In this algorithm, all the decomposed matrices share the computer memories with the original matrix, so that no additional computer memories are required. When $i > j$, L_{ij} shares the computer memory with A_{ij}; when $i = j$, D_{jj} shares the computer memory with A_{jj}; when $i < j$, U_{ij} shares with the computer memory with A_{ij}. Step (a) computes $\lfloor A_j \rfloor \{L_i\}$. Step (b) computes Equation (14), excluding the coefficient U_{jj}. Step (c) computes Equation (31), which is invalid when the denominator is zero. This situation indicates the condition in Equation (13) is not held. By Lemma 3, there are two possibilities for this situation, one is that $\{L_j\}$ and $\lfloor U_j \rfloor$ do not exist, i.e. the corresponding *tailing principal submatrix* is singular, the other is that $[A]$ is singular. If $\{L_j\}$ and $\lfloor U_j \rfloor$ do not exist, we may apply pivoting strategies to rearrange $[A]$ such that the new *tailing principal submatrix* is nonsingular and $\lfloor U_j \rfloor \{A_j\}$ is not zero. If $[A]$ is singular, the inverse of $[A]$ does not exist. Step (d) computes $\lfloor U_i \rfloor \{A_j\}$. Step (e) computes Equation (15), excluding the coefficient L_{jj}. Based upon Algorithm I for computing $[A]^{-1}$, the solution of Equation (4) may be computed as follows.

ALGORITHM II. *A Sequential Procedure For Computing* $\{X\} = [L][D][U]\{B\}$.

(a) For $i = 1 \rightarrow n$ *do*

$$X_i \leftarrow X_i + \sum_{k=i+1}^{n} A_{ik} * X_k;$$

(b) For $i = 1 \rightarrow n$ *do*

$$X_i \leftarrow A_{ii} * X_i;$$

(c) For $i = n \rightarrow 1$ *with step* (-1) *do*

$$X_i \leftarrow X_i + \sum_{k=1}^{i-1} A_{ik} * X_k,$$

in which the solution vector $\{X\}$ shares the computer memory with the coefficient vector $\{B\}$. It can be seen that a complete procedure for solving Equation (4) requires the minimal computer memories, a total of $(n^2 + n)$ in which n^2 is for $[A]$, $[L]$, $[D]$, and $[U]$, and n is for $\{X\}$ and $\{B\}$.

The algorithms also can be easily transformed into parallel procedures with a temporary vector $\{S\}$ of order n. The parallel algorithm for $[A]^{-1} = [L][D][U]$ is as follows.

ALGORITHM III. *A Parallel Procedure for Decomposing* $[A]$ *into* $[A]^{-1} = [L][D][U]$.

For $j = n \rightarrow 1$ *with step* (-1) *do*

(a) For $i = j + 1 \rightarrow n$, *do independently*

$$S_i \leftarrow A_{ji} + \sum_{k=i+1}^{n} A_{jk} * A_{ki};$$

(b) For $i = j + 1 \rightarrow n$ *do independently*

$$A_{ji} \leftarrow -S_i * A_{ii} - \sum_{k=j+1}^{i-1} S_k * A_{kk} * A_{ki};$$

$$(c)\ A_{jj} \leftarrow \frac{1}{A_{jj} + \sum_{k=j+1}^{n} A_{jk} * A_{kj}};$$

(d) For $i = j + 1 \rightarrow n$ *do independently*

$$S_i \leftarrow A_{ij} + \sum_{k=i+1}^{n} A_{ik} * A_{kj};$$

(e) For $i = j + 1 \rightarrow n$ *do independently*

$$A_{ij} \leftarrow -A_{ii} * S_i - \sum_{k=j+1}^{i-1} A_{ik} * A_{kk} * S_k,$$

in which the components in Steps (a), (b), (d), and (e) can be computed concurrently, and the summation in Step (c) may be computed by a parallel inner product. Efficient performances in parallel environments can be expected. Then, the corresponding parallel version of the matrix-vector multiplication for the solution of Equation (4) may be written as the following algorithm.

ALGORITHM IV. *A Parallel Procedure For Computing* $\{X\} = [L][D][U]\{B\}$.

(a) For $i = 1 \rightarrow n$, *do independently*

$$S_i \leftarrow X_i + \sum_{k=i+1}^{n} A_{ik} * X_k;$$

(b) For $i = 1 \rightarrow n$, *do independently*

$$S_i \leftarrow A_{ii} * S_i;$$

(c) For $i = 1 \rightarrow n$, *do independently*

$$X_i \leftarrow S_i + \sum_{k=1}^{i-1} A_{ik} * S_k.$$

If $[A]$ is a banded and sparse matrix, then the bounds of each loop in Algorithms I and III may be modified by the procedures as shown in [2] so as to reduce the number of arithmetic operations.

4. EXAMPLES AND DISCUSSION

An Alliant/FX8 computer, which is a memory-sharing machine and permits certain instructions to be executed concurrently, will be employed to demonstrate the method. A parallel computer code in FORTRAN 77 has been developed for inverting nonsingular matrices, which also can be applied to solve a system of linear equations. The first example is a (3×3) nonsingular matrix as

$$[A] = \begin{bmatrix} 1 & 8 & 7 \\ 2 & 9 & 6 \\ 3 & 4 & 5 \end{bmatrix}, \tag{32}$$

which is then decomposed by the computer code. The output of the first example is

$$\begin{bmatrix} -0.437500 & -0.571429 & -0.714286 \\ 0.380952 & 0.238095 & -1.200000 \\ -0.904762 & -0.800000 & 0.200000 \end{bmatrix}. \tag{33}$$

Equation (33) represents

$$[L] = \begin{bmatrix} 1 & 0 & 0 \\ 0.380952 & 1 & 0 \\ -0.904762 & -0.8 & 1 \end{bmatrix}, \tag{34}$$

$$[D] = \begin{bmatrix} -0.437500 & 0 & 0 \\ 0 & 0.238095 & 0 \\ 0 & 0 & 0.2 \end{bmatrix}, \tag{35}$$

$$[U] = \begin{bmatrix} 1 & -0.571429 & -0.714286 \\ 0 & 1 & -1.200000 \\ 0 & 0 & 1 \end{bmatrix}. \tag{36}$$

It can be verified that $[L][D][U]$ is the inverse of $[A]$. The method shows a special feature for inverting a nonsingular matrix, and is classified as a direct method. It can be counted from Algorithms I–IV that the complexity of the method is $O(n^3)$, which is equivalent to the Gaussian elimination, the conventional direct method. Since Gaussian elimination does not have sufficient degree of data independence, the conventional direct method cannot take full advantage of multiprocessors. The presented method provides an efficient algorithm for parallel computations. An algorithm for distributing the computing streams of this class of decomposition among employed processors had been studied in [3].

In order to demonstrate the performance on an Alliant/FX8 computer, a series of examples in the form of Equation (4) had been tested. The elapsed CPU time including the user time, the system time, and the total time can be collected by the system call ETIME on an Alliant/FX8 computer. Table 1 shows a performance in solving a system of linear equations with a dense, asymmetric, and indefinite matrix of order (256×256) in single precision, in which the time 60.20 seconds spent on one processor can be significantly reduced to 8.05 seconds on 8 processors, giving a speedup of 7.48 and an efficiency of 93.50%. From the 4th column in Table 1, it can be seen that the required time is almost reduced by half when doubling the processors. This performance convinces that the presented method is a paralleliable direct method.

Table 1. Performance in solving a dense, asymmetric and indefinite system of order (256 × 256) in single precision.

Number of Processors	User Time (sec.)	System Time (sec.)	Total Time (sec.)	Speedup	Efficiency (%)
1	60.19	0.01	60.20	1.00	100.00
2	30.15	0.00	30.15	2.00	100.00
4	15.52	0.00	15.52	3.88	97.00
8	8.05	0.00	8.05	7.48	93.50

Tables 2 and 3 also show two sets of timing results in solving systems of linear equations with asymmetric and indefinite matrices in double precision. These performances also show high efficient parallelism, especially in the example shown in Table 3 of order (1024×1024) with an almost perfect speedup. This is consistent with a common fact in parallel computations that a larger scale problem may produce a better performance. Since the examples with respect to Tables 1 and 2 are of the same order, we can conclude that the performances on an Alliant/FX8 computer do not significantly change in different precisions. The characteristics of this class of decomposition can be summarized as follows:

(a) The method is a direct method without the requirement of convergence, which can be applied to some types of problems where the iterative methods, such as Jacobian method, Gauss-Seidel Iteration, and Conjugate Gradient method, are not available;
(b) The decomposed matrices have represented the inverse, the presented method provides a more convenient way of applications;
(c) The method requires the minimal amount of computer memories;
(d) The method provides efficient algorithms for inverting an asymmetric and indefinite matrix.

This new class of decomposition can be a fundamental tool in scientific and engineering computing.

Table 2. Performance in solving a dense, asymmetric and indefinite system of order (256 × 256) in double precision.

Number of Processors	User Time (sec.)	System Time (sec.)	Total Time (sec.)	Speedup	Efficiency (%)
1	65.72	0.00	65.72	1.00	100.00
2	32.96	0.00	32.96	1.99	99.50
4	17.10	0.00	17.10	3.84	96.00
8	8.79	0.00	8.79	7.48	93.50

Table 3. Performance in solving a dense, asymmetrical and indefinite system of order (256 × 256) in double precision.

Number of Processors	User Time (sec.)	System Time (sec.)	Total Time (sec.)	Speedup	Efficiency (%)
1	5164.24	0.00	5164.24	1.00	100.00
2	2584.90	0.00	2584.90	2.00	100.00
4	1303.86	0.00	1303.86	3.96	99.00
8	665.18	0.00	655.18	7.76	97.00

References

1. J.-C. Luo, A new class of decomposition for symmetric systems, *Mechanics Research Communications* **19** (3) (1992).
2. J.-C. Luo, An incomplete inverse as a preconditioner for the conjugate gradient method, *Computers & Mathematics with Applications* (to appear).
3. J.-C. Luo, A note on parallel processing, *Applied Mathematics Letters* **5** (2), 75–78 (1992).