What Is an LU Factorization?

Nicholas J. Higham*

April 20, 2021

An LU factorization of an $n \times n$ matrix A is a factorization $A = LU$, where L is unit lower triangular and U is upper triangular. "Unit" means that L has ones on the diagonal. Example:

$$\begin{bmatrix} 3 & -1 & 1 & 1 \\ -1 & 3 & 1 & -1 \\ -1 & -1 & 3 & 1 \\ 1 & 1 & 1 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -\frac{1}{3} & 1 & 0 & 0 \\ -\frac{1}{3} & -\frac{1}{2} & 1 & 0 \\ \frac{1}{3} & \frac{1}{2} & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 & -1 & 1 & 1 \\ 0 & \frac{8}{3} & \frac{4}{3} & -\frac{2}{3} \\ 0 & 0 & 4 & 1 \\ 0 & 0 & 0 & 3 \end{bmatrix}. \qquad (1)$$

An LU factorization simplifies the solution of many problems associated with linear systems. In particular, solving a linear system $Ax = b$ reduces to solving the triangular systems $Ly = b$ and $Ux = y$, since then $b = L(Ux)$.

For a given A, an LU factorization may or may not exist, and if it does exist it may not be unique. Conditions for existence and uniqueness are given in the following result (see Higham, 2002, Thm. 9.1 for a proof). Denote by $A_k = A(1\colon k, 1\colon k)$ the leading principal submatrix of A of order k.

> **Theorem 1**. *The matrix $A \in \mathbb{R}^{n\times n}$ has a unique LU factorization if and only if A_k is nonsingular for $k = 1\colon n-1$. If A_k is singular for some $1 \le k \le n-1$ then the factorization may exist, but if so it is not unique.*

Note that the (non)singularity of A plays no role in Theorem 1. However, if A is nonsingular and has an LU factorization then the factorization is unique. Indeed if A has LU factorizations $A = L_1U_1 = L_2U_2$ then the U_i are necessarily nonsingular and so $L_2^{-1}L_1 = U_2U_1^{-1}$. The left side of this equation is unit lower triangular and the right side is upper triangular; therefore both sides must equal the identity matrix, which means that $L_1 = L_2$ and $U_1 = U_2$, as required.

Equating leading principal submatrices in $A = LU$ gives $A_k = L_kU_k$, which implies that $\det(A_k) = \det(U_k) = u_{11}u_{22}\dots u_{kk}$. Hence $u_{kk} = \det(A_k)/\det(A_{k-1})$. In fact, such determinantal formulas hold for all the elements of L and U:

$$\ell_{ij} = \frac{\det\bigl(A([1:j-1,\, i], 1:j)\bigr)}{\det(A_j)}, \quad i > j,$$
$$u_{ij} = \frac{\det\bigl(A(1:i, [1:i-1,\, j])\bigr)}{\det(A_{i-1})}, \quad i \le j.$$

Here, $A(u, v)$, where u and v are vectors of subscripts, denotes the submatrix formed from the intersection of the rows indexed by u and the columns indexed by v.

*Department of Mathematics, University of Manchester, Manchester, M13 9PL, UK (nick.higham@manchester.ac.uk).

Relation with Gaussian Elimination

LU factorization is intimately connected with Gaussian elimination. Recall that Gaussian elimination transforms a matrix $A^{(1)} = A \in \mathbb{R}^{n\times n}$ to upper triangular form $U = A^{(n)}$ in $n-1$ stages. At the kth stage, multiples of row k are added to the later rows to eliminate the elements below the diagonal in column k, using the formulas

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik}a_{kj}^{(k)}, \quad i = k+1\colon n,\ j = k+1\colon n,$$

where the quantities $m_{ik} = a_{ik}^{(k)}/a_{kk}^{(k)}$ are the multipliers. Of course each $a_{kk}^{(k)}$ must be nonzero for these formulas to be defined, and this is connected with the conditions of Theorem 1, since $u_{kk} = a_{kk}^{(k)}$. The final U is the upper triangular LU factor, with $u_{ij} = a_{ij}^{(i)}$ for $j \geq i$, and $\ell_{ij} = m_{ij}$ for $i > j$, that is, the multipliers make up the L factor (for a proof of these properties see any textbook on numerical linear algebra).

The matrix factorization viewpoint is well established as a powerful paradigm for thinking and computing. Separating the computation of LU factorization from its application is beneficial. For example, given $A = LU$ we saw above how to solve $Ax = b$. If we need to solve for another right-hand side b_2 we can just solve $Ly_2 = b_2$ and $Ux_2 = y_2$, re-using the LU factorization. Similarly, solving $A^T z = c$ reduces to solving the triangular systems $U^T w = c$ and $L^T z = w$.

Computation

An LU factorization can be computed by directly solving for the components of L and U in the equation $A = LU$. Indeed because L has unit diagonal the first row of U is the same as the first row of A, and $a_{k1} = \ell_{k1}u_{11} = \ell_{k1}a_{11}$ then determines the first column of L. One can go on to determine the kth row of U and the kth row of L, for $k = 2\colon n$. This leads to the Doolittle method, which involves inner products of partial rows of L and partial columns of U.

Given the equivalence between LU factorization and Gaussian elimination we can also employ the Gaussian elimination equations:

```
% kji form of LU factorization.
for k = 1 : n − 1
    for j = k + 1 : n
        for i = k + 1 : n
            a_ij^(k+1) = a_ij^(k) − a_ik^(k) a_kj^(k) / a_kk^(k)
        end
    end
end
```

This kji ordering of the loops in the factorization is the basis of early Fortran implementations of LU factorization, such as that in LINPACK. The inner loop travels down the columns of A, accessing contiguous elements of A since Fortran stores arrays by column. Interchanging the two inner loops gives the kij ordering, which updatesTuesday April 20, 2021 the matrix a row at a time, and is appropriate for a language such as C that stores arrays by row.

The ijk and jik orderings correspond to the Doolittle method. The last two of the $3! = 6$ orderings are the ikj and jki orderings, to which we will return later.

Schur Complements

For $A \in \mathbb{R}^{n\times n}$ with $\alpha = a_{11} \neq 0$ we can write

$$A = \begin{bmatrix} \alpha & b^T \\ c & D \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ c/\alpha & I_{n-1} \end{bmatrix} \begin{bmatrix} \alpha & b^T \\ 0 & D - cb^T/\alpha \end{bmatrix} =: L_1 U_1. \qquad (2)$$

The $(n-1)\times(n-1)$ matrix $S = D - cb^T/\alpha$ is called the *Schur complement* of α in A.

The first row and column of L_1 and U_1 have the correct forms for a unit lower triangular matrix and an upper triangular matrix, respectively. If we can find an LU factorization $S = L_2U_2$ then

$$A = \begin{bmatrix} 1 & 0 \\ c/\alpha & L_2 \end{bmatrix} \begin{bmatrix} \alpha & b^T \\ 0 & U_2 \end{bmatrix}$$

is an LU factorization of A. Note that this is simply another way to express the kji algorithm above.

For several matrix structures it is immediate that $\alpha \neq 0$. If we can show that the Schur complement inherits the same structure then it follows by induction that we can compute the factorization for S, and so an LU factorization of A exists. Classes of matrix for which $a_{11} \neq 0$ and A being in the class implies the Schur complement S is also in the class include

- symmetric positive definite matrices,
- M-matrices,
- matrices (block) diagonally dominant by rows or columns.

(The proofs of these properties are nontrivial.) Note that the matrix (1) is row diagonally dominant, as is its U factor, as must be the case since its rows are contained in Schur complements.

We say that A has *upper bandwidth* q if $a_{ij} = 0$ for $j > i+q$ and *lower bandwidth* p if $a_{ij} = 0$ for $i > j+p$. Another use of (2) is to show that L and U inherit the bandwidths of A.

> **Theorem 2**. *Let $A \in \mathbb{R}^{n\times n}$ have lower bandwidth p and upper bandwidth q. If A has an LU factorization then L has lower bandwidth p and U has upper bandwidth q.*
>
> *Proof.* In (2), the first column of L_1 and the first row of U_1 have the required structure and S has upper bandwidth q and lower bandwidth p, since c and b have only p and q nonzero components, respectively. The result follows by induction.

Block Implementations

In order to achieve high performance on modern computers with their hierarchical memories, LU factorization is implemented in a block form expressed in terms of matrix multiplication and the solution of multiple right-hand side triangular systems. We describe two block forms of LU factorization. First, consider a block form of (2) with block size p, where A_{11} is $p \times p$:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 \\ L_{21} & I_{n-p} \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ 0 & S \end{bmatrix}.$$

Here, S is the Schur complement of A_{11} in A, given by $S = A_{22} - A_{21}A_{11}^{-1}A_{12}$. This leads to the following algorithm:

1. Factor $A_{11} = L_{11}U_{11}$.
2. Solve $L_{11}U_{12} = A_{12}$ for U_{12}.
3. Solve $L_{21}U_{11} = A_{21}$ for L_{21}.
4. Form $S = A_{22} - L_{21}U_{12}$.
5. Repeat steps 1–4 on S to obtain $S = L_{22}U_{22}$.

The factorization on step 1 can be done by any form of LU factorization. This algorithm is known as a *right-looking* algorithm, since it accesses data to the right of the block being worked on (in particular, at each stage lines 2 and 4 access the last few columns of the matrix).

An alternative algorithm can derived by considering a block 3×3 partitioning, in which we assume we have already computed the first block column of L and U:

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & I \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} & \times \\ 0 & U_{22} & \times \\ 0 & 0 & \times \end{bmatrix}.$$

We now compute the middle block column of L and U, comprising p columns, by

1. Solve $L_{11}U_{12} = A_{12}$ for U_{12}.
2. Factor $A_{22} - L_{21}U_{12} = L_{22}U_{22}$.
3. Solve $L_{32}U_{22} = A_{32} - L_{31}U_{12}$ for L_{32}.
4. Repartition so that the first two block columns become a single block column and repeat steps 1–4.

This algorithm corresponds to the jki ordering. Note that the Schur complement is updated only a block column at a time. Because the algorithm accesses data only to the left of the block column being worked on, it is known as a *left-looking* algorithm.

Our description of these block algorithms emphasizes the mathematical ideas. The implementation details, especially for the left-looking algorithm, are not trivial. The optimal choice of block size p will depend on the machine, but p is typically in the range 64–512.

An important point is that all these different forms of LU factorization, no matter which ijk ordering or which value of p, carry out the same operations. The only difference is the order in which the operations are performed (and the order in which the data is accessed). Even the rounding errors are the same for all versions (assuming the use of "plain vanilla" floating-point arithmetic).

Rectangular Matrices

Although it is most commonly used for square matrices, LU factorization is defined for rectangular matrices, too. If $A \in \mathbb{R}^{m\times n}$ then the factorization has the form $A = LU$ with $L \in \mathbb{R}^{m\times m}$ lower triangular and $U \in \mathbb{R}^{m\times n}$ upper trapezoidal. The conditions for existence and uniqueness of an LU factorization of A are the same as those for $A(1\colon p, 1\colon p)$, where $p = \min(m, n)$.

Block LU Factorization

Another form of LU factorization relaxes the structure of L and U from triangular to block triangular, with L having identity matrices on the diagonal:

$$L = \begin{bmatrix} I & & & \\ L_{21} & I & & \\ \vdots & & \ddots & \\ L_{m1} & \dots & L_{m,m-1} & I \end{bmatrix}, \quad U = \begin{bmatrix} U_{11} & U_{12} & \dots & U_{1m} \\ & U_{22} & & \vdots \\ & & \ddots & U_{m-1,m} \\ & & & U_{mm} \end{bmatrix}.$$

Note that U is not, in general, upper triangular.

An example of a block LU factorization is

$$A = \left[\begin{array}{cc|cc} 0 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ \hline -2 & 3 & 4 & 2 \\ -1 & 2 & 1 & 3 \end{array}\right] = \left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \hline 1 & 2 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{array}\right] \left[\begin{array}{cc|cc} 0 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ \hline 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{array}\right].$$

LU factorization fails on A because of the zero $(1,1)$ pivot. This block LU factorization corresponds to using the leading 2×2 principal submatrix of A to eliminate the elements in the $(3\colon 4, 1\colon 2)$ submatrix. In the context of a linear system $Ax = b$, we have effectively solved for the variables x_1 and x_2 in terms of x_3 and x_4 and then substituted for x_1 and x_2 in the last two equations.

Conditions for the existence of a block LU factorization are analogous to, but less stringent than, those for LU factorization in Theorem 1.

> **Theorem 3**. *The matrix $A \in \mathbb{R}^{n\times n}$ has a unique block LU factorization if and only if the first $m-1$ leading principal block submatrices of A are nonsingular.*

The conditions in Theorem 3 can be shown to be satisfied if A is block diagonally dominant by rows or columns.

Note that to solve a linear system $Ax = b$ using a block LU factorization we need to solve $Ly = b$ and $Ux = y$, but the latter system is not triangular and requires the solution of systems $U_{ii}x_i = y_i$ involving the diagonal blocks of U, which would normally be done by (standard) LU factorization.

Sensitivity

If A has a unique LU factorization then for a small enough perturbation ΔA an LU factorization $A + \Delta A = (L + \Delta L)(U + \Delta U)$ exists. To first order, this equation is $\Delta A = \Delta L U + L \Delta U$, which gives

$$L^{-1}\Delta A\, U^{-1} = L^{-1}\Delta L + \Delta U\, U^{-1}.$$

Since ΔL is strictly lower triangular and ΔU is upper triangular, we have, to first order,

$$\Delta L = L\,\mathrm{tril}(L^{-1}\Delta A U^{-1}), \quad \Delta U = \mathrm{triu}(L^{-1}\Delta A U^{-1})U,$$

where tril denotes the strictly lower triangular part and triu the strictly upper triangular part. Clearly, the sensitivity of the LU factors depends on the inverses of L and U. However, in most situations, such as when we are solving a linear system $Ax = b$, it is the backward stability of the LU factors, not their sensitivity, that is relevant.

Pivoting and Numerical Stability

Since not all matrices have an LU factorization, we need the option of applying row and column interchanges to ensure that the pivots are nonzero unless the column in question is already in triangular form.

In finite precision computation it is important that computed LU factors $\widehat{L}$ and $\widehat{U}$ are numerically stable in the sense that $\widehat{L}\widehat{U} = A + \Delta A$ with $\|\Delta A\| \leq c_n u \|A\|$, where c_n is a constant and u is the unit roundoff. For certain matrix properties, such as diagonal dominance by rows or columns, numerical stability is guaranteed, but in general it is necessary to incorporate row interchanges, or row or column interchanges, in order to obtain a stable factorization.

See What Is the Growth Factor for Gaussian Elimination? for details of pivoting strategies and see Randsvd Matrices with Large Growth Factors for some recent research on growth factors.

References

This is a minimal set of references, which contain further useful references within.

- Jack J. Dongarra, Iain S. Duff, Danny C. Sorensen, and Henk A. Van der Vorst, Numerical Linear Algebra for High-Performance Computers, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1998. (For different implementations of LU factorization.)
- Nicholas J. Higham, Accuracy and Stability of Numerical Algorithms, second edition, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002.

Related Blog Posts

- Randsvd Matrices with Large Growth Factors (2020)
- What Is a Block Matrix? (2020)
- What is a Diagonally Dominant Matrix? (2021)
- What Is the Growth Factor for Gaussian Elimination? (2020)

This article is part of the "What Is" series, available from `https://nhigham.com/category/what-is` and in PDF form from the GitHub repository `https://github.com/higham/what-is`.