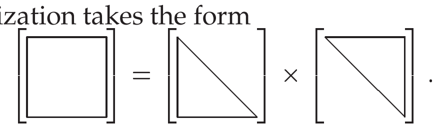
Mathematical Tripos Part IB: Easter 2001

Numerical Analysis – Lecture 1[1]

1 LU factorization of matrices

1.1 Definition and applications

Let A be a real $n \times n$ matrix. We say that the $n \times n$ matrices L and U are an *LU factorization* of A if ***(1)*** L is lower-triangular (i.e., $L_{i,j} = 0$, $i < j$); ***(2)*** U is upper-triangular, $U_{i,j} = 0$, $i > j$; and ***(3)*** $A = LU$. Therefore the factorization takes the form



Application 1 Calculation of a determinant: $\det A = \det L \det U = (\prod_{k=1}^{n} L_{k,k}) \cdot (\prod_{k=1}^{n} U_{k,k})$.

Application 2 Testing nonsingularity: $A = LU$ is nonsingular iff all the diagonal elements of L and U are nonzero.

Application 3 Solution of linear systems: Let $A = LU$ and suppose we wish to solve $A\boldsymbol{x} = \boldsymbol{b}$. This is the same as $L(U\boldsymbol{x}) = \boldsymbol{b}$, which we decompose into $L\boldsymbol{y} = \boldsymbol{b}$, $U\boldsymbol{x} = \boldsymbol{y}$. Both latter systems are triangular and can be calculated easily. Thus, $L_{1,1}y_1 = b_1$ gives y_1, next $L_{2,1}y_1 + L_{2,2}y_2 = b_2$ yields y_2 etc. Having found $\boldsymbol{y}$, we solve for $\boldsymbol{x}$ by reversing the order: $U_{n,n}x_n = y_n$ gives x_n, $U_{n-1,n-1}x_{n-1} + U_{n-1,n}x_n = y_{n-1}$ produces x_{n-1} and so on. This requires $\mathcal{O}(n^2)$ computational operations (usually we only bother to count multiplications/divisions).

Application 4 The inverse of A: It is straightforward to devise a direct way of calculating the inverse of triangular matrices, subsequently forming $A^{-1} = U^{-1}L^{-1}$.

1.2 The calculation of LU factorization

We denote the *columns* of L by $\boldsymbol{l}_1, \boldsymbol{l}_2, \ldots, \boldsymbol{l}_n$ and the *rows* of U by $\boldsymbol{u}_1^{\mathrm{T}}, \boldsymbol{u}_2^{\mathrm{T}}, \ldots, \boldsymbol{u}_n^{\mathrm{T}}$. Hence

$$A = LU = [\ \boldsymbol{l}_1 \quad \boldsymbol{l}_2 \quad \cdots \quad \boldsymbol{l}_n\ ] \begin{bmatrix} \boldsymbol{u}_1^{\mathrm{T}} \\ \boldsymbol{u}_2^{\mathrm{T}} \\ \vdots \\ \boldsymbol{u}_n^{\mathrm{T}} \end{bmatrix} = \sum_{k=1}^{n} \boldsymbol{l}_k \boldsymbol{u}_k^{\mathrm{T}}. \tag{1.1}$$

Since the first $k-1$ components of $\boldsymbol{l}_k$ and $\boldsymbol{u}_k$ are all zero, each rank-one matrix $\boldsymbol{l}_k\boldsymbol{u}_k^{\mathrm{T}}$ has zeros in its first k rows and columns.

Assume that the factorization exists (hence the diagonal elements of L are nonzero) and that A is nonsingular. Since $\boldsymbol{l}_k\boldsymbol{u}_k^{\mathrm{T}}$ stays the same if we replace $\boldsymbol{l}_k \to \alpha\boldsymbol{l}_k$, $\boldsymbol{u}_k \to \alpha^{-1}\boldsymbol{u}_k$, where $\alpha \neq 0$, we may assume w.l.o.g. that all diagonal elements of L equal one. In other words, the kth row of $\boldsymbol{l}_k\boldsymbol{u}_k^{\mathrm{T}}$ is $\boldsymbol{u}_k^{\mathrm{T}}$ and its kth column is $U_{k,k}$ times $\boldsymbol{l}_k$.

We begin our calculation by extracting $\boldsymbol{l}_1$ and $\boldsymbol{u}_1^{\mathrm{T}}$ from A, and then proceed similarly to extract $\boldsymbol{l}_2$ and $\boldsymbol{u}_2^{\mathrm{T}}$, etc.

First we note that since the leading $k-1$ elements of $\boldsymbol{l}_k$ and $\boldsymbol{u}_k$ are zero for $k \geq 2$, it follows from (1.1) that $\boldsymbol{u}_1^{\mathrm{T}}$ is the first row of A and $\boldsymbol{l}_1$ is the first column of A, divided by $A_{1,1}$ (so that $L_{1,1} = 1$).

[1] Please email all corrections and suggestions to these notes to `a.shadrin@damtp.cam.ac.uk`. All handouts are available on the WWW at the URL `http://www.damtp.cam.ac.uk/user/na/PartIB/Handouts.html`.

Next, having found $\boldsymbol{l}_1$ and $\boldsymbol{u}_1$, we form the matrix $A - \boldsymbol{l}_1\boldsymbol{u}_1^{\mathrm{T}} = \sum_{k=2}^{n} \boldsymbol{l}_k\boldsymbol{u}_k^{\mathrm{T}}$. The first row & column of A are zero and it follows that $\boldsymbol{u}_2^{\mathrm{T}}$ is the second row of $A - \boldsymbol{l}_1\boldsymbol{u}_1^{\mathrm{T}}$, while $\boldsymbol{l}_2$ is its second column, scaled so that $L_{2,2} = 1$.

The LU algorithm: Set $A_0 := A$. For all $k = 1, 2, \ldots, n$ set $\boldsymbol{u}_k^{\mathrm{T}}$ to the kth row of A_{k-1} and $\boldsymbol{l}_k$ to the kth column of A_{k-1}, scaled so that $L_{k,k} = 1$. Further, calculate $A_k := A_{k-1} - \boldsymbol{l}_k\boldsymbol{u}_k^{\mathrm{T}}$ before incrementing k.

Note that all elements in the first k rows & columns of A_k are zero. Hence, we can use the storage of the original A to accumulate L and U. The full LU factorization requires $\mathcal{O}(n^3)$ computational operations.

1.3 Relation to Gaussian elimination

The equation $A_k = A_{k-1} - \boldsymbol{l}_k\boldsymbol{u}_k^{\mathrm{T}}$ has the property that the jth row of A_k is the jth row of A_{k-1} minus $L_{j,k}$ times $\boldsymbol{u}_k^{\mathrm{T}}$ (the kth row of A_{k-1}). Moreover, the multipliers $L_{k,k}, L_{k+1,k}, \ldots, L_{n,k}$ are chosen so that the outcome of this *elementary row operation* is that the kth column of A_k is zero. This construction is analogous to Gaussian elimination for solving $A\boldsymbol{x} = \boldsymbol{b}$. An important difference is that in LU we do not consider the right hand side $\boldsymbol{b}$ until the factorization is complete. This is useful e.g. when there are many right hand sides, in particular if not all the $\boldsymbol{b}$'s are known at the outset: in Gaussian elimination the solution for each new $\boldsymbol{b}$ would require $\mathcal{O}(n^3)$ computational operations, whereas with LU factorization $\mathcal{O}(n^3)$ operations are required for the initial factorization, but then the solution for each new $\boldsymbol{b}$ only requires $\mathcal{O}(n^2)$ operations.

1.4 Pivoting

Naive LU factorization fails when, for example, $A_{1,1} = 0$. The remedy is to exchange rows of A, a technique called *column pivoting*. This is equivalent to picking a suitable equation for eliminating the first unknown in Gaussian elimination. Specifically, column pivoting means that, having obtained A_{k-1}, we exchange two rows of A_{k-1} so that the element of largest magnitude in the kth column is in the 'pivotal position' (k, k). In other words,

$$|(A_{k-1})_{k,k}| = \max\{|(A_{k-1})_{j,k}| \,:\, j = 1, 2, \ldots, n\}.$$

Of course, the same exchange is required in the portion of L that has been formed already (i.e., the first $k-1$ columns). Also, we need to record the permutation of rows to solve for the right hand side and/or to compute the determinant. (The exchange of rows can be regarded as the pre-multiplication of the relevant matrix by a permutation matrix.)

Column pivoting copes with zeros at the pivot position, except when the whole kth column of A_{k-1} is zero – in that case it is usual to let $\boldsymbol{l}_k$ be the kth unit vector while, as before, choose $\boldsymbol{u}_k^{\mathrm{T}}$ as the kth row of A_k). Such a choice preserves the condition that the matrix $\boldsymbol{l}_k\boldsymbol{u}_k^{\mathrm{T}}$ has the same kth row and column as A_{k-1}. Thus $A_k := A_{k-1} - \boldsymbol{l}_k\boldsymbol{u}_k^{\mathrm{T}}$ still has zeros in its kth row and column as required.

An important advantage of column pivoting is that every element of L has magnitude at most one. This avoids not just division by zero but also tends to reduce the chance of very large numbers occuring during the factorization, a phenomenon that might lead to*ill conditioning* and to accumulation of *roundoff error.*

In *row pivoting* one exchanges columns of A_{k-1}, rather than rows (sic!), whereas *total pivoting* corresponds to exchange of both rows and columns, so that the modulus of the pivotal element $(A_{k-1})_{k,k}$ is maximised.