LU Factorizations

Suppose that a matrix $A$ has a factorization

$$A = LU$$

where $L$ and $U$ are respectively lower triangular and upper triangular matrices. Then the linear system (10.1)

$$Ax = b$$

can be solved quite easily. For we can rewrite (10.1) as

$$L(Ux) = b$$

But then $Ux$ must be the solution of

$$Lz = b$$

(which we can solve explicitly for $z$ since $L$ is lower triangular), and so

$$Ux = z$$

(which we can solve explicitly for $x$ since $U$ is upper triangular).

We shall now develop an algorithm for finding a $LU$ factorization of a given matrix $A$. Not all matrices will have $LU$ factorizations. However, it will turn out that the lack of an $LU$ factorization can be attributed to the fact that the application of the algorithm given below is eventually nullified by an illegal operation; namely a division by 0. In other words, if a matrix has an $LU$ factorization our algorithm will find it. If not, our algorithm will instead churn out a divide by zero error.

To develop this algorithm we start with the matrix multiplication formula

$$(10.2) \quad a_{ij} = \sum_{k=1}^{n} l_{ik}u_{kj}$$

Now the components of $L$, being lower triangular, satisfy

$$l_{ik} = 0 \quad \text{if} \quad k > i$$

and the components of $U$, being upper triangular, satisfy

$$u_{kj} = 0 \quad \text{if} \quad k > j$$

Therefore the sum (10.2) can be written

$$(10.3) \quad a_{ij} = \sum_{k=1}^{\min(i,j)} l_{ik}u_{kj}$$

Now before we actually try solving equations (10.3) for the components $l_{ik}$ of $L$ and the components $u_{kj}$ of $U$, let’s first observe that the matrix $A$ has $n^2$ components $a_{ij}$; so we have a total of $n^2$ equations in (10.3). Now the number of (possibly) non-zero components of the matrix $L$ is determined by observing that
it has the same number, \( n \), of diagonal components as matrix \( A \), but only half the number of off diagonal components as \( A \). Therefore, the number of nontrivial entries of \( L \) is

\[
n + \frac{1}{2}(n^2 - n) = \frac{n(n + 1)}{2}
\]

Similarly, the upper triangular matrix \( U \) has \( \frac{1}{2}n(n + 1) \) components. Thus the total number of unknowns in the system (10.3) is

\[
\frac{n(n + 1)}{2} + \frac{n(n + 1)}{2} = n^2 + n
\]

We thus have \( n \) more unknowns than we have equations. The general solution of (10.3) will thus contain \( n \) free parameters. We can remove these extra degrees of freedom, without destroying the possibility of finding a solution, by imposing \( n \) additional conditions on the components of \( L \) and/or \( U \). We shall do so in a manner that simplifies the subsequent solution of (10.3); namely we shall require

\[
(10.4) \quad l_{ii} = 1, \quad i = 1, 2, \ldots, n
\]

In other words, we force the lower diagonal matrix \( L \) to have only 1’s along its diagonal. (The text refers to such a matrix as \textbf{unit lower triangular}.)

Let’s now separate the \( n^2 \) equations (10.3) into three subsets corresponding to the cases when \( i = j \), \( i < j \), and \( i > j \).

\[
(10.5) \quad a_{ii} = \sum_{k=1}^{i} l_{ik}u_{ki}, \quad i = 1, 2, \ldots, n
\]

\[
(10.6) \quad a_{ij} = \sum_{k=1}^{i} l_{ik}u_{kj}, \quad i < j
\]

\[
(10.7) \quad a_{ij} = \sum_{k=1}^{j} l_{ik}u_{kj}, \quad j < i
\]

Setting \( i = 1 \) in (10.4) and (10.5) we have

\[
(10.8) \quad a_{11} = l_{11}u_{11} = u_{11} \quad \Rightarrow \quad u_{11} = a_{11}
\]

Setting \( i = 1 \) in (10.6) yields

\[
(10.9) \quad a_{1j} = \sum_{k=1}^{1} l_{ik}u_{kj} = l_{11}u_{1j} = u_{1j} \quad \Rightarrow \quad u_{1j} = a_{1j}, \quad j = 2, 3, \ldots, n
\]

Note that equations (10.8) and (10.9) completely determine the first row of \( U \).

Setting \( j = 1 \) in (10.7) we obtain

\[
(10.10) \quad a_{i1} = \sum_{k=1}^{1} l_{ik}u_{k1} = l_{i1}u_{11} \quad \Rightarrow \quad l_{i1} = \frac{1}{u_{11}}a_{i1} = \frac{a_{i1}}{a_{11}}, \quad i = 2, 3, \ldots, n
\]

Since \( l_{11} = 1 \) by hypothesis, equations (10.10) fix all the elements of the first column of the matrix \( L \).

Let’s now set \( i = 2 \) in (10.5). This yields

\[
(10.11) \quad a_{22} = \sum_{k=1}^{2} l_{2k}u_{k2} = l_{21}u_{12} + l_{11}u_{22} = l_{21}u_{12} + u_{22} \quad \Rightarrow \quad u_{22} = a_{22} - l_{21}u_{12}
\]
Since \( l_{21} \) (an element of the first column of \( L \)) and \( u_{12} \) (an element of the first row of \( U \)) have already been determined (10.11) determines \( u_{22} \). Setting \( i = 2 \) in (10.6) yields

\[
(10.12) \quad a_{2j} = \sum_{k=1}^{2} l_{2k} u_{kj} = l_{21} u_{1j} + l_{22} u_{2j} = l_{21} u_{1j} + u_{oj} \quad \Rightarrow \quad u_{oj} = a_{2j} - l_{21} u_{1j} \quad , \quad j = 3, 4, \ldots, n
\]

Since \( u_{21} \equiv 0 \), equations (10.11) and (10.12) completely fix the second row of \( U \).

Now set \( j = 2 \) in (10.7). This yields

\[
(10.13) \quad a_{i2} = \sum_{k=1}^{2} l_{ik} u_{k2} = l_{i1} u_{12} + l_{i2} u_{22} \quad \Rightarrow \quad l_{i2} = \frac{1}{u_{22}} (a_{i2} - l_{i1} u_{12}) \quad , \quad i = 3, 4, \ldots, n
\]

Since \( l_{12} \equiv 0 \), \( l_{22} \equiv 1 \), and because \( u_{22}, l_{11}, \) and \( u_{12} \) have all been previously determined, these relations suffice to fix the second column of \( L \).

Let me now review the steps taken so far so that we can bring to life the general algorithm.

1. We set all the diagonal elements of \( L \) equal to 1.
2. We determined \( u_{11} \) from equation (10.5) with \( i = 1 \).
3. We determined the first row of \( U \) from the preceding results and the equations (10.6) with \( i = 1 \) and \( j = 2, 3, \ldots, n \).
4. We determined the first column of \( L \) from the preceding results and the equations (10.7) with \( j = 1 \) and \( i = 2, 3, \ldots, n \).
5. We determined \( u_{22} \) from the preceding results and equation (10.5) with \( i = 2 \).
6. We determined the second row of \( U \) from the preceding results and the equations (10.6) with \( i = 2 \) and \( j = 3, 4, \ldots, n \).
7. We determined the second column of \( L \) from the preceding results and the equations (10.7) with \( j = 2 \) and \( i = 3, 4, \ldots, n \).

The general algorithm can now be stated.

For each \( k \) from 1 to \( n \)

1. Set \( l_{ik} = 0 \) for \( i = 1, 2, \ldots, k - 1 \) (so that \( L \) is lower triangular).
2. Set \( u_{ik} = 0 \) for \( i = 1, 2, \ldots, k - 1 \) (so that \( U \) is upper triangular).
3. Set \( l_{kk} = 1 \).
4. Determine \( u_{kk} \) from the equation (10.5)

\[
a_{kk} = \sum_{s=1}^{k} l_{ks} u_{sk} \quad \Rightarrow \quad u_{kk} = \frac{1}{l_{kk}} \left( a_{kk} - \sum_{s=1}^{k-1} l_{ks} u_{sk} \right)
\]

Note that the expression on the far right involves only the first \( k - 1 \) columns of \( L \) and the first \( k - 1 \) rows of \( U \).
5. Determine the remaining elements of the \( k^{th} \) row of \( U \) from the equations (10.6) with \( i = k \) and \( j = k + 1, k + 2, \ldots, n \)

\[
a_{kj} = \sum_{s=1}^{k} l_{ks} u_{sj} \quad \Rightarrow \quad u_{kj} = \frac{1}{l_{kk}} \left( a_{kj} - \sum_{s=1}^{k-1} l_{ks} u_{sj} \right)
\]
6. Determine the remaining elements of the \( k^{th} \) column of \( L \) from the equations (10.7) with \( j = k \) and \( i = k + 1, k + 2, \ldots, n \)

\[
a_{ik} = \sum_{s=1}^{k} l_{is} u_{sk} \quad \Rightarrow \quad l_{ik} = \frac{1}{u_{kk}} \left( a_{ik} - \sum_{s=1}^{k-1} l_{is} u_{sk} \right)
\]
Example 10.1. Consider the following $3 \times 3$ matrix:

$$
A = \begin{pmatrix}
5 & 6 & 7 \\
10 & 20 & 23 \\
15 & 50 & 67
\end{pmatrix}
$$

Write a Maple program that carries out an LU factorization of $A$.

The following code works.

```maple
n := 3; # all matrices are nxn=3x3
A := array(1..n,1..n);
L := array(1..n,1..n);
U := array(1..n,1..n);
A := [[5,6,7],[10,20,23],[15,50,67]];

for k from 1 to n do  # calculate kth column of L and kth row of U
  for s from 1 to k-1 do
    L[s,k] := 0; # so that L is lower triangular
    U[k,s] := 0; # so that U is upper triangular
  od;
  L[k,k] := 1;      # by convention
  k1 := k-1;
  # calculate the kth element of kth row of U
  U[k,k] := A[k,k] - sum(L[k,j0]*U[j0,k],j0=1..k1);
  for t from k+1 to n do
    # calculate remaining elements in kth column of L
    L[t,k] := (A[t,k] - sum(L[t,j1]*U[j1,k],j1=1..k1))/U[k,k];
    # calculate remaining elements in kth row of U
    U[k,t] := A[k,t] - sum(L[k,j2]*U[j2,t],j2=1..k1);
  od;
  print(L);
  print(U);
```