LU Factorisation and the solution of linear systems of equations

LU factorization or *decomposition* involves a method for writing a matrix as the product of a lower-triangular matrix (denoted L) and an upper-triangular matrix (U)[1]. LU factorization is useful in solving problems such as linear systems of equations or matrix-vector systems[2], generally of the form

$$A\,\underline{x} = \underline{b}\,,$$

where A is a matrix and $\underline{x}$ and $\underline{b}$ are vectors. Through the determination of the LU factors of A, we can write $A = LU$, and hence that

$$LU\,\underline{x} = \underline{b}\,.$$

The solutions of the latter equation can now be realised in two stages, both of which are a straightforward sequence of substitutions. Firstly a solution '$\underline{y}$' is found to the system $L\,\underline{y} = \underline{b}$ and then the sought solution $\underline{x}$ is found from solving $U\underline{x} = \underline{y}$.

The method of solving systems of the form $A\,\underline{x} = \underline{b}$ through LU factorisation has parallels with the alternative method of Gaussian elimination[3]. From the point of view of a direct computational comparison, there is little to choose between the two methods. However LU factorisation has a significant strategic advantage in that the bulk of the computation occurs in the determination of the factors L and U and hence, once they are determined, a range of vectors $\underline{b}$ can be applied, whereas in Gaussian eliminations the $\underline{b}$ (or set of $\underline{b}$ s) is specified before the processing takes place.

In most cases LU factorisation is applied to square matrices and this will be the focus in this document. An Excel spreadsheet[4] demonstrating LU factorisation and back and forward substitution has been developed along with a guide to using the spreadsheet. The spreadsheet illustrates the methods on 3×3, 5×5 and 10×10 systems. The spreadsheet includes two subroutines in the visual basic (for applications) language: LUfac.bas[5] for finding the LU factors of a matrix and LUbfsb.bas[6] for carrying out the back and forward substitution.

LU factorisation of a 2x2 Matrix

In order to understand, illustrate the method, let is consider applying it to an example of a 2×2 system:

$$\begin{pmatrix} 2 & 1 \\ 3 & 2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 7 \\ 12 \end{pmatrix}$$

[1] Matrix Definitions
[2] Linear systems and 2x2 matrices
[3] Gaussian Elimination
[4] LU.xlsm
[5] LUfac.bas
[6] LUfbsub.bas

The matrix in this system is $A = \begin{pmatrix} 2 & 1 \\ 3 & 2 \end{pmatrix}$ or $\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = \begin{pmatrix} 2 & 1 \\ 3 & 2 \end{pmatrix}$.

The LU factorisation requires that the matrix A is written as the product of a lower and an upper triangular matrix;

$$\begin{pmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}.$$

Matrix multiplication[7] is then applied in order to derive equations relating the values of $l_{11}, l_{21}, l_{22}, u_{11}, u_{12}$ and u_{22} to $a_{11}, a_{12}, a_{21}$and a_{22}:

$$l_{11}u_{11} = a_{11}\,,$$

$$l_{11}u_{12} = a_{12}\,,$$

$$l_{21}u_{11} = a_{21}\,,$$

$$l_{21}u_{12} + l_{22}u_{22} = a_{22}\,,$$

For a 2×2 matrix, there are four equations and six unknowns. In general there are '*n*' more unknowns than there are equations in the general LU factorisation, where n is the dimension of the matrix. In order to secure a unique solution, therefore, *n* values are pre-determined and it is the convention that the diagonals or the lower-triangular matrix L are all set to a value of 1. Hence for a 2×2 system

$$\begin{pmatrix} 1 & 0 \\ l_{21} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

and

$$u_{11} = a_{11}\,,$$

$$u_{12} = a_{12}\,,$$

$$l_{21}u_{11} = a_{21}\,,$$

$$l_{21}u_{12} + u_{22} = a_{22}\,,$$

For the example

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = \begin{pmatrix} 2 & 1 \\ 3 & 2 \end{pmatrix},$$

the equations above give $u_{11} = 2$ and $u_{12} = 1$ directly and then $l_{21} = 1.5$ and $u_{22} = 0.5$ by substitution. This gives the following LU factorisation

[7] Matrix Arithmetic

$$\begin{pmatrix} 1 & 0 \\ 1.5 & 1 \end{pmatrix}\begin{pmatrix} 2 & 1 \\ 0 & 0.5 \end{pmatrix} = \begin{pmatrix} 2 & 1 \\ 3 & 2 \end{pmatrix}.$$

Returning to the equation that we set out to solve:

$$\begin{pmatrix} 2 & 1 \\ 3 & 2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 7 \\ 12 \end{pmatrix},$$

this may be written in the following form

$$\begin{pmatrix} 1 & 0 \\ 1.5 & 1 \end{pmatrix}\begin{pmatrix} 2 & 1 \\ 0 & 0.5 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 7 \\ 12 \end{pmatrix},$$

by replacing *A* by its *LU* factors.

This latter equation can be solved in two steps, using the method of *forward substitution*, followed by *back substitution.*

Forward Substitution

Forward substitution is applied in order to solve

$$\begin{pmatrix} 1 & 0 \\ 1.5 & 1 \end{pmatrix}\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} 7 \\ 12 \end{pmatrix}.$$

In forward substitution the elements of the unknown vector are found in sequence by following the matrix equation row by row.

The top row of the matrix multiplied by the vector $\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$ is as follows:

$$1\,y_1 + 0\,y_2 = 7, \text{hence } y_1 = 7.$$

Similarly the second row gives the following:

$$1.5\,y_1 + 1\,y_2 = 12, \text{hence } y_2 = 1.5, \text{after the } foward\ substition \text{ of } '7' \text{for } y_1.$$

Backsubstitution

Backsubstitution is applied in order to solve the remaining problem

$$\begin{pmatrix} 2 & 1 \\ 0 & 0.5 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 7 \\ 1.5 \end{pmatrix}.$$

In backward substitution the elements of the unknown vector are found in reverse order:

$$0\,x_1 + 0.5\,x_2 = 1.5, \text{hence } x_2 = 3$$

$$2\,x_1 + 1\,x_2 = 7, \text{hence } x_1 = 2, \text{after the } backsubstition \text{ of } '3' \text{for } x_2.$$

LU factorisation of a 3x3 Matrix

In order to understand, illustrate and refine the method, let is consider applying it to an example of a 3×3 system:

$$\begin{pmatrix} 1 & 2 & 2 \\ 1 & 0 & 1 \\ 1 & 2 & 1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 5 \\ 1 \\ 3 \end{pmatrix}$$

The matrix in this system is $A = \begin{pmatrix} 1 & 2 & 2 \\ 1 & 0 & 1 \\ 1 & 2 & 1 \end{pmatrix}$ or $\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & 2 & 2 \\ 1 & 0 & 1 \\ 1 & 2 & 1 \end{pmatrix}$.

Let us return to the LU factorisation of a general 3x3 matrix:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix}\begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$

At the beginning of the process the unknown elements of L and U are greyed-out in the equation above. The LU factorisation method must determine these values.

There are nine unknowns and nine equations:

$$\begin{pmatrix} 1 \times u_{11} = a_{11} & 1 \times u_{12} = a_{12} & 1 \times u_{13} = a_{13} \\ l_{21} \times u_{11} = a_{21} & l_{21} \times u_{12} + u_{22} = a_{22} & l_{21} \times u_{13} + u_{23} = a_{23} \\ l_{31} \times u_{11} = a_{31} & l_{31} \times u_{12} + l_{32} \times u_{22} = a_{32} & l_{31} \times u_{13} + l_{32} \times u_{23} + u_{33} = a_{33} \end{pmatrix}$$

We first note that the determination of the top row of U is straightforward.

<u>U – row 1</u>

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix}\begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$

In the equation above the blue elements are the ones that are evaluated at this stage and the yellow elements are the elements that are used. This leads to the following assignments.

$$\begin{aligned} u_{11} &= a_{11} \\ u_{12} &= a_{12} \\ u_{13} &= a_{13} \end{aligned}$$

L - column 1

Once u_{11} is evaluated, l_{21} and l_{31} can be evaluated straightforwardly

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$

Giving the equations:

$$l_{21} \times u_{11} = a_{21},$$
$$l_{31} \times u_{11} = a_{31}.$$

That is

$$l_{21} = a_{21} / u_{11},$$
$$l_{31} = a_{31} / u_{11}.$$

U – row 2

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$

$$u_{22} = a_{22} - l_{21} \times u_{12}$$

$$u_{23} = a_{23} - l_{21} \times u_{13}$$

L - column 2

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$

$$l_{32} = \frac{a_{32} - l_{31} \times u_{12}}{u_{22}}.$$

U - row 3

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$

$$u_{33} = a_{33} - (l_{31} \times u_{13} + l_{32} \times u_{23})$$

Example

For the matrix in the example, $\begin{pmatrix}1 & 2 & 2\\ 1 & 0 & 1\\ 1 & 2 & 1\end{pmatrix}$, the application of the method outlined above gives the following results.

U – row 1

$$u_{11} = a_{11} = 1$$
$$u_{12} = a_{12} = 2$$
$$u_{13} = a_{13} = 2$$

L - column 1

$$l_{21} = a_{21} / u_{11} = \frac{1}{1} = 1,$$
$$l_{31} = a_{31} / u_{11} = \frac{1}{1} = 1\,.$$

U – row 2

$$u_{22} = a_{22} - l_{21} \times u_{12} = 0 - 1\times 2 = -2$$

$$u_{23} = a_{23} - l_{21} \times u_{13} = 1 - 1\times 2 = -1$$

L - column 2

$$l_{32} = \frac{a_{32} - l_{31} \times u_{12}}{u_{22}} = \frac{2 - 1 \times 2}{-2} = 0.$$

U - row 3

$$u_{33} = a_{33} - (l_{31} \times u_{13} + l_{32} \times u_{23}) = 1 - (1\times 2 + 0\times(-1)) = -1$$

Hence the original matrix can be factorised as follows:

$$\begin{pmatrix}1 & 0 & 0\\ 1 & 1 & 0\\ 1 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 2 & 2\\ 0 & -2 & -1\\ 0 & 0 & -1\end{pmatrix} = \begin{pmatrix}1 & 2 & 2\\ 1 & 0 & 1\\ 1 & 2 & 1\end{pmatrix}$$

Substituting the LU factorisation into the original matrix-vector problem gives

$$\begin{pmatrix}1 & 0 & 0\\ 1 & 1 & 0\\ 1 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 2 & 2\\ 0 & -2 & -1\\ 0 & 0 & -1\end{pmatrix}\begin{pmatrix}x_1\\ x_2\\ x_3\end{pmatrix} = \begin{pmatrix}5\\ 1\\ 3\end{pmatrix}.$$

As with the 2x2 example, forward substitution is used to solve

$$\begin{pmatrix}1 & 0 & 0\\ 1 & 1 & 0\\ 1 & 0 & 1\end{pmatrix}\begin{pmatrix}y_1\\ y_2\\ y_3\end{pmatrix} = \begin{pmatrix}5\\ 1\\ 3\end{pmatrix}.$$

giving $y_1 = 5, y_2 = -4, y_3 = -2$. Backsubstitution is then used to solve

$$\begin{pmatrix} 1 & 2 & 2 \\ 0 & -2 & -1 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 5 \\ -4 \\ -2 \end{pmatrix}.$$

yielding the solution $x_3 = 2, x_2 = 1, x_1 = -1$,

The LU factorisation algorithm

The LU factorisation method described above evaluates the elements of the *L* and *U* matrices by first calculating the first row of the *U* matrix, then the first sub-diagonal row of the *L* matrix, then the second row's diagonal and super-diagonal element of the *U* matrix and then the second row of sub-diagonal elements of the *U* matrix.

Each of the expressions of l_{ij} and u_{ij} have only the corresponding element of the matrix *A* (a_{ij}), the remaining terms are evaluated elements of *L* and *U*. This is a useful practical property of *LU* factorisation as a computational method; the *L* and *U* matrix elements can successively overwrite the original matrix *A*, *L* forming its lower triangle (excluding the diagonal, since the diagonal values of *L* are predefined) and *U* forming its upper triangle (including the diagonal). The over-writing of the matrix *A* by the elements of *L* and *U* saves computer memory and is standard within most implementations of the LU factorisation algorithm.

For example in the LU factorisation of $A = \begin{pmatrix} 1 & 2 & 2 \\ 1 & 0 & 1 \\ 1 & 2 & 1 \end{pmatrix}$, the L and U matrices $\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$ and $\begin{pmatrix} 1 & 2 & 2 \\ 0 & -2 & -1 \\ 0 & 0 & -1 \end{pmatrix}$ can be stored separately and separate to the matrix *A*. However, the upper triangular elements of L and the lower triangular elements of U are pre-defined and hence it wastes computer memory to store them as two separate matrices and they will be normally be stored as $\begin{pmatrix} 1 & 2 & 2 \\ 1 & -2 & -1 \\ 1 & 0 & -1 \end{pmatrix}$, and again to save computer memory, that this matrix would over-write the original matrix *A.*

The LU factorisation method discussed so far can be written in the form of the following pseudocode, which is also known as *Crout's Algorithm*. The following algorithm is implemented in visual basic with the subrouting LUfac.bas[8] within the module LUfac_module in the LU.xlsm[9] Excel spreadsheet.

[8] LUFac.bas
[9] LU.xlsm

The LU factorisation method; Crout's Algorithm

```
' The main loop is a counter effectively down the diagonal
for i=1 to n
' The first inner loop counts j=i..n, so that a(i,j) addresses the diagonal and super-
'  diagonal elements of a on row i. That is ith row of the U matrix is formed.
 for j=i to n
  sum=0
   for k=1 to i-1
     sum=sum+a(i,k)*a(k,j)
   a(i,j)=a(i,j)-sum
' The second  inner loop counts j=i..n, so that a(j,i) addresses the sub-
'  diagonal elements of A on column i. That is the ith column of the L matrix is formed.
    for j=i+1 to n
     sum=0
      for k=1 to i-1
        sum=sum+a(j,k)*a(k,i)
      a(i,j)=(a(i,j)-sum)/a(i,i)
```

The LU factorisation algorithm involves $2n^3/3$ flops and hence is $O(n^3)$ [10].

However, in the terms of a robust numerical algorithm, the method is not complete. We have not dealt with the possibility of a singular matrix A. Also the last line of the algorithm contains a division by a(i,i), which could take the value of zero (or close to zero) even when the original matrix is non-singular (and well-conditioned), causing the algorithm to fail. The algorithm requires improvement, if it is to be useful as a general purpose method for solving systems of equations, and, as with Gaussian elimination a 'pivoting' method is normally included, and this is considered in the next section.

Pivoting

Pivoting is included in the LU factorisation method so that division by zero or a 'small' number is avoided. The LU factorisation algorithm can be viewed as passing through the diagonal elements and setting the row of U followed by the column on L in turn. The worry with the LU factorisation method above is in the line a(i,j)=(a(i,j)-sum)/a(i,i); if a(i,i) is zero or 'small' then the method will fail or will not sustain numerical confidence.

For example the following matrix is non-singular, but it would fail very early into the application of the method outlined above because of the zero in the first row and first column:

$$\begin{pmatrix} 0 & 4 & -3 \\ 1 & 2 & -1 \\ -2 & 0 & 1 \end{pmatrix}$$

The inclusion of (partial) pivoting involves exchanging rows. The exchange of rows has to be recorded and this is equivalent to maintaining a permutation matrix[11] 'P'. Whereas the LU factorisation (without pivoting) of a matrix is unique (if it exists), the

[10] Big O notation in computing
[11] Permutation Matrix

inclusion of pivoting has the result of the outcome of the factorisation method depending on the pivoting method.

In (partial) pivoting method, the divisor from each row is calculated. Exchanging the row with the largest pivot for the superdiagonal row that is to eliminated is the most straightforward form of pivoting. However, in the case when one row is made up of a set of element that are all smaller in magnitude then this may cause an unnecessary and actually counter-productive exchange of rows. For example with the matrix

$$\begin{pmatrix} 0.2 & 0.1 \\ 1 & 2 \end{pmatrix}$$

the initial temptation would be to exchange rows since 1>>0.2, but this would be unnecessary and on balance unhelpful. Hence it is proposed that the row-norm is also taken into account. The following pseudo-code relates such a pivoting method.

The Pivoting Method

```
'  Pass down from the diagonal to check which row would have the
'   largest divisor.
 uiicolmax = 0#
 for j = i To n
   sum = 0
   for k = 1 To i - 1
     sum = sum + a(j, k) * a(k, i)
   reldivisor = (Abs(a(j, i) - sum)) / rownorms(perm(i))
   if (reldivisor > uiicolmax) Then
     uiicolmax = reldivisor
     imax = j

' If the largest divisor is zero or very small then the matrix is
'  singular or ill-conditioned and the factorisation is abandoned
 If (uiicolmax < Tiny) Then
   MsgBox ("Matrix is Singular or very Ill-conditioned. LU factorisation is abandoned")
   Exit Sub

' The rows are swapped and the exchange is recorded in perm
 if (i <> imax) Then
   for k = 1 to n
     dum = a(imax, k)
     a(imax, k) = a(i, k)
     a(i, k) = dum
   permi = perm(i)
   perm(i) = perm(imax)
   perm(imax) = permi
```

Through applying a (partial) pivoting method within the LU factorisation algorithm, the result is generalised so that the matrix A is factorised as follows;

$$PLU = A\,,$$

with a permutation matrix P included.

The pivoting method above is implemented in visual basic with the subroutine LUfac.bas[12] within the module LUfac_module in the LU.xltm[13] Excel spreadsheet.

Forward and Back Substitution Algorithm

The forward and back substitution algorithms were introduced earlier in this document. Returning to the original matrix-vector equation

$$A\,\underline{x} = \underline{b}\,,$$

where A is a matrix and $\underline{x}$ and $\underline{b}$ are vectors. Through the determination of the LU factors of A, with pivoting, we can write $A = PLU$, and hence that $PLU\,\underline{x} = \underline{b}$. Using the elementary property of permutation matrices[14] ($P^{-1}=P^T$) and hence we may write $LU\,\underline{x} = P^T\underline{b}$, and hence there is effectively a rearrangement of the elements of $\underline{b}$ prior to the forward and back substitution, which was outlined earlier. The pseudo-code for the overall method is given below.

The forward and back substitution methods

```
Initialise pb as b with rows exchanged as indicated by perm
for i = 1 to n
   pb(i) = b(perm(i))

Solve Ly=b by forward substitution (pb stores y on completion)
for i = 1 to n
   sum = pb(i)
   for j = 1 to i - 1
      sum = sum - a(i, j) * pb(j)
   pb(i) = sum

Solve Ux=y by back substitution (pb stores the solution 'x' on completion)
for i = n to 1 step -1
   sum = pb(i)
   for j = i + 1 to n
      sum = sum - a(i, j) * pb(j)
   pb(i) = sum / a(i, i)

b is over-written by pb, the solution 'x'.
for i = 1 to n
   b(i) = pb(i)
```

For a typical implementation of the forward and back substitution method, see LUfbsub.bas[15], and its demonstration on the Excel Spreadsheet LU.xlsm[13]

[12] LUfac.bas
[13] LU.xlsm
[14] Permutation Matrix
[15] LUfbsub_bas