Notes on LU Factorization

Robert A. van de Geijn
Department of Computer Science
The University of Texas
Austin, TX 78712
rvdg@cs.utexas.edu

October 11, 2014

The LU factorization is also known as the LU decomposition and the operations it performs are equivalent to those performed by Gaussian elimination. We STRONGLY recommend that the reader consult "Linear Algebra: Foundations to Frontiers - Notes to LAFF With" [12] Weeks 6 and 7.

1 Definition and Existence

Definition 1. LU factorization (decomposition) *Given a matrix $A \in \mathbb{C}^{m\times n}$ with $m \leq n$ its LU factorization is given by $A = LU$ where $L \in \mathbb{C}^{m\times n}$ is unit lower trapezoidal and $U \in \mathbb{C}^{n\times n}$ is upper triangular.*

The first question we will ask is when the LU factorization exists. For this, we need a definition.

Definition 2. *The $k\times k$ principle leading submatrix of a matrix A is defined to be the square matrix $A_{TL} \in \mathbb{C}^{k\times k}$ such that $A = \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$.*

This definition allows us to indicate when a matrix has an LU factorization:

Theorem 3. Existence *Let $A \in \mathbb{C}^{m\times n}$ and $m \leq n$ have linearly independent columns. Then A has a unique LU factorization if and only if all its principle leading submatrices are nonsingular.*

The proof of this theorem is a bit involved and can be found in Section 4.

2 LU Factorization

We are going to present two different ways of deriving the most commonly known algorithm. The first is a straight forward derivation. The second presents the operation as the application of a sequence of `Gauss transforms`.

2.1 First derivation

Partition A, L, and U as follows:

$$A \rightarrow \left(\begin{array}{c|c} \alpha_{11} & a_{12}^T \\ \hline a_{21} & A_{22} \end{array}\right), \quad L \rightarrow \left(\begin{array}{c|c} 1 & 0 \\ \hline l_{21} & L_{22} \end{array}\right), \quad \text{and} \quad U \rightarrow \left(\begin{array}{c|c} \upsilon_{11} & u_{12}^T \\ \hline 0 & U_{22} \end{array}\right).$$

Then $A = LU$ means that

$$\left(\begin{array}{c|c} \alpha_{11} & a_{12}^T \\ \hline a_{21} & A_{22} \end{array}\right) = \left(\begin{array}{c|c} 1 & 0 \\ \hline l_{21} & L_{22} \end{array}\right)\left(\begin{array}{c|c} \upsilon_{11} & u_{12}^T \\ \hline 0 & U_{22} \end{array}\right) = \left(\begin{array}{c|c} \upsilon_{11} & u_{12}^T \\ \hline l_{21}\upsilon_{11} & l_{21}u_{12}^T + L_{22}U_{22} \end{array}\right).$$

This means that

$$\begin{array}{c|c} \alpha_{11} = \upsilon_{11} & a_{12}^T = u_{12}^T \\ \hline a_{21} = \upsilon_{11} l_{21} & A_{22} = l_{21}u_{12}^T + L_{22}U_{22} \end{array}$$

or, equivalently,

$$\begin{array}{c|c} \alpha_{11} = \upsilon_{11} & a_{12}^T = u_{12}^T \\ \hline a_{21} = \upsilon_{11} l_{21} & A_{22} - l_{21}u_{12}^T = L_{22}U_{22} \end{array}.$$

If we let U overwrite the original matrix A this suggests the algorithm

- $l_{21} = a_{21}/\alpha_{11}$.
- $a_{21} = 0$.
- $A_{22} := A_{22} - l_{21}a_{12}^T$.
- Continue by overwriting the updated A_{22} with its LU factorization.

This is captured in the algorithm in Figure 1.

2.2 Gauss transforms

Definition 4. *A matrix L_k of the form* $L_k = \left(\begin{array}{c|c|c} I_k & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & l_{21} & 0 \end{array}\right)$ *where I_k is $k \times k$ is called a* Gauss transform.

Example 5. *Gauss transforms can be used to take multiples of a row and subtract these multiples from other rows:*

$$\left(\begin{array}{c|cc|c} 1 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 \\ \hline 0 & -\lambda_{21} & 1 & 0 \\ 0 & -\lambda_{31} & 0 & 1 \end{array}\right)\left(\begin{array}{c} \widehat{a}_0^T \\ \hline \widehat{a}_1^T \\ \hline \widehat{a}_2^T \\ \widehat{a}_3^T \end{array}\right) = \left(\begin{array}{c} \widehat{a}_0^T \\ \hline \widehat{a}_1^T \\ \hline \left(\begin{array}{c} \widehat{a}_2^T \\ \widehat{a}_3^T \end{array}\right) - \left(\begin{array}{c} \lambda_{21} \\ \lambda_{31} \end{array}\right)\widehat{a}_1^T \end{array}\right) = \left(\begin{array}{c} \widehat{a}_0^T \\ \hline \widehat{a}_1^T \\ \hline \widehat{a}_2^T - \lambda_{21}\widehat{a}_1^T \\ \widehat{a}_3^T - \lambda_{31}\widehat{a}_1^T \end{array}\right).$$

Notice the similarity with what one does in Gaussian Elimination: take a multiples of one row and subtract these from other rows.

Now assume that the LU factorization in the previous subsection has proceeded to where A contains

$$\left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & a_{21} & A_{22} \end{array}\right)$$

where A_{00} is upper triangular (recall: it is being overwritten by U!). What we would like to do is eliminate the elements in a_{21} by taking multiples of the "current row" $\left(\begin{array}{c|c} \alpha_{11} & a_{12}^T \end{array}\right)$ and subtract these from the rest

Algorithm: Compute LU factorization of A, overwriting L with factor L and A with factor U

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$, $L \rightarrow \left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right)$

where A_{TL} and L_{TL} are 0×0

while $n(A_{TL}) < n(A)$ **do**

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right), \left(\begin{array}{c|c} L_{TL} & L_{TR} \\ \hline L_{BL} & L_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} L_{00} & 0 & 0 \\ \hline l_{10}^T & \lambda_{11} & 0 \\ \hline L_{20} & l_{21} & L_{22} \end{array}\right)$$

where α_{11}, λ_{11} are 1×1

$$\left\{\begin{array}{l} l_{21} := a_{21}/\alpha_{11} \\ A_{22} := A_{22} - l_{21}a_{12}^T \\ (a_{21} := 0) \end{array}\right.$$

or, alternatively,

$$\left\{\begin{array}{l} l_{21} := a_{21}/\alpha_{11} \\ \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & a_{21} & A_{22} \end{array}\right) := \left(\begin{array}{c|c|c} I & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & -l_{21} & 0 \end{array}\right) \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & a_{21} & A_{22} \end{array}\right) \\ \qquad = \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & 0 & A_{22} - l_{21}a_{12}^T \end{array}\right) \end{array}\right.$$

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right), \left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} L_{00} & 0 & 0 \\ \hline l_{10}^T & \lambda_{11} & 0 \\ \hline L_{20} & l_{21} & L_{22} \end{array}\right)$$

endwhile

Figure 1: Most commonly known algorithm for overwriting a matrix with its LU factorization.

of the rows: $\left(\begin{array}{c|c} a_{21} & A_{22} \end{array}\right)$. The vehicle is a Gauss transform: we must determine l_{21} so that

$$\left(\begin{array}{c|c|c} I & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & -l_{21} & I \end{array}\right) \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & a_{21} & A_{22} \end{array}\right) = \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & 0 & A_{22} - l_{21}a_{12}^T \end{array}\right).$$

This means we must pick $l_{21} = a_{21}/\alpha_{11}$ since

$$\left(\begin{array}{c|c|c} I & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & -l_{21} & I \end{array}\right) \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & a_{21} & A_{22} \end{array}\right) = \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & a_{21} - \alpha_{11}l_{21} & A_{22} - l_{21}a_{12}^T \end{array}\right).$$

The resulting algorithm is summarized in Figure 1 under "or, alternatively,". Notice that this algorithm is

identical to the algorithm for computing LU factorization discussed before!

How can this be? The following set of exercises explains it.

Exercise 6. *Show that*

$$\left(\begin{array}{c|c|c} I_k & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & -l_{21} & I \end{array}\right)^{-1} = \left(\begin{array}{c|c|c} I_k & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & l_{21} & I \end{array}\right)$$

Now, clearly, what the algorithm does is to compute a sequence of n Gauss transforms $\widehat{L}_0, \ldots, \widehat{L}_{n-1}$ such that $\widehat{L}_{n-1}\widehat{L}_{n-2}\cdots\widehat{L}_1\widehat{L}_0 A = U$. Or, equivalently, $A = L_0 L_1 \cdots L_{n-2}L_{n-1}U$, where $L_k = \widehat{L}_k^{-1}$. What will show next is that $L = L_0 L_1 \cdots L_{n-2}L_{n-1}$ is the unit lower triangular matrix computed by the LU factorization.

Exercise 7. *Let $\tilde{L}_k = L_0 L_1 \ldots L_k$. Assume that $\tilde{L}_k$ has the form $\tilde{L}_{k-1} = \left(\begin{array}{c|c|c} L_{00} & 0 & 0 \\ \hline l_{10}^T & 1 & 0 \\ \hline L_{20} & 0 & I \end{array}\right)$, where $\tilde{L}_{00}$ is $k \times k$.*

Show that $\tilde{L}_k$ is given by $\tilde{L}_k = \left(\begin{array}{c|c|c} L_{00} & 0 & 0 \\ \hline l_{10}^T & 1 & 0 \\ \hline L_{20} & l_{21} & I \end{array}\right)$.. (Recall: $\widehat{L}_k = \left(\begin{array}{c|c|c} I & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & -l_{21} & I \end{array}\right)$.)

What this exercise shows is that $L = L_0 L_1 \cdots L_{n-2}L_{n-1}$ is the triangular matrix that is created by simply placing the computed vectors l_{21} below the diagonal of a unit lower triangular matrix.

2.3 Cost of LU factorization

The cost of the LU factorization algorithm given in Figure 1 can be analyzed as follows:

- Assume A is $n \times n$.
- During the kth iteration, A_{TL} is initially $k \times k$.
- Computing $l_{21} := a_{21}/\alpha_{11}$ is typically implemented as $\beta := 1/\alpha_{11}$ and then the scaling $l_{21} := \beta a_{21}$. The reason is that divisions are expensive relative to multiplications. We will ignore the cost of the division (which will be insignificant if n is large). Thus, we count this as $n-k-1$ multiplies.
- The rank-1 update of A_{22} requires $(n-k-1)^2$ multiplications and $(n-k-1)^2$ additions.
- Thus, the total cost (in flops) can be approximated by

$$\begin{aligned}
\sum_{k=0}^{n-1} \left[(n-k-1) + 2(n-k-1)^2\right] &= \sum_{j=0}^{n-1} \left[j + 2j^2\right] \qquad \text{(Change of variable: } j = n-k-1\text{)} \\
&= \sum_{j=0}^{n-1} j + 2\sum_{j=0}^{n-1} j^2 \\
&\approx \frac{n(n-1)}{2} + 2\int_0^n x^2 dx \\
&= \frac{n(n-1)}{2} + \frac{2}{3}n^3 \\
&\approx \frac{2}{3}n^3
\end{aligned}$$

Notice that this involves roughly half the number of floating point operations as are required for a Householder transformation based QR factorization.

3 LU Factorization with Partial Pivoting

It is well-known that the LU factorization is numerically unstable under general circumstances. In particular, a backward stability analysis, given for example in [2, 5, 4] and summarized in Section 9, shows that the computed matrices $\check{L}$ and $\check{U}$ statisfy

$$(A + \Delta A) = \check{L}\check{U} \quad \text{where } |\Delta A| \leq \gamma_n |\check{L}||\check{U}|.$$

(This is the backward error result for the Crout variant for LU factorization, discussed later in this note. Some of the other variants have an error result of $(A + \Delta A) = \check{L}\check{U}$ where $|\Delta A| \leq \gamma_n(|A| + |\check{L}||\check{U}|)$.) Now, if α is small in magnitude compared to the entries of a_{21} then not only will l_{21} have large entries, but the update $A_{22} - l_{21}a_{12}^T$ will potentially introduce large entries in the updated A_{22} (in other words, the part of matrix A from which the future matrix U will be computed), a phenomenon referred to as *element growth*. To overcome this, we take will swap rows in A as the factorization proceeds, resulting in an algorithm known as LU factorization with partial pivoting.

3.1 Permutation matrices

Definition 8. *An $n \times n$ matrix P is said to be a* permutation *matrix, or permutation, if, when applied to a vector $x = (\chi_0, \chi_1, \ldots, \chi_{n-1})^{\mathrm{T}}$, it merely rearranges the order of the elements in that vector. Such a permutation can be represented by the vector of integers, $(\pi_0, \pi_1, \ldots, \pi_{n-1})^{\mathrm{T}}$, where $\{\pi_0, \pi_1, \ldots, \pi_{n-1}\}$ is a permutation of the integers $\{0, 1, \ldots, n-1\}$ and the permuted vector Px is given by $(\chi_{\pi_0}, \chi_{\pi_1}, \ldots, \chi_{\pi_{n-1}})^{\mathrm{T}}$.*

If P is a permutation matrix then PA rearranges the rows of A exactly as the elements of x are rearranged by Px.

We will see that when discussing the LU factorization with partial pivoting, a permutation matrix that swaps the first element of a vector with the π-th element of that vector is a fundamental tool. We will denote that matrix by

$$P(\pi) = \begin{cases} I_n & \text{if } \pi = 0 \\ \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & I_{\pi-1} & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{n-\pi-1} \end{pmatrix} & \text{otherwise,} \end{cases}$$

where n is the dimension of the permutation matrix. In the following we will use the notation P_n to indicate that the matrix P is of size n. Let p be a vector of integers satisfying the conditions

$$p = (\pi_0, \ldots, \pi_{k-1})^{\mathrm{T}}, \quad \text{where } 1 \leq k \leq n \text{ and } 0 \leq \pi_i < n - i, \tag{1}$$

then $P_n(p)$ will denote the permutation:

$$P_n(p) = \begin{pmatrix} I_{k-1} & 0 \\ 0 & P_{n-k+1}(\pi_{k-1}) \end{pmatrix} \begin{pmatrix} I_{k-2} & 0 \\ 0 & P_{n-k+2}(\pi_{k-2}) \end{pmatrix} \cdots \begin{pmatrix} 1 & 0 \\ 0 & P_{n-1}(\pi_1) \end{pmatrix} P_n(\pi_0).$$

Remark 9. *In the algorithms, the subscript that indicates the matrix dimensions is omitted.*

Algorithm: Compute LU factorization with partial pivoting of A, overwriting L with factor L and A with factor U. The pivot vector is returned in p.

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$,

$L \rightarrow \left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right)$, $p \rightarrow \left(\begin{array}{c} p_T \\ \hline p_B \end{array}\right)$.

where A_{TL} and L_{TL} are 0×0 and p_T is 0×1

while $n(A_{TL}) < n(A)$ **do**

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right),$$

$$\left(\begin{array}{c|c} L_{TL} & L_{TR} \\ \hline L_{BL} & L_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} L_{00} & 0 & 0 \\ \hline l_{10}^T & \lambda_{11} & 0 \\ \hline L_{20} & l_{21} & L_{22} \end{array}\right), \left(\begin{array}{c} p_T \\ \hline p_B \end{array}\right) \rightarrow \left(\begin{array}{c} p_0 \\ \hline \pi_1 \\ \hline p_2 \end{array}\right)$$

where α_{11}, λ_{11}, π_1 are 1×1

$$\left\{\begin{array}{l} \pi_1 = \text{maxi}\left(\begin{array}{c} \alpha_{11} \\ \hline a_{21} \end{array}\right) \\ \left(\begin{array}{c|c} \alpha_{11} & a_{12}^T \\ \hline a_{21} & A_{22} \end{array}\right) := P(\pi_1) \left(\begin{array}{c|c} \alpha_{11} & a_{12}^T \\ \hline a_{21} & A_{22} \end{array}\right) \\ l_{21} := a_{21}/\alpha_{11} \\ A_{22} := A_{22} - l_{21} a_{12}^T \\ (a_{21} := 0) \end{array}\right.$$

or, alternatively,

$$\left\{\begin{array}{l} \pi_1 = \text{maxi}\left(\begin{array}{c} \alpha_{11} \\ \hline a_{21} \end{array}\right) \\ \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & a_{21} & A_{22} \end{array}\right) := \left(\begin{array}{c|c} I & 0 \\ \hline 0 & P(\pi_1) \end{array}\right) \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & a_{21} & A_{22} \end{array}\right) \\ l_{21} := a_{21}/\alpha_{11} \\ \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & a_{21} & A_{22} \end{array}\right) := \left(\begin{array}{c|c|c} I & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & -l_{21} & 0 \end{array}\right) \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & a_{21} & A_{22} \end{array}\right) \\ \qquad = \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & 0 & A_{22} - l_{21} a_{12}^T \end{array}\right) \end{array}\right.$$

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right),$$

$$\left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} L_{00} & 0 & 0 \\ \hline l_{10}^T & \lambda_{11} & 0 \\ \hline L_{20} & l_{21} & L_{22} \end{array}\right), \left(\begin{array}{c} p_T \\ \hline p_B \end{array}\right) \leftarrow \left(\begin{array}{c} p_0 \\ \hline \pi_1 \\ \hline p_2 \end{array}\right)$$

endwhile

Figure 2: LU factorization with partial pivoting.

Example 10. *Let $a_0^T, a_1^T, \ldots, a_{n-1}^T$ be the rows of a matrix A. The application of $P(p)$ to A yields a matrix that results from swapping row a_0^T with $a_{\pi_0}^T$, then swapping a_1^T with $a_{\pi_1+1}^T$, a_2^T with $a_{\pi_2+2}^T$, until finally a_{k-1}^T is swapped with $a_{\pi_{k-1}+k-1}^T$.*

Remark 11. *For those familiar with how pivot information is stored in LINPACK and LAPACK, notice that those packages store the vector of pivot information $(\pi_0 + 1, \pi_1 + 2, \ldots, \pi_{k-1} + k)^{\mathrm{T}}$.*

3.2 The algorithm

Having introduced our notation for permutation matrices, we can now define the LU factorization with partial pivoting: Given an $n \times n$ matrix A, we wish to compute a) a vector p of n integers which satisfies the conditions (1), b) a unit lower trapezoidal matrix L, and c) an upper triangular matrix U so that $P(p)A = LU$. An algorithm for computing this operation is typically represented by

$$[A, p] := \text{LUPIV}(A),$$

where upon completion A has been overwritten by $\{L\backslash U\}$.

Let us start with revisiting the first derivation of the LU factorization. The first step is to find a first permutation matrix $P(\pi_1)$ such that the element on the diagonal in the first column is maximal in value. For this, we will introduce the function maxi(x) which, given a vector x, returns the index of the element in x with maximal magnitude (absolute value). The algorithm then proceeds as follows:

- Partition A, L as follows:

$$A \rightarrow \left(\begin{array}{c|c} \alpha_{11} & a_{12}^T \\ \hline a_{21} & A_{22} \end{array}\right), \quad \text{and} \quad L \rightarrow \left(\begin{array}{c|c} 1 & 0 \\ \hline l_{21} & L_{22} \end{array}\right).$$

- Compute $\pi_1 = \text{maxi}\left(\begin{array}{c} \alpha_{11} \\ \hline a_{21} \end{array}\right)$.

- Permute the rows: $\left(\begin{array}{c|c} \alpha_{11} & a_{12}^T \\ \hline a_{21} & A_{22} \end{array}\right) := P(\pi_1) \left(\begin{array}{c|c} \alpha_{11} & a_{12}^T \\ \hline a_{21} & A_{22} \end{array}\right)$.

- Compute $l_{21} := a_{21}/\alpha_{11}$.

- Update $A_{22} := A_{22} - l_{21} a_{12}^T$.

Now, in general, assume that the computation has proceeded to the point where matrix A has been overwritten by

$$\left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & a_{21} & A_{22} \end{array}\right)$$

where A_{00} is upper triangular. If no pivoting was added one would compute $l_{21} := a_{21}/\alpha_{11}$ followed by the update

$$\left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & a_{21} & A_{22} \end{array}\right) := \left(\begin{array}{c|c|c} I & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & -l_{21} & I \end{array}\right) \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & a_{21} & A_{22} \end{array}\right) = \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & 0 & A_{22} - l_{21} a_{12}^T \end{array}\right).$$

Now, instead one performs the steps

- Compute $\pi_1 = \text{maxi}\left(\begin{array}{c} \alpha_{11} \\ \hline a_{21} \end{array}\right)$.
- Permute the rows: $\left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & a_{21} & A_{22} \end{array}\right) := \left(\begin{array}{c|c} I & 0 \\ \hline 0 & P(\pi_1) \end{array}\right) \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & a_{21} & A_{22} \end{array}\right)$
- Compute $l_{21} := a_{21}/\alpha_{11}$.
- Update

$$\left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & a_{21} & A_{22} \end{array}\right) := \left(\begin{array}{c|c|c} I & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & -l_{21} & I \end{array}\right) \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & a_{21} & A_{22} \end{array}\right) = \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & 0 & A_{22} - l_{21} a_{12}^T \end{array}\right).$$

This algorithm is summarized in Figure 2.

Now, what this algorithm computes is a sequence of Gauss transforms $\widehat{L}_0, \ldots, \widehat{L}_{n-1}$ and permulations $P_0, \ldots, P_{n-1}$ such that

$$\widehat{L}_{n-1} P_{n-1} \cdots \widehat{L}_0 P_0 A = U$$

or, equivalently,

$$A = P_0^T L_0 \cdots \widehat{P}_{n-1}^T L_{n-1} U,$$

where $L_k = \widehat{L}_k^{-1}$. What we will finally show is that there are Gauss transforms $\bar{L}_0, \ldots \bar{L}_{n-1}$ (here the "bar" does NOT mean conjugation. It is just a symbol) such that

$$A = P_0^T \cdots P_{n-1}^T \underbrace{\bar{L}_0 \cdots \bar{L}_{n-1}}_{L} U$$

or, equivalently,

$$P(p)A = P_{n-1} \cdots P_0 A = \underbrace{\bar{L}_0 \cdots \bar{L}_{n-1}}_{L} U,$$

which is what we set out to compute.

Here is the insight. Assume that after k steps of LU factorization we have computed p_T, L_{TL}, L_{BL}, etc. so that

$$P(p_T)A = \left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & I \end{array}\right) \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline 0 & A_{BR} \end{array}\right),$$

where A_{TL} is upper triangular and $k \times k$.

Now compute the next step of LU factorization with partial pivoting with $\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline 0 & A_{BR} \end{array}\right)$:

- Partition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline 0 & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & a_{01} & A_{02} \end{array}\right)$$

- Compute $\pi_1 = \text{maxi}\left(\begin{array}{c} \alpha_{11} \\ \hline a_{21} \end{array}\right)$

Algorithm: Compute LU factorization with partial pivoting of A, overwriting L with factor L and A with factor U. The pivot vector is returned in p.

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$, $L \rightarrow \left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right)$, $p \rightarrow \left(\begin{array}{c} p_T \\ \hline p_B \end{array}\right)$.

where A_{TL} and L_{TL} are 0×0 and p_T is 0×1

while $n(A_{TL}) < n(A)$ **do**

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right),$$

$$\left(\begin{array}{c|c} L_{TL} & L_{TR} \\ \hline L_{BL} & L_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} L_{00} & 0 & 0 \\ \hline l_{10}^T & \lambda_{11} & 0 \\ \hline L_{20} & l_{21} & L_{22} \end{array}\right), \left(\begin{array}{c} p_T \\ \hline p_B \end{array}\right) \rightarrow \left(\begin{array}{c} p_0 \\ \hline \pi_1 \\ \hline p_2 \end{array}\right)$$

where α_{11}, λ_{11}, π_1 are 1×1

$$\left\{\begin{array}{l} \pi_1 = \text{maxi}\left(\begin{array}{c} \alpha_{11} \\ \hline a_{21} \end{array}\right) \\ \left(\begin{array}{c|c|c} l_{10}^T & \alpha_{11} & a_{12}^T \\ \hline L_{20} & a_{21} & A_{22} \end{array}\right) := P(\pi_1) \left(\begin{array}{c|c|c} l_{10}^T & \alpha_{11} & a_{12}^T \\ \hline L_{20} & a_{21} & A_{22} \end{array}\right) \\ l_{21} := a_{21}/\alpha_{11} \\ A_{22} := A_{22} - l_{21} a_{12}^T \\ (a_{21} := 0) \end{array}\right.$$

or, alternatively,

$$\left\{\begin{array}{l} \pi_1 = \text{maxi}\left(\begin{array}{c} \alpha_{11} \\ \hline a_{21} \end{array}\right) \\ \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline l_{10}^T & \alpha_{11} & a_{12}^T \\ \hline L_{20} & a_{21} & A_{22} \end{array}\right) := \left(\begin{array}{c|c} I & 0 \\ \hline 0 & P(\pi_1) \end{array}\right) \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline l_{10}^T & \alpha_{11} & a_{12}^T \\ \hline L_{20} & a_{21} & A_{22} \end{array}\right) \\ l_{21} := a_{21}/\alpha_{11} \\ \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & a_{21} & A_{22} \end{array}\right) := \left(\begin{array}{c|c|c} I & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & -l_{21} & 0 \end{array}\right) \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & a_{21} & A_{22} \end{array}\right) \\ \qquad = \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & 0 & A_{22} - l_{21} a_{12}^T \end{array}\right) \end{array}\right.$$

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right),$$

$$\left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} L_{00} & 0 & 0 \\ \hline l_{10}^T & \lambda_{11} & 0 \\ \hline L_{20} & l_{21} & L_{22} \end{array}\right), \left(\begin{array}{c} p_T \\ \hline p_B \end{array}\right) \leftarrow \left(\begin{array}{c} p_0 \\ \hline \pi_1 \\ \hline p_2 \end{array}\right)$$

endwhile

Figure 3: LU factorization with partial pivoting.

Algorithm: Compute LU factorization with partial pivoting of A, overwriting A with factors L and U. The pivot vector is returned in p.

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$, $p \rightarrow \left(\begin{array}{c} p_T \\ \hline p_B \end{array}\right)$.

where A_{TL} is 0×0 and p_T is 0×1

while $n(A_{TL}) < n(A)$ **do**

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right), \left(\begin{array}{c} p_T \\ \hline p_B \end{array}\right) \rightarrow \left(\begin{array}{c} p_0 \\ \hline \pi_1 \\ \hline p_2 \end{array}\right)$$

where α_{11}, λ_{11}, π_1 are 1×1

$\pi_1 = \text{maxi}\left(\begin{array}{c} \alpha_{11} \\ \hline a_{21} \end{array}\right)$

$$\left(\begin{array}{c|c|c} a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right) := P(\pi_1) \left(\begin{array}{c|c|c} a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right)$$

$a_{21} := a_{21}/\alpha_{11}$

$A_{22} := A_{22} - a_{21}a_{12}^T$

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right), \left(\begin{array}{c} p_T \\ \hline p_B \end{array}\right) \leftarrow \left(\begin{array}{c} p_0 \\ \hline \pi_1 \\ \hline p_2 \end{array}\right)$$

endwhile

Figure 4: LU factorization with partial pivoting, overwriting A with the factors.

- Permute $\left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & a_{21} & A_{22} \end{array}\right) := \left(\begin{array}{c|c} I & 0 \\ \hline 0 & P(\pi_1) \end{array}\right) \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & a_{21} & A_{22} \end{array}\right)$
- Compute $l_{21} := a_{21}/\alpha_{11}$.
- Update

$$\left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & a_{21} & A_{22} \end{array}\right) := \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & 0 & A_{22} - l_{21}a_{12}^T \end{array}\right)$$

After this,

$$P(p_T)A = \left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & I \end{array}\right) \left(\begin{array}{c|c} I & 0 \\ \hline 0 & P(\pi_1) \end{array}\right) \left(\begin{array}{c|c|c} I & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & l_{21} & I \end{array}\right) \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & 0 & A_{22} - l_{21}a_{12}^T \end{array}\right) \tag{2}$$

But

$$\left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & I \end{array}\right)\left(\begin{array}{c|c} I & 0 \\ \hline 0 & P(\pi_1) \end{array}\right)\left(\begin{array}{c|c|c} I & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & l_{21} & I \end{array}\right)\left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & 0 & A_{22}-l_{21}a_{12}^T \end{array}\right)$$

$$= \left(\begin{array}{c|c} I & 0 \\ \hline 0 & P(\pi_1) \end{array}\right)\left(\begin{array}{c|c} L_{TL} & 0 \\ \hline P(\pi_1)L_{BL} & I \end{array}\right)\left(\begin{array}{c|c|c} I & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & l_{21} & I \end{array}\right)\left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & 0 & A_{22}-l_{21}a_{12}^T \end{array}\right)$$

$$= \left(\begin{array}{c|c} I & 0 \\ \hline 0 & P(\pi_1) \end{array}\right)\left(\begin{array}{c|c} L_{TL} & 0 \\ \hline \bar{L}_{BL} & I \end{array}\right)\left(\begin{array}{c|c|c} I & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & l_{21} & I \end{array}\right)\left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & 0 & A_{22}-l_{21}a_{12}^T \end{array}\right),,$$

NOTE: There is a "bar" above some of L's and l's. Very hard to see!!! For this reason, these show up in red as well. Here we use the fact that $P(\pi_1) = P(\pi_1)^T$ because of its very special structure.

Bringing the permutation to the left of (2) and "repartitioning" we get

$$\underbrace{\left(\begin{array}{c|c} I & 0 \\ \hline 0 & P(\pi_1) \end{array}\right)P(p_0)}_{P\left(\begin{array}{c} p_0 \\ \hline \pi_1 \end{array}\right)} A = \underbrace{\left(\begin{array}{c|c|c} L_{00} & 0 & 0 \\ \hline \bar{l}_{10}^T & 1 & 0 \\ \hline \bar{L}_{20} & 0 & I \end{array}\right)\left(\begin{array}{c|c|c} I & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & l_{21} & I \end{array}\right)}_{\left(\begin{array}{c|c|c} L_{00} & 0 & 0 \\ \hline \bar{l}_{10}^T & 1 & 0 \\ \hline \bar{L}_{20} & l_{21} & I \end{array}\right)}\left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & 0 & A_{22}-l_{21}a_{12}^T \end{array}\right).$$

This explains how the algorithm in Figure 3 compute p, L, and U (overwriting A with U) so that $P(p)A = LU$.

Finally, we recognize that L can overwrite the entries of A below its diagonal, yielding the algorithm in Figure 4.

4 Proof of Theorem 1

Proof:

($\Rightarrow$) Let nonsingular A have a (unique) LU factorization. We will show that its principle leading submatrices are nonsingular. Let

$$\underbrace{\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)}_{A} = \underbrace{\left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right)}_{L}\underbrace{\left(\begin{array}{c|c} U_{TL} & U_{TR} \\ \hline 0 & U_{BR} \end{array}\right)}_{U}$$

be the LU factorization of A where A_{TL}, L_{TL}, and U_{TL} are $k \times k$. Notice that U cannot have a zero on the diagonal since then A would not have linearly independent columns. Now, the $k \times k$ principle leading submatrix A_{TL} equals $A_{TL} = L_{TL}U_{TL}$ which is nonsingular since L_{TL} has a unit diagonal and U_{TL} has no zeroes on the diagonal. Since k was chosen arbitrarily, this means that all principle leading submatrices are nonsingular.

($\Leftarrow$) We will do a proof by induction on n.

Base Case: $n = 1$. Then A has the form $A = \left(\begin{array}{c} \alpha_{11} \\ \hline a_{21} \end{array}\right)$ where $\alpha 11$ is a scalar. Since the principle leading submatrices are nonsingular $\alpha_{11} \neq 0$. Hence $A = \underbrace{\left(\begin{array}{c} 1 \\ \hline a_{21}/\alpha_{11} \end{array}\right)}_{L} \underbrace{\alpha_{11}}_{U}$ is the LU factorization of A.

This LU factorization is unique because the first element of L must be 1.

Inductive Step: Assume the result is true for all matrices with $n = k$. Show it is true for matrices with $n = k + 1$.

Let A of size $n = k + 1$ have nonsingular principle leading submatrices. Now, *if* an LU factorization of A exists, $A = LU$, then it would have to form

$$\underbrace{\left(\begin{array}{c|c} A_{00} & a_{01} \\ \hline a_{10}^T & \alpha_{11} \\ \hline A_{20} & a_{21} \end{array}\right)}_{A} = \underbrace{\left(\begin{array}{c|c} L_{00} & 0 \\ \hline l_{10}^T & 1 \\ \hline L_{20} & l_{21} \end{array}\right)}_{L} \underbrace{\left(\begin{array}{c|c} U_{00} & u_{01} \\ \hline 0 & \upsilon_{11} \end{array}\right)}_{U}. \tag{3}$$

If we can show that the different parts of L and U exist and are unique, we are done. Equation (3) can be rewritten as

$$\left(\begin{array}{c} A_{00} \\ \hline a_{10}^T \\ \hline A_{20} \end{array}\right) = \left(\begin{array}{c} L_{00} \\ \hline l_{10}^T \\ \hline L_{20} \end{array}\right) U_{00} \quad \text{and} \quad \left(\begin{array}{c} a_{01} \\ \hline \alpha_{11} \\ \hline a_{21} \end{array}\right) = \left(\begin{array}{c} L_{00}u_{01} \\ \hline l_{10}^T u_{01} + \upsilon_{11} \\ \hline L_{20}u_{01} + l_{21}\upsilon_{11} \end{array}\right).$$

Now, by the Induction Hypothesis L_{11}, l_{10}^T, and L_{20} exist and are unique. So the question is whether u_{01}, υ_{11}, and l_{21} exist and are unique:

- u_{01} **exists and is unique.** Since L_{00} is nonsingular (it has ones on its diagonal) $L_{00}u_{01} = a_{01}$ has a solution that is unique.
- υ_{11} **exists, is unique, and is nonzero.** Since l_{10}^T and u_{01} exist and are unique, $\upsilon_{11} = \alpha_{11} - l_{10}^T u_{01}$ exists and is unique. It is also nonzero since the principle leading submatrix of A given by

  $$\left(\begin{array}{c|c} A_{00} & a_{01} \\ \hline a_{10}^T & \alpha_{11} \end{array}\right) = \left(\begin{array}{c|c} L_{00} & 0 \\ \hline l_{10}^T & 1 \end{array}\right) \left(\begin{array}{c|c} U_{00} & u_{01} \\ \hline 0 & \upsilon_{11} \end{array}\right),$$

  is nonsingular by assumption and therefore υ_{11} must be nonzero.
- l_{21} **exists and is unique.** Since υ_{11} exists and is nonzero, $l_{21} = a_{21}/\upsilon_{11}$ exists and is uniquely determined.

Thus the $m \times (k + 1)$ matrix A has a unique LU factorization.

By the Principle of Mathematical Induction the result holds.

Exercise 12. *Implement LU factorization with partial pivoting with the FLAME@lab API, in M-script.*

5 LU with Complete Pivoting

LU factorization with partial pivoting builds on the insight that pivoting (rearranging) rows in a linear system does not change the solution: if $Ax = b$ then $P(p)Ax = P(p)b$, where p is a pivot vector. Now, if r is another pivot vector, then notice that $P(r)^T P(r) = I$ (a simple property of pivot matrices) and $AP(r)^T$ permutes the columns of A in exactly the same order as $P(r)A$ permutes the rows of A.

What this means is that if $Ax = b$ then $P(p)AP(r)^T[P(r)x] = P(p)b$. This supports the idea that one might want to not only permute rows of A, as in partial pivoting, but also columns of A. This is done in a variation on LU factorization that is known as LU factorization with *complete pivoting*.

The idea is as follows: Given matrix A, partition

$$A = \left(\begin{array}{c|c} \alpha_{11} & a_{12}^T \\ \hline a_{21} & A_{22} \end{array}\right).$$

Now, instead of finding the largest element in magnitude in the first column, find the largest element in magnitude in the entire matrix. Let's say it is element (π_0, ρ_0). Then, one permutes

$$\left(\begin{array}{c|c} \alpha_{11} & a_{12}^T \\ \hline a_{21} & A_{22} \end{array}\right) := P(\pi_0) \left(\begin{array}{c|c} \alpha_{11} & a_{12}^T \\ \hline a_{21} & A_{22} \end{array}\right) P(\rho_0)^T,$$

making α_{11} the largest element in magnitude. This then reduces the magnitude of multipliers and element growth.

It can be shown that the maximal element growth experienced when employing LU with complete pivoting indeed reduces element growth. The problem is that it requires $O(n^2)$ comparisons per iteration. Worse, it completely destroys the ability to utilize blocked algorithms, which attain much greater performance.

In practice LU with complete pivoting is not used.

6 Solving $Ax = y$ Via the LU Factorization with Pivoting

Given nonsingular matrix $A \in \mathbb{C}^{m \times m}$, the above discussions have yielded an algorithm for computing permutation matrix P, unit lower triangular matrix L and upper triangular matrix U such that $PA = LU$. We now discuss how these can be used to solve the system of linear equations $Ax = y$.

Starting with

$$Ax = y$$

we multiply both sizes of the equation by permutation matrix P

$$PAx = \underbrace{Py}_{\widehat{y}}$$

and substitute LU for PA

$$L \underbrace{Ux}_{z} \widehat{y}.$$

We now notice that we can solve the lower triangular system

$$Lz = \widehat{y}$$

after which x can be computed by solving the upper triangular system

$$Ux = z.$$

Algorithm: Solve $Lz = y$, overwriting y (Variant 1)

Partition $L \rightarrow \left(\begin{array}{c|c} L_{TL} & L_{TR} \\ \hline L_{BL} & L_{BR} \end{array}\right)$, $y \rightarrow \left(\begin{array}{c} y_T \\ \hline y_B \end{array}\right)$

where L_{TL} is 0×0, y_T has 0 rows

while $m(L_{TL}) < m(L)$ **do**

Repartition

$$\left(\begin{array}{c|c} L_{TL} & L_{TR} \\ \hline L_{BL} & L_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} L_{00} & l_{01} & L_{02} \\ \hline l_{10}^T & \lambda_{11} & l_{12}^T \\ \hline L_{20} & l_{21} & L_{22} \end{array}\right),$$

$$\left(\begin{array}{c} y_T \\ \hline y_B \end{array}\right) \rightarrow \left(\begin{array}{c} y_0 \\ \hline \psi_1 \\ \hline y_2 \end{array}\right)$$

where λ_{11} is 1×1, ψ_1 has 1 row

$y_2 := y_2 - \psi_1 l_{21}$

Continue with

$$\left(\begin{array}{c|c} L_{TL} & L_{TR} \\ \hline L_{BL} & L_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} L_{00} & l_{01} & L_{02} \\ \hline l_{10}^T & \lambda_{11} & l_{12}^T \\ \hline L_{20} & l_{21} & L_{22} \end{array}\right),$$

$$\left(\begin{array}{c} y_T \\ \hline y_B \end{array}\right) \leftarrow \left(\begin{array}{c} y_0 \\ \hline \psi_1 \\ \hline y_2 \end{array}\right)$$

endwhile

Algorithm: Solve $Lz = y$, overwriting y (Variant 2)

Partition $L \rightarrow \left(\begin{array}{c|c} L_{TL} & L_{TR} \\ \hline L_{BL} & L_{BR} \end{array}\right)$, $y \rightarrow \left(\begin{array}{c} y_T \\ \hline y_B \end{array}\right)$

where L_{TL} is 0×0, y_T has 0 rows

while $m(L_{TL}) < m(L)$ **do**

Repartition

$$\left(\begin{array}{c|c} L_{TL} & L_{TR} \\ \hline L_{BL} & L_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} L_{00} & l_{01} & L_{02} \\ \hline l_{10}^T & \lambda_{11} & l_{12}^T \\ \hline L_{20} & l_{21} & L_{22} \end{array}\right),$$

$$\left(\begin{array}{c} y_T \\ \hline y_B \end{array}\right) \rightarrow \left(\begin{array}{c} y_0 \\ \hline \psi_1 \\ \hline y_2 \end{array}\right)$$

where λ_{11} is 1×1, ψ_1 has 1 row

$\psi_1 := \psi_1 - l_{10}^T y_0$

Continue with

$$\left(\begin{array}{c|c} L_{TL} & L_{TR} \\ \hline L_{BL} & L_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} L_{00} & l_{01} & L_{02} \\ \hline l_{10}^T & \lambda_{11} & l_{12}^T \\ \hline L_{20} & l_{21} & L_{22} \end{array}\right),$$

$$\left(\begin{array}{c} y_T \\ \hline y_B \end{array}\right) \leftarrow \left(\begin{array}{c} y_0 \\ \hline \psi_1 \\ \hline y_2 \end{array}\right)$$

endwhile

Figure 5: Algorithms for the solution of a unit lower triangular system $Lz = y$ that overwrite y with z.

7 Solving Triangular Systems of Equations

7.1 $Lz = y$

First, we discuss solving $Lz = y$ where L is a unit lower triangular matrix.

Variant 1

Consider $Lz = y$ where L is unit lower triangular. Partition

$$L \rightarrow \left(\begin{array}{c|c} 1 & 0 \\ \hline l_{21} & L_{22} \end{array}\right), \quad z \rightarrow \left(\begin{array}{c} \zeta_1 \\ \hline z_2 \end{array}\right) \quad \text{and} \quad y \rightarrow \left(\begin{array}{c} \psi_1 \\ \hline y_2 \end{array}\right).$$

Then

$$\underbrace{\left(\begin{array}{c|c} 1 & 0 \\ \hline l_{21} & L_{22} \end{array}\right)}_{L} \underbrace{\left(\begin{array}{c} \zeta_1 \\ \hline z_2 \end{array}\right)}_{z} = \underbrace{\left(\begin{array}{c} \psi_1 \\ \hline y_2 \end{array}\right)}_{y}.$$

Multiplying out the left-hand side yields

$$\left(\begin{array}{c} \zeta_1 \\ \hline \zeta_1 l_{21} + L_{22} z_2 \end{array}\right) = \left(\begin{array}{c} \psi_1 \\ \hline y_2 \end{array}\right)$$

and the equalities

$$\begin{array}{rcl} \zeta_1 & = & \psi_1 \\ \zeta_1 l_{21} + L_{22} z_2 & = & y_2, \end{array}$$

which can be rearranged as

$$\begin{array}{rcl} \zeta_1 & = & \psi_1 \\ L_{22} z_2 & = & y_2 - \zeta_1 l_{21}. \end{array}$$

These insights justify the algorithm in Figure 5 (left), which overwrites y with the solution to $Lz = y$.

Variant 2

An alternative algorithm can be derived as follows: Partition

$$L \rightarrow \left(\begin{array}{c|c} L_{00} & 0 \\ \hline l_{10}^T & 1 \end{array} \right), \quad z \rightarrow \left(\begin{array}{c} z_0 \\ \hline \zeta_1 \end{array} \right) \quad \text{and} \quad y \rightarrow \left(\begin{array}{c} y_0 \\ \hline \psi_1 \end{array} \right).$$

Then

$$\underbrace{\left(\begin{array}{c|c} L_{00} & 0 \\ \hline l_{10}^T & 1 \end{array} \right)}_{L} \underbrace{\left(\begin{array}{c} z_0 \\ \hline \zeta_1 \end{array} \right)}_{z} = \underbrace{\left(\begin{array}{c} y_0 \\ \hline \psi_1 \end{array} \right)}_{y}.$$

Multiplying out the left-hand side yields

$$\left(\begin{array}{c} L_{00} z_0 \\ \hline l_{10}^T z_0 + \zeta_1 \end{array} \right) = \left(\begin{array}{c} y_0 \\ \hline \psi_1 \end{array} \right)$$

and the equalities

$$\begin{array}{rcl} L_{00} z_0 & = & y_0 \\ l_{10}^T z_0 + \zeta_1 & = & \psi_1. \end{array}$$

The idea now is as follows: Assume that the elements of z_0 were computed in previous iterations in the algorithm in Figure 5 (left), overwriting y_0. Then in the current iteration we can compute $\zeta_1 := \psi_0 - l_{10}^T z_0$, overwriting ψ_1.

Discussion

Notice that Variant 1 casts the computation in terms of an AXPY operation while Variant 2 casts it in terms of DOT products.

7.2 $Ux = z$

Next, we discuss solving $Ux = y$ where U is an upper triangular matrix (with no assumptions about its diagonal entries).

Exercise 13. *Derive an algorithm for solving $Ux = y$, overwriting y with the solution, that casts most computation in terms of* DOT *products. Hint: Partition*

$$U \rightarrow \left(\begin{array}{c|c} \upsilon_{11} & u_{12}^T \\ \hline 0 & U_{22} \end{array} \right).$$

Call this Variant 1 and use Figure 6 to state the algorithm.

Exercise 14. *Derive an algorithm for solving $Ux = y$, overwriting y with the solution, that casts most computation in terms of* AXPY *operations. Call this Variant 2 and use Figure 6 to state the algorithm.*

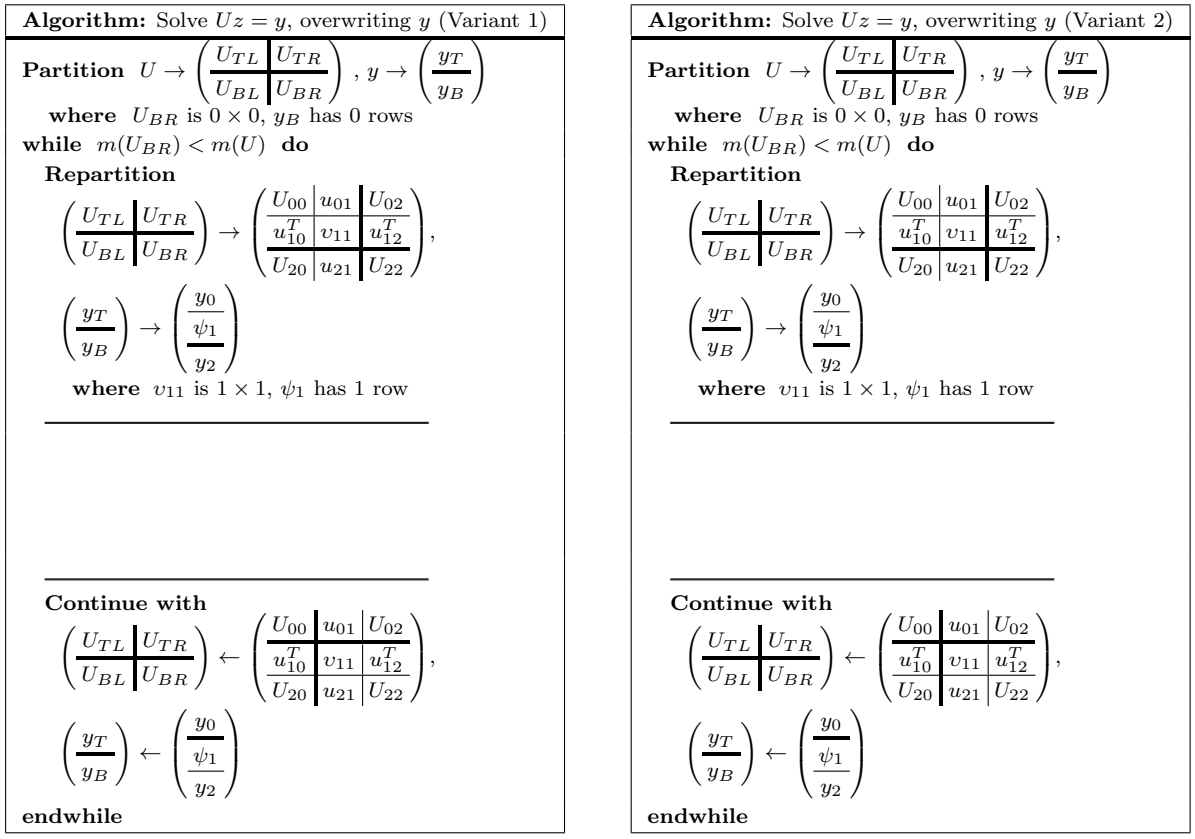

Figure 6: Algorithms for the solution of an upper triangular system $Ux = y$ that overwrite y with x.

8 Other LU Factorization Algorithms

There are actually five different (unblocked) algorithms for computing the LU factorization that were discovered over the course of the centuries[1]. The LU factorization in Figure 1 is sometimes called *classical LU factorization* or the *right-looking* algorithm. We now briefly describe how to derive the other algorithms.

Finding the algorithms starts with the following observations.

- Our algorithms will overwrite the matrix A, and hence we introduce $\widehat{A}$ to denote the original contents of A. We will say that the *precondition* for the algorithm is that

  $$A = \widehat{A}$$

  (A starts by containing the original contents of A.)

- We wish to overwrite A with L and U. Thus, the postcondition for the algorithm (the state in which we wish to exit the algorithm) is that

  $$A = L\backslash U \wedge LU = \widehat{A}$$

  (A is overwritten by L below the diagonal and U on and above the diagonal, where multiplying L and U yields the original matrix A.)

[1]For a thorough discussion of the different LU factorization algorithms that also gives a historic perspective, we recommend "Matrix Algorithms Volume 1" by G.W. Stewart [13]

- All the algorithms will march through the matrices from top-left to bottom-right. Thus, at a representative point in the algorithm, the matrices are viewed as quadrants:

$$A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right), \quad L \rightarrow \left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right), \quad \text{and} \quad U \rightarrow \left(\begin{array}{c|c} U_{TL} & U_{TR} \\ \hline 0 & U_{BR} \end{array}\right).$$

  where A_{TL}, L_{TL}, and U_{TL} are all square and equally sized.

- In terms of these exposed quadrants, in the end we wish for matrix A to contain

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) = \left(\begin{array}{c|c} L\backslash U_{TL} & U_{TR} \\ \hline L_{BL} & L\backslash U_{BR} \end{array}\right)$$
$$\text{where} \left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right)\left(\begin{array}{c|c} U_{TL} & U_{TR} \\ \hline 0 & U_{BR} \end{array}\right) = \left(\begin{array}{c|c} \widehat{A}_{TL} & \widehat{A}_{TR} \\ \hline \widehat{A}_{BL} & \widehat{A}_{BR} \end{array}\right)$$

- Manipulating this yields what we call the Partitioned Matrix Expression (PME), which can be viewed as a recursive definition of the LU factorization:

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) = \left(\begin{array}{c|c} L\backslash U_{TL} & U_{TR} \\ \hline L_{BL} & L\backslash U_{BR} \end{array}\right)$$
$$\wedge \left(\begin{array}{c|c} L_{TL}U_{TL} = \widehat{A}_{TL} & L_{TL}U_{TR} = \widehat{A}_{TR} \\ \hline L_{BL}U_{TL} = \widehat{A}_{BL} & L_{BL}U_{TR} + L_{BR}U_{BR} = \widehat{A}_{BR} \end{array}\right)$$

  Now, consider the code skeleton for the LU factorization in Figure 7. At the top of the loop (right after the **while**), we want to maintain certain contents in matrix A. Since we are in a loop, we haven't yet overwritten A with the final result. Instead, some progress toward this final result have been made. The way we can find what the state of A is that we would like to maintain is to take the PME and delete subexpression. For example, consider the following condition on the contents of A:

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) = \left(\begin{array}{c|c} L\backslash U_{TL} & U_{TR} \\ \hline L_{BL} & \widehat{A}_{BR} - L_{BL}U_{TR} \end{array}\right)$$
$$\wedge \left(\begin{array}{c|c} L_{TL}U_{TL} = \widehat{A}_{TL} & L_{TL}U_{TR} = \widehat{A}_{TR} \\ \hline L_{BL}U_{TL} = \widehat{A}_{BL} & \cancel{L_{BL}U_{TR} + L_{BR}U_{BR} = \widehat{A}_{BR}} \end{array}\right).$$

  What we are saying is that A_{TL}, A_{TR}, and A_{BL} have been completely updated with the corresponding parts of L and U, and A_{BR} has been partially updated. **This is exactly the state that the algorithm that we discussed previously in this document maintains!** What is left is to factor A_{BR}, since it contains $\widehat{A}_{BR} - L_{BL}U_{TR}$, and $\widehat{A}_{BR} - L_{BL}U_{TR} = L_{BR}U_{BR}$.

- By carefully analyzing the order in which computation must occur (in compiler lingo: by performing a dependence analysis), we can identify five states that can be maintained at the top of the loop, by deleting subexpressions from the PME. These are called *loop invariants* and are listed in Figure 8.

- Key to figuring out what updates must occur in the loop for each of the variants is to look at how the matrices are repartitioned at the top and bottom of the loop body.

Algorithm: $A := \mathrm{LU}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$, $L \rightarrow \left(\begin{array}{c|c} L_{TL} & L_{TR} \\ \hline L_{BL} & L_{BR} \end{array}\right)$, $U \rightarrow \left(\begin{array}{c|c} U_{TL} & U_{TR} \\ \hline U_{BL} & U_{BR} \end{array}\right)$

where A_{TL} is 0×0, L_{TL} is 0×0, U_{TL} is 0×0

while $m(A_{TL}) < m(A)$ **do**

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right), \left(\begin{array}{c|c} L_{TL} & L_{TR} \\ \hline L_{BL} & L_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} L_{00} & l_{01} & L_{02} \\ \hline l_{10}^T & \lambda_{11} & l_{12}^T \\ \hline L_{20} & l_{21} & L_{22} \end{array}\right),$$

$$\left(\begin{array}{c|c} U_{TL} & U_{TR} \\ \hline U_{BL} & U_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} U_{00} & u_{01} & U_{02} \\ \hline u_{10}^T & \upsilon_{11} & u_{12}^T \\ \hline U_{20} & u_{21} & U_{22} \end{array}\right)$$

where α_{11} is 1×1, λ_{11} is 1×1, υ_{11} is 1×1

- ---

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right), \left(\begin{array}{c|c} L_{TL} & L_{TR} \\ \hline L_{BL} & L_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} L_{00} & l_{01} & L_{02} \\ \hline l_{10}^T & \lambda_{11} & l_{12}^T \\ \hline L_{20} & l_{21} & L_{22} \end{array}\right),$$

$$\left(\begin{array}{c|c} U_{TL} & U_{TR} \\ \hline U_{BL} & U_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} U_{00} & u_{01} & U_{02} \\ \hline u_{10}^T & \upsilon_{11} & u_{12}^T \\ \hline U_{20} & u_{21} & U_{22} \end{array}\right)$$

endwhile

Figure 7: Code skeleton for LU factorization.

Variant	Algorithm	State (loop invariant)
1	Bordered	$\left(\begin{array}{c\|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) = \left(\begin{array}{c\|c} L\backslash U_{TL} & \widehat{A}_{TR} \\ \hline \widehat{A}_{BL} & \widehat{A}_{BR} \end{array}\right)$ $\wedge \begin{array}{c\|c} L_{TL}U_{TL} = \widehat{A}_{TL} & \cancel{L_{TL}U_{TR} = \widehat{A}_{TR}} \\ \hline \cancel{L_{BL}U_{TL} = \widehat{A}_{BL}} & L_{BL}U_{TR} + \cancel{L_{BR}U_{BR}} = \widehat{A}_{BR} \end{array}$
2	Left-looking	$\left(\begin{array}{c\|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) = \left(\begin{array}{c\|c} L\backslash U_{TL} & \widehat{A}_{TR} \\ \hline L_{BL} & \widehat{A}_{BR} \end{array}\right)$ $\wedge \begin{array}{c\|c} L_{TL}U_{TL} = \widehat{A}_{TL} & \cancel{L_{TL}U_{TR} = \widehat{A}_{TR}} \\ \hline L_{BL}U_{TL} = \widehat{A}_{BL} & L_{BL}U_{TR} + \cancel{L_{BR}U_{BR}} = \widehat{A}_{BR} \end{array}$
3	Up-looking	$\left(\begin{array}{c\|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) = \left(\begin{array}{c\|c} L\backslash U_{TL} & U_{TR} \\ \hline \widehat{A}_{BL} & \widehat{A}_{BR} \end{array}\right)$ $\wedge \begin{array}{c\|c} L_{TL}U_{TL} = \widehat{A}_{TL} & L_{TL}U_{TR} = \widehat{A}_{TR} \\ \hline \cancel{L_{BL}U_{TL} = \widehat{A}_{BL}} & L_{BL}U_{TR} + \cancel{L_{BR}U_{BR}} = \widehat{A}_{BR} \end{array}$
4	Crout variant	$\left(\begin{array}{c\|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) = \left(\begin{array}{c\|c} L\backslash U_{TL} & U_{TR} \\ \hline L_{BL} & \widehat{A}_{BR} \end{array}\right)$ $\wedge \begin{array}{c\|c} L_{TL}U_{TL} = \widehat{A}_{TL} & L_{TL}U_{TR} = \widehat{A}_{TR} \\ \hline L_{BL}U_{TL} = \widehat{A}_{BL} & L_{BL}U_{TR} + \cancel{L_{BR}U_{BR}} = \widehat{A}_{BR} \end{array}$
5	Classical LU	$\left(\begin{array}{c\|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) = \left(\begin{array}{c\|c} L\backslash U_{TL} & U_{TR} \\ \hline L_{BL} & \widehat{A}_{BR} - L_{BL}U_{TR} \end{array}\right)$ $\wedge \begin{array}{c\|c} L_{TL}U_{TL} = \widehat{A}_{TL} & L_{TL}U_{TR} = \widehat{A}_{TR} \\ \hline L_{BL}U_{TL} = \widehat{A}_{BL} & L_{BL}U_{TR} + \cancel{L_{BR}U_{BR}} = \widehat{A}_{BR} \end{array}$

Figure 8: Loop invariants for various LU factorization algorithms.

8.1 Variant 1: Bordered algorithm

Consider the loop invariant:

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) = \left(\begin{array}{c|c} L\backslash U_{TL} & \widehat{A}_{TR} \\ \hline \widehat{A}_{BL} & \widehat{A}_{BR} \end{array}\right)$$
$$\wedge \begin{array}{c|c} L_{TL}U_{TL} = \widehat{A}_{TL} & \cancel{L_{TL}U_{TR} = \widehat{A}_{TR}} \\ \hline \cancel{L_{BL}U_{TL} = \widehat{A}_{BL}} & \cancel{L_{BL}U_{TR} + L_{BR}U_{BR} = \widehat{A}_{BR}} \end{array}$$

At the top of the loop, after repartitioning, A contains

$$\begin{array}{c|c|c} L\backslash U_{00} & \widehat{a}_{01} & \widehat{A}_{02} \\ \hline \widehat{a}_{10}^T & \widehat{\alpha}_{11} & \widehat{a}_{12}^T \\ \hline \widehat{A}_{20} & \widehat{a}_{21} & \widehat{A}_{22} \end{array}$$

while at the bottom it must contain

$$\begin{array}{c|c|c} L\backslash U_{00} & u_{01} & \widehat{A}_{02} \\ \hline l_{10}^T & \upsilon_{11} & \widehat{a}_{12}^T \\ \hline \widehat{A}_{20} & \widehat{a}_{21} & \widehat{A}_{22} \end{array}$$

where the entries in blue are to be computed. Now, considering $LU = \widehat{A}$ we notice that

$$\begin{array}{c|c|c} L_{00}U_{00} = \widehat{A}_{00} & L_{00}u_{01} = \widehat{a}_{01} & L_{00}U_{02} = \widehat{A}_{02} \\ \hline l_{10}^T U_{00} = \widehat{a}_{10}^T & l_{10}^T u_{01} + \upsilon_{11} = \widehat{\alpha}_{11} & l_{10}^T U_{02} + u_{12}^T = \widehat{a}_{12}^T \\ \hline L_{20}U_{00} = \widehat{A}_{20} & L_{20}u_{01} + \upsilon_{11}l_{21} = \widehat{a}_{21} & L_{20}U_{02} + l_{21}u_{12}^T + L_{22}U_{22} = \widehat{A}_{22} \end{array}$$

where the entries in red are already known. The equalities in yellow can be used to compute the desired parts of L and U:

- Solve $L_{00}u_{01} = a_{01}$ for u_{01}, overwriting a_{01} with the result.
- Solve $l_{10}^T U_{00} = a_{10}^T$ (or, equivalently, $U_{00}^T (l_{10}^T)^T = (a_{10}^T)^T$ for l_{10}^T), overwriting a_{10}^T with the result.
- Compute $\upsilon_{11} = \alpha_{11} - l_{10}^T u_{01}$, overwriting α_{11} with the result.

Exercise 15. *If A is an $n \times n$ matrix, show that the cost of Variant 1 is approximately $\frac{2}{3}n^3$ flops.*

8.2 Variant 2: Left-looking algorithm

Consider the loop invariant:

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) = \left(\begin{array}{c|c} L\backslash U_{TL} & \widehat{A}_{TR} \\ \hline L_{BL} & \widehat{A}_{BR} \end{array}\right)$$
$$\wedge \begin{array}{c|c} L_{TL}U_{TL} = \widehat{A}_{TL} & \cancel{L_{TL}U_{TR} = \widehat{A}_{TR}} \\ \hline L_{BL}U_{TL} = \widehat{A}_{BL} & \cancel{L_{BL}U_{TR} + L_{BR}U_{BR} = \widehat{A}_{BR}} \end{array}$$

At the top of the loop, after repartitioning, A contains

$$\begin{array}{c|c|c} L\backslash U_{00} & \widehat{a}_{01} & \widehat{A}_{02} \\ \hline l_{10}^T & \widehat{\alpha}_{11} & \widehat{a}_{12}^T \\ \hline L_{20} & \widehat{a}_{21} & \widehat{A}_{22} \end{array}$$

while at the bottom it must contain

$L\backslash U_{00}$	u_{01}	$\widehat{A}_{02}$
l_{10}^T	υ_{11}	$\widehat{a}_{12}^T$
L_{20}	l_{21}	$\widehat{A}_{22}$

where the entries in blue are to be computed. Now, considering $LU = \widehat{A}$ we notice that

$L_{00}U_{00} = \widehat{A}_{00}$	$L_{00}u_{01} = \widehat{a}_{01}$	$L_{00}U_{02} = \widehat{A}_{02}$
$l_{10}^T U_{00} = \widehat{a}_{10}^T$	$l_{10}^T u_{01} + \upsilon_{11} = \widehat{\alpha}_{11}$	$l_{10}^T U_{02} + u_{12}^T = \widehat{a}_{12}^T$
$L_{20}U_{00} = \widehat{A}_{20}$	$L_{20}u_{01} + \upsilon_{11}l_{21} = \widehat{a}_{21}$	$L_{20}U_{02} + l_{21}u_{12}^T + L_{22}U_{22} = \widehat{A}_{22}$

The equalities in yellow can be used to compute the desired parts of L and U:

- Solve $L_{00}u_{01} = a_{01}$ for u_{01}, overwriting a_{01} with the result.
- Compute $\upsilon_{11} = \alpha_{11} - l_{10}^T u_{01}$, overwriting α_{11} with the result.
- Compute $l_{21} := (\alpha_{21} - L_{20}u_{01})/\upsilon_{11}$, overwriting a_{21} with the result.

8.3 Variant 3: Up-looking variant

Exercise 16. *Derive the up-looking variant for computing the LU factorization.*

8.4 Variant 4: Crout variant

Consider the loop invariant:

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) = \left(\begin{array}{c|c} L\backslash U_{TL} & U_{TR} \\ \hline L_{BL} & \widehat{A}_{BR} \end{array}\right)$$
$$\wedge \begin{array}{c|c} L_{TL}U_{TL} = \widehat{A}_{TL} & L_{TL}U_{TR} = \widehat{A}_{TR} \\ \hline L_{BL}U_{TL} = \widehat{A}_{BL} & L_{BL}U_{TR} + L_{BR}U_{BR} = \widehat{A}_{BR} \end{array}$$

At the top of the loop, after repartitioning, A contains

$L\backslash U_{00}$	u_{01}	U_{02}
l_{10}^T	$\widehat{\alpha}_{11}$	$\widehat{a}_{12}^T$
L_{20}	$\widehat{a}_{21}$	$\widehat{A}_{22}$

while at the bottom it must contain

$L\backslash U_{00}$	u_{01}	U_{02}
l_{10}^T	υ_{11}	u_{12}^T
L_{20}	l_{21}	$\widehat{A}_{22}$

where the entries in blue are to be computed. Now, considering $LU = \widehat{A}$ we notice that

$L_{00}U_{00} = \widehat{A}_{00}$	$L_{00}u_{01} = \widehat{a}_{01}$	$L_{00}U_{02} = \widehat{A}_{02}$
$l_{10}^T U_{00} = \widehat{a}_{10}^T$	$l_{10}^T u_{01} + \upsilon_{11} = \widehat{\alpha}_{11}$	$l_{10}^T U_{02} + u_{12}^T = \widehat{a}_{12}^T$
$L_{20}U_{00} = \widehat{A}_{20}$	$L_{20}u_{01} + \upsilon_{11}l_{21} = \widehat{a}_{21}$	$L_{20}U_{02} + l_{21}u_{12}^T + L_{22}U_{22} = \widehat{A}_{22}$

The equalities in yellow can be used to compute the desired parts of L and U:

Algorithm: $A := L\backslash U = \mathrm{LU}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$

where A_{TL} is 0×0

while $n(A_{TL}) < n(A)$ **do**

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right)$$

where α_{11} is 1×1

A_{00} contains L_{00} and U_{00} in its strictly lower and upper triangular part, respectively.

Variant 1 Bordered	Variant 2 Left-looking	Variant 3 Up-looking	Variant 4 Crout variant	Variant 5 Classical LU
$a_{01} := L_{00}^{-1} a_{01}$	$a_{01} := L_{00}^{-1} a_{01}$	Exercise in 8.3		
$a_{10}^T := a_{10}^T U_{00}^{-1}$				
$\alpha_{11} := \alpha_{11} - a_{21}^T a_{01}$	$\alpha_{11} := \alpha_{11} - a_{21}^T a_{01}$		$\alpha_{11} := \alpha_{11} - a_{21}^T a_{01}$	
			$a_{12}^T := a_{12}^T - a_{10}^T A_{02}$	
	$a_{21} := a_{21} - A_{20} a_{01}$		$a_{21} := a_{21} - A_{20} a_{01}$	
	$a_{21} := a_{21}/\alpha_{11}$		$a_{21} := a_{21}/\alpha_{11}$	$a_{21} := a_{21}/\alpha_{11}$
				$A_{22} := A_{22} - a_{21} a_{12}^T$

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right)$$

endwhile

Figure 9: All five LU factorization algorithms.

- Compute $\upsilon_{11} = \alpha_{11} - l_{10}^T u_{01}$, overwriting α_{11} with the result.
- Compute $l_{21} := (\alpha_{21} - L_{20} u_{01})/\upsilon_{11}$, overwriting a_{21} with the result.
- Compute $u_{12}^T := a_{12}^T - l_{10}^T U_{02}$, overwriting a_{12}^T with the result.

8.5 Variant 5: Classical LU factorization

We have already derived this algorithm. You may want to try rederiving it using the techniques discussed in this section.

8.6 All algorithms

All five algorithms for LU factorization are summarized in Figure 9.

Exercise 17. *Implement all five LU factorization algorithms with the FLAME@lab API, in M-script.*

Exercise 18. *Which of the five variants can be modified to incorporate partial pivoting?*

8.7 Formal derivation of algorithms

The described approach to deriving algorithms, linking the process to the *a priori* identification of loop invariants, was first proposed in [10]. It was refined into what we call the "worksheet" for deriving algorithms hand-in-hand with their proofs of correctness, in [3]. A book that describes the process at a level also appropriate for the novice is "The Science of Programming Matrix Computations" [14].

9 Numerical Stability Results

The numerical stability of various LU factorization algorithms as well as the triangular solve algorithms can be found in standard graduate level numerical linear algebra texts and references [8, 11, 13]. Of particular interest may be the analysis of the Crout variant (Variant 4) in [5], since it uses our notation as well as the results in "Notes on Numerical Stability". (We recommend the technical report version [4] of the paper, since it has more details as well as exercises to help the reader understand.) In that paper, a systematic approach towards the derivation of backward error results is given that mirrors the systematic approach to deriving the algorithms given in [10, 3, 14].

Here are pertinent results from that paper, assuming floating point arithmetic obeys the model of computation given in "Notes on Numerical Stability" (as well as [5, 4, 11]). It is assumed that the reader is familiar with those notes.

Theorem 19. *Let $A \in \mathbb{R}^{n\times n}$ and let the LU factorization of A be computed via the Crout variant (Variant 4), yielding approximate factors $\check{L}$ and $\check{U}$. Then*

$$(A + \Delta A) = \check{L}\check{U} \quad \textit{with} \quad |\Delta A| \leq \gamma_n |\check{L}||\check{U}|.$$

Theorem 20. *Let $L \in \mathbb{R}^{n\times n}$ be lower triangular and $y, z \in \mathbb{R}^n$ with $Lz = y$. Let $\check{z}$ be the approximate solution that is computed. Then*

$$(L + \Delta L)\check{z} = y \quad \textit{with} \quad |\Delta L| \leq \gamma_n |L|.$$

Theorem 21. *Let $U \in \mathbb{R}^{n\times n}$ be upper triangular and $x, z \in \mathbb{R}^n$ with $Ux = z$. Let $\check{x}$ be the approximate solution that is computed. Then*

$$(U + \Delta U)\check{x} = z \quad \textit{with} \quad |\Delta U| \leq \gamma_n |U|.$$

Theorem 22. *Let $A \in \mathbb{R}^{n\times n}$ and $x, y \in \mathbb{R}^n$ with $Ax = y$. Let $\check{x}$ be the approximate solution computed via the following steps:*

- *Compute the LU factorization, yielding approximate factors $\check{L}$ and $\check{U}$.*
- *Solve $\check{L}z = y$, yielding approximate solution $\check{z}$.*
- *Solve $\check{U}x = \check{z}$, yielding approximate solution $\check{x}$.*

Then $(A + \Delta A)\check{x} = y \quad \textit{with} \quad |\Delta A| \leq (3\gamma_n + \gamma_n^2)|\check{L}||\check{U}|$.

The analysis of LU factorization without partial pivoting is related that of LU factorization with partial pivoting. We have shown that LU with partial pivoting is equivalent to the LU factorization without partial pivoting on a pre-permuted matrix: $PA = LU$, where P is a permutation matrix. The permutation doesnt involve any floating point operations and therefore does not generate error. It can therefore be argued that, as a result, the error that is accumulated is equivalent with or without partial pivoting

10 Is LU with Partial Pivoting Stable?

Exercise 23. *Apply LU with partial pivoting to*

$$A = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 1 \\ -1 & 1 & 0 & \cdots & 0 & 1 \\ -1 & -1 & 1 & \cdots & 0 & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ -1 & -1 & & \cdots & 1 & 1 \\ -1 & -1 & & \cdots & -1 & 1 \end{pmatrix}.$$

Pivot only when necessary.

From this exercise we conclude that even LU factorization with partial pivoting can yield large (exponential) element growth in U. You may enjoy the collection of problems for which Gaussian elimination with partial pivoting in unstable by Stephen Wright [15].

In practice, this does not seem to happen and LU factorization is considered to be stable.

11 Blocked algorithms

It is well-known that matrix-matrix multiplication can achieve high performance on most computer architectures [1, 9, 7]. As a result, many dense matrix algorithms are reformulated to be rich in matrix-matrix multiplication operations. An interface to a library of such operations is known as the level-3 Basic Linear Algebra Subprograms (BLAS) [6]. In this section, we show how LU factorization can be rearranged so that most computation is in matrix-matrix multiplications.

11.1 Blocked classical LU factorization (Variant 5)

Partition A, L, and U as follows:

$$A \rightarrow \left(\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right), \quad L \rightarrow \left(\begin{array}{c|c} L_{11} & 0 \\ \hline L_{21} & L_{22} \end{array}\right), \quad \text{and} \quad U \rightarrow \left(\begin{array}{c|c} U_{11} & U_{12} \\ \hline 0 & U_{22} \end{array}\right),$$

where A_{11}, L_{11}, and U_{11} are $b \times b$. Then $A = LU$ means that

$$\left(\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right) = \left(\begin{array}{c|c} L_{11} & 0 \\ \hline L_{21} & L_{22} \end{array}\right)\left(\begin{array}{c|c} U_{11} & U_{12} \\ \hline 0 & U_{22} \end{array}\right) = \left(\begin{array}{c|c} L_{11}U_{11} & L_{11}U_{12} \\ \hline L_{21}U_{11} & L_{21}U_{12} + L_{22}U_{22} \end{array}\right).$$

This means that

$$\begin{array}{c|c} A_{11} = L_{11}U_{11} & A_{12} = L_{11}U_{12} \\ \hline A_{21} = L_{21}U_{11} & A_{22} = L_{21}U_{12} + L_{22}U_{22} \end{array}$$

or, equivalently,

$$\begin{array}{c|c} A_{11} = L_{11}U_{11} & A_{12} = L_{11}U_{12} \\ \hline A_{21} = L_{21}U_{11} & A_{22} - L_{21}U_{12} = L_{22}U_{22} \end{array}.$$

If we let L and U overwrite the original matrix A this suggests the algorithm

- Compute the LU factorization $A_{11} = L_{11}U_{11}$, overwriting A_{11} with $L\backslash U_{11}$. Notice that any of the "unblocked" algorithms previously discussed in this note can be used for this factorization.

Algorithm: $A := \mathrm{LU}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$

where A_{TL} is 0×0

while $m(A_{TL}) < m(A)$ **do**

Determine block size b

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

where A_{11} **is** $b \times b$

A_{00} contains L_{00} and U_{00} in its strictly lower and upper triangular part, respectively.

Variant 1:	Variant 2:	Variant 3:
$A_{01} := {L_{00}}^{-1} A_{01}$	$A_{01} := {L_{00}}^{-1} A_{01}$	$A_{10} := A_{10} {U_{00}}^{-1}$
$A_{10} := A_{10} {U_{00}}^{-1}$	$A_{11} := A_{11} - A_{10} A_{01}$	$A_{11} := A_{11} - A_{10} A_{01}$
$A_{11} := A_{11} - A_{10} A_{01}$	$A_{11} := \mathrm{LU}(A_{11})$	$A_{11} := \mathrm{LU}(A_{11})$
$A_{11} := \mathrm{LU}(A_{11})$	$A_{21} := (A_{21} - A_{20} A_{01}) {U_{11}}^{-1}$	$A_{12} := A_{12} - A_{10} A_{02}$

Variant 4:	Variant 5:
$A_{11} := A_{11} - A_{10} A_{01}$	$A_{11} := \mathrm{LU}(A_{11})$
$A_{11} := \mathrm{LU}(A_{11})$	$A_{21} := A_{21} {U_{11}}^{-1}$
$A_{21} := (A_{21} - A_{20} A_{01}) U_{11}^{-1}$	$A_{12} := {L_{11}}^{-1} A_{12}$
$A_{12} := {L_{11}}^{-1} (A_{12} - A_{10} A_{02})$	$A_{22} := A_{22} - A_{21} A_{12}$

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

endwhile

Figure 10: Blocked algorithms for computing the LU factorization.

- Solve $L_{11} U_{12} = A_{12}$, overwriting A_{12} with U_{12}. (This can also be expressed as $A_{12} := L_{11}^{-1} A_{12}$.)
- Solve $L_{21} U_{11} = A_{21}$, overwiting A_{21} with U_{21}. (This can also be expressed as $A_{21} := A_{21} U_{11}^{-1}$.)
- Update $A_{22} := A_{22} - A_{21} A_{12}$.
- Continue by overwriting the updated A_{22} with its LU factorization.

If b is small relative to n, then most computation is in the last step, which is a matrix-matrix multiplication. Similarly, blocked algorithms for the other variants can be derived. All are given in Figure 10.

11.2 Blocked classical LU factorization with pivoting (Variant 5)

Pivoting can be added to some of the blocked algorithms. Let us focus once again on Variant 5.

Algorithm: $[A, p] := \text{LUPIV\_BLK}(A, p)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$, $p \rightarrow \left(\begin{array}{c} p_T \\ \hline p_B \end{array}\right)$

where A_{TL} is 0×0, p_T has 0 elements.

while $n(A_{TL}) < n(A)$ **do**

Determine block size b

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right), \left(\begin{array}{c} p_T \\ \hline p_B \end{array}\right) \rightarrow \left(\begin{array}{c} p_0 \\ \hline p_1 \\ \hline p_2 \end{array}\right)$$

where A_{11} **is** $b \times b$**,** p_1 **has** b **elements**

Variant 2:

$A_{01} := L_{00}^{-1} A_{01}$

$A_{11} := A_{11} - A_{10}A_{01}$

$A_{21} := A_{21} - A_{20}A_{01}$

$$\left[\left(\begin{array}{c} A_{11} \\ \hline A_{21} \end{array}\right), p_1\right] := \text{LUPIV}\left(\left(\begin{array}{c} A_{11} \\ \hline A_{21} \end{array}\right), p_1\right)$$

$$\left(\begin{array}{c|c} A_{10} & A_{12} \\ \hline A_{20} & A_{22} \end{array}\right) := P(p_1)\left(\begin{array}{c|c} A_{10} & A_{12} \\ \hline A_{20} & A_{22} \end{array}\right)$$

Variant 4:

$A_{11} := A_{11} - A_{10}A_{01}$

$A_{21} := A_{21} - A_{20}A_{01}$

$$\left[\left(\begin{array}{c} A_{11} \\ \hline A_{21} \end{array}\right), p_1\right] := \text{LUPIV}\left(\left(\begin{array}{c} A_{11} \\ \hline A_{21} \end{array}\right), p_1\right)$$

$$\left(\begin{array}{c|c} A_{10} & A_{12} \\ \hline A_{20} & A_{22} \end{array}\right) := P(p_1)\left(\begin{array}{c|c} A_{10} & A_{12} \\ \hline A_{20} & A_{22} \end{array}\right)$$

$A_{12} := A_{12} - A_{10}A_{02}$

$A_{12} := L_{11}^{-1} A_{12}$

Variant 5:

$$\left[\left(\begin{array}{c} A_{11} \\ \hline A_{21} \end{array}\right), p_1\right] := \text{LUPIV}\left(\left(\begin{array}{c} A_{11} \\ \hline A_{21} \end{array}\right), p_1\right)$$

$$\left(\begin{array}{c|c} A_{10} & A_{12} \\ \hline A_{20} & A_{22} \end{array}\right) := P(p_1)\left(\begin{array}{c|c} A_{10} & A_{12} \\ \hline A_{20} & A_{22} \end{array}\right)$$

$A_{12} := L_{11}^{-1} A_{12}$

$A_{22} := A_{22} - A_{21}A_{12}$

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right), \left(\begin{array}{c} p_T \\ \hline p_B \end{array}\right) \leftarrow \left(\begin{array}{c} p_0 \\ \hline p_1 \\ \hline p_2 \end{array}\right)$$

endwhile

Figure 11: Blocked algorithms for computing the LU factorization with partial pivoting.

.

Partition A, L, and U as follows:

$$A \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right), \quad L \rightarrow \left(\begin{array}{c|c|c} L_{00} & 0 & 0 \\ \hline L_{10} & L_{11} & 0 \\ \hline L_{20} & L_{21} & L_{22} \end{array}\right), \quad \text{and} \quad U \rightarrow \left(\begin{array}{c|c|c} U_{00} & U_{01} & U_{02} \\ \hline 0 & U_{11} & U_{12} \\ \hline 0 & 0 & U_{22} \end{array}\right),$$

where A_{00}, L_{00}, and U_{00} are $k \times k$, and A_{11}, L_{11}, and U_{11} are $b \times b$.

Assume that the computation has proceeded to the point where A contains

$$\left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right) = \left(\begin{array}{c|c|c} L\backslash U_{00} & U_{01} & U_{02} \\ \hline L_{10} & \widehat{A}_{11} - L_{10}U_{01} & A_{12} - L_{10}U_{02} \\ \hline L_{20} & \widehat{A}_{21} - L_{20}U_{01} & A_{22} - L_{20}U_{02} \end{array}\right),$$

where, as before, $\widehat{A}$ denotes the original contents of A and

$$P(p_0)\left(\begin{array}{c|c|c} \widehat{A}_{00} & \widehat{A}_{01} & \widehat{A}_{02} \\ \hline \widehat{A}_{10} & \widehat{A}_{11} & \widehat{A}_{12} \\ \hline \widehat{A}_{20} & \widehat{A}_{21} & \widehat{A}_{22} \end{array}\right) = \left(\begin{array}{c|c|c} L_{00} & 0 & 0 \\ \hline L_{10} & I & 0 \\ \hline L_{20} & 0 & I \end{array}\right)\left(\begin{array}{c|c|c} U_{00} & U_{01} & U_{02} \\ \hline 0 & A_{11} & A_{12} \\ \hline 0 & A_{21} & A_{22} \end{array}\right).$$

In the current blocked step, we now perform the following computations

- Compute the LU factorization with pivoting of the "current panel" $\left(\begin{array}{c} A_{11} \\ \hline A_{21} \end{array}\right)$:

  $$P(p_1)\left(\begin{array}{c} A_{11} \\ \hline A_{21} \end{array}\right) = \left(\begin{array}{c} L_{11} \\ \hline L_{21} \end{array}\right)U_{11},$$

  overwriting A_{11} with $L\backslash U_{11}$ and A_{21} with L_{21}.

- Correspondingly, swap rows in the remainder of the matrix

  $$\left(\begin{array}{c|c} A_{10} & A_{12} \\ \hline A_{20} & A_{22} \end{array}\right) := \qquad P(p_1)\left(\begin{array}{c|c} A_{10} & A_{12} \\ \hline A_{20} & A_{22} \end{array}\right).$$

- Solve $L_{11}U_{12} = A_{12}$, overwriting A_{12} with U_{12}. (This can also be more concisely written as $A_{12} := L_{11}^{-1}A_{12}$.)

- Update $A_{22} := A_{22} - A_{21}A_{12}$.

Careful consideration shows that this puts the matrix A in the state

$$\left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right) = \left(\begin{array}{c|c|c} L\backslash U_{00} & U_{01} & U_{02} \\ \hline L_{10} & L\backslash U_{11} & U_{12} \\ \hline L_{20} & L_{21} & \widehat{A}_{22} - L_{20}U_{02} - L_{21}U_{12} \end{array}\right),$$

where

$$P(\left(\begin{array}{c} p_0 \\ \hline p_1 \end{array}\right))\left(\begin{array}{c|c|c} \widehat{A}_{00} & \widehat{A}_{01} & \widehat{A}_{02} \\ \hline \widehat{A}_{10} & \widehat{A}_{11} & \widehat{A}_{12} \\ \hline \widehat{A}_{20} & \widehat{A}_{21} & \widehat{A}_{22} \end{array}\right) = \left(\begin{array}{c|c|c} L_{00} & 0 & 0 \\ \hline L_{10} & L_{11} & 0 \\ \hline L_{20} & L_{21} & I \end{array}\right)\left(\begin{array}{c|c|c} U_{00} & U_{01} & U_{02} \\ \hline 0 & U_{11} & U_{12} \\ \hline 0 & 0 & A_{22} \end{array}\right).$$

Similarly, blocked algorithms with pivoting for some of the other variants can be derived. All are given in Figure 10.

12 Variations on a Triple-Nested Loop

All LU factorization algorithms presented in this note perform exactly the same floating point operations (with some rearrangement of data thrown in for the algorithms that perform pivoting) as does the triple-nested loop that implements Gaussian elimination:

```
for j = 0, ..., n − 1                          (zero the elements below (j, j) element)
    for i = j + 1, ... n − 1
        α_{i,j} := α_{i,j} / α_{j,j}           (compute multiplier λ_{i,j}, overwriting α_{i,j})
        for k = j + 1, ..., n − 1              (subtract λ_{i,j} times the jth row from ith row)
            α_{i,k} := α_{i,k} − α_{i,j} α_{j,k}
        endfor
    endfor
endfor
```

References

[1] E. Anderson, Z. Bai, J. Demmel, J. E. Dongarra, J. DuCroz, A. Greenbaum, S. Hammarling, A. E. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users' Guide*. SIAM, Philadelphia, 1992.

[2] Paolo Bientinesi. *Mechanical Derivation and Systematic Analysis of Correct Linear Algebra Algorithms*. PhD thesis, 2006.

[3] Paolo Bientinesi, John A. Gunnels, Margaret E. Myers, Enrique S. Quintana-Ortí, and Robert A. van de Geijn. The science of deriving dense linear algebra algorithms. *ACM Trans. Math. Soft.*, 31(1):1–26, March 2005.

[4] Paolo Bientinesi and Robert A. van de Geijn. The science of deriving stability analyses. FLAME Working Note #33. Technical Report AICES-2008-2, Aachen Institute for Computational Engineering Sciences, RWTH Aachen, November 2008.

[5] Paolo Bientinesi and Robert A. van de Geijn. Goal-oriented and modular stability analysis. *SIAM J. Matrix Anal. Appl.*, 32(1):286–308, March 2011. We suggest you read FLAME Working Note #33 for more details.

[6] Jack J. Dongarra, Jeremy Du Croz, Sven Hammarling, and Iain Duff. A set of level 3 basic linear algebra subprograms. *ACM Trans. Math. Soft.*, 16(1):1–17, March 1990.

[7] Jack J. Dongarra, Iain S. Duff, Danny C. Sorensen, and Henk A. van der Vorst. *Solving Linear Systems on Vector and Shared Memory Computers*. SIAM, Philadelphia, PA, 1991.

[8] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, 3rd edition, 1996.

[9] Kazushige Goto and Robert van de Geijn. Anatomy of high-performance matrix multiplication. *ACM Trans. Math. Soft.*, 34(3):12:1–12:25, May 2008.

[10] John A. Gunnels, Fred G. Gustavson, Greg M. Henry, and Robert A. van de Geijn. FLAME: Formal linear algebra methods environment. *ACM Trans. Math. Soft.*, 27(4):422–455, December 2001.

[11] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second edition, 2002.

[12] Margaret E. Myers, Pierce M. van de Geijn, and Robert A. van de Geijn. *Linear Algebra: Foundations to Frontiers - Notes to LAFF With*. Self published, 2014. Available from `http://www.ulaff.net`.

[13] G. W. Stewart. *Matrix Algorithms Volume 1: Basic Decompositions*. SIAM, 1998.

[14] Robert A. van de Geijn and Enrique S. Quintana-Ortí. *The Science of Programming Matrix Computations*. `www.lulu.com/contents/contents/1911788/`, 2008.

[15] Stephen J. Wright. A collection of problems for which Gaussian elimination with partial pivoting is unstable. *SIAM J. Sci. Comput.*, 14(1):231–238, 1993.