CHAPTER 1

Gaussian elimination

1.1. Preliminaries

We will deal with matrices having entries in a field $\mathbb{F}$ that can be either the field of real numbers $\mathbb{R}$ or that of complex numbers $\mathbb{C}$. The set of $m \times n$ matrices with m rows and n columns with entries in $\mathbb{F}$ is denoted by

$$\mathbb{F}^{m\times n}.$$

The (i, j)-entry of an $m \times n$ matrix A is denoted by $a_{i,j}$, so that

$$A = \begin{bmatrix} a_{1,1} & \dots & a_{1,n} \\ \vdots & & \vdots \\ a_{m,1} & \dots & a_{m,n} \end{bmatrix} \quad \text{with } a_{i,j} \in \mathbb{F}.$$

Similarly, the i-th entry of an n-vector x is denoted by x_i, so that $x = (x_1, \dots, x_n)$. By default, these vectors will be identified with $n \times 1$ matrices:

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}. \tag{1.1}$$

The *rank* of the matrix A, denoted by $\operatorname{rank}(A)$, is the maximal number of linearly independent vectors among the columns

$$\operatorname{col}_j(A) \in \mathbb{F}^m, \quad j = 1, \dots, n.$$

Equivalently, it can be defined as the dimension of the linear subspace of $\mathbb{F}^m$ generated by these columns. The matrix is said to have *full rank* if $\operatorname{rank}(A) = m$.

Multiplying A by an $n \times 1$ matrix gives an $m \times 1$ matrix, and so the identification in (1.1) allows to consider the linear map

$$L_A \colon \mathbb{F}^n \longrightarrow \mathbb{F}^m \tag{1.2}$$

defined by $L_A(x) = A\,x$. For instance, for $A = \begin{bmatrix} 2 & -1 & 3 \\ -5 & 0 & 7 \end{bmatrix}$ we have that

$$A \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2x_1 - x_2 + 3x_2 \\ -5x_1 + 7x_3 \end{bmatrix}$$

and so $L_A(x_1, x_2, x_3) = (2x_1 - x_2 + 3x_2, -5x_1 + 7x_3)$.

The *image* of L_A is the subset vectors in $\mathbb{F}^m$ that are images of vectors in $\mathbb{F}^n$, whereas its *kernel* is the set of vectors in $\mathbb{F}^n$ where this linear map vanishes, that is

$$\operatorname{Im}(L_A) = \{L_A(x) \mid x \in \mathbb{F}^n\} \subset \mathbb{F}^m \quad \text{and} \quad \operatorname{Ker}(L_A) = \{x \in \mathbb{F}^n \mid L_A(x) = 0\} \subset \mathbb{F}^n.$$

A fundamental result in linear algebra, the *rank-nullity theorem*, says that the dimensions of these linear subspaces are complementary in the sense that

$$\dim(\mathrm{Im}(L_A)) + \dim(\mathrm{Ker}(L_A)) = n. \tag{1.3}$$

The image of L_A coincides with the *column space* of the matrix, that is, the linear span of its columns

$$\mathrm{span}(\mathrm{col}_1(A), \ldots, \mathrm{col}_n(A)),$$

and so its dimension coincides with the rank of the matrix, that is $\dim(L_A) = \mathrm{rank}(A)$.

An $r \times s$ *block matrix* is an $m \times n$ matrix written as

$$A = \begin{bmatrix} A_{1,1} & \cdots & A_{1,s} \\ \vdots & & \vdots \\ A_{r,1} & \cdots & A_{r,s} \end{bmatrix} \in \mathbb{F}^{m \times n}$$

with *blocks* $A_{i,j}$ that are $m_i \times n_j$ matrices, for given sequences of nonnegative integers m_i, n_j, $i = 1, \ldots, r$, $j = 1, \ldots, s$, such that

$$\sum_{i=1}^{r} m_i = m \quad \text{and} \quad \sum_{j=1}^{s} n_j = n.$$

We can also say that A is an $(m_1, \ldots, m_r) \times (n_1, \ldots, n_s)$ *block matrix*, specially when we want to precise the sizes of the blocks.

Block matrices with corresponding blocks of equal sizes can be added through the block structure: if

$$B = \begin{bmatrix} B_{1,1} & \cdots & B_{1,r} \\ \vdots & & \vdots \\ B_{s,1} & \cdots & B_{r,s} \end{bmatrix}$$

is also an $(m_1, \ldots, m_r) \times (n_1, \ldots, n_s)$ block matrix, then

$$A + B = [A_{i,j} + B_{i,j}]_{i,j} \quad \text{with } 1 \le i \le r, 1 \le j \le s.$$

Similarly, block matrices with blocks of compatible sizes can be multiplied in the natural way: for an $n \times p$ matrix written as

$$C = \begin{bmatrix} C_{1,1} & \cdots & C_{1,t} \\ \vdots & & \vdots \\ C_{s,1} & \cdots & C_{s,t} \end{bmatrix},$$

where each $C_{j,k}$ is an $n_j \times p_k$ matrix for another sequence of nonnegative integers p_k, $k = 1, \ldots, t$, with $\sum_{k=1}^{t} p_k = p$, we have that

$$A\,C = \left[\sum_{j=1}^{s} A_{i,j}\, C_{j,k} \right]_{i,k} \quad \text{with } 1 \le i \le r, 1 \le k \le t,$$

which an $r \times t$ block matrix with blocks of sizes $m_i \times p_k$, $i = 1, \ldots, r$, $k = 1, \ldots, t$.

Beware that not every operation or computation can be directly rewritten using blocks. For instance, the determinant of a block matrix cannot be computed with the usual formula applied to the blocks, so that if

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}$$

then in general $\det(A) \neq \det(A_{1,1})\det(A_{2,2}) - \det(A_{2,1})\det(A_{1,2})$. Nevertheless, this formula does hold when the matrix is block triangular: if $A_{2,1} = \mathbb{0}$ then

$$\det(A) = \det(A_{1,1})\det(A_{2,2}).$$

1.2. From Gaussian elimination to the PLU factorization

The first and most fundamental problem of numerical linear algebra is to solve the equation

$$A\,x = b \tag{1.4}$$

for a given $m \times n$ matrix A and m-vector b in terms of an unknown n-vector x, all of them with entries in the field $\mathbb{F}$.

In general, this equation can have either a single solution, an infinite number of them, or none at all. Indeed, it admits a solution if and only if b belongs to the image of the linear map L_A in (1.2), that is $b \in \mathrm{Im}(L_A)$. Assuming that such a solution x exists, then any other solution x' can be obtained by adding an element of the kernel of L_A, that is $x' = x + y$ with $y \in \mathrm{Ker}(A)$. Hence when a solution exists, it is unique if and only if this kernel is trivial, that is $\mathrm{Ker}(L_A) = \{0\}$.

Thus, the equation (1.4) has a unique solution for every $b \in \mathbb{F}^m$ if and only if

$$\mathrm{Im}(L_A) = \mathbb{F}^m \quad \text{and} \quad \mathrm{Ker}(L_A) = \{0\}$$

or equivalently, if and only if $\dim(\mathrm{Im}(L_A)) = n$ and $\dim(\mathrm{Ker}(L_A)) = 0$. By the formula in (1.3), these conditions are equivalent to the fact that $m = n$ and that A has full rank.

When $m = n$, the condition that $\mathrm{rank}(A) = n$ is equivalent to A being *nonsingular*. This means that A admits an *inverse*, that is, there is an $n \times n$ matrix A^{-1} such that $A\,A^{-1} = A^{-1}A = \mathbb{1}_n$ with

$$\mathbb{1}_n = \begin{bmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{bmatrix},$$

the *identity* $n \times n$ matrix. Hence

Equation (1.4) has a unique solution for all $b \iff$ $m = n$ and A is nonsingular

This is the main case of interest, and the one that we are going to consider in this part. Of course, in this situation the solution of the equation can be written as

$$x = A^{-1}b. \tag{1.5}$$

However, this expression does not give an efficient way to obtain this solution, since computing the inverse is too expensive, as we will see later. Instead, we will consider more efficient methods for this task, both direct and iterative.

Gaussian elimination is the basis of most direct methods for solving the equation (1.4). We next recall how its standard version works in a simple example.

Example 1.2.1. Consider the system of linear equations

$$\begin{cases} x_1 + 2\,x_2 = b_1 \\ 3\,x_1 + 4\,x_2 = b_2 \end{cases}$$

Since the coefficient of the variable x_1 in the first equation is $a_{1,1} = 1$ and the corresponding coefficient in the second equation is $a_{2,1} = 3$, we have to subtract the

multiple $a_{2,1}/a_{1,1} = 3$ of the first equation to the second, to eliminate x_1 in this second equation. We obtain

$$\begin{cases} x_1 + 2\,x_2 = b_1 \\ \quad -2\,x_2 = -3\,b_1 + b_2 \end{cases}$$

Next we solve the modified system of linear equations by *backward substitution*: we first compute the value of x_2 using the last equation and then plug it into the first to get the value of x_1:

$$x_2 = \frac{1}{-2}(-3\,b_1 + b_2) = \frac{3}{2}\,b_1 - \frac{1}{2}\,b_2 \quad \text{and} \quad x_1 = b_1 - 2\,x_2 = -2\,b_1 + b_2.$$

In matrix notation, the system of linear equations in Example 1.2.1 writes down as

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix},$$

and the elimination procedure therein is equivalent to a left multiplication of both sides of this vector equation by a lower triangular matrix:

$$\begin{bmatrix} 1 & 0 \\ -3 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -3 & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \tag{1.6}$$

which gives the equivalent equation $\begin{bmatrix} 1 & 2 \\ 0 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ -3\,b_1 + b_2 \end{bmatrix}$. Setting

$$A = \begin{bmatrix} 1 & 2 \\ 0 & -2 \end{bmatrix}, \quad L = \begin{bmatrix} 1 & 0 \\ 3 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 1 & 2 \\ 0 & -2 \end{bmatrix},$$

we have that $L^{-1} = \begin{bmatrix} 1 & 0 \\ -3 & 1 \end{bmatrix}$ and so the above procedure implies that $L^{-1}A = U$, or equivalently the *matrix factorization*

$$A = L\,U.$$

Notice that L is *unit lower triangular* (all its diagonal entries are 1 and all its superdiagonals entries are 0) and U is *upper triangular* (all its subdiagonals entries are 0).

In Example 1.2.1, the quantities $a_{1,1}$ and $a_{1,2}/a_{1,1}$ in the elimination step are respectively called the *pivot* and the *multiplier*. In general, Gaussian elimination will perform successive elimination steps to bring the system of equations to a triangular one. At each step, we need the pivot to be nonzero to be able to define the corresponding multiplier

Hence when some pivot vanishes, we modify Gaussian elimination adding to it a *pivoting strategy*, permuting the equations so that the algorithm can continue eliminating variables, as in the next example.

Example 1.2.2. Consider the system of linear equations

$$\begin{cases} 2\,x_1 - x_2 = b_1 \\ 2\,x_1 - x_2 + x_3 = b_2 \\ -2\,x_1 + 3\,x_2 - x_3 = b_3 \end{cases}$$

Gaussian elimination proceeds subtracting the first equation to the second and adding the first equation to the third, thus eliminating the variable x_1 from both of them.

In the resulting system, the coefficient of x_2 in the second equation is 0, and so the algorithm permutes this second equation with the third, to obtain a system in *row echelon form*:

$$\begin{cases} 2\,x_1 - \;\; x_2 \qquad\;\; = b_1 \\ \qquad\;\; 2\,x_2 - x_3 = b_1 + b_3 \\ \qquad\qquad\;\; x_3 = -b_1 + b_2 \end{cases}$$

As before, this system can be easily solved by backward substitution:

$$x_3 = -b_1 + b_2, \quad x_2 = \frac{1}{2}(b_1 + b_3 + x_3) = \frac{1}{2}\,b_2 + \frac{1}{2}\,b_3, \tag{1.7}$$
$$x_1 = \frac{1}{2}(b_1 + x_2) = \frac{1}{2}\,b_1 + \frac{1}{4}\,b_2 + \frac{1}{4}\,b_3.$$

Consider the 3×3 matrices arising in Example 1.2.2

$$A = \begin{bmatrix} 2 & -1 & 0 \\ 2 & -1 & 1 \\ -2 & 3 & -1 \end{bmatrix}, \quad L = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}, \quad P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad U = \begin{bmatrix} 2 & -1 & 0 \\ 0 & 2 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

corresponding respectively to the given system of linear equations, the elimination step, the permutation of the second and the third rows, and the obtained system in row echelon form. The procedure therein can be condensed as the equality $P\,L\,A = U$. We can verify by a direct computation that $(P\,L)^{-1} = P\,L$, and so this equality can be equivalently written as

$$A = \begin{bmatrix} 2 & -1 & 0 \\ 2 & -1 & 1 \\ -2 & 3 & -1 \end{bmatrix} = P\,L\,U = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & -1 & 0 \\ 0 & 2 & -1 \\ 0 & 0 & 1 \end{bmatrix}. \tag{1.8}$$

The good news are that this matrix factorization can be extended to the general case: for any given nonsingular $n \times n$ matrix A, the Gaussian elimination algorithm can be used to factor it as

$$A = P\,L\,U \tag{1.9}$$

with P a permutation, L a unit lower triangular, and U an upper triangular matrix. With this factorization, the equation $A\,x = b$ gets reduced to similar equations for each of these factors that, thanks to their special structure, are easy to solve. Precisely, we perform the following the steps:

(1) solve $P\,z = b$ permuting the entries of b,
(2) solve $L\,y = z$ by forward substitution,
(3) solve $U\,x = y$ by backward substitution.

Then $A\,x = P\,(L\,(U\,x)) = P\,(L\,y) = P\,z = b$, and so x is the searched solution.

Example 1.2.3. Let us solve the system of linear equations in Example 1.2.2 using the PLU factorization in (1.8). We first solve $P\,z = b$, which gives

$$z_1 = b_1, \quad z_2 = b_3, \quad z_3 = b_2.$$

Next we solve $L\,y = z$ to obtain

$$y_1 = z_1 = b_1, \quad y_2 = z_2 + y_1 = b_1 + b_3, \quad y_3 = z_3 - y_1 = -b_1 + b_2.$$

Finally, the solution is computed by solving $U\,x = y$:

$$x_3 = y_3 = -b_1 + b_2, \quad x_2 = \frac{1}{2}(y_2 + x_3) = \frac{1}{2}\,b_2 + \frac{1}{2}\,b_3,$$
$$x_1 = \frac{1}{2}(y_1 + x_2) = \frac{1}{2}\,b_1 + \frac{1}{4}\,b_2 + \frac{1}{4}\,b_3,$$

in agreement with (1.7).

1.3. The GEPP algorithm

In §1.2, we discussed examples where the standard version of the Gaussian elimination algorithm applied to a given system of linear equations, produces a PLU factorization of the associated matrix as in (1.9). This is a general phenomenon and indeed, this algorithm can be modified to produce such a factorization for any given nonsingular square matrix.

Instead of going this way, we will devise a dedicated algorithm to compute such a factorization. This algorithm will make this computation in a more direct way although of course, the underlying ideas will be same as those in the standard version. More importantly, it will give us a more advanced point of view on Gaussian elimination: we will shift from an algorithm that performs a specific task (solving a linear equation) to the more general framework of matrix factorizations, which is the technique of choice in numerical linear algebra.

Let A be a nonsingular $n \times n$ matrix with entries in $\mathbb{F}$. A *PLU factorization* for A is its expression as a product of $n \times n$ matrices with entries in $\mathbb{F}$

$$A = P\,L\,U \tag{1.10}$$

with P a permutation, L unit lower triangular, and U upper triangular.

Before explaining how we can compute this factorization, we introduce and discuss the basic properties of permutations and their associated matrices.

An *n-permutation* is a bijection $\sigma\colon \{1,\dots,n\} \to \{1,\dots,n\}$. It is usually written verbose as an explicit sequence of preimage/image pairs:

$$\sigma = \begin{pmatrix} 1 & \cdots & n \\ \sigma(1) & \cdots & \sigma(n) \end{pmatrix}.$$

An *n-transposition* is an n-permutation σ whose only effect on the set $\{1,\dots,n\}$ consists of swapping two different elements a and b. It is usually written using cycle notation as

$$\sigma = (a, b).$$

The set of all n-permutations, denoted by $\mathcal{S}_n$, forms a group with respect to the composition of functions. Every n-permutation can be expressed as the composition of a finite number of transpositions.

To each $\sigma \in \mathcal{S}_n$ we associate the *permutation $n \times n$-matrix* constructed permuting the columns of the identity matrix according to σ, that is

$$P_\sigma = \begin{bmatrix} e_{\sigma(1)} & \cdots & e_{\sigma(n)} \end{bmatrix}$$

where the e_i's are the vectors in the *canonical basis* of $\mathbb{F}^n$, that is

$$e_i = (0,\dots,0,\overset{i}{1},0,\dots,0),$$

plugged into P_σ as columns following our convention in (1.1). For instance, to the permutation $\begin{pmatrix}1&2&3\\3&1&2\end{pmatrix}\in\mathcal{S}_3$ corresponds the permutation matrix

$$\begin{bmatrix}0&1&0\\0&0&1\\1&0&0\end{bmatrix}\in\mathbb{F}^3.$$

Pre-multiplication of A by a permutation $n\times n$ matrix produces a permutation of its rows, whereas its post-multiplication produces a permutation of its columns. Precisely

$$P_\sigma^T A=\begin{bmatrix}\mathrm{row}_{\sigma(1)}(A)\\ \vdots\\ \mathrm{row}_{\sigma(n)}(A)\end{bmatrix}\quad\text{and}\quad A\,P_\sigma=\begin{bmatrix}\mathrm{col}_{\sigma(1)}(A)&\cdots&\mathrm{col}_{\sigma(n)}(A)\end{bmatrix}. \tag{1.11}$$

Finally, we note that the correspondence between n-permutations and permutation matrices is compatible with the group structure of $\mathcal{S}_n$: for $\sigma'\in\mathcal{S}_n$ we have that

$$P_\sigma\,P_{\sigma'}=P_{\sigma\circ\sigma'}\quad\text{and}\quad P_\sigma^{-1}=P_{\sigma^{-1}}=P_\sigma^T \tag{1.12}$$

with P_σ^T the transpose of P_σ.

To compute the searched factorization of A, we will apply the following iterative procedure of pivoting and elimination by columns.

In the first step, choose $i_1\in\{1,\dots,n\}$ such that $a_{i_1,1}\neq 0$, let σ_1 be the permutation of the set $\{1,\dots,n\}$ that swaps 1 and i_1, and consider the associated permutation matrix P_{σ_1}. Relabel the entries of A so that $P_{\sigma_1}^T A=\big[a_{i,j}\big]_{i,j}$ and write it as the $(1,n-1)\times(1,n-1)$ block matrix

$$P_{\sigma_1}^T A=\begin{bmatrix}a_{1,1}&A_{1,2}\\A_{2,1}&A_{2,2}\end{bmatrix}\begin{matrix}1\\n-1\end{matrix}\quad\begin{matrix}1&n-1\end{matrix}. \tag{1.13}$$

Here $a_{1,1}\neq 0$ is the $(1,1)$ entry of $P_{\sigma_1}^T A$, whereas $A_{1,2}$ and $A_{2,1}$ denote respectively the rest of its first row and column, whereas $A_{2,2}$ is the remaining $(n-1)\times(n-1)$ block. Setting

$$u_{1,1}=a_{1,1},\quad L_{2,1}=a_{1,1}^{-1}\,A_{2,1},\quad U_{1,2}=A_{1,2},\quad A^{(1)}=A_{2,2}-L_{2,1}\,U_{1,2} \tag{1.14}$$

we obtain the block LU factorization

$$P_{\sigma_1}^T A=\begin{bmatrix}a_{1,1}&A_{1,2}\\A_{2,1}&A_{2,2}\end{bmatrix}=\begin{bmatrix}1&\mathbb{0}\\L_{2,1}&\mathbb{1}_{n-1}\end{bmatrix}\begin{bmatrix}u_{1,1}&U_{1,2}\\\mathbb{0}&A^{(1)}\end{bmatrix}. \tag{1.15}$$

The $(n-1)\times(n-1)$ matrix $A^{(1)}$ is nonsingular, and is called the *Schur complement* of $P_{\sigma_1}^T A$.

In the second step, we proceed similarly with this latter matrix: we choose $i_2\in\{2,\dots,n\}$ such that $a^{(1)}_{i_2,2}\neq 0$ and we denote by σ_2 the permutation of the set $\{2,\dots,n\}$ swapping 2 and i_2. Then we relabel the entries of the permuted matrix $P_{\sigma_2}^T A^{(1)}$, write it as a $(1,n-2)\times(1,n-2)$ block matrix as in (1.13) and, applying the formulae in (1.14), compute its block LU factorization

$$P_{\sigma_2}^T A^{(1)}=L_2\,U_2\quad\text{with}\quad L_2=\begin{bmatrix}1&\mathbb{0}\\L_{3,2}&\mathbb{1}_{n-2}\end{bmatrix}\quad\text{and}\quad U_2=\begin{bmatrix}u_{2,2}&U_{2,3}\\\mathbb{0}&A^{(2)}\end{bmatrix}. \tag{1.16}$$

It follows from (1.15) and (1.16) together with some algebraic manipulations that

$$\begin{aligned}\begin{bmatrix}1 & \mathbb{0}\\ \mathbb{0} & P_{\sigma_2}^T\end{bmatrix} P_{\sigma_1}^T A &= \begin{bmatrix}1 & \mathbb{0}\\ \mathbb{0} & P_{\sigma_2}^T\end{bmatrix}\begin{bmatrix}1 & \mathbb{0}\\ L_{2,1} & \mathbb{1}_{n-1}\end{bmatrix}\begin{bmatrix}u_{1,1} & U_{1,2}\\ \mathbb{0} & A^{(1)}\end{bmatrix}\\ &= \begin{bmatrix}1 & \mathbb{0}\\ P_{\sigma_2}^T L_{2,1} & \mathbb{1}_{n-1}\end{bmatrix}\begin{bmatrix}u_{1,1} & U_{1,2}\\ \mathbb{0} & P_{\sigma_2}^T A^{(1)}\end{bmatrix}\\ &= \begin{bmatrix}1 & \mathbb{0}\\ P_{\sigma_2}^T L_{2,1} & \mathbb{1}_{n-1}\end{bmatrix}\begin{bmatrix}u_{1,1} & U_{1,2}\\ \mathbb{0} & L_2 U_2\end{bmatrix}\\ &= \begin{bmatrix}1 & \mathbb{0}\\ P_{\sigma_2}^T L_{2,1} & L_2\end{bmatrix}\begin{bmatrix}u_{1,1} & U_{1,2}\\ \mathbb{0} & U_2\end{bmatrix}.\end{aligned} \tag{1.17}$$

In the subsequent steps, we choose for each $j = 3, \dots, n-1$ an index i_j such that $a_{i_j,j}^{(j-1)} \neq 0$ and denote by σ_j the permutation of the set $\{j, \dots, n\}$ swapping j with i_j. Then we relabel the entries of the permuted Schur complement $P_{\sigma_j}^T A^{(j-1)}$ and compute the corresponding $(1, n-j) \times (1, n-j)$ block LU factorization

$$P_{\sigma_j}^T A^{(j-1)} = L_j U_j$$

applying the formulae in (1.14), which produces the next Schur complement $A^{(j)}$ as the $(n-j) \times (n-j)$ block of U_j.

Finally, the permutation matrix in (1.10) is obtained considering for each j the permutation $n \times n$ matrix $P_j = \begin{bmatrix}\mathbb{1}_{j-1} & \mathbb{0}\\ \mathbb{0} & P_{\sigma_j}\end{bmatrix}$ and setting

$$P = \prod_{j=1}^{n-1} P_j. \tag{1.18}$$

For each j we have that P_j coincides with the permutation matrix corresponding to the transposition (j, i_j). By the compatibility formulae in (1.12), we deduce that $P = P_\sigma$ for the n-permutation

$$\sigma = (1, i_1) \circ \cdots \circ (n-1, i_{n-1}) \in \mathcal{S}_n.$$

On the other hand, the upper triangular matrix U in (1.10) is given by the first row of the U_j's, and the unit lower triangular matrix L is similarly obtained from the first column of the L_j's permuted according to the σ_k's for $k > j$ as in (1.17).

There is one further point, that is crucial for the numerical stability of the algorithm: the pivots have to be chosen wisely, because dividing by numbers that are too small can have disastrous consequences for the reliability of the computed solution. In *Gaussian elimination with partial pivoting (GEPP)*, for each step j this is done chosing the index i_j so that $|a_{i_j,j}|$ is maximal among $|a_{p,j}|$, $j \leq p \leq n$. By (1.14), this implies that all the entries of L will have absolute value bounded by 1.

We next describe the obtained algorithm in *pseudocode notation*. Therein we denote by P, L and U three $n \times n$ matrices, all of them initialized to $\mathbb{0}$, that will eventually become the searched factors. Also, for $j = 0, \dots, n-1$ we denote by

$$A^{(j)} = \left[a_{i,k}^{(j)}\right]_{j+1 \leq i,k \leq n}$$

an $(n-j) \times (n-j)$ matrix, the first one initialized to A and the others to $\mathbb{0}$, that will become the successive Schur complements.

Algorithm 1.3.1 (GEPP)

for $j = 1, \dots, n-1$
  choose $i_j \in \{j, \dots, n\}$ such that $\left|a_{i_j,i}^{(j-1)}\right| = \max_{j \le p \le n} \left|a_{p,j}^{(j-1)}\right|$
  swap row j and row i_j of L and of $A^{(j-1)}$
  for $k = j+1, \dots, n$
    $u_{j,k} \leftarrow a_{j,k}^{(j-1)}$ (compute the j-th row of U)
  for $i = j+1, \dots, n$
    $l_{i,j} \leftarrow a_{i,j}^{(j-1)} / a_{j,j}^{(j-1)}$ (compute the j-th column of L)
  for $i, k = i+1, \dots, n$
    $a_{i,k}^{(j)} \leftarrow a_{i,k}^{(j-1)} - l_{i,j}\, u_{j,k}$ (compute the j-th Schur complement)
$u_{n,n} \leftarrow a_{n,n}^{(n-1)}$ (compute the (n,n) entry of U)
$P \leftarrow P_\sigma$ for the n-permutation $\sigma = (1, i_1) \circ \cdots \circ (n-1, i_{n-1})$

Taking a closer look at this algorithm, we see that at the j-th step each entry $a_{j,k}^{(j-1)}$ is assigned to the corresponding entry of U and is never used again. Moreover, each entry $a_{i,j}^{(j-1)}$ is used to compute the corresponding entry of L and is neither used again, and the same happens with the entries that compute the j-th Schur complement. Hence at that step we can safely rewrite A with the nontrivial entries in the j-th column of L and the j-th row of U. and with those of the j-th Schur complement, and so we need no extra space to store the computed entries.

With this strategy, at the end of the algorithm the matrix A would contain all the nontrivial entries of L and of U distributed as in Figure 1.3.1.

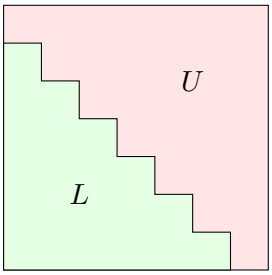

FIGURE 1.3.1. Storage distribution for GEPP

Moreover, the permutation matrix P can be coded by the corresponding element of $\mathcal{S}_n$ or even as the sequence of transpositions definining it, leading to a more efficient implementation (Algorithm 1.3.2).

As an illustration, we apply it to recompute the PLU factorization in Example 1.2.3.

EXAMPLE 1.3.1. Set

$$A = \begin{bmatrix} 2 & -1 & 0 \\ 2 & -1 & 1 \\ -2 & 3 & -1 \end{bmatrix}.$$

For $j = 1$ we choose $i_1 = 1$, and so the rows of the matrix need not be permuted at this step. We compute the first column of L and the entries of the first Schur complement, and overwrite the matrix A with the obtained values: the used operations and updated

Algorithm 1.3.2 (GEPP with overwritting)

for $j = 1, \dots, n-1$
  choose $i_j \in \{j, \dots, n\}$ such that $|a_{i_j,i}| = \max_{j \le p \le n} |a_{p,j}|$
  swap row j and row i_j of A
  for $i = j+1, \dots, n$
    $a_{i.j} \leftarrow a_{i.j}/a_{j,j}$
  for $i, k = j+1, \dots, n$
    $a_{i,k} \leftarrow a_{i,k} - a_{i,j}\, a_{j,k}$
$\sigma \leftarrow (1, i_1) \circ \cdots \circ (n-1, i_{n-1})$

matrix are

$$A = \begin{bmatrix} 2 & -1 & 0 \\ 2/2 & -1-1\cdot(-1) & 1-1\cdot 0 \\ -2/2 & 3-(-1)\cdot(-1) & -1-(-1)\cdot 0 \end{bmatrix} = \begin{bmatrix} 2 & -1 & 0 \\ 1 & 0 & 1 \\ -1 & 2 & -1 \end{bmatrix}.$$

For $j = 2$ we choose $i_2 = 3$, and then swap the second and the third rows of A. We compute the second column of L and the entries of the second Schur complement, and overwrite the matrix A with the obtained values to get

$$A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 1 & 0/2 & 1-0\cdot(-1) \end{bmatrix} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 1 & 0 & 1 \end{bmatrix}.$$

This matrix together with the chosen indexes $i_1 = 1$ and $i_2 = 3$ encode the PLU factorization of A:

$$P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad L = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 2 & -1 & 0 \\ 0 & 2 & -1 \\ 0 & 0 & 1 \end{bmatrix}.$$

Another variant of this algorithm is *Gaussian elimination with complete pivoting (GECP)*, that at the j-th step reorders the rows and all the columns of the $(j-1)$th Schur complement, so to choose a pivot whose absolute value is maximal among all the entries of this matrix. It yields a factorization of the form

$$A = P_1\, L\, U P_2$$

where both P_1 and P_2 are permutation matrices. This variant is slower than GEPP, but can be also more numerically stable.

As discussed in §1.2, the factorization $A = PLU$ allows to solve the linear equation $A\,x = b$ by reducing it to the three analogous but simpler equations

$$Pz = b, \quad Ly = z, \quad Ux = y.$$

The equation $Pz = b$ can be solved permuting the entries of the n-vector b according to the permutation $\sigma = (1, i_1) \circ \cdots \circ (n-1, i_{n-1})$, which can be done by successively swapping b_j and b_{i_j}, $j = 1, \dots, n-1$. On the other hand, $Ly = z$ and $Ux = y$ can be solved by applying Algorithms 1.3.3 and 1.3.4, respectively.

Algorithm 1.3.3 (Forward substitution)

for $i = 1, \dots, n$
  $y_i \leftarrow z_i - \sum_{j=1}^{i-1} l_{i,j} z_j$