Chapter 3

Direct Methods for Solving Systems of Linear Equations

The problem of solving a linear system of equations

$$Ax = b, \quad \text{where} \quad A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}, \tag{3.1}$$

and

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

is to be solved, is central in scientific computation. In this chapter we focus on the method of Gaussian elimination, the algorithm of choice when A is square, dense, and unstructured.

3.1 Triangular Systems

3.1.1 Diagonal System

The easiest linear system of equations (3.1) is when the coefficient matrix A is a diagonal, i.e.,

$$A = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix}.$$

Then computing the solution x is trivial

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1/a_{11} \\ b_2/a_{22} \\ \vdots \\ b_n/a_{nn} \end{bmatrix}. \tag{3.2}$$

However, a linear system with diagonal coefficient matrix is not common and reducing a general matrix to diagonal form is not practical.

3.1.2 Forward Substitution

Traditional factorization methods for linear systems involve the reduction of the given square system to a triangular system that has the same solution. First we consider a triangular system $Lx = b$, where the coefficient matrix L is lower triangular and all diagonals $\ell_{ii} \neq 0$,

$$\begin{bmatrix} \ell_{11} & 0 & \cdots & 0 \\ \ell_{21} & \ell_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \ell_{n1} & \ell_{n2} & \cdots & \ell_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}. \tag{3.3}$$

Then x_i can be computed with the following procedure.

$$\begin{aligned} x_1 &= b_1/\ell_{11} \\ x_2 &= (b_2 - \ell_{21}x_1)/\ell_{22} \\ x_3 &= (b_3 - \ell_{31}x_1 - \ell_{32}x_2)/\ell_{33} \\ &\vdots \\ x_n &= (b_n - \ell_{n1}x_1 - \ell_{n2}x_2 - \cdots - \ell_{n,n-1}x_{n-1})/\ell_{nn} \end{aligned}$$

The general formulation for computing x_i is

$$x_i = \left(b_i - \sum_{j=1}^{i-1} \ell_{ij}x_j \right) \Big/ \ell_{ii}, \tag{3.4}$$

and the corresponding algorithm for solving a lower triangular linear system is called forward substitution.

Algorithm 3.1 (Forward substitution: dot-product version) *If $L \in \mathbb{R}^{n\times n}$ is nonsingular lower triangular and $b \in \mathbb{R}^n$, then this algorithm computes the solution of $Lx = b$ with row-oriented procedure.*

```
for i = 1, ... , n do
   tmp = 0.0
   for j = 1, ... , i − 1 do
      tmp = tmp + L(i, j) ⋆ x(j)
   end for
   x(i) = (b(i) − tmp)/L(i, i)
end for
```

Algorithm 3.2 (Forward substitution: SAXPY version) *If $L \in \mathbb{R}^{n\times n}$ is nonsingular lower triangular and $b \in \mathbb{R}^n$, then this algorithm computes the solution of $Lx = b$ with column-oriented procedure.*

```
for i = 1, ... , n do
   x(i) = b(i)
end for
for j = 1, ... , n do
   x(j) = x(j)/L(j, j)
   for i = (j + 1), ... , n do
      x(i) = x(i) − L(i, j) ⋆ x(j)
   end for
end for
```

Forward substitution requires $n^2 + O(n)$ flops. A flop stands for one floating-point operation, $+, -, \star$, or $\div$.

3.1.3 Back Substitution

The analogous to the lower triangular linear system is the upper triangular system $Ux = b$,

$$\begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}. \tag{3.5}$$

The solution x_i are computed in a reversed order by

$$\begin{aligned} x_n &= b_n/u_{nn} \\ x_{n-1} &= (b_{n-1} - u_{n-1,n}x_n)/u_{n-1,n-1} \\ x_{n-2} &= (b_{n-2} - u_{n-2,n-1}x_{n-1} - u_{n-2,n}x_n)/u_{n-2,n-2} \\ &\vdots \\ x_1 &= (b_1 - u_{12}x_2 - u_{13}x_3 - \cdots - u_{1n}x_n)/u_{11} \end{aligned}$$

The general formulation for computing x_i is

$$x_i = \left(b_i - \sum_{j=i+1}^{n} u_{ij}x_j \right) \Big/ u_{ii}, \tag{3.6}$$

and the corresponding alogrithm is called back substitution.

Algorithm 3.3 (Back substitution: Dot-Product version) *If $U \in \mathbb{R}^{n\times n}$ is nonsingular upper triangular and $b \in \mathbb{R}^n$, then this algorithm computes the solution of $Ux = b$ with row-oriented procedure.*

$x(n) = b(n)/U(n,n)$
for $i = (n-1), \ldots, 1$ **do**
  $tmp = 0.0$
  for $j = i+1 : n$ **do**
    $tmp = tmp + U(i,j) \star x(j)$
  end for
  $x(i) = (b(i) - tmp)/U(i,i)$
end for

Algorithm 3.4 (Back substitution: SAXPY version) *If $U \in \mathbb{R}^{n\times n}$ is nonsingular upper triangular and $b \in \mathbb{R}^n$, then this algorithm computes the solution of $Ux = b$ with row-oriented procedure.*

for $i = 1, \ldots, n$ **do**
  $x(i) = b(i)$
end for
for $j = n, \ldots, 1$ **do**
  $x(j) = x(j)/U(j,j)$
  for $i = 1 : (j-1)$ **do**
    $x(i) = x(i) - U(i,j) \star x(j)$
  end for
end for

Back subsitution requires $n^2 + O(n)$ flops.

3.2 Gaussian Elimination and LU Factorization

As we have just seen, triangular systems are "easy" to solve. The idea behind Gaussian elimination is to convert a given system $Ax = b$ to an equivalent triangular system. The conversion is achieved by taking appropriate linear combination of the equations. In this section we will give a complete specification of this central procedure and show that the algorithm computes a unit lower triangular matrix L and an upper trianguar matrix U such that $A = LU$. The solution to the original problem

$$Ax = LUx = Ly = b \tag{3.7}$$

is then found by a two-step triangular solve process:

$$Ly = b, \qquad Ux = y. \tag{3.8}$$

3.2.1 Gaussian Elimination

Gaussian elimination is based on the elementary operations:

1. interchange two equations in the system (interchange two rows in A):
$$E_i \leftrightarrow E_j.$$
2. multiply an equation by a non-zero constant (multiply one row by a non-zero constant):
$$E_i \leftarrow \lambda E_i.$$
3. add the multiple of another equation (add to one row a multiple of another row):
$$E_i \leftarrow E_i + \lambda E_j.$$

By applying a sequence of elementary operations, A can be reduced to upper triangular form, i.e.,

$$A = A^{(1)} \to A^{(2)} \to \cdots \to A^{(n)},$$

where $A^{(n)}$ is upper triangular. Suppose at the end of $k-1$ step, $A^{(k)}$ has the following form:

$$A^{(k)} = \left[\begin{array}{ccc|c|cccc}
a_{11}^{(k)} & \cdots & a_{1,k-1}^{(k)} & a_{1k}^{(k)} & \cdots & a_{1j}^{(k)} & \cdots & a_{1n}^{(k)} \\
\vdots & \ddots & \vdots & \vdots & & \vdots & & \vdots \\
0 & \cdots & a_{k-1,k-1}^{(k)} & a_{k-1,k}^{(k)} & \cdots & a_{k-1,j}^{(k)} & \cdots & a_{k-1,n}^{(k)} \\ \hline
0 & \cdots & 0 & a_{kk}^{(k)} & \cdots & a_{kj}^{(k)} & \cdots & a_{kn}^{(k)} \\ \hline
\vdots & & \vdots & \vdots & & \vdots & & \vdots \\
0 & \cdots & 0 & a_{ik}^{(k)} & \cdots & a_{ij}^{(k)} & \cdots & a_{in}^{(k)} \\
\vdots & & \vdots & \vdots & & \vdots & & \vdots \\
0 & \cdots & 0 & a_{nk}^{(k)} & \cdots & a_{nj}^{(k)} & \cdots & a_{nn}^{(k)}
\end{array}\right]$$

At the k-th step, we use $a_{kk}^{(k)}$ as pivot element, the k-th row of $A^{(k)}$ as pivot row, and apply elementary operations to obtain $A^{(k+1)}$ in which

1. row 1 through row k are unchanged;
2. $a_{k+1,k}^{(k)}, \cdots, a_{nk}^{(k)}$ are zeroed;
3. row $k+1$ through n are modified.

The formulation for the multipliers is

$$\ell_{ik} = \begin{cases} 0, & \text{if } i < k; \\ 1, & \text{if } i = k; \\ \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}, & \text{if } i > k, \end{cases} \tag{3.9}$$

and the entries of $A^{(k+1)}$ are updated by

$$a_{ij}^{(k+1)} = \begin{cases} a_{ij}^{(k)}, & \text{for } i = 1, \dots, k, \text{ and } j = 1, \dots, n; \\ a_{ij}^{(k)} - \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \star a_{kj}^{(k)}, & \text{for } i = k+1, \dots, n, \text{ and } j = k+1, \dots, n; \\ 0, & \text{for } i = k+1, \dots, n, \text{ and } j = 1, \dots, k. \end{cases} \tag{3.10}$$

Algorithm 3.5 (Gaussian elimination) *Given $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$, this algorithm implements the Gaussian elimination procedure to reduce A to upper triangular and modify the entries of b accordingly.*

```
for k = 1, ... , n − 1 do
  for i = k + 1, ... , n do
    t = A(i, k)/A(k, k)
    A(i, k) = 0
    b(i) = b(i) − t ⋆ b(k)
    for j = k + 1, ... , n do
      A(i, j) = A(i, j) − t ⋆ A(k, j)
    end for
  end for
end for
```

Obviously the algorithm will break down if a zero $A(k, k)$ is encountered, a check for nonzero pivot should be included in practical implementation. After the reduction, A becomes upper triangular and the back substitution can be used to obtain the solution vector x.

Example 3.1 *This example illustrates the application of Gaussian elimination in solving system of linear equations.*

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 12 \\ 34 \\ 27 \\ -38 \end{bmatrix}$$

Sol:

1^{st} **step** Use 6 as pivot element, the first row as pivot row, and multipliers $2, \frac{1}{2}, -1$, to reduce the system to

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & -12 & 8 & 1 \\ 0 & 2 & 3 & -14 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 12 \\ 10 \\ 21 \\ -26 \end{bmatrix}$$

2^{nd} **step** Use -4 as pivot element, the second row as pivot row, and multipliers $3, -\frac{1}{2}$, to reduce the system to

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 4 & -13 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 12 \\ 10 \\ -9 \\ -21 \end{bmatrix}$$

3^{rd} **step** Use 2 as pivot element, the third row as pivot row, and multipliers 2, to reduce the system to

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 12 \\ 10 \\ -9 \\ -3 \end{bmatrix}$$

Now the solution can be obtained by solving the triangular system with back substitution. Note that if we collect all the multipliers and let

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ \frac{1}{2} & 3 & 1 & 0 \\ -1 & -\frac{1}{2} & 2 & 1 \end{bmatrix} \quad \text{and} \quad U = \begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{bmatrix},$$

then one can verify that $LU = A$. ■

3.2.2 Gaussian Transformation and LU Factorization

For a given n-vector

$$v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix},$$

with $v_k \neq 0$ for some $1 \leq k \leq n$, let

$$\ell_{ik} = \frac{v_i}{v_k}, \quad i = k+1, \dots, n, \qquad l_k = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \ell_{k+1,k} \\ \vdots \\ \ell_{n,k} \end{bmatrix}$$

and

$$M_k = I - l_k e_k^T = \begin{bmatrix} 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & \cdots & -\ell_{k+1,k} & 1 & \cdots & 0 \\ \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & -\ell_{n,k} & 0 & \cdots & 1 \end{bmatrix}. \tag{3.11}$$

Then one can verify that

$$M_k v = \begin{bmatrix} v_1 \\ \vdots \\ v_k \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

That is, M_k zeros $v_{k+1}, \ldots, v_n$ and keeps $v_1, \ldots, v_k$ unchanged. This unit lower triangular matrix M_k is called a Gaussian transformation, the vector l_k a Gauss vector, and the entries $\ell_{k+1}, \ldots, \ell_n$ the multipliers. Furthermore, one can verify that

$$M_k^{-1} = (I - l_k e_k^T)^{-1} = I + l_k e_k^T = \begin{bmatrix} 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & \cdots & \ell_{k+1,k} & 1 & \cdots & 0 \\ \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \ell_{n,k} & 0 & \cdots & 1 \end{bmatrix}. \tag{3.12}$$

Given a nonsingular matrix $A \in \mathbb{R}^{n\times n}$, Gauss transformations $M_1, M_2, \ldots, M_{n-1}$ can usually be found such that $M_{n-1} \cdots M_2 M_1 A \equiv U$ is upper triangular. To see this, we denote $A^{(1)} = [a_{ij}^{(1)}] = A$. If $a_{11}^{(1)} \neq 0$, then $M_1 = I - l_1 e_1^T$, where

$$l_1 = \begin{bmatrix} 0 \\ \ell_{21} \\ \vdots \\ \ell_{n1} \end{bmatrix}, \qquad \ell_{i1} = \frac{a_{i1}^{(1)}}{a_{11}^{(1)}}, \quad i = 2, \ldots, n,$$

can be formed such tht

$$A^{(2)} = M_1 A^{(1)} = \begin{bmatrix} a_{11}^{(2)} & a_{12}^{(2)} & \cdots & a_{1n}^{(2)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} \end{bmatrix},$$

where

$$a_{ij}^{(2)} = \begin{cases} a_{ij}^{(1)}, & \text{for } i = 1 \text{ and } j = 1, \dots, n; \\ a_{ij}^{(1)} - \ell_{i1} \star a_{1j}^{(1)}, & \text{for } i = 2, \dots, n \text{ and } j = 2, \dots, n. \end{cases}$$

The procedure can proceed to zero the entries in the second column below the diagonal, and so on. In general, at the k-th step, we are confronted with a matrix

$$\begin{aligned} A^{(k)} &= M_{k-1} \cdots M_2 M_1 A^{(1)} \\ &= \left[\begin{array}{cccc|ccc} a_{11}^{(k)} & a_{12}^{(k)} & \cdots & a_{1,k-1}^{(k)} & a_{1k}^{(k)} & \cdots & a_{1n}^{(k)} \\ 0 & a_{22}^{(k)} & \cdots & a_{2,k-1}^{(k)} & a_{2k}^{(k)} & \cdots & a_{2n}^{(k)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & a_{k-1,k-1}^{(k)} & a_{k-1,k}^{(k)} & \cdots & a_{k-1,n}^{(k)} \\ \hline 0 & 0 & \cdots & 0 & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} \\ \vdots & \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & a_{kn}^{(k)} & \cdots & a_{nn}^{(k)} \end{array}\right] \\ &= \left[\begin{array}{cccc|ccc} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1,k-1}^{(1)} & a_{1k}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2,k-1}^{(2)} & a_{2k}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & a_{k-1,k-1}^{(k)} & a_{k-1,k}^{(k)} & \cdots & a_{k-1,n}^{(k)} \\ \hline 0 & 0 & \cdots & 0 & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} \\ \vdots & \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & a_{kn}^{(k)} & \cdots & a_{nn}^{(k)} \end{array}\right] \end{aligned}$$

If the pivot $a_{kk}^{(k)} \neq 0$, then the multipliers

$$\ell_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}, \quad i = k+1, \dots, n$$

can be computed and the Gauss transform $M_k = I - l_k e_k^T$ with $l_k = [0, \cdots 0, \ \ell_{k+1,k}, \ \cdots \ell_{nk}]^T$ exists and can be applied to $A^{(k)}$ to obtain

$$A^{(k+1)} = M_k A^{(k)} = \left[\begin{array}{cccc|cccc} a_{11}^{(k+1)} & a_{12}^{(k+1)} & \cdots & a_{1,k-1}^{(k+1)} & a_{1k}^{(k+1)} & \cdots & \cdots & a_{1n}^{(k+1)} \\ 0 & a_{22}^{(k+1)} & \cdots & a_{2,k-1}^{(k+1)} & a_{2k}^{(k+1)} & \cdots & \cdots & a_{2n}^{(k+1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \cdots & \vdots \\ 0 & 0 & \cdots & a_{k-1,k-1}^{(k+1)} & a_{k-1,k}^{(k+1)} & \cdots & \cdots & a_{k-1,n}^{(k+1)} \\ \hline 0 & 0 & \cdots & 0 & a_{kk}^{(k+1)} & \cdots & \cdots & a_{kn}^{(k+1)} \\ \vdots & \vdots & & \vdots & 0 & \ddots & & \vdots \\ \vdots & \vdots & & \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & \cdots & a_{nn}^{(k+1)} \end{array}\right],$$

where

$$a_{ij}^{k+1)} = \begin{cases} a_{ij}^{(k)}, & \text{for } i = 1, \ldots, k,\ j = 1, \ldots, n; \\ 0, & \text{for } i = k+1, \ldots, n,\ j = k; \\ a_{ij}^{(k)} - \ell_{ik} a_{kj}^{(k)}, & \text{for } i = k+1, \ldots, n,\ j = k+1, \ldots, n. \end{cases}$$

Upon the completion,

$$U \equiv A^{(n)} = M_{n-1} \cdots M_2 M_1 A$$

is upper triangular. Hence

$$A = M_1^{-1} M_2^{-1} \cdots M_{n-1}^{-1} U \equiv LU, \tag{3.13}$$

is called the LU-factorization of A, where

$$\begin{aligned} L \equiv M_1^{-1} M_2^{-1} \cdots M_{n-1}^{-1} &= (I - l_1 e_1^T)^{-1} (I - l_2 e_2^T)^{-1} \cdots (I - l_{n-1} e_{n-1}^T)^{-1} \\ &= (I + l_1 e_1^T)^{-1} (I + l_2 e_2^T)^{-1} \cdots (I + l_{n-1} e_{n-1}^T)^{-1} \\ &= I + l_1 e_1^T + l_2 e_2^T + \cdots + l_{n-1}^T e_{n-1}^T \\ &= \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ \ell_{21} & 1 & 0 & \cdots & 0 \\ \ell_{31} & \ell_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \ell_{n1} & \ell_{n2} & \ell_{n3} & \cdots & 1 \end{bmatrix} \end{aligned} \tag{3.14}$$

is unit lower triangular.

Alternatively, the process for computing the LU-factorization can be derived by componentwise comparision in $A = LU$:

$$\begin{aligned} &\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix} \\ = &\begin{bmatrix} \ell_{11} & 0 & 0 & \cdots & 0 \\ \ell_{21} & \ell_{22} & 0 & \cdots & 0 \\ \ell_{31} & \ell_{32} & \ell_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \ell_{n1} & \ell_{n2} & \ell_{n3} & \cdots & \ell_{nn} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} \\ 0 & 0 & u_{33} & \cdots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & u_{nn} \end{bmatrix} \\ = &\begin{bmatrix} \ell_{11}u_{11} & \ell_{11}u_{12} & \ell_{11}u_{13} & \cdots & \ell_{11}u_{1n} \\ \ell_{21}u_{11} & \ell_{21}u_{12} + \ell_{22}u_{22} & \ell_{21}u_{13} + \ell_{22}u_{23} & \cdots & l_{21}u_{1n} + \ell_{22}u_{2n} \\ \ell_{31}u_{11} & \ell_{31}u_{12} + \ell_{32}u_{22} & \ell_{31}u_{13} + \ell_{32}u_{23} + \ell_{33}u_{33} & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \ell_{n1}u_{11} & \ell_{n1}u_{12} + \ell_{n2}u_{2n} & \ell_{n1}u_{13} + \ell_{n2}u_{23} + \ell_{n3}u_{33} & \cdots & \ell_{n1}u_{1n} + \cdots + \ell_{nn}u_{nn} \end{bmatrix} \end{aligned}$$

Therefore, $a_{11} = \ell_{11}u_{11}$. Assign $\ell_{11} = 1$ we can compute $u_{11} = a_{11}$. Once ℓ_{11} and u_{11} are determined, $\ell_{21} = a_{21}/u_{11}, \ell_{31} = a_{13}/u_{11}, \ldots, \ell_{n1} = a_{1n}/u_{11}$ and $u_{12} = a_{12}/\ell_{11}, u_{13} = a_{13}/\ell_{11}, \ldots, u_{1n} = a_{1n}/\ell_{11}$ can be obtained. And then we proceed to compute l_{22} and u_{22} and so on.

In general, we have the following identity

$$a_{ij} = \sum_{p=1}^{n} \ell_{ip}u_{pj} = \sum_{p=1}^{\min(i,j)} \ell_{ip}u_{pj}. \tag{3.15}$$

Suppose at the k-th step we have computed the first $k-1$ columns of L and the first $k-1$ rows of U. The next step is to assign $\ell_{kk} = 1$ and determine u_{kk}. Since

$$a_{kk} = \sum_{p=1}^{k} \ell_{kp}u_{pk} = \sum_{p=1}^{k-1} \ell_{kp}u_{pk} + \ell_{kk}u_{kk}$$

and ℓ_{kp}, u_{pk}, $p = 1, 2, \ldots, k-1$, are known, hence

$$u_{kk} = a_{kk} - \sum_{p=1}^{k-1} \ell_{kp}u_{pk}. \tag{3.16}$$

Once ℓ_{kk} and u_{kk} are determined, we can compute the k-th column of L and the k-th row of U by

$$\ell_{ik} = \left(a_{ik} - \sum_{p=1}^{k-1} \ell_{ip}u_{pk} \right) \Big/ u_{kk}, \qquad i = k+1, \ldots, n, \tag{3.17}$$

and

$$u_{kj} = \left(a_{kj} - \sum_{p=1}^{k-1} \ell_{kp}u_{pj} \right) \Big/ \ell_{kk}, \qquad j = k+1, \ldots, n. \tag{3.18}$$

Note that (3.17) and (3.18) follow from the identities

$$a_{ik} = \sum_{p=1}^{k-1} l_{ip}u_{pk} + l_{ik}u_{kk}, \qquad i = k+1, \ldots, n,$$

and

$$a_{kj} = \sum_{p=1}^{k-1} l_{kp}u_{pj} + l_{kk}u_{kj}, \qquad j = k+1, \ldots, n.$$

From (3.17) and (3.18), we know that if $u_{kk} \neq 0$, we can obtain $\ell_{ik}, i = k+1, \ldots, n$, and $u_{kj}, j = k+1, \ldots, n$, at the k-th stage.This gives the Doolittle LU-factorization algorithm.

Algorithm 3.6 (Doolittle LU Factorization) *Given a nonsingular square matrix $A \in \mathbb{R}^{n\times n}$, this algorithm computes a unit lower triangular matrix L and an upper triangular matrix U such that $A = LU$.*

Initialize $L = 0$, $U = 0$
for $k = 1, \ldots, n$ **do**
  $L(k,k) = 1$
  $t = 0$
  for $p = 1, \ldots, k-1$ **do**
    $t = t + L(k,p) \star U(p.k)$
  end for
  $U(k,k) = A(k,k) - t$
  for $i = k+1, \ldots, n$ **do**
    $t = 0$
    for $p = 1, \ldots, k-1$ **do**
      $t = t + L(i,p) \star U(p,k)$
    end for
    $L(i,k) = (A(i,k) - t)/U(k,k)$
  end for
  for $j = k+1, \ldots, n$ **do**
    $t = 0$
    for $p = 1, \ldots, k-1$ **do**
      $t = t + L(k,p) \star U(p,j)$
    end for
    $U(k,j) = A(k,j) - t$
  end for
end for

For practical implementation, the storage of A can be overwritten by the entries of L and U. The upper triangular components of U overwrite upper triangular part of A, and the multipliers (the strictly lower triangular part of L) can be stored in the strictly lower triangualr part of A. The diagonal part of L needs not to be stored since they are all 1. An outer-product version of the LU-factorization is as follows.

Algorithm 3.7 (LU Factorization) *Given a nonsingular square matrix $A \in \mathbb{R}^{n\times n}$, this algorithm computes a unit lower triangular matrix L and an upper triangular matrix U such that $A = LU$. The matrix A is overwritten by L and U.*

for $k = 1, \ldots, n-1$ **do**
  for $i = k+1, \ldots, n$ **do**
    $A(i,k) = A(i,k)/A(k,k)$
    for $j = k+1, \ldots, n$ **do**
      $A(i,j) = A(i,j) - A(i,k) \star A(k,j)$
    end for
  end for
end for

This algorithm requires $\frac{2}{3}n^3 - \frac{1}{2}n^2 + O(n)$ flops.

The LU-factorization together with forward substitution and back substitution is a standard procedure for solving the linear system of equations. The total cost for this approach is $\frac{2}{3}n^3 + \frac{3}{2}n^2 + O(n)$ flops.

3.2.3 Existence and Uniqueness of LU Factorization

Definition 3.1 (Leading principal minor) *The k-th leading principal minor of a given matrix $A \in \mathbb{R}^{n\times n}$ is the determinant of the leading $k \times k$ principal submatrix*

$$A_k = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22} & \cdots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & \cdots & a_{kk} \end{bmatrix}.$$

Theorem 3.1 *If all leading principal minor of $A \in \mathbb{R}^{n\times n}$ are nonzero, then A has an LU-factorization.*

Proof: . By mathematical induction, when $k = 1$, $A_1 = [a_{11}]$ is nonsingular by assumption, this implies $a_{11} \neq 0$. Let $L_1 = [1]$ and $U_1 = [a_{11}]$. Then $A_1 = L_1U_1$. The theorem holds.

Assume that the theorem holds for $i = 1, \ldots, k$. That is, $A_k = L_kU_k$, where L_k is unit lower triangular and U_k is upper triangular. We claims that there exist an unit lower triangular matrix L_{k+1} and an upper triangular matrix U_{k+1} such that $A_{k+1} = L_{k+1}U_{k+1}$.

We write

$$A_{k+1} = \begin{bmatrix} A_k & v_k \\ w_k^T & a_{k+1,k+1} \end{bmatrix},$$

where

$$v_k = \begin{bmatrix} a_{1,k+1} \\ a_{2,k+1} \\ \vdots \\ a_{k,k+1} \end{bmatrix} \quad \text{and} \quad w_k = \begin{bmatrix} a_{k+1,1} \\ a_{k+1,2} \\ \vdots \\ a_{k+1,k} \end{bmatrix}.$$

Since A_k is nonsingular which implies that L_k and U_k are all nonsingular. Therefore the system $L_ky_k = v_k$ has a unique solution $y_k \in \mathbb{R}^n$, and $z^tU_k = w_k^T$ has a unique solution $z_k \in \mathbb{R}^n$.

Let

$$L_{k+1} = \begin{bmatrix} L_k & 0 \\ z_k^T & 1 \end{bmatrix} \quad \text{and} \quad U_{k+1} = \begin{bmatrix} U_k & y_k \\ 0 & a_{k+1,k+1} - z_k^T y_k \end{bmatrix}.$$

Then

$$L_{k+1}U_{k+1} = \begin{bmatrix} L_kU_k & L_ky_k \\ z_k^TU_k & z_k^Ty_k + a_{k+1,k+1} - z_k^Ty_k \end{bmatrix} = \begin{bmatrix} A_k & v_k \\ w_k^T & a_{k+1,k+1} \end{bmatrix} = A_{k+1}.$$

This proves the theorem. ∎

Theorem 3.2 *If A is nonsingular and the LU factorization exists, then the LU factorization is unique and* $\det(A) = u_{11} \cdots u_{nn}$.

Proof: Suppose both $A = L_1U_1$ and $A = L_2U_2$ are LU factorizations. Since A is nonsingular, L_1, U_1, L_2, U_2 are all nonsingular, and

$$A = L_1U_1 = L_2U_2 \Longrightarrow L_2^{-1}L_1 = U_2U_1^{-1}.$$

Since L_1 and L_2 are unit lower triangular which implies $L_2^{-1}L_1$ is unit lower triangular, and U_1 and U_2 are upper triangular which implies $U_2U_1^{-1}$ is upper triangular, it follows that $L_2^{-1}L_1 = I = U_2U_1^{-1}$. Therefore $L_1 = L_2$ and $U_1 = U_2$. ■

3.3 Pivoting

3.3.1 The Need for Pivoting

The Gaussian elimination algorithm 3.5 could fail on systems that are in fact easy to solve. For example, the algorithm would fail at the first step on the input system

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

But if we interchange the rows, the system

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

becomes trivial to solve.

Moreover, the algorithm would produce relatively large error on the system

$$\begin{bmatrix} \varepsilon & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix},$$

where $\varepsilon < \epsilon_M$, since algorithm 3.5 would give

$$\begin{bmatrix} \varepsilon & 1 \\ 0 & 1-\frac{1}{\varepsilon} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2-\frac{1}{\varepsilon} \end{bmatrix} \Longrightarrow \begin{bmatrix} \varepsilon & 1 \\ 0 & -\frac{1}{\varepsilon} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -\frac{1}{\varepsilon} \end{bmatrix},$$

and back substitution results

$$x_2 = \frac{-\frac{1}{\varepsilon}}{-\frac{1}{\varepsilon}} = 1 \quad \text{and} \quad x_1 = \frac{1-1}{\varepsilon} = 0.$$

But actually $x_1 = x_2 = 1$ would be a much better solution since

$$\begin{bmatrix} \varepsilon & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1+\varepsilon \\ 2 \end{bmatrix} \approx \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

However, it would produce a much better result if we interchange the rows.

$$\begin{bmatrix} 1 & 1 \\ \varepsilon & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ x_1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \Longrightarrow \begin{bmatrix} 1 & 1 \\ 0 & 1-\varepsilon \end{bmatrix} \begin{bmatrix} x_2 \\ x_1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1-2\varepsilon \end{bmatrix}$$

Then it would give

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ x_1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \Longrightarrow \begin{bmatrix} x_2 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

The strategy of interchange rows/columns as described above is called "pivoting". And we have seen that pivoting is necessary for some systems.

3.3.2 Partial Pivoting and Complete Pivoting

As we have observed in the previous subsection that if $a_{kk}^{(k)}$ is small in magnitude compared to $a_{ik}^{(k)}$, $i = k+1, \ldots, n$, then the multipliers

$$\ell_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$$

will have magnitude much larger than 1. Roundoff introduced in computing

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - \ell_{ik} \star a_{kj}^{(k)}$$

will be large. Also when performing the back substitution for

$$x_k = \left(\widetilde{b}_k - \sum_{j=k+1}^{n} a_{kj}^{(k)} \right) \Big/ a_{kk}^{(k)},$$

where $\widetilde{b}_k$ is the modified b_k by algorithm 3.5, any error in the numerator will be dramatically increased when dividing by a small $a_{kk}^{(k)}$.

To ensure that no large entries appear in the computed triangular factors, one can choose a pivot element to be the largest entry among $|a_{kk}^{(k)}|, \ldots, |a_{nk}^{(k)}|$. Then permute this row with the k-th row. The multipliers ℓ_{ik} are computed after the permutation. Suppose E_k denotes the permutation matrix at the k-th step. The permutation strategy can be formulated as

$$|(E_k M_{k-1} \cdots M_1 E_1 A)_{kk}| = \max_{k \le i \le n} |(E_k M_{k-1} \cdots M_1 E_1 A)_{ik}|.$$

As a consequence, no multipliers is greater than one in magnitude. Upon completion, we obtain an upper triangular matrix

$$U \equiv M_{n-1} E_{n-1} \cdots M_1 E_1 A. \tag{3.19}$$

This row interchange strategy is called partial pivoting. Gaussian elimination with partial pivoting leads to the LU factorization

$$PA = LU, \tag{3.20}$$

where P is a permutation matrix, L is unit lower triangular, and U is upper triangular.

While in the partial pivoting algorithm, the k-th pivot is determined by scanning the current k-th subcolumn $A^{(k)}(k:n,k)$, another pivoting strategy called complete pivoting searches for the largest entry in magnitude in the current submatrix $A^{(k)}(k:n,k:n)$ and permutes to the (k,k) position. That is, at the k-th step two permutation matrices E_k and F_k are determined so that

$$\left|(E_k A^{(k)} F_k)_{kk}\right| = \max_{k\le i,j\le n} \left|(A^{(k)})_{ij}\right|.$$

Gaussian elimination with complete pivoting leads to the LU factorization

$$PAQ = LU, \tag{3.21}$$

where P,Q are permutation matrices, L is unit lower triangular, and U is upper triangular.

In practical implementation, no permutation matrix should ever been explicitly formed. It can be efficiently represented by an integer n-vector. Usually an integer array p is used for partial pivoting to keep the indices of rows such that $p(k)$ stores the row index which interchanges with row k. In other words, (3.10) becomes

$$a^{(k+1)}_{p(i),j} = \begin{cases} a^{(k)}_{p(i),j}, & \text{for } i=1,\dots,k, \text{ and } j=1,\dots,n; \\ a^{(k)}_{p(i),j} - \frac{a^{(k)}_{p(i),k}}{a^{(k)}_{p(k),k}} \star a^{(k)}_{p(k),j}, & \text{for } i=k+1,\dots,n, \text{ and } j=k+1,\dots,n; \\ 0, & \text{for } i=k+1,\dots,n, \text{ and } j=k. \end{cases} \tag{3.22}$$

Two interger arrays will be required to store the permutation information for complete pivoting.

It should be noted that although there is no floating-point arithmetic involved when a permutation is applied to a matrix, only row/columns exchanged, it often causes irregular data movement and significant computational overhead. The LU factorization algorithm incorporated with partial pivoting is as follows. The one for complete pivoting is similar.

Algorithm 3.8 (LU-factorization with Partial Pivoting) *Given a nonsingular square matrix $A \in \mathbb{R}^{n\times n}$, this algorithm finds an appropriate permutation matrix P, and computes a unit lower triangular matrix L and an upper triangular matrix U such that $PA = LU$. The matrix A is overwritten by L and U, the matrix P is not formed. An integer array p is instead used for storing the row/column indices.*

```
for i = 1, ... , n do
   p(i) = i        % initialize the pivoting vector
end for
for k = 1, ... , n − 1 do
   m = k
   for i = k + 1, ... , n do
      if |A(p(m), k)| < |A(p(i), k)| then
         m = i
      end if
   end for
   ℓ = p(k)
   p(k) = p(m)
   p(m) = ℓ
   for i = k + 1, ... , n do
      A(p(i), k) = A(p(i), k)/A(p(k), k)
      for j = k + 1, ... , n do
         A(p(i), j) = A(p(i), j) − A(p(i), k) ⋆ A(p(k), j)
      end for
   end for
end for
```

Since the Gaussian elimination with partial pivoting produces the factorization (3.20), the linear system problem should comply accordingly.

$$Ax = b \Longrightarrow PAx = Pb \Longrightarrow LUx = Pb.$$

Hence the permutation needs to be applied to the right-hand-side vetor b, and the forward substitution should be modified to

Algorithm 3.9 (Permuted Forward Substitution) *Given an unit lower triangular matrix L, a permutation matrix P, and an n-vector b, this algorithm solves the triangular system $Ly = Pb$. The matrix L is stored in the strictly lower triangular part of matrix A, and the premutation matrix P is represented by an integer array p.*

```
y(p(1)) = b(p(1))/A(p(1), 1)
for i = 1, ... , n do
   t = 0
   for j = 1, ... , i − 1 do
      t = t + A(p(i), j) ⋆ y(p(j))
   end for
   y(p(i)) = b(p(i)) − t
end for
```

3.3.3 Scaled Row Pivoting

Let's consider another example of solving the system of linear equations

$$\begin{bmatrix} 1 & \frac{1}{\varepsilon} \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{\varepsilon} \\ 2 \end{bmatrix},$$

where $\varepsilon << \epsilon_M$. The algorithm of Gaussian elimination with or without pivoting would give

$$\begin{bmatrix} 1 & \frac{1}{\varepsilon} \\ 0 & 1-\frac{1}{\varepsilon} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{\varepsilon} \\ 2-\frac{1}{\varepsilon} \end{bmatrix} \Longrightarrow \begin{bmatrix} 1 & \frac{1}{\varepsilon} \\ 0 & -\frac{1}{\varepsilon} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{\varepsilon} \\ -\frac{1}{\varepsilon} \end{bmatrix},$$

and result

$$\begin{cases} x_2 = \frac{-1/\varepsilon}{-1/\varepsilon} = 1, \\ x_1 = 1/\varepsilon - 1/\varepsilon \cdot x_2 = 0. \end{cases}$$

But

$$\begin{bmatrix} 1 & \frac{1}{\varepsilon} \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 & \frac{1}{\varepsilon} \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\varepsilon} \\ 1 \end{bmatrix}.$$

What causes the problem? We observe that the pivot element is too small relative to the other entries in the pivot row. Now we introduce another pivoting strategy called scaled row pivoting.

First we let

$$s_i = \max_{1 \le j \le n} |a_{ij}|, \quad i = 1, \ldots, n,$$

and pivoting vector $p = (1, 2, \ldots, n)$. At each Gaussian elimination step k, we scan the number

$$\frac{|a_{p(i),k}|}{s_{p(i)}}, \quad i = k, \ldots, n,$$

looking for a maximal one. Suppose m is the index of the maximal ratio, we exchange rows $p(k)$ and $p(m)$, and use the new $a_{p(k),k}$ as the pivot element and row $p(k)$ as the pivot row to zero entries $a_{p(k+1),k}$ throught $a_{p(n),k}$. This procedure is repeated for $k = 1, \ldots, n-1$.

The following example illustrates the application of Gaussian elimination with scaled row pivoting.

Example 3.2 *Consider solving the linear system of equations in Example 3.1 using Gaussian elimination with scaled row pivoting.*

Sol: Initially, we have

$$A = \begin{bmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{bmatrix}, \quad p = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}, \quad s = \begin{bmatrix} 6 \\ 12 \\ 13 \\ 18 \end{bmatrix}.$$

Step through the Gaussian elimination with scaled row pivoting we have

$k = 1$ **:** Scan $\frac{6}{6}, \frac{12}{12}, \frac{3}{13}, \frac{6}{18}$, choose row 1 as the pivot row, vector p unchanged.

$$A = \begin{bmatrix} 6 & -2 & 2 & 4 \\ 2 & -4 & 2 & 2 \\ \frac{1}{2} & -12 & 8 & 1 \\ -1 & 2 & 3 & -14 \end{bmatrix}, \quad p = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}.$$

$k = 2$ **:** Scan $\frac{4}{12}, \frac{12}{13}, \frac{2}{18}$, choose row 3 as the pivot row.

$$A = \begin{bmatrix} 6 & -2 & 2 & 4 \\ 2 & \frac{1}{3} & -\frac{2}{3} & \frac{5}{3} \\ \frac{1}{2} & -12 & 8 & 1 \\ -1 & -\frac{1}{6} & \frac{13}{3} & -\frac{83}{6} \end{bmatrix}, \quad p = \begin{bmatrix} 1 \\ 3 \\ 2 \\ 4 \end{bmatrix}.$$

$k = 3$ **:** Scan $\frac{2/3}{12}, \frac{13/3}{18}$, choose row 4 as the pivot row.

$$A = \begin{bmatrix} 6 & -2 & 2 & 4 \\ 2 & \frac{1}{3} & -\frac{2}{13} & -\frac{18}{39} \\ \frac{1}{2} & -12 & 8 & 1 \\ -1 & -\frac{1}{6} & \frac{13}{3} & -\frac{83}{6} \end{bmatrix}, \quad p = \begin{bmatrix} 1 \\ 3 \\ 4 \\ 2 \end{bmatrix}.$$

Note that the entries of A are not actually moved and some positions of A are overwritten by the multipliers. If we let

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

denote the permutation matrix and

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 0 & 0 \\ -1 & -\frac{1}{6} & 1 & 0 \\ 2 & \frac{1}{3} & -\frac{2}{13} & 1 \end{bmatrix}, \qquad U = \begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -12 & 8 & 1 \\ 0 & 0 & \frac{13}{3} & -\frac{83}{6} \\ 0 & 0 & 0 & -\frac{18}{39} \end{bmatrix},$$

then one can verify that $PA = LU$. ■

Algorithm 3.10 (Gaussian Elimination with Partial Pivoting) *Given a nonsingular square matrix $A \in \mathbb{R}^{n\times n}$, this algorithm performs the Gaussian elimination with scaled row pivoting. The matrix A is modified and the multipliers are stored in A. An integer array p is used to store the permutation information.*

for $i = 1, \ldots, n$ **do**
  $p(i) = i$
  $s(i) = \max_{1 \le j \le n} |a_{ij}|$
end for
for $k = 1, \ldots, n-1$ **do**
  $m = k$
  for $i = k+1, \ldots, n$ **do**
    if $|A(p(m), k)|/s(p(m)) \quad < \quad |A(p(i), k)|/s(p(i))$
    then
      $m = i$
    end if
  end for
  $\ell = p(k)$
  $p(k) = p(m)$
  $p(m) = \ell$
  for $i = k+1, \ldots, n$ **do**
    $A(p(i), k) = A(p(i), k)/A(p(k), k)$
    for $j = k+1, \ldots, n$ **do**
      $A(p(i), j) = A(p(i), j) - A(p(i), k) \star A(p(k), j)$
    end for
  end for
end for

Algorithm 3.11 (Updating & Back Substitution) *Given $b \in \mathbb{R}^n$, an integer array p, and $A \in \mathbb{R}^{n \times n}$, where A is the resulting matrix after Gaussian elimination with scaled row pivoting, this algorithm solves $Ax = b$ for x.*

for $k = 1, \ldots, n-1$ **do**
  for $i = k+1, \ldots, n$ **do**
    $b(p(i)) = b(p(i)) - A(p(i), k) \star b(p(k))$
  end for
end for
for $i = n, \ldots, 1$ **do**
  $t = 0$
  for $j = i+1, \ldots, n$ **do**
    $t = t + A(p(i), j) \star x(j)$
  end for
  $x(i) = (b(p(i)) - t)/A(p(i), i)$
end for

In summary, it costs $\frac{2}{3}n^3 + O(n^2)$ flops to solve the linear system of equations $Ax = b$ using Gaussian elimination with scaled row pivoting. Note that it would take $\frac{4}{3}n^3 + O(n^2)$ just to compute A^{-1}.

3.4 Some Special Linear Systems

In this section, we consider some linear systems in which the coefficient matrix A has special property or structure.

3.4.1 Symmetric Positive Definite System and Cholesky Factorization

An $n \times n$ matrix A is positive definite if $x^T Ax > 0$ for all $x \in \mathbb{R}^n$, $x \neq 0$. A positive definite matrix is nonsingular and all diagonal entries are positive.

Lemma 3.1 *If $A \in \mathbb{R}^{n\times n}$ is positive definite, then A is nonsingular and $a_{ii} > 0$ for $i = 1, \ldots, n$.*

Proof: Suppose A is singular. Then there exists $x \in \mathbb{R}^n$ and $x \neq 0$ such that $Ax = 0$. This implies $x^T Ax = 0$, which contradicts the fact that A is positive definite. Therefore A is nonsingular. It is also easy to see that

$$a_{ii} = e_i^T Ae_i > 0,$$

where e_i is the i-th column of the $n \times n$ identify matrix. ∎

Lemma 3.2 *If $A \in \mathbb{R}^{n\times n}$ is positive definite, then all leading principal submatrices of A are nonsingular.*

Proof: For $1 \leq k \leq n$, let $x = [x_1, \ldots, x_k, 0, \ldots, 0]^T \in \mathbb{R}^n$ and $z_k = [x_1, \ldots, x_k]^T \in \mathbb{R}^k$, where $x_1, \ldots, x_k$ are not all zero. Since A is positive definite,

$$x^T Ax = z_k^T A_k z_k > 0,$$

where A_k is the $k \times k$ leading principal submatrix of A. This shows that A_k are also positive definite. Hence A_k are nonsingular. ∎

Theorem 3.3 *If $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite, then A has the Cholesky factorization*

$$A = GG^T, \tag{3.23}$$

where G is a lower triangular matrix with positive diagonals and is called the Cholesy factor of A.

Proof: From the previous Lemma, all leading principal submatrices of a positive definite matrix are nonsingular, A has the LU factorization $A = LU$, where L is unit lower triangular and U is upper triangular. Since A is symmetric,

$$LU = A = A^T = U^T L^T \quad \Longrightarrow \quad U(L^T)^{-1} = L^{-1}U^T.$$

Note that $U(L^T)^{-1}$ is upper triangular and $L^{-1}U^T$ is lower triangular, this forces $U(L^T)^{-1}$ to be a diagonal matrix, say, $U(L^T)^{-1} = D$. Then $U = DL^T$. Hence

$$A = LDL^T.$$

Since A is positive definite,

$$x^T Ax > 0 \quad \Longrightarrow \quad x^T LDL^T x = (L^T x)^T D(L^T x) > 0.$$

This means D is also positive definite, and hence $d_{ii} > 0$. Thus $D^{1/2}$ is well-defined and we have

$$A = LDL^T = LD^{1/2}D^{1/2}L^T \equiv GG^T.$$

■

Algorithm 3.12 *Given an $n \times n$ symmetric positive definite matrix A, this algorithm computes the Cholesky factorization $A = GG^T$.*

Initialize $G = 0$
for $k = 1, \ldots, n$ **do**
$$G(k,k) = \sqrt{A(k,k) - \sum_{j=1}^{k-1} G(k,j) \star G(k,j)}$$
for $i = k+1, \ldots, n$ **do**
$$G(i,k) = \left(A(i,k) - \sum_{j=1}^{k-1} G(i,j) \star G(k,j) \right) \Big/ G(k,k)$$
end for
end for

Cholesky factorization is a special case of LU factorization for symmetric positive definite matrices. In the k-th step of the algorithm, we have

$$g_{kk} = \sqrt{a_{kk} - \sum_{j=1}^{k-1} g_{kj}^2}. \tag{3.24}$$

This implies that

$$a_{kk} = \sum_{j=1}^{k} g_{kj}^2 \geq g_{km}, \quad m = 1, \ldots, k.$$

Hence

$$|g_{km}| \leq \sqrt{a_{kk}}, \quad \forall\, m = 1, \ldots, k. \tag{3.25}$$

This means that the elements of G will not grow large relative to the diagonals of A. Therefore, the Cholesky factorization is a stable numerical algorithm.

3.4.2 Diagonally Dominant Systems

A matrix $A \in \mathbb{R}^{n\times n}$ is said to be diagonally dominant if

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}|.$$

In this subsection we will show that pivoting is not necessary when Gaussian elimination is applied to a such matrix.

Lemma 3.3 *If $A \in \mathbb{R}^{n\times n}$ is diagonally dominant, then A is nonsingular.*

Proof: Suppose A is singular. Then there exists $x \in \mathbb{R}^n$, $x \neq 0$ such that $Ax = 0$. Let k be the integer index such that

$$|x_k| = \max_{1\le i\le n} |x_i| \quad \Longrightarrow \quad \frac{|x_i|}{|x_k|} < 1, \quad \forall\, i \neq k.$$

Since $Ax = 0$, we have

$$\sum_{j=1}^{n} a_{kj}x_j = 0 \quad \Longrightarrow \quad a_{kk}x_k = -\sum_{j=1, j\neq k}^{n} a_{kj}x_j.$$

Hence

$$|a_{kk}||x_k| \le \sum_{j=1, j\neq k}^{n} |a_{kj}||x_j| \quad \Longrightarrow \quad |a_{kk}| \le \sum_{j=1, j\neq k}^{n} |a_{kj}|\frac{|x_j|}{|x_k|} < \sum_{j=1, j\neq k}^{n} |a_{kj}|.$$

But this contradicts the assumption that A is diagonally dominant. Therefore A must be nonsingular. ∎

Theorem 3.4 *Gaussian elimination without pivoting preserve the diagonal dominance of a matrix.*

Proof: Let $A \in \mathbb{R}^{n\times n}$ be a diagonally dominant matrix and $A^{(2)} = [a_{ij}^{(2)}]$ is the result of applying one step of Gaussian elimination to A without any pivoting strategy.

Since the first row is unchanged, the property $a_{11}^{(2)} > \sum_{j=2}^{n} |a_{1j}^{(2)}|$ is preserved. Moreover $a_{i1}^{(2)} = 0$ for $i = 2, \dots, n$. All we need to show is that

$$a_{ii}^{(2)} > \sum_{j=2, j\neq i}^{n} |a_{ij}^{(2)}|, \quad \text{for} \quad i = 2, \dots, n.$$

Using the Gaussian elimination formula, we have

$$
\begin{aligned}
|a_{ii}^{(2)}| &= \left| a_{ii} - \frac{a_{i1}}{a_{11}} \star a_{1i} \right| \\
&\geq |a_{ii}| - \frac{|a_{i1}|}{|a_{11}|} \star |a_{1i}| \\
&= |a_{ii}| - |a_{i1}| + |a_{i1}| - \frac{|a_{i1}|}{|a_{11}|} \star |a_{1i}| \\
&= |a_{ii}| - |a_{i1}| + \frac{|a_{i1}|}{|a_{11}|} \star (|a_{11}| - |a_{1i}|) \\
&> \sum_{j=2, j\neq i}^{n} |a_{ij}| + \frac{|a_{i1}|}{|a_{11}|} \sum_{j=2, j\neq i}^{n} |a_{1j}| \\
&= \sum_{j=2, j\neq i}^{n} |a_{ij}| + \sum_{j=2, j\neq i}^{n} \frac{|a_{i1}|}{|a_{11}|} |a_{1j}| \\
&\geq \sum_{j=2, j\neq i}^{n} \left| a_{ij} - \frac{a_{i1}}{a_{11}} a_{1j} \right| \\
&= \sum_{j=2, j\neq i}^{n} |a_{ij}^{(2)}|
\end{aligned}
$$

■

Theorem 3.5 *If the Gaussian elimination with scaled row pivoting algorithm is applied to a diagonally dominant matrix $A \in \mathbb{R}^{n\times n}$, then the pivot vector will be $p = [1, 2, \ldots, n]^T$. That is, no permutation is needed.*

Proof: It is sufficient to prove that the first pivot row chosen is row 1. That is, to prove

$$
\frac{|a_{11}|}{s_1} > \frac{|a_{i1}|}{s_i}, \quad \forall\, i = 2, \ldots, n.
$$

Because of the diagonal dominance,

$$
s_i = \max_{1\leq j\leq n} |a_{ij}| = |a_{ii}|,
$$

hence

$$
\frac{|a_{11}|}{s_1} = \frac{|a_{11}|}{|a_{11}|} = 1, \quad \text{and} \quad \frac{|a_{i1}|}{s_i} = \frac{|a_{i1}|}{|a_{11}|} < 1, \quad \forall\, i \geq 2.
$$

Thus the theorem is proved. ■

3.4.3 Tridiagonal Systems

We consider solving the linear system $Tx = b$, where $T \in \mathbb{R}^{n\times n}$ is tridiagonal of the form

$$T = \begin{bmatrix} d_1 & c_1 & & & \\ a_2 & d_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ & & & a_n & d_n \end{bmatrix}. \tag{3.26}$$

Tridiagonal linear systems arise in many applications.

A tridiagonal matrix can be stored in three n-vectors. Suppose that pivoting is not necessary in solving sytems, e.g., T is symmetric positive definite or diagonally dominant, the Gaussian elimination procedure can be simplified greatly.

Algorithm 3.13 (Thomas Algorithm) *This algorithm solving a tridiagonal linear sytem $Tx = b$ of the form (3.26) using Gaussian elimination without pivoting.*

for $i = 2, \ldots, n$ **do**
  $d(i) = d(i) - \frac{a(i)}{d(i-1)} \star c(i-1)$
  $b(i) = b(i) - \frac{a(i)}{d(i-1)} \star b(i-1)$
end for
$x(n) = b(n)/d(n)$
for $i = n-1, \ldots, 1$ **do**
  $x(i) = (b(i) - c(i) \star x(i+1))/d(i)$
end for

If pivoting is necessary, then, in addition to the pivoting vector, an extra vector is stored the entries appear because of pivoting.

3.5 Perturbation Analysis

Let $A \in \mathbb{R}^{n\times n}$ be nonsingular. If A^{-1} is perturbed to C so that the solution of system of linear equations $x = A^{-1}b$ is perturbed to $\widetilde{x} = Cb$, then

$$\begin{aligned} \|x - \widetilde{x}\| &= \|x - Cb\| = \|x - CAx\| = \|(I - CA)x\| \\ &\leq \|I - CA\|\|x\|. \end{aligned}$$

Hence

$$\frac{\|x - \widetilde{x}\|}{\|x\|} \leq \|I - CA\| \tag{3.27}$$

Now suppose that b is perturbed to $\widetilde{b}$ and x and $\widetilde{x}$ satisfy $Ax = b$ and $A\widetilde{x} = \widetilde{b}$. Then

$$\begin{aligned}
\|x - \widetilde{x}\| &= \|A^{-1}b - A^{-1}\widetilde{b}\| = \|A^{-1} - (b - \widetilde{b})\| \\
&\leq \|A^{-1}\|\|b - \widetilde{b}\| = \|A^{-1}\|\|b\|\frac{\|b - \widetilde{b}\|}{\|b\|} = \|A^{-1}\|\|Ax\|\frac{\|b - \widetilde{b}\|}{\|b\|} \\
&\leq \|A^{-1}\|\|A\|\|x\|\frac{\|b - \widetilde{b}\|}{\|b\|}
\end{aligned}$$

Therefore

$$\frac{\|x - \widetilde{x}\|}{\|x\|} \leq \kappa(A)\frac{\|b - \widetilde{b}\|}{\|b\|}, \tag{3.28}$$

where

$$\kappa(A) = \|A\|\|A^{-1}\|$$

is called the condition number of A.

Definition 3.2 *Let $\widetilde{x}$ be the computed solution to the linear system of equations $Ax = b$. Then the vector*

$$r = b - A\widetilde{x} \tag{3.29}$$

is called the residual vector.

If $e = x - \widetilde{x}$ denotes the error vector, then

$$Ae = Ax - A\widetilde{x} = b - A\hat{x} = r. \tag{3.30}$$

With the definition of residual vector, we can derive

$$\|r\|\|x\| = \|Ae\|\|A^{-1}b\| \leq \|A\|\|A^{-1}\|\|e\|\|b\| = \kappa(A)\|x - \widetilde{x}\|\|b\|.$$

Hence

$$\frac{1}{\kappa(A)}\frac{\|r\|}{\|b\|} \leq \frac{\|x - \widetilde{x}\|}{\|x\|}$$

Also

$$\frac{\|x - \widetilde{x}\|}{\|x\|} \leq \kappa(A)\frac{\|b - \widetilde{b}\|}{\|b\|} = \kappa(A)\frac{\|b - A\widetilde{x}\|}{\|b\|} = \kappa(A)\frac{\|r\|}{\|b\|}.$$

Therefore

$$\frac{1}{\kappa(A)}\frac{\|r\|}{\|b\|} \leq \frac{\|x - \widetilde{x}\|}{\|x\|} \leq \kappa(A)\frac{\|r\|}{\|b\|}. \tag{3.31}$$