2. NUMERICAL METHODS AND ALGORITHMS OF LINEAR ALGEBRA

This chapter discusses basic numerical methods and algorithms of computational mathematics in the field of linear algebra. At first, numerical methods for solving systems of linear algebraic equations are presented. Direct (sweep method, Gaussian elimination method) and iterative (fixed-point iteration method, Seidel method) methods are described. This problem is called the first basic problem in linear algebra. Problems of calculating the determinant and elements of the inverse matrix are linked to it and are sometimes called the second and third basic problems of linear algebra. The last topics of this chapter are devoted to the problem of finding matrix eigenvalues and eigenvectors (Jacobi rotation method, power iteration method). It is this problem in linear algebra that is usually called the second basic problem.

2.1. SYSTEMS OF LINEAR ALGEBRAIC EQUATIONS

Suppose it is required to find a solution to the system of linear algebraic equations (SLAE):

$$Ax = b,$$

where $A \equiv \left[a_{ij} \right]$ is the square $n \times n$ matrix of coefficients with unknowns; $x \equiv \left[x_j \right]$ is the column vector of the unknowns; $b \equiv \left[b_j \right]$ is the column vector of the right-hand sides of the system of linear algebraic equations:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n = b_2 \\ ..... \\ a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n = b_n \end{cases}.$$

Systems of linear equations could be treated as a special case of systems of nonlinear equations. However, the relative simplicity of linear systems has led to the emergence of special highly efficient methods for their solution.

At the same time, linear systems are of great independent interest for two reasons. First, many practical problems lead to systems of linear equations.

Second, almost all sections of computational mathematics are an important and inexhaustible source of systems of linear algebraic equations: finding a solution to a system of linear algebraic equations is necessary in problems related to refining the roots of systems of nonlinear equations, approximation of functions, finding eigenvalues and eigenvectors of a matrix, solving ordinary differential equations and partial differential equations, etc. Methods for solving systems of linear equations are also used for some classes of integral and integro-differential equations.

From the point of view of the classical theory of systems of linear algebraic equations, their solution does not give rise to any difficulties. The simplest analytical method for solving a SLAE is the method of successive elimination of unknowns.

Example. Let's solve the system using successive elimination of unknowns:

$$\begin{cases} x_1 + 3x_2 - 2x_3 = 5 \\ 3x_1 + 5x_2 + 6x_3 = 7 \\ 2x_1 + 4x_2 + 3x_3 = 8 \end{cases} \rightarrow \begin{cases} x_1 = -3x_2 + 2x_3 + 5 \\ 3x_1 + 5x_2 + 6x_3 = 7 \\ 2x_1 + 4x_2 + 3x_3 = 8 \end{cases} \rightarrow \begin{cases} x_1 = -3x_2 + 2x_3 + 5 \\ -4x_2 + 12x_3 = -8 \\ -2x_2 + 7x_3 = -2 \end{cases} \rightarrow$$

$$\rightarrow \begin{cases} x_1 = -3x_2 + 2x_3 + 5 \\ x_2 = 3x_3 + 2 \\ -2x_2 + 7x_3 = -2 \end{cases} \rightarrow \begin{cases} x_1 = -3x_2 + 2x_3 + 5 \\ x_2 = 3x_3 + 2 \\ x_3 = 2 \end{cases} \rightarrow \begin{cases} x_1 = -3x_2 + 2x_3 + 5 \\ x_2 = 8 \\ x_3 = 2 \end{cases} \rightarrow \begin{cases} x_1 = -15 \\ x_2 = 8 \\ x_3 = 2 \end{cases}.$$

Besides, it is known from linear algebra that, according to the Cramer's rule, a system of *n* linear algebraic equations with *n* unknowns has the unique solution if the determinant of the matrix of the system is nonzero ($\det A \neq 0$) and the value of each of the unknowns is calculated as the ratio of two determinants of *n*-th order:

$$x_j = \frac{\det A_j}{\det A}, \quad j = 1, ..., n,$$

where $\det A_j$ is the determinant of the matrix obtained by replacing the *j*-th column of the matrix *A* with the column of the right-hand sides.

Example. Let's solve the system using the Cramer's method:

$\begin{cases} x_1 + 3x_2 - 2x_3 = 5 \\ 3x_1 + 5x_2 + 6x_3 = 7 \\ 2x_1 + 4x_2 + 3x_3 = 8 \end{cases}$. Here $A = \begin{pmatrix} 1 & 3 & -2 \\ 3 & 5 & 6 \\ 2 & 4 & 3 \end{pmatrix}$, $b = \begin{pmatrix} 5 \\ 7 \\ 8 \end{pmatrix}$, then:

$$x_1 = \frac{\begin{vmatrix} 5 & 3 & -2 \\ 7 & 5 & 6 \\ 8 & 4 & 3 \end{vmatrix}}{\begin{vmatrix} 1 & 3 & -2 \\ 3 & 5 & 6 \\ 2 & 4 & 3 \end{vmatrix}} = \frac{60}{-4} = -15, \quad x_2 = \frac{\begin{vmatrix} 1 & 5 & -2 \\ 3 & 7 & 6 \\ 2 & 8 & 3 \end{vmatrix}}{\begin{vmatrix} 1 & 3 & -2 \\ 3 & 5 & 6 \\ 2 & 4 & 3 \end{vmatrix}} = \frac{-32}{-4} = 8, \quad x_3 = \frac{\begin{vmatrix} 1 & 3 & 5 \\ 3 & 5 & 7 \\ 2 & 4 & 8 \end{vmatrix}}{\begin{vmatrix} 1 & 3 & -2 \\ 3 & 5 & 6 \\ 2 & 4 & 3 \end{vmatrix}} = \frac{-8}{-4} = 2.$$

However, in the course of direct calculation of determinants as an algebraic sum of $n!$ products of elements, around $n \cdot n!$ arithmetic operations such as multiplication will be required to find a solution to the system of linear algebraic equations according to the Cramer's rule, which is algorithmically extremely inefficient.

Therefore, further we shall discuss the most common direct and iterative methods. Direct methods give a solution to a problem for a finite (precisely determined for each method) number of operations. The term “exact methods” is not used here intentionally, because due to rounding errors that occur on any computer in calculations with a finite number of digits, the exact solution cannot be reached and it always turns out to have errors. Among direct methods, we shall consider the sweep method for SLAE with a tridiagonal matrix and the Gaussian method with its modifications for general systems of linear algebraic equations.

Iterative methods give a solution as the limit of an infinite sequence of approximate solutions, in which each subsequent more accurate approximation is based on the already found previous one (or ones). Among iterative methods for solving systems of linear algebraic equations, we shall consider the fixed-point iteration method and the Seidel method.

2.2. MATRIX CONDITIONALITY

Another important circumstance associated with the solution to systems of linear algebraic equations is as follows. From the point of view of the theory of linear systems, these two cases are distinguished: when the determinant of the system matrix is not equal to zero ($\det A \neq 0$), i.e. SLAE is nondegenerate, and when the determinant of the system matrix is equal to zero ($\det A = 0$), meaning the

system is degenerate. In the latter case, the system either does not have a solution (if $b \neq 0$) or has an infinite number of solutions (if $b = 0$). However, from the point of view of practical calculations, there are “almost degenerate” SLAEs – systems of equations in which the determinant is close to zero, but is still different from zero ($\det A \approx 0$). Small changes in the system's matrix coefficients or the system's right-hand sides of “almost degenerate” systems can lead to large solution errors.

All these cases are well illustrated by the example of solving a system of two linear equations:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 = b_1 \\ a_{21}x_1 + a_{22}x_2 = b_2 \end{cases}.$$

In fig. 1, each equation corresponds to a line on the plane (x_1, x_2), and the intersection point of these lines is a solution to the system.

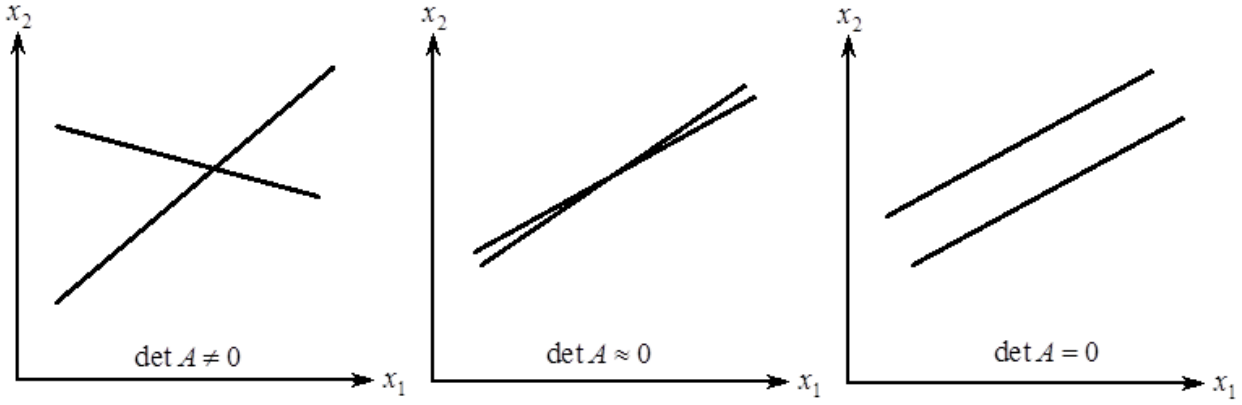

Fig. 1. Geometric interpretation of a system of two linear equations

If $\det A = 0$, then the slopes of the lines are equal and they are either parallel or coincide. If $\det A \approx 0$, then small errors in the coefficients and right-hand sides can lead to large errors in the solution, i.e. the position of the intersection point.

Systems of this type, in which small errors in the system's coefficients or right-hand sides (these errors can, in particular, result from rounding when calculating or storing numbers in the computer memory) lead to large errors in the solution, are called ill-conditioned. An ill-conditioned system geometrically corresponds to almost parallel lines.

Definition. To study errors that arise when solving SLAEs, the concept of the *condition number* of the matrix *cond*(*A*) is introduced:

$cond(A) = \|A\| \cdot \|A^{-1}\|$.

At any norm $cond(A) \geq 1$: $cond(A) = \|A\| \cdot \|A^{-1}\| \geq \|A \cdot A^{-1}\| = \|E\| = 1$.

Example. Let's calculate the condition number of the matrix $A = \begin{pmatrix} -1 & 2 \\ 3 & -5 \end{pmatrix}$ using $\|\cdot\|_c$ as the matrix norm.

First, $\|A\|_c = \max(|-1| + |2|, |3| + |-5|) = 8$.

Second, let's find the inverse matrix and calculate its norm:

$A^{-1} = \begin{pmatrix} 5 & 2 \\ 3 & 1 \end{pmatrix}$, $\|A^{-1}\|_c = \max(|5| + |2|, |3| + |1|) = 7$.

As a result: $cond(A) = \|A\|_c \cdot \|A^{-1}\|_c = 8 \cdot 7 = 56$.

The condition number characterizes the level of dependence of the relative error of the solution to the SLAE on the relative error of the input data (right-hand sides and matrix elements). It can be shown that the following inequalities are valid for nonzero vectors *x*:

$$\frac{\|\Delta x\|}{\|x\|} \leq cond(A) \frac{\|\Delta b\|}{\|b\|}, \quad \frac{\|\Delta x\|}{\|x\|} \leq cond(A) \frac{\|\Delta A\|}{\|A + \Delta A\|}.$$

The larger this number is, the worse the conditionality of the system is; thus, when $cond(A) >> 10^2$, the system is already ill-conditioned. So, the larger the condition number is, the higher the influence of the input data error on the final result of the solution to the SLAE is. In practice, however, calculators are often limited to checking the condition $\det A \approx 0$.

Example. Let's calculate the condition number of the matrix $A = \begin{pmatrix} 1 & 10 \\ 100 & 1001 \end{pmatrix}$ using $\|\cdot\|_c$ as the matrix norm.

It is easy to see that $\det A = 1$.

Herewith $\|A\|_c = \max(|1| + |10|, |100| + |1001|) = 1101$.

Let's find the inverse matrix and calculate its norm:

$A^{-1} = \begin{pmatrix} 1001 & -10 \\ -100 & 1 \end{pmatrix}$, $\|A^{-1}\|_c = \max(|1001| + |-10|, |-100| + |1|) = 1011$.

As a result: $cond(A) = \|A\|_c \cdot \|A^{-1}\|_c = 1101 \cdot 1011 = 1113111 > 10^6$.

Let's solve the SLAE $Ax = b$, where $b = \begin{pmatrix} 11 \\ 1101 \end{pmatrix}$. It is easy to show that $x = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$.

Let's introduce a small perturbation into the vector of the right-hand sides of the original system of equations: $\tilde{b} = \begin{pmatrix} 11.01 \\ 1101 \end{pmatrix}$. Now the new solution is $\tilde{x} = \begin{pmatrix} 11.01 \\ 0 \end{pmatrix}$.

It is easy to see that for this ill-conditioned SLAE a small perturbation in the input data has led to a significant change in the solution. And indeed:

$$\frac{\|\Delta x\|}{\|x\|} = \frac{10.01}{1} = 10.01 \quad \leq \quad cond(A)\frac{\|\Delta b\|}{\|b\|} = 1113111 \cdot \frac{0.01}{1101} = 10.11.$$

So, if we change the input data (in this case the vector of the right-hand sides) by less than 0.001%, we can get a change in the output data (the solution to the SLAE) by more than 10 times from the input data if the original matrix of coefficients is ill-conditioned.

2.3. SWEEP METHOD

The sweep method (or Thomas algorithm) belongs to direct methods for solving the SLAE and is used in those cases where many of the matrix coefficients are equal to zero. This circumstance was taken into account when implementing the sweep method, in which transformations with zero elements are excluded.

Systems of linear algebraic equations with a tridiagonal matrix of coefficients with unknowns represent the most important and widespread case of special type systems. In such systems, only the elements located on the main diagonal and on the lower and upper diagonals adjacent to it are nonzero. Problems related to spline interpolation, solving ordinary differential equations and partial differential equations by difference methods result in systems with tridiagonal matrices. The sweep method is one of the efficient direct methods for solving SLAEs with tridiagonal matrices.

Let's consider the following SLAE with the tridiagonal matrix *A*:

$$a_1 = 0 \begin{cases} b_1 x_1 + c_1 x_2 = d_1 \\ a_2 x_1 + b_2 x_2 + c_2 x_3 = d_2 \\ \qquad a_3 x_2 + b_3 x_3 + c_3 x_4 = d_3 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \qquad\qquad a_{n-1} x_{n-2} + b_{n-1} x_{n-1} + c_{n-1} x_n = d_{n-1} \\ \qquad\qquad\qquad a_n x_{n-1} + b_n x_n = d_n, \qquad c_n = 0 \end{cases},$$

the solution of which we shall seek in the recurrent form (when each member of the sequence is expressed through one or more of the previous ones):

$$x_i = P_i x_{i+1} + Q_i, \, i = 1, \ldots, n,$$

where $P_i, Q_i, \, i = 1, \ldots, n$ are the sweep coefficients to be determined.

To determine them, we shall first express x_1 from the first equation of the SLAE through x_2 and shall obtain:

$$x_1 = \frac{-c_1}{b_1} x_2 + \frac{d_1}{b_1} = P_1 x_2 + Q_1, \text{ wherefrom } P_1 = \frac{-c_1}{b_1}, \, Q_1 = \frac{d_1}{b_1}.$$

From the second equation of the SLAE, we shall express x_2 through x_3 by substituting x_1 and shall obtain:

$$b_2 x_2 = -a_2 x_1 - c_2 x_3 + d_2 = -a_2 \left(P_1 x_2 + Q_1 \right) - c_2 x_3 + d_2, \; b_2 x_2 + a_2 P_1 x_2 = -a_2 Q_1 - c_2 x_3 + d_2,$$

$$x_2 = \frac{-c_2}{b_2 + a_2 P_1} x_3 + \frac{d_2 - a_2 Q_1}{b_2 + a_2 P_1} = P_2 x_3 + Q_2, \text{ wherefrom } P_2 = \frac{-c_2}{b_2 + a_2 P_1}, \, Q_2 = \frac{d_2 - a_2 Q_1}{b_2 + a_2 P_1}.$$

Continuing this process in a similar way (only the indices are changing), we shall obtain the following from the *i*-th equation of the SLAE:

$$x_i = \frac{-c_i}{b_i + a_i P_{i-1}} x_{i+1} + \frac{d_i - a_i Q_{i-1}}{b_i + a_i P_{i-1}}, \text{ therefore } P_i = \frac{-c_i}{b_i + a_i P_{i-1}}, \, Q_i = \frac{d_i - a_i Q_{i-1}}{b_i + a_i P_{i-1}}.$$

From the last equation of the SLAE we get the following:

$$x_n = \frac{d_n - a_n Q_{n-1}}{b_n + a_n P_{n-1}} = 0 \cdot x_{n+1} + Q_n, \text{ i.e. } P_n = 0 \text{ (as } c_n = 0), \; Q_n = \frac{d_n - a_n Q_{n-1}}{b_n + a_n P_{n-1}} = x_n.$$

So, the forward path of the sweep method for the determination of sweep coefficients $P_i, Q_i, \, i = 1, \ldots, n$ is completed. As a result, sweep coefficients are calculated by the following formulas:

$$P_1 = \frac{-c_1}{b_1}, \, Q_1 = \frac{d_1}{b_1}, \text{ as } a_1 = 0, \, i = 1;$$

$$P_i = \frac{-c_i}{b_i + a_i P_{i-1}},\ Q_i = \frac{d_i - a_i Q_{i-1}}{b_i + a_i P_{i-1}},\ i = 2,\ldots,n-1;$$

$$P_n = 0,\ \text{as}\ c_n = 0,\ Q_n = \frac{d_n - a_n Q_{n-1}}{b_n + a_n P_{n-1}},\ i = n.$$

The backward path of the sweep method to find a solution to the system is applied in accordance with the expression $x_i = P_i x_{i+1} + Q_i,\ i = 1,\ldots,n$:

$$\begin{cases} x_n = P_n x_{n+1} + Q_n = 0 \cdot x_{n+1} + Q_n = Q_n \\ x_{n-1} = P_{n-1} x_n + Q_{n-1} \\ x_{n-2} = P_{n-2} x_{n-1} + Q_{n-2} \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ x_1 = P_1 x_2 + Q_1 \end{cases}.$$

The formulas given above are right-sweep formulas. Similarly, starting the derivation of sweep coefficients from the last equation of the SLAE, we can derive left-sweep formulas.

The total number of executed arithmetic operations in the sweep method is 8*n*, i.e. it is proportional to the number of equations.

Such numerical methods, for which the number of arithmetic operations is proportional to the dimension *n*, are called *economical*.

Let's note that herewith the determinant of the tridiagonal matrix *A* of the SLAE is calculated in the process of the forward path as follows:

$$\det A = \prod_{i=1}^{n} [\, b_i + a_i P_{i-1}] = b_1 \cdot \prod_{i=2}^{n} [\, b_i + a_i P_{i-1}].$$

The sweep method is stable, if $|P_i| \le 1, i = 1,\ldots,n$. Based on that, it can be proven that the following conditions are sufficient for the computational stability of the sweep method:

$$a_i \ne 0,\ c_i \ne 0,\ i = 2,\ldots,n-1;\ |b_i| \ge |a_i| + |c_i|,\ i = 1,\ldots,n,$$

moreover, a strict inequality is preserved with at least one *i* (the condition of diagonal dominance of the matrix).

Here, the stability is understood in the sense of non-accumulation of the solution error during the computational process when rounding or with small errors in the input data (right-hand sides and matrix elements).

Example. Solve the SLAE using the sweep method, calculate the determinant for the matrix of the SLAE:

$$\begin{cases} 7x_1 - 3x_2 = 1 \\ -4x_1 + 9x_2 + 3x_3 = 23 \\ 3x_2 - 8x_3 + 4x_4 = -2 \\ -2x_3 + 7x_4 + 4x_5 = 42 \\ -5x_4 + 6x_5 = 10 \end{cases}$$. Here $a = \begin{pmatrix} 0 \\ -4 \\ 3 \\ -2 \\ -5 \end{pmatrix}$, $b = \begin{pmatrix} 7 \\ 9 \\ -8 \\ 7 \\ 6 \end{pmatrix}$, $c = \begin{pmatrix} -3 \\ 3 \\ 4 \\ 4 \\ 0 \end{pmatrix}$, $d = \begin{pmatrix} 1 \\ 23 \\ -2 \\ 42 \\ 10 \end{pmatrix}$.

Let's implement the forward path of the sweep method:

$$P_1 = \frac{-c_1}{b_1} = \frac{3}{7} = 0.429,\ Q_1 = \frac{d_1}{b_1} = \frac{1}{7} = 0.143;$$

$$P_2 = \frac{-c_2}{b_2 + a_2 P_1} = -0.412,\ Q_2 = \frac{d_2 - a_2 Q_1}{b_2 + a_2 P_1} = 3.235;$$

$$P_3 = \frac{-c_3}{b_3 + a_3 P_2} = 0.433,\ Q_3 = \frac{d_3 - a_3 Q_2}{b_3 + a_3 P_2} = 1.268;$$

$$P_4 = \frac{-c_4}{b_4 + a_4 P_3} = -0.652,\ Q_4 = \frac{d_4 - a_4 Q_3}{b_4 + a_4 P_3} = 7.261;$$

$$P_5 = 0,\ Q_5 = \frac{d_5 - a_5 Q_4}{b_5 + a_5 P_4} = 5.0.$$

Let's find the determinant of the matrix:

$$\det A = b_1 \cdot \prod_{i=2}^{5} \left[b_i + a_i P_{i-1} \right] = b_1 (b_2 + a_2 P_1)(b_3 + a_3 P_2)(b_4 + a_4 P_3)(b_5 + a_5 P_4) =$$

$$= 7(9 - 4 \cdot 0.429)(-8 - 3 \cdot 0.412)(7 - 2 \cdot 0.433)(6 + 5 \cdot 0.652) = -26754.$$

Let's implement the backward path of the sweep method:

$$x_5 = Q_5 = 5.0,\ x_4 = P_4 x_5 + Q_4 = 4.0,\ x_3 = P_3 x_4 + Q_3 = 3.0,$$

$$x_2 = P_2 x_3 + Q_2 = 2.0,\ x_1 = P_1 x_2 + Q_1 = 1.0.$$

Solution to the SLAE: $x = (1 \quad 2 \quad 3 \quad 4 \quad 5)^T$.

2.4. PROGRAM #01

Below is a proposed variant of the program algorithm for solving SLAE with a tridiagonal matrix and calculating the determinant of the SLAE's matrix using the sweep method.

```
ALGORITHM “Sweep method”
INPUT       n, a[n], b[n], c[n], d[n]
OUTPUT      i, p[n], q[n], x[n], y
BEGIN
        p[1]:=-c[1]/b[1]
        q[1]:=d[1]/b[1]
        y:=b[1]
        CYCLE “Forward Path” FOR i FROM 2 TO n BY 1
                p[i]:=-c[i]/(b[i]+a[i]*p[i-1])
                q[i]:=(d[i]-a[i]*q[i-1])/(b[i]+a[i]*p[i-1])
                y:=y*(b[i]+a[i]*p[i-1])
        x[n]:=q[n]
        CYCLE “Backward Path” FOR i FROM n-1 TO 1 BY -1
                x[i]:=p[i]*x[i+1]+q[i]
        PRINT x, y
END
```