2. Prove that for a square matrix $\boldsymbol{A}$ and its transpose $\boldsymbol{A}^T$, the following equalities hold: $\mu_\infty(\boldsymbol{A}) = \mu_1(\boldsymbol{A}^T)$, $\mu_1(\boldsymbol{A}) = \mu_\infty(\boldsymbol{A}^T)$.
3. Show that the condition number of the operator $\boldsymbol{A}$ does not change if the operator is multiplied by an arbitrary non-zero real number.
4. Let $\mathbb{L}$ be a Euclidean space, and let $\boldsymbol{A} : \mathbb{L} \longmapsto \mathbb{L}$. Show that the condition number $\mu_{\boldsymbol{B}}(\boldsymbol{A}) = 1$ if and only if at least one of the following conditions holds:

   a) $\boldsymbol{A} = \alpha \boldsymbol{I}$, where $\alpha \in \mathbb{R}$;

   b) $\boldsymbol{A}$ is an orthogonal operator, i.e., $\forall \boldsymbol{x} \in \mathbb{L} : [\boldsymbol{Ax}, \boldsymbol{Ax}]_{\boldsymbol{B}} = [\boldsymbol{x}, \boldsymbol{x}]_{\boldsymbol{B}}$.

   c) $\boldsymbol{A}$ is a composition of $\alpha \boldsymbol{I}$ and an orthogonal operator.

5.* Prove that $\mu_{\boldsymbol{B}}(\boldsymbol{A}) = \mu_{\boldsymbol{B}}(\boldsymbol{A}^*_{\boldsymbol{B}})$, where $\boldsymbol{A}^*_{\boldsymbol{B}}$ is the operator adjoint to $\boldsymbol{A}$ in the sense of the scalar product $[\boldsymbol{x}, \boldsymbol{y}]_{\boldsymbol{B}}$.

6. Let $\boldsymbol{A}$ be a non-singular matrix, $\det \boldsymbol{A} \neq 0$. Multiply one row of the matrix $\boldsymbol{A}$ by some scalar α, and denote the new matrix by $\boldsymbol{A}_\alpha$. Show that $\mu(\boldsymbol{A}_\alpha) \longrightarrow \infty$ as $\alpha \longrightarrow \infty$.

7.* Prove that for any linear operator $\boldsymbol{A} : \mathbb{L} \longmapsto \mathbb{L}$:

$$\mu_{\boldsymbol{B}}(\boldsymbol{A}^*_{\boldsymbol{B}} \boldsymbol{A}) = (\mu_{\boldsymbol{B}}(\boldsymbol{A}))^2,$$

   where $\boldsymbol{A}^*_{\boldsymbol{B}}$ is the operator adjoint to $\boldsymbol{A}$ in the sense of the scalar product $[\boldsymbol{x}, \boldsymbol{y}]_{\boldsymbol{B}}$.

8.* Let $\boldsymbol{A} = \boldsymbol{A}^* > 0$ and $\boldsymbol{B} = \boldsymbol{B}^* > 0$ in the sense of some scalar product introduced on the linear space $\mathbb{L}$. Let the following inequalities hold for every $\boldsymbol{x} \in \mathbb{L}$:

$$\gamma_1(\boldsymbol{Bx}, \boldsymbol{x}) \leq (\boldsymbol{Ax}, \boldsymbol{x}) \leq \gamma_2(\boldsymbol{Bx}, \boldsymbol{x}),$$

   where $\gamma_1 > 0$ and $\gamma_2 > 0$ are two real numbers. Consider the operator $\boldsymbol{C} = \boldsymbol{B}^{-1}\boldsymbol{A}$ and prove that the condition number $\mu_{\boldsymbol{B}}(\boldsymbol{C})$ satisfies the estimate:

$$\mu_{\boldsymbol{B}}(\boldsymbol{C}) \leq \frac{\gamma_2}{\gamma_1}.$$

   Remark. We will solve this problem in Section 6.1.4 as it has numerous applications.

5.4 Gaussian Elimination and Its Tri-Diagonal Version

We will describe both the standard Gaussian elimination algorithm and the Gaussian elimination with pivoting, as they apply to solving an $n \times n$ system of linear algebraic equations in its canonical form:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= f_1, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots&\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= f_n. \end{aligned} \tag{5.44}$$

Recall that the Gaussian elimination procedures belong to the class of direct methods, i.e., they produce the exact solution of system (5.44) after a finite number of arithmetic operations.

5.4.1 Standard Gaussian Elimination

From the first equation of system (5.44), express x_1 through the other variables:

$$x_1 = a'_{12}x_2 + \ldots + a'_{1n}x_n + f'_1. \tag{5.45}$$

Substitute expression (5.45) for x_1 into the remaining $n-1$ equations of system (5.44) and obtain a new system of $n-1$ linear algebraic equations with respect to $n-1$ unknowns: $x_2, x_3, \ldots, x_n$. From the first equation of this updated system express x_2 as a function of all other variables:

$$x_2 = a'_{23}x_3 + \ldots + a'_{2n}x_n + f'_2, \tag{5.46}$$

substitute into the remaining $n-2$ equations and obtain yet another system of further reduced dimension: $n-2$ equations with $n-2$ unknowns. Repeat this procedure, called elimination, for $k = 3, 4, \ldots, n-1$:

$$x_k = a'_{k,k+1}x_{k+1} + \ldots + a'_{kn}x_n + f'_k, \tag{5.47}$$

and every time obtain a new smaller linear system of dimension $(n-k) \times (n-k)$. At the last stage $k = n$, no substitution is needed, and we simply solve the 1×1 system, i.e., a single scalar linear equation from the previous stage $k = n-1$, which yields:

$$x_n = f'_n. \tag{5.48}$$

Having obtained the value of x_n by formula (5.48), we can find the values of the unknowns $x_{n-1}, x_{n-2}, \ldots, x_1$ one after another by employing formulae (5.47) in the ascending order: $k = n-1, n-2, \ldots, 1$. This procedure, however, is not fail-proof. It may break down because of a division by zero. Even if no division by zero occurs, the algorithm may still end up generating a large error in the solution due to an amplification of the small round-off errors. Let us explain these phenomena.

If the coefficient a_{11} in front of the unknown x_1 in the first equation of system (5.44) is equal to zero, $a_{11} = 0$, then already the first step of the elimination algorithm, i.e., expression (5.45), becomes invalid, because $a'_{12} = -a_{12}/a_{11}$. A division by zero can obviously be encountered at any stage of the algorithm. For example, having the coefficient in front of x_2 equal to zero in the first equation of the $(n-1) \times (n-1)$ system invalidates expression (5.46). But even if no division by zero is ever encountered, and formulae (5.47) are obtained for all $k = 1, 2, \ldots, n$, then the method can still develop a computational instability when solving for $x_n, x_{n-1}, \ldots, x_1$. For example, if it happens that $a'_{k,k+1} = 2$ for $k = n-1, n-2, \ldots, 1$, while all other a'_{ij} are equal to zero, then the small round-off error committed when evaluating x_n by formula (5.48) increases by a factor of two when computing x_{n-1}, then by another factor of two when computing x_{n-2}, and eventually by a factor of 2^{n-1} when computing x_n. Already for $n = 11$ the error grows more than a thousand times.

Another potential danger that one needs to keep in mind is that the quantities f'_k can rapidly increase when k increases. If this happens, then even small relative errors committed when computing f'_k in expression (5.47) can lead to a large absolute error in the value of x_k.

Let us now formulate a sufficient condition that would guarantee the computational stability of the Gaussian elimination algorithm.

THEOREM 5.6

Let the matrix $\boldsymbol{A}$ *of system (5.44) be a matrix with diagonal dominance of magnitude* $\delta > 0$*, see formula (5.40). Then, no division by zero will be encountered in the standard Gaussian elimination algorithm. Moreover, the following inequalities will hold:*

$$\sum_{j=1}^{n-k} |a'_{k,k+j}| < 1, \quad k = 1,2,\dots,n-1, \tag{5.49}$$

$$|f'_k| \le \frac{2}{\delta} \max_j |f_j|, \quad k = 1,2,\dots,n. \tag{5.50}$$

To prove Theorem 5.6, we will first need the following Lemma.

LEMMA 5.1

Let

$$\begin{gathered} b_{11}y_1 + b_{12}y_2 + \ldots + b_{1m}y_m = g_1, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ b_{m1}y_1 + b_{m2}y_2 + \ldots + b_{mm}y_m = g_m, \end{gathered} \tag{5.51}$$

be a system of linear algebraic equations with diagonal dominance of magnitude $\delta > 0$*:*

$$|b_{ll}| \ge \sum_{j \ne l} |b_{lj}| + \delta, \quad l = 1,2,\dots,m. \tag{5.52}$$

Then, when reducing the first equation of system (5.51) to the form

$$y_1 = b'_{12}y_2 + \ldots + b'_{1m}y_m + g'_1 \tag{5.53}$$

there will be no division by zero. Besides, the inequality

$$\sum_{j=2}^{m} |b'_{1j}| < 1 \tag{5.54}$$

will hold. Finally, if $m > 1$*, then the variable* y_1 *can be eliminated from system (5.51) with the help of expression (5.53). In doing so, the resulting* $(m-1) \times (m-1)$ *system with respect to the unknowns* $y_2, y_3, \dots, y_m$ *will also be a system with diagonal dominance of the same magnitude* δ *as in formula (5.52).*

PROOF According to formula (5.52), for $l = 1$ we have:

$$|b_{11}| \ge |b_{12}| + |b_{13}| + \ldots + |b_{1m}| + \delta.$$

Consequently, $b_{11} \neq 0$, and expression (5.53) makes sense, where

$$b'_{1j} = -\frac{b_{1j}}{b_{11}} \quad \text{for} \quad j = 2, 3, \dots, m, \quad \text{and} \quad g'_1 = \frac{g_1}{b_{11}}.$$

Moreover,

$$\sum_{j=2}^{m} |b'_{1j}| = \frac{|b_{12}| + |b_{13}| + \ldots + |b_{1m}|}{|b_{11}|},$$

hence inequality (5.54) is satisfied.

It remains to prove the last assertion of the Lemma for $m > 1$. Substituting expression (5.53) into the equation number j $(j > 1)$ of system (5.51), we obtain:

$$(b_{j2} + b_{j1}b'_{12})y_2 + (b_{j3} + b_{j1}b'_{13})y_3 + \ldots (b_{jm} + b_{j1}b'_{1m})y_m = g'_j,$$
$$j = 2, 3, \dots, m.$$

In this system of $m-1$ equations, equation number l $(l = 1, 2, \dots, m-1)$ is the equation:

$$(b_{l+1,2} + b_{l+1,1}b'_{12})y_2 + (b_{l+1,3} + b_{l+1,1}b'_{13})y_3 + \ldots (b_{l+1,m} + b_{l+1,1}b'_{1m})y_m = g'_{l+1},$$
$$l = 1, 2, \dots, m-1. \tag{5.55}$$

Consequently, the entries in row number l of the matrix of system (5.55) are:

$$(b_{l+1,2} + b_{l+1,1}b'_{12}),\ (b_{l+1,3} + b_{l+1,1}b'_{13}),\ \ldots,\ (b_{l+1,m} + b_{l+1,1}b'_{1m}),$$

and the corresponding diagonal entry is: $(b_{l+1,l+1} + b_{l+1,1}b'_{1,l+1})$.

Let us show that there is a diagonal dominance of magnitude δ, i.e., that the following estimate holds:

$$|b_{l+1,l+1} + b_{l+1,1}b'_{1,l+1}| \geq \sum_{\substack{j=2,\\ j\neq l+1}}^{m} |b_{l+1,j} + b_{l+1,1}b'_{1j}| + \delta. \tag{5.56}$$

We will prove an even stronger inequality:

$$|b_{l+1,l+1}| - |b_{l+1,1}b'_{1,l+1}| \geq \sum_{\substack{j=2,\\ j\neq l+1}}^{m} \left[|b_{l+1,j}| + |b_{l+1,1}b'_{1j}|\right] + \delta,$$

which, in turn, is equivalent to the inequality:

$$|b_{l+1,l+1}| \geq \sum_{\substack{j=2,\\ j\neq l+1}}^{m} |b_{l+1,j}| + |b_{l+1,1}| \sum_{j=2}^{m} |b'_{1j}| + \delta. \tag{5.57}$$

Let us replace the quantity $\sum\limits_{j=2}^{m} |b'_{1j}|$ in formula (5.57) by the number 1:

$$|b_{l+1,l+1}| \geq \sum_{\substack{j=2,\\ j\neq l+1}}^{m} |b_{l+1,j}| + |b_{l+1,1}| + \delta = \sum_{\substack{j=1,\\ j\neq l+1}}^{m} |b_{l+1,j}| + \delta. \tag{5.58}$$

According to estimate (5.54), if inequality (5.58) holds, then inequality (5.57) will automatically hold. However, inequality (5.58) is true because of estimate (5.52). Thus, we have proven inequality (5.56). ▯

PROOF OF THEOREM 5.6 We will first establish the validity of formula (5.47) and prove inequality (5.49). To do so, we will use induction with respect to k. If $k = 1$ and $n > 1$, formula (5.47) and inequality (5.49) are equivalent to formula (5.53) and inequality (5.54), respectively, proven in Lemma 5.1. In addition, Lemma 5.1 implies that the $(n-1)\times(n-1)$ system with respect to $x_2, x_3, \dots, x_n$ obtained from (5.44) by eliminating the variable x_1 using (5.45) will also be a system with diagonal dominance of magnitude δ.

Assume now that formula (5.47) and inequality (5.49) have already been proven for all $k = 1, 2, \dots, l$, where $l < n$. Also assume that the $(n-l)\times(n-l)$ system with respect to $x_{l+1}, x_{l+2}, \dots, x_n$ obtained from (5.44) by consecutively eliminating the variables $x_1, x_2, \dots, x_l$ using (5.47) is a system with diagonal dominance of magnitude δ. Then this system can be considered in the capacity of system (5.51), in which case Lemma 5.1 immediately implies that the assumption of the induction is true for $k = l+1$ as well. This completes the proof by induction.

Next, let us justify inequality (5.50). According to Theorem 5.5, the following estimate holds for the solution of system (5.44):

$$\max_j |x_j| \le \frac{1}{\delta}\max_i |f_i|.$$

Employing this inequality along with formula (5.47) and inequality (5.49) that we have just proven, we obtain for any $k = 1, 2, \dots, n$:

$$\begin{aligned} |f'_k| &= \left| x_k - \sum_{j=k+1}^{n} a'_{kj} x_j \right| \le |x_k| + \sum_{j=k+1}^{n} |a'_{kj}||x_j| \\ &\le \max_{1\le j\le n} |x_j| \left(1 + \sum_{j=k+1}^{n} |a'_{kj}| \right) \le 2 \max_{1\le j\le n} |x_j| \le \frac{2}{\delta}\max_i |f_i|. \end{aligned}$$

This estimate obviously coincides with (5.50). ▯

Let us emphasize that the hypothesis of Theorem 5.6 (diagonal dominance) provides a sufficient but not a necessary condition for the applicability of the standard Gaussian elimination procedure. There are other linear systems (5.44) that lend themselves to the solution by this method. If, for a given system (5.44), the Gaussian elimination is successfully implemented on a computer (no divisions by zero and no instabilities), then the accuracy of the resulting exact solution will only be limited by the machine precision, i.e., by the round-off errors.

Having computed the approximate solution of system (5.44), one can substitute it into the left-hand side and thus obtain the residual $\Delta \boldsymbol{f}$ of the right-hand side. Then, estimate

$$\frac{\|\Delta \boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leq \mu(\boldsymbol{A}) \frac{\|\Delta \boldsymbol{f}\|}{\|\boldsymbol{f}\|},$$

can be used to judge the error of the solution, provided that the condition number $\mu(\boldsymbol{A})$ or its upper bound is known. This is one of the ideas behind the so-called a posteriori analysis.

Computational complexity of the standard Gaussian elimination algorithm is cubic with respect to the dimension n of the system. More precisely, it requires roughly $\frac{2}{3}n^3 + 2n^2 = \mathcal{O}(n^3)$ arithmetic operations to obtain the solution. In doing so, the cubic component of the cost comes from the elimination per se, whereas the quadratic component is the cost of solving back for $x_n, x_{n-1}, \dots, x_1$.

5.4.2 Tri-Diagonal Elimination

The Gaussian elimination algorithm of Section 5.4.1 is particularly efficient for a system (5.44) of the following special kind:

$$\begin{array}{lllll}
b_1x_1 + c_1x_2 & & & & = f_1, \\
a_2x_1 + b_2x_2 + & c_2x_3 & & & = f_2, \\
\phantom{a_2x_1 +{}} a_3x_2 + & b_3x_3 & + \quad c_3x_4 & & = f_3, \\
\multicolumn{5}{c}{\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots} \\
 & & a_{n-1}x_{n-2} + b_{n-1}x_{n-1} & + \; c_{n-1}x_n & = f_{n-1}, \\
 & & & a_nx_{n-1} + b_nx_n & = f_n.
\end{array} \tag{5.59}$$

The matrix of system (5.59) is tri-diagonal, i.e., all its entries are equal to zero except those on the main diagonal, on the super-diagonal, and on the sub-diagonal. In other words, if $\boldsymbol{A} = \{\mathfrak{a}_{ij}\}$, then $\mathfrak{a}_{ij} = 0$ for $j > i+1$ and $j < i-1$. Adopting the notation of formula (5.59), we say that $\mathfrak{a}_{ii} = b_i$, $i = 1,2,\dots,n$; $\mathfrak{a}_{i,i-1} = a_i$, $i = 2,3,\dots,n$; and $\mathfrak{a}_{i,i+1} = c_i$, $i = 1,2,\dots,n-1$.

The conditions of diagonal dominance (5.40) for system (5.59) read:

$$\begin{aligned}
&|b_1| \geq |c_1| + \delta, \\
&|b_k| \geq |a_k| + |c_k| + \delta, \quad k = 2,3,\dots,n-1, \\
&|b_n| \geq |a_n| + \delta.
\end{aligned} \tag{5.60}$$

Equalities (5.45)–(5.48) transform into:

$$\begin{aligned}
x_k &= A_kx_{k+1} + F_k, \quad k = 1,2,\dots,n-1, \\
x_n &= F_n,
\end{aligned} \tag{5.61}$$

where A_k and F_k are some coefficients. Define $A_n = 0$ and rewrite formulae (5.61) as a unified expression:

$$x_k = A_kx_{k+1} + F_k, \quad k = 1,2,\dots,n. \tag{5.62}$$

From the first equation of system (5.59) it is clear that for $k = 1$ the coefficients in formula (5.62) are:

$$A_1 = -\frac{c_1}{b_1}, \quad F_1 = \frac{f_1}{b_1}. \tag{5.63}$$

Suppose that all the coefficients A_k and F_k have already been computed up to some fixed k, $1 \leq k \leq n-1$. Substituting the expression $x_k = A_k x_{k+1} + F_k$ into the equation number $k+1$ of system (5.59) we obtain:

$$x_{k+1} = -\frac{c_{k+1}}{b_{k+1} + a_{k+1}A_k} x_{k+2} + \frac{f_{k+1} - a_{k+1}F_k}{b_{k+1} + a_{k+1}A_k}.$$

Therefore, the coefficients A_k and F_k satisfy the following recurrence relations:

$$\begin{gathered} A_{k+1} = -\frac{c_{k+1}}{b_{k+1} + a_{k+1}A_k}, \quad F_{k+1} = \frac{f_{k+1} - a_{k+1}F_k}{b_{k+1} + a_{k+1}A_k}, \\ k = 1, 2, \dots, n-1. \end{gathered} \tag{5.64}$$

As such, the algorithm of solving system (5.59) gets split into two stages. At the first stage, we evaluate the coefficients A_k and F_k for $k = 1, 2, \dots, n$ using formulae (5.63) and (5.64). At the second stage, we solve back for the actual unknowns $x_n, x_{n-1}, \dots, x_1$ using formulae (5.62) for $k = n, n-1, \dots, 1$.

In the literature, one can find several alternative names for the tri-diagonal Gaussian elimination procedure that we have described. Sometimes, the term *marching* is used. The first stage of the algorithm is also referred to as the forward stage or forward marching, when the marching coefficients A_k and B_k are computed. Accordingly, the second stage of the algorithm, when relations (5.62) are applied consecutively in the reverse order is called backward marching.

We will now estimate the computational complexity of the tri-diagonal elimination. At the forward stage, the elimination according to formulae (5.63) and (5.64) requires $\mathscr{O}(n)$ arithmetic operations. At the backward stage, formula (5.62) is applied n times, which also requires $\mathscr{O}(n)$ operations. Altogether, the complexity of the tri-diagonal elimination is $\mathscr{O}(n)$ arithmetic operations. It is clear that no algorithm can be built that would be asymptotically cheaper than $\mathscr{O}(n)$, because the number of unknowns in the system is also $\mathscr{O}(n)$.

Let us additionally note that the tri-diagonal elimination is apparently the only example available in the literature of a direct method with linear complexity, i.e., of a method that produces the exact solution of a linear system at a cost of $\mathscr{O}(n)$ operations. In other words, the computational cost is directly proportional to the dimension of the system. We will later see examples of direct methods that produce the exact solution at a cost of $\mathscr{O}(n \ln n)$ operations, and examples of iterative methods that cost $\mathscr{O}(n)$ operations but only produce an approximate solution. However, no other method of computing the exact solution with a genuinely linear complexity is known.

The algorithm can also be generalized to the case of the banded matrices. Matrices of this type may contain non-zero entries on several neighboring diagonals, including the main diagonal. Normally we would assume that the number m of the non-zero diagonals, i.e., the bandwidth, satisfies $3 \leq m \ll n$. The complexity of the Gaussian elimination algorithm when applied to a banded system (5.44) is $\mathscr{O}(m^2 n)$ operations. If m is fixed and n is arbitrary, then the complexity, again, scales as $\mathscr{O}(n)$.

High-order systems of type (5.59) drew the attention of researchers in the fifties. They appeared when solving the heat equation numerically with the help of the so-called implicit finite-difference schemes. These schemes, their construction and their importance, will be discussed later in Part III of the book, see, in particular, Section 10.6.

The foregoing tri-diagonal marching algorithm was apparently introduced for the first time by I. M. Gelfand and O. V. Lokutsievskii around the late forties or early fifties. They conducted a complete analysis of the algorithm, showed that it was computationally stable, and also built its continuous "closure," see Appendix II to the book [GR64] written by Gelfand and Lokutsievskii.

Alternatively, the tri-diagonal elimination algorithm is attributed to L. H. Thomas [Tho49], and is referred to as the Thomas algorithm.

The aforementioned work by Gelfand and Lokutsievskii was one of the first papers in the literature where the question of stability of a computational algorithm was accurately formulated and solved for a particular class of problems. This question has since become one of the key issues for the entire large field of knowledge called scientific computing. Having stability is crucial, as otherwise computer codes that implement the algorithms will not execute properly. In Part III of the book, we study computational stability for the finite-difference schemes.

Theorem 5.6, which provides sufficient conditions for applicability of the Gaussian elimination, is a generalization to the case of full matrices of the result by Gelfand and Lokutsievskii on stability of the tri-diagonal marching.

Note also that the conditions of strict diagonal dominance (5.60) that are sufficient for stability of the tri-diagonal elimination can actually be relaxed. In fact, one can only require that the coefficients of system (5.59) satisfy the inequalities:

$$\begin{aligned} &|b_1| \geq |c_1|, \\ &|b_k| \geq |a_k| + |c_k|, \quad k = 2, 3, \dots, n-1, \\ &|b_n| \geq |a_n|, \end{aligned} \tag{5.65}$$

so that at least one out of the total of n inequalities (5.65) actually be strict, i.e., "$>$" rather than "$\geq$." We refer the reader to [SN89a, Chapter II] for detail.

Overall, the idea of transporting, or marching, the condition $b_1 x_1 + c_1 x_2 = f_1$ specified by the first equation of the tri-diagonal system (5.59) is quite general. In the previous algorithm, this idea is put to use when obtaining the marching coefficients (5.63), (5.64) and relations (5.62). It is also exploited in many other elimination algorithms, see [SN89a, Chapter II]. We briefly describe one of those algorithms, known as the cyclic tri-diagonal elimination, in Section 5.4.3.

5.4.3 Cyclic Tri-Diagonal Elimination

In many applications, one needs to solve a system which is "almost" tri-diagonal, but is not quite equivalent to system (5.59):

$$\begin{array}{l} b_1x_1 + c_1x_2 \qquad\qquad\qquad\qquad + a_1x_n = f_1, \\ a_2x_1 + b_2x_2 + c_2x_3 \qquad\qquad\qquad\quad = f_2, \\ \qquad a_3x_2 + b_3x_3 + c_3x_4 \qquad\qquad = f_3, \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ \qquad\qquad a_{n-1}x_{n-2} + b_{n-1}x_{n-1} + c_{n-1}x_n = f_{n-1}, \\ c_nx_1 \qquad\qquad\qquad + a_nx_{n-1} + b_nx_n = f_n. \end{array} \tag{5.66}$$

In Section 2.3.2, we have analyzed one particular example of this type that arises when constructing the nonlocal Schoenberg splines; see the matrix $\boldsymbol{A}$ given by formula (2.66) on page 52. Other typical examples include the so-called central difference schemes (see Section 9.2.1) for the solution of second order ordinary differential equations with periodic boundary conditions, as well as many schemes built for solving partial differential equations in the cylindrical or spherical coordinates. Periodicity of the boundary conditions gave rise to the name *cyclic* attached to the version of the tri-diagonal elimination that we are about to describe.

The coefficients a_1 and c_n in the first and last equations of system (5.66), respectively, are, generally speaking, non-zero. Their presence does not allow one to apply the tri-diagonal elimination algorithm of Section 5.4.2 to system (5.66) directly. Let us therefore consider two auxiliary linear systems of dimension $(n-1)\times(n-1)$:

$$\begin{array}{l} b_2u_2 + c_2u_3 \qquad\qquad\qquad = f_2, \\ a_3u_2 + b_3u_3 + c_3u_4 \qquad\quad = f_3, \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ \qquad a_{n-1}u_{n-2} + b_{n-1}u_{n-1} + c_{n-1}u_n = f_{n-1}, \\ \qquad\qquad\qquad a_nu_{n-1} + b_nu_n = f_n. \end{array} \tag{5.67}$$

and

$$\begin{array}{l} b_2v_2 + c_2v_3 \qquad\qquad\qquad = -a_2, \\ a_3v_2 + b_3v_3 + c_3v_4 \qquad\quad = 0, \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ \qquad a_{n-1}v_{n-2} + b_{n-1}v_{n-1} + c_{n-1}v_n = 0, \\ \qquad\qquad\qquad a_nv_{n-1} + b_nv_n = -c_n. \end{array} \tag{5.68}$$

Having obtained the solutions $\{u_2,u_3,\dots,u_n\}$ and $\{v_2,v_3,\dots,v_n\}$ to systems (5.67) and (5.68), respectively, we can represent the solution $\{x_1,x_2,\dots,x_n\}$ to system (5.66) in the form:

$$x_i = u_i + x_1v_i, \quad i = 1,2,\dots,n, \tag{5.69}$$

where for convenience we additionally define $u_1 = 0$ and $v_1 = 1$. Indeed, multiplying each equation of system (5.68) by x_1, adding with the corresponding equation of system (5.67), and using representation (5.69), we immediately see that the equations number 2 through n of system (5.66) are satisfied. It only remains to satisfy equation number 1 of system (5.66). To do so, we use formula (5.69) for $i = 2$ and $i = n$, and substitute $x_2 = u_2 + x_1 v_2$ and $x_n = u_n + x_1 v_n$ into the first equation of system (5.66), which yields one scalar equation for x_1:

$$b_1 x_1 + c_1(u_2 + x_1 v_2) + a_1(u_n + x_1 v_n) = f_1.$$

As such, we find:

$$x_1 = \frac{f_1 - a_1 u_n - c_1 u_2}{b_1 + a_1 v_n + c_1 v_2}. \tag{5.70}$$

Altogether, the solution algorithm for system (5.66) reduces to first solving the two auxiliary systems (5.67) and (5.68), then finding x_1 with the help of formula (5.70), and finally obtaining $x_2, x_3, \dots, x_n$ according to formula (5.69). As both systems (5.67) and (5.68) are genuinely tri-diagonal, they can be solved by the original tri-diagonal elimination described in Section 5.4.2. In doing so, the overall computational complexity of solving system (5.66) obviously remains linear with respect to the dimension of the system n, i.e., $\mathscr{O}(n)$ arithmetic operations.

5.4.4 Matrix Interpretation of the Gaussian Elimination. *LU* Factorization

Consider system (5.44) written as $\boldsymbol{Ax} = \boldsymbol{f}$ or alternatively, $\boldsymbol{A}^{(0)}\boldsymbol{x} = \boldsymbol{f}^{(0)}$, where

$$\boldsymbol{A} \equiv \boldsymbol{A}^{(0)} = \begin{bmatrix} a_{11}^{(0)} & \dots & a_{1n}^{(0)} \\ \vdots & \ddots & \vdots \\ a_{n1}^{(0)} & \dots & a_{nn}^{(0)} \end{bmatrix} \quad \text{and} \quad \boldsymbol{f} \equiv \boldsymbol{f}^{(0)} = \begin{bmatrix} f_1^{(0)} \\ \vdots \\ f_n^{(0)} \end{bmatrix},$$

and the superscript "(0)" emphasizes that this is the beginning, i.e., the zeroth stage of the Gaussian elimination procedure. The notations in this section are somewhat different from those of Section 5.4.1, but we will later show the correspondence.

At the first stage of the algorithm we assume that $a_{11}^{(0)} \neq 0$ and introduce the transformation matrix:

$$\boldsymbol{T}_1 = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ -t_{21} & 1 & 0 & \dots & 0 \\ -t_{31} & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -t_{n1} & 0 & 0 & \dots & 1 \end{bmatrix}, \quad \text{where} \quad t_{i1} = \frac{a_{i1}^{(0)}}{a_{11}^{(0)}}, \quad i = 2, 3, \dots, n.$$

Applying this matrix is equivalent to eliminating the variable x_1 from equations number 2 through n of system (5.44):

$$\boldsymbol{T}_1\boldsymbol{A}^{(0)}\boldsymbol{x} \equiv \boldsymbol{A}^{(1)}\boldsymbol{x} = \begin{bmatrix} a_{11}^{(0)} & a_{12}^{(0)} & \dots & a_{1n}^{(0)} \\ 0 & a_{22}^{(1)} & \dots & a_{2n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(1)} & \dots & a_{nn}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} f_1^{(0)} \\ f_2^{(1)} \\ \vdots \\ f_n^{(1)} \end{bmatrix} = \boldsymbol{f}^{(1)} \equiv \boldsymbol{T}_1\boldsymbol{f}^{(0)}.$$

At the second stage, we define:

$$\boldsymbol{T}_2 = \begin{bmatrix} 1 & 0 & 0 \dots 0 \\ 0 & 1 & 0 \dots 0 \\ 0 & -t_{32} & 1 \dots 0 \\ \vdots & & \ddots \vdots \\ 0 & -t_{n2} & 0 \dots 1 \end{bmatrix}, \quad \text{where} \quad t_{i2} = \frac{a_{i2}^{(1)}}{a_{22}^{(1)}}, \quad i = 3, 4, \dots, n,$$

and eliminate x_2 from equations 3 through n, thus obtaining the system $\boldsymbol{A}^{(2)}\boldsymbol{x} = \boldsymbol{f}^{(2)}$, where

$$\boldsymbol{A}^{(2)} = \boldsymbol{T}_2\boldsymbol{A}^{(1)} = \begin{bmatrix} a_{11}^{(0)} & a_{12}^{(0)} & a_{13}^{(0)} & \dots & a_{1n}^{(0)} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \dots & a_{2n}^{(1)} \\ 0 & 0 & a_{33}^{(2)} & \dots & a_{3n}^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & a_{n3}^{(2)} & \dots & a_{nn}^{(2)} \end{bmatrix} \quad \text{and} \quad \boldsymbol{f}^{(2)} = \boldsymbol{T}_2 f^{(1)} = \begin{bmatrix} f_1^{(0)} \\ f_2^{(1)} \\ f_3^{(2)} \\ \vdots \\ f_n^{(2)} \end{bmatrix}.$$

In general, at stage number k we have the transformation matrix:

$$\boldsymbol{T}_k = \begin{bmatrix} 1 & \dots & 0 & 0 \dots 0 \\ \vdots & \ddots & \vdots & \vdots \quad \vdots \\ 0 & \dots & 1 & 0 \dots 0 \\ 0 & \dots & -t_{k+1,k} & 1 \dots 0 \\ \vdots & & \vdots & \vdots \ddots \vdots \\ 0 & \dots & -t_{n,k} & 0 \dots 1 \end{bmatrix}, \quad \text{where} \quad t_{i,k} = \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}}, \quad i = k+1, \dots, n, \quad (5.71)$$

and the system $\boldsymbol{A}^{(k)}\boldsymbol{x} = \boldsymbol{f}^{(k)}$, where:

$$\boldsymbol{A}^{(k)} = \boldsymbol{T}_k\boldsymbol{A}^{(k-1)} = \begin{bmatrix} a_{11}^{(0)} & \dots & a_{1,k}^{(0)} & a_{1,k+1}^{(0)} & \dots & a_{1,n}^{(0)} \\ \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & \dots & a_{k,k}^{(k-1)} & a_{k,k+1}^{(k-1)} & \dots & a_{k,n}^{(k-1)} \\ 0 & \dots & 0 & a_{k+1,k+1}^{(k)} & \dots & a_{k+1,n}^{(k)} \\ \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & a_{n,k+1}^{(k)} & \dots & a_{n,n}^{(k)} \end{bmatrix}, \qquad \boldsymbol{f}^{(k)} = \boldsymbol{T}_k f^{(k-1)} = \begin{bmatrix} f_1^{(0)} \\ \vdots \\ f_k^{(k-1)} \\ f_{k+1}^{(k)} \\ \vdots \\ f_n^{(k)} \end{bmatrix}.$$

Performing all $n-1$ stages of the elimination, we arrive at the system:

$$\boldsymbol{A}^{(n-1)}\boldsymbol{x} = \boldsymbol{f}^{(n-1)},$$

where

$$\boldsymbol{A}^{(n-1)} = \boldsymbol{T}_{n-1}\boldsymbol{T}_{n-2}\dots\boldsymbol{T}_1\boldsymbol{A} \quad \text{and} \quad \boldsymbol{f}^{(n-1)} = \boldsymbol{T}_{n-1}\boldsymbol{T}_{n-2}\dots\boldsymbol{T}_1\boldsymbol{f}.$$

The resulting matrix $\boldsymbol{A}^{(n-1)}$ is upper triangular and we re-denote it by $\boldsymbol{U}$. All entries of this matrix below its main diagonal are equal to zero:

$$\boldsymbol{A}^{(n-1)} \equiv \boldsymbol{U} = \begin{bmatrix} a_{11}^{(0)} & \dots & a_{1,k}^{(0)} & a_{1,k+1}^{(0)} & \dots & a_{1,n}^{(0)} \\ \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & \dots & a_{k,k}^{(k-1)} & a_{k,k+1}^{(k-1)} & \dots & a_{k,n}^{(k-1)} \\ 0 & \dots & 0 & a_{k+1,k+1}^{(k)} & \dots & a_{k+1,n}^{(k)} \\ \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & 0 & \dots & a_{n,n}^{(n-1)} \end{bmatrix}. \tag{5.72}$$

The entries on the main diagonal, $a_{k,k}^{(k-1)}$, $k = 1,2,\dots,n$, are called pivots, and none of them may turn into zero as otherwise the standard Gaussian elimination procedure will fail. Solving the system $\boldsymbol{U}\boldsymbol{x} = \boldsymbol{f}^{(n-1)} \equiv \boldsymbol{g}$ is a fairly straightforward task, as we have seen in Section 5.4.1. It amounts to computing the values of all the unknowns one after another in the reverse order, i.e., starting from $x_n = f_n^{(n-1)}/a_{n,n}^{(n-1)}$, then $x_{n-1} = (f_{n-1}^{(n-2)} - a_{n-1,n}^{(n-2)}x_n)/a_{n-1,n-1}^{(n-2)}$, etc., and all the way up to x_1.

We will now show that the representation $\boldsymbol{U} = \boldsymbol{T}_{n-1}\boldsymbol{T}_{n-2}\dots\boldsymbol{T}_1\boldsymbol{A}$ implies that $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}$, where $\boldsymbol{L}$ is a lower triangular matrix, i.e., a matrix that only has zero entries above its main diagonal. In other words, we will need to demonstrate that $\boldsymbol{L} \overset{\text{def}}{=} (\boldsymbol{T}_{n-1}\boldsymbol{T}_{n-2}\dots\boldsymbol{T}_1)^{-1} = \boldsymbol{T}_1^{-1}\boldsymbol{T}_2^{-1}\dots\boldsymbol{T}_{n-1}^{-1}$ is a lower triangular matrix. Then, the formula

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U} \tag{5.73}$$

will be referred to as an $\boldsymbol{L}\boldsymbol{U}$ factorization of the matrix $\boldsymbol{A}$.

One can easily verify by direct multiplication that the inverse matrix for a given $\boldsymbol{T}_k$, $k = 1,2,\dots,n-1$, see formula (5.71), is:

$$\boldsymbol{T}_k^{-1} = \begin{bmatrix} 1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & \dots & 1 & 0 & \dots & 0 \\ 0 & \dots & t_{k+1,k} & 1 & \dots & 0 \\ \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & t_{n,k} & 0 & \dots & 1 \end{bmatrix}. \tag{5.74}$$

It is also clear that if the meaning of the operation rendered by $\boldsymbol{T}_k$ is "take equation number k, multiply it consecutively by $t_{k+1,k},\dots,t_{n,k}$, and *subtract from* equations number $k+1$ through n, respectively," then the meaning of the inverse operation has to be "take equation number k, multiply it consecutively by the factors $t_{k+1,k},\dots,t_{n,k}$, and *add to* equations number $k+1$ through n, respectively," which is precisely what the foregoing matrix $\boldsymbol{T}_k^{-1}$ does.

We see that all the matrices $\boldsymbol{T}_k^{-1}$, $k = 1,2,\dots,n-1$, given by formula (5.74) are lower triangular matrices with the diagonal entries equal to one. Their product can be calculated directly, which yields:

$$\boldsymbol{L} = \boldsymbol{T}_1^{-1}\boldsymbol{T}_2^{-1}\dots\boldsymbol{T}_{n-1}^{-1} = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 & \dots & 0 \\ t_{2,1} & 1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ t_{k,1} & t_{k,2} & \dots & 1 & 0 & \dots & 0 \\ t_{k+1,1} & t_{k+1,2} & \dots & t_{k+1,k} & 1 & \dots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \ddots & \vdots \\ t_{n,1} & t_{n,2} & \dots & t_{n,k} & t_{n,k+1} & \dots & 1 \end{bmatrix}. \tag{5.75}$$

Consequently, the matrix $\boldsymbol{L}$ is indeed a lower triangular matrix (with all its diagonal entries equal to one), and the factorization formula (5.73) holds.

The $\boldsymbol{LU}$ factorization of the matrix $\boldsymbol{A}$ allows us to analyze the computational complexity of the Gaussian elimination algorithm as it applies to solving multiple linear systems that have the same matrix $\boldsymbol{A}$ but different right-hand sides. The cost of obtaining the factorization itself, i.e., that of computing the matrix $\boldsymbol{U}$, is cubic: $\mathscr{O}(n^3)$ arithmetic operations. This factorization obviously stays the same when the right-hand side changes. For a given right-hand side $\boldsymbol{f}$, we need to solve the system $\boldsymbol{LUx} = \boldsymbol{f}$. This amounts to first solving the system $\boldsymbol{Lg} = \boldsymbol{f}$ with a lower triangular matrix $\boldsymbol{L}$ of (5.75) and obtaining $\boldsymbol{g} = \boldsymbol{L}^{-1}\boldsymbol{f} = \boldsymbol{T}_{n-1}\boldsymbol{T}_{n-2}\dots\boldsymbol{T}_1\boldsymbol{f}$, and then solving the system $\boldsymbol{Ux} = \boldsymbol{g}$ with an upper triangular matrix $\boldsymbol{U}$ of (5.72) and obtaining $\boldsymbol{x}$. The cost of either solution is $\mathscr{O}(n^2)$ operations. Consequently, once the $\boldsymbol{LU}$ factorization has been built, each additional right-hand side can be accommodated at a quadratic cost.

In particular, consider the problem of finding the inverse matrix $\boldsymbol{A}^{-1}$ using Gaussian elimination. By definition, $\boldsymbol{AA}^{-1} = \boldsymbol{I}$. In other words, each column of the matrix $\boldsymbol{A}^{-1}$ is the solution to the system $\boldsymbol{Ax} = \boldsymbol{f}$ with the right-hand side $\boldsymbol{f}$ equal to the corresponding column of the identity matrix $\boldsymbol{I}$. Altogether, there are n columns, each adding an $\mathscr{O}(n^2)$ solution cost to the $\mathscr{O}(n^3)$ initial cost of the $\boldsymbol{LU}$ factorization that is performed only once ahead of time. We therefore conclude that the overall cost of computing $\boldsymbol{A}^{-1}$ using Gaussian elimination is also cubic: $\mathscr{O}(n^3)$ operations.

Finally, let us note that for a given matrix $\boldsymbol{A}$, its $\boldsymbol{LU}$ factorization is, generally speaking, not unique. The procedure that we have described yields a particular form of the $\boldsymbol{LU}$ factorization (5.73) defined by an additional constraint that all diagonal entries of the matrix $\boldsymbol{L}$ of (5.75) be equal to one. Instead, we could have required, for example, that the diagonal entries of $\boldsymbol{U}$ be equal to one. Then, the matrices $\boldsymbol{T}_k$ of (5.71) get replaced by:

$$\tilde{\boldsymbol{T}}_k = \begin{bmatrix} 1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & \dots & t_{k,k} & 0 & \dots & 0 \\ 0 & \dots & -t_{k+1,k} & 1 & \dots & 0 \\ \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & -t_{n,k} & 0 & \dots & 1 \end{bmatrix}, \quad \text{where} \quad t_{k,k} = \frac{1}{a_{kk}^{(k-1)}}, \tag{5.76}$$

and instead of (5.72) we have:

$$\tilde{\boldsymbol{U}} = \begin{bmatrix} 1 & \dots & -a'_{1,k} & -a'_{1,k+1} & \cdots & -a'_{1,n} \\ \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & \dots & 1 & -a'_{k,k+1} & \cdots & -a'_{k,n} \\ 0 & \dots & 0 & 1 & \dots & -a'_{k+1,n} \\ \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & 0 & \dots & 1 \end{bmatrix}, \tag{5.77}$$

where the off-diagonal entries of the matrix $\tilde{\boldsymbol{U}}$ are the same as introduced in the beginning of Section 5.4.1. The matrices $\tilde{\boldsymbol{T}}_k^{-1}$ and $\tilde{\boldsymbol{L}}$ are obtained accordingly; this is the subject of Exercise 1 after this section.

5.4.5 Cholesky Factorization

If $\boldsymbol{A}$ is a symmetric positive definite (SPD) matrix (with real entries), $\boldsymbol{A} = \boldsymbol{A}^T > 0$, then it admits a special form of $\boldsymbol{LU}$ factorization, namely:

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^T, \tag{5.78}$$

where $\boldsymbol{L}$ is a lower triangular matrix with positive entries on its main diagonal. Factorization (5.78) is known as the Cholesky factorization. The proof of formula (5.78), which uses induction with respect to the dimension of the matrix, can be found, e.g., in [GL81, Chapter 2]. Moreover, one can derive explicit formulae for the entries of the matrix $\boldsymbol{L}$:

$$l_{jj} = \left(a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2 \right)^{\frac{1}{2}}, \quad j = 1, 2, \dots, n,$$
$$l_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk} \right) \cdot l_{jj}^{-1}, \quad i = j+1, j+2, \dots, n.$$

Computational complexity of obtaining the Cholesky factorization (5.78) is also cubic with respect to the dimension of the matrix: $\mathscr{O}(n^3)$ arithmetic operations. However, the actual cost is roughly one half compared to that of the standard Gaussian elimination: $n^3/3$ vs. $2n^3/3$ arithmetic operations. This reduction of the cost is the key benefit of using the Cholesky factorization, as opposed to the standard Gaussian elimination, for large symmetric matrices. The reduction of the cost is only enabled by the special SPD structure of the matrix $\boldsymbol{A}$; there is no Cholesky factorization for general matrices, and one has to resort back to the standard $\boldsymbol{LU}$. This is one of many examples in numerical analysis when a more efficient computational procedure can be obtained for a more narrow class of objects that define the problem.

5.4.6 Gaussian Elimination with Pivoting

The standard Gaussian procedure of Section 5.4.1 fails if a pivotal entry in the matrix appears equal to zero. Consider, for example, the following linear system:

$$\begin{aligned} x_1 + 2x_2 + 3x_3 &= 1, \\ 2x_1 + 4x_2 + 5x_3 &= 2, \\ 7x_1 + 8x_2 + 9x_3 &= 3. \end{aligned}$$

After the first stage of elimination we obtain:

$$\begin{aligned} x_1 + 2x_2 + 3x_3 &= 1, \\ 0 \cdot x_1 + 0 \cdot x_2 - x_3 &= 0, \\ 0 \cdot x_1 - 6x_2 - 12x_3 &= -4. \end{aligned}$$

The pivotal entry in the second equation of this system, i.e., the coefficient in front of x_2, is equal to zero. Therefore, this equation cannot be used for eliminating x_2 from the third equation. As such, the algorithm cannot proceed any further. The problem, however, can be easily fixed by changing the order of the equations:

$$\begin{aligned} x_1 + 2x_2 + 3x_3 &= 1, \\ -6x_2 - 12x_3 &= -4, \\ x_3 &= 0, \end{aligned}$$

and we see that the system is already reduced to the upper triangular form.

Different strategies of *pivoting* are designed to help overcome or alleviate the difficulties that the standard Gaussian elimination may encounter — division by zero and the loss of accuracy due to an instability. Pivoting exploits pretty much the same idea as outlined in the previous simple example — changing the order of equations and/or the order of unknowns in the equations. In the case $\det \boldsymbol{A} \neq 0$, the elimination procedure with pivoting guarantees that there will be no division by zero, and also improves the computational stability compared to the standard Gaussian elimination. We will describe partial pivoting and complete pivoting.

When performing partial pivoting for a non-singular system, we start with finding the entry with the maximum absolute value[4] in the first column of the matrix $\boldsymbol{A}$. Assume that this is the entry a_{i1}. Clearly, $a_{i1} \neq 0$, because otherwise $\det \boldsymbol{A} = 0$. Then, we change the order of the equations in the system — the first equation becomes equation number i, and equation number i becomes the first equation. The system after this operation obviously remains equivalent to the original one. Finally, one step of the standard Gaussian elimination is performed, see Section 5.4.1. In doing so, the entries $-t_{21}, -t_{31}, \dots, -t_{n1}$ of the matrix $\boldsymbol{T}_1$, see formula (5.71) will all have absolute values less than one. This improves stability, because multiplication of the subtrahend by a quantity smaller than one before subtraction helps reduce the

[4]If several entries have the same largest absolute value, we take one of those.

effect of the round-off errors. Having eliminated the variable x_1, we apply the same approach to the resulting smaller system of order $(n-1)\times(n-1)$. Namely, among all the equations we first find the equation that has the maximum absolute value of the coefficient in front of x_2. We then change the order of the equations so that this coefficient moves to the position of the pivot, and only after that eliminate x_2.

Complete pivoting is similar to partial pivoting, except that at every stage of elimination, the entry with the maximum absolute value is sought for not only in the first column, but across the entire current-stage matrix:

$$\begin{bmatrix} a_{kk}^{(k-1)} \dots a_{kn}^{(k-1)} \\ \dots\dots\dots\dots\dots \\ a_{nk}^{(k-1)} \dots a_{nn}^{(k-1)} \end{bmatrix}.$$

Once the maximum entry has been determined, it needs to be moved to the position of the pivot $a_{kk}^{(k-1)}$. This is achieved by the appropriate permutations of both the equations and the unknowns. It is clear that the system after the permutations remains equivalent to the original one.

Computational complexity of Gaussian elimination with pivoting remains cubic with respect to the dimension of the system: $\mathscr{O}(n^3)$ arithmetic operations. Of course, this estimate is only asymptotic, and the actual constants for the algorithm with partial pivoting are larger than those for the standard Gaussian elimination. This is the pay-off for its improved robustness. The algorithm with complete pivoting is even more expensive than the algorithm with partial pivoting; but it is typically more robust as well. Note that for the same reason of improved robustness, the use of Gaussian elimination with pivoting can be recommended in practice, even for those systems for which the standard algorithm does not fail.

Gaussian elimination with pivoting can be put into the matrix framework in much the same way as it has been done in Section 5.4.4 for standard Gaussian elimination. Moreover, a very similar result on the ***LU*** factorization can be obtained; this is the subject of Exercise 3. To do so, one needs to exploit the so-called permutation matrices, i.e., the matrices that change the order of rows or columns in a given matrix when applied as multipliers, see [HJ85, Section 0.9]. For example, to swap the second and the third equations in the example analyzed in the beginning of this section, one would need to multiply the system matrix from the left by the permutation matrix:

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

5.4.7 An Algorithm with a Guaranteed Error Estimate

As has already been discussed, all numbers in a computer are represented as fractions with a finite fixed number of significant digits. Consequently, many actual numbers have to be truncated, or in other words, rounded, before they can be stored in the computer memory, see Section 1.3.3. As such, when performing computations on a real machine with a fixed number of significant digits, along with the effect of inaccuracies in the input data, round-off errors are introduced at every arithmetic operation. For linear systems, the impact of these round-off errors depends on the number of significant digits, on the condition number of the matrix, as well as on the particular algorithm chosen for computing the solution. In work [GAKK93], a family of algorithms has been proposed that directly take into account the effect of rounding on a given computer. These algorithms either produce the result with a guaranteed accuracy, or otherwise determine in the course of computations that the system is conditioned so poorly that no accuracy can be guaranteed when computing its solution on the machine with a given number of significant digits.

Exercises

1. Write explicitly the matrices $\tilde{\boldsymbol{T}}_k^{-1}$ and $\tilde{\boldsymbol{L}}$ that correspond to the $\boldsymbol{LU}$ decomposition of $\boldsymbol{A}$ with $\tilde{\boldsymbol{T}}_k$ and $\tilde{\boldsymbol{U}}$ defined by formulae (5.76) and (5.77), respectively.

2. Compute the solution of the 2×2 system:
$$10^{-3}x+y=5, \quad x-y=6,$$
using standard Gaussian elimination and Gaussian elimination with pivoting. Conduct all computations with two significant digits (decimal). Compare and explain the results.

3.* Show that when performing Gaussian elimination with partial pivoting, the $\boldsymbol{LU}$ factorization is obtained in the form:
$$\boldsymbol{PA}=\boldsymbol{LU},$$
where $\boldsymbol{P}$ is the composition (i.e., product) of all permutation matrices used at every stage of the algorithm.

4. Consider a boundary value problem for the second order ordinary differential equation:
$$\frac{d^2u}{dx^2}-u=f(x), \quad x\in[0,1],$$
$$u(0)=0, \;\; u(1)=0,$$
where $u=u(x)$ is the unknown function and $f=f(x)$ is a given right-hand side. To solve this problem numerically, we first partition the interval $0\le x\le 1$ into N equal subintervals and thus build a uniform grid of $N+1$ nodes: $x_j=j\cdot h$, $h=1/N$, $j=0,1,2,\dots,N$. Then, instead of looking for the continuous function $u=u(x)$ we will be looking for its approximate table of values $\{u_0,u_1,u_2,\dots,u_N\}$ at the grid nodes $x_0,x_1,x_2,\dots,x_N$, respectively.

   At every interior node x_j, $j=1,2,\dots,N-1$, we approximately replace the second derivative by the difference quotient (for more detail, see Section 9.2):
$$\left.\frac{d^2u}{dx^2}\right|_{x=x_j}\approx\frac{u_{j+1}-2u_j+u_{j-1}}{h^2},$$
and arrive at the following finite-difference counterpart of the original problem (a central-difference scheme):
$$\frac{u_{j+1}-2u_j+u_{j-1}}{h^2}-u_j=f_j, \quad j=1,2,\dots,N-1,$$
$$u_0=0, \;\; u_N=0,$$

where $f_j \equiv f(x_j)$ is the discrete right-hand side assumed given, and u_j is the unknown discrete solution.

a) Write down the previous scheme as a system of linear algebraic equations ($N-1$ equations with $N-1$ unknowns) both in the canonical form and in an operator form with the operator specified by an $(N-1)\times(N-1)$ matrix.

b) Show that the system is tri-diagonal and satisfies the sufficient conditions of Section 5.4.2, so that the tri-diagonal elimination can be applied.

c) Assume that the exact solution u is known: $u(x) = \sin(\pi x)e^x$. Then, $f(x) = (-\pi^2\sin(\pi x)+2\pi\cos(\pi x))e^x$. Implement on the computer the tri-diagonal elimination using double precision arithmetic. Solve the system for the right-hand side $f_j = f(x_j)$ on a sequence of grids with $N = 32, 64, 128, 256$, and 512. On each grid, calculate the relative error in the maximum norm:

$$\varepsilon_\infty(N) = \frac{\max\limits_{1\le j\le N-1}|u(x_j)-u_j|}{\max\limits_{1\le j\le N-1}|u(x_j)|}.$$

By plotting $\log_2(\varepsilon_\infty(N))$ vs. $\log_2 N$, show that every time the grid is refined by a factor of 2, the error drops by approximately a factor of 4. This should indicate the second order of grid convergence of the scheme.

5.5 Minimization of Quadratic Functions and Its Relation to Linear Systems

Consider a Euclidean space $\mathbb{R}^n$ of vectors $\boldsymbol{x}$ with n real components. Let $\boldsymbol{A}$ be a linear operator, $\boldsymbol{A}: \mathbb{R}^n \longmapsto \mathbb{R}^n$, $\boldsymbol{f} \in \mathbb{R}^n$ be a fixed element of the vector space $\mathbb{R}^n$, and c be a real constant. The function $F = F(\boldsymbol{x})$ of the argument $\boldsymbol{x}$ defined as:

$$F(\boldsymbol{x}) = (\boldsymbol{A}\boldsymbol{x},\boldsymbol{x}) - 2(\boldsymbol{f},\boldsymbol{x}) + c \tag{5.79}$$

is called a *quadratic function*.

Note that since $(\boldsymbol{A}\boldsymbol{x},\boldsymbol{x}) = (\boldsymbol{x},\boldsymbol{A}^*\boldsymbol{x}) = (\boldsymbol{A}^*\boldsymbol{x},\boldsymbol{x})$, the quadratic function $F(\boldsymbol{x})$ coincides with another quadratic function:

$$F(\boldsymbol{x}) = (\boldsymbol{A}^*\boldsymbol{x},\boldsymbol{x}) - 2(\boldsymbol{f},\boldsymbol{x}) + c,$$

and consequently, with the function:

$$F(\boldsymbol{x}) = \left(\frac{\boldsymbol{A}+\boldsymbol{A}^*}{2}\boldsymbol{x},\boldsymbol{x}\right) - 2(\boldsymbol{f},\boldsymbol{x}) + c.$$

Therefore, with no loss of generality we can assume that the operator $\boldsymbol{A}$ in formula (5.79) is self-adjoint. Otherwise, we can simply replace a given operator $\boldsymbol{A}$ with the corresponding self-adjoint operator $(\boldsymbol{A}+\boldsymbol{A}^*)/2$.

From now on, we will suppose that $\boldsymbol{A} = \boldsymbol{A}^*$ and $\boldsymbol{A} > 0$ in formula (5.79), i.e., $(\boldsymbol{Ax}, \boldsymbol{x}) > 0$ for any $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{x} \neq \boldsymbol{0}$. Given the function $F(\boldsymbol{x})$ of (5.79), let us formulate the problem of its minimization, i.e., the problem of finding a particular element $\boldsymbol{z} \in \mathbb{R}^n$ that delivers the minimum value to $F(\boldsymbol{x})$:

$$F(\boldsymbol{z}) = \min_{\boldsymbol{x} \in \mathbb{R}^n} F(\boldsymbol{x}). \tag{5.80}$$

It turns out that the minimization problem (5.80) and the problem of solving the system of linear algebraic equations:

$$\boldsymbol{Ax} = \boldsymbol{f}, \quad \text{where } \boldsymbol{A} = \boldsymbol{A}^* > 0, \tag{5.81}$$

are equivalent. We formulate this result in the form of a theorem.

THEOREM 5.7

Let $\boldsymbol{A} = \boldsymbol{A}^* > 0$. There is one and only one element $\boldsymbol{z} \in \mathbb{R}^n$ that delivers a minimum to the quadratic function $F(\boldsymbol{x})$ of (5.79). This vector $\boldsymbol{z}$ is the solution to the linear system (5.81).

PROOF As the operator $\boldsymbol{A}$ is positive definite, it is non-singular. Consequently, system (5.81) has a unique solution $\boldsymbol{z} \in \mathbb{R}^n$. Let us now show that for any $\boldsymbol{\delta} \in \mathbb{R}^n$, $\boldsymbol{\delta} \neq \boldsymbol{0}$, we have $F(\boldsymbol{z} + \boldsymbol{\delta}) > F(\boldsymbol{z})$:

$$\begin{aligned} F(\boldsymbol{z}+\boldsymbol{\delta}) &= (\boldsymbol{A}(\boldsymbol{z}+\boldsymbol{\delta}), \boldsymbol{z}+\boldsymbol{\delta}) - 2(\boldsymbol{f}, \boldsymbol{z}+\boldsymbol{\delta}) + c = \\ &= [(\boldsymbol{Az}, \boldsymbol{z}) - 2(\boldsymbol{f}, \boldsymbol{z}) + c] + 2(\boldsymbol{Az}, \boldsymbol{\delta}) - 2(\boldsymbol{f}, \boldsymbol{\delta}) + (\boldsymbol{A\delta}, \boldsymbol{\delta}) \\ &= F(\boldsymbol{z}) + 2(\boldsymbol{Az} - \boldsymbol{f}, \boldsymbol{\delta}) + (\boldsymbol{A\delta}, \boldsymbol{\delta}) \\ &= F(\boldsymbol{z}) + (\boldsymbol{A\delta}, \boldsymbol{\delta}) > F(\boldsymbol{z}). \end{aligned}$$

Consequently, the solution $\boldsymbol{z}$ of system (5.81) indeed delivers a minimum to the function $F(\boldsymbol{x})$, because any nonzero deviation $\boldsymbol{\delta}$ implies a larger value of the function. $\square$

Equivalence of problems (5.80) and (5.81) established by Theorem 5.7 allows one to reduce the solution of either problem to the solution of the other problem.

Note that linear systems of type (5.81) that have a self-adjoint positive definite matrix represent an important class of systems. Indeed, a general linear system

$$\boldsymbol{Ax} = \boldsymbol{f},$$

where $\boldsymbol{A} : \mathbb{R}^n \longmapsto \mathbb{R}^n$ is an arbitrary non-singular operator, can be easily reduced to a new system of type (5.81): $\boldsymbol{Cx} = \boldsymbol{g}$, where $\boldsymbol{C} = \boldsymbol{C}^* > 0$. To do so, one merely needs to set: $\boldsymbol{C} = \boldsymbol{A}^*\boldsymbol{A}$ and $\boldsymbol{g} = \boldsymbol{A}^*\boldsymbol{f}$. This approach, however, may only have a theoretical significance because in practice an extra matrix multiplication performed on a machine with finite precision is prone to generating large additional errors.

However, linear systems with self-adjoint matrices can also arise naturally. For example, many boundary value problems for elliptic partial differential equations can be interpreted as the Lagrange-Euler problems for minimizing certain quadratic functionals. It is therefore natural to expect that a "correct," i.e., appropriate, discretization of the corresponding variational problem will lead to a problem of minimizing a quadratic function on a finite-dimensional space. The latter problem will, in turn, be equivalent to the linear system (5.81) according to Theorem 5.7.

Exercises

1. Consider a quadratic function of two scalar arguments x_1 and x_2:

$$F(x_1,x_2) = x_1^2 + 2x_1x_2 + 4x_2^2 - 2x_1 + 3x_2 + 5.$$

Recast this function in the form (5.79) as a function of the vector argument $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ that belongs to the Euclidean space $\mathbb{R}^2$ supplied with the inner product $(\boldsymbol{x},\boldsymbol{y}) = x_1y_1 + x_2y_2$. Verify that $\boldsymbol{A} = \boldsymbol{A}^* > 0$, and solve the corresponding problem (5.81).

2.* Recast the function $F(x_1,x_2)$ from the previous exercise in the form:

$$F(\boldsymbol{x}) = [\boldsymbol{A}\boldsymbol{x},\boldsymbol{x}]_{\boldsymbol{B}} - 2[\boldsymbol{f},\boldsymbol{x}]_{\boldsymbol{B}} + c,$$

where $[\boldsymbol{x},\boldsymbol{y}]_{\boldsymbol{B}} = (\boldsymbol{B}\boldsymbol{x},\boldsymbol{y}), \boldsymbol{B} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$, and $\boldsymbol{A} = \boldsymbol{A}^*$ in the sense of the inner product $[\boldsymbol{x},\boldsymbol{y}]_{\boldsymbol{B}}$.

5.6 The Method of Conjugate Gradients

Consider a system of linear algebraic equations:

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{f}, \quad \boldsymbol{A} = \boldsymbol{A}^* > 0, \quad \boldsymbol{f} \in \mathbb{R}^n, \quad \boldsymbol{x} \in \mathbb{R}^n, \tag{5.82}$$

where $\mathbb{R}^n$ is an n-dimensional real vector space with the scalar product $(\boldsymbol{x},\boldsymbol{y})$, and $\boldsymbol{A} : \mathbb{R}^n \longmapsto \mathbb{R}^n$ is a self-adjoint positive definite operator with respect to this product. To introduce the method of conjugate gradients, we will use the equivalence of system (5.82) and the problem of minimizing the quadratic function $F(\boldsymbol{x}) = (\boldsymbol{A}\boldsymbol{x},\boldsymbol{x}) - 2(\boldsymbol{f},\boldsymbol{x})$ that was established in Section 5.5, see Theorem 5.7.

5.6.1 Construction of the Method

In the core of the method is the sequence of vectors $\boldsymbol{x}^{(p)} \in \mathbb{R}^n$, $p = 0,1,2,\dots$, which is to be built so that it would converge to the vector that minimizes the value of $F(\boldsymbol{x})$. Along with this sequence, the method requires constructing another sequence of vectors $\boldsymbol{d}^{(p)} \in \mathbb{R}^k$, $p = 0,1,2,\dots$, that would provide the so-called descent directions. In other words, we define $\boldsymbol{x}^{(p+1)} = \boldsymbol{x}^{(p)} + \alpha_p\boldsymbol{d}^{(p)}$, and choose the scalar α_p in order to minimize the composite function $F(\boldsymbol{x}^{(p)} + \alpha\boldsymbol{d}^{(p)})$.