NUMERICAL METHODS IN LINEAR ALGEBRA

I.1 ELEMENTS OF MATRIX CALCULUS

I.1.1 LR factorization of quadratic matrix

During solution of systems of linear equation there is often case to present a quadratic matrix in a form of product of two triangular matrices. This section is devoted to this problem.

Theorem 1.1.1. *If all determinants of form*

$$\Delta_k = \begin{vmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & & \\ a_{k1} & \cdots & a_{kk} \end{vmatrix}$$

are different from zero, the matrix $\mathbf{A} = [\mathbf{a_{ij}}]_{\mathbf{n}\times\mathbf{n}}$ *can be written in form*

$$\mathbf{A} = \mathbf{LR}, \tag{1.1.1}$$

where $\mathbf{L}$ *lower, and* $\mathbf{R}$ *upper triangular matrix.*

Triangular matrices $\mathbf{L}$ and $\mathbf{R}$ of order n are of following forms:

$$\mathbf{L} = [l_{ij}]_{n\times n} \qquad (l_{ij} = 0 \Leftarrow i < j), \tag{1.1.2}$$

$$\mathbf{R} = [r_{ij}]_{n\times n} \qquad (r_{ij} = 0 \Leftarrow i > j), \tag{1.1.3}$$

Decomposition (1.1.1), known as $\mathbf{LR}$ factorization (decomposition), is not unique, having in mind the equality

$$\mathbf{LR} = (c\mathbf{L})(\frac{1}{c}\mathbf{R}) \qquad \forall c \neq 0).$$

Nevertheless, if diagonal elements of matrix $\mathbf{R}$ (or $\mathbf{L}$) take fixed values, not one being equal to zero, the decomposition is unique. In regards to (1.1.2) and (1.1.3), and having in mind

$$a_{ij} = \sum_{k=1}^{max(i,j} l_{ik}r_{kj} \qquad (i, j = 1, \ldots, n)$$

the elements of matrices $\mathbf{L}$ and $\mathbf{R}$ can be easy determined by recursive procedure, giving in advance the values for elements $r_{ii}(\neq 0)$ or $l_{ii}(\neq 0) \quad (i = 1, \ldots, n)$. For example, if given numbers $r_{ii}(\neq 0) \quad (i = 1, \ldots, n)$, it holds

$$l_{11} = \frac{a_{11}}{r_{11}}$$

$$\left\{ \begin{array}{l} r_{1i} = \dfrac{a_{1i}}{l_{11}} \\ \\ l_{i1} = \dfrac{a_{i1}}{r_{11}} \end{array} \right\} \qquad (i = 2, \ldots, n);$$

$$\left\{ \begin{array}{l} \qquad\qquad l_{ii} = \dfrac{1}{r_{ii}}(a_{ii} - \displaystyle\sum_{k=1}^{i-1} l_{ik} r_{ki}) \\ \\ \left\{ \begin{array}{l} r_{ij} = \dfrac{1}{l_{ii}}(a_{ij} - \displaystyle\sum_{k=1}^{i-1} l_{ik} r_{kj}) \\ \\ l_{ji} = \dfrac{1}{r_{ii}}(a_{ji} - \displaystyle\sum_{k=1}^{i-1} l_{jk} r_{ki}) \end{array} \right\} \ (j = i+1, \ldots, n); \end{array} \right\} \ (i = 2, \ldots, n).$$

In similar way can be defined recursive procedure for determination of matrix elements of matrices $\mathbf{L}$ and $\mathbf{R}$, if the numbers $l_{ii}(\neq 0) i = 1, \ldots, n)$ are given in advance. In practical applications one usually takes $r_{ii} = 1 (i = 1, \ldots, n)$ or $l_{ii} = 1 (i = 1, \ldots, n)$.

Very frequent case in application is of multi-diagonal matrices, i.e. matrices with elements different from zero on the main diagonal and around the main diagonal. For example, if $a_{ij} \neq 0$ for $|i - j| \leq 1$ and $a_{ij} = 0$ for $|i - j| > 1$, the matrix is tri-diagonal. The elements of such a matrix are usually written as vectors $(a_2, \ldots, a_n), (b_1, \ldots, b_n), (c_1, \ldots, c_{n-1}$, i.e.

$$\mathbf{A} = \begin{bmatrix} b_1 & c_1 & 0 & \ldots & 0 & 0 \\ a_2 & b_2 & c_3 & & 0 & 0 \\ 0 & a_3 & b_3 & & 0 & 0 \\ \vdots & & & & & \\ 0 & 0 & 0 & & a_n & b_n \end{bmatrix}.$$

If $a_{ij} \neq 0$ $(|i-j| \leq 2)$ and $a_{ij} = 0$ $(|i-j| > 2)$, we have a case of five-diagonal matrix. Let us now suppose that tri-diagonal matrix (1.1.4) fulfills the conditions of Theorem 1.1.1. For decomposition of such a matrix it is enough to suppose that

$$\mathbf{L} = \begin{bmatrix} \beta_1 & 0 & 0 & \dots & 0 & 0 \\ \alpha_2 & \beta_2 & 0 & & 0 & 0 \\ 0 & \alpha_3 & \beta_3 & & 0 & 0 \\ \vdots & & & & & \\ 0 & 0 & 0 & & \alpha_n & \beta_n \end{bmatrix} \qquad (\beta_1\beta_2\dots\beta_n \neq 0)$$

and

$$\mathbf{R} = \begin{bmatrix} 1 & \gamma_1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \gamma_2 & & 0 & 0 \\ 0 & 0 & 1 & & 0 & 0 \\ \vdots & & & & & \\ 0 & 0 & 0 & & 0 & 1 \end{bmatrix}$$

By comparing corresponding elements of matrix A and matrix

$$\mathbf{LR} = \begin{bmatrix} \beta_1 & \beta_1\gamma_1 & 0 & \dots & 0 & 0 \\ \alpha_2 & \alpha_2\gamma_1 & \beta_2\gamma_2 & & 0 & 0 \\ 0 & \alpha_3 & \alpha_3\gamma_2+\beta_3 & & 0 & 0 \\ \vdots & & & & & \\ 0 & 0 & 0 & & \alpha_n & \alpha_n\gamma_{n-1}+\beta_n \end{bmatrix},$$

we get the following recursive formulas for determination of elements $\alpha_i, \beta_i, \gamma_i$:

$$\begin{array}{lll} & \beta_1 = b_1 & \gamma_1 = \frac{c_1}{\beta_1} \\ \alpha_1 = a_i & \beta_i = b_i - \alpha_i\gamma_{i-1} & \gamma_i = \frac{c_i}{\beta_i} \qquad (i = 2,\dots,n-1), \\ \alpha_n = a_n & \beta_n = b_n - \alpha_n\gamma_{n-1} & \end{array}$$

I.1.2 Matrix eigenvectors and eigenvalues

Definition 1.2.1. *Let* **A** *complex quadratic matrix of order* n*. Every vector* $\vec{x} \in \mathcal{C}^n$*, different from null-vector is named eigenvector of matrix* **A** *if there exists scalar* $\lambda \in C$ *such that*

$$\mathbf{A}\vec{\mathbf{x}} = \lambda\vec{x}. \tag{1.2.1}$$

Scalar λ *is then named the corresponding eigenvalue.*

Considering that (1.2.1) can be written in form

$$(\mathbf{A} - \lambda\mathbf{I})\vec{\mathbf{x}} = \vec{\mathbf{0}},$$

one can conclude that equation (1.2.1) has non-trivial solutions (in $\vec{x}$) if and only if $det(\mathbf{A} - \lambda\mathbf{I}) = \mathbf{0}$.

I.2 DIRECT METHODS IN LINEAR ALGEBRA

I.2.1 Introduction

Numerical problems in linear algebra can be classified in several groups:

1. Solution of system of linear algebraic equations

   $$\mathbf{A}\vec{x} = \vec{b},$$

   where **A** regular matrix, calculation of determinant of matrix **A**, and matrix **A** inversion;
2. Solution of arbitrary system of linear equations using less-square method;
3. Determination of eigenvalues and eigenvectors of given quadratic matrix;
4. Solution of problems in linear programming.

For solution of these problems, a number of methods is developed. They can be separated in two classes, as follows.

The first class contains so-called direct methods, known sometimes as exact methods. The basic characteristic of those methods is that after final number of transformations (steps) one gets the result. Presuming all operations being performed exact, the gained result would be absolutely exact. Of course, because the performed computations are performed with rounding intermediate results, the final result is of limited exactness.

The second class is made of iterative methods, obtaining the result after infinite number of steps. As started values at iterative methods are usually used the results obtained by some direct method.

In subchapter **1.3** the main characteristics of iterative methods used in linear algebra will be described. Let us note that at solution of systems with big number of equation, used for solution of partial differential equations, the iterative methods are usually used.

I.2.2 Gauss elimination with pivoting

Consider the system of linear algebraic equations

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n, \end{aligned} \tag{2.2.1}$$

or, in matrix form

$$\mathbf{A}\vec{x} = \vec{b}, \tag{2.2.2}$$

where

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & & \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}, \quad \vec{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}, \quad \vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

Suppose that system of equation (2.2.2) has an unique solution. It is very known that solutions of system (2.2.1), i.e. (2.2.2), can be expressed using Crammer's rules

$$x_i = \frac{\det \mathbf{A_i}}{\det \mathbf{A}} \quad (i = 1, 2, \cdots n),$$

where matrix $\mathbf{A_i}$ is obtained from matrix $\mathbf{A}$ by replacing i-th column by vector $\vec{b}$. Nevertheless, these formulas are inappropriate for practical calculations because for calculation of $n+1$ determinants one needs a big number of calculations. Namely, if we would like to calculate the value of determinant of n-th degree by developing of determinant through rows or columns, it would be necessary to proceed $S_n = n! - 1$ addings and $M_n \cong n!(e-1)$ multiplications $(n > 4)$, what gives the total number of calculations $P_n = M_n + S_n \cong n!e$. Supposing that one operation demands $100\mu s$ (what is the case with fast computers), the total time for calculation of value of determinant of order thirty $(n = 30)$ would take approximately $2.3 \cdot 10^{20}$ years. Generally speaking, such one procedure is practically unusable for determinants of order $n > 5$. One of the most suitable direct methods for solution of system of linear equations is Gauss method of elimination. This method is based on reduction of system (2.2.2), using equivalent transformations, to the triangular system

$$\mathbf{R}\vec{x} = \vec{c}, \tag{2.2.3}$$

where

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ & r_{22} & \dots & r_{2n} \\ & & \ddots & \\ & & & r_{nn} \end{bmatrix}, \quad \vec{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}.$$

System (2.2.3) is solved successively starting from the last equation. Namely,

$$x_n = \frac{c_n}{r_{nn}},$$
$$x_i = \frac{1}{r_{ii}}(c_i - \sum_{k=i+1}^{n} r_{ik}x_k) \quad (i = n-1, \dots, 1).$$

Let us note that coefficients $r_{ii} \neq 0$, because according to assumption the system (2.2.2), i.e. (2.2.3) has an unique solution.

We will show now how system (2.2.1) can be reduced to equivalent system with triangular matrix.

Supposing $a_{11} \neq 0$, let us compute first the factors

$$m_{i1} = \frac{a_{i1}}{a_{11}} \quad (i = 2, \dots, n),$$

and then, by multiplication of first equation in system (2.2.1) by m and subtracting from i-th equation, one gets the system of $n-1$ equations

$$\begin{aligned} a_{22}^{(2)}x_2 + \dots + a_{2n}^{(2)}x_n &= b_2^{(2)} \\ &\vdots \\ a_{n2}^{(2)}x_2 + \dots + a_{nn}^{(2)}x_n &= b_n^{(2)} \end{aligned} \tag{2.2.4}$$

where

$$a_{ij}^{(2)} = a_{ij} - m_{i1}a_{1j}, \quad b_i^{(2)} = b_i - m_{i1}b_1 \quad (i, j = 2, \dots, n).$$

Assuming $a_{22} \neq 0$, and applying the same procedure to (2.2.4), with

$$m_{i2} = \frac{a_{i2}}{a_{22}} \quad (i = 3, \ldots, n)$$

one gets the system of $n-2$ equations

$$\begin{aligned} a_{33}^{(3)} x_3 + \ldots + a_{3n}^{(3)} x_n &= b_3^{(3)} \\ &\vdots \\ a_{n3}^{(3)} x_3 + \ldots + a_{nn}^{(3)} x_n &= b_n^{(3)} \end{aligned}$$

where

$$a_{ij}^{(3)} = a_{ij}^{(2)} - m_{i2} a_{2j}, \quad b_i^{(3)} = b_i^{(2)} - m_{i2} b_2^{(2)} \quad (i, j = 3, \ldots, n).$$

Continuing this procedure, after $n-1$ steps, one gets the equation

$$a_{nn}^{(n)} x_n = b_n^{(n)}.$$

From the obtained systems, taking the first equations, one gets the system of equations

$$\begin{aligned} a_{11}^{(1)} x_1 + a_{12}^{(1)} x_2 + a_{13}^{(1)} x_3 + \cdots + a_{1n}^{(1)} x_n &= b_1^{(1)} \\ a_{22}^{(2)} x_2 + a_{23}^{(2)} x_3 + \cdots + a_{2n}^{(2)} x_n &= b_2^{(2)} \\ a_{33}^{(3)} x_3 + \cdots + a_{3n}^{(3)} x_n &= b_3^{(3)} \\ &\vdots \\ a_{nn}^{(n)} x_n &= b_n^{(n)} \end{aligned}$$

where we putted $a_{ij}^{(1)} = a_{ij}, \ b_i^{(1)} = b_i$.

The presented triangular reduction, or as often called Gauss elimination, is actually determination of coefficients

$$\begin{aligned} m_{ik} &= \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}, \\ a_{ij}^{(k+1)} &= a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}, \\ b_i^{(k+1)} &= b_i^{(k)} - m_{ik} b_k^{(k)} \quad (i, j = k+1, \ldots, n) \end{aligned}$$

for $k = 1, 2, \ldots, n-1$. Note that the elements of matrix $\mathbf{R}$ and vector $\vec{c}$ are given as

$$r_{ij} = a_{ij}^{(i)}, \;\; c_i = b_i^{(i)} \quad (i = 1, \ldots . n).$$

In order the presented reduction to exists, it is necessary to obtain the condition $a_{kk}^{(k)} \neq 0$. Elements $a_{kk}^{(k)}$ are known as main elements (pivotal elements or pivot). Assuming matrix $\mathbf{A}$ of system (2.2.2) being regular, the conditions $a_{kk}^{(k)} \neq 0$ are to be obtained by permutation of equations in system.

Moreover, from the point of view of exactness of results, it is necessary to use so known strategy of choice of pivotal elements. Modification of Gauss elimination method in this sense is called Gauss method with choice of pivotal element. In accordance to this method, for pivotal element in k-th elimination step one takes the element $a_{rk}^{(k)}$, for which holds

$$|a_{rk}^{(k)}| = \max_{k \leq i \leq n} |a_{ik}^{(k)}|,$$

with permutation of k-th and r-th row.

If one obtains in addition to permutation of equations the permutation of unknowns, it is the best way to take for pivotal element in the k-th elimination step the element $a_{rk}^{(k)}$, for which it holds

$$|a_{rs}^{(k)}| = \max_{k \leq i,j \leq n} |a_{ij}^{(k)}|$$

with permutation of k-th and r-th row (equation) and k-th and s-th column (unknown). Such method is called the method with total choice of pivotal element.

One can show (see Milovanović, G.V., *Numerical Analysis, Part I*, Niš, 1979. (Serbian)) that total number of calculations by applying Gauss method is

$$N(n) = \frac{1}{6}(4n^3 + 9n^2 - 7n).$$

For n big enough, one gets $N(n) \cong 2n^3/3$. It was long time opinion that Gauss method is optimal regarding number of computations . Nowadays, V. Strassen, by involving iterative algorithm for multiplying and inverse of matrices, gave a new method for solution of system of linear equations, by which the number of computations is of order $n^{\log_2 7}$. Strassen method is thus better than Gauss method $\log_2 7 < 3$.

Triangular reduction obtains simple computation of system determinant. Namely, it holds

$$det\mathbf{A} = a_{11}^{(1)} a_{22}^{(2)} \dots a_{nn}^{(n)}.$$

When used Gauss method with choice of pivotal element, one should take care about number of permutations of rows (and columns by using method of total choice of pivotal element), what influences the sign of determinant. This way of determinant calculation is high efficient. For example, for calculation of determinant of order $n = 30$, one needs $0.18sec$, presuming that one arithmetic operation takes $10\mu s$.

I.2.3. Matrix inversion using Gauss method

Let $\mathbf{A} = [a_{ij}]_{n\times n}$ be regular matrix and let

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & & & \\ x_{n1} & x_{n2} & \dots & x_{nn} \end{bmatrix} = [\,\vec{x}_1 \quad \vec{x}_2 \quad \dots \quad \vec{x}_n\,]$$

be its inverse matrix. Vectors $\vec{x}_1, \vec{x}_2, \dots \vec{x}_n$ are first, second,..., n-th column of matrix $\mathbf{X}$. Let us now define vectors $\vec{e}_1, \vec{e}_2, \dots \vec{e}_n$ as

$$\vec{e}_1 = [1\ 0 \dots 0]^T,\ \vec{e}_2 = [0\ 1\ 0 \dots 0]^T,\ , \dots, \vec{e}_n = [0\ 0 \dots 1]^T.$$

Regarding to equality

$$\mathbf{AX} = [\mathbf{A}\vec{x}_1 \ \mathbf{A}\vec{x}_2 \ \ldots \mathbf{A}\vec{x}_n] = \mathbf{I} = [\vec{e}_1 \ \vec{e}_2 \ldots \vec{e}_n],$$

the problem of determination of inverse matrix can reduce to solving of n systems of linear equations

$$\mathbf{A}\vec{x}_i = \vec{e}_i, \quad (i = 1, \ldots, n). \tag{2.3.1}$$

For solving of system (2.3.1) it is convenient to use Gauss method, taking in account that matrix **A** appears as a matrix of all systems, so that its triangular reduction shell be done once only. By this procedure all the transformations necessary for triangular reduction of matrix **A** should be applied to the unit matrix $\mathbf{I} = [\vec{e}_1\vec{e}_2 \ldots \vec{e}_n]$. too. In this way matrix **A** transforms to triangular matrix **R**, and matrix **I** to matrix $\mathbf{C} = [\vec{c}_1\vec{c}_2 \ldots \vec{c}_n]$. Finally, triangular systems of form

$$\mathbf{R}\vec{x}_i = \vec{c}_i, \quad (i = 1, \ldots, n)$$

should be solved.

I.2.4. Factorization methods

Factorization methods for solving of system of linear equations are based on factorization of matrix of system to product of two matrices in such form that enables reduction of system to two systems of equations which can be simple successive solved. In this section we will show up at the methods based on **LR** matrix factorization (see Section **I.1.1**).

Given the system of equations

$$\mathbf{A}\vec{x} = \vec{b}, \tag{2.4.1}$$

with quadratic matrix $\mathbf{A}$, with all main diagonal minors zero different. Then, based on Theorem 1.1.1, it exists factorization matrix $\mathbf{A} = \mathbf{LR}$, where L lower and R upper triangular matrix. The factorization is unique defined, if, for example, one adopts unit diagonal of matrix L. In this case, system (2.4.1), i.e. system $\mathbf{LR}\vec{x} = \vec{b}$ can be presented in equivalent form

$$\mathbf{L}\vec{y} = \vec{b}, \quad \mathbf{R}\vec{x} = \vec{y}. \tag{2.4.2}$$

Based on previous, for solving of system of equations (2.4.1), the following method can be formulated:

1. Put $l_{ii} = 1 \quad (i = 1, \ldots, n)$;
2. Determine other elements of matrix $\mathbf{L} = [\mathbf{l_{ij}}]_{\mathbf{n\times n}}$ and matrix $\mathbf{R} = [\mathbf{r_{ij}}]_{\mathbf{n\times n}}$ (see Section I.1.1);
3. Solve first system of equations in (2.4.2);
4. Solve second system of equations in (2.4.2).

Steps 3. and 4. are simple to be performed. Namely, let

$$\vec{b} = [b_1 \ b_2 \ldots b_n]^T, \quad \vec{y} = [y_1 \ y_2 \ldots y_n]^T, \quad \vec{x} = [x_1 \ x_2 \ldots x_n]^T.$$

Then

$$y_1 = b_1, \ y_i = b_i - \sum_{k=1}^{i-1} l_{ik} y_k \quad (i = 2, \ldots, n)$$

and

$$x_n = \frac{y_n}{r_{nn}}, \quad x_i = \frac{1}{r_{ii}}(y_i - \sum_{k=i+1}^{n} r_{ik} x_k) \quad (i = n-1, \ldots, 1).$$

The method presented is known in bibliography as method of Cholesky. In the case when matrix $\mathbf{A}$ is normal, i.e. symmetric and positive definite, the Cholesky method can be simplified. Namely, in this case one can take that $\mathbf{L} = \mathbf{R^T}$. . Thus,

the factorization of matrix $\mathbf{A}$ in form $\mathbf{A} = \mathbf{R^T R}$ should be performed. Based on formulas from Section I.1.1 for elements of matrix $\mathbf{R}$ it holds:

$$
\begin{aligned}
r_{11} &= \sqrt{a_{11}} \\
r_{1j} &= \frac{a_{1j}}{r_{11}} \qquad (j = 2, \ldots, n), \\
r_{ii} &= \sqrt{a_{ii} - \sum_{k=1}^{i-1} r_{ki}^2} \\
& \qquad\qquad\qquad\qquad\qquad\qquad (i = 2, \ldots, n). \\
r_{ij} &= \frac{1}{r_{ii}}(a_{ij} - \sum_{k=1}^{i-1} r_{ki} r_{kj}) \qquad (j = i+1, \ldots, n).
\end{aligned}
$$

In this case the systems (2.4.2) become

$$\mathbf{R^T}\vec{y} = \vec{b}, \qquad \mathbf{R}\vec{x} = \vec{y}.$$

Remark 2.4.1. *The determinant of normal matrix can be calculated by method of square root as*

$$\det \mathbf{A} = (r_{11}\ r_{22} \ldots r_{nn})^2.$$

Factorization methods are specially convenient for solving of systems of linear equations where matrix of systems does not change, but only free vector $\vec{b}$. Such systems are very frequent in engineering.

Now it will be shown that Gauss method of elimination can be interpreted as $\mathbf{LR}$ factorization of matrix $\mathbf{A}$. Take matrix $\mathbf{A}$ such that during the elimination process permutation of rows and columns should not be performed. Denote the starting system as $\mathbf{A^{(1)}}\vec{\mathbf{x}} = \vec{\mathbf{b}}^{\mathbf{(1)}}$. Gauss elimination procedure gives $n-1$

equivalent systems $\mathbf{A}^{(2)}\vec{x} = \vec{b}^{(2)} \ldots \mathbf{A}^{(n)}\vec{x} = \vec{b}^{(n)}$. where matrix $\mathbf{A}^{(k)}$ is of form

$$\mathbf{A}^{(k)} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \ldots & a_{1k}^{(1)} & \ldots & a_{1n}^{(1)} \\ & a_{22}^{(2)} & & a_{2k}^{(2)} & & a_{2n}^{(2)} \\ & & \ddots & \vdots & & \\ & & & a_{kk}^{(k)} & & a_{kn}^{(k)} \\ & & & \vdots & & \\ & & & a_{nk}^{(k)} & & a_{nn}^{(k)} \end{bmatrix}$$

Let us analyze modification of elements $a_{ij} (= a_{ij}^{(1)}$ during the process of triangular reduction. Because, for $k = 1, 2, \ldots, n-1$,

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)} \quad (i, j = k+1, \ldots, n),$$

and

$$a_{i1}^{(k+1)} = a_{i2}^{(k+1)} = \ldots = a_{ik}^{(k+1)} = 0 \quad (i = k+1, \ldots, n),$$

by summation we get

$$a_{ij} = a_{ij}^{(1)} = a_{ij}^{(i)} + \sum_{k=1}^{i-1} m_{ik} a_{kj}^{(k)} \quad (i \le j)$$

and

$$a_{ij} = a_{ij}^{(1)} = 0 + \sum_{k=1}^{i} m_{ik} a_{kj}^{(k)} \quad (i > j).$$

By defining $m_{ij} = 1$ $(i = 1, \ldots, n)$, the last two equalities can be given in form

$$a_{ij} = \sum_{k=1}^{p} m_{ik} a_{kj}^{(k)} \quad (i, j = 1, \ldots, n), \tag{2.4.3}$$

where $p = \min(i, j)$. Equality (2.4.3) is pointing out that Gauss elimination procedure gives **LR** factorization of matrix **A**, where

$$\mathbf{L} = \begin{bmatrix} 1 & & & \\ m_{21} & 1 & & \\ & & \ddots & \\ m_{n1} & m_{n2} & \ldots & 1 \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & \ldots & r_{1n} \\ & r_{22} & \ldots & r_{2n} \\ & & \ddots & \\ & & & r_{nn} \end{bmatrix}.$$

and $r_{ik} = a_{kj}^{(k)}$. During program realization of Gauss method in order to obtain **LR** factorization of matrix **A** , it is not necessary to use new memory space for matrix **L**, but it is convenient to load factors m_{ik} in the place of matrix **A** coefficients which are anulled in process of triangular reduction. In this way, after completed triangular reduction, in the memory space of matrix **A** will be memorized matrices **L** and **R**, according to following scheme:

$$\mathbf{A} \Rightarrow \mathbf{LR}$$

Consider that diagonal elements of matrix **L**, all equal to unit, should not be memorized.

Cholesky method, based on **LR** factorization, is used when matrix **A** fulfils conditions of Theorem 1.1.1. Nevertheless, usability of this method can be broaden to other systems with regular matrix, taking in account permutation of equations in system. For factorization is used Gauss elimination method with f pivoting. There will be $\mathbf{LR} = \mathbf{A}'$, where matrix $\mathbf{A}'$ is obtained from matrix **A** by finite number of row interchange. This means that in elimination process set of indices of pivot elements $I = (p_1, \ldots, p_{n-1}$, where p_k is number of row from which the main element is taken in k-th elimination step, should be memorized. By solving of system $\mathbf{A}\vec{x} = \vec{b}$, after accomplishing a process of factorization, according to set of indices I, coordinates of vector $\vec{b}$ should be permuted. In this way the transformed vector $\vec{b}'$ is obtained, so that solving of given system reduces to successive solving of triangular systems

$$\mathbf{L}\vec{y} = \vec{b}, \quad \mathbf{R}\vec{x} = \vec{y}.$$

I.3 ITERATIVE METHODS IN LINEAR ALGEBRA

I.3.1 Introduction

Consider system of linear equations

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n, \end{aligned} \tag{3.1.1}$$

which can be written in matrix form

$$\mathbf{A}\vec{x} = \vec{b}, \tag{3.1.2}$$

where

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & & \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}, \quad \vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \vec{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.$$

In this chapter we always suppose that system (3.1.1), i.e. (3.1.2) has an unique solution.

Iterative methods for solving systems (3.1.2) have as goal determination of solution $\vec{x}$ with exactness given in advance. Namely, starting with arbitrary vector $\vec{x}^{(0)}$ $(= [x_1^{(0)} \dots x_n^{(0)}]^T)$ by iterative methods one defines the series $\vec{x}^{(k)}$ $(= [x_1^{(k)} \dots x_n^{(k)}]^T)$ such that

$$\lim_{k \to +\infty} \vec{x}^{(k)} = \vec{x}.$$

I.3.2 Simple iteration method

One of the most simplest methods for solving of system of linear equations is method of simple iteration. For application of this method it is necessary to transform previously system (3.1.2) to the following equivalent form

$$\vec{x} = \mathbf{B}\vec{x} + \vec{\beta}. \tag{3.2.1}$$

Then, the method of simple iteration is given as

$$\vec{x}^{(k)} = \mathbf{B}\vec{x}^{(k-1)} + \vec{\beta} \quad (k = 1, 2, \ldots). \tag{3.2.2}$$

Starting from arbitrary vector $\vec{x}^{(0)}$ and using (3.2.2) one generates series $\{\vec{x}^{(k)}\}$, which under some conditions converges to solution of given system.

If

$$\mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & \ldots & b_{1n} \\ b_{21} & b_{22} & \ldots & b_{2n} \\ \vdots & & & \\ b_{n1} & b_{n2} & \ldots & b_{nn} \end{bmatrix}, \quad \text{and} \quad \vec{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix},$$

iterative method (3.2.2) can be written in scalar form

$$\begin{aligned} x_1^{(k)} &= b_{11}x_1^{(k-1)} + \ldots + b_{1n}x_n^{(k-1)} + \beta_1, \\ x_2^{(k)} &= b_{21}x_1^{(k-1)} + \ldots + b_{2n}x_n^{(k-1)} + \beta_2, \\ &\vdots \\ x_n^{(k)} &= b_{n1}x_1^{(k-1)} + \ldots + b_{nn}x_n^{(k-1)} + \beta_n, \end{aligned}$$

where $k = 1, 2, \ldots$.

One can prove (see [2]) that iterative process (3.2.2) converges if all eigenvalues of matrix $\mathbf{B}$ are by modulus less than one. Taking in account that determination of eigenvalues

of matrix is rather complicated, in practical applications of method of simple iteration only sufficient convergence conditions are examined. Namely, for matrix $\mathbf{B}$ several norms can be defined, as for example,

$$\|\mathbf{B}\|_1 = (\sum_{ij} b_{ij}^2)^{1/2},$$

$$\|\mathbf{B}\|_2 = \max_i \sum_{j=1}^{n} |b_{ij}|, \tag{3.2.3}$$

$$\|\mathbf{B}\|_3 = \max_j \sum_{i=1}^{n} |b_{ij}|$$

It is not difficult to prove that iterative process (3.2.2) converges if $\|B\| < 1$, for arbitrary starting vector $\vec{x}^{(0)}$.

I.3.3 Gauss-Seidel method

Gauss-Seidel method is constructed by modification of simple iterative method. As we have seen, at simple iteration method, the value of i-th component $x_i^{(k)}$ of vector $\vec{x}^{(k)}$ is obtained from values $x_1^{(k-1)}, \ldots, x_n^{(k-1)}$, i.e.

$$x_i^{(k)} = \sum_{j=1}^{n} b_{ij} x_j^{(k-1)} + \beta_i \quad (i = 1, \ldots, n;\ k = 1, 2, \ldots).$$

This method can be modified in this way so that for computation of $x_i^{(k)}$ are used all previously computed values $x_1^{(k)}, \ldots, x_{i-1}^{(k)}, x_i^{(k-1)}, \ldots, x_n^{(k-1)}$ and the rest will be part of vector , obtained in previous iteration, i.e

$$x_i^{(k)} = \sum_{j=1}^{i-1} b_{ij} x_j^{(k)} + \sum_{j=1}^{n} b_{ij} x_j^{(k-1)} + \vec{\beta}_i \quad (i = 1, \ldots, n;\ k = 1, 2, \ldots). \tag{3.3.1}$$

Noted modification of simple iterative method is known as Gauss-Seidel method.

The iterative process (3.3.1) can be written in matrix form too. Namely, let

$$\mathbf{B} = \mathbf{B}_1 + \mathbf{B}_2,$$

where

$$\mathbf{B}_1 = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ b_{21} & 0 & & 0 & 0 \\ \vdots & & & & \\ b_{n1} & b_{n2} & & b_{n,n-1} & b_{nn} \end{bmatrix}, \quad \mathbf{B}_2 = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ 0 & b_{22} & & b_{2n} \\ \vdots & & & \\ 0 & 0 & & b_{nn} \end{bmatrix}.$$

Then (3.3.1) becomes

$$\vec{x}^{(k)} = \mathbf{B}_1\vec{x}^{(k)} + \mathbf{B}_2\vec{x}^{(k-1)} + \vec{\beta} \quad (k = 1, 2, \dots). \tag{3.3.2}$$

Theorem 3.3.1. *For arbitrary vector* $\vec{x}^{(0)}$ *, iterative process* (3.3.2) *converges then and only then if all roots of equation*

$$\det[\mathbf{B}_2 - (\mathbf{I} - \mathbf{B}_1)\lambda] = \begin{bmatrix} b_{11} - \lambda & b_{12} & \dots & b_{1n} \\ b_{21}\lambda & b_{22} - \lambda & & b_{2n} \\ \vdots & & & \\ b_{n1}\lambda & b_{n2}\lambda & & b_{nn} - \lambda \end{bmatrix} = 0$$

are by modulus less than one.

I.4 PROGRAM REALIZATION

This subchapter is devoted to software realization of methods previously exposed in this chapter. For successful following of material in this subchapter it is necessary knowledge exposed in all previous subchapters of this chapter. In presented subprograms the matrices are treated as vectors.

Program 4.4.1. Subprogram for matrix transpose `MTRN` is of form:

```
      SUBROUTINE MTRN (A, B, N, M)
C
C TRANSPONTOVANJE MATRICE A
C
      DIMENSION A(1), B(1)
      IC=0
      DO 5 I=1, N
      IJ=I-N
      DO 5 J=1, M
      IJ=IJ+N
      IC=IC+1
    5 B(IC)=A(IJ)
      RETURN
      END
```

Parameters in the list of subprogram parameters have the following meaning:

`A` - input matrix of type $N \times M$, treated as vector of length `NM` (taken in form column by column);

`B` - output matrix of type $M \times N (\mathbf{B} = A^T)$. Matrix is treated in the same way as matrix **A**.

Program 4.4.2. Subprogram for multiplication of matrices `A` (of dimension $N \times M$) and `B` (of dimension $M \times xL$) is of form

```
      SUBROUTINE MMAT (A, B, C, N, M, L)
C
C MATRICA A TIPA N*M
C MATRICA B TIPA M*L
C MATRICA C TIPA N*L
C MNOZENJE MATRICA C=A*B
C
      DIMENSION A(1), B (1), C (1)
      IC=0
      I2=-M
      DO 5 J=1,L
```

```
      I2=I2+M
      DO 5 I=1, N
      IC=IC+1
      IA=I-N
      IB=I2
      C(IC)=0.
      DO 5 K=1, M
      IA=IA+N
      IB=IB+1
    5 C(IC)=C(IC) + A(IA)*B(IB)
      RETURN
      END
```

Output matrix C ($\mathbf{C} = \mathbf{A} \times \mathbf{B}$) is of dimension $N \times L$.

Program 4.4.3. Let us write a program for computing matrix $\mathbf{B}^T\mathbf{A}$, by using previously given subprograms, for given matrices $\mathbf{A}$ and $\mathbf{B}$. Let matrix $\mathbf{A}$ be of type $N \times M$, and matrix $\mathbf{B}$ of type $N \times K$ (with maximal number of matrix elements for both matrices 100).

This program has the following form:

```
      DIMENSION A(100), B(100), C(100)
      OPEN(8,FILE='MTMM.IN')
      OPEN(5,FILE='MTMM.OUT')
      READ (8,10) N,M,K
   10 FORMAT (3I2)
      NM=N*M
      NK=K*M
      KM=K*M
      READ (8,20) (A(I), I=1, NM), (B(I), I=1, NK)
   20 FORMAT(16F5.0)
      CALL MTRN( B, C, N, K)
      CALL MMAT (C, A, B, K, N, M)
      WRITE (5,30) ((B(J), J=I, KM, K), I=1, K)
   30 FORMAT (5X, 'MATRIX C=B(TR)* A'// (2X,4F6.1))
      CLOSE(8)
      CLOSE(5)
      STOP
      END
```

Test of program, being proceeded with matrices

$$\mathbf{A} = \begin{bmatrix} -1 & 3 & 0 & 2 \\ 1 & 4 & 1 & 5 \\ 0 & 1 & -2 & 0 \\ -2 & 3 & 1 & 3 \end{bmatrix}, \quad \text{and} \quad \mathbf{B} = \begin{bmatrix} 1 & -3 & 0 \\ 0 & 4 & -6 \\ 2 & -1 & 2 \\ -1 & 5 & 12 \end{bmatrix},$$

gave the following result:

```
 MATRIX C=B(TR)* A
  1.0   2.0  -5.0  -1.0
 -3.0  21.0  11.0  29.0
 -8.0 -19.0  -9.0 -27.0
```

Program 4.4.4. Method of Cholesky for solving of system of linear equations (see subchapter 4.2.4) can be realized in the following way:

```
C=====================================================
C        CHOLESKY  METHOD
C=====================================================
      DIMENSION A(10,10), B(10)
      OPEN(8,FILE='CHOLESKY.IN')
      OPEN(5,FILE='CHOLESKY.OUT')
   33 READ(8,100)N
  100 FORMAT(I2)
      IF(N)11,22,11
   11 READ(8,101)(B(I),I=1,N)
  101 FORMAT(8F10.4)
C   READ IN THE UPPER MATRIX TRIANGLE OF A
      READ(8,101)((A(I,J),J=1,N),I=1,N)
      WRITE(5,102)N
  102 FORMAT(/ 5X,'MATRIX DIMENSION =',I3//
     1 5X,'MATRICA  A',
     2 <(N-1)*12+3>X,'VEKTOR B'/)
      WRITE(5,103)((A(I,J),J=1,N),B(I),I=1,N)
  103 FORMAT(1X,<N>F12.7,F13.7)
C   FACTORIZATION OF MATRIX  A TO THE FORM  A=L*R
      DO 10 I=2,N
   10 A(1,I)=A(1,I)/A(1,1)
```

```
      DO 25 I=2,N
      I1=I-1
      S=A(I,I)
      DO 20 K=1,I1
   20 S=S-A(I,K)*A(K,I)
      A(I,I)=S
      IF(I.EQ.N) GO TO 40
      IJ=I+1
      DO 25 J=IJ,N
      S=A(I,J)
      T=A(J,I)
      DO 30 K=1,I1
      S=S-A(I,K)*A(K,J)
   30 T=T-A(J,K)*A(K,I)
      A(I,J)=S/A(I,I)
   25 A(J,I)=T
   40 WRITE(5,107)
  107 FORMAT(//5X,'MATRIX L'/)
      DO 111 I=1,N
  111 WRITE(5,103)(A(I,J),J=1,I)
      WRITE(5,108)
  108 FORMAT(//5X,'MATRIX R'/)
      N1=N-1
      DO 222 I=1,N1
      II=I+1
      M=N-I
  222 WRITE(5,99) (A(I,J),J=II,N)
      WRITE(5,99)
   99 FORMAT(<12*I-8>X,'1.0000000',<M>F12.7)
C    OBTAINING THE VECTOR OF SOLUTIONS
      B(1)=B(1)/A(1,1)
      DO 55 I=2,N
      I1=I-1
      DO 45 K=1,I1
   45 B(I)=B(I)-A(I,K)*B(K)
   55 B(I)=B(I)/A(I,I)
      DO 50 J=1,N1
      I=N-J
      I1=I+1
      DO 50 K=I1,N
   50 B(I)=B(I)-A(I,K)*B(K)
      WRITE(5,109)
  109 FORMAT(//13X,'VEKTOR OF SOLUTIONS'/)
      WRITE(5,104)(B(I),I=1,N)
```

```
  104 FORMAT(12X,F12.7)
      GO TO 33
   22 CLOSE(5)
      CLOSE(8)
      STOP
      END
```

For factorization of matrix $\mathbf{A}(=\mathbf{LR})$ we take in upper triangular matrix $\mathbf{R}$ unit diagonal, i.e. $r_{ii} = 1$ $(i = 1, \ldots, n)$. Program is organized in this way so that matrix $\mathbf{A}$ transforms to matrix $\mathbf{A}_1$, which lower triangle (including main diagonal) is equal to matrix $\mathbf{L}$, and strict upper triangle to matrix $\mathbf{R}$. Note that diagonal elements in matrix $\mathbf{R}$ are not memorized, but only formally printed, using statement `FORMAT`. Note also that in Section 4.2.4. the unit diagonal has been adopted into matrix $\mathbf{L}$.

By applying this program to the applicable system of equations, the following results are obtained:

```
          MATRIX DIMENSION =  4
          MATRICA  A                                          VEK-■
TOR B
         1.0000000   4.0000000   1.0000000   3.0000000    9.0000000
          .0000000  -1.0000000   2.0000000  -1.0000000     .0000000
         3.0000000  14.0000000   4.0000000   1.0000000   22.0000000
         1.0000000   2.0000000   2.0000000   9.0000000   14.0000000
          MATRIX L
         1.0000000
          .0000000  -1.0000000
         3.0000000   2.0000000   5.0000000
         1.0000000  -2.0000000  -3.0000000   2.0000000
          MATRIX R
         1.0000000   4.0000000   1.0000000   3.0000000
                     1.0000000  -2.0000000   1.0000000
                                 1.0000000  -2.0000000
                                 1.0000000
                  VEKTOR OF SOLUTIONS
                    1.0000000
                    1.0000000
```

```
 1.0000000
 1.0000000
```

Program 4.4.5. In similar way can be realized square root method for solution of system of linear equations with symmetric, positive definite matrix. In this case it is enough to read in only main diagonal elements of matrix **A**, and, for example, elements from upper triangle.

The program and output listing for given system of equations are given in the following text. Note that from the point of view of memory usage it is convenient to treat matrix **A** as a vector. Nevertheless, due to easier understanding, we did not follow this convenience on this place.

Program is organized in this way so that, in addition to solution of system of equation, the determinant of system is also obtained. In output listing the lower triangle of symmetric matrix is omitted.

```
$DEBUG
C================================================
C  SOLUTION OF SYSTEM OF LINEAR EQUATIONS
C       BY SQARE ROOT METHOD
C================================================
      DIMENSION A(10,10),B(10)
      OPEN(8,FILE='SQR.IN')
      OPEN(5,FILE='SQR.OUT')
    3 READ(8,100)N
  100 FORMAT(I2)
      IF(N) 1,2,1
C READ IN VECTOR B
    1 READ(8,101) (B(I),I=1,N)
  101 FORMAT(8F10.4)
C READ IN UPPER TRIANGULAR PART OF MATRIX  A
      READ(8,101)((A(I,J),J=I,N),I=1,N)
      WRITE(5,102)
  102 FORMAT(////5X,'MATRIX OF SYSTEM'/)
      WRITE(5,99)((A(I,J),J=I,N),I=1,N)
   99 FORMAT(<12*I-11>X,<N-I+1>F12.7)
```

```
      WRITE(5,105)
  105 FORMAT(//5X,'VECTOR OF FREE MEMBERS'/)
      WRITE(5,133)(B(I),I=1,N)
  133 FORMAT(1X,10F12.7)
C  OBTAINING OF ELEMENTS OF UPPER TRIANGULAR MATRIX
      A(1,1)=SQRT(A(1,1))
      DO 11 J=2,N
   11 A(1,J)=A(1,J)/A(1,1)
      DO 12 I=2,N
      S=0.
      IM1=I-1
      DO 13 K=1,IM1
   13 S=S+A(K,I)*A(K,I)
      A(I,I)=SQRT(A(I,I)-S)
      IF(I-N) 29,12,29
   29 IP1=I+1
      DO 14 J=IP1,N
      S=0.
      DO 15 K=1,IM1
   15 S=S+A(K,I)*A(K,J)
   14 A(I,J)=(A(I,J)-S)/A(I,I)
   12 CONTINUE
C  CALCULATION OF DETERMINANT
      DET=1.
      DO 60 I=1,N
   60 DET=DET*A(I,I)
      DET=DET*DET
C SOLUTION OF SYSTEM  L*Y=B
      B(1)=B(1)/A(1,1)
      DO 7 I=2,N
      IM1=I-1
      S=0.
      DO 8 K=1,IM1
    8 S=S+A(K,I)*B(K)
      P=1./A(I,I)
    7 B(I)=P*(B(I)-S)
C
C SOLUTION OF SYSTEM  R*X=Y
C MEMORIZING OF RESULTS INTO VECTOR  B
C
      B(N)=B(N)/A(N,N)
      NM1=N-1
      DO 30 II=1,NM1
      JJ=N-II
```

```
      S=0.
      JJP1=JJ+1
      DO 50 K=JJP1,N
   50 S=S+A(JJ,K)*B(K)
   30 B(JJ)=(B(JJ)-S)/A(JJ,JJ)
C
C PRINTING  OF RESULTS
C
      WRITE (5,201)
  201 FORMAT(//5X,'MATRIX  R'/)
      Pause 1
C     DO 222 I=1,N
  222 WRITE(5,199)((A(I,J),J=I,N),I=1,N)
  199 FORMAT(<12*I-11>X,<N-I+1>F12.7)
      WRITE(5,208) DET
  208 FORMAT(//5X,'SYSTEM DETERMINANT  D=',F11.7/)
      WRITE(5,109)
  109 FORMAT(//5X,'SYSTEM SOLUTION '/)
      WRITE(5,133)(B(I),I=1,N)
      GO TO 3
    2 CLOSE(5)
      CLOSE(8)
      STOP
      END
```

```
   MATRIX OF SYSTEM
                          3.0000000
                           .0000000
                          1.0000000
                          2.0000000
                          1.0000000
                          1.0000000
   VECTOR OF FREE MEMBERS
  4.0000000   3.0000000   3.0000000
   MATRIX  R
                          1.7320510
                           .0000000
                           .5773503
                          1.4142140
                           .7071068
                           .4082483
   SYSTEM DETERMINANT  D=  1.0000000
```

```
      SYSTEM SOLUTION
      .9999999    .9999998   1.0000000
```

Program 4.4.6. Method of factorization for solution of systems of linear equations based on Gauss elimination with choice of pivotal element (see Sections I.2.2 and I.2.4) can be programmable realized using the following subprograms:

```
      SUBROUTINE LRFAK(A,N,IP,DET,KB)
      DIMENSION A(1),IP(1)
      KB=0
      N1=N-1
      INV=0
      DO 45 K=1,N1
      IGE=(K-1)*N+K
C
C  FINDING THE PIVOTAL ELEMENT IN  K-TH
C  ELIMINATION STEP
C
      GE=A(IGE)
      I1=IGE+1
      I2=K*N
      IMAX=IGE
      DO 20 I=I1,I2
      IF(ABS(A(I))-ABS(GE)) 20,20,10
 10   GE=A(I)
      IMAX=I
 20   CONTINUE
      IF(GE)25,15,25
 15   KB=1
C
C  MATRIX OF SYSTEM IS SINGULAR
C
      RETURN
 25   IP(K)=IMAX-N*(K-1)
      IF(IP(K)-K) 30,40,30
 30   I=K
      IK=IP(K)
C
C  ROW PERMUTATION
C
      DO 35 J=1,N
```

```
      S=A(I)
      A(I)=A(IK)
      A(IK)=S
      I=I+N
 35   IK=IK+N
      INV=INV+1
C
C  K-TH ELIMINATION STEP
C
 40   DO 45 I=I1,I2
      A(I)=A(I)/GE
      IA=I
      IC=IGE
      K1=K+1
      DO 45 J=K1,N
      IA=IA+N
      IC=IC+N
 45   A(IA)=A(IA)-A(I)*A(IC)
C
C  CALCULATION OF DETERMINANT
C
      DET=1.
      DO 50 I=1,N
      IND=I+(I-1)*N
 50   DET=DET*A(IND)
      IF(INV-INV/2*2) 55,55,60
 60   DET=-DET
 55   RETURN
      END
C
C
      SUBROUTINE RSTS(A,N,IP,B)
      DIMENSION A(1),IP(1),B(1)
C
C  SUCCESSIVE SOLUTION OF TRIANGULAR SYSTEMS
C
      N1=N-1
C  VECTOR B PERMUTATION
      DO 10 I=1,N1
      I1=IP(I)
      IF(I1-I) 5,10,5
 5    S=B(I)
      B(I)=B(I1)
      B(I1)=S
```

```
 10   CONTINUE
C  SOLUTION OF LOWER TRIANGULAR SYSTEM
      DO 15 K=2,N
      IA=-N+K
      K1=K-1
      DO 15 I=1,K1
      IA=IA+N
 15   B(K)=B(K)-A(IA)*B(I)
C  SOLUTION OF UPPER TRIANGULAR SYSTEM
      NN=N*N
      B(N)=B(N)/A(NN)
      DO 25 KK=1,N1
      K=N-KK
      IA=NN-KK
      I=N+1
      DO 20 J=1,KK
      I=I-1
      B(K)=B(K)-A(IA)*B(I)
 20   IA=IA-N
 25   B(K)=B(K)/A(IA)
      RETURN
      END
```

Parameters in subprogram list of LRFAK are of following meaning:

A - Ulazna matrica reda N memorisana kao niz kolona po kolona. Posle N-1 eliminacionih koraka matrica A se transformiše u matricu koja sadrži trougaone matrice L i R (videti odeljak I.2.4);

N - red matrice A;

IP - vektor dužine N-1, koji se formira u procesu eliminacije i predstavlja niz indeksa glavnih elemenata (videti odeljak I.2.4);

DET - izlazna veličina koja daje vrednost determinante matrice sistema A, kao proizvod elemenata na glavnoj dijagonali u matrici R, sa tačnošću do na znak. Ova vrednost se koriguje znakom, na kraju potprograma, imajući u vidu broj permutacija vrsta u matrici u toku eliminacionog procesa;

KB - kontrolni broj sa vrednostima KB=0 ako je faktorizacija korektno izvedena i KB=1 ako je matrica sistema singularna. U poslednjem slučaju LR faktorizacija ne egzistira;

Potprogram RSTS sukcesivno rešava sisteme jednačina (2.3.4). Parametri potprogramskoj listi imaju sledeće značenje:

A - matrica dobijena u potprogramu LRFAK;

N - red matrice A;

IP - vektor dobijen u potprogramu LRFAK;

B - vektor slobodnih članova u sistemu jednačina koji se rešava. Ovaj vektor se transformiše u vektor rešenja datog sistema.

Glavni program je organizovan tako da se, najpre, data matrica A faktorizuje, pomoću potprograma LRFAK, a zatim je mogućno rešiti sistem jednačina $\mathbf{A}\vec{x} = \vec{b}$ za proizvoljan broj različitih vektora $\vec{b}$, pozivanjem potprograma **RSTS**.

Glavni program i izlazna lista imaju oblik:

```
      DIMENSION A(100),B(10),IP(9)
      OPEN(8,FILE='FACTOR.IN')
      OPEN(5,FILE='FACTOR.OUT')
      READ(8,5)N
    5 FORMAT(I2)
      NN=N*N
      READ(8,10)(A(I),I=1,NN)
   10 FORMAT(16F5.0)
      WRITE(5,34)
   34 FORMAT(1H1,5X,'MATRICA A'/)
      DO 12 I=1,N
   12 WRITE(5,15)(A(J),J=I,NN,N)
   15 FORMAT(10F10.5)
      CALL LRFAK(A,N,IP,DET,KB)
      IF(KB) 20,25,20
   20 WRITE(5,30)
   30 FORMAT(1H0,'MATRICA JE SINGULARNA'//)
      GO TO 70
   25 WRITE(5,35)
   35 FORMAT(1H0,5X,'FAKTORIZOVANA MATRICA'/)
      DO 55 I=1,N
```

```
   55 WRITE(5,15)(A(J),J=I,NN,N)
      WRITE(5,75)DET
   75 FORMAT(/5X,'DETERMINANTA MATRICE A='F10.6/)
   50 READ(8,10,END=70) (B(I),I=1,N)
      WRITE(5,40)(B(I),I=1,N)
   40 FORMAT(/5X,'VEKTOR B'//(10F10.5))
      CALL RSTS(A,N,IP,B)
      WRITE(5,45) (B(I),I=1,N)
   45 FORMAT(/5X,'RESENJE'//(10F10.5))
      GO TO 50
   70 CLOSE(5)
      CLOSE(8)
      STOP
      END
```

```
1     MATRICA A
   3.00000   1.00000   6.00000
   2.00000   1.00000   3.00000
   1.00000   1.00000   1.00000
0     FAKTORIZOVANA MATRICA
   3.00000   1.00000   6.00000
    .33333    .66667  -1.00000
    .66667    .50000   -.50000
     DETERMINANTA MATRICE A=  1.000000
     VEKTOR B
   2.00000   7.00000   4.00000
     RESENJE
  18.99999  -7.00000  -8.00000
     VEKTOR B
   1.00000   1.00000   1.00000
     RESENJE
    .00000   1.00000    .00000
```

Program 4.4.7. Korišćenjem potprograma LRFAK i RSTS i imajući u vidu odeljak 4.2.3, lako se može obrazovati program za inverziju matrica. Odgovarajući program i izlazni rezultat (za matricu iz aprethodnog primera) imaju oblik:

```
C=============================================
C   INVERZIJA MATRICE
```

```
C================================================
      DIMENSION A(100), B(10), IP(9),AINV(100)
      open(8,file='invert.in')
      open(5,file='invert.out')
      READ(8,5) N
    5 FORMAT(I2)
      NN=N*N
      READ(8,10)(A(I),I=1,NN)
   10 FORMAT(16F5.0)
      WRITE(5,34)
   34 FORMAT(1H1, 5X, 'MATRICA A'/)
      DO 12 I=1,N
   12 WRITE(5,15) (A(J),J=I,NN,N)
   15 FORMAT(10F10.5)
      CALL LRFAK(A,N,IP,DET,KB)
      IF(KB) 20,25,20
   20 WRITE(5,30)
   30 FORMAT(1H0,'MATRICA A JE SINGULARNA'//)
      GO TO 70
   25 DO 45 I=1,N
      DO 40 J=1,N
   40 B(J)=0.
      B(I)=1.
      CALL RSTS(A,N,IP,B)
      IN=(I-1)*N
      DO 45 J=1,N
      IND=IN+J
   45 AINV(IND)=B(J)
      WRITE(5,50)
   50 FORMAT(1H0,5X,'INVERZNA MATRICA'/)
      DO 55 I=1,N
   55 WRITE(5,15)(AINV(J),J=I,NN,N)
   70 CLOSE(5)
      CLOSE(8)
      STOP
      END



1     MATRICA A
   3.00000   1.00000   6.00000
   2.00000   1.00000   3.00000
   1.00000   1.00000   1.00000
0     INVERZNA MATRICA
```

```
-2.00000   5.00000  -3.00000
 1.00000  -3.00000   3.00000
 1.00000  -2.00000   1.00000
```

Program 4.4.8. Sastavimo sada program za rešavanje sistema linearnih jednačina oblika $\vec{x} = \mathbf{B}\vec{x} + \beta$, metodom proste iteracije (videti odeljak 4.3.2). S obzirom da metod proste iteracije konvergira kada je norma matrice **B** manja od jedinice, to ćemo za ispitivanje ovog uslova obrazovati potprogram `NORMA`, po kome se u zavistnosti od k (u potprogramu `K` izračunavaju norme $(k = 1, 2, 3)$ saglasno formuli (3.2.3) iz odeljka 4.3.2. Parametri u listi imaju sledeće značenje:

`A` - matrica memorisana kao vektor, čija se norma traži;

`N` - red matrice;

`K` - broj koji definiše normu `(K=1,2,3)`;

`ANOR` - odgovarajuća norma matrice `A`.

```
      SUBROUTINE NORMA(A,N,K,ANOR)
      DIMENSION A(1)
      NU=N*N
      ANOR=0
      GO TO (10, 20,40),K
10    DO 15 I=1,NU
15    ANOR=ANOR+A(I)**2
      ANOR=SQRT(ANOR)
      RETURN
20    DO 25 I=1,N
      L=-N
      S=0.
      DO 30 J=1,N
      L=L+N
      IA=L+I
30    S=S+ABS(A(IA))
      IF(ANOR-S) 35,25,25
35    ANOR=S
25    CONTINUE
      RETURN
40    L=-N
      DO 50 J=1,N
```

```
      S=0.
      L=L+N
      DO 45 I=1,N
      LI=L+I
45    S=S+ABS(A(LI))
      IF(ANOR-S) 55,50,50
55    ANOR=S
50    CONTINUE
      RETURN
      END
```

Glavni program je organizovan tako da se pre početka iterativnog procesa ustanovljava konvergencija. Naime, ukoliko je bar jedna od normi $\|\mathbf{B}\| < 1$ $(k = 1, 2, 3)$, prelazi se na iterativni proces, dok se u protivnom slučaju štampa poruka da uslovi za konvergenciju nisu zadovoljeni i u tom slučaju se program završava.

Za množenje matrice $\mathbf{B}$ vektorom $\vec{x}^{(k+1)}$ koristimo potprogram **MMAT**, koji je dat u 4.4.2. Za početni vektor uzimamo vektor $\vec{x}^{(0)}$.

Kao kriterijum za zavretak iterativnog procesa usvojili smo

$$|x_i^{(k)} - x_i^{(k-1)}|leq\varepsilon \quad (i = 1, \ldots, n)$$

.

Na izlazu štampamo poslednju iteraciju koja zadovoljava gornji kriterijum.

```
      DIMENSION B(100), BETA(10), X(10), X1(10)
      OPEN(8,FILE='ITER.IN')
      OPEN(5,FILE='ITER.OUT')
      READ(8,5) N, EPS
    5 FORMAT(I2,E5.0)
      NN=N*N
      READ(8,10)(B(I),I=1,NN),(BETA(I),I=1,N)
   10 FORMAT(16F5.1)
      WRITE(5,13)
   13 FORMAT(1H1,5X,'MATRICA B', 24X,'VEKTOR BETA')
```

```
      DO 15 I=1,N
   15 WRITE(5,20) (B(J),J=I,NN,N),BETA(I)
   20 FORMAT(/2X,4F8.1,5X,F8.1)
      DO 30 K=1,3
      CALL NORMA(B,N,K,ANOR)
      IF(ANOR-1.) 25,30,30
   30 CONTINUE
      WRITE(5,35)
   35 FORMAT(5X,'USLOVI ZA KONVERGENCIJU'
     1' NISU ZADOVOLJENI')
      GO TO 75
   25 ITER=0
      DO 40 I=1,N
   40 X(I)=BETA(I)
   62 ITER=ITER+1
      CALL MMAT(B,X,X1,N,N,1)
      DO 45 I=1,N
   45 X1(I)=X1(I)+BETA(I)
      DO 55 I=1,N
      IF(ABS(X1(I)-X(I))-EPS)55,55,60
   55 CONTINUE
      WRITE(5,42)ITER
   42 FORMAT(/3X,I3,'.ITERACIJA'/)
      WRITE(5,50)(I,X1(I),I=1,N)
   50 FORMAT(3X,4(1X,'X(',I2,')=',F9.5))
      GO TO 75
   60 DO 65 I=1,N
   65 X(I)=X1(I)
      GO TO 62
   75 CLOSE(8)
      CLOSE(5)
      STOP
      END
```

Uzimajući tačnost $\varepsilon = 10^{-5}$, za jedan konkretan sistem jednačina četvrtog reda (videti izlaznu listu) dobijamo rešenje u četrnaestoj iteraciji.

```
1     MATRICA B                          VEKTOR BETA
       -.1      .4      .1      .1            .7
        .4     -.1      .1      .1            .7
        .1      .1     -.2      .2           1.2
```

```
          .1      .1      .2     -.2          -1.6
       14.ITERACIJA
       X( 1)=  1.00000 X( 2)=  1.00000 X( 3)=  1.00000 X(
4)= -1.00000
```

Program 4.4.9. Kod rešavanja sistema sa velikim brojem jednačina pojavljuje se problem smeštanja matrice sistema u centralnu memoriju računara. U tim slučajevima koristimo virtualnu memoriju na disku. Sledeći program je realizovan za takve sisteme, a zasnivan je na Gaussovoj eliminaciji (bez izbora glavnog elementa. U programu je obezbe—eno mesto za matricu stotog reda. Medjutim, po potrebi se, bez teškoća, mogu rešavati i sistemi jednačina proizvoljnog reda, uz proširenje datoteke na disku. Program je realizovan u dvostrukoj tačnosti.

```
C PROGRAM ZA RESAVANJE VELIKIH SISTEMA JEDNACINA
C DATOTEKA VIRTUELNE MEMORIJE
      REAL*8 A(100) ,B(100),X(100), DT, E1, E2,Z1
C     DEFINE FILE 7(102,400,U,KL)
      OPEN(7,FILE='VELIKI',ACCESS='DIRECT',
     1 FORM='BINARY', MODE='READWRITE', RECL=800)
      OPEN(8,FILE='VELIKI.IN')
      OPEN(5,FILE='VELIKI.OUT')
C  N=BROJ JEDNACINA LINEARNOG SISTEMA (N>1)
      READ(8,10) N
   10 FORMAT(I3)
C UPISIVANJE PARAMETARA LINEARNOG SISTEMA
C U DATOTEKU 7 NA DISKU
      DO 11 L=1,N
      READ(8,20) (A(I),I=1,N)
   20 FORMAT (5D16.10)
      KL=L
   11 WRITE(7,REC=KL) (A(I),I=1,N)
      READ(8,20) (B(I),I=1,N)
      KL=N+1
      WRITE(7,REC=KL) (B(I),I=1,N)
      PAUSE 1
C RESAVANJE LINEARNOG SISTEMA
```

```
C PRIMENOM VIRTUELNE MEMORIJE
      KL=N+1
      READ(7,REC=KL)B
      N1=N-1
      DO 1 K=1,N1
      KL=K
      READ(7,REC=KL)A
      N2=K+1
      IF(A(K).NE.0.D0) GO TO 111
      DO 2 K1=N2,N
      KL=K1
      READ(7,REC=KL)X
      IF(X(K).NE.0.D0) GOTO 21
      DO 3 L=K,N
      Z1=A(L)
      A(L)=X(L)
      X(L)=Z1
    3 CONTINUE
      KL=K
      WRITE(7,REC=KL)A
      KL=K1
      WRITE(7,REC=KL)X
      Z1=B(K)
      B(K)=B(K1)
      B(K1)=Z1
      GO TO 111
   21 CONTINUE
    2 CONTINUE
      GO TO 222
  111 E1=B(K)/A(K)
      DO 4 J=N2,N
      KL=J
      READ(7,REC=KL)X
      B(J)=B(J)-X(K)*E1
      E2=X(K)/A(K)
      DO 5 I=K,N
      X(I)=X(I)-A(I)*E2
    5 CONTINUE
      KL=J
      WRITE(7,REC=KL)X
    4 CONTINUE
    1 CONTINUE
      DO 6 KIN=2,N
      K=N+2-KIN
```

```
        KL=K
        READ(7,REC=KL)A
        E1=B(K)/A(K)
        N3=K-1
        DO 7 J=1,N3
        KL=J
        READ(7,REC=KL)X
        B(J)=B(J)-X(K)*E1
        X(K)=0.D0
        KL=J
        WRITE(7,REC=KL)X
      7 CONTINUE
      6 CONTINUE
        DT=1.D0
        DO 8 L=1,N
        KL=L
        READ(7,REC=KL)A
        X(L)=B(L)/A(L)
        DT=DT*A(L)
      8 CONTINUE
        KL=N+1
        WRITE(7,REC=KL)B
        KL=N+2
        WRITE(7,REC=KL)X
        GO TO 333
    222 DT=0.D0
        DO 9 L=1,N
        X(L)=0.D0
      9 CONTINUE
    333 CONTINUE
C STAMPANJE RESENJA SISTEMA JEDNACINA
        KL=N+2
        READ(7,REC=KL)X
        WRITE(5,30) (X(I),I=1,N)
     30 FORMAT(2D24.16)
        CLOSE(8)
        CLOSE(7)
        CLOSE(5)
        END
```

Program 4.4.10. Obrazujmo program za nalaženje matrice $\mathbf{S}, = e^{\mathbf{A}}$ gde je $\mathbf{A}$ data kvadratna matrica reda n, korišćenjem

formule

$$e^{\mathbf{A}} = \sum_{k=0}^{+\infty} \frac{1}{k!} \mathbf{A}^k. \tag{1}$$

Neka je S_k k-ta parcijalna suma reda (1), a U_k njen opšti član. Tada važe jednakosti

$$U_k = \frac{1}{k} U_{k-1} \mathbf{A}, \quad S_k = S_{k-1} + U_k \quad (k = 1, 2, \ldots). \tag{2}$$

pri čemu je $\mathbf{U}_0 = \mathbf{S}_0 = \mathbf{I}$ (jedinična matrica reda n). Korišćenjem jednakosti (2) može se obrazovati program za sumiranje reda (1), pri čemu se, kao kriterijum za prekidanje procesa sumiranja, obično uzima slučaj kada je norma matrice manja od unapred zadatog malog pozitivnog broja ε. U našem slučaju uzećemo normu $\|.\|_2$ (videti formulu (3.2.3)) i $\varepsilon = 10^{-5}$.

Korišćenjem potprograma `MMAT` za množenje matrica (videti 4.4.2) i potpprograma `NORMA` za izračunavanje norme matrica (videti 4.4.8), sastavili smo sledeći program za nalaženje matrice $e^{\mathbf{A}}$

```
C================================================
C   ODREDJIVANJE MATRICE EXP(A)
C================================================
      DIMENSION A(100), S(100), U(100), P(100)
      OPEN(8,FILE='EXPA.IN')
      OPEN(5,FILE='EXPA.OUT')
      READ(8,10) N,EPS
   10 FORMAT(I2,E5.0)
      NN=N*N
      READ(8,15)(A(I),I=1,NN)
   15 FORMAT(16F5.0)
C  FORMIRANJE JEDINICNE MATRICE
      DO 20 I=1,NN
      S(I)=0.
   20 U(I)=0.
```

```
      N1=N+1
      DO 25 I=1,NN,N1
      S(I)=1.
 25   U(I)=1.
C SUMIRANJE MATRICNOG REDA
      K=0
 30   K=K+1
      CALL MMAT(U,A,P,N,N,N)
      B=1./K
      DO 35 I=1,NN
      U(I)=B*P(I)
 35   S(I)=S(I)+U(I)
C ISPITIVANJE USLOVA ZA PREKID SUMIRANJA
      CALL NORMA(U,N,2,ANOR)
      IF(ANOR.GT.EPS)GO TO 30
      WRITE(5,40) ((A(I),I=J,NN,N),J=1,N)
 40   FORMAT(2X,<5*N-9>X,'M A T R I C A      A'
     1 //(<N>F10.5))
      WRITE(5,45)((S(I),I=J,NN,N),J=1,N)
 45   FORMAT(//<5*n-9>X,'M A T R I C A   EXP(A)'
     1 //(<N>F10.5))
      CLOSE(8)
      CLOSE(5)
      END
```

Dobijeni program testirali smo na primeru

$$\mathbf{A} = \begin{bmatrix} 4 & 3 & -3 \\ 2 & 3 & -2 \\ 4 & 4 & -3 \end{bmatrix},$$

za koji se analitički može dobiti

$$\mathbf{A} = \begin{bmatrix} 3e-2 & 3e-3 & -3e+3 \\ 2e-2 & 2e-1 & -2e+2 \\ 4e-4 & 4e-4 & -4e+5 \end{bmatrix}. \tag{3}$$

Izlazna lista ima oblik

```
      M A T R I C A      A
   4.00000
```

```
 3.00000
-3.00000
 2.00000
 3.00000
-2.00000
 4.00000
 4.00000
-3.00000
    M A T R I C A   EXP(A)
 16.73060
 14.01232
-14.01232
  9.34155
 12.05983
 -9.34155
 18.68310
 18.68310
-15.96482
```

Korišćenjem (3) nije teško proveriti da su sve decimale u dobijenim elementima matrice tačne.