Ministry of education and science of Ukraine

Kyiv national university of technologies and design

Department of computer sciences

PROGRAMMING OF NUMERICAL METHODS AND EXAMPLES OF PRACTICAL APPLICATION

Shcherban' V. Yu.
Rezanova V.G.
Demkivska T.I.

PROGRAMMING OF NUMERICAL METHODS AND EXAMPLES OF PRACTICAL APPLICATION

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

KYIV NATIONAL UNIVERSITY OF TECHNOLOGIES AND DESIGN

Shcherban V.Yu., Rezanova V.G., Demkivska T.I.

PROGRAMMING OF NUMERICAL METHODS AND EXAMPLES OF PRACTICAL APPLICATION

Recommended by the Academic Council of the Kyiv National University of Technology and Design (Protocol № 5 of December, 15, 2021)

Kyiv-2021

УДК 004.42
ББК 65.9(4Укр)306.4-6
Щ 610

Recommended by the Academic Council of the Kyiv National University of Technology and Design for a wide range of researchers, teachers and engineers (Protocol № 5 of December, 15, 2021)

Authors:

SHCHERBAN' V.Yu. – Laureate of the State Prize of Ukraine in the field of science and technology, professor;
REZANOVA V.G. - Candidate of Technical Sciences, Associate Professor of the Department of Computer Science, Kyiv National University of Technology and Design ;
DEMKIVSKA T.I. - Candidate of Technical Sciences, Associate Professor of the Department of Computer Science, Kyiv National University of Technology and Design

Reviewers:

OPANASENKO V, M. - Laureate of the State Prize of Ukraine in the field of science and technology, Doctor of Technical Sciences, Professor, Leading Research Fellow of the Institute of Cybernetics of the National Academy of Sciences of Ukraine;
CHEPELYUK O. - Doctor of Technical Sciences, Professor, Head of the Department, Kherson National Technical University;
KRASNITSKY S.M. - Doctor of Technical Sciences, Professor, Kyiv National University of Technology and Design

Щ 610 Shcherban' V.Yu., Rezanova V.G., Demkivska T.I. Programming of numerical methods and examples of practical application. Monography. – K.: Education of Ukraine, 2021. – 150 c.

ISBN 978-617-8077-04-4

The monograph summarizes the experimental and theoretical developments of the authors and describes the developed mathematical models and software for research in the field of polymer composites with adjustable structure and properties. Particular attention is paid to the created software.

The monograph is intended for teachers, scientists, graduate students majoring in computer science and a wide range of engineers. The book can also be useful for senior students and graduate students of these specialties.

УДК 004.42
ББК 65.9(4Укр)306.4-6

ISBN 978-617-8077-04-4

CONTENT

FOREWORD 4

PART 1. Numerical methods and their programming

1. Algebra of matrices. Calculating of determinants. Actions with matrices. 6

2. Systems of linear equations (SLE), their solution by Kramer formulas, the method of inverse matrix, the method of Gauss 21

3. Solving of SLE by iterative methods. Method of simple iterations. Seidel method. Terms of convergence of iterative processes. 25

4. Transcendental equation with one variable. Separation of roots. Clarification of roots (methods dichotomy, chords, tangents, simple iterations) 31

5. Systemsof transcendental equations. The solution of two nonlinear equations by Newton method 37

6. Differential equations. Methods for solving differential equations. . Systems of differential equations 40

7. The characteristic determinant and characteristic equation of the matrix. The eigenvalues and eigenvectors of matrices 49

8. Interpolation problem with simple nodes. Vector interpolation problem with simple nodes 56

9. Bezier curves on the plane and in space 60

10. Linear and homogeneous coordinates on the plane 63

11. Basic conversion in the plane. The main symbol of affine transformations. 67

12. The compositions of affine transformations on the plane. 73

13. Curves of the second order: representation by matrix and invariants. Reduction of the second-order curve to the canonical form. Classification of second-order curves 75

14. Output of second-order curves on display. Method of cross-section . Iterative algorithms displaying the curves of the second order 84

PART 2. Practical application and Software

1. Mathematical modelling of dispersed phase drop deformation in nano-filled polyner mixture melts 92

2.Planning the experiment and optimization of the content of nanoadition in polypropylene monothreads 98

LITERATURE 105

ANNEX 109

FOREWORD

Analysis of the current state and prospects of the information technology industry shows that active research has state priorities in countries with the most developed economies. The implementation of their results changes the world development trends in the direction of significantly expanding the capabilities of a wide range of industries: chemistry, pharmaceuticals, pharmacology, construction, aviation, aeronautics and astronautics, energy, defense, transport and more.In studies of technical, technological, economic directions often have to build and analyze mathematical models of real phenomena and processes. Scientific problems of light industry are not an exceptions here.

In studies of technical, technological, economic directions often have to build and analyze mathematical models of real phenomena and processes. Scientific problems of light industry are not an exceptions here.

The purpose of mathematical modeling can be different. Often this purpose is the prediction (forecasting) behavior of certain characteristics of the objects. Types of mathematical models used are very different.

Of great importance are mathematical models in the form of differential equations, which are one of the main instruments of study a variety of phenomena and processes.

Linear algebraic equations does not necessarily serve as a means of approximating. In many situations, they provide a direct description of the phenomenon. These are, in particular, the situation are reduced to a certain number of relations "balance" type. Examples of this may be the problem of balancing economic sectors, resource allocation (of different nature), some electrical circuits etc.

In mathematical modeling of the phenomenon often have to deal nonlinear equations (algebraic or transcendental) or systems, and researchers need to have available methods for solving such relationship.

By the very specific problems of mathematical modeling in light industry refers selection of mathematical expressions to describe the various curves and surfaces. These curves can be, for example, outlines the real parts of articles of clothing or footwear, and surface - spatial fragments of such products.

A number of important industrial and economic problems (not just light industry) naturally united not so much the content as methods for their solution.

The goal of teaching monography is to study the application of mathematical methods for solving complex problems using modern computers.

PART 1. NUMERICAL METHODS AND THEIR PROGRAMMING

1. Algebra of matrices. Calculating of determinants. Actions with matrices

Key provisions

System of *m*n* numbers (real and complex), placed in a rectangular table with *m* lines and *n* columns

$$A = \begin{bmatrix} a_{11}a_{12}a_{13}...a_{1n} \\ a_{21}a_{22}a_{23}...a_{2n} \\ ..... \\ a_{m1}a_{m2}a_{m3}...a_{n} \end{bmatrix}, \qquad (1.1)$$

is called the matrix (numerical).

The numbers $a_{ij}\,(i=1,2,...,m;\ j=1,2,...,n)$, that make up this matrix, are called its elements. The first index i means line number of the element, and the second j — column number of it .

For matrix (1.1) is often used abbreviated representation

$A = \lfloor a_{ij} \rfloor \qquad (i=1,2,...,m;\ j=1,2,...,n)$ або $A = \lfloor a_{ij} \rfloor_{m,n}$,

and they say that the matrix *A* is of type $m \times n$.

If $m = n$, then matrix *A* is called square matrix of order n. If $m \neq n$, then matrix is called rectangular. In particular, the matrix of type $1 \times n$ is called vector-line and matrix of type $m \times 1$ — vector-column. Number (scalar) can be viewed as a matrix of type 1×1. Square matrix

$$A = \begin{bmatrix} a_1\ 0\ 0...0 \\ 0\ a_2\ 0...0 \\ ..... \\ 0\ 0\ 0...a_n \end{bmatrix} \qquad (1.2)$$

is called diagonal matrix.

If the $a_{ij} = 1\,(i = 1,2,...,n)$, then matrix (1.2) is called the identity matrix and is denoted by the letter *E,* i.e.

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

By entering Kronecker character

$$\delta_{ij} = \begin{cases} 0, & якщо \quad i \neq j; \\ 1, & якщо \quad i = j, \end{cases}$$

we can write: $E = \lfloor \delta_{ij} \rfloor$.

Matrix, all elements of which are zero, called zero-matrix and is denoted by $\mathbf{0}$. To mark number of rows and columns of zero-matrix, they use designation: 0_{mn}.

For the square matrix $A = \lfloor a_{ij} \rfloor_{n,n}$ there is the determinant

$$\det A = \begin{vmatrix} a_{11} a_{12} \ldots a_{1n} \\ a_{21} a_{22} \ldots a_{2n} \\ a_{n1} a_{n2} \ldots a_{nn} \end{vmatrix}.$$

We should not equate these two concepts: the matrix is an ordered system of numbers recorded in the form of a rectangular table, and its determinant is a number which can be defined by certain rules:

$$\det A = \sum_{(a_1, a_2 \ldots a_n)} (-1)^{x} a_{1a_1} a_{2a_2} \ldots a_{na_n} \qquad (1.3)$$

where the sum (1.3) includes all possible permutations $(\alpha_1, \alpha_{2,} \ldots \alpha_n)$ of elements $1,2,...,n$ and contains *n*! of summand, and $\chi = 0$, if an even permutation, and $\chi = 1$, if an odd permutation.

Actions with matrices

The equality of matrices

Two matrices $A=\lfloor a_{ij}\rfloor$ and $B=\lfloor b_{ij}\rfloor$ are considered as equal: $A=B$, if they are of the same type, i. e. They have the same number of rows and columns, and their respective elements are equal, i.e. $a_{ij}=b_{ij}$.

The sum of matrices

The sum of two matrices $A=\lfloor a_{ij}\rfloor$ and $B=\lfloor b_{ij}\rfloor$ of the same type is a matrix $C=\lfloor c_{ij}\rfloor$ of the same type, the elements of which c_{ij} are equal to the sums of corresponding elements a_{ij} and b_{ij} of those matrices A and B, i.e. $c_{ij}=a_{ij}+b_{ij}$. So,

$$A+B=\begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \cdots & a_{1n}+b_{1n} \\ a_{21}+b_{21} & a_{22}+b_{22} & \cdots & a_{2n}+b_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1}+b_{m1} & a_{m2}+b_{m2} & \cdots & a_{mn}+b_{mn} \end{bmatrix}$$

From the determination of the sum of two matrices immediately follows its properties:

1) $A+(B+C)=(A+B)+C$;

2) $A+B=B+A$;

3) $A+0=A$.

Similarly the difference of matrices is determinated :

$$A+B=\begin{bmatrix} a_{11}-b_{11} & a_{12}-b_{12} & \cdots & a_{1n}-b_{1n} \\ a_{21}-b_{21} & a_{22}-b_{22} & \cdots & a_{2n}-b_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1}-b_{m1} & a_{m2}-b_{m2} & \cdots & a_{mn}-b_{mn} \end{bmatrix}.$$

Multiply matrix by the number

The product of matrix $A=\lfloor a_{ij}\rfloor$ by the number α (or the product of the number by matrix) is matrix, the elements of which are obtained by multiplying all elements of the matrix A by that number α, so

$$A\alpha = \alpha A = \begin{bmatrix} \alpha a_{11} & \alpha a_{12} & \cdots & \alpha a_{1n} \\ \alpha a_{21} & \alpha a_{22} & \cdots & \alpha a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ \alpha a_{m1} & \alpha a_{m2} & \cdots & \alpha a_{mn} \end{bmatrix}.$$

From the determination of the product of the matrix by the number immediately follows its properties:

1) $1A = A$;

2) $0A = 0$;

3) $\alpha(\beta A) = (\alpha\beta)A$;

4) $(\alpha + \beta)A = \alpha A + \beta A$;

5) $\alpha(A + B) = \alpha A + \alpha B$

(here A and B – are matrices; α and β – are numbers).

Note, that if the matrix A - is a square order n, then

$\det \alpha A = \alpha^n \det A$.

Matrix $-A = (-1)A$ is called opposite. Not difficult to see that if the matrix A and B are of the same types, then $A - B = A + (-B)$.

Multiply matrices

Let

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \quad \text{i} \quad B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{bmatrix}$$

-matrices of types $m \times n$ and $p \times q$ correspondingly. If the number of columns of the matrix A equals the number of rows of the matrix B, i.e.

$$n = p,$$

then for these matrices is defined matrix C of type $m \times q$, called their product:

$$C = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1q} \\ c_{21} & c_{22} & \cdots & c_{2q} \\ \cdots & \cdots & \cdots & \cdots \\ c_{m1} & c_{m2} & \cdots & c_{mq} \end{bmatrix},$$

where $c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in}b_{nj} \, (i = 1,2,\cdots,m; \, j = 1,2,\cdots,q)$.

From the determination of the product of two matrices immediately follows the rule to multiply matrices: to receive an element which is in *i*-th line and *j*-th column of the product of two matrices, it is necessary to multiply elements of *i*-th row of the first matrix by the respective elements of *j*-th column of the second matrix and then to add obtained products.

The product *AB* has sense if and only if the matrix *A* has so many rows , how many columns has matrix *B*. In particular, it is possible to multiply square matrices only of the same order.

In cases when *AB*=*BA*, matrices *A* and *B* are called rearrangement (commutative). For example, it is easy to see that identity matrix *E* rearrangement with any square matrix *A* of the same order, and

$AE = EA = A$

Thus, the identity matrix *E* plays a role of "*one*" in multiplication.

If *A* and *B* – are square matrices of the same order, then

$\det(AB) = \det(BA) = \det A \cdot \det B$.

For example, for such matrices we have:

$$\begin{vmatrix} 19 & 22 \\ 43 & 50 \end{vmatrix} = \begin{vmatrix} 1 & 2 \\ 3 & 4 \end{vmatrix} \cdot \begin{vmatrix} 5 & 6 \\ 7 & 8 \end{vmatrix}$$

and

$$\begin{vmatrix} 23 & 34 \\ 31 & 46 \end{vmatrix} = \begin{vmatrix} 1 & 2 \\ 3 & 4 \end{vmatrix} \cdot \begin{vmatrix} 5 & 6 \\ 7 & 8 \end{vmatrix}.$$

Transposed matrix

If we change in matrix

$$A=\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

of the type $m\times n$ the rows with corresponding colomns, we obtain so called *transposed matrix*:

$$A'=A^T=\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix},$$

of the type $m\times n$. In particular, for vector-line $a=\begin{bmatrix} a_1 & a_2 & \dots & a_n \end{bmatrix}$ the transposed matrix is vector-column

$$a'=\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}.$$

Transposed matrix has such properties:

1) the twice transposed matrix is the original one:

$$A''=(A')'=A;$$

2) the transposed matrix of sum is equal to sum of transposed matrices:

$$(A+B)'=A'+B';$$

3) The transposed matrix of the product is equal to product of transposed matrices:

$$(AB)'=B'A';$$

Really, the element of i-th row and j-th colomn of matrix $(AB)'$ is equal to the element of j-th row and i-th colomn of matrix AB, i.e.:

$$a_{j1}b_{1i}+a_{j2}b_{2i}+\dots+a_{jn}b_{ni}.$$

The last expression is obviously the sum of the products of elements of *i*-th line of matrix B' and respective elements of *j*-th column of the matrix A', that is equal to the general element of matrix $B'A'$.

If matrix A – is square, then obviously

$$\det A' = \det A.$$

Matrix $A = [a_{ij}]$ is called symmetric, if it matches with its transposed, i.e. if:

$$A' = A. \qquad (1.4)$$

From equation (1.4) follows that: 1) symmetric matrix – is square $(m = n)$ and 2) its elements, which are symmetric relatively main diagonal, are equal to each other, i.e.

$$a_{ji} = a_{ij}.$$

The product $C = AA',$ is obviously a symmetric matrix, so how

$$C' = (AA')' = (A')'A' = AA' = C.$$

The inverse matrix

Definition 1. *Inverse* matrix in relation to this matrix is a matrix, which is being multiplied right and left side with this matrix gives the identity matrix.

For matrix A let's denote A^{-1} - inverse matrix. Then according to the definitioin we have:

$$AA^{-1} = A^{-1}A = E, \qquad (1.5)$$

where E – identity matrix.

Finding the inverse matrix to this is called inversion of the matrix.

A square matrix is called *nonsingular* if its determinant is different from zero.

Otherwise matrix called special or singular.

Every nonsingular matrix has an inverse matrix.

Let's we have nonsingular matrix of n-th order

$$A=\begin{bmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ ... & ... & ... & ... \\ a_{n1} & a_{n2} & ... & a_{nn} \end{bmatrix},$$

where $\det A=\Delta\neq 0$.

Let's construct for it so-called adjoint matrix

$$\tilde{A}=\begin{bmatrix} A_{11} & A_{21} & ... & A_{n1} \\ A_{12} & A_{22} & ... & A_{n2} \\ ... & ... & ... & ... \\ A_{1n} & A_{2n} & ... & A_{nn} \end{bmatrix},$$

where A_{ij} – algebraic additions (minors with signs) of the respective elements $a_{IJ}\,(i,j=1,2,...,n)$.

Note that the algebraic additions of elements of rows are plaved in corresponding columns, so it is an operation of transposition.

Let's divide all elements of the last matrix on the value of determinant of the matrix A, i.e. on Δ:

$$A^{*}=\begin{bmatrix} \frac{A_{11}}{\Delta} & \frac{A_{21}}{\Delta} & ... & \frac{A_{n1}}{\Delta} \\ \frac{A_{12}}{\Delta} & \frac{A_{22}}{\Delta} & ... & \frac{A_{n2}}{\Delta} \\ ... & ... & ... & \\ \frac{A_{1n}}{\Delta} & \frac{A_{2n}}{\Delta} & ... & \frac{A_{nn}}{\Delta} \end{bmatrix}.$$

Notes 1. For a given matrix A its inverse matrix A^{-1} is only.

Notes 2. Special square matrix has not the inverse.

Some basic properties of the inverse matrix:

The determinant of inverse matrix is equal to the reciprocal of the determinant of the original matrix.

Indeed, let

$$A^{-1}A=E.$$

Given that the determinant of the product of two square matrices is the product of determinants of matrices, we get:

$\det A^{-1} \det A = \det E = 1.$

So,

$$\det A^{-1} = \frac{1}{\det A}.$$

The inverse matrix of product of square matrices is the product of the inverse matrices of multipliers, taken in reverse order, i.e.

$(AB)^{-1} = B^{-1}A^{-1}$

Indeed,

$AB(B^{-1}A^{-1}) = A(BB^{-1})A^{-1} = AEA^{-1} = AA^{-1} = E$

and

$(B^{-1}A^{-1})AB = B^{-1}(A^{-1}A)B = B^{-1}EB = B^{-1}B = E$

So, $B^{-1}A^{-1}$ is inverse matrix to AB.

In more general

$(A_1A_2...A_p)^{-1} = A_p^{-1}A_{p-1}^{-1}...A_1^{-1}.$

The transpose inverse matrix is equal to the inverse transpose matrix:

$(A^{-1})' = (A')^{-1}.$

Indeed, if transposed the main matrix equality $A^{-1}A = E$, we get: $(A^{-1}A)' = A'(A^{-1})' = E' = E.$

Hence, multiplying last equality on the left on matrix $(A')^{-1}$, will have: $(A')^{-1}A'(A^{-1})' = (A')^{-1}E$

or

$(A^{-1})' = (A')^{-1}$,

as was required to proof.

Note. The matrix equations are easily solved With the help of inverse matrices.

Equations $AX = B$ and $YA = B$.

Indeed, if $\det A \neq 0$, then $X = A^{-1}B$ and $Y = BA^{-1}$.

Degree matrix

Let A - square matrix. If p - integer, the considered

$\underbrace{AA...A}_{p-раз} = A^p$.

Additionally set, that $A^0 = E$, where E - is identity matrix. If A is nonsingular matrix, you can introduce the concept of a negative degree, defining it by relation:

$A^{-p} = \left(A^{-1}\right)^p$

For degrees of the matrices with integer exponent are valid ordinary rules:

1) $A^p A^q = A^{p+q}$;

2) $\left(A^p\right)^q = A^{pq}$.

Non-square matrix, as is known, not to be present degree.

Norm of the matrix

Inequality $A \leq B$ between matrices $A = \lfloor a_{ij} \rfloor$ and $B = \lfloor b_{ij} \rfloor$ of the same types means, that

$$a_{ij} \leq b_{ij}$$

Absolute value (modulus) of the matrix $A = \lfloor a_{ij} \rfloor$ we will understand matrix

$$[A] = \lfloor |a_{ij}| \rfloor$$

where $\lfloor a_{ij} \rfloor$ - are the modulus of elements of matrix A.

If A and B – are matrices, for which the operations $A+B$ and AB have sense, then:

$a) |A + B| \leq |A| + |B|$;

$b) |AB| \leq |A| \cdot |B|$;

$в) |\alpha A| = |\alpha||A|$

(α – is number)

In particular, we have

$$\left|A^p\right| \le \left|A\right|^p$$

(p – integer) .

The norm of the matrix $A = \left|a_{ij}\right|$ means the real number $\left\|A\right\|$, which have such properties:

$$а)\left\|A\right\| \ge 0, \left\|A\right\| = 0 \quad iff \ A = 0$$

$$б)\left\|\alpha A\right\| = \left|\alpha\right|\left\|A\right| \quad (\alpha - number), \_ in \_ particular, \left\|- A\right\| = \left\|A\right\|$$

$$в)\left\|A + B\right\| \le \left\|A\right\| + \left\|B\right\|$$

$$г)\left\|AB\right\| \le \left\|A\right\| \cdot \left\|B\right\|$$

(A and B – are matrices, for which the appropriate operations have sense). In particular, for square matrix we have:

$$\left\|A^p\right\| \le \left\|A\right\|^p,$$

where p – integer.

Let's note another important inequality between the norms of matrices A and B of the same type. Using condition c), we have:

$$\left\|B\right\| = \left\|A - (B - A)\right\| \le \left\|A\right\| + \left\|B - A\right\|$$

From here

$$\left\|A - B\right\| = \left\|B - A\right\| \ge \left\|B\right\| - \left\|A\right\|.$$

Similarly,

$$\left\|A - B\right\| \ge \left\|A\right\| - \left\|B\right\|.$$

So, $\left\|A - B\right\| \ge \left\|\left\|B\right\| - \left\|A\right\|\right\|.$

Thereafter, for the matrix $A = \left\lfloor a_{ij}\right\rfloor$ of arbitrary type we will consider three main easyly calculated norms:

$$1)\|A\|_m = \max_j \sum_j |a_{ij}| \quad (m-norm);$$

$$2)\|A\|_l = \max_j \sum_i |a_{ij}| \quad (l-norm);$$

$$3)\|A\|_k = \sqrt{\sum_{i,j} |a_{ij}|^2} \quad (k-norm).$$

Rank of the matrix

We have a rectangular matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots\dots & \dots\dots & & \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

If in this matrix randomly select k rows and k columns, where $k \leq \min(m, n)$, the elements that are at the intersection of this rows and columns, are forming a square matrix of order k. The determinant of this matrix is called the minor of k-th order matrix A.

Definition. The maximum order of minor of matrix , different from zero, is called the *rank* of matrix.

In other words, the matrix A has rank r, if:

1) There is at least one minor of order r that is different from zero;

2) all the minors of matrix A of order $r + 1$ and higher are equal to zero.

Rank of the zero matrix, i.e. matrix consisting of zeros, is zero. The difference between the smallest of the numbers m and n and rank is called the defect of the matrix.

Elementary transformation matrices

The following transformation matrices are called elementary:

1) permutation of two rows or columns;

2) multiplication of all the elements of any row (column) on the same number different from zero;

3) adding to the elements of a row (column) the elements of the other row (column) multiplied by the same number.

Two matrices are called equivalent, if one can be obtained from another through a finite number of elementary transformations. These matrices are not, in general, equal, but have the same rank.

Easy to ensure, that each elementary transformation of a square matrix A is equivalent to multiplication last for some nonsingular matrix. However, if the conversion is done on lines (columns) matrix A, the multiplier should be left (right) and represent the result of the related elementary transformation to the identity matrix.

For example, moving the matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

second and third lines, we obtain the equivalent matrix:

$$\tilde{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{31} & a_{32} & a_{33} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}.$$

The same matrix $\tilde{A}$ can be obtained, if in identity matrix

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

rearranged second and third lines

$$\tilde{E} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

and the resulting matrix multiply by the left side on the matrix A, i.e. $\tilde{A} = \tilde{E}A$.

Similar way are performed other elementary transformation.

Note that if in the equation $AA^{-1} = E$ we perform the same transformation of rows of matrices A and E as long as the A is not converted into a identity

matrix, we will have $\tilde{E}AA^{-1}=\tilde{E}$, where $\tilde{E}$ - – transformed the identity matrix. Hence, i.e. $\tilde{E}A=E$, we have $A^{-1}=\tilde{E}$, i.e. the inverse matrix A^{-1} is the converted identity matrix. There is the method of calculation of inverse matrix is based on this idea of converting lines.

Calculation of determinants

Elementary transformation matrices provide the most convenient method of calculating the determinant of this matrix. Suppose, for example,

$$\Delta_n=\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}.$$

Assuming that $a_{11}\neq 0$, we have:

$$\Delta_n=a_{11}\begin{bmatrix} 1 & a_{12} & \dots & a_{1n} \\ \frac{a_{21}}{a_{11}} & a_{22}^{(1)} & \dots & a_{2n}^{(1)} \\ \dots & \dots & \dots & \dots \\ \frac{a_{n1}}{a_{11}} & a_{n2} & \dots & a_{nn} \end{bmatrix}$$

Hence, subtracting from the elements a_{ij}, which belong to the j-th column (j>=2) the relevant elements of the first column multiplied by a_{1j}, we get:

$$\Delta_n=a_{11}\begin{bmatrix} 1 & 0 & \dots & 0 \\ \frac{a_{21}}{a_{11}} & a_{22}^{(1)} & \dots & a_{2n}^{(1)} \\ \dots & \dots & \dots & \dots \\ \frac{a_{n1}}{a_{11}} & a_{n2}^{(1)} & \dots & a_{nn}^{(1)} \end{bmatrix}=a_{11}\Delta_{n-1}$$

where

$$\Delta_{n-1}=\begin{bmatrix} a_{22}^{(1)} & a_{23}^{(1)} & \dots & a_{2n}^{(1)} \\ a_{32}^{(1)} & a_{33}^{(1)} & \dots & a_{3n}^{(1)} \\ \dots & \dots & \dots & \dots \\ a_{n2}^{(1)} & a_{n3}^{(1)} & \dots & a_{nn}^{(1)} \end{bmatrix}$$

and $a_{ij}^{(1)} = a_{ij} - \frac{a_{i1}a_{1j}}{a_{11}} (i, j = 2,3..., n).$

Apply to the determinant Δ_{n-1} the same way. If all the elements

$$a_{ii}^{(i-1)} \neq 0 (i = 1,2,..., n),$$

then finally obtain:

$$\Delta_n = a_{11}a_{22}^{(1)}...a_{nn}^{(n-1)}$$

If in any determinant Δ_{n-k} the upper left element $a_{k+1,k+1}^{(k)} = 0$, we have to rearrange the rows or columns of the determinant Δ_{n-k}, that needed element was different from zero (it is always possible if the determinant $\Delta \neq 0$). Of course, you will need to consider changing the sign of the determinant Δ_{n-k}.

It's possible to give more general rule. Let determinant $\tilde{\Delta}_n = \det[a_{ij}]$ is changed so, that $\alpha_{pq} = 1$ (α_{pq} – the main element), i.e.

$$\tilde{\Delta}_n = \begin{vmatrix} a_{11} & ... & a_{1q} & ... & a_{1j} & ... & a_{1n} \\ ..... & & & & & & \\ a_{i1} & ... & \overline{|a_{iq}|} & ... & a_{ij} & ... & a_{in} \\ ..... & & & & & & \\ a_{p1} & ... & 1 & ... & \overline{|a_{pj}|} & ... & a_{pn} \\ ..... & & & & & & \\ a_{n1} & ... & a_{nq} & ... & a_{nj} & ... & a_{nn} \end{vmatrix}$$

Then $\tilde{\Delta}_n = (-1)^{p+q} \cdot \tilde{\Delta}_{n-1}$,

where $\tilde{\Delta}_{n-1} = \det[\alpha_{ij}^{(1)}]$ – is the determinant of the *(n - 1)*-th order, which we obtain from Δ_n by deleting *p*-th row and *q*-th colomn, followed by conversion elements by the formula:

$$\alpha_{ij}^{(1)} = \alpha_{ij} - \alpha_{iq}\alpha_{pj}$$

i.e., each element $\alpha_{ij}^{(1)}$ of the determinant $\tilde{\Delta}_{n-1}$ equal to the corresponding element α_{ij} of determinant of the matrix $\tilde{\Delta}_n$, reduced by the product of its "projections" α_{iq} and α_{pj}

on the erased column and row of the original determinant. Confirmation of this statement easily follows from the general properties of determinants.

Software that implements the described algorithm has been developed [26, 29, 32]. The text of the main program procedures is given in the appendix 1.

2. Systems of linear equations (SLE), their solution by Kramer formulas, the method of inverse matrix, Gauss method

General concepts and definitions

When conducting research of mechanical systems often have to face the necessity of solving systems of linear equations.

In general, the system of linear equations can be represented as follows:

$$\left.\begin{array}{c} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n = b_2, \\ .... \\ a_{m1}x_1 + a_{m2}x_2 + ... + a_{mn}x_n = b_m \end{array}\right\} \quad (2.1)$$

where $a_{ij}\ (i = 1,2,...,m;\ j = 1,2,...,n)$ — coefficients of the unknowns;

$x_i\ (i = 1,2,...,n)$ — unknowns;

$b_i\ (i = 1,2,...,m)$ — constants;

n — the number of unknowns in the system;

m — number of equations.

The system of equations (2.1) conveniently presented in vector form

$$Ax = b \tag{2.2}$$

where A — matrix of матриця coefficients a_{ij} $(i = 1, 2, ..., m;\ j = 1, 2, ..., n)$;

$x = (x_1, \ldots x_n)$ — sought-for n - component vector;

$b = (b_1, \ldots, b_m)$ — given m - component vector (vector right parts).

Tasks, that are reduced to solving systems of linear equations, generally are mostly statically defined (number of unknowns equals the number of possible equations, i.e. $m = n$). In this case, a system of linear equations (2.1) can be represented as follows (coefficient matrix A is square):

$$\left.\begin{array}{c} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n = b_2, \\ ..... \\ a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n = b_n \end{array}\right\} \tag{2.3}$$

For the square matrix $A = \lfloor a_{ij} \rfloor_{n,n}$ there is the determinant det A.

Determinant is a number which can be defined by certain rules:

$$\det A = \sum_{(a_1, a_2 ... a_n)} (-1)^{x} a_{1a_1} a_{2a_2} ... a_{na_n}$$

where the sum (3) includes all possible permutations $(\alpha_1, \alpha_{2,} ... \alpha_n)$ of elements $1, 2, ..., n$ and contains $n!$ of summand, and $\chi = 0$, if an even permutation, and $\chi = 1$, if an odd permutation.

The determinant of the coefficient matrix A can be denoted by one of the following methods:

$$\Delta_n = \det A = \begin{vmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ ... & ... & ... & ... \\ a_{n1} & a_{n2} & ... & a_{nn} \end{vmatrix} \tag{2.4}$$

If $\det A \neq 0$ the matrix A is called nondegenerate. If the system of equations (2.3) has a solution, it is called compatible. Otherwise it will be called imcompatible or contradictory.

If the right parts vector b is **0**, the system (2.3) is called homogeneous. Homogeneous system of equations always compatible. It has non-zero solutions when $\det A = 0$.

If the system (2.3) has the only solution they say that the system of equations is defined. If there are two or more solutions of the system it is called uncertain.

The case when the determinant $\det A \neq 0$ provides for the only solution.

Methods for solving systems of linear equations

Formulas of Kramer

The exact solution of system (2.3) in explicit form can be obtained using formulas Kramer. The method consists in sequential dividing of the transformed determinant (in which the coefficients of the corresponding column of the system are replaced by column $b_1, b_2, ..b_n$) by the initial determinant composed of elements of left side of equations (2.3). Thus, the vector of solutions of system $x = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{bmatrix}$ can be defined as follows:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{bmatrix} = \frac{1}{\Delta} \begin{bmatrix} \Delta_1 \\ \Delta_2 \\ . \\ . \\ . \\ \Delta_n \end{bmatrix}, \qquad (2.5)$$

where $\Delta_i = \sum_{i=1}^{n} A_{ji} b_j = \begin{vmatrix} a_{11}...a_{1,i-1} b_1 a_{1,i+1}...a_{1n} \\ a_{21}...a_{2,i-1} b_2 a_{2,i+1}...a_{2n} \\ ..... \\ a_{n1}...a_{n,i-1} b_n a_{n,i+1}...a_{nn} \end{vmatrix}$ — are the determinants, obtained from the determinant Δ by replacing of its *i*-th column by column of right parts of the system (2.3).

From equality (2.5) we obtain the formulas of Kramer

$$x_1 = \frac{\Delta_1}{\Delta}, x_2 = \frac{\Delta_2}{\Delta}, ..., x_n = \frac{\Delta_n}{\Delta} \qquad (2.6)$$

Software that implements the described algorithm has been developed [26, 29, 32]. The text of the main program procedures is given in the appendix 2.

Gauss method

The method of successive elimination of variables (Gauss method) is based on the consistent lowering order of the system (2.3) by eliminating the unknowns $x_1, x_2, ...x_{n-1}$ in linear equations This results in obtaining of a triangular coefficients matrix $a_{ij} (i = 1, 2, ..., m; j = 1, 2, ..., n)$. This sequence of procedures is the direct way exceptions variables. For its implementation must consistently subtract from the first equation the other equations, multyplying their left and right parts on the constant ratio, which is a share of the division of constant coefficients $m_{ij} = a_{ji} / a_{ii}$, where $i = 1, 2, ..., n-1; j = i+1, i+2, ..., n$. The final system of equations on a direct way looks:

$$\left.\begin{array}{r} a_{11}x_1 + a_{12}x_2 + ... + a_{1k}x_k + ... + a_{1n}x_n = b_1 \\ a_{22}^{(1)}x_2 + ... + a_{2k}^{(1)}x_k + ... + a_{2n}^{(1)}x_n = b_2^{(1)} \\ ..... \\ a_{kk}^{(k-1)}x_1 + ... + a_{kn}^{(k-1)}x_n = b_n^{(k-1)} \\ ..... \\ a_{nn}^{(n-1)}x_n = b_n^{(n-1)} \end{array}\right\} \qquad (2.7)$$

where $a_{ij}^{(1)} = a_{ij} - a_{i1}b_{1j} \quad (i, j \geq 2)$, wherein $b_{1j} = \frac{a_{1j}}{a_{11}} \quad (j > 2)$;

$a_{ij}^{(2)} = a_{ij}^{(1)} - a_{i2}^{(1)} b_{2j}^{(1)}$ $(i, j \geq 3)$, wherein $b_{3j}^{(2)} = \frac{a_{3j}^{(2)}}{a_{33}^{(2)}}$ $(j > 3)$, i.e.

From the system (2.7) the solutions are obtained trivially.

Thus, the process of solving a linear system (2.3) by the method of Gauss reduces to the construction of an equivalent system (2.7), which has a triangular matrix. A necessary and sufficient condition for the applicability of the method is that all the "leading elements" are not equal to zero. The process of finding of the coefficients of the triangular system are usually called direct way, the process of obtaining values of the unknowns - the back way.

Software that implements the described algorithm has been developed [26, 29, 32]. The text of the main program procedures is given in the appendix 3.

3. Solving of SLE by iterative methods. Method of simple iterations. Seidel method. Terms of convergence of iterative processes

Method of simple iterations

With a large number of unknowns in linear system the Gauss method scheme becomes very difficult. In these conditions, to find the roots of the system is sometimes convenient to use approximate numerical methods. One of these methods - the method of iteration.

Let's we have a linear system

$$\left.\begin{array}{l} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_{n.} \end{array}\right\} \qquad (3.1)$$

By entering into consideration matrix

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

system (3.1) can be briefly written as the matrix equation

$$Ax = b. \tag{3.1'}$$

Considering that the diagonal coefficients

$$a_{ij} \neq 0 \quad (i = 1,\ 2,\ \dots,\ n)$$

let's solve the first equation of system (3.1) relative x_1, the second - a relatively x_2 etc. Then get the equivalent system:

$$\left.\begin{aligned} x_1 &= \beta_1 + \alpha_{12}x_2 + \alpha_{13}x_3 + \dots + \alpha_{1n}x_n, \\ x_2 &= \beta_2 + \alpha_{21}x_1 + \alpha_{23}x_3 + \dots + \alpha_{2n}x_n, \\ &\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ x_n &= \beta_n + \alpha_{n1}x_1 + \alpha_{n2}x_2 + \dots + \alpha_{n,n-1}x_{n-1}, \end{aligned}\right\} \tag{3.2}$$

where $\beta_i = \dfrac{b_i}{a_{ii}}$; $\alpha_{ij} = -\dfrac{a_{ij}}{a_{ii}}$; wherein $i \neq j$

and $\alpha_{ij} = 0$ wherein $i = j$ $(i, j = 1,\ 2,\ \dots,\ n)$.

After entering matrix

$$\alpha=\begin{bmatrix} \alpha_{11} & \alpha_{12} & \dots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \dots & \alpha_{2n} \\ \dots\dots\dots\dots\dots\dots\dots \\ \alpha_{n1} & \alpha_{n2} & \dots & \alpha_{nn} \end{bmatrix} \text{ i } \beta=\begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix},$$

system (3.2) we can write in matrix form:

$$x = \beta + \alpha x. \tag{3.2'}$$

The system (3.2) will be solved by *successive approximations*. As an initial approximation let's take, for example, the right parts column $x^{(0)} = \beta$.

Then, gradually construct the vector- columns

$x^{(1)} = \beta + \alpha x^{(0)}$ (*first approximation)*

$x^{(2)} = \beta + \alpha x^{(1)}$ (*second approximation*) etc.

Generally, any $(k+1)-th$ approximation is calculated by the formula:

$$x^{(k+1)} = \beta + \alpha x^{(k)} \qquad (k = 0,\ 1,\ 2,\ \ldots). \tag{3.3}$$

If the sequence of approximations $x^{(0)},\ x^{(1)},\ \ldots,\ x^{(k)},\ \ldots$ has limit

$$x = \lim_{k\to\infty} x^{(k)},$$

then the limit is a solution of system (3.2). Indeed, passing to the limit in equality (3.3), we have:

$$x = \lim_{k\to\infty} x^{(k+1)} = \beta + \alpha \lim_{k\to\infty} x^{(k)},$$

or

$$x = \beta + \alpha x,$$

i.e. the limit vector x is the solution of system (3.2'), and thus the system (3.1).

Let's write the approximation formula in expanded form:

$$\left.\begin{aligned} & x_i^{(0)} = \beta_i, \\ & x_i^{(k+1)} = \beta_i + \sum_{j=1}^{n} \alpha_{ij} x_j^{(k)} \\ & \left(\alpha_{ij} = 0;\quad i = 1,\quad \ldots,\quad n;\ k = 0,\quad 1,\quad 2,\quad \ldots\right). \end{aligned}\right\} \tag{3.3'}$$

Note that sometimes more convenient to convert system (3.1) to the form (3.2) so that coefficients α_{ij} were not zero.

Overall, with the system

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \qquad (i = 1,2,\ldots, n)$$

you can put:

$$a_{ij} = a_{ij}^{(1)} + a_{ij}^{(2)},$$

where $a_{ij}^{(1)} \neq 0$. Then the system is equivalent to the reduced system

$$x_i = \beta_i + \sum_{j=1}^{n} \alpha_{ij} x_j \quad (i = 1,2,\ldots, n),$$

where

$$\beta_i = \frac{b_i}{a_{ij}^{(1)}}\ , \qquad \alpha_{ij} = -\frac{\alpha_{ii}^{(2)}}{\alpha_{ii}^{(1)}}\ , \quad \alpha_{ij} = -\frac{\alpha_{ij}}{\alpha_{ij}^{(1)}} \quad \text{wherein} \quad i \neq j$$

Therefore, in further considerations we will not, in general, assume that $\alpha_{ij} = 0$.

The method of successive approximations determined by formula (3.3) or (3.3'), is called *method of iterations*. The iteration process (3.3) converges good, ie the number of approximations necessary to obtain roots of system (3.1) with the required accuracy, is little if elements of matrix α are small in absolute value. In other words, the successful process of iterations will be if modules of the diagonal coefficients of system (3.1) are large compared with modules non-diagonal coefficients of the system (free members play no role).

Remark. In applying the method of iterations it's no need for initial approximation to accept a column of right parts. The convergence of the iteration process depends on the properties of the matrix, and if this process convergences with any choice of initial approximation of the home, it will be the same to the same vector and with any other selection of initial approximation. Therefore, in the initial vector iteration can be taken arbitrarily.

Software that implements the described algorithm has been developed [26, 29, 32]. The text of the main program procedures is given in the appendix 4.

Seidel method

Seidel method is a modification of the method of iterations. Its main idea is that in the calculation of *$(\kappa+1)$-th* approximation of unknown x_i are considered previously calculated *$(\kappa+1)$-th* approximation of unknown values $x_1, x_2, \ldots, x_{i-1}$.

Let's we have reduced linear system

$$x_i = \beta_i + \sum_{j=1}^{n} \alpha_{ij} x_j \quad (i = 1,2,..., n).$$

Let's choose an arbitrary initial approximations of roots

$$x_1^{(0)},\ x_2^{(0)},\ \ldots,\ x_n^{(0)},$$

Onward, assuming that the κ-*th* approximations of roots $x_i^{(k)}$ are known, according to Seidel we will build *(*κ*+1)-th* approximation of roots by the following formulas:

$$x_1^{(k+1)} = \beta_1 + \sum_{j=1}^{n} \alpha_{1j} x_j^{(k)};$$

$$x_2^{(k+1)} = \beta_2 + \alpha_{21} x_1^{(k+1)} + \sum_{j=2}^{n} \alpha_{2j} x_j^{(k)};$$

……

$$x_i^{(k+1)} = \beta_i + \sum_{j=1}^{i-1} \alpha_{ij} x_j^{k+1} + \sum_{j=i}^{n} \alpha_{ij} x_j^{(k)};$$

……

$$x_i^{(k+1)} = \beta_n + \sum_{j=1}^{n-1} \alpha_{nj} x_j^{k+1} + \alpha_{nn} x_n^{(k)} \quad (k = 0,1,2,\ldots).$$

Usually Seidel method gives better convergence than simple iteration method, but generally speaking, it leads to more cumbersome calculations. Seidel process may be convergence, even if the iteration process diverges.

Software that implements the described algorithm has been developed [26, 29, 32]. The text of the main program procedures is given in the appendix 5.

Terms of convergence of iterative processes

Let's we have reduced linear system:

$$x = \alpha x + \beta \qquad (3.4)$$

where $\alpha = [\alpha_{ij}]$, $\beta = \begin{bmatrix} \beta_1 \\ . \\ . \\ . \\ \beta_n \end{bmatrix}$ - given matrix and vector and $x = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{bmatrix}$ – unknown vector.

Theorem. The process of iteration for the reduced linear system (3.4) converges to its the only solution, if any matrix α norm less than unity, i.e. for the iteration process

$$x^{(k)} = \beta + \alpha x^{(k-1)} \qquad (k = 1,2,\ldots)$$

($x^{(0)}$ - arbitrary) is a sufficient condition for convergence

$$\|\alpha\| < 1. \tag{3.5}$$

Let $x^{(k-1)}$ та $x^{(k)} (k \geq 1)$ – two successive approximation solution of linear system $x = \alpha x + \beta$. At $p \geq 1$ we have:

$$\left\|x^{(k+p)} - x^{(k)}\right\| \leq \left\|x^{(k+1)} - x^{(k)}\right\| + \left\|x^{(k+2)} - x^{(k+1)}\right\| + \ldots + \left\|x^{(k+p)} - x^{(k+p-1)}\right\|. \tag{3.6}$$

Since $x^{(m+1)} = \alpha x^{(m)} + \beta$ and $x^{(m)} = \alpha x^{(m-1)} + \beta$, then

$x^{(m+1)} - x^{(m)} = \alpha(x^{(m)} - x^{(m-1)})$ and hence:

$$\left\|x^{(m+1)} - x^{(m)}\right\| \leq \|\alpha\|\left\|x^{(m)} - x^{(m-1)}\right\| \leq \|\alpha\|^{m-k}\left\|x^{(k+1)} x^{(k)}\right\|.$$

Because of the formula (3.6) we obtain:

$$\left\|x^{(p+k)} - x^{(k)}\right\| \leq \left\|x^{(k+1)} - x^{(k)}\right\| + \ldots + \|\alpha\|^{p-1}\left\|x^{(k+1)} - x^{(k)}\right\| \leq \frac{1}{1-\|\alpha\|}\left\|x^{(k+1)} - x^{(k)}\right\|$$

Passing in the last inequality to the limit wherein p→∞, we obtain:

$$\left\|x - x^{(k)}\right\| \leq \frac{\left\|x^{(k+1)} - x^{(k)}\right\|}{1 - \|\alpha\|} \tag{3.7}$$

at $k \geq 1$, or

$$\left\|x - x^{(k)}\right\| \leq \frac{\|\alpha\|}{1-\|\alpha\|}\left\|x^{(k)} - x^{(k-1)}\right\|.$$

If in the process of calculation found that

$$\left\|x^{(k)} - x^{(k-1)}\right\| \le \frac{1-q}{q}\varepsilon,$$

where $q = \|\alpha\| < 1$, то $\left\|x - x^{(k)}\right\| \le \varepsilon$, and, so $\left|x_i - x_i^{(k)}\right| \le \varepsilon$

$(\mathrm{i} = 1, 2, \ldots, \mathrm{n})$.

It is assumed that the successive approximation $x^{(j)}\,(j = 0,1,\ldots, k)$ calculated accurately, that there are completely absent rounding error.

From formula (3.7), using obtained estimates for norm of difference of two successive approximations, we have:

$$\left\|x - x^{(k)}\right\| \le \frac{\|\alpha\|^k}{1-\|\alpha\|}\left\|x^{(1)} - x^{(0)}\right\|.$$

In particular, if you choose $x^{(0)} = \beta$, то $x^{(1)} = \alpha\beta + \beta$ and

$\left\|x^{(1)} - x^{(0)}\right\| = \|\alpha\beta\| \le \|\alpha\|\|\beta\|.$

So, $\left\|x - x^{(k)}\right\| \le \frac{\|\alpha\|^{k+1}}{1-\|\alpha\|}\|\beta\|.$

4. Transcendental equation with one variable. Separation of roots. Clarification of roots (methods dichotomy, chords, tangents, simple iterations)

Introductory provisions

Solving nonlinear equations of the form *f (x) = 0* often can be done in the next two stages. In the first stage of a rough definition of the root. Of course this can be done graphical manner. The second stage means the root clarify . It is often useful following famous theorem on the existence of a root of continuous function.

Theorem. If the function *f (x)* is defined and continuous on the interval [*a, b*], and on the ends of the segment takes values of different signs (so $f(a)f(b) < 0$), then in the interval (*a, b*) there is at least one root of the

equation $f(x) = 0$. In other words, in these conditions, there is a point c, $a < c < b$, such that the equality $f(p) = 0$ justifies.

Solution of nonlinear equations

Separation of the roots

From the geometric point of view the real root of the equation

$$f(x) = 0 \tag{4.1}$$

is the abscissa of the point of intersection graph of $y = f(x)$ with the axis Ox

This note is used for graphic separation of roots of the equation (4.1) when this equation has not closely roots, and the graph of $y = f(x)$ constructed accurately.

In practice, it is often convenient to replace the equation (4.1) with equivalent equation

$$\varphi(x) = \psi(x) \tag{4.2}$$

where functions $\varphi(x)$ and $\psi(x)$ — are more easy than function $f(x)$. Then, construct graphics $y = \varphi(x)$ and $y = \psi(x)$, desired roots get as abscissa of the point of intersection of these graphs.

Clarification of roots

Method dichotomy

Consider method division-on-half - method dichotomy. The method consists in the construction of iterative sequence of nested segments, the ends of which are the monotonous sequence $\{a_n\}$, $\{b_n\}$, and $a_n \le \xi$, $b_n \ge \xi$, $n = 1,2...$, where ξ - the root of the transcendental equation (4.1) on the segment $[a,b]$.

The convergence of this method is slow. However, in any interval the convergence is guaranteed.

We assume that $f(a)<0$, $f(b)>0$. Then we find the middle of segment $[a,b]$ - point $\xi_1=\frac{a+b}{2}$. Calculate the function $f(x)$ in this point. Choose one of the obtained segments where the condition $f(a)f(\xi_1)<0$ or $f(\xi_1)f(b)<0$ justifies. The selected segment divide in half again by taking $a_1=a,\quad b_1=\xi_1$ or $a_1=\xi_1,\quad b_1=b$, and then $\xi_2=\frac{a_1+b_1}{2}$.

Continuing of iterative process of division allows you to obtain a sequence of nested segments, and $a_n\le a_{n+1}<b_{n+1}\le b_n$. Left ends of segments form a monotonous sequence which in the limit represents the value z_1:

$$\lim_{n\to\infty}a_n=z_1,$$

and the right ends of segments form a monotonous sequence which in the limit represents the value z_2:

$$\lim_{n\to\infty}b_n=z_2.$$

Obviously,

$$a_n\le z_1\le z_2\le b_n,$$
$$z_2-z_1\le b_n-a_n=\frac{b-a}{2^n}. \qquad (4.4)$$

This error does not exceed the length of the segment b_n-a_n and goes to zero by increasing n according to the law geometric progression with denominator 1/2.

Software that implements the described algorithm has been developed [26, 29, 32]. The text of the main program procedures is given in the appendix 6, 7.

Newton's method (tangent)

Let's define the root of the transcendental equation (2.1) using Newton's method (or the method of tangents). In figure 4.1 is shown a graphic scheme that implements the method of Newton.

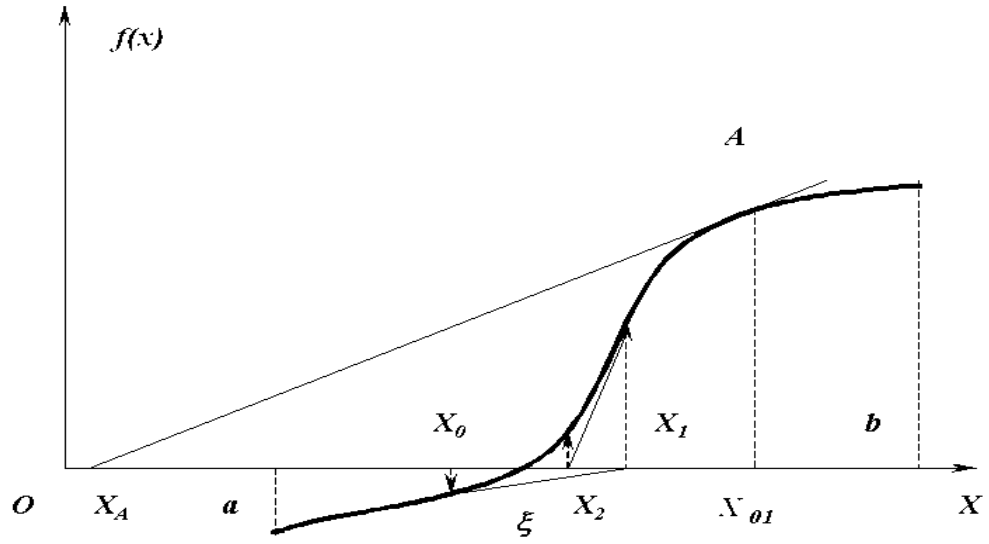

Fig. 4.1. Geometric interpretation of Newton's method of solving the transcendental equation

With the graphical method we determine the initial approximation of root of the transcendent equation $f(x)=0$. The equation of the tangent to the graph of $f(x)$ in the point x_0 looks like

$$f_K = f(x_O) + f'(x_O)(x - x_O)\ ,$$

(4.5)

where $f'(x_0)$ - value of derivative of the function $f(x)$ at the point x_0.

At $x = x_1$, $f_K = 0$. Then, from (4.5) we receive

$$x_1 = x_O - \frac{f(x_O)}{f'(x_O)}. \qquad (4.6)$$

Continuing the process of constructing iterative sequence $\{x_n\}$, obtain the following recurrent formula for the implementation of the iterative process of approximation to the root of the transcendental equation

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)},\ \ n = 0,1,2... \qquad (4.7)$$

Newton's method, implemented by (4.6), has a high rate of convergence, but it is very sensitive to the choice of initial approximation x_0. Figure 4.5 shows that the choice as the initial approximation x_{01} (point A on $f(x)$, which is located

further from the desired root ξ, than x_0) leads at the first step to the "loss" the point x_A from limits of the interval $[a,b]$.

If $|f'(x)| \geq m > 0,\ |f''(x)| \leq M,\ x \in [a,b]$ (m - the smallest value of derivative $f'(x)$ в $[a,b]$; M - the largest value of derivative in $[a,b]$), then there such $\delta:\ 0 < \delta \leq \min(\xi - a, b - \xi)$, that for any choice of initial approximation on the interval $[\xi - \delta, \xi + \xi] \in [a,b]$ there is an endless iterative sequence (4.6) and this sequence convergences to the root of the transcendental equation $f(x) = 0$.

To evaluate the error of the n - *th* approximation x_n you can use the formula

$$|\xi - x_n| \leq \frac{|f(x_n)|}{m_1} \tag{4.8}$$

where m_1 - the smallest value of the first derivative module $|f'(x)|$ in the interval $[a,b]$.

Iterations method

One of the most important methods of numerical solution of transcendental equations is the method of iterations (or - the method of successive approximations or method of simple iterations).

Transcendental equation (2.1)

$f(x) = 0$

present to the form

$$x = f_1(x) \tag{4.9}$$

where $f_1(x) = f(x) + x$.

Using graphical method, define the approximate value of the root x_0 from the area of the function definition $f_1(x)$ and substitute it in the right side of equation (4.9). Let's build sequence $\{x_n\}$ of numbers, determined using iterative formula

$$x_{n+1} = f_1(x_n), \quad n = 0,1,2,...$$

Sequence $\{x_n\}$ of numbers $\{x_n\}$ is called iterative sequence. If there $\lim_{n\to\infty} x_n = \xi$, then passing in equity (4.9) to the limit and assuming function $f_1(x)$ is continuous, we obtain:

$$\lim_{n\to\infty} x_{n+1} = f_1\left(\lim_{n\to\infty} x_n\right), \text{ або} \qquad \xi = f_1(\xi).$$

From the last equality comes, that ξ will be the root of the transcendental equation (4.9), and therefore, and equation (4.1). Iterative process continues until justifies the condition

$$|x_{n+1} - x_n| \geq \varepsilon, \tag{4.10}$$

where ε - задана похибка обчислення кореня ξ .

Before the proof of the convergence of iterative sequence dwell on Lipschitz condition, which is as follows. Function $f(x)$ satisfies Lipschitz if there exists a constant $q<1$, that any x_1, x_2, owned segment $[a,b]$, performed inequality

$$|f(x_1) - f(x_2)| \leq q|x_1 - x_2|.$$

If the function of (4.1) satisfies (4.10), it is continuous on the interval $[a,b]$. Give argument x the increment Δx. Using the Lipschitz condition, we get the confirmation of continuity of functions $f(x)$ on the interval $[a,b]$.

$$|\Delta f| \leq \alpha|\Delta x|,$$
$$\lim_{\Delta x\to 0} \Delta f = 0.$$

Theorem on the convergence of iterative sequence can be formulated as follows. Suppose that the function $f_1(x)$ s defined and differentiated in the interval $[a,b]$, and all of its values are in $[a,b]$. Then using Lipschitz conditions $\left|\frac{df_1(x)}{dx}\right| \leq q < 1$ (at $a < x < b$) we obtain, that the iteration process $x_{n+1} = f_1(x_n)$ convergences regardless of the initial value $x_O \in [a,b]$ and the limit value $\xi = \lim_{n\to\infty} x_n$ is the only root of equation $x = f_1(x)$ on the interval $[a,b]$.

Let's prove this statement. Take initial approximation solution of the transcendental equation (4.1) x_0 on the interval $[\xi-\delta,\xi+\delta]$, which is remote from the point ξ at a distance of no more than δ ($|\xi - x_O| \le \delta$).Perform iterative process using Lipschitz conditions and taking into account (4.9)

$$\left.\begin{array}{c} x_1 = f_1(x_O),\ \ x_1 - \xi = f_1(x_O) - f_1(\xi), \\ |x_1 - \xi| = |f_1(x_O) - f_1(\xi)| \le q|x_O - \xi| \le q\delta, \\ |x_2 - \xi| = |f_1(x_2) - f_1(\xi)| \le q|x_1 - \xi| \le q^2|x_O - \xi| \le q^2\delta, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ |x_n - \xi| \le q^n|x_O - \xi| \le q^n\delta. \end{array}\right\} \qquad (4.11)$$

The theorem remains valid if the function $f_1(x)$ is defined and differentiated in the interval $]-\infty,+\infty[$. To assess the approximations let's use inequality

$$|\xi - x_{n+1}| \le \frac{q}{1-q}|x_{n+1} - x_n|.$$

If $q < \frac{1}{2}$ then come to (4.10). Finally

$$\xi = x_{n+1} \pm \varepsilon.$$

Software that implements the described algorithm has been developed [26, 29, 32]. The text of the main program procedures is given in the appendix 8.

5. Systems of transcendental equations. The solution of two nonlinear equations by Newton method

Systems of nonlinear equations

Consider a system of nonlinear equations

$$\left.\begin{array}{c} f_1(x_1, x_2, \ldots, x_n) = 0 \\ f_2(x_1, x_2, \ldots, x_n) = 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ f_n(x_1, x_2, \ldots, x_n) = 0 \end{array}\right\} \qquad (5.1)$$

with real left parts.

Write shorter system (5.1). The set of arguments $x_1, x_2, ..., x_n$ can be seen as n-dimensional column-vector x. Similarly, a set of functions $f_1, f_2, ..., f_n$ is also n-dimensional column-vector (vector function) f.

To solve the system (5.1') we will use the following method of successive approximations. Suppose that k-*th* approximation was found

$$x^{(k)} = (x_1^{(k)}, x_2^{(k)}, ..., x_n^{(k)})$$

$$x = x^{(k)} + \varepsilon^{(k)} \qquad (5.2)$$

where $\varepsilon^{(k)} = (\varepsilon_1^{(k)}, \varepsilon_2^{(k)}, ..., \varepsilon_n^{(k)})$ - amendment (error) of the root.

Substituting expression (5.2) in equation (5.1 '), we have:

$$f(x^{(k)} + \varepsilon^{(k)}) = 0.$$

Assuming that the function $f(x)$ continuously differentiated in some area, which contains x and $x^{(k)}$, we decompose the left side of the last equation in powers of the small vector $\varepsilon^{(k)}$, leaving only the linear terms of the series:

$$f(x^{(k)} + \varepsilon^{(k)}) = f(x^{(k)}) + f'(x^{(k)})\varepsilon^{(k)} = 0 \qquad (5.3)$$

From equalities (5.3) follows that if denote $W(x)$ the Jacobi matrix of the derivatives of system of functions $f_1, f_2, ..., f_n$ relatively to variables $x_1, x_2, ..., x_n$, ie

$$f'(x) = W(x) = \left[\frac{\partial f_i}{\partial x_j} \right], \qquad i, j = 1,2,...,n,$$

the system (2.14 ') will be the linear system regarding modifications $\varepsilon_i^{(k)}$ $i = 1,2,...,n$ with the matrix $W(x)$, and therefore formula (5.3) can be written as (5.3').

$$f(x^{(k)}) + W(x^{(k)})\varepsilon^{(k)} = 0.$$

Hence, assuming that the matrix $W(x^{(k)})$ is nonsingular, we get:

$$\varepsilon^{(k)} = -W^{-1}(x^{(k)}) f(x^{(k)}).$$

So,

$$x^{(k+1)} = x^{(k)} - W^{-1}(x^{(k)})\, f(x^{(k)}) \qquad p = 1,2,... \tag{5.4}$$

(5.4) – *the Newton method.*

For the initial approximation $x^{(0)}$ we can take rough approximation of the desired root.

The Newton method for a system of two equations

Let x_n, y_n - approximated roots of the system of equations

$$\left.\begin{aligned} F(x,y) &= 0 \\ G(x,y) &= 0 \end{aligned}\right\} \tag{5.5}$$

where F and G - continuously differentiated functions. Suppose

$$x = x_n + h_n; \qquad y = y_n + k_n,$$

we have:

$$\left.\begin{aligned} F(x_n,y_n) + h_n F'_x(x_n,y_n) + k_n F'_y(x_n,y_n) &= 0 \\ G(x_n,y_n) + h_n G'_x(x_n,y_n) + k_n G'_y(x_n,y_n) &= 0 \end{aligned}\right\} \tag{5.6}$$

If Jacobian

$$J(x_n,y_n) = \begin{vmatrix} F'_x(x_n,y_n) & F'_y(x_n,y_n) \\ G'_x(x_n,y_n) & G'_y(x_n,y_n) \end{vmatrix} \neq 0,$$

then from system (5.6) we obtain

$$h_n = -\frac{1}{J(x_n,y_n)} \begin{vmatrix} F(x_n,y_n) & F'_y(x_n,y_n) \\ G(x_n,y_n) & G'_y(x_n,y_n) \end{vmatrix}, \tag{5.7}$$

$$k_n = -\frac{1}{J(x_n,y_n)} \begin{vmatrix} F'_x(x_n,y_n) & F(x_n,y_n) \\ G'_x(x_n,y_n) & G(x_n,y_n) \end{vmatrix}. \tag{5.8}$$

So we can put:

$$x = x_n - \frac{1}{J(x_n,y_n)} \begin{vmatrix} F(x_n,y_n) & F'_y(x_n,y_n) \\ G(x_n,y_n) & G'_y(x_n,y_n) \end{vmatrix} \tag{5.9}$$

$$y = y_n - \frac{1}{J(x_n,y_n)} \begin{vmatrix} F'_x(x_n,y_n) & F(x_n,y_n) \\ G'_x(x_n,y_n) & G(x_n,y_n) \end{vmatrix} \tag{5.9'}$$

$(n = 0,1,2,...)$.

The condition for stopping the iterative process will be the following:

$\delta = \max(|h_n|, \quad |k_n|) < \varepsilon$,

where ε - the given precision of solving the problem.

The initial values of roots are determined roughly approximated.

Software that implements the described algorithm has been developed [26, 29, 32]. The text of the main program procedures is given in the appendix 9.

6. Differential equations. Methods for solving differential equations. Systems of differential equations

Basic concepts

Differential equations - are such equations which containing derivatives of the unknown function of one or more independent variables.

Equations containing derivatives by several independent variables, are called differential equations with *partial derivatives*.

Equations containing derivatives of several independent variables, called partial differential equations.

General view of the differential equation of n-th order is following:

$$F\left(x, y, y', y'', ..., y^{(n)}\right) = 0. \qquad (6.1)$$

This is an implicit form of differential equation. Explicitly form of the equation of n-th order will be the equation which is solved relatively older derivate:

$$y^{(n)} = f\left(x, y, y', y'', ..., y^{(n-1)}\right). \qquad (6.2)$$

Let the variable x takes values in the interval $I \subset R = (-\infty, \infty)$. *The solution of the differential equation* on the interval I is called such a differentiated in I function $y = \varphi(x)$, after setting to the equation it rotates in equality for all $x \in I$ (identity on the set I). The chart of the solution of differential equations called

the *integral curve*. *The general solution* of equation usually contains one free numeric parameter and has the form

$$y = \varphi(x, C) \tag{6.3}$$

where C — said parameter, φ — any function. Equality (6.3) determines the family of functions, whichdepend on the parameter C. Allocation of single solution from a family of solutions (6.3) can be satisfied if the known initial value $y(x_0) = y_0$ for some $x_0 \in I$.

The general solution of equation (6.1) or (6.2) is a family of functions of form:

$$y = \varphi(x, C_1,\ldots,C_n), \tag{6.4}$$

where $C_1,\ldots,C_n$ — numeric parameters that are called arbitrary constant, and each function of the family is a solution of equation (at some numerical interval). Parameters $C_1,\ldots,C_n$ can be determined by the initial conditions of the form $y(x_0) = y_{10},\ldots,y^{(n-1)}(x_0) = y_{n0}$.

There are situations where solutions of differential equations in explicit form (6.3), (6.4) can not get, but can be found the so-called general integrals, or general solutions of these equations. Thus the general integral differential of the equation (6.1) or (6.2) is the equation which is not an identity

$$\psi(x, y, C_1, C_2, ..., C_n) = 0, \tag{6.3a}$$

The function ψ is also called a general integral equation.

Existence and uniqueness of solutions of differential equations of the first and n-th order

We further consider equations solved relatively senior derivative. Consider the equation of the first order

$$y' = f(x, y).$$

Let function $f(x, y)$ is defined in some open area D of the plane XOY (Fig. 6.1), the interval I is a subset of D projection on the set *R*. Let in D is the point M with coordinates (x_0, y_0) ($x_0 \in I$).

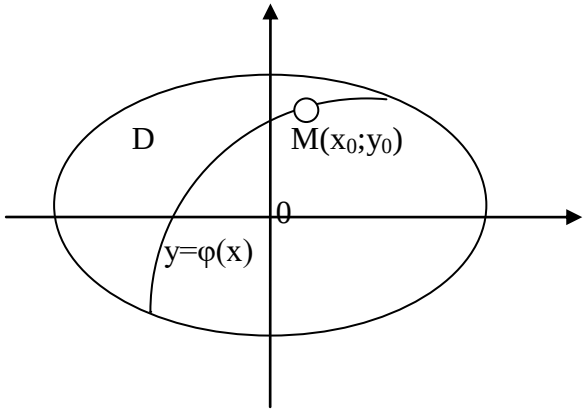

Fig. 4.1. Geometric interpretation of solutions of differential equations

The problem: to find in the interval *I* the solution of the equation, integral curve of which passes through the point *M*, i.e. to find a function $y = \varphi(x)$, $x \in I$, satisfying the initial condition

$$y\big|_{x=x_0} = \varphi(x_0) = y_0. \qquad (6.5)$$

This problem is called the *Cauchy problem.* The following theorem formulated conditions of existence and uniqueness of "local" solution to this problem.

1 Existence and uniqueness. If the function $f(x, y)$ is defined and continuous in D with its partial derivate $\frac{\partial f}{\partial y}$, then for any point $M(x_0, y_0)$, owned area D, exists the interval *I*, containing a point x_0 and in which is defined and the unique solution $y = \varphi(x)$ of the equation, which satisfies the initial condition (6.5).

Under the uniqueness of the solution is to understand the following: if there are two solutions of the equation which are the same at the point x_0, then the solutions coincide on the common part of interval of their definition.

Geometrically in theorem states that under the conditions of the theorem through each point inside the area D is the only integrated curve.

Now formulate theorem which is the analog for previous for the equation of *n-th* order solved relatively senior derivative (of the form (6.2)) with initial conditions:

$$y\Big|_{x=x_0} = y_0; y'\Big|_{x=x_0} = y'_0, ..., y^{(n-1)}\Big|_{x=x_0} = y_0^{(n-1)} \quad . \qquad (6.6)$$

2 Existence and uniqueness. If the function $f(x, y, z_1, z_2, ..., z_{n-1})$, which depends upon $n+1$ variables x, y, z_1,…, z_{n-1}, defined and continuous in some *(n+1)*-measurable area D together with its derivates $\frac{\partial f}{\partial y}, \frac{\partial f}{\partial z_1}, \frac{\partial f}{\partial z_2}; ..., \frac{\partial f}{\partial z_{n-1}}$, then for any point $M(x_0, y_0, y'_0, y''_0, ..., y_0^{(n-1)})$ that belongs to the area $M(x_0, y_0, y'_0, y''_0, ..., y_0^{(n-1)})$ D, exists the interval I, containing a point x_0 and in which is defined and the only solution $y = \varphi(x)$ of the equation (6.2), that satisfies the initial condition (6.6).

The concept of uniqueness of the solution in this theorem is the same as in the previous.

From 1, 2, it follows that on their conditions if the presence of general solution of equation (6.2) in the form (6.4):

$y = \varphi(x, C_1, …, C_n)$

constants $C_1, C_2, ..., C_n$ are defined uniquely by initial conditions (6.6) for an arbitrary vector $(x_0, y_0, y_0', …, y_0^{(n-1)}) \in D$, i.e. the system of equations

$$\begin{aligned} &\varphi(x_0, C_1, C_2, ..., C_n) = y_0; \\ &\varphi'(x_0, C_1, C_2, ..., C_n) = y'_0; \\ &.......... .... \\ &\varphi^{(n-1)}(x_0, C_1, C_2, ..., C_n) = y_0^{(n-1)} \end{aligned} \qquad (6.7)$$

has a unique solution. Conversely, if for an arbitrary vector $(x_0, y_0, y_0', …, y_0^{(n-1)})$ system (6.7) has a unique solution, then the Cauchy problem (6.2), (6.6) has a unique solution for any point $M \in D$.

The conditions imposed in theorems 1.2 on the right sides of equations (6.2) sufficient for the existence and uniqueness of solutions to the equation. For the existence of local solutions (such referred to Theorem 2) is sufficient to require continuity of f in the area D.

Methods for solving differential equations

Problem solving ordinary differential equations in the general case is more complicated than the problem dealing with calculation of single integrals, and therefore the fate of cases of explicitly integration here is much lower.

Numerical methods for solving differential equations can be divided into two classes. One of them includes methods that use one starting value of solution at every step, and the other methods use multiple values at every turn (multistep methods). The last are characterized that on the basis of earlier got a few values of function are built the new which are then specified with differential equations.

The first class include Runge - Kutta methods, including methods of Euler - Cauchy and trapezoids. The second include, for example, the method of Adams, Adams-Krylov method.

Consider first the Euler-Cauchy method.

Let is given the differential equation

$$\frac{dy}{dx} = f(x, y), \tag{6.8}$$

where (x, y) belongs to area G with the initial condition

$$x = x_0, y_0 = y(x_0) \tag{6.8′}$$

Method of constructing an approximate solution of the Cauchy problem (6.8), (6.8′) is based on the concept of so-called *Euler polyline*. Euler polyline is a graph of piecewise linear function that is built based on the following rule. Let h — small positive number (step of method). Consider a Cartesian plane point with coordinates (x_1, y_1), where

$x_1 = x_0 + h, y_1 = y_0 + hf(x_0, y_0)$.

Note that, according to Taylor's formula, thanks to equality (6.8), (6.8′) y_1 - value can be seen as approaching of the values of the solution $y(x_1)$ of Cauchy problem. If the point (x_1, y_1) belongs to set G, then we continue to build on inductive rule $y_{i+1} = y_i + hf(x_i, y_i)$, $i = 0, 1, 2, \ldots$. Each value y_i is seen as approximation to the value of the desired solution y at the point x_i. So we get a sequence of points (x_i, y_i) , $i = 0,1,2, \ldots$, where all x_i are situated on right of the point x_0. A similar construction, if necessary, carry out and on left of point x_0. According to this sequence we build piecewise-linear function

$$y(x) = y_i + f(x_i, y_i)(x - x_i),\ x \in [x_i, x_{i+1}],\ i = 0, 1, 2, \ldots ,$$

which (or its chart) is called the Euler polyline. There are several theories that guarantee that under certain conditions the Euler polyline aims to the solution of the Cauchy problem (3.10), (3.10′), when the method step h aims to 0.

Graphical representation of the calculation scheme of the method Euler - Cauchy shown in Fig. 6.2.

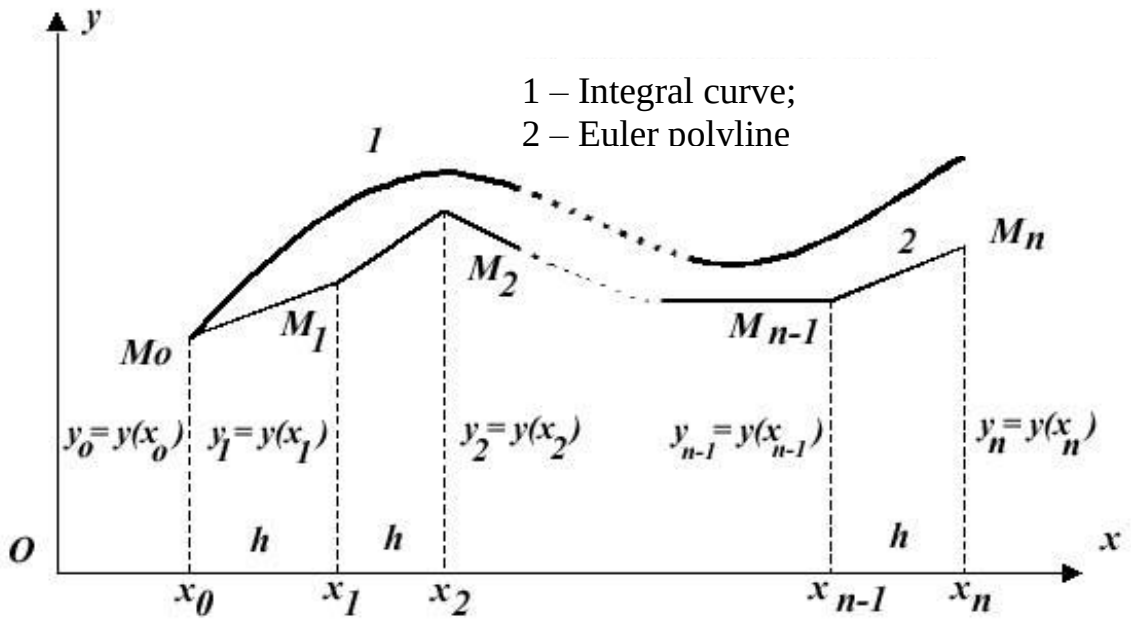

Рис. 6.2. Calculation scheme of the Euler - Cauchy method

Let's given the differential equation

$$\frac{dy}{dx} = f(x, y). \qquad (6.8)$$

Need to find an approximate solution (6.8) at the points with coordinates $x_1 = x_O + h,\ x_2 = x_O + 2h, ... x_n = x_O + nh$, where h - constant pitch; x_O - coordinate of start point of interval.

The initial condition $x = x_O,\ y_O = y(x_O)$. The approximate value of the first derivative has the form

$$\frac{dy_k}{dx_k} \cong \frac{\Delta y_k}{\Delta x_k} = \frac{y_{k+1} - y_k}{h}, \tag{6.9}$$

where $k = 0,1...n-1$.

Equating (6.8) and (6.9), we obtain:

$$\frac{y_{k+1} - y_k}{h} = f(x_k, y_k),$$

from whence:

$$y_{k+1} = y_k + hf(x_k, y_k). \tag{6.10}$$

Using the recurrence formula (6.10) for points $k = 0,1...n-1$ we build the Euler polyline 2, which replaces approximately the integral curve 1 (see. fig.6.2). The gist of Euler-Cauchy method is that in the beginning of each interval $[x_k, x_{k+1}]$ we held tangent to the integral curve 1.

The accuracy of the method Euler-Cauchy is small. The error of method is proportional to h^2.

A variation of the method of Euler-Cauchy is the trapezoidal method. It is implemented at each step using recurrent formula

$$y_{k+1} = y_k + \frac{h}{2}\left\{ f(x_k, y_k) + f\left[x_k + h\ , y_k + hf(x_k, y_k) \right] \right\}. \tag{6.11}$$

The error of the trapezoidal method is proportional to h^3 and it also includes the general methods of Runge-Kutta.

Multi-step solving of differential equations (finite-difference methods) are based on the using of the rezults of solving of the previous steps. This can increase the speed of computing. For the realization of the finite-difference

methods for the numerical integration of differential equations need to know the function and its derivatives at several points close to the original. Here we can mark Picard method and the method of decomposition.

Systems of differential equations

The aggregate of mutual relations

$$\begin{cases} F_1(x, y_1, ..., y_1, y_1', ..., y_n') = 0 \\ F_2(x, y_1, ..., y_1, y_1', ..., y_n') = 0 \\ \quad ..... \\ F_n(x, y_1, ..., y_1, y_1', ..., y_n') = 0 \end{cases} \tag{6.12}$$

where x- independent variable, $y_1, y_2, ..., y_n$ - unknown functions of x, $F_1, F_2, ..., F_n$ – known function, called a *system of first order differential equations*. The *solution* of this system are functions $y_1(x), y_2(x), ..., y_n(x)$, which when substituted in (6.12) turn the system on identity.

If the system of differential equations (6.12) admits the possibility of solving relatively derivatives, we get a system

$$\begin{cases} \dfrac{dy_1}{dx} = f_1(x, y_1, y_2, ..., y_n) \\ \dfrac{dy_2}{dx} = f_2(x, y_1, y_2, ..., y_n) \\ \quad ..... \\ \dfrac{dy_n}{dx} = f_n(x, y_1, y_2, ..., y_n) \end{cases} \tag{6.13}$$

which is called normal.

An example of one normal equation of first order is

$$\frac{dy}{dx} = f(x, y).$$

This equation gives the field of directions in the plane x, y. The solution of the equation is the one-parameter family of curves, located in one plane. If on this

plane is given point (x_0, y_0) and functions $f(x,y)$, $\frac{\partial f}{\partial y}$ – continious, then the equation has a unique solution that satisfies the initial conditions $y(x_0) = y_0$.

Now, take two equations

$$\begin{cases} \frac{dy_1}{dx} = f_1(x, y_1, y_2) \\ \frac{dy_2}{dx} = f_2(x, y_1, y_2) \end{cases} \quad \text{or} \quad \begin{cases} \frac{dy}{dx} = f_1(x, y, z) \\ \frac{dz}{dx} = f_2(x, y, z) \end{cases}.$$

Under certain conditions, we get the solution

$$y_1 = y = \varphi_1(x); \quad y_2 = z = \varphi_2(x).$$

This solutions can be regarded as parametric equations of the curve in the spatial coordinate system x, y, z.

Thus, the solution of one equation can be represented as the curve in two-dimensional space. Solution of two equations of the first order can be visualized by the curve in three dimensions. Solution of n equations of the first order forms a curve in the $(n+1)$-dimensional space. These curves are called integral curves.

The numerical solution of systems of differential equations is carried out similarly solving a differential equation.

Software that implements the described algorithm has been developed [26, 29, 32]. The text of the main program procedures is given in the appendix 10.

7. The characteristic determinant and characteristic equation of the matrix. The eigenvalues and eigenvectors of matrices

The characteristic determinant

and characteristic equation of the matrix

Let's we have a square matrix $A = \lfloor A_{ij} \rfloor$. Consider a linear transformation

$$y = Ax, \tag{7.1}$$

where ***x*** and ***y*** — *n*-dimensional vectors of some *n*-dimensional space.

Definitions. Nonzero vector called *eigenvector* of the matrix, if in the result of the corresponding linear transformation this vector becomes the collinear to it, i.e. the converted vector is different from the original only by scalar multiplier.

In other words, a vector $x \neq 0$ is called eigenvector matrix *A*, if the matrix transforms the vector ***x*** in the vector λx :

$$Ax = \lambda x \tag{7.2}$$

The number λ of equality (7.2) is called *eigenvalues* or *characteristic number* of the matrix *A*, appropriate the eigenvector ***x***.

$$(A - \lambda E)x = 0, \tag{7.3}$$

Where matrix $A - \lambda E$ called characteristic matrix. Equation (7.3) is a linear homogeneous system that has nonzero solution if and only if the determinant of the system is zero, ie when the condition

$$\det(A - \lambda E) = 0. \tag{7.4}$$

The determinant (7.4) is called the *characteristic determinant* of matrix *A*, and the equation (7.4) is called the *characteristic equation* of *A*. In expanded form the characteristic equation (7.4) can be written as follows:

$$\begin{vmatrix} a_{11} - \lambda & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} - \lambda & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} - \lambda \end{vmatrix} = 0 \tag{7.4'}$$

or

$$\lambda^n - \sigma_1\lambda^{n-1} + \sigma_1\lambda^{n-2} - ... + (-1)^{n-1}\sigma_{n-1}\lambda + (-1)^n\sigma_n = 0 \qquad (7.5)$$

Polynom, which is standing in the left side of the equation (7.5) is called the *characteristic polynom* of matrix *A*. Its coefficients $\sigma_i (i = 1,2,...,n)$ are determined by the following rules. Coefficient σ_1 is equal to the sum of the diagonal elements of the matrix *A*, i.e. $\sigma_1 = \sum_{i=1}^{n} a_{ii}$. This number is called the *track* of matrix *A* and denoted: $\sigma_1 = SpA$. Coefficient σ_2 is the sum of all the diagonal minors of the second order of the matrix *A*. Generally, coefficient σ_k is the sum of all the diagonal minors of the k*-th* order of the matrix A. Finally, free term σ_n is equal to the determinant of the matrix A: $\sigma_n = \det A$.

Characteristic equation (7.5) is an algebraic equation of n*-th* degree relative λ and has at least one real or complex root. Let $\lambda_1, \lambda_2, \ldots, \lambda_m (m \le n)$ - different roots of equation (7.5). These roots are called *eigenvalues* or *characteristic numbers* of the matrix, and the set of all eigenvalues are called the *spectrum* of matrix A. Let's take any root $\lambda = \lambda_j$ and substitute it in equation (7.4). Then we have $(A - \lambda_j E)x = 0$ or, in an expanded form,

$$\left.\begin{array}{l} (a_{11} - \lambda_j)x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = 0, \\ a_{21}x_1 + (a_{22} - \lambda_j)x_2 + \ldots + a_{2n}x_n = 0, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + (a_{nn} - \lambda_j)x_n = 0. \end{array}\right\} \qquad (7.6)$$

Since the determinant of the system (7.6) $\det(A - \lambda_j E) = 0$, then this system has non-zero solutions which are the eigenvectors of matrix *A* corresponding to its eigenvalues λ_j. If the rank of the matrix $A - \lambda_j$ - is equal $r (r < n)$, then exist $k = n - r$ linearly independent eigenvectors $x^{(1j)}, x^{(2j)}, \ldots, x^{(kj)}$, corresponding to λ_j

.

Remark. We can prove that the number of linearly independent eigenvectors does not exceed the multiplicity of this root. It follows that if the

roots of the characteristic equation (7.5) are different, each eigenvalues corresponds to within a proportionality factor one and only one eigenvector.

Finding eigenvalues and eigenvectors matrix

Introductory remarks

In solving theoretical and practical problems it is often necessary to determine the eigenvalues of matrix, which means calculate the roots of its characteristic equation $\det(A-\lambda E)=0$ and find the corresponding eigenvectors of matrix *A*. The second problem is more easier: if the roots of the characteristic equation are known, then the calculation of eigenvectors reduced to finding of some nonzero solutions of homogeneous linear systems. Therefore, we will first deal with the first problem - calculation the roots of characteristic equation.

Here it is mainly used two methods:

1) deployment of characteristic determinant to the polynom of *n-th* degree: $D(\lambda)=\det(A-\lambda E)$ with following solving of the equation $D(\lambda)=0$ with one of the known approximated methods and

2) an approximate determination of the roots of the characteristic equation without prior deployment of characteristic determinant.

Deploying characteristic determinants

May have characteristic determinant of matrix $A=\lfloor a_{ij}\rfloor$ as following:

$$D(\lambda)=\det(A-\lambda E)=\begin{vmatrix} a_{11}-\lambda & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22-\lambda} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn-\lambda} \end{vmatrix}$$

Equating this determinant to zero, we obtain the characteristic equation

$D(\lambda)=0$.

If you want to find all the roots of the characteristic equation, it is advisable to pre-disclose determinant.

Deploying the determinant, get the polynom of n-th degree:

$$D(\lambda) = (-1)^n (\lambda^n - \sigma_1 \lambda^{n-1} + \sigma_2 \lambda^{n-2} - ... + (-1)^n \sigma_n),$$

where: $\sigma_1 = \sum_{\alpha=1}^{n} a_{\alpha\alpha}$ - the sum of the diagonal elements of the matrix *A*;

$\sigma_2 = \sum_{\alpha<\beta} \begin{vmatrix} a_{\alpha\alpha} & a_{\alpha\beta} \\ a_{\beta\alpha} & a_{\beta\beta} \end{vmatrix}$ - the sum of all the diagonal minors of the second order of the matrix *A*;

$\sigma_3 = \sum_{\alpha<\beta<\gamma} \begin{vmatrix} a_{\alpha\alpha} & a_{\alpha\beta} & a_{\alpha\gamma} \\ a_{\beta\alpha} & a_{\beta\beta} & a_{\beta\gamma} \\ a_{\gamma\alpha} & a_{\gamma\beta} & a_{\gamma\gamma} \end{vmatrix}$ - the sum of all the diagonal minors of the third order of the matrix *A*;

and finally, $\sigma_n = \det A$.

It is easy to ensure that the number of the diagonal minors of *k- th* order of matrix *A* equals:

$$C_n^k = \frac{n(n-1)...(n-k+1)}{k!} \qquad (k = 1,2..., n).$$

Hence we find that the calculating of the coefficients of characteristic polynom is equivalent to calculation of $C_n^1 + C_n^2 + ... + C_n^n = 2^n - 1$ determinants of different order. The latest problem, generally speaking, is technically difficult implemented for large values of n. Therefore created special methods for deployment of characteristic determinants.

O.M. Krylov method

Let

$$D(\lambda) \equiv \det(\lambda E - A) = \lambda^n + p_1 \lambda^{n-1} + \ldots + p_n \qquad (7.7)$$

characteristic polynom (up to sign) of matrix A.

According to the identity of Hamilton-Cayley, matrix A turns in a zero its characteristic polynom, so

$$A^n + p_1 A^{n-1} + \ldots + p_n E = 0.$$

Take arbitrary nonzero vector $y^{(0)} = \begin{bmatrix} y_1^{(0)} \\ \vdots \\ y_n^{(0)} \end{bmatrix}$.

Multiplying both parts of (7.8) on the right on $y^{(0)}$, we get:

$$A^n y^{(0)} + p_1 A^{n-1} y^{(0)} + \ldots + p_n y^{(0)} = 0. \qquad (7.8)$$

Let's set

$$A^k y^{(0)} = y^{(k)} \quad (k = 1,2,\ldots,n), \qquad (7.9)$$

then equation (7.8) takes the form:

$$y^{(n)} + p_1 y^{(n-1)} + \ldots + p_n y^{(0)} = 0 \qquad (7.10)$$

or

$$\begin{bmatrix} y_1^{(n-1)} & y_1^{(n-2)} & \cdot & y_1^{(0)} \\ y_2^{(n-1)} & y_2^{(n-2)} & \cdot & y_2^{(0)} \\ \vdots & \vdots & \vdots & \vdots \\ y_n^{(n-1)} & y_n^{(n-2)} & \cdot & y_n^{(0)} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix} = -\begin{bmatrix} y_1^{(n)} \\ y_2^{(n)} \\ \vdots \\ y_n^{(n)} \end{bmatrix}, \qquad (7.10')$$

where $y^{(k)} = \begin{bmatrix} y_1^{(k)} \\ y_2^{(k)} \\ \vdots \\ y_n^{(k)} \end{bmatrix}$ $(k = 0,1,2,\ldots,n)$.

Thus, the vector equality (7.10) is equivalent to the system of equations:

$$p_1 y_j^{(n-1)} + p_2 y_j^{(n-2)} + \ldots + p_n y_j^{(0)} = -y_j^{(0)} \quad (j = 1,2,\ldots n).$$

from which, generally speaking, we can determine the unknown coefficients $p_1, p_2, \ldots, p_n$.

So based on the formula (7.9): $y^{(k)} = Ay^{(k-1)}$ $(k = 1,2,\ldots,n)$, the coordinates $y_1^{(k)}, y_2^{(k)}, \ldots, y_n^{(k)}$ of vector $y^{(k)}$ are sequentially calculated by the formula:

$$\left.\begin{aligned} y_i^{(1)} &= \sum_{j=1}^{n} a_{ij} y_j^{(0)}, \\ y_i^{(2)} &= \sum_{j=1}^{n} a_{ij} y_j^{(1)}, \\ &\cdots\cdots\cdots\cdots \\ y_i^{(n)} &= \sum_{j=1}^{n} a_{ij} y_j^{(n-1)} \quad (i = 1,2,\ldots,n). \end{aligned}\right\} \tag{7.11}$$

Thus, according to by Krylov method, calculating of the coefficients of the characteristic polynom (7.7) is reduced to solving a linear system of equations (7.10), the coefficients of which are calculated by formulas (7.11). And coordinates of the initial vector

$$y^{(0)} = \begin{bmatrix} y_1^{(0)} \\ \vdots \\ y_n^{(0)} \end{bmatrix}$$

are arbitrary. If the system (7.10) has a unique solution, then its roots $p_1, p_2, \ldots, p_n$ are the characteristic polynomial (7.7) coefficients. This solution can be found, for example, the method of Gauss. If the system (6) has no unique solution, the problem is complicated. In this case, it is recommended to change the initial vector.

Leverier method

This method of deployment of the characteristic determinant is based on Newton formulas for sums of powers of the roots of algebraic equations.

Let

$$\det(\lambda E - A) = \lambda^n + p_1\lambda^{n-1} + \ldots + p_n \tag{7.12}$$

— characteristic polynom of the matrix $A = \left[a_{ij}\right]$ and $\lambda_1, \lambda_2, \ldots, \lambda_n$ — complete aggregate of its roots, where each root is repeated as many times as its multiplicity.

Suppose $s_k = \lambda_1^k + \lambda_2^k + \ldots + \lambda_n^k \qquad (k = 0,1,2,\ldots, n)$.

Then, at $k \le n$ Newton's formulas justify:

$$s_k + p_1 s_{k-1} + ... + p_{k-1} s_1 = -k p_k \qquad (k = 0,1,2,..., n) \quad (7.13)$$

From here:

$$\left.\begin{aligned} & p_1 = -s_1, \\ & p_2 = -\frac{1}{2}\left(s_2 + p_1 s_1\right), \\ & \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ & p_n = -\frac{1}{n}\left(s_n + p_1 s_{n-1} + ... + p_{n-1} s_1\right). \end{aligned}\right\} \qquad (7.14)$$

If the sums $s_1, s_2, \ldots, s_n$ are known, then using formulas (7.14) we can step by step determine the coefficients $p_1, p_2, \ldots, p_n$ of the characteristic polynom (7.12).

The sums $s_1, s_2, \ldots, s_n$ are calculated as following: for s_1 we have: $s_1 = \lambda_1 + \lambda_2 + ... + \lambda_n = SpA$, i.e.

$$s_1 = \sum_{i=1}^{n} a_{ii}. \qquad (7.15)$$

Further, as we know, $\lambda_1^k, \lambda_2^k, ..., \lambda_n^k$ are the eigenvalues of matrix A^k. So $s_k = \lambda_1^k + \lambda_2^k + ... + \lambda_n^k = SpA^k$, that is, if $A^k = \left[a_{ii}^{(k)}\right]$, then

$$s_k = \sum_{i=1}^{n} a_{ii}^{(k)}. \qquad (7.16)$$

Degrees $A^k = A^{k-1}A$ are calculated by direct multiplication.

Thus, the scheme of deployment of characteristic determinant by Leverier method is very simple, namely:

- calculation of degrees A^k $(k = 1,2,..., n)$ of matrix A,
- then are found the corresponding s_k - sums of the elements of main diagonals matrix A^k,
- and, finally, by formulas (7.14) determine the unknown coefficients p_i $(i = 1,2,..., n)$.

Leverier method rather laborious because of counting the high degrees of matrix. Its main advantage - easy scheme of calculation and the absence of exceptional situations.

8. Interpolation problem with simple nodes. Vector interpolation problem with simple nodes

Formulation of the problem of interpolation

In the most general case the interpolation problem consists in constructing such a function $F(x)$, which in the given points $x_0, x_1, x_2, ...x_n$ gets values $f(x_0), f(x_1), f(x_2), ...f(x_n)$ of given function $f(x)$, and at other points of interval $[a,b]$ approximates it. Function $F(x)$ is called *interpolating* function towards $f(x)$.

Let in the interval $[a,b]$ are given $n+1$ points $x_0, x_1, ..., x_n$, which are called *interpolation nodes*, and the values of a function $f(x)$ at these points

$$f(x_0) = y_0, f(x_1) = y_1, ..., f(x_n) = y_n. \qquad (8.1)$$

We must construct a function $F(x)$, which belongs to the known class and takes in the interpolation nodes the same values as $f(x)$, i.e. such that

$$F(x_0) = y_0, F(x_1) = y_1, ..., F(x_n) = y_n. \qquad (8.2)$$

Geometrically this means that you need to find the curve $y = F(x)$ of a certain type, which is passing through a given system of points $M_i(x_i, y_i)$ $(i = 0,1,2,...)$

In this general formulation, the problem can have many solutions or do not have any.

However, this task becomes unambiguous, if instead of an arbitrary function $F(x)$ we search the polynom $P_n(x)$ of degree not greater than n, satisfying the condition (8.2), i.e. such as $P_n(x_0) = y_0, P_n(x_1) = y_1, ..., P_n(x_n) = y_n$.

The obtained interpolation formula $y = F(x)$ is usually used for calculating approximate values of the function $f(x)$ for values of the argument x that differs from interpolation nodes. Such operation is called the *interpolation of function* $f(x)$. Wherein is considered interpolation in the narrow sense, when $x \in [x_0, x_n]$, i.e. values of x are intermediate between x_0 and x_n, and *extrapolating,* when $x \bar{\in} [x_0, x_n]$. Further, the term interpolation we will use for the first and second operation.

The standard interpolation by Lagrange

For any given interpolation nodes often use so-called *Lagrange interpolation formula.*

Let in the interval $[a,b]$ are given $n+1$ different values of argument: $x_0, x_1, x_2, ..., x_n$ and known for the function $y = f(x)$ corresponding values: $f(x_0) = y_0,\ f(x_1) = y_1, ...,\ f(x_n) = y_n$.

We must build a polynomial $L_n(x)$ of degree not higher n, that has in given nodes $x_0, x_1, ..., x_n$ the same values as function $f(x)$, i.e. such as

$L_n(x_i) = y_i \quad (i = 0,1,2,...,n)$

At first let's solve the partial task: to build a polynom $p_i(x)$ such as

$p_i(x_i) = 0$ при $j \neq i$ і $p_i(x_i) = 1$.

In short, these conditions can be written as follows:

$$p_i(x_i) = \delta_{ij} = \begin{cases} 1, & if \quad j = i \\ 0, & if \quad j \neq i \end{cases} \tag{8.3}$$

where δ_{ij} - Kronecker symbol.

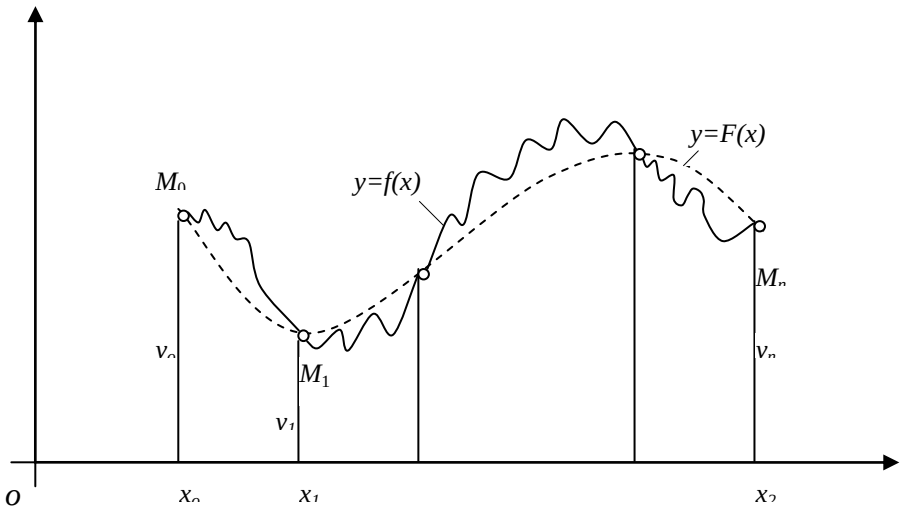

Рис. 8.1. Interpolation polynom of Lagrange

As the sought-for polynom becomes zero at n points $x_0, x_1, ..., x_{i-1}, x_{i+1}, ..., x_n$, it looks as following:

$$p_i(x) = C_i(x - x_0)(x - x_1)...(x - x_{i-1})(x - x_{i+1})...(x - x_n), \qquad (8.4)$$

where C_i - constant coefficient. Putting $x = x_i$ in the formula (8.4) and considering that $p_i(x_i) = 1$, we get:

$$C_i(x_i - x_0)(x_i - x_1)...(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_n) = 1.$$

From here:

$$C_i = \frac{1}{(x_i - x_0)(x_i - x_1)...(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_n)}.$$

Substituting this value in the formula (8.4), we have:

$$p_i(x) = \frac{(x - x_0)(x - x_1)...(x - x_{i-1})(x - x_{i+1})...(x - x_n)}{(x_i - x_0)(x_i - x_1)...(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_n)}. \qquad (8.5)$$

Now let's solve the general problem: to find a polynomial $L_n(x)$ that satisfies the conditions specified above: $L_n(x_i) = y_i$.

This polynom is as follows:

$$L_n(x) = \sum_{i=0}^{n} p_i(x) y_i. \qquad (8.6)$$

In fact, the first, obviously, the degree of the polynom $L_n(x)$ is not higher than n, and secondly, on condition (8.3) we have:

$$L_n(x_j) = \sum_{i=0}^{n} p_i(x_j) y_j = p_j(x_j) y_j = y_j \qquad (j = 0,1,...,n).$$

The vector interpolation by Lagrange

Let in the interval $[a,b]$ are given point t_k (interpolation nodes) and the vectors $v_k = \{x_k, y_k\}$ as the values of some vector-function $f(t)$ at these points, i.e.

$$f(t_k) = v_k = \{x_k, y_k\}, \qquad k = 0,1,.....,n. \qquad (8.7)$$

We must find a vector-function $r(t) = \{x(t), y(t)\}$, $t \in [a,b]$, graph of which contains points $M(t_j) = v_j = \{x_j, y_j\}$.

In case of difference of nodes ($t_j \neq t_k$ при $j \neq k$, $j,k \in 0,n$), the interpolation problem (4.9) always has a unique solution:

$$r(t) = P_n(t) = \sum_{s=0}^{n} p_s t^s \qquad (8.8)$$

in the class P_n of polynoms of degree *n* of one variable with vector coefficients p_s. Finding of the polynomial $P_n(x) = \sum_{s=0}^{n} p_s t^s$ is reduced to solving of compatible systems of linear equations:

$$P_n(t_k) = \sum_{s=0}^{n} p_s t_k^s = v_k = \{x_k, y_k\}, \qquad k = 0,1,2,...,n \quad (8.9)$$

relatively to the vector-coefficients $\{p_0, p_1,....., p_n\}$.

But there's another way to solve this problem - you need to use the Lagrange formula for vector interpolation problem:

$$P_n(t) = L(t) = \sum_{s=0}^{n} v_s L_s(t) = \sum_{s=0}^{n} \{x_s, y_s\}_s L_s(t), \qquad (8.10)$$

where $L_s(t)$ – elementary Lagrange polynomials defined by the formulas:

$$L_s(t_k) = \frac{(t-t_0).....(t-t_{s-1})(t-t_{s+1}).....(t-t_n)}{(t_s-t_0).....(t_s-t_{s-1})(t_s-t_{s+1}).....(t_s-t_n)}, \qquad s=0,n. \qquad (8.11)$$

Software that implements the described algorithm has been developed . The text of the main program procedures is given in the appendix 12.

9. Bezier curves on the plane and in space

Bernstein polynomials

Let's we have function $f(x) \in C([0,1])$. Bernstein polynomial $b_n(f;x)$ - is the polynom

$$b_n(f;x) := \sum_{m=0}^{n} p_{n,m}(x) f\left(\frac{m}{n}\right) = \sum_{m=0}^{n} f\left(\frac{m}{n}\right) C_n^m x^m (1-x)^{n-m}, \ x \in [0;1] \quad (9.1)$$

At $n \to \infty$ the Bernstein polynomial $b_n(f;x)$ converges uniformly on the interval $[0;1]$ to the function $f(x)$:

$$\lim_{n\to\infty} \| b_n(f;x) - f(x) \| = 0 \qquad (9.2)$$

Thus we have the estimate

$$\| b_n(f;x) - f(x) \| \le \omega\left(f; \frac{1}{\sqrt{n}}\right), \qquad (9.3)$$

Standard operator Bernstein $B_n : C([0,1]) \to C([0,1])$ acts by the formula:

$$B_n(f)(x) := b_n(f;x) = \sum_{m=0}^{n} p_{n,m}(x) f\left(\frac{m}{n}\right) = \sum_{m=0}^{n} f\left(\frac{m}{n}\right) C_n^m x^m (1-x)^{n-m}. \qquad (9.4)$$

Function $y = \varphi(x) = a(1-x) + bx$ reflects the interval $[0,1]$ to the interval $[a,b]$, and function $x = \psi(y) = \frac{y-a}{b-a}$ performs inverse transformation the interval $[a,b]$ to the interval $[0,1]$. These functions generate mutually inverse reflections

$\varphi^* : C([a,b]) \to C([0,1])$ і $\psi^* : C([0,1]) \to C([a,b])$

by formulas:

$(\varphi^* f)(x) = f(\varphi(x)) = f(a + (b-a)x)$, ,

$$(\psi^* f)(y) = f(\psi(y)) = f\left(\frac{y-a}{b-a}\right), \quad y \in [a,b].$$

Bezier curves on the plane and its properties

Let's define basic Bernstein polynomials

$$p_{n,k}(t) = C_n^k t^k (1-t)^{n-k} = \frac{n!}{k!(n-k)!} t^k (1-t)^{n-k}, \quad t \in [0,1], \quad k = 0,\ldots,n .$$

Bezier curve is defined by vertices P_k (k = 0,1,…,n) of basic polygon, which uniquely identifies the position of the curve in the plane or in space. To construct a Bezier curve we using Bernstein polynomials and the vertices $P_k(x_k,y_k)$, k = 0,1,…,n, that is, Bezier curve looks like:

$$P(t) = \sum_{k=0}^{n} p_{n,k}(t) P_k = \sum_{k=0}^{n} C_n^k t^k (1-t)^{n-k} P_k, \quad t \in [0,1] \qquad (9.5)$$

where $p_{n,k}(t)$ - *κ-th* function of Bernstein basis of order n, its maximum is reached at $t=k/n$.

Or in the the components:

$$\begin{cases} x(t) = \sum_{k=0}^{n} p_{n,k}(t)\, x_k = \sum_{k=0}^{n} C_n^k\, t^k (1-t)^{n-k}\, x_k \\ y(t) = \sum_{k=0}^{n} p_{n,k}(t)\, y_k = \sum_{k=0}^{n} C_n^k\, t^k (1-t)^{n-k}\, y_k \end{cases}, \quad t \in [0,1] \qquad (9.6)$$

It is easy to see that each of these components can be count separately as Bernstein polynomial for corresponding coordinate functions of the parameter t.

For Bezier curve we have the following *properties*:

1. $P(0) = P_0$ (The point P_0 is the starting point of Bezier curve).
2. $P(1) = P_n$ (The point P_0 is the end point of Bezier curve).
3. $P'(0) = n[P_1 - P_0]$ (vector $P_1 - P_0$ determines the direction of the tangent to the Bezier curve at the start point of the curve).
4. $P'(1) = n[P_n - P_{n-1}]$ (vector $P_n - P_{n-1}$ determines the direction of the tangent to the Bezier curve at the end point of the curve).
5. $P''(0) = n(n-1)[P_0 - 2P_1 + P_2]$
6. $P''(1) = n(n-1)[P_n - 2P_{n-1} + P_{n-2}]$

If we conjugate two segment Bezier curve with the basic polygons $\mathbf{P}_0, \mathbf{P}_1, ...\mathbf{P}_n$ and $Q_0, Q_1, ...Q_m$, then the condition of matching of last point of the 1-*st* segment and the first point of 2-*nd* segment (condition coincidence of segments) takes the form:

$$\mathrm{P}_n = Q_0. \tag{9.7}$$

Conditions preservation slope tangents at the point of connection of segments has the form:

$$\mathrm{P}'_n = \lambda Q'_0 \Leftrightarrow n[\mathrm{P}_n - \mathrm{P}_{n-1}] = \lambda m[Q_1 - Q_0]. \tag{9.8}$$

If you need to not change the length of the tangent, we get a condition of tangential connection:

$$\mathrm{P}'_n = Q'_0 \Leftrightarrow n[\mathrm{P}_n - \mathrm{P}_{n-1}] = m[Q_1 - Q_0] \tag{9.9}$$

Similarly, there is a condition of conjugate of derivates of 2-*nd* order*:*

$$\mathrm{P}''_n = Q''_0 \Leftrightarrow n(n-1)[\mathrm{P}_{n-2} - 2\mathrm{P}_{n-1} + \mathrm{P}_n] = m(m-1)[Q_0 - 2Q_1 + Q_2] \tag{9.10}$$

Example. Let $\mathrm{P}_0 = \{1,1\}$, $\mathrm{P}_1 = \{2,3\}$, $\mathrm{P}_2 = \{4,3\}$, $\mathrm{P}_3 = \{3,1\}$.

Then

$$n = 3, b_{3,0}(t) = C_3^{\,0} t^0 (1-t)^3 = (1-t)^3, b_{3,1}(t) = C_3^{\,1} t^1 (1-t)^2 = 3t(1-t)^2,$$
$$b_{3,2}(t) = C_3^{\,2} t^2 (1-t)^1 = 3t^2(1-t), b_{3,3}(t) = C_3^{\,3} t^3 (1-t)^0 = t^3,$$
$$\mathrm{P}(t) = b_{3,0}\mathrm{P}_0 + b_{3,1}\mathrm{P}_1 + b_{3,2}\mathrm{P}_2 + b_{3,3}\mathrm{P}_3 = (1-t)^3\{1,1\} + 3t(1-t)^2\{2,3\} + 3t^2(1-t)\{4,3\} + t^3\{3,1\} =$$
$$= \{x(t), y(t)\}$$

Or by coordinates:

$$\begin{cases} x(t) = -4t^3 + 3t^2 + 3t + 1 = (t+1)^3 - 5t^3 \\ y(t) = -6t^2 + 6t + 1 = 5/2 - 6(t - 1/2)^2 \end{cases}.$$

Features approximation using Bezier curves

1. Degree (order) of curve is by one less than the number of vertices of the base polygon. So, the only way to reduce the degree of curve - is to decrease of the number of vertices of the base polygon.

2. All the functions $b_{n,k}(t)$ are not equal to zero in the interval [0,1], so changing one vertex of base polygon changes the entire curve.

10. Linear and homogeneous coordinates on the plane

Basic concepts and definitions

Three points $\{A_o, A_1, A_2\}$ on a plane π are called *points of general location,* if there is no line, which includes all these points. These points do not coincide with each other and the line, passing through two of these points, contains no third point.

Rapper A on the plane π we will call the ordered three points $A = \{A_o, A_1, A_2\}$ of general location on this plane π.

Linear coordinates on the plane π, which are defined by rapper $A = \{A_o, A_1, A_2\}$ or the *system of linear coordinates,* we will call a couple of reflections: $\mathrm{Crd}_A : \pi \to R^2$ (two-dimensional coordinates of the point) and

$\mathrm{Pnt}_A : R^2 \to \pi$ (point with two-dimensional coordinates)

that satisfy the following conditions:

1) Reflections Crd_A and Pnt_A are reciprocal

$$\begin{cases} \mathrm{Pnt}_A(\mathrm{Crd}_A(X)) = X \ \forall X \in \pi, \\ \mathrm{Crd}_A(\mathrm{Pnt}_A(x)) = x \ \forall x \in R^2. \end{cases} \quad \text{або} \quad \begin{cases} \mathrm{Pnt}_A \circ \mathrm{Crd}_A = 1_\pi, \\ \mathrm{Crd}_A \circ \mathrm{Pnt}_A = 1_{R^2}. \end{cases}$$

2) Normalization performed

$$\begin{cases} \mathrm{Crd}_A(A_0)) = \{0;0\}, \ \mathrm{Crd}_A(A_1)) = \{1;0\}, \ \mathrm{Crd}_A(A_2)) = \{0;1\}, \\ \mathrm{Pnt}_A(\{0;0\}) = A_0, \quad \mathrm{Pnt}_A(\{1;0\}) = A_1, \quad \mathrm{Pnt}_A\{0;1\} = A_2. \end{cases}$$

Let on plane π are two systems of linear coordinates (SLC)**.** The first (*x*-coordinates) with rapper $A = \{A_o, A_1, A_2\}$, and the second (*y*-coordinates) – with rapper $B = \{B_o, B_1, B_2\}$. Let the point $X \in \pi$ has coordinates $x = \{x_1, x_2\}$ in the first SLCand coordinates $y = \{y_1, y_2\}$ in the second SLC. Then the reflection $f : R^2 \to R^2$ and $q : R^2 \to R^2$, which are defined by formulas $y = f(x) := \mathrm{Crd}_B(\mathrm{Pnt}_A(x)), \ x = q(y) := \mathrm{Crd}_A(\mathrm{Pnt}_B(y)) \ \forall x, y \in R^2$ define replacement of *x*-coordinates to *y*-coordinates and vice versa.

Structure of the totality of systems of linear coordinates on the plane

Let on plane π there is a systems of linear coordinates with rapper A of three points of general location i $\varphi: R^2 \to R^2$ – affine species of the type $\varphi(x) = x\alpha + \beta (x, b \in R^2, \alpha \in M_2, \det\alpha \neq 0)$. Then the reflection $\psi(y) = y\alpha^{-1} - \beta\alpha^{-1}$ is inverse to reflection φ and exists the rapper $C = \{C_0, C_1, C_2\}$, for which the following conditions satisfy: $\psi \circ C\ r_A d \equiv C\ r_C d$ $P\ n_A t \varphi \equiv P\ n_C t$. Wherein $C_0 = Pnt_A(\beta), C_1 = Pnt_A(\{1,0\}\alpha + \beta)$.

Consequently, transferring the coordinates of points at a fixed SLC using arbitrary reflection $\varphi: R^2 \to R^2$ $(\varphi(x) = x\alpha + \beta,$ $x, b \in R^2, \alpha \in M_2, \det\alpha \neq 0),$ for which the transfer to fixed coordinates SLC via display the coordinates of an arbitrary point in the SLC.

Homogeneous coordinates on the plane

Homogeneous coordinates on the plane π - is a set of ordered triples of numbers $\{x_1, x_2, x_3\}$ $(x_1^2 + x_2^2 + x_3^2 \neq 0),$ for which the following equivalence is introduced. The triple $\{x_1, x_2, x_3\}$ is equivalent to the triple $\{y_1, y_2, y_3\}$ $(\{x_1, x_2, x_3\} \approx \{y_1, y_2, y_3\}),$ if exists such t>0, for which $y_k = tx_k (k = 1,2,3).$

Usually coordinates $\{x_1, x_2\}$ define a single class $[x_1, x_2, 1]$. Reverse requires consideration of two cases.

Class $[x_1, x_2, x_3]$ at $x_3 \neq 0$, i.e. consider triples $\{x_1, x_2, x_3\}$ $(x_3 \neq 0).$ Let $x_1^* = x_1 / x_3,$ $x_2^* = x_2 / x_3.$

At $x_3 > 0$ we have $\{x_1, x_2, x_3 \approx \{x_1^*, x_2^*, 1\} \in \pi$ (points of front). So, front $\pi_+ \equiv \pi$ – is the set of classes $[x_1, x_2, 1].$

At $x_3 < 0$ we have $\{x_1, x_2, x_3\} \approx \{-x_1^*, x_2^*, 1\}$ (points of rear front). So, rear front π – is the set of classes $[x_1, x_2, -1].$

1) Class $[x_1, x_2, 0] \in P^2$, i.e. each triple $\{x_1, x_2, 0\}$ $(x_1^2 + x_2^2 + x_3^2 \neq 0),$

defines on the plane the only point at infinity, which corresponds to a ray that starts at the origin *O*. Always for each class $[x_1 x_2 0]$ we can consider normalized triple $\{x, x_2, 0\}$ $(x_1^2 + x_2^2 = 1)$. Wherein vector $x = \{x_1, x_2\} \in S^1$ defines the ray $p_x = \{(y_1, y_1, 0) \in R^3 \mid y_1 = tx_1, y_2 = tx_2, t > 0\}, x \in S^1$.

So, due to the determination of uniformity at $R^3 \setminus \{0\}$ for classes $x_1 x_2 x_3$ we can assume that the third coordinate is $\pm 1 (x_3 \neq 0)$ or 0 $(x_3 = 0)$, i.e. we have following cases:

1. At $x_3 = 1$ we have the front points π_+
2. At $x_3 = -1$ we have the points of rare front π_-,

3. At $x_3 = 0$ we have points at infinity of plane π or the skyline π_0.

Mathematical coordinates and the coordinates of the device

Consider on the plane π the Cartesian system of linear coordinates $(\{x, y\}$-coordinates) centered at point O(0,0).

Definition. *Mathematical window* – is a rectangle *ABCD* on the plane, (bypass circuit - counterclockwise) whose sides parallel to the axes *OX* and *OY*. The lenthes of the rectangle sides determine the size $r_x = \| AB \| = \| CD \|$ and $r_y = \| BC \| = \| AD \|$ of mathematical window along axes, respectively, OX and OY.

Let $S(s_x, s_y)$ – the center of the screen. Then for the vertices of the rectangle of window we have the coordinates

$\{s_x - \frac{r_x}{2}, s_y + \frac{r_y}{2}\}$ (point *A*), $\{s_x + \frac{r_x}{2}, s_y + \frac{r_y}{2}\}$ (point *B*),

$\{s_x + \frac{r_x}{2}, s_y - \frac{r_y}{2}\}$ (point *C*), $\{s_x - \frac{r_x}{2}, s_y - \frac{r_y}{2}\}$ (point *D*).

So, knowing the coordinates of the center of the screen *S* and the size of the rectangle, we completely determine the mathematical window.

Definition. On its part, with mathematical Cartesian SLC is related so-called coordinate system of the device $\{\xi, \eta\}$, - a physical screen coordinate

system which addresses the physical point (pixel) of screen. The origin of coordinate of device system coincides with the top left corner of the mathematical window (point *D*), direction of axes *X* and ξ coincide (direction - right) and axes *Y* and η - is opposite (Y- upwards, η - downwards). Screen size along the axes ξ and η is measured in pixels – m_x along the axis ξ and m_y along the axis η.

Definition. *Window of device* or *display area* - is also a rectangle *ABCD*, whose sides are parallel to the bounds of screen, and the points *A*, *B*, *C*, *D* coincide with the boundary corner points of the screen. Sizes of sides of the rectangle are specified in pixels and determine the size of the screen: $m_x = \| AB \|_{pixels} = \| CD \|_{pixels}$ (width sweep) and $m_y = \| BC \|_{pixels} = \| AD \|_{pixels}$ (height sweep). But we must note, that when programming the coordinates of points are measured from zero (*0* - is the *first* pixel, *1* - *second* pixel, etc.) then the last *m-th* pixel will be addressed at the number *m-1*. Therefore, coordinates of the vertices of the display area are: $\{0, m_y - 1\}$ (point *A*), $m_x - 1, m_y - 1\}$ (point *B*), $\{m_x - 1, 0\}$ (point *C*), і $\{0,0\}$ (point *D*).

Formula changes *{x, y}* - coordinates to $\{\xi, \eta\}$ - coordinates should look like $\begin{cases} \xi = \alpha x + \beta y + \varepsilon, \\ \eta = \gamma x + \delta y + \omega. \end{cases}$ Therefore, after the substitution of mathematical coordinates and corresponding device coordinates for points *A*, *B*, *C* (point *D* is uniquely determined by points *A*, *B*, *C*, and so it can be neglected), obtain a system of six linear equations with six unknowns $(\alpha, \beta, \varepsilon, \gamma, \delta, \varpi)$. By solving these equations, we obtain matching math coordinates and the coordinates of the device while centering on the center $S(s_1, s_2)$ of the window:

$$\begin{cases} \xi = k_x (x - s_x + \frac{1}{2} r_x), \\ \eta = k_y (s_y + \frac{1}{2} r_y - y). \end{cases} \quad \text{і} \quad \begin{cases} x = s_x - \frac{1}{2} r_x + \frac{\xi}{k_x}, \\ y = s_y + \frac{1}{2} r_y - \frac{\eta}{k_y}. \end{cases}$$

where $k_x = \frac{m_x - 1}{r_x}$, $k_y = \frac{m_y - 1}{r_y}$. But preserving the aspect ratio while displaying on the screen, we must assume that the relation $r_x / r_y = m_x / m_y$ are satisfied, тобто $r_y = r_y \frac{m_y}{m_x}$ and we must calculate k_y by the formula: $k_y = \frac{m_y - 1}{r_x} \cdot \frac{m_x}{m_y}$.

11. Basic conversion in the plane. The main symbol of affine transformations

The concept of transformation on the plane

Let on the plane π we have some system of linear coordinates. For any linear transformation A its mathematical expression in ordinary coordinates has the form $(A): x' = x\alpha + \beta, \alpha = \begin{pmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{pmatrix} \in \mathrm{M}_2, x' x, \beta \in R^2$, i.e. the coordinate expression in ordinary coordinates has the form $(A): \begin{cases} x_1' = a_{11}x_1 + a_{21}x_2 + \beta_1, \\ x_2' = a_{12}x_1 + a_{22}x_2 + \beta_2. \end{cases}$.

Substitution $\{x_1', x_2', x_1, x_2\} \to \{x_1' / x_3', x_2' / x_3', x_1 / x_3, x_2 / x_3\}$, on condition, that $x_3' = x_3$, gives the coordinate expression of transformation in *homogeneous coordinates*:

$$(A): \begin{cases} x_1' = \alpha_{11}x_1 + \alpha_{21}x_2 + \beta_1 x_3, \\ x_2' = \alpha_{12}x_1 + \alpha_{22}x_2 + \beta_2 x_3, \\ x_3' = x_3. \end{cases}$$

or we get the matrix expression of transformation in homogeneous coordinates:

$$[x'] = [x] \cdot [A] \Leftrightarrow [x_1', x_2', x_3'] = [x_1, x_2, x_3] \cdot \begin{bmatrix} \alpha_{11} & \alpha_{12} & 0 \\ \alpha_{21} & \alpha_{22} & 0 \\ \beta_1 & \beta_2 & 1 \end{bmatrix}.$$

Let's consider transformation A, linear on plane π. The main and the full symbols of transformation A we will call the matrices:

$$\sigma_m(A) := \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \sigma(A) := \begin{bmatrix} \alpha_{11} & \alpha_{12} & 0 \\ \alpha_{21} & \alpha_{22} & 0 \\ \beta 1 & \beta_2 & 1 \end{bmatrix}$$

(here symbol "'*m*" means main). We shall also denote the symbols $\sigma_m(A)$ and $\sigma(A)$ as $[A]_m$ i $[A]$.

The *fixed point* of transformation $P: \pi \to \pi$ is the point $X \in \pi$, which remains in place under the action of transformation P, i.e. it is a solution of equation $X = P(X)$.

Basic transformation on the plane

1. Parallel transfer

Parallel transfer $T_a: \pi \to \pi$ by vector *a* on the plane π can be defined in different ways.

Geometric definition. 1) At $a = 0 = \{0,0\}$ (zero vector) let $X' = T_0(X) = X$, i.e. $T_0 = 1$ (identity transformation on plane π); 2) At $a \neq 0 = \{0,0\}$ (nonzero vector) for any point $X \in \pi$ from point X postpone segment XX', which has a length of vector *a*. Wherein the direction of vector from point X to point X' coincides with the direction of vector *a*. Point X' is required $(T_a(X) := X')$.

Vector definition. For any point $X \in \pi$ let's define the point $X' \in \pi$ as the only solution of the vector equation relative X' (vector expression of parallel transfer): $\overline{XX'} = a, X \in \pi, a \| \pi$. Then we put: $T_a(X) := X'$.

Affine definition. If O - some fixed point in space, then the vector expression can be written as affine expression of parallel transfer: $(T_a): \overline{X'} = \overline{X} + a, \quad \forall X \in \pi$. So, with this formula we find the coordinates of vector $\overline{X'}$ and then the point X' and put: $T_a(X) := X'$.

Coordinate definition. Let on the plane π we have system of linear coordinates. If as the origin of the coordinate system we choose $O = P_0$, then from affine definition we receive the coordinate definition of parallel transfer:

$$(T_a):\begin{cases} x_1' = x_1 + a_1 \\ x_2' = x_2 + a_2 \end{cases},$$

where $\{a_1, a_2\}$ *and* $\{x_1, x_2\}$ - the coordinates of vector *a* and point *X* in our system of linear coordinates. I.e., with the last expression we find the coordinates $\{x'_1, x'_2\}$ of point X' and put: $T_a(X) := X'$.

The set of transformations $Shifts(\pi) := \{T_a : \pi \to \pi \,|\, a \| \pi\}$ forms a commutative group of parallel transfers relative composition $\circ$. Wherein: 1) There is a law of multiplication $T_b \circ T_a = T_{a+b}, a, b \| \pi$; 2) The unit of group is the element $T_0 = 1$; 3) Inverse to element T_a is the element T_{-a}.

2. *Rotation*

Rotation $R_A^{(\alpha)} : \pi \to \pi$ on the plane π around the point $A \in \pi$ by the angle $\alpha \in R$ (radian).

Geometric definition 1) At *X=A* (the center of rotation) we put: $X' = R_A^{(\alpha)}(X) = X$ (i.e. $A \in Fix(R_A^{(\alpha)})$); 2) At $X \neq A$ for any point $X \in \pi$ from the point A postpone segment AX', which has a length of the segment AX ($\|AX'\| = \|AX\|$) and the angle between the segments AX i AX' (taking into account the direction of rotation) is equal $\alpha(\angle(AX, AX') = \alpha)$. The point X' is required. So, we put: $R_A^{(\alpha)}(X) := X'$

3. Axial homotetia

The homotetia $H_l^{(k)} : \pi \to \pi (k \neq 0)$ on the plane π relative to the line $l \subset \pi$ can be defined in different ways.

Geometric definition. 1) At $X \in l$ (axis of symmetry) let's put: $H_l^{(k)}(X) = X$; 2) At $X \notin l$ we draw the line through the point X. This line $l'_X \in \pi$, which is perpendicular to the line l $(X \in l'_X \subset \pi, l'_X \perp l)$. Let $A' = l'_A \times l$ - point of intersection of lines l'_X *il.* On the line l'_X from point A' (at k <0 - in the opposite direction from point *X;* at k >0 - on the same side as the point *X*) we

postpone segment $A'X'$, which length is equal to the length of the segment $A'X$, which multiplied by number $|k| > 0 \left(\|A'X'\| = |k| \cdot \|A'X\|\right)$.

Vector definition. For any point $X \in \pi$ let's define the point $A' \in l$, for which the vector $A'X$ has the smallest length: $\|A'X\| = \min_{A \in l} \|AX\|$ (corresponding segment is perpendicular to the line l). It is clear that point A' depends on the line l and the point X. Let's define the point $X' \in \pi$ as the only solution of the vector equation relative X': $\overline{A'X'} = k\overline{A'X} \forall A, X \in \pi$ (vector expression of the axial homotetia). Thus, first of all we find the point A' (as a basis of the perpendicular XA' on the line l), from vector expression of the axial homotetia, we find the vector $\overline{A'X'}$ and then – the point X'. Then we put: $H_l^{(k)}(X) := X'$. Wherein, if $X \in \pi$, then $X' \in \pi$.

Affine definition. Let O - any point in space and the point A' is found as in vector expression of the axial homotetia. Then, at first, vector expression of the axial homotetia can be written as an affine expression of the axial homotecia: $\left(H_l^{(k)}\right)$: $\overline{X'} = k\overline{X} - (1-k)\overline{A'} \quad \forall X \in \pi$. So, from the affine expression of the axial homotecia we find the vector $\overline{X'}$, and then the point $X' := O + \overline{X'}$ and put: $H_l^{(k)}(X) := X'$. Wherein, if $A, X \in \pi$, then $X' \in \pi$.

Coordinate definition. Let on the plane π we have system of linear coordinates (x - coordinates). If the point $O = A_0$, then from the affine expression of the axial homotetia we can obtain the coordinate expression $H_l^{(k)}$. But here we have some uncertainty: we don't know the dependence of the point $A'\left(x_1, x_2'\right)$ upon the coordinates of point $X\{x_1, x_2\}$. But here we have some uncertainty: we don't know the dependence on the point $A'\left(x_1, x_2'\right)$ upon the coordinates of point $X\{x_1, x_2\}$ and the coefficients of the equality of line l. From analytic geometry we know, that the point A' satisfies the condition: if l_X' - the line passing through the point X and is perpendicular to the line l, the point A' belongs to the line l. This allows you to find the coordinates of the point A' over

the coordinates $\{x_1, x_2\}$ of X and normalized equation coefficients $\{c, s, p\}$ of line l:

$$(l): cx_1 + sx_2 + p = 0 \qquad c := \cos\alpha, s := \sin\alpha, p \in R \tag{11.1}$$

(here $n = \{c, s\} = \{\cos a, \sin a\}$ - orthogonal vector normal to the line l and a - the angle of the vector normal to the axis OX_1). Really, let $\{a_1', a_2'\}$ – unknown coordinates of the point A'. Then the condition $A' \in l$ gives the first equation to find the coordinates of the point A':

$$ca_1' + sa_2' + p = 0 \tag{11.2}$$

From analytic geometry we know, that the unit vector $n := \{c, s\}$ is perpendicular to the line l, which is defined by the equation (11.1). Therefore parametric equation of the line $l_X \in \pi$, which is perpendicular to the axis of homotecia l and for which $X \in l_X$, looks like: $\begin{cases} \xi_1 = c \cdot t + x_1, \\ \xi_2 = s \cdot t + x_2. \end{cases}$

Let at $t = \tau$: $\begin{cases} a_1' = c \cdot \tau + x_1, \\ a_2' = s \cdot \tau + x_2. \end{cases}$

Substituting these values in (11.2), we obtain:

$$c(c \cdot \tau + x_1) + s(s \cdot \tau + x_2) + p = 0 \Rightarrow \tau = -(c \cdot x_1 + s \cdot x_2 + p),$$

so

$$\begin{cases} a_1' = -c(c \cdot x_1 + s \cdot x_2 + p) + x_1 = (1 - c^2)x_1 - scx_2 - cp \\ a_2' = -s(c \cdot x_1 + s \cdot x_2 + p) + x_2 = -scx_1 + (1 - s^2)x_2 - sp \end{cases}$$

Substituting this coordinates to the affine expression of the axial homotetia, we get the coordinate expression of the axis homotecia for normalized line equation:

$$(H_l^{(k)}): \begin{cases} x_1' = [kc^2 + s^2]x_1 + (k-1)scx_2 + (k-1)cp, \\ x_2' = (k-1)scx_1 + [c^2 + ks^2]x_2 + (k-1)sp. \end{cases} \tag{11.3}$$

Thus, we find with the formulas (9) the coordinates (x_1', x_2') of the point X' and put: $H_l^{(k)}(X) := X'$. Wherein if $X \in \pi$, then $X' \in \pi$.

4. Axial symmetry

Symmetry $S_l : \pi \to \pi$ relatively line $l \subset \pi$ can also be defined in different ways.

Geometric definition. 1) At $X \in l$ (the axis of symmetry) we put $S_l(A) = A$. 2) At $X \notin l$ we hold a straight line $l'_X \in \pi$ through the point A, and this line is perpendicular to the line l ($X \notin l'_X \subset \pi, l'_X \perp l$). Let $A' = l'_A \times l$ - the point of intersection of straight lines l'_x i l. On the line l'_X from a point A' in the opposite direction from the point X the segment $A'X'$ is postponed. Its lenth is equal to the lenth of segment $A'X$ ($\| A'X' \| = \| A'X \|$). The obtained point X' is required (i.e., we put $S_l(X) := X'$).

Coordinate expression of the axis homotetia for the normalized line equation:

$$(H_l^{(k)}): \begin{cases} x'_1 = [kc^2 + s^2]x_1 + (k-1)scx_2 + (k-1)cp, \\ x'_2 = (k-1)scx_1 + [c^2 + ks^2]x_2 + (k-1)sp. \end{cases} \qquad (11.4)$$

Fixed points of transformations in the plane

A *fixed point* of transformation is determined using the coordinate expression $[x'] = [x][A]$ in homogenious coordinates. It satisfies the condition $x' = \lambda x$ $(\lambda > 0)$. So, to find fixed points of the transformation A, we must find positive eigenvalues of the full character $[A]$. The corresponding eigenvectors (i.e., non-zero solutions of equations $x[A] = \lambda x$) there are the homogeneous coordinates of fixed points.

12. The compositions of affine transformations on the plane

Transformations on the plane

By this time, the points on the plane π were considered as fixed: for each fixed point X in each system of linear coordinates from the set $\mathrm{Crd}(\pi)$ were considered different coordinates of this fixed point and we studied only changesof *x - coordinates* on *y - coordinates*: $y = x\alpha + \beta$ $(\alpha \in \mathrm{M}_2,\ \det\alpha \neq 0,\ \beta \in \mathrm{R}^2)$ or $\mathrm{x} = \mathrm{Crd}_{\mathrm{A}}(X) \to \mathrm{Crd}_{\mathrm{A}'}(X)$, where X - fixed point and A, A' - two alterable rappers. But you can do the conversely: fix (once for all) some system of linear coordinates with $\mathrm{Crd}(\pi)$ and consider the affine transformation of the plane π.

Let on the plane π we have system of linear coordinates (*x* - coordinates). Linear transformation $P: \pi \to \pi$ is the transformation, which has the coordinate expression of the type: $x' = x\alpha + \beta$, where x, x' - vector-lines of coordinates of corresponding points, $\alpha \in \mathrm{M}_2, \beta \in \mathrm{R}^2$. If we add here the condition $\det\alpha \neq 0$, we get the definition of affine transformation. Coordinate expression of transformation P can be written in such matrix way: $(P): x' = x\alpha + \beta$ (for ordinary coordinates).

There is an inclusion $\mathrm{Aff}(\pi) \subset \mathrm{Lin}(\pi)$. The converse is not true. For example, constructing the projection in fixed point $B(b)$ with coordinate expression $(P): x' = b$ is the linear transformation but it is not affine. But if the linear transformation has an inverse one, then it is affine transformation

The concept of composition of transformations on the plane

Composition of transformations $A, B: \pi \to \pi$ - is the transformation $C: \pi \to \pi$, which is defined by formula: $C(X) := A(B(X)),\ \forall X \in \pi$. The

composition of transformations A and B(order of transformations is essential) we will denote as $B \circ A$.

Note. The general line l equation is not normalized, i.e.

$$l_1 x_1 + l_2 x_2 + l_3 = 0, \qquad l_1^2 + l_2^2 \neq 0 \tag{12.1}$$

When you divide (11) on $L = \sqrt{l_1^2 + l_2^2}$ obtain the normalized equation. The last action is equivalent to substitution $\{c, s, p\} \to \{\frac{l_1}{L}, \frac{l_2}{L}, \frac{l_3}{L}\}$ in formulas (9). Therefore, in the case of general line equation (11) we obtain the coordinate expression of the axis homotetia for general line equation:

$$(H_l^{(k)}): \begin{cases} x_1' = \dfrac{kl_1^2}{l_1^2 + l_2^2} x_1 + \dfrac{(k-1)l_1 l_3}{l_1^2 + l_2^2} x_2 + \dfrac{(k-1)l_1 l_3}{l_1^2 + l_2^2} \\ x_2' = \dfrac{(k-1)l_1 l_2}{l_1^2 + l_2^2} x_1 + \dfrac{l_1^2 1 + kl_2^2}{l_1^2 + l_2^2} x_2 + \dfrac{(k-1)l_2 l_3}{l_1^2 + l_2^2} \end{cases} \tag{12.2}$$

Note. The general line l equation is not normalized, i.e.

$$l_1 x_1 + l_2 x_2 + l_3 = 0, \qquad l_1^2 + l_2^2 \neq 0 \tag{12.3}$$

When you divide (11) on $L = \sqrt{l_1^2 + l_2^2}$ obtain the normalized equation. The last action is equivalent to substitution $\{c, s, p\} \to \{\frac{l_1}{L}, \frac{l_2}{L}, \frac{l_3}{L}\}$ in formulas:

$$(H_l^{(k)}): \begin{cases} x_1' = \dfrac{kl_1^2}{l_1^2 + l_2^2} x_1 + \dfrac{(k-1)l_1 l_3}{l_1^2 + l_2^2} x_2 + \dfrac{(k-1)l_1 l_3}{l_1^2 + l_2^2} \\ x_2' = \dfrac{(k-1)l_1 l_2}{l_1^2 + l_2^2} x_1 + \dfrac{l_1^2 1 + kl_2^2}{l_1^2 + l_2^2} x_2 + \dfrac{(k-1)l_2 l_3}{l_1^2 + l_2^2} \end{cases} \tag{12.4}$$

Therefore, in the case of general line equation (11) we obtain the coordinate expression of the axis homotetia for general line equation:

13. Curves of the second order: representation by matrix and invariants. Reduction of the second-order curve to the canonical form. Classification of second-order curves

Curves of the second order

General *curve of the second order* on the plane π with system of normal (x, y)-coordinates is geometric set of points of the plane which coordinates satisfy the equation:

$$L(x, y) = Ax^2 + 2Bxy + Cy^2 + 2Dx + 2Ey + F = 0 \quad (A^2 + B^2 + C^2 \neq 0) \tag{13.1}$$

Note. Conditions $A^2 + B^2 + C^2 \neq 0$ is equivalent to being in (13.1) members of second order relative variables x, y.

Note. The curve of the second order often defined as the algebraic curve that in some affine coordinate system has the form (13.1). This definition, like the previous one, is correct so that the change affine coordinate system to another affine coordinate system by the formulas:

$$\begin{cases} x' = a_{11}x + a_{12}y + b_1 \\ y' = a_{21}x + a_{22}y + b_2 \end{cases}, \qquad A := \begin{Vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{Vmatrix}, \qquad \det A \neq 0 \tag{13.2}$$

the degree and type of equation (13.1) does not change.

The curves of the second order are also called *conic cross-sections* because historically they were concidered as sections of conical surfaces, in fact, direct circular cone. For example,

1. C*ircle* (or point - degenerate circle) is a cross-section of direct circular cone by plane that is perpendicular to the axis of the cone.

2. *Ellipse* (or point - degenerate ellipse) is a cross-section of direct circular cone by plane, sloped to the axis of the cone on angle over than angle between the cone axis and generatrix of cone but less than 90 degrees.

3. *Parabola* is a cross-section of direct circular cone by plane, which is parallel to some generatrix of the cone, thus generatrix not belong to this plane (if the generatrix of cone belongs to the plane, we have the case of two lines that degenerate in a one line). Thus the angle between plane and the axis of the cone is equal to the angle between the axis of the cone and its generatrix.

4. *Hyperbole* (or two lines, intersecting) is a cross-section of direct circular cone by plane, sloped to the axis of the cone on angle less than angle between the cone axis and generatrix of cone.

Symmetric matrix

$$L = \left\| l_{jk} \right\| := \begin{bmatrix} A & B & D \\ B & C & E \\ D & E & F \end{bmatrix} \tag{13.3}$$

completely determines the curve (13.1) and corresponds homogeneous quadratic form of the second order relative variables $(\mathrm{x}, \mathrm{y}, \mathrm{z})$.

$$L(x,y,z) = Ax^2 + 2Bxy + Cy^2 + 2Dxz + 2Eyz + Fz^2 =$$
$$= \begin{bmatrix} x & y & z \end{bmatrix} \cdot \begin{bmatrix} A & B & D \\ B & C & E \\ D & E & F \end{bmatrix} \cdot \begin{bmatrix} x & y & z \end{bmatrix}^T = \begin{bmatrix} x & y & z \end{bmatrix} \cdot L \cdot \begin{bmatrix} x & y & z \end{bmatrix}^T \tag{13.4}$$

But the latter form defines a conical surface $\mathrm{L}(\mathrm{x}, \mathrm{y}, \mathrm{z}) = 0$ in space Π with coordinates $\{x, y, z\}$. So the curve (l) can be regarded as (spatial) projection on a plane $z = 0$ parallel to the axis *OZ* (spatial) curve *C*, which is obtained at the intersection of the conical surface $\mathrm{L}(\mathrm{x}, \mathrm{y}, \mathrm{z}) = 0$ with plane $z = 1$.

The equation

$$L(x,y,z) = Ax^2 + 2Bxy + Cy^2 + 2Dxz + 2Eyz + Fz^2 = \begin{bmatrix} x & y & z \end{bmatrix} \cdot L \cdot \begin{bmatrix} x & y & z \end{bmatrix}^T = 0 \tag{13.5}$$

can be seen as a second-order curve equation in homogeneous coordinates $\begin{bmatrix} \mathrm{x} & \mathrm{y} & \mathrm{z} \end{bmatrix}$. Therefore, the matrix α will be called the matrix of the quadratic form (13.l) or matrix of curve (l) or matrix of homogeneous curve.

Invariants of curve of the second order

The *invariant* of second order curve (relative to change of normal coordinates) is a function $f(A,B,C,D,E,F)$ of the coefficients A,B,C,D,E,F of equation (1) of this curve, which does not depend upon change of the normal coordinates on the plane π.

For the equation (13.1) or equation (13.5) (or matrix α) let's denote:

$$S := A+C, \widetilde{S} := A+C+F = S+F, \delta = \Delta_F := \begin{vmatrix} A & B \\ B & C \end{vmatrix}, \tag{13.6}$$

$$\Delta := \begin{vmatrix} A & B & D \\ B & C & E \\ D & E & F \end{vmatrix}; \qquad K := \Delta_A + \Delta_C = \begin{vmatrix} C & E \\ E & F \end{vmatrix} + \begin{vmatrix} A & D \\ D & F \end{vmatrix}, \tag{13.7}$$

$$\widetilde{K} := \Delta_A + \Delta_C + \Delta_F = \begin{vmatrix} A & B \\ B & C \end{vmatrix} + \begin{vmatrix} C & E \\ E & F \end{vmatrix} + \begin{vmatrix} A & D \\ D & F \end{vmatrix} = \delta + K. \tag{13.8}$$

Here is used the following notation: Δ_U - is (nonalgebraic) minor of element U of matrix α, it is the determinant that we get from the determinant of matrix α, expunging the row and column containing the element U.

The values $S := A+C$ *and* $\delta := \begin{vmatrix} A & B \\ B & C \end{vmatrix}$ does not depend on changes in normal coordinates in equation (13.1). Specifically, this values are invariant relatively to change of normal coordinates .

The values $S := A+C, \quad \delta := \begin{vmatrix} A & B \\ B & C \end{vmatrix} = AC - B^2$ and $\Delta := \begin{vmatrix} A & B & D \\ B & C & E \\ C & E & F \end{vmatrix}$ are called respectively the invariants of the equation (13.1) *of* first, second and third order.

The value $K := \begin{vmatrix} C & E \\ E & F \end{vmatrix} + \begin{vmatrix} A & D \\ D & F \end{vmatrix} = \overline{K} - \delta$ is called semiinvariant of equation (13.1).

Reducing of curve equation to the canonical form

The main problems of the theory of second-order curves are,

First, the classification of these curves (in terms of matrix L) and secondly, the adductionof the equation of any such curve to the canonical form, more precisely, obtaining change of coordinates at which the curve of the second order equation takes the simplest (canonical) form.

But our main goal - is to use second-order curves in the practical output to the screen, and we had little formal grading curves of the second order, that is, the knowledge that there affine coordinate system, which has, for example, an ellipse. We additionally need to find how to calculate the matrix of change of coordinates that performs construction curve equation to canonical form to be able to carry out the withdrawal of the curve in the terminal window computer.

Therefore, we will implement the dual approach using as a representation of the curve in the form of (13.1) and the presentation of the curve (13.4) using affine change of coordinates (ie homogeneous coordinates transformation matrix). In this case, we will need the following statement.

Statement. Let $\{x, y, z\}$ - homogeneous affine coordinate system on the plane and $\{x', y', z'\}$ - other homogeneous affine coordinate system on the same plane. Let M - corresponding matrix change of homogeneous coordinates, i.e.,

$$[x'\ y'\ z'] = [x\ y\ z]\cdot M \quad \text{or} \quad [x\ y\ z] = [x'\ y'\ z']\cdot M^{-1} \qquad (13.9)$$

Then the curve equation (5) transformes into the equation

$$L'(x'\ y'\ z') = A'x'^2 + 2B'x'\ y' + C'y'^2 + 2D'x'\ z' + 2E'y'\ z' + F'z'^2 = [x'\ y'\ z']\cdot C\cdot [x'\ y'\ z']^T = 0, \qquad (13.10)$$

Where

$$L' = M^{-1}L\left(M^{-1}\right)^T = M^{-1}L\left(M^T\right)^{-1} \qquad (13.11)$$

i.e. the matrix of curve at changing of the affine coordinates (13.9) is calculated using the formula (13.11).

1. The matrix of parallel transfer of coordinates

$$M_{sh} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m & n & 1 \end{bmatrix} \qquad \left(\{m,n\} \in R^2\right) \tag{13.12}$$

with change of coordinates $\{\bar{x} = x + m; \bar{y} = y + n\}$ $(\text{при } \{m,n\} = \{0,0\})$ get the change $\{\bar{x} = x; \bar{y} = y\}$ that actually is redefinition of coordinates.

2. The matrix of rotation around the origin

$$M_{rot} = \begin{bmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{13.13}$$

by the angle $\alpha = arctg\frac{s}{c}(c,s \geq 0 \text{ и } c^2 + s^2 = 1)$ with change of coordinates $\{\bar{x} = cx - sy; \bar{y} = sx + cy\}$. This matrix is always orthogonal, that means that its transposed coincides with the inverse: $M^T = M^{-1}$

Parallel shift of affine coordinate system

The formulas for change of coordinates, which we call parallel shift of affine coordinates on the plane π, defined as (here $\{x,y\}$) ($\{x,y\}$ - coordinates of the point X in the original affine coordinate system; $\{\bar{x},\bar{y}\}$ - coordinates of the same point X in the new affine coordinate system):

direct change: $\begin{cases} \bar{x} = x + m \\ \bar{y} = y + n \end{cases}$; $\qquad M_{sh} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m & n & 1 \end{bmatrix}$; (13.14)

inverse change: $\begin{cases} x = \bar{x} + m \\ y = \bar{y} + n \end{cases}$; $\qquad M_{sh}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -m & -n & 1 \end{bmatrix}$ (13.15)

At the change of the coordinates (13.14), curve equation (1) takes the form:

$$\bar{L}(\bar{x},\bar{y}) := \bar{A}\,\bar{x}^2 + 2\bar{B}\bar{x}\bar{y} + \bar{C}\bar{y}^2 + 2\bar{D}\bar{x} + 2\bar{E}\bar{y} + \bar{F} = 0, \quad (\bar{A}^2 + \bar{B}^2 + \bar{C}^2 \neq 0), \tag{13.16}$$

where

$$\begin{cases} \overline{A}=A,\overline{B}=B,\overline{C}=C, \\ \overline{D}=-Am-Bn+D=\dfrac{1}{2}L'_x(-m,-n), \\ \overline{E}=-Bn-Cn+E=\dfrac{1}{2}L'_y(-m,-n), \\ \overline{F}=Am^2+2Bmn+Cn^2-2Dm-2En+F=L(-m;-n). \end{cases} \tag{13.17}$$

The following statement highlights the value $\delta = \Delta_F$ in the reducing of a general second-order curve equation to canonical form.

Statement. 1. At $\delta = \Delta_F \neq 0$ ($AC \neq B^2$) there is always a change of coordinates (13.15) which reduces (13.1) to the form in which there are no first degrees variables x and y (i.e. $\overline{D} = \overline{E} = 0$). Wherein:

$$m = -\frac{\Delta_D}{\Delta_F}, \qquad n = \frac{\Delta_E}{\Delta_F} \tag{13.18}$$

The point $M(x_0, y_0)$ with coordinates

$$\{x_0, y_0\} = \{-m, -n\} = \left\{\frac{\Delta_D}{\Delta_F}, \frac{\Delta_E}{\Delta_F}\right\} \tag{13.19}$$

is the center of symmetry of the curve (in homogeneous coordinates $\lfloor x_0 : y_0 : 1] \equiv \lfloor \Delta_D : -\Delta_E : \Delta_F \rfloor$). Wherein the curve equation takes the form:

$$Ax^2 + 2Bxy + Cy^2 + \frac{\Delta}{\delta} = 0 \tag{13.20}$$

2. At $\delta = \Delta_F = 0$ (i.e. $AC = B^2$) change of coordinates (13.15) which reduces (13.1) to the form in which there are no first degrees variables x and y, exists if and only if the conditions:

$$[\Delta_D = 0,\ \Delta_E = 0,\ \Delta_F = 0] \Leftrightarrow \left[CD=BE,\ AE=BD,\ AC=B^2\right] \tag{13.21}$$

This is the condition of proportionality of the first two rows of matrix L. Wherein is necessarily: $\Delta = 0$.

Rotation of affine coordinate system

Let's introduce rotation around the origin *{x, y}* of coordinates, i.e. around the point *{x, y} = {0,0}*, as the change of *{x,y}*- coordinates on $\{\bar{x}, \bar{y}\}$ - coordinates by the formulas (*{x,y}* – are the coordinates of point X before rotation, $\{\bar{x}, \bar{y}\}$ - coordinates of point $\bar{X}$, which corresponds point X after rotation):

direct change: $$\begin{cases} \bar{x} = cx + sy, \quad c^2 + s^2 = 1 \\ \bar{y} = -sx + cy, \quad c > 0,\ s > 0 \end{cases} \tag{13.22}$$

inverse change: $$\begin{cases} x = c\bar{x} - s\bar{y}, \quad c^2 + s^2 = 1 \\ y = s\bar{x} + c\bar{y}, \quad c > 0,\ s > 0 \end{cases} \tag{13.23}$$

Of course, the parameters *c* and *s* - are respectively *cos* and *sin* of some angle of rotation $\varphi \in (0, \frac{\pi}{2})$. In this case, this rotation translates perpendicular axis *x* and *y* to the perpendicular axis $\bar{x}$ and $\bar{y}$ represents the rotation by angle (radian) toward the opposite direction clockwise, i.e. (13.22) and (13.23) can be written as:

direct change: $$\begin{cases} \bar{x} = \cos\varphi\, x + \sin\varphi\, y, \\ \bar{y} = -\sin\varphi\, x + \cos\varphi\, y \end{cases} \tag{13.24}$$

inverse change: $$\begin{cases} \bar{x} = \cos\varphi\, \bar{x} - \sin\varphi\, \bar{y}, \\ \bar{y} = \sin\varphi\, \bar{x} + \cos\varphi\, \bar{y} \end{cases} \tag{13.25}$$

At $B \neq 0$ change of coordinates, at which the product of variables in the curve equation (13.1) disappears is determined by changing coordinates (13.22), where:

$$c = \sqrt{\frac{1}{2}\left(1 + \frac{(A-C)signB}{\sqrt{d}}\right)},\ s = \sqrt{\frac{1}{2}\left(1 - \frac{(A-C)signB}{\sqrt{d}}\right)}, \tag{13.26}$$

Here $d=(A-C)^2+4B^2=S^2-4\delta$. If we put $c=\cos\varphi, b=\sin\varphi$ (i.e. choose the change of coordinates in the form (13.24)), then the angle φ can be obtained by formula:

$$\varphi = arctg\frac{2|B|}{(A-C)signB+\sqrt{d}}, \qquad (13.27)$$

or by formula (at $A\neq C$):

$$\varphi=\frac{1}{2}arctg\frac{2B}{A-C}, \quad A\neq C.$$

Reduction of curve equation to the canonical form

Reduction of curve equation to the canonical form will be carried out in several steps.

Step 1. As has been proved, exists a change of homogeneous coordinates, which combines the center of coordinates with the center of the curve $\{\xi',\eta',\zeta'\}=\{\xi,\eta,\zeta\}M_1$ with matrix

$$M_1=\begin{Vmatrix} 1 & 0 & 0\\ 0 & 1 & 0\\ -\frac{\Delta_D}{\delta} & \frac{\Delta_E}{\delta} & 1\end{Vmatrix}\equiv\begin{Vmatrix} \Delta_F & 0 & 0\\ 0 & \Delta_F & 0\\ -\Delta_D & \Delta_E & \Delta_F\end{Vmatrix}, \qquad (13.28)$$

for which the curve equation in homogeneous coordinates takes the form:

$$A\xi'^2+2B\xi'\eta'+C\eta'^2+\frac{\Delta}{\delta}\zeta'^2=0, \qquad (13.29)$$

or in ordinary coordinates:

$$Ax'^2+2Bx'y'+Cy'^2+\frac{\Delta}{\delta}=0. \qquad (13.30)$$

Step 2. Construct the change of homogeneous coordinate $\{\bar{\xi},\bar{\eta},\bar{\zeta}\}=\{\xi',\eta',\zeta'\}\cdot M_2$, i.e. matrix M_2 by the following rule:

1. At $B=0$ we put $M_2:=\begin{Vmatrix}1&0&0\\0&1&0\\0&0&1\end{Vmatrix}$.

2. At $B\neq 0$ we put

$$M_2 := \begin{Bmatrix} \sqrt{\frac{1}{2}\left(1+\frac{(A-C)signB}{\sqrt{d}}\right)} & -\sqrt{\frac{1}{2}\left(1-\frac{(A-C)signB}{\sqrt{d}}\right)} & 0 \\ -\sqrt{\frac{1}{2}\left(1-\frac{(A-C)signB}{\sqrt{d}}\right)} & \sqrt{\frac{1}{2}\left(1+\frac{(A-C)signB}{\sqrt{d}}\right)} & 0 \\ 0 & 0 & 1 \end{Bmatrix}, \qquad (13.31)$$

where $d = (A-B)^2 + B^2 = S^2 - 4\delta$.

Thus, at the transformation of homogeneous coordinates $\{\bar{\xi},\bar{\eta},\bar{\zeta}\} = \{\xi,\eta,\zeta\}\cdot M$ with matrix $M = M_1 \cdot M_2$ it will be always $\bar{B} = 0$ in the curve equation, i.e. we obtain the equation of the curve:

$$\bar{A}\bar{x}^2 + \bar{C}\bar{y}^2 + \bar{F} = 0, \qquad (13.32)$$

or in homogeneous coordinates:

$$\bar{A}\bar{\xi}^2 + \bar{C}\bar{\eta}^2 + \frac{\Delta}{\delta}\zeta^2 = 0. \qquad (13.33)$$

In the case when $\delta = 0$ **,** $B \neq 0$ the change of coordinates (rotation around the origin of coordinates with parameters *c* and *s*, which are obtained by formulas (13.26), the equation (13.1) changes by formulas (13.32) or (13.33), where $\bar{B} = 0$. Wherein, the invariants of the curve are not changed. So, in new coordinates $\{\bar{x},\bar{y}\}$ we receive the previous case $\bar{\delta} = \delta$ *i* $\bar{B} = 0$.

In the case when $\delta = 0$ always will be $S \neq 0$. Really, at $S = 0$ we have $\begin{cases} AC - B^2 > 0 \\ A + C = 0 \end{cases} \rightarrow \begin{cases} C = -A \\ -A^2 - B^2 = 0 \end{cases} \rightarrow A = B = C = 0$, which is impossible. So, $S \neq 0$ at $\delta > 0$. But then there are also equivalence $\frac{\Delta}{S} > 0 \leftrightarrow \Delta S > 0$, $\frac{\Delta}{S} = 0 \leftrightarrow \Delta S = 0$, $\frac{\Delta}{S} < 0 \leftrightarrow \Delta S < 0$.

14. Output of second-order curves on display. Method of cross-section. Iterative algorithms displaying the curves of the second order

Construction of iterative scheme

Let's consider algorithms display the curves of the second order by the example of hyperbole. To implement iterative algorithm we will build the iterative scheme. For this let's parametrize the curve:

$$\begin{cases} x = a \cdot \cos\varphi \\ y = b \cdot \sin\varphi \end{cases}$$

where φ - is parameter, angle.

Next we need the following formulas for hyperbolic functions of sum of arguments:

$$sh(x+y) = shx \cdot chy + chx \cdot shy,$$
$$ch(x+y) = chx \cdot chy + shx \cdot shy.$$

Let's transform in recurrence relations:

$$\varphi_{k+1} = \varphi_0 + (k+1)h = \varphi_0 + kh + h = \varphi_k + h,$$
$$u_{k+1} = \pm ch\varphi_{k+1} = \pm ch(\varphi_k + h) = \pm ch\varphi_k \cdot ch\,h \pm sh\varphi_k \cdot sh\,h = ch\,h \cdot u_k + sh\,h \cdot u_k,$$
$$v_{k+1} = \pm sh\varphi_{k+1} = \pm sh(\varphi_k + h) = \pm sh\varphi_k \cdot ch\,h \pm ch\varphi_k \cdot sh\,h = sh\,h \cdot u_k + ch\,h \cdot v_k.$$

That is, we have the following iterative scheme

$$\begin{cases} [u_0, v_0, 1] = [\pm 1, 0, 1], \\ [u_{k+1}, v_{k+1}, 1] = [u_k, v_k, 1] H_h \end{cases}, \qquad H_h := \begin{bmatrix} ch\,h & sh\,h & 0 \\ sh\,h & ch\,h & 0 \\ 0 & 0 & 1 \end{bmatrix}. \qquad (14.1)$$

Wherein $\det H_h = ch^2 h - sh^2 h = 1$;

H_h - hyperbolic rotation matrix.

(Note, that similar matrix can be found for an ellipse:

$$E_h := \begin{bmatrix} \cos h & \sin h & 0 \\ -\sin h & \cos h & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (14.2)$$

- elliptical rotation matrix.)

If equation (1) of the curve of second order by change of coordinates $[x, y,1] = [u, v,1] \cdot M$ reduces to equation in canonical form, then the coordinates of vertices of the polyline which approximates the curve in $\{x, y\}$ - coordinates will be equal $[x_k, y_k,1] = [u_k, v_k,1] \cdot M$. Hence we obtain the following iterative scheme for the coordinates of the vertices:

$$\begin{cases} [x_0, y_0,1] := [1,0,1] \cdot M, \\ [x_{k+1}, y_{k+1},1] := [u_{k+1}, v_{k+1},1] \cdot M = [u_k, v_k,1] \cdot H_h \cdot M = [x_k, y_k,1] \cdot M_h, \\ M_h := M^{-1} \cdot H_h \cdot M \end{cases}$$

Wherein $\det M_h = \det M^{-1} \det H_h \det M = 1$, i.e. change of coordinates with matrix M_h preserves the area of figures.

Matrix M_h can be calculated prior to the cycle of calculation of sequence of points $\{x_k, y_k\}$, since it does not depend on k.

In the case of hyperbole that is not degenerate $(\delta < 0, \Delta \neq 0)$ matrix M_h determines the hyperbolic transformation of plane which is defined by hyperbole, and it is the hyperbolic rotation of plane.

Iterative algorithm for displaying the arc

of hyperbole (ellipse)

1) Entrance:

- Taking coefficients A, B, C, D, E, F of curve equation (13.1).
- Finding and invariants δ and Δ (see formula (13.6) and (13.7)). Test conditions $\delta < 0,\ \Delta \neq 0$ (non-degenerate hyperbole). If these conditions are not met, then go to Exit.
- Finding the changes of coordinates with matrix M that is not degenerate, of size 3×3, at which equation (13.1) is reduced to canonical form. Matrix M can be found as the product of matrix M_1 from formula (13.28) and matrix M_2 from formula (13.31).

- Obtaining values of the initial φ_n and final φ_k angles of parameter φ $(\varphi_n < \varphi_k)$.
- Getting of the number of sides (or vertices) $m > 2$ of polyline by which we approximate the desired curve.
- Initial installation (before arithmetic cycle):

  - Count $h = \dfrac{\varphi_n - \varphi_k}{m} > 0$.

  - Count matrix H_h by the formula (34) (in the case of an ellipse - corresponding formula (13.33)) and then the matrix $M_h = M^{-1} H_h M$.

  - Count the coordinates $[x_0, y_0, 1] = [-ch\varphi_n, -sh\varphi_n, 1] \cdot M$ of the initial point A_0.

  2) - The beginning of the arithmetic cycle from $n = 0$ to $n = m$ (from the beginning of the cycle $n := 0$, when the current cycle $\boldsymbol{n := n + 1}$). Further test of conditions $\boldsymbol{n < m}$, if performed, so - go to step 1, if not - go to Exit. - Calculation of the coordinates $\{x_{n+1}, y_{n+1}, 1\} = \{x_n, y_n, 1\} \cdot M_h$ of the current point A_{n+1}.

  - Display of the interval $[A_n, A_{n+1}]$ in the window.

  3) The end of the arithmetic cycle by n (transition at the Beginning of the arithmetic cycle).

  4) Exit.

Iterative algorithm for displaying parabola

Let $N > 0$ - natural numbers describing the number of vertices (approximately $K + N$) approximating polyline. Let's put:

$$\begin{cases} t_k = T + k \cdot h, \\ u_k = t_k^2, \\ v_k = t_k \end{cases} \qquad k = 0,1,..., N$$

Transform in recurrence relations:

$$t_{k+1} = T + (k+1) \cdot h = T + k \cdot h + h = t_k + h,$$

$$u_{k+1} = t_{k+1}^2 = (t_k + 2ht_k + h^2 = u_k + 2h \cdot u_k + h^2),$$

$$v_{k+1} = t_{k+1} = t_k + h = v_k + h,$$

That is, in the homogeneous coordinates we have the desired recurrence relations:

$$\begin{cases} [u_0, v_0, 1] = [T^2, T, 1], \\ [u_{n+1}, v_{n+1}, 1] = [u_n, v_n, 1] \cdot P_h \end{cases}$$

where $P_h = \begin{bmatrix} 1 & 0 & 0 \\ 2h & 1 & 0 \\ h^2 & h & 1 \end{bmatrix}$ - матриця параболічного обертання.

Wherein $\det P_h = 1$.

We have the following iterative scheme:

$$\begin{cases} [x_0, y_0, 1] = [T^2, T, 1] \cdot M, \\ [x_{n+1}, y_{n+1}, 1] = [x_n, y_n, 1] \cdot M_h \end{cases}$$

where $M_h = M^{-1} P_h M$.

Matrix M_h defines the parabolic transformation of the plane. As in previous cases, this matrix can be calculated before the main calculating cycle.

Thus, we obtain an iterative algorithm for displaying the arc of a parabola

1) Entrance:

- Taking the value p - the parabola parameter $(p > 0)$.
- Taking the value $h > 0$.
- Taking the values of coordinates $x_0 = T, y_0 = T^2$ of the initial (start) point A_0.
- Taking the number $m > 2$, which is the number of the parties (or vertices) of polyline by which we approximate the desired curve.

2) Initial installation (before arithmetic cycle):

- Calculation of matrix M_h (similar to the case of the ellipse or hyperbola).

3) The beginning of the arithmetic cycle from $n = 0$ to $n = m - 1$ (When entering in the cycle $n = 0$. In the cycle $n := n + 1$). Further test of conditions $n < m$, if performed, so - go to step 1, if not - go to Exit.

- Calculation of the coordinates $\{x_{n+1}, y_{n+1}, 1\} = \{x_n, y_n, 1\} \cdot M_h$ of the current point A_{n+1}.

- Display of the interval $[A_n, A_{n+1}]$ in the window

- The end of the arithmetic cycle by n (transition at the Beginning of the arithmetic cycle).

4) Exit.

Method of cross-sections

Preliminary considerations indicate how to implement output curve of the second order in the shortest time and in the most rational way. But often there are cases when the output curve of the second order on the screen (window) terminal could spend a lot of time, that is, the time performance of output curve is not critical. In this case, after finding the type of curve by invariants method, for displaying an ellipse, parabola or hyperbole can be used the cross-sections method.

If we know the mathematical boundaries of window (x_n - left, x_n - right, y_n - lower, y_n - upper) and the window size $M_x \times M_y$ (pixels), the displaying of curve (1) on the screen can be made by the individual pixels.

I. The case $A \neq 0,\ C \neq 0$.

In this case, equation (1) has the general form. So we get the following algorithm:

1) Calculate step $\Delta x = (x_n - x_n) / M_x$.

2) The beginning of the first arithmetic cycle from $j = 0$ to $j = M_x - 1$.

3) Calculate the value $x_j = x_n + (j-1)\Delta x$.

4) Consider the equation (1), as equation relatively y if $x = x_j$ (cross-section parallel to axis y), i.e. the equation $Cy^2 + 2(Bx_j + E)y + Ax_j^2 + 2Dx_j + F = 0$ and find the discriminant $D = (Bx_j + E)^2 - C(Ax^2) + 2Dx_j + F)$.

5) At $D \geq 0$ calculate y_1 and y_2 by the formulas for the roots of quadratic equation:

$$y_1 = \frac{-(Bx_j + E) - \sqrt{D}}{C}, \qquad y_2 = \frac{-(Bx_j + E) + \sqrt{D}}{C}$$

and carry out displaying to the screen of pixels with coordinates $\{x_j, y_1\}$ and $\{x_j, y_2\}$.

6) The end of the first arithmetic cycle.

7) Calculate step $\Delta y = (y_в - y_н)/M_y$.

8) The beginning of the second arithmetic cycle from $j = 1$ to $j = M_y - 1$

9) Calculate values $y_j = y_н + (j-1)\Delta y$.

10) Consider the equation (1), as equation relatively x if $y = y_j$ (cross-section parallel to axis x), i.e. the equation $Ax^2 + 2(By_j + D)x + Cy_j^2 + 2Ey_j + F = 0$ and find the discriminant $D = (By_j + D)^2 - A(Cy_j^2 + 2Ex_j + F)$.

11) At $D \geq 0$ calculate x_1 and x_2 by the formulas for the roots of quadratic equation:

$$x_1 = \frac{-(By_j + D) - \sqrt{D}}{A}, \qquad x_2 = \frac{-(By_j + D) + \sqrt{D}}{A}$$

12) The end of the second arithmetic cycle.

13) The end of the algorithm.

II. **The case** $A = 0, C \neq 0$.

In this case the equation (1) takes the form $2(By + D)x + Cy^2 + 2Ey + F = 0$. So we get the following algorithm:

1) Calculate step $\Delta y = (y_в - y_н)/M_y$.

2) The beginning of the first arithmetic cycle from $j = 0$ to $j = M_y - 1$

3) Calculate values $y_j = y_н + (j-1)\Delta y$

4) At $By_j + D \neq 0$ calculate $x = \dfrac{Cy_j^2 + 2Ey_j + F}{2(By_j + D)}$ and carry out displaying to the screen of pixels with coordinates $\{x, y_j\}$

5) The end of the first arithmetic cycle.

6) Calculate step $\Delta x = (x_п - x_л)/M_x$

7) The beginning of the second arithmetic cycle from $j = 0$ to $j = M_x - 1$

8) Calculate values $x_j = x_л + (j-1)\Delta x$

9) Consider the equation (1), as equation relatively y if $x = x_j$ (cross-section parallel to axis y), тобто рівняння $Cy^2 + 2(Bx_j + E)y + 2Dx_j + F = 0$ і знаходимо дискримінант $D = (Bx_j + E)^2 - C(2Dx_j + F)$. i.e. the equation $Cy^2 + 2(Bx_j + E)y + Ax_j^2 + 2Dx_j + F = 0$ and find the discriminant $D = (Bx_j + E)^2 - C(Ax^2) + 2Dx_j + F)$.

10) At $D \geq 0$ calculate y_1 and y_2 by the formulas for the roots of quadratic equation:

$$y_1 = \frac{-(Bx_j + E) - \sqrt{D}}{C}, \qquad y_2 = \frac{-(Bx_j + E) + \sqrt{D}}{C}$$

and carry out displaying to the screen of pixels with coordinates $\{x_j, y_1\}$ and $\{x_j, y_2\}$.

11) The end of the first arithmetic cycle.

12) Calculate step $\Delta y = (y_в - y_н)/M_y$.

13) The beginning of the second arithmetic cycle from $j = 1$ to $j = M_y - 1$

14) Calculate values $y_j = y_н + (j-1)\Delta y$.

15) Consider the equation (1), as equation relatively x if $y = y_j$ (cross-section parallel to axis x), i.e. the equation $Ax^2 + 2(By_j + D)x + Cy_j^2 + 2Ey_j + F = 0$ and find the discriminant $D = (By_j + D)^2 - A(Cy_j^2 + 2Ex_j + F)$.

16) At $D \geq 0$ calculate x_1 and x_2 by the formulas for the roots of quadratic equation:

$$x_1 = \frac{-(By_j + D) - \sqrt{D}}{A}, \quad x_2 = \frac{-(By_j + D) + \sqrt{D}}{A}$$

17) The end of the second arithmetic cycle.

18) The end of the algorithm.

So, in cases when the output curve of the second order on the screen (window) of terminal could spend a lot of time, that is, the time performance of output curve is not critical, we carried out output the curve. In this case, after finding the type of curve by invariants method, for displaying an ellipse, parabola or hyperbole can be used the cross-sections method.

A number of important industrial and economic problems (not just light industry) naturally united not so much the content as methods for their solution. As a result of studying "Numerical Methods" we knew the application of mathematical methods for solving complex problems using modern computers.

PART II. PRACTICAL APPLICATION AND SOFTWARE

1. Mathematical modelling of dispersed phase drop deformation in nano-filled polyner mixture melts

Key provisions

The purpose was to study using mathematical modeling method of the influence of nano-additive on dispersed phase component drop deformation during polymer dispersion melt flow in the entry area of forming hole.

To study the process of drop deformation in a polymer dispersion the mathematical model developed on the standpoint of structural-continual approach was improved. The model takes into account the main provisions of classical fluid mechanics and changes in the structure of the dispersed phase during its flowing.

It is shown that the modified mathematical model of deformation of the polymer dispersed phase drop adequately describes the process of structureformation during real nano-filled polymer compositions flowing. The values of polypropylene (PP) drops deformation, calculated using the model, correlate the experimental results: inter-phase tension reduce leads to drops in deformation increase and to the average diameter of PP microfibers reduction.

The mathematical model of deformation of dispersed phase polymer drop was improved in order to carry out for theoretical research of nano-filled polymer mixtures.

Using the developed mathematical model will accelerate researches and reduce material and energy costs of them.

Introduction

One promising way modification of polymers and their blends are creating nanocomposites, in which a set of desired properties is achieved through the optimal combination of components. The use of fillers of different sizes, shapes and chemical nature allows to improve mechanical properties of

materials and provide them with new functional characteristics (incombustibility, bactericidal, conductivity, sorption capacity, etc.). Herewith essential is the ability of nanoparticles (NP) surface be getting wet by polymer and the nature and degree of interaction between the NP and macromolecules polymer on the interphase [1,2]. It is shown that the introduction of silica nanoparticles in a mixture melt of polypropylene / copoliamide (PP / SPA) allows to adjust the processes of structureformation of PP in the SPA matrix and thus improve the structure of the filter material (FM), obtained in processing of the said mixture. These filters combine high cleaning efficiency and productivity, and the presence of nanofiller in the FM structure provides them bactericidal properties[2]. To create new nanomaterials and regulation of their properties is necessary conducting basic research and the establishment of appropriate laws.

Problem

Polymers are generally thermodynamically incompatible with each other in the melt, but the section on individual phases prevents high viscosity of the components. Shear flow contributes to the formation of different types of structures by the component of dispersed phase: liquid cylinders (jets), layers, drops, etc To describe the rheological behavior of polymer dispersion melts are used the laws of classical mechanics, same as for modeling systems such as suspensions and emulsions [3]. At the same time a polymer mixture is a special class of colloidal dispersions of the "polymer in the polymer." An important difference is formation between the two its components interphase transition layer whose properties are very different from those of the characteristics of polymer melt in volume. In nano-filled polymer melts an interphase layer around the nanoparticles is formed as well at the interphase filler / polymer and its thickness ranges (0,0004 ÷ 0,16) mm [4]. Thus, depending on the degree of affinity between the polymer and additive nanoparticles can be localized in the bulk melt or at the interface and influence the magnitude of surface tension.

Purpose ─ studying by the mathematical modeling of the influence of nano-additives on deformation of dispersed phase component drop during polymer composition melt flow.

The main material

Study of flow patterns and structureformation in polymer dispersions subject of many articles and books. However, because of the complexity of such systems research experimental approaches outweigh theoretical. Today received a number of empirical regularities and mathematical models that describe with sufficient accuracy the behavior of such systems. In [5] from the standpoint of structural and continual approach developed a mathematical model that allows to determine the value of drop deformation depending on the volume concentration and the rheological properties of the components (viscosity of the dispersed phase and dispersion medium, their interrelation and flexibility). The advantage of this model is that it takes into account the main provisions of continuum mechanics (integrity protection, continuity of functions, describing its movement and state) and the particular structure of the dispersed phase. Form drops - is ellipsoid of revolution, which changes size during the interaction with its dispersion medium but retains volume. Deformation drops depending on the orientation in the flow accounted for using the tensor strain rate uniaxial tension. The model is a system of differential equations in dimensionless variables has the form:

$$\begin{cases} \dot{\varphi} = 0 \\ \dot{\theta} = -\frac{3}{4} u\lambda_3 \sin(2\theta) \\ \frac{\dot{q}}{q} = \frac{3}{2}(\lambda_1 + \frac{u}{2}(\lambda_2 r_0^2 q^{4/3} + \lambda_3)(2 - 3\sin^2\theta)) \end{cases} \quad (1.1)$$

where: φ, θ - angles that define the orientation of the drop in the stream;

u - the intensity of the current uniaxial stretching;

q - the value of deformation (stretching dimensionless);

λ1, λ2, λ3 - values that take into account the rheological characteristics of the components.

In the above equation point means complete original in time.

It is known that solid fillers cause thickening thixotropic effect, which leads to an increase of viscosity of the polymer melt. In carrying out modifications of polymer mixture melts an additive is usually pre-injected into one component. In determining the value of drop deformation using model (1) the influence of nano-additive can be taken into account due to changes in melt viscosity of the dispersed phase and dispersion medium, using Einstein's formula for dilute suspensions:

$$\eta_E = \eta_0 (1+2{,}5V) \qquad (1.2)$$

where: η0 - viscosity of the medium; V - volume concentration of suspended particles.

Experimental studies show that for compositions with a low content of nano-additive (0,05 ÷ 3,0) masses. %, the viscosity increases slightly within the error and it coincides with the effective viscosity (ηE) defined by the formula (1.2). Calculations made using the model showed that the concentration of nano-additive (0,05 ÷ 3,0) masses. % virtually no effect on the amount of strain drops of the dispersed phase. However, this is inconsistent with research on the impact of nanofillers on micro and macro-rheological processes in polymer mixture melt flowing. Thus, in [2] is shown that the introduction of (0,1 ÷ 3,0) masses. % silica in a mixture melt of polypropylene / co-poliamide improves fiberization PP in SPA matrix: an average microfibers diameter reduced and their uniformity of distribution by diameters increasing. The authors attribute this to the influence of nanoparticles on the interphase phenomena, namely with decreasing values of surface tension at the interphase.

From classic fundamental ratios that describe thermodynamic equilibrium in Low-molecular dispersion system it follows that the dispersion medium in a flow is acting on a drop dispersed phase therein with a force proportional to

the gradient of shear velocity and medium viscosity and besides this is a function of the ratio of viscosities components. A drop of polymer dispersed phase reacts on deformation with force [6]:

$$T_\gamma = 2\,\gamma_{\alpha\beta} / r \qquad (1.3)$$

where: γαβ - interphase tension; r - the radius of the drop.

At the same time, the ability to drop deformation is largely determined by its elasticity. In mathematical model (1.1) resistance of drop on its deformation is taken into account due to the value of the elastic modulus G, which is included into the relation to determine the rheological function λ1:

$$\lambda_1 = \frac{-2ab^2\beta_0^{''}G\frac{a}{a_0}(1-\frac{q}{q_0})}{\mu(2+3ab^2\beta_0^{''}\frac{\eta}{\mu})}(1-M\Phi) \qquad (1.4)$$

where: a, b, a0, b0 - ellipsoid axis in deformed and undeformed state;

G, F - modulus of elasticity and volume concentration of the dispersed phase;

μ, η - viscosity of the dispersion medium and dispersed phase;

$$M = \frac{4}{ab^2(2+3ab^2\frac{\eta}{\mu}\beta_0^{''})} * \{ \frac{5}{6(\alpha_0+2\beta_0-2\beta_0^{'}(a^2+b^2))} -$$

$$-\frac{100\beta_0^{'}a^2(2\beta_0^{'}a^2-\alpha_0-2\beta_0)}{(\alpha_0+2\beta_0)(\alpha_0+2\beta_0-2\beta_0^{'}(a^2+b^2))} * [\ \frac{1}{24a\beta_0} - \frac{1}{2\beta_0^{'2}a^2-(\alpha_0+2\beta_0)}\ ] \}$$

The values of $\alpha_0, \beta_0, a_0', \beta_0', \alpha_o'', \beta_o''$ are obtained in [3].

To assess the effect of the interfacial tension on the ability to deformation of the dispersed phase drops in the expression for the determination of rheological function λ1 were made changes based on the fact that G = Tγ. With the balance of the elastic power inside (G) and resistance (Tγ) equation to determine λ1 will look like:

$$\lambda_1 = \frac{-2ab^2\beta_0^{''}\frac{\gamma_{\alpha\beta}}{R_0}\frac{a}{a_0}q^{2/3}}{\mu(2+3ab^2\beta_0^{''}\frac{\eta}{\mu})}(1-M\Phi) \qquad (1.5)$$

where: a_0 - ellipsoid axis, which volume is equivalent to the volume of sphere drop with radius r.

The system of differential equations (1) was solved numerically by the Runge-Kutta method using specially written program in Delphi environment with Object Pascal language. Modified model tested for adequacy, ie the ability to predict the results of research in some area with the required accuracy by comparing the amount of strain drops obtained when using it with experimental data. This was used in the investigation results of about 1.0. methyl silica % additive (MC) on the value of interfacial tension (γαβ) and average diameter jets (micro) mixtures PE / spa and polypropylene / polyvinyl alcohol (PE / PVA) of the 30.6 / 68.4 vol. % (Table).

Table. The dependence of the deformation of the dispersed phase drops on the value of interphase tension

Mixture	$\gamma_{\alpha\beta,}$ мН/м$^{-1}$	$\bar{d}$ мкм	q
PP / SPA	2,60	4,0	125
PP / SPA / MS	0,75	2,6	620
PP / PVA	0,73	3,5	273
PP / PVA / MS	0,47	1,7	531

The table shows that the values of interfacial tension obtained by using the theory of fracture liquid cylinder for nano-filled compositions are much lower compared to the initial mixture. This results in reduction of energy consumption in the formation of new surfaces dispersed phase, that promotes the dispersion and deformation of the droplets in the matrix polymer PP, PP microfiber average diameter lower than in the initial mixture of (1,5 ÷ 2,1)

times. Improved model actually describes the process of deformation of a PP drop in matrix: γαβ reduction in nano-filled mixtures is accompanied by increasing values of deformation. The results produced by the model are in good agreement with the experimental data on the influence of nano-filler on processes of structure -formation. Introduction filler reduces the average diameter polypropylene microfibers by reducing the surface tension at the interface.

Software that implements the described algorithm has been developed [26, 29, 32]. The text of the main program procedures is given in the appendix 14.

Conclusion

It is shown that the improvement of previously established mathematical model of deformation of drop dispersed phase polymer in a of polymer mixture melt flow in the entry area forming hole can expand its capabilities and to use it to predict droplets deformation of component dispersed phase in nano-filled mixtures. Found that the modified model includes the effect of Nano-additive on droplet deformation in the terms of interphase tension at the interphase of the components.

2.Planning the experiment and optimization of the content of nanoadition in polypropylene monothreads

Key provisions

The purpose was planning the experiment and optimization of the content of the composition Polypropylene\ binary nanoaddition in order to obtain Polypropylene monothreads with high mechanical and antibacterial properties.

For planning the experiment the simplex-grid method has been used in pseudo coordinates . The optimization of the content of the nanoaddition has been carried out using the Harrington criterion.

The influence of the nanoaddition silver\silica (Ag/SiO_2) on the properties of the Polypropylene (PP) monothreads has been explored using the method of mathematical modeling and the content of composition for their forming has been optimized.

The mathematical model, that defines the interconnection between the content of the mixture components and the properties of the nanofilled PP threads, has been created.

Modified monothreads formed of the optimal content of the PP\nanoaddition composition combine high level of strength and elasticity and develop antibacterial effect.

Introduction

Topicality of working out methods of obtaining fibers and threads with antibacterial effect is caused by necessity in creating some medical products to cure and protect medical workers and biologists. Attaching bactericidal properties to threads by inserting metal nanoparticles is one of the most perspective. Using binary nanocompos, where nanoparticles of biometals are brought in the surface of inert sorbents, enables creating fundamentally new materials, that combine antibacterial and sorption effect. Thus, nanocompo Ag/SiO_2 is almost ten times more effective compared to original components, shows high prolonged antibacterial effect and is safer for peoples' health and the environment [1].

Problem

In modern medicine biologically active materials made from Polypropylene (PP) have become really meaningful, because they are chemically inert, resistant to microorganisms and have high level of strength and elasticity. It is known that metallic ions are of high antibacterial properties and at the same time they have a toxic effect on living beings. Within the transition to the nanostate, toxicity of metals decreases [2]. Nanoadditions also have a great influence on mechanical indicators of threads. To define the

interconnection between the composition content and the characteristics of threads it is necessary to carry out a great number of multivariate experiments. They are connected with time and materials' expenditure, because the impact of each factor is explored apart from others, with fixed meanings of other parameters. One of the ways, which allows to carry out scientific researches fast enough and find the decisions most approximate to optimal ones with minimal expenditures, is the usage of mathematical methods of planning the experiment.

Purpose of this work – planning the experiment and optimization of composition content Polypropylene/ binary nanoaddition in order to obtain Polypropylene monothreads with high mechanical and antibacterial properties.

Main material

Strength and elasticity are the main parameters that define the safety of the surgical stitch. When planning the experiment such parameters were chosen as original ones:

y_1 - strength of monothreads when ruining, y_2 - the original module of threads, y_3 - diameter of the retardment of the microorganisms' growth, y_3 - diameter of the retardment of the St. aureus microorganisms' growth, and original ones were: x1, x2, x3 – approximate concentrations of PP, Ag and SiO_2 respectively.

The simplex-grid method in pseudo coordinates is the most appropriate method for mixture systems optimization [3]. Simplex is the simplest geometrical figure, formed by the set *k+1* independent points in k-dimensional space. Independent variables are called 'factors', space with coordinates x1, x2, x3 is called «factorial space», and the geometrical delineation of the function of response in factorial space is called «response surface». Correlation of the ingredients in systems being explored must satisfy the following condition: $\sum_{i=1}^{q} x_i = 1$, where x_i is approximate concentration of ingredients ($x_i \geq 0$); q – quantity of the ingredients ($q \geq 2$).

As certain limits are put on the concentration of some ingredients of three-component mixture, the researches were carried out in the limited part of the factorial space. As the result the ‘cut-out ’ part was received , which was unsimilar to simplex, and experimental points were located in it. Having written the coordinates of experimental points of the simplex grid, we received matrix of planning. In order to use the standard plan the part being explored was transformed into the new coordinate system ($z_1, z_2\ z_3 \ldots\ z_q$) [3]. Simplex vertices were being accepted as independent ingredients of the mixture, so called pseudocomponents. To transit from the previous coordinate system ($x_1,\ x_2, \ldots x_q$) to the new one ($z_1,\ z_2\ \ldots\ z_q$) the following matrix equation was used: $X = AZ$.

It can be written in the detailed way:

$$\begin{vmatrix} x_1^{(u)} \\ x_2^{(u)} \\ \vdots \\ x_q^{(u)} \end{vmatrix} = \begin{vmatrix} x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(q)} \\ x_2^{(1)} & x_2^{(2)} & \cdots & x_2^{(q)} \\ \vdots & \vdots & \cdots & \vdots \\ x_q^{(1)} & x_q^{(2)} & \cdots & x_q^{(q)} \end{vmatrix} \times \begin{vmatrix} z_1^{(u)} \\ z_2^{(u)} \\ \vdots \\ z_q^{(u)} \end{vmatrix} \qquad (2.1)$$

In equation (1) elements of matrix A are the coordinates of vertices of transformed simplex, and $x_i^{(u)}$ та $z_i^{(u)}$ (i = 1, 2,…, q) – original and new coordinates of u- transformed point. Herewith such conditions are being done in z-coordinates: $0 \le z_i \le 1, \quad (i = 1, 2, \ldots, q), \quad z_1^{(u)} + z_2^{(u)} + \ldots + z_q^{(u)} = 1,$

where u is any point of the factorial space.

To work out a model, which defines the interconnection between the content of the components and the properties of the modified monothreads, the incomplete cubical polynoma was used:

$$\hat{y} = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_{12} x_1 x_2 + \beta_{13} x_1 x_3 + \beta_{23} x_2 x_3 + \beta_{123} x_1 x_2 x_3 \qquad (2.2)$$

where $\beta_i, \beta_{ij}, \beta_{ijk}$ - are polynomial coefficients, moreover $i \ne j \ne k$ = 1, 2, 3.

To estimate numeric values of the coefficients of the equation, the plan of carrying out the experiments in the area of the factorial space being explored

was prepared (table 1), herewith z-coordinates were chosen from the standard plan for the model given [3], x-coordinates were counted according to the formula (1).

Table 1. Simplex-grid plan

№ of the experime	Plan of experiment						obtained
	Plan in			Working plan			
	z_1	z_2	z_3	x_1	x_2	x_3	
1	1	0	0	0.9851	0.0050	0.00	$\bar{y}_1$
2	0	1	0	0.9880	0.0021	0.00	$\bar{y}_2$
3	0	0	1	0.9920	0.0040	0.00	$\bar{y}_3$
4	0,5	0,5	0	0.9866	0.0036	0.00	$\bar{y}_{12}$
5	0,5	0	0,5	0.9900	0.0030	0.00	$\bar{y}_{13}$
6	0	0,5	0,5	0.9886	0.0045	0.00	$\bar{y}_{23}$
7	0,33	0,33	0,33	0.9785	0.0037	0.00	$\bar{y}_{123}$

To define the influence of correlation PP/ Ag/SiO_2 the mixture on the mechanical and antibacterial properties of monothreads according to the plan a series of experiments was carried out and original and obtained parameters were defined (table 2).

Table 2. Influence of the concentration Ag/SiO_2 on the properties of PP monothreads

Original variable	Number of the experiment						
	1	2	3	4	5	6	7
y_1	480	540	590	510	530	570	540
y_2	6200	7800	7900	6500	6900	8000	7700
y_3	14.1	8,5	13.8	13.5	9,5	13.3	11.4

On the basis of the data mentioned in table 2 polynomial coefficients (2) have been counted using the method of the least squares in the matrix form. The countings have been done using the specially created programme in the Delphi programming environment on the Object Pascal language. As the result, the system of the equations has been received (3). It is a mathematical model, that describes the process being explored in z-coordinates.

$$\begin{cases} y_1 = 479{,}99z_1 + 539{,}99z_2 + 590z_3 + 0z_1z_2 - 19{,}99z_1z_3 + 20{,}00z_2z_3 + 104{,}80z_1z_2z_3 \\ y_2 = 6200z_1 + 7800z_2 + 7899{,}99z_3 - 2000z_1z_2 - 599{,}99z_1z_3 + 600z_2z_3 + 17036{,}16z_1z_2z_3 \\ y_3 = 14{,}1z_1 + 8{,}5z_2 + 13{,}8z_3 + 8{,}8z_1z_2 - 17{,}8z_1z_3 + 8{,}6z_2z_3 - 18{,}33z_1z_2z_3 \end{cases} \quad (2.3)$$

Having defined the coefficients, the mathematical model was being checked in adequacy, which means ability to predict the results of the research in some area with necessary exactness. For this, additional experiments were being put in so called control points, the value of the Student criterion was being counted and compared with the table data. Received values of the criterion mentioned are the evidence of the adequacy of this model.

To solve the problem of optimization the so called generalized function of advisability (D) was used. Harrington offered to use it as the generalized criterion of optimization [4]. To count value D state values of responses were transformed into the non-dimensional scale of advisability for each original parameter using exponential dependency. The generalized criterion of D optimization was being counted as the geometric mean of partial functions of advisability. The value of the Harrington criterion is limited within the interval [0...1] (0 stands for absolutely unacceptable value of the response given, 1 stands for the most optimal value of the response).

Software that implements the described algorithm has been developed [26, 29, 32]. The text of the main program procedures is given in the appendix 13.

Optimal content of the mixture being explored was being defined using the method of scanning by step 0,01 in z-coordinates. According to the matrix

equation (1) the content of original components was transformed into the x-system. While the criterion of advisability D=0.8256 the determined optimal correlation of mixture components for monothreads formation is mas%: PP – 99,16; Ag – 0,38; SiO_2 – 0,46, and indicators that characterize the quality of modified threads, are as following: comparative strength of monothreads when ruining – 587 MPa, original module – 7944 MPa, diameter of area of St.aureus bacteria growth retardment – 14,0 mm.

Laboratorial patterns of monothreads have been worked out from the composition of optimal content and their properties have been explored. It has been found out, that stitch threads have an antibacterial effect; they also have good operating characteristics and fix the surgical knot in a proper way due to high strength and elasticity.

Conclusions

Planning the experiment concerning the influence of the binary nanoaddition silver\silica on the properties of the Polypropylene monothreads has been carried out using the method of mathematical modeling**.** The content of Ag/SiO_2 in the PP fusion has been optimized and biologically active monothreads with maximal mechanical characteristics have been formed.

LITERATURE

1. Бахвалов Н.С., Жидков Н.П., Кобельков Г.М. Численные методы. 8-е изд., М.: БИНОМ. Лаборатория знаний, 2015. - 636 с.

2. Baklavaridis A., Zuburtikudis I., Panayiotou C. Porous Composite Structures Derived from Multiphase Polymer Blends // Polym. Eng. Sci. – 2015. – V.55, №8. – P. 1856-1863.

3. Beloshenko V.A., Plavan V.P., Rezanova N.M., Savchenko B.M., Vozniak I. Production of high-performance multi-layer fine-fibrous filter materials by application of material extrusion-based additive manufacturing // The Int. J. of Adv. Manufac. Techn. – 2019. – №.101. – P. 2681-2688.

4. Васильев А. Программирование на C++ в примерах и задачах. М.: Эксмо, 2021. – 368 с.

5. Guo J., Briggs N., Crossley S., Grady B.P. Morphology of Polystyrene/poly(methyl Methacrylate) Blends: Effects of Carbon Nanotubes Aspect Ratio and Surface Modification // AIChE J. – 2015. – V. 61, № 10. – P. 3500–3510.

6. Демидович Б. П., Марон И. А. Основы вычислительной математики: учебное пособие для студентов втузов - 4-е изд., испр. - М. : Наука, 1970. - 664 с.

7. Зедгинидзе И.Г. Планирование эксперимента для исследования многокомпонентных систем. – М.: Наука, 1976. – 392 с.

8. Исаков В.Н. Элементы численных методов: учеб. пособ. – М.: Академия, 2008. – 192с.

9. Колдаев В. Д. Численные методы и программирование. М.: ИД «ФОРУМ». - 2021. - 336с.

10. Культин Н. Основы программирования в Delphi 7. С-Пб.: БХВ-Петербург, 2012. – 608с.

11. Липпман С., Лажойе Ж., Му Б. Язык программирования C++. Базовый курс. - М.: Вильямс, 2016. - 1120 с.

12. Мейерс С. Эффективный и современный C++. М.: Вильямс, 2016. - 304 с.

13. Прата С. Язык программирования C++ (C++11). Лекции и упражнения, 6-е издание — М.: Вильямс, 2012. — 1248 с.

14. Резанова В.Г. Математичне моделювання та компютерна візуалізація процесу специфічного волокно утворення // К.: Вісник КНУТД. - № 4, 2017. - с. 19-26

15. Резанова В.Г., Резанова Н.М. Програмне забезпечення для дослідження полімерних систем // К.:АртЕк. - 2020. 358 с.

16. Резанова В.Г., Резанова Н.М. Розробка програмного забезпечення для визначення реологічних характеристик розплавів полімерів // Вісник Хмельницького національного університету. - № 5, 2018. - с. 21-25

17. Rezanova V.G., Shchotkina V.I., Tsebrenko M.V. Planning the experiment and optimization of the content of nanoaddition in polypropylene monothreads // Вісник КНУТД. – 2014. – № 2. – С. 42-47.

18. Резанова В.Г. Дослідження властивостей чотирикомпонентних систем методом математичного моделювання // Вісник КНУТД. – 2014. – № 3. – С. 113-120.

19. Резанова В.Г. Перетворення задачі оптимізації при дослідженні чотирикомпонентних сумішей полімерів / К.: Вісник КНУТД. – 2016. – №2 . – С. 40-47.

120. Резанова В.Г. Оптимізація складу чотирикомпонентних сумішей полімерів із застосуванням методу штрафних функцій Оптимізація методом штрафних функцій // К.: Вісник КНУТД. – 2016. – №3 . – С. 59-67.

21. Резанова В.Г., Резанова Н.М., Коршун А.В. Дослідження морфології сумішей полімерів з використанням розробленого програмного забезпечення // Вісник КНУТД. – 2017. – №2. – С.120–127.

22. V. G. Rezanova, N. M. Rezanova Mathematical Modelling and Software Development to Optimize the Composition of Four-Component Nanofilled Systems // Наносистеми, наноматеріали, нанотехнології. ("Nanosistemi, Nanomateriali, Nanotehnologii") - 2020, т. 18, № 4, с. 863–874

23. Rezanova V.G., Shchotkina V.I., Tsebrenko M.V. Planning the experiment and optimization of the content of nanoaddition in polypropylene monothreads // Вісник КНУТД. – 2014. – № 2. – С. 42-47

24. Роджерс Д., Адамс Дж. Математические основы машинной графики. – М.: Мир, 2008. – 239с.

25. Tran N. H. A, Brünig, H., Landwehr M.A., Vogel R., Heinrich G. Controlling micro- and nanofibrillar morphology of polymer blends in low-speed melt spinning process. Part II: Influences of extrusion rate on morphological changes of PLA/PVA through a capillary die // J. Appl. Polym. Sci. – 2016. – № 133. – P.442-573.

26. Фленов М. Библия программиста (Delphi), 3-е издание С-Пб.: БХВ-Петербург, 2011. – 688с

27. Цебренко М.В., Резанова Н.М., Мельник І.А., Резанова В.Г., Вільцанюк О.А., Хуторянський М.О. Нанонаповнені поліпропіленові мононитки // Вісник КНУТД. – 2012. – №4. – С 93-97.

28. Шахов Ю.Н., Деза Е.И., Численные методы. – М.: Либроком, 2017. – 248 с.

29 Шилдт Г. C++. Базовый курс. – М.: Диалектика-Вильямс, 2018. – 624 с.

30. Шлее М. Qt 5.10. Профессиональное программирование на C++. С.-Пб.: БХВ-Петербург, 2018. – 1074 с.

31. Щербань В.Ю., Краснитський С.М., Резанова В.Г.. Математические моделі в САПР. Обрані розділи та приклади застосування: – К.:КНУТД, 2011. – 219с.

32. Stroustrup B. Programming: Principles and Practice Using C++ (2nd Edition). Addison-Wesley Professional, 2014. – 1312 p.

33. Vrsaljko D., Macut D., Kovačević V. Potential Role of Nanofillers as Compatibilizers in Immiscible PLA/LDPE Blends // J. Appl. Polym. Sci. – 2015. – V. 132, № 6. – P. 119-127.

Annex 1

The procedure for calculating the determinant of a matrix of arbitrary order

```
procedure PrDetN(KoefN:Matr;n:integer; var DetN:Real);
var Koef:Matr;
   i,j:integer;
   Det,Det3:real;
begin
if n=3 then begin
          Det3:=KoefN[1,1]*KoefN[2,2]*KoefN[3,3]+
               KoefN[2,1]*KoefN[3,2]*KoefN[1,3]+
               KoefN[1,2]*KoefN[2,3]*KoefN[3,1]-
               KoefN[1,3]*KoefN[2,2]*KoefN[3,1]-
               KoefN[2,1]*KoefN[1,2]*KoefN[3,3]-
               KoefN[1,1]*KoefN[2,3]*KoefN[3,2];
          DetN:=Det3;
         end
      else
begin
Det:=KoefN[1,1];
   for i:=2 to n do
     begin
       for j:=2 to n do
         begin
           Koef[i-1,j-1]:=
            (KoefN[1,1]*KoefN[i,j]-KoefN[i,1]*KoefN[1,j])/KoefN[1,1];
         end;
     end;
     for i:=1 to n-1 do
     begin
       for j:=1 to n-1 do
         begin
           KoefN[i,j]:=Koef[i,j];
         end;
     end;
PrDetN(KoefN,n-1,DetN);
DetN:=DetN*Det;
end;
end;
```

Annex 2

The procedure for solving the system of linear equations by the Cramer method

```
procedure Kramer(A:matr;b:vector;n:integer;var x:vector);
var i,j:integer;
   DetAo,DetAd:real;
   temp:vector;
begin
PrDetN(A,n,DetAo);   // annex 1
if DetAo=0 then ShowMessage('Kramer metod can not be used')
        else
          begin
           for j:=1 to n do
            begin
             for i:=1 to n do
              begin
               temp[i]:=A[i,j];
               A[i,j]:=b[i];
              end;
              PrDetN(A,n,DetAd);           // annex 1
              x[j]:=DetAd/DetAo;
              for i:=1 to n do
              begin
               A[i,j]:=temp[i];
              end;
             end;
          end;
end;
```

Annex 3

Procedure that implements the Gaussian method

```
procedure pram_hid(var A:mas);
var
i,j,z,rad,c:integer;
max,temp:real;
x:odnomir;
begin
for i:=1 to n do begin //schotchik stovbchikiv
//perestavlayem yakscho diagonalniy element =0
if A[i,i]=0 then begin
  for j:=i to n do
    if (A[j,i]<>0) then begin
     rad:=j; break; end;
               //perestavlayem radki
    for z:=i to n+1 do begin
       temp:=A[i,z]; A[i,z]:=A[rad,z];
       a[rad,z]:=temp;
       end;
//delim  radok z diagonalnim elementom
temp:=A[i,i];
for j:=i to n+1 do    begin
 A[i,j]:=A[i,j]/temp;
 end;
       if  i<=n then
      for j:=i+1 to n do    BEGIN TEMP:=-a[J,I];
        for z:=i to n+1 do
        A[j,z]:=(TEMP*A[I,Z])+A[J,z];
          end;
           end;
end;

procedure zvor_hid(var A:mas);
var
i,j,z,rad,c:integer;
max,temp:real;
x:odnomir;
begin
for i:=n downto 1 do begin
     if  i>=1 then
      for j:=i-1 downto 1 do    BEGIN TEMP:=-a[J,I];
         for z:=n+1 downto i do
        A[j,z]:=(TEMP*A[I,Z])+A[J,z];
           end;
end;
end;
```

Annex 4

The procedure for solving a system of linear equations (in normal form) by simple iterations

```
procedure Iter(A:matr;S1:vector;X0:vector;n:integer;eps:real;var X1:vector);
var Xk,Xk_p1:vector;
    t,t0:real;
    i,j,k,k_iter:integer;
begin
t0:=0;
k_iter:=round(ln(eps*(1-norma)/norma_v)/ln(norma))+1;
   for i:=1 to n do
     begin
      Xk[i]:=X0[i];
     end;
 for k:=1 to k_iter do
  begin
     for i:=1 to n do
      begin
        Xk_p1[i]:=S1[i];
        for j:=1 to n do
         begin
          Xk_p1[i]:=Xk_p1[i]+A[i,j]*Xk[j];
          end;
        t:=abs(Xk_p1[i]-Xk[i]);
        if t>t0 then t0:=t;
        Xk[i]:=Xk_p1[i];
      end;
    for i:=1 to n do
     begin
      X1[i]:=Xk_p1[i];
     end;
  end;
end;
```

Annex 5

The procedure for solving a system of linear equations (in normal form) by the Seidel method

```
procedure Zeidel(A:matr;S1:vector;X0:vector;n:integer;eps:real;var X1:vector);
var Xk,Xk_p1:vector;
    t,t0,delta:real;
    i,j,k,k_iter:integer;
begin  {proc}
   for i:=1 to n do
     begin
      Xk[i]:=X0[i];
     end;
      delta:=eps*(1-1/norma);
repeat
     for i:=1 to n do
      begin
        Xk_p1[i]:=S1[i];
        for j:=1 to n do
         begin
          if i>j then Xk_p1[i]:=Xk_p1[i]+A[i,j]*Xk_p1[j]
                 else Xk_p1[i]:=Xk_p1[i]+A[i,j]*Xk[j];
         end;
         t:=abs(Xk_p1[i]-Xk[i]);
         Xk[i]:=Xk_p1[i];
      end;
until t>delta;
   for i:=1 to n do
     begin
      X1[i]:=Xk_p1[i];
     end;

end;
```

Annex 6

Auxiliary procedures for the implementation of methods for refining the roots of transcendental equations

```
procedure znaki_f(a,b:real; var flag:boolean);
var x1:real;
begin
 flag:=true;
 x1:=f(a)*f(b);
 if x1>0 then begin
              showMessage('Function doesnt change signum. Choose other interval!');
              flag:=false;
             end;
end;

procedure znak_f_(a,b:real; var flag1:boolean);
var x1,x2,f_x1,f_x2,h,pr:real;
begin
 flag1:=true;
 h:=0.00001;
 x1:=a;
 f_x1:=f_(x1);
 while x1<=b do
  begin
   x2:=x1+h;
   f_x2:=f_(x2);
   pr:=f_x1*f_x2;
   if pr<0 then begin
                showMessage('First proizv. changes signum. Choose other interval!');
                flag:=false;
               end;
    x1:=x2;
   end;
end;

procedure min_f_(a,b,h:real; var m1:real); var x1,x2,f_x1,f_x2:real;
begin
  x1:=a;
  f_x1:=abs(f_(x1));
  m1:=f_x1;
  while x1<=b do
   begin
    x2:=x1+h;
    f_x2:=abs(f_(x2));
    if f_x2<m1 then m1:=f_x2;
    x1:=x2;
    end;
end;
```

```
procedure max_f__(a,b:real; var M2:real);
var x1,x2,f__x1,f__x2,h:real;
begin
  h:=0.00001;
  x1:=a;
  f__x1:=abs(f__(x1));
  M2:=f__x1;
  while x1<=b do
   begin
    x2:=x1+h;
    f__x2:=abs(f__(x2));
    if f__x2>M2 then M2:=f__x2;
    x1:=x2;
    end;
end;
```

Annex 7

The procedure that implements the clarification of the root of the transcendental equation by the method of half division

```
procedure M_Dihot(a,b:real; var koren:real);
var c, fa, fb, fc: real;
   i:integer;
begin
if abs(b-a)<eps then begin c:=(a+b)/2;
                     koren:=c;end
             else
 begin
  c:=(a+b)/2;
 fc:=f(c);
 fa:=f(a);
 fb:=f(b);
 if fc=0 then koren:=c
        else
          begin

            if fa*fc<0 then begin a:=a; b:=c; end else begin a:=c; b:=b; end;
            M_Dihot(a,b,c);
          end;
end;
end;
```

Annex 8

The procedure that implements the refinement of the root of the transcendental equation by the method of tangents

```
procedure M_Dot(a,b,eps:real;var x:real);
var xk,xk1,razn,delta:real;
    i:integer;
begin
 i:=1;
 min_f_(a,b,0.0001,m1); // Виклик процедури, описаної в додатку 6
 max_f__(a,b,M2); // Виклик процедури, описаної в додатку 6
 if f(a)*f__(a)>0 then xk:=a else xk:=b;
 delta:=power((eps*m1/M2),(1/2));
 repeat
  xk1:=xk-f(xk)/f_(xk);
  razn:=abs(xk1-xk);
   xk:=xk1;
   i:=i+1;
  until(razn<delta);
 x:=xk;
end;
```

The procedure for finding the compression ratio

```
procedure k_szhatia(a,b:real;var q:real; var flag:boolean);
var x1,x2,
    f_1,f_2:real;
begin
 flag:=true;
 f_1:=abs(fi_(x1));
 q:=abs(fi_(x1));
 if f_1>1 then flag:=false;
  while x1<=b do
   begin
    x2:=x1+0.00001;
    f_2:=abs(fi_(x2));
    if f_2>1 then flag:=false
            else if f_2>q then q:=f_2;
    x1:=x2;
    end;
end;
```

Annex 9

The procedure that implements the refinement of the root of the transcendental equation by the method of iterations

```
procedure ur_iter(x0,eps:real;var x:real);
var xn,xn1,razn:real;
begin
 k_szhatia(a,b,q,flag);
 if flag=false then showMessage('Method can not be used!')
              else
               begin
                 xn:=x0;
                 delta:=eps*(1/q-1);
                 repeat
                  xn1:=fi(xn);
                  razn:=abs(xn1-xn);
                  vivid_iter(xn,xn1,razn);
                  xn:=xn1;
                 until razn<delta;
  x:=xn1;
               end;
end;
```

Annex 10

A procedure that clarifies the roots of a system of two nonlinear equations by Newton's method

```
procedure Sys2_Newton(x0,y0,eps:real;var x,y:real);
var deltax, deltay, delta:real;
begin
 repeat
  A[1,1]:=f1_x(x0,y0);
  A[1,2]:=f1_y(x0,y0);
  A[2,1]:=f2_x(x0,y0);
  A[2,2]:=f2_y(x0,y0);

 b[1]:=-f1(x0,y0);
  b[2]:=-f2(x0,y0);

Kramer(A,b,2,x1);// annex 2
  deltax:=x1[1];
  deltay:=x1[2];

  x:=x0+deltax;
  y:=y0+deltay;

  x0:=x;
  y0:=y;

  if abs(deltax)>abs(deltay) then delta:=abs(deltax) else delta:=abs(deltay);

  until (delta<eps);

  end;
```

Annex 11

A procedure that implements the solution of a system of two differential equations by the Runge-Kutta method

```
procedure M_Runge_Kutta(CurrT,CurrTeta,CurrQ,h:real;Var Yk1,Zk1:real);
var k1,k2,k3,k4,
    m1,m2,m3,m4,
    FQ,FTeta:real;
begin
if CurrQ>0 then begin
 FuncQ(CurrT,CurrTeta,CurrQ,FQ);
 FuncTeta(CurrT,CurrTeta,CurrQ,FTeta);
 k1:=FTeta*h;
 m1:=FQ*h;

 FuncQ(CurrT+h/2,CurrTeta+k1/2,CurrQ+m1/2,FQ);
 FuncTeta(CurrT+h/2,CurrTeta+k1/2,CurrQ+m1/2,FTeta);
 k2:=FTeta*h;
 m2:=FQ*h;

 FuncQ(CurrT+h/2,CurrTeta+k2/2,CurrQ+m2/2,FQ);
 FuncTeta(CurrT+h/2,CurrTeta+k2/2,CurrQ+m2/2,FTeta);
 k3:=FTeta*h;
 m3:=FQ*h;

 FuncQ(CurrT+h,CurrTeta+k3,CurrQ+m3,FQ);
 FuncTeta(CurrT+h,CurrTeta+k3,CurrQ+m3,FTeta);
 k4:=FTeta*h;
 m4:=FQ*h;

 Yk1:=CurrTeta+(1/6)*(k1+2*k2+2*k3+k4);
 Zk1:=CurrQ+(1/6)*(m1+2*m2+2*m3+m4);
  end
end;
```

Annex 12

Procedures for calculating the Lagrange polynomial

```
procedure znamen;
var k,i:integer;
begin
for k:=0 to z-1 do begin
 znam[k]:=1;
 for i:=0 to z-1 do begin
   if k<>i then begin znam[k]:=znam[k]*(t[k]-t[i]);    end;
end;
 end;end;


Procedure l(dx:real; var xc,yc:real);
var k,i:integer;
begin
xc:=0; yc:=0;
for k:=0 to z-1 do begin
for i:=0 to z-1 do
if i<>k then   begin
        c[k]:=c[k]*(dx-t[i]);
        end;
        xc:=xc+(x[k]*(c[k]/znam[k]));
        yc:=yc+(y[k]*(c[k]/znam[k]));
       end;
end;

procedure paint1(mas:real);
var g:real;
begin
znamen;
g:=-10;
repeat
  l(g,xc,yc);
  form1.image1.Canvas.Pixels[round((mas*xc)+(w/2)),round((mas*-yc)+(h/2))]:=clred;
  g:=g+0.001;
until g>10;
end;
```

Annex 13

Basic procedures and functions for calculation and graphical display of rheological characteristics of polymer blends

```
procedure TForm1.Button2Click(Sender: TObject);
var i:integer;
    q:string;
begin
for i:=0 to 10 do
   begin
     SdTcp[i]:=S[i]/Tcp[i]/1000;
     LgSdTcp[i]:=ln(SdTcp[i])/ln(10);
     Lg_D[i]:=LgK2+LgSdTcp[i];
   end;
   for i:=0 to 9 do
     begin
       dLg_D[i]:=Lg_D[i]-Lg_D[i+1];
       dLgT[i]:=LgT[i]-LgT[i+1];
       N[i]:=dLg_D[i]/dLgT[i];
     end;
  for i:=0 to 9 do
   begin
     LgD[i]:=ln(N[i]+3)/ln(10)+Lg_D[i];
     LgEta[i]:=LgT[i]-LgD[i];
    Eta[i]:=Power(10,LgEta[i]);
  end;
  Form2.Show;
  Form5.Hide;
    for i:=0 to 9 do
  begin
    Form2.Memo1.Lines.add(IntToStr(i+1));
    q:=format('%*.*f',[8,7,SdTcp[i]]);
    Form2.Memo5.Lines.add(q);
    q:=format('%*.*f',[5,4,Lg_D[i]]);
    Form2.Memo6.Lines.add(q);
    q:=format('%*.*f',[5,4,N[i]]);
    Form2.Memo7.Lines.add(q);
    q:=format('%*.*f',[2,1,Tcp[i]]);
    Form2.Memo2.Lines.add(q);
    q:=format('%*.*f',[2,0,S[i]]);
    Form2.Memo3.Lines.add(q);
    q:=format('%*.*f',[5,4,LgT[i]]);
    Form2.Memo4.Lines.add(q);
    q:=format('%*.*f',[5,4,LgD[i]]);
    Form2.Memo8.Lines.add(q);
    q:=format('%*.*f',[5,4,LgEta[i]]);
    Form2.Memo9.Lines.add(q);
    q:=format('%*.*f',[5,1,Eta[i]]);
    Form2.Memo10.Lines.add(q);
  end;
```

```
end;


procedure TForm1.Button3Click(Sender: TObject);
var i:integer;
    q:string;
begin
 for i:=0 to kd-1 do
   begin
     SdTcp[i]:=S[i]/Tcp[i]/1000;
     LgSdTcp[i]:=ln(SdTcp[i])/ln(10);
     Lg_D[i]:=LgK2+LgSdTcp[i];
   end;
   for i:=0 to kd-2 do
     begin
       dLg_D[i]:=Lg_D[i]-Lg_D[i+1];
       dLgT[i]:=LgT[i]-LgT[i+1];
       N[i]:=dLg_D[i]/dLgT[i];
     end;
  for i:=0 to kd-2 do
   begin
     LgD[i]:=ln(N[i]+3)/ln(10)+Lg_D[i];
     LgEta[i]:=LgT[i]-LgD[i];
    Eta[i]:=Power(10,LgEta[i]);
  end;
  Form2.Show;
  Form5.Hide;
      for i:=0 to kd-2 do
  begin
    Form2.Memo1.Lines.add(IntToStr(i+1));
    q:=format('%*.*f',[8,7,SdTcp[i]]);
    Form2.Memo5.Lines.add(q);
    q:=format('%*.*f',[5,4,Lg_D[i]]);
    Form2.Memo6.Lines.add(q);
    q:=format('%*.*f',[5,4,N[i]]);
    Form2.Memo7.Lines.add(q);
    q:=format('%*.*f',[2,1,Tcp[i]]);
    Form2.Memo2.Lines.add(q);
    q:=format('%*.*f',[2,0,S[i]]);
    Form2.Memo3.Lines.add(q);
    q:=format('%*.*f',[5,4,LgT[i]]);
    Form2.Memo4.Lines.add(q);
    q:=format('%*.*f',[5,4,LgD[i]]);
    Form2.Memo8.Lines.add(q);
    q:=format('%*.*f',[5,4,LgEta[i]]);
    Form2.Memo9.Lines.add(q);
    q:=format('%*.*f',[5,1,Eta[i]]);
    Form2.Memo10.Lines.add(q);
  end;
  for i:=0 to kd-2 do
```

```
   begin
     LgD2[i]:=ln(N2[i]+3)/ln(10)+Lg_D2[i];
     dLgD[i]:=LgD[i]-LgD[i+1];
     LgEta2[i]:=LgT2[i]-LgD2[i];
    Eta2[i]:=Power(10,LgEta2[i]);
  end;
   end;

procedure TForm1.Button5Click(Sender: TObject);
begin
 kd:=StrToInt(Edit34.Text);
end;

procedure vvod(c:STRING; var F_Tcp,F_S,F_LgT:TS );
var f:textfile;
i,j:integer;
begin
    assignfile(f,c);
    reset(f);
    readln(f,kd);
        for i:=0 to kd-1 do
    begin
         read(f,F_Tcp[i]);
      read(f,F_S[i]);
      read(f,F_LgT[i]);
      readln(f);
    end;
end;

procedure TForm1.Button7Click(Sender: TObject);
var f:textfile;
i,j:integer;

begin
   assignfile(f,'inp.txt');
    reset(f);
    readln(f,kd);
      for i:=0 to kd-1 do
    begin
       read(f,Tcp[i]);
      read(f,S[i]);
      read(f,LgT[i]);
           readln(f);
    end;
closefile(f);}
Form6.Show;
end;
end.
```

```
Unit2
Procedure vivid(C:STRING;D:TS);
var f1:textfile;
i,j:integer;
begin
   assignfile(f1,C);
   append(f1);
   for i:=1 to kd-1 do
   begin
     write(f1,D[i]:0:4);  //write(f1,' ');
     writeln(f1);
   end;
close(f1);
end;


procedure TForm2.Button1Click(Sender: TObject);
var i:integer;
   f2, f3,f4:textfile;
    q:string;
begin
 kNazhatiy:=kNazhatiy+1;
 Nr:=StrToInt(Edit21.Text);
 Form3.Show;
  Ncp1:=0;
 Ncp2:=0;
for i:=0 to Nr-1 do
  begin
    Ncp1:=Ncp1+N[i];
  end;
 Ncp1:=Ncp1/Nr;
 for i:=0 to Nr-1 do
  begin
     LgD[i]:=ln(Ncp1+3)/ln(10)+Lg_D[i];
     LgEta[i]:=LgT[i]-LgD[i];
     Eta[i]:=Power(10,LgEta[i]);
  end;
     for i:=0 to Nr-1 do
  begin
     if kNazhatiy=1 then LgEta1[i]:=LgEta[i];
     if kNazhatiy=2 then LgEta2[i]:=LgEta[i];
     if kNazhatiy=3 then LgEta3[i]:=LgEta[i];
  end;
  for i:=0 to Nr-1 do
  begin
     LgEta123[i]:=LgEta1[i];
  end;
  for i:=Nr to (Nr-1)*2 do
  begin
     LgEta123[i]:=LgEta2[i];
```

```
  end;
  for i:=(Nr-1)*2+1 to (Nr-1)*3 do
  begin
     LgEta123[i]:=LgEta3[i];
  end;
 for i:=Nr to kd-1 do
  begin
    Ncp2:=Ncp2+N[i];
  end;
 Ncp2:=Ncp2/(kd-1-Nr);
Form3.Edit1.Text:=FloatToStr(Ncp1);
Form3.Edit2.Text:=FloatToStr(Ncp2);
  for i:=Nr to kd-2 do
  begin
     LgD[i]:=ln(Ncp2+3)/ln(10)+Lg_D[i];
     LgEta[i]:=LgT[i]-LgD[i];
    Eta[i]:=Power(10,LgEta[i]);
    Memo1.Lines.add(FloatToStr(LgEta[i]));
    Memo2.Lines.add(FloatToStr(Eta[i]));
      end;
   for i:=0 to kd-2 do
  begin
    Form3.Memo4.Lines.add(IntToStr(i+1));
    q:=format('%*.*f',[8,7,SdTcp[i]]);
    Form3.Memo1.Lines.add(q);
    q:=format('%*.*f',[5,4,Lg_D[i]]);
    Form3.Memo2.Lines.add(q);
    q:=format('%*.*f',[5,4,N[i]]);
    Form3.Memo3.Lines.add(q);
    q:=format('%*.*f',[2,1,Tcp[i]]);
    Form3.Memo5.Lines.add(q);
    q:=format('%*.*f',[2,0,S[i]]);
    Form3.Memo6.Lines.add(q);
    q:=format('%*.*f',[5,4,LgT[i]]);
    Form3.Memo7.Lines.add(q);
    q:=format('%*.*f',[5,4,LgD[i]]);
    Form3.Memo8.Lines.add(q);
    q:=format('%*.*f',[5,4,LgEta[i]]);
    Form3.Memo9.Lines.add(q);
    q:=format('%*.*f',[5,1,Eta[i]]);
    Form3.Memo10.Lines.add(q);
  end;
  vivid ('out.txt',LgT);
  vivid ('out2.txt',LgEta);
 end;

Unit3
procedure Det3x3(Koef:MyArr; var Det:Real);
begin
  Det:=Koef[1,1]*Koef[2,2]*Koef[3,3]+
```

```
      Koef[2,1]*Koef[3,2]*Koef[1,3]+
      Koef[1,2]*Koef[2,3]*Koef[3,1]-
      Koef[1,3]*Koef[2,2]*Koef[3,1]-
      Koef[2,1]*Koef[1,2]*Koef[3,3]-
      Koef[1,1]*Koef[2,3]*Koef[3,2];
end;

procedure MinSq(X,Y:TS; Ac,Bc,Cc:real);
var i,j:integer;
   S1,S2,S3,S4,S5,S6,S7:real;
begin
for i:=0 to 9 do
  begin
    S1:=S1+power(X[i],4);
    S2:=S2+power(X[i],3);
    S3:=S3+X[i]*X[i];
    S4:=S4+X[i];
    S5:=S5+X[i]*X[i]*Y[i];
    S6:=S6+X[i]*Y[i];
    S7:=S7+Y[i];
  end;
 Koef[1,1]:=S1;
 Koef[2,1]:=S2;
 Koef[3,1]:=S3;
 Koef[1,2]:=S2;
 Koef[2,2]:=S3;
 Koef[3,2]:=S4;
 Koef[1,3]:=S3;
 Koef[2,3]:=S4;
 Koef[3,3]:=1;
 St[1]:=S5;
 St[2]:=S6;
 St[3]:=S7;
 Kramer3(Koef,St,Ac,Bc,Cc);
end;

Unit4
procedure Scale(A,B:TS;var Mx,My,KA,KB,MinA,MinB,ShA,ShB:Real);
var k:integer;
   pr,pr1,Min,Min1,Max,Max1:Real;
begin
  Min:=A[0];
  Max:=A[0];
  Min1:=B[0];
  Max1:=B[0];
  for k:=1 to kd-2 do
   begin
     if A[k]>Max then Max:=A[k];
     if A[k]<Min then Min:=A[k];
     if B[k]>Max1 then Max1:=B[k];
```

```
    if B[k]<Min1 then Min1:=B[k];
   end;
MinA:=Min;
MinB:=Min1;
 pr:=Max-Min;
 Mx:=290/pr;
 pr1:=Max1-Min1;
 My:=270/pr1;
 ShA:=pr/10;
 ShB:=pr1/10;
 KA:=0;
 KB:=0;
if Min>pr then KA:=1;
if Min1>pr1 then KB:=1;
end;

procedure OsiCoord;
var i:integer;
begin
 X0:=35;
 Y0:=300;
Form4.Image1.Canvas.MoveTo(X0,Y0+10);
Form4.Image1.Canvas.LineTo(X0,5);
Form4.Image1.Canvas.MoveTo(X0-10,Y0);
Form4.Image1.Canvas.LineTo(370,Y0);
for i:=1 to 11 do
 begin
   Form4.Image1.Canvas.MoveTo(X0+10+29*(i-1),Y0-2);
   Form4.Image1.Canvas.LineTo(X0+10+29*(i-1),Y0+2);
   Form4.Image1.Canvas.MoveTo(X0+2,Y0-10-27*(i-1));
   Form4.Image1.Canvas.LineTo(X0-2,Y0-10-27*(i-1));
 end;
end;

procedure RazmetkaOsey(A,B:TS);
var i:integer;
    q,q1:string;
    Mx,My,KA,KB,MinA,MinB,ShA,ShB:Real;
begin
Scale(A,B,Mx,My,KA,KB,MinA,MinB,ShA,ShB);
 for i:=0 to 5  do
  begin
   q:=format('%*.*f',[4,3,MinA]);
   Form4.Image1.Canvas.TextOut(X0+29*2*i+2,Y0+4,q) ;
   MinA:=MinA+2*ShA;
  end;
  for i:=0 to 11  do
  begin
   q1:=format('%*.*f',[4,3,MinB]);
   Form4.Image1.Canvas.TextOut(X0-30,Y0-14-27*i,q1) ;
```

```
   MinB:=MinB+ShB;
  end;
end;

procedure Griphic_Lg_D_LgT;
var Mx,My,KA,KB,MinA,MinB,ShA,ShB:Real;
    i:integer;
begin
RazmetkaOsey(LgT,Lg_D);
Scale(LgT,Lg_D,Mx,My,KA,KB,MinA,MinB,ShA,ShB);
Form4.Image1.Canvas.MoveTo
     (X0+round(LgT[0]*Mx-KA*MinA{4.9}*Mx+10),
     Y0-(round(Lg_D[0]*My-{KB*}{0.3}MinB*My{177.8}))-10);
for i:=1 to kd-2 do
 begin
  Form4.Image1.Canvas.LineTo
     (X0+round(LgT[i]*Mx-KA*MinA{4.9}*Mx+10{4.6*MSx{181.8-4.6*181.8}),
      Y0-(round(Lg_D[i]*My-{KB*}MinB{0.3}*My{177.8}))-10);
 end;
for i:=0 to kd-2 do
 begin
  Form4.Image1.Canvas.Pen.Color:=clRed;
  Form4.Image1.Canvas.Ellipse(
      X0+round(LgT[i]*Mx-KA*MinA{4.9}*Mx)-2+10,
      Y0-round(Lg_D[i]*My-{KB*}{0.3}MinB*My)-2-10,
      X0+round(LgT[i]*Mx-KA*MinA{4.9}*Mx)+2+10,
      Y0-round(Lg_D[i]*My-{KB*}MinB{0.3}*My)+2-10
                          );
 end;

   Form4.Image1.Canvas.Pen.Color:=clBlack;
end;

procedure Griphic_LgEta_LgT;
var Mx,My,KA,KB,MinA,MinB,ShA,ShB:Real;
    i:integer;
begin
RazmetkaOsey(LgT,LgEta);
Scale(LgT,LgEta,Mx,My,KA,KB,MinA,MinB,ShA,ShB);
Form4.Image1.Canvas.MoveTo(
            X0+round(LgT[0]*Mx-KA*MinA*Mx)+10,
            Y0-round(LgEta[0]*My-{KB*}MinB*My)-10
                        );
for i:=1 to kd-2 do
 begin
  Form4.Image1.Canvas.LineTo
     (X0+round(LgT[i]*Mx-KA*MinA*Mx)+10,
      Y0-round(LgEta[i]*My-{KB*}MinB*My)-10);
 end;
 for i:=0 to kd-2 do
```

```
 begin
  Form4.Image1.Canvas.Pen.Color:=clRed;
  Form4.Image1.Canvas.Ellipse(
      X0+round(LgT[i]*Mx-KA*MinA{4.9}*Mx)-2+10,
      Y0-round(LgEta[i]*My-{KB*}{0.3}MinB*My)-2-10,
      X0+round(LgT[i]*Mx-KA*MinA{4.9}*Mx)+2+10,
      Y0-round(LgEta[i]*My-{KB*}MinB{0.3}*My)+2-10
                          );
 end;

   Form4.Image1.Canvas.Pen.Color:=clBlack;
end;

procedure Griphic_Eta_T;
var Mx,My,KA,KB,MinA,MinB,ShA,ShB:Real ;
    i:integer;
    T:TS;
begin
for i:=0 to kd-2 do
 begin
  T[i]:=power(10,LgT[i]);
 end;
RazmetkaOsey(T,Eta);
Scale(T,Eta,Mx,My,KA,KB,MinA,MinB,ShA,ShB);
Form4.Image1.Canvas.MoveTo
     (X0+round(T[0]*Mx-KA*MinA*Mx)+10,
     Y0-round(Eta[0]*My-{KB*}MinB*My{177.8})-10);
for i:=1 to kd-2 do
 begin
  Form4.Image1.Canvas.LineTo
     (X0+round(T[i]*Mx-KA*MinA*Mx)+10,
      Y0-round(Eta[i]*My-{KB*}MinB*My)-10);
 end;

  for i:=0 to kd-2 do
 begin
  Form4.Image1.Canvas.Pen.Color:=clRed;
  Form4.Image1.Canvas.Ellipse(
      X0+round(T[i]*Mx-KA*MinA{4.9}*Mx)-2+10,
      Y0-round(Eta[i]*My-{KB*}{0.3}MinB*My)-2-10,
      X0+round(T[i]*Mx-KA*MinA{4.9}*Mx)+2+10,
      Y0-round(Eta[i]*My-{KB*}MinB{0.3}*My)+2-10
                          );
 end;
   Form4.Image1.Canvas.Pen.Color:=clBlack;
end;

procedure Points_Lg_D_LgT;
var Mx,My,KA,KB,MinA,MinB,ShA,ShB:Real;
    i:integer;
```

```
begin
RazmetkaOsey(LgT,Lg_D);
Scale(LgT,Lg_D,Mx,My,KA,KB,MinA,MinB,ShA,ShB);
for i:=0 to kd-2 do
 begin
  Form4.Image1.Canvas.Pen.Color:=clRed;
  Form4.Image1.Canvas.Ellipse(
     X0+round(LgT[i]*Mx-KA*MinA{4.9}*Mx)-2+10,
     Y0-round(Lg_D[i]*My-{KB*}{0.3}MinB*My)-2-10,
     X0+round(LgT[i]*Mx-KA*MinA{4.9}*Mx)+2+10,
     Y0-round(Lg_D[i]*My-{KB*}MinB{0.3}*My)+2-10
                          );
 end;
   Form4.Image1.Canvas.Pen.Color:=clBlack;
end;

procedure Points_LgEta_LgT;
var Mx,My,KA,KB,MinA,MinB,ShA,ShB:Real;
    i:integer;
begin
kv:=kv+1;
if kv=1 then RazmetkaOsey(LgT,LgEta);
if kv=3 then RazmetkaOsey(LgT,LgEta123);
if kv=1 then Scale(LgT,LgEta,Mx,My,KA,KB,MinA,MinB,ShA,ShB);
if kv=3 then Scale(LgT,LgEta123,Mx,My,KA,KB,MinA,MinB,ShA,ShB);
 for i:=0 to kd-2 do
 begin
  if kv=1 then Form4.Image1.Canvas.Pen.Color:=clRed;
  if kv=2 then Form4.Image1.Canvas.Pen.Color:=clGreen;
  if kv=3 then Form4.Image1.Canvas.Pen.Color:=clBlue;
  Form4.Image1.Canvas.Pen.width:=2;
  Form4.Image1.Canvas.Ellipse(
     X0+round(LgT[i]*Mx-KA*MinA{4.9}*Mx)-3+10,
     Y0-round(LgEta[i]*My-{KB*}{0.3}MinB*My)-3-10,
     X0+round(LgT[i]*Mx-KA*MinA{4.9}*Mx)+3+10,
     Y0-round(LgEta[i]*My-{KB*}MinB{0.3}*My)+3-10
                          );
 end;

   Form4.Image1.Canvas.Pen.Color:=clBlack;
end;

procedure Points_Eta_T;
var Mx,My,KA,KB,MinA,MinB,ShA,ShB:Real ;
    i:integer;
    T:TS;
begin
for i:=0 to kd-2 do
 begin
  T[i]:=power(10,LgT[i]);
```

```
 end;
RazmetkaOsey(T,Eta);
Scale(T,Eta,Mx,My,KA,KB,MinA,MinB,ShA,ShB);
  for i:=0 to kd-2 do
 begin
  Form4.Image1.Canvas.Pen.Color:=clRed;
  Form4.Image1.Canvas.Ellipse(
      X0+round(T[i]*Mx-KA*MinA{4.9}*Mx)-2+10,
      Y0-round(Eta[i]*My-{KB*}{0.3}MinB*My)-2-10,
      X0+round(T[i]*Mx-KA*MinA{4.9}*Mx)+2+10,
      Y0-round(Eta[i]*My-{KB*}MinB{0.3}*My)+2-10
                              );
 end;
   Form4.Image1.Canvas.Pen.Color:=clBlack;
end;

procedure Points_3_LgEta_LgT;
var Mx,My,KA,KB,MinA,MinB,ShA,ShB:Real;
    i:integer;
begin
RazmetkaOsey(LgT,LgEta123);
Scale(LgT,LgEta123,Mx,My,KA,KB,MinA,MinB,ShA,ShB);
 for i:=0 to kd-2 do
 begin
  Form4.Image1.Canvas.Pen.Color:=clRed;
  Form4.Image1.Canvas.Pen.width:=2;
  Form4.Image1.Canvas.Ellipse(
      X0+round(LgT[i]*Mx-KA*MinA{4.9}*Mx)-3+10,
      Y0-round(LgEta[i]*My-{KB*}{0.3}MinB*My)-3-10,
      X0+round(LgT[i]*Mx-KA*MinA{4.9}*Mx)+3+10,
      Y0-round(LgEta[i]*My-{KB*}MinB{0.3}*My)+3-10
                              );
 end;
   Form4.Image1.Canvas.Pen.Color:=clBlack;
end;

procedure TForm4.Button1Click(Sender: TObject);
begin
OsiCoord;
if RadioButton1.Checked then Griphic_Lg_D_LgT;
if RadioButton2.Checked then Griphic_LgEta_LgT;
if RadioButton3.Checked then Griphic_Eta_T;
//RazmetkaOsey(LgT,Lg_D);
end;

procedure Approx_LgEta_LgT;
var Mx,My,KA,KB,MinA,MinB,ShA,ShB:real;
    i:integer;
    Ma,Mb,hA,hB, Ac,Bc,Cc:real;
begin
```

```
RazmetkaOsey(LgT,LgEta);
Scale(LgT,LgEta,Mx,My,KA,KB,MinA,MinB,ShA,ShB);
MinSq(LgT,LgEta, Ac,Bc,Cc);
Form4.Image1.Canvas.MoveTo(X0+10,Y0-10-round(
                    (Ac*MinA*MinA+Bc*MinA+Cc)*My-MinB*My)
                              );
 Form4.Edit1.Text:=FloatToStr(Ac);
 Form4.Edit2.Text:=FloatToStr(Bc);
 Form4.Edit3.Text:=FloatToStr(Cc);
for i:=1 to 101 do
 begin
   Form4.Image1.Canvas.LineTo
     (X0+round((MinA+i*ShA/10)*Mx-MinA*Mx)+10,
      Y0-round((Ac*(MinA+i*ShA/10)*(MinA+i*ShA/10)+
                    Bc*(MinA+i*ShA/10)+Cc)*My-
                    MinB*My)-10);
 end;
 for i:=0 to kd-2 do
 begin
  Form4.Image1.Canvas.Pen.Color:=clRed;
  Form4.Image1.Canvas.Ellipse(
      X0+round(LgT[i]*Mx-MinA*Mx)-2+10,
      Y0-round(LgEta[i]*My-MinB*My)-2-10,
      X0+round(LgT[i]*Mx-KA*MinA*Mx)+2+10,
      Y0-round(LgEta[i]*My-MinB*My)+2-10
                          );
 end;
   Form4.Image1.Canvas.Pen.Color:=clBlack;
end;


procedure Approx4_LgEta_LgT;
var Mx,My,KA,KB,MinA,MinB,ShA,ShB:real;
    i:integer;
    Ma,Mb,hA,hB, Ac4,Bc4,Cc4,Dc4:real;
begin
RazmetkaOsey(LgT,LgEta);
Scale(LgT,LgEta,Mx,My,KA,KB,MinA,MinB,ShA,ShB);
MinSq4(LgT,LgEta, Ac4,Bc4,Cc4,Dc4);

Form4.Image1.Canvas.MoveTo(X0+10,Y0-10-round(
                    (Ac4*MinA*MinA*MinA+Bc4*MinA*MinA+Cc4*MinA+Dc4)*My
                          -MinB*My)
                              );
 Form4.Edit1.Text:=FloatToStr(Ac4);
 Form4.Edit2.Text:=FloatToStr(Bc4);
 Form4.Edit3.Text:=FloatToStr(Cc4);
for i:=1 to 101 do
 begin
   Form4.Image1.Canvas.LineTo
```

```
    (X0+round((MinA+i*ShA/10)*Mx-MinA*Mx)+10,
     Y0-round((Ac4*(MinA+i*ShA/10)*(MinA+i*ShA/10)*(MinA+i*ShA/10)+
            Bc4*(MinA+i*ShA/10)*(MinA+i*ShA/10)+
            Cc4*(MinA+i*ShA/10)+Dc4)*My-
                MinB*My)-10);
 end;
 for i:=0 to kd-2 do
 begin
  Form4.Image1.Canvas.Pen.Color:=clRed;
  Form4.Image1.Canvas.Ellipse(
     X0+round(LgT[i]*Mx-MinA*Mx)-2+10,
     Y0-round(LgEta[i]*My-MinB*My)-2-10,
     X0+round(LgT[i]*Mx-KA*MinA*Mx)+2+10,
     Y0-round(LgEta[i]*My-MinB*My)+2-10
                    );
 end;
   Form4.Image1.Canvas.Pen.Color:=clBlack;
end;
procedure Approx_Eta_T;
var Mx,My,KA,KB,MinA,MinB,ShA,ShB:real;
    i:integer;
    Ma,Mb,hA,hB, Ac,Bc,Cc:real;
    T:TS;
begin
for i:=0 to kd-2 do
 begin
  T[i]:=power(10,LgT[i]);
 end;

RazmetkaOsey(T,Eta);
Scale(T,Eta,Mx,My,KA,KB,MinA,MinB,ShA,ShB);
MinSq(T,Eta, Ac,Bc,Cc);
Form4.Image1.Canvas.MoveTo(X0+10,Y0-10-round(
                (Ac*MinA*MinA+Bc*MinA+Cc)*My-MinB*My)
                        );
 Form4.Edit1.Text:=FloatToStr(Ac);
 Form4.Edit2.Text:=FloatToStr(Bc);
 Form4.Edit3.Text:=FloatToStr(Cc);
for i:=1 to 101 do
 begin
   Form4.Image1.Canvas.LineTo
     (X0+round((MinA+i*ShA/10)*Mx-MinA*Mx)+10,
      Y0-round((Ac*(MinA+i*ShA/10)*(MinA+i*ShA/10)+
                Bc*(MinA+i*ShA/10)+Cc)*My-
                MinB*My)-10);
 end;
 for i:=0 to kd-2 do
 begin
  Form4.Image1.Canvas.Pen.Color:=clRed;
  Form4.Image1.Canvas.Ellipse(
```

```
      X0+round(T[i]*Mx-MinA*Mx)-2+10,
      Y0-round(Eta[i]*My-MinB*My)-2-10,
      X0+round(T[i]*Mx-KA*MinA*Mx)+2+10,
      Y0-round(Eta[i]*My-MinB*My)+2-10
                              );
 end;
   Form4.Image1.Canvas.Pen.Color:=clBlack;
end;


procedure Approx_Lg_D_LgT;
var Mx,My,KA,KB,MinA,MinB,ShA,ShB:real;
    i:integer;
    Ma,Mb,hA,hB, Ac,Bc,Cc:real;
begin
RazmetkaOsey(LgT,Lg_D);
Scale(LgT,Lg_D,Mx,My,KA,KB,MinA,MinB,ShA,ShB);
Form4.Image1.Canvas.MoveTo(
            X0+round(LgT[0]*Mx-MinA*Mx)+10,
            Y0-round(Lg_D[0]*My-MinB*My)-10
                          );
MinSq(LgT,Lg_D, Ac,Bc,Cc);
 Form4.Edit1.Text:=FloatToStr(Ac);
 Form4.Edit2.Text:=FloatToStr(Bc);
 Form4.Edit3.Text:=FloatToStr(Cc);
for i:=1 to 99 do
 begin
   Form4.Image1.Canvas.LineTo
     (X0+round((MinA+i*ShA/10)*Mx-MinA*Mx)+10,
      Y0-round((Ac*(MinA+i*ShA/10)*(MinA+i*ShA/10)+
                     Bc*(MinA+i*ShA/10)+Cc)*My-
                     MinB*My)-10);
 end;
 for i:=0 to 9 do
 begin
  Form4.Image1.Canvas.Pen.Color:=clRed;
  Form4.Image1.Canvas.Ellipse(
      X0+round(LgT[i]*Mx-MinA*Mx)-2+10,
      Y0-round(Lg_D[i]*My-MinB*My)-2-10,
      X0+round(LgT[i]*Mx-KA*MinA*Mx)+2+10,
      Y0-round(Lg_D[i]*My-MinB*My)+2-10
                              );
 end;
   Form4.Image1.Canvas.Pen.Color:=clBlack;
end;

Unit6
procedure TForm6.FormCreate(Sender: TObject);
begin
x0:=-1000;
```

```
y0:=2200;
Mx:=300;
My:=300;
kn:=0;
end;

function xk(xmat:real):integer;
begin
 xk:=trunc(x0+xmat*Mx);
end;

function yk(ymat:real):integer;
begin
 yk:=trunc(y0-ymat*My);
end;

procedure osi;
var i:integer;
 begin
  Form6.Image1.Canvas.Pen.Width:=1;
  //ShowMessage("0");
  Form6.Image1.Canvas.Pen.Color:=clWhite;
  Form6.Image1.Canvas.Rectangle(0,0,Form6.Image1.Width,Form6.Image1.Height);
  Form6.Image1.Canvas.Pen.Color:=clMoneyGreen;
    for i:=0 to 100 do
  begin
   Form6.Image1.Canvas.MoveTo(5,y0+i*My);
   Form6.Image1.Canvas.LineTo(Form6.Image1.Width-5,y0+i*My);
   Form6.Image1.Canvas.MoveTo(5,y0-i*My);
   Form6.Image1.Canvas.LineTo(Form6.Image1.Width-5,y0-i*My);
   Form6.Image1.Canvas.MoveTo(x0+i*Mx,5);
   Form6.Image1.Canvas.LineTo(x0+i*Mx,Form6.Image1.Height);
   Form6.Image1.Canvas.MoveTo(x0-i*Mx,5);
   Form6.Image1.Canvas.LineTo(x0-i*Mx,Form6.Image1.Height);
  end;
  Form6.Image1.Canvas.Pen.Color:=clBlack;
  Form6.Image1.Canvas.MoveTo(5,y0);
  Form6.Image1.Canvas.LineTo(Form6.Image1.Width-5,y0);
  Form6.Image1.Canvas.MoveTo(x0,5);
  Form6.Image1.Canvas.LineTo(x0,Form6.Image1.Height);
 end;

procedure vvod(c:STRING; var F_Tcp,F_S,F_LgT:TS );
var f:textfile;
i,j:integer;
begin
       assignfile(f,c);
    reset(f);
    readln(f,kd);
    //ShowMessage(IntToStr(kd));
```

```
    for i:=0 to kd-1 do
    begin
     for j:=1 to kd do
      read(f,F_Tcp[i]);
      read(f,F_S[i]);
      read(f,F_LgT[i]);
      readln(f);
    end;
end;

procedure TForm6.Button1Click(Sender: TObject);
var i:integer;
    f:textfile;
begin
assignfile(f,'outinp.txt');
    reset(f);
      kd:=11;
       for i:=0 to kd-3 do
    begin
    for j:=1 to kd do
      readln(f,LgT[i]);
        readln(f);
      end;
    for i:=0 to kd-3 do
    begin
      read(f,LgEta1[i]);
    readln(f);
      end;
      for i:=0 to kd-3 do
    begin
      read(f,LgEta2[i]);
     readln(f);
      end;
      for i:=0 to kd-2 do
    begin
      read(f,LgEta3[i]);
      readln(f);
      end;
kn:=kn+1;
      osi;
      Form6.Image1.Canvas.Pen.width:=2;
 Form6.Image1.Canvas.Pen.Color:=clRed;
 for i:=0 to kd-3 do
    begin
    Form6.Image1.Canvas.Ellipse(xk(LgT[i])-3,yk(LgEta1[i])-
3,xk(LgT[i])+3,yk(LgEta1[i])+3);
    end;
    Form6.Image1.Canvas.Pen.Color:=clGreen;
  for i:=0 to kd-3 do
    begin
```

```
    Form6.Image1.Canvas.Ellipse(xk(LgT[i])-3,yk(LgEta2[i])-
3,xk(LgT[i])+3,yk(LgEta2[i])+3);
    end;
    Form6.Image1.Canvas.Pen.Color:=clBlue;
  for i:=0 to kd-3 do
    begin
    Form6.Image1.Canvas.Ellipse(xk(LgT[i])-3,yk(LgEta3[i])-
3,xk(LgT[i])+3,yk(LgEta3[i])+3);
    end;
end;

procedure TForm6.ScrollBar1Scroll(Sender: TObject; ScrollCode: TScrollCode;
  var ScrollPos: Integer);
  var i:integer;
begin
  Form6.Image1.Canvas.Pen.Color:=clWhite;
  Form6.Image1.Canvas.Rectangle(0,0,Image1.Width,Image1.Height);
Mx:=ScrollPos;
My:=ScrollPos;
osi;
for i:=0 to kd-1 do
 begin
   Form6.Image1.Canvas.Pen.Color:=clRed;
  Form6.Image1.Canvas.Pen.width:=2;
  Form6.Image1.Canvas.Ellipse(xk(LgT[i])-3,yk(LgEta1[i])-
3,xk(LgT[i])+3,yk(LgEta1[i])+3);
   Form6.Image1.Canvas.Pen.Color:=clGreen;
   Form6.Image1.Canvas.Ellipse(xk(LgT[i])-3,yk(LgEta2[i])-
3,xk(LgT[i])+3,yk(LgEta2[i])+3);
   Form6.Image1.Canvas.Pen.Color:=clBlue;
   Form6.Image1.Canvas.Ellipse(xk(LgT[i])-3,yk(LgEta3[i])-
3,xk(LgT[i])+3,yk(LgEta3[i])+3);
   end;
end;
end.
```

Basic procedures and functions for graphical display of calculation results by mathematical model of polymer droplet deformation in flow

```
Unit3
procedure TForm3.FormShow(Sender: TObject);
Var
    i:integer;
    SX,SIX,SIY:Real;
    StepX,StepY:Real;
    OldX,OldY,NewX,NewY:integer;
    MasX:array [0..50] of Real;
    MasY:array [1..4,0..50] of Real;
    BY:Real;
    SY:Real;
     MaxY:integer;
    Z:integer;
    iii:integer;
    countL:integer;
    index:integer;
begin
    for i:=1 to 50 do
    MasX[i-1]:= strtofloat(form1.Memo1.Lines.Strings[i]);
    for i:=1 to form1.Memo3.Lines.Count-1 do
    begin
        if (i<51) then MasY[1][i-1]:= strtofloat(form1.Memo3.Lines.Strings[i]);
        if (i>=101) and (i<151) then MasY[2][i-101]:=
strtofloat(form1.Memo3.Lines.Strings[i]);
        if (i>=201) and(i<251) then MasY[3][i-201]:=
strtofloat(form1.Memo3.Lines.Strings[i]);
        if (i>=301) and (i<351) then MasY[4][i-301]:=
strtofloat(form1.Memo3.Lines.Strings[i]);
    end;
    Image1.Canvas.Pen.Color:=clBlack;
    Image1.Canvas.MoveTo(100,370);
    Image1.Canvas.LineTo(500,370);
    Image1.Canvas.MoveTo(100,370);
    Image1.Canvas.LineTo(100,0);
    Image1.Canvas.Pen.Color:=clBlack;
    SX:=MasX[49]-MasX[0];
    SY:=MasY[1][49]-MasY[1][0];
    if SY<MasY[2][49]-MasY[2][0] then SY:=MasY[2][49]-MasY[2][0];
    if SY<MasY[3][49]-MasY[3][0] then SY:=MasY[3][49]-MasY[3][0];
    if SY<MasY[4][49]-MasY[4][0] then SY:=MasY[4][49]-MasY[4][0];
    index:=1;
    while index<=CountI do
    for index:=1 to 4 do
    begin
    NewX:=round(100+MasX[0]);
```

```
NewY:=round(370-MasY[index][0]);
Image1.Canvas.Pen.Width:=2;
Image1.Canvas.MoveTo(NewX,NewY);
Image1.Canvas.Pen.Color:=clRed;
Image1.Canvas.LineTo(NewX,NewY);
Image1.Canvas.TextOut(80,NewY,floattostr(round(MasY[index][0])));
Image1.Canvas.Pen.Width:=1;
Image1.Canvas.MoveTo(80,NewY);
Image1.Canvas.LineTo(100,NewY);
Image1.Canvas.TextOut(NewX,380,floattostr((MasX[0])));
Image1.Canvas.MoveTo(NewX,370);
Image1.Canvas.LineTo(NewX,380);
Image1.Canvas.MoveTo(NewX,NewY);
Image1.Canvas.Pen.Color:=clBlack;
for i:=1 to 49 do
begin
Image1.Canvas.Pen.Width:=2;
     SIX:=MasX[i]-MasX[i-1];
     SIY:=MasY[index][i]-MasY[index][i-1];
     StepX:=(350*SIX)/SX;
     StepY:=(350*SIY)/SY;
     NewX:=Round(NewX+StepX);
     NewY:=Round(NewY-StepY);
     Image1.Canvas.LineTo(NewX,NewY);
     Image1.Canvas.Pen.Width:=5;
     Image1.Canvas.Pen.Color:=clRed;
     Image1.Canvas.LineTo(NewX,NewY);
     if i = 49 then
     begin
     Image1.Canvas.Pen.Width:=1;
     Image1.Canvas.TextOut(60,NewY-5,floattostr(round(MasY[index][i])));
    Image1.Canvas.MoveTo(80,NewY);
    Image1.Canvas.LineTo(100,NewY);
    Z:=round(MasY[index][i]);
     if MaxY>NewY then MaxY:=NewY;
     end;
     if i mod 10 = 0 then
     begin
     Image1.Canvas.Pen.Width:=1;
     Image1.Canvas.TextOut(NewX,380,floattostr(((MasX[i])*100000000)));
     Image1.Canvas.MoveTo(NewX,370);
     Image1.Canvas.LineTo(NewX,380);
     Image1.Canvas.MoveTo(NewX,NewY);
     end;
end;
if index=1 then
Image1.Canvas.TextOut(NewX+10,NewY,'Teta1 = ' +form8.edit1.text);
     if index=2 then
Image1.Canvas.TextOut(NewX+10,NewY,'Teta2 = ' +form8.edit4.text);
     if index=3 then
```

```
     Image1.Canvas.TextOut(NewX+10,NewY,'Teta3 = ' +form8.edit5.text);
          if index=4 then
     Image1.Canvas.TextOut(NewX+10,NewY,'Teta4 = ' +form8.edit6.text);
     inc(index);
     end;
    iii:=0;
          countL:=0;
     while iii<round(370-maxy) do
     begin
          iii:=iii+round((370-maxy)/20);
          Image1.Canvas.Pen.Width:=1;
          Z:=round((MaxY-iii)/MaxY);
          if countL mod 2 =0 then
          begin
          Image1.Canvas.MoveTo(90,370-iii);
          Image1.Canvas.LineTo(100,370-iii);
          end
          else
          begin
          Image1.Canvas.MoveTo(80,370-iii);
          Image1.Canvas.LineTo(100,370-iii);
          Image1.Canvas.TextOut(80,370-iii,floattostr(round((iii)*Z/(370-MaxY))));
          end;
         inc(countL);
     end;
end;

procedure TForm3.Button2Click(Sender: TObject);
begin
Image1.canvas.fillrect(Image1.canvas.cliprect);
form1.Memo3.Clear;
form1.Memo3.Text:='Пусто';
Image1.canvas.fillrect(Image1.canvas.cliprect);
Label1.Visible:=false;
Label2.Visible:=false;
  form1.Fr:=0;
  form1.eta:=0;
  form1.mu:=0;
  form1.R0:=0;
  form1.R0_3:=0;
  form1.d:=0;
  form1.d_:=0;
  form1.Teta0:=0;
  form1.CurrTeta:=0;
  form1.h:=0;
  form1.Q0:=0;
  form1.CurrQ:=0;
  Sigma:=0
end;
```

```
procedure TForm3.Save1Click(Sender: TObject);
begin
     if      SaveDialog1.Execute then
     Image1.Picture.SaveToFile(SaveDialog1.FileName+'.bmp');
end;

procedure TForm3.Print1Click(Sender: TObject);
var
 X1,X2,Y1,Y2:Integer;
 PointsX,PointsY:double;
 PrintDlg:TPrintDialog;
begin
 PrintDlg:=TPrintDialog.Create(Owner);
 if PrintDlg.Execute then
  begin
   Printer.BeginDoc;
   Printer.Canvas.Refresh;
   Printer.Title:='Results';
   PointsX:=GetDeviceCaps(Printer.Canvas.Handle,LOGPIXELSX)/100;
   PointsY:=GetDeviceCaps(Printer.Canvas.Handle,LOGPIXELSY)/100;
   X1:=50;
   Y1:=500;
   X2:=round(X1+Image1.Picture.Bitmap.Width*PointsX);
   Y2:=round(Y1+Image1.Picture.Bitmap.Height*PointsY);
   Printer.Canvas.CopyRect(Rect(X1,Y1,X2,Y2),Image1.Picture.Bitmap.Canvas,
           Rect(0,0,Image1.Picture.Bitmap.Width,Image1.Picture.Bitmap.Height));
   Printer.EndDoc;
  end;
 PrintDlg.Free;
 end;
end.

 Unit7
procedure TForm7.Button1Click(Sender: TObject);
begin
     form9.CountI:=0;
  if Edit1.Text<>'' then
  begin
   inc(form9.CountI);
  form1.Fr:=StrToFloat(form1.Edit4.Text);
  form1.eta:=StrToFloat(Edit3.Text);
  form1.mu:=StrToFloat(Edit2.Text);
  Form1.Edit9 .Text:=FloatToStr(form1.eta/form1.mu);
 V:=StrToFloat(Edit1.Text);
 R0_3:=V/((4/3)*3.1415);
R0:=power(R0_3,1/3);
form1.R0:=StrToFloat(form1.Edit12.Text);
 form1.R0_3:=form1.R0*form1.R0*form1.R0;
form1.V:=(4/3)*3.1415*form1.R0_3;
Form1.Edit1.Text:=FloatToStr(form1.V);
```

```
 form1.d:=StrToFloat(form1.Edit10.Text);
 form1.d_:=StrToFloat(form1.Edit11.Text);
 form1.G:=G;
   form1.Teta0:=StrToFloat(Edit7.Text);
    form1.CurrTeta:=form1.Teta0;
    form1.h:=StrToFloat(form1.Edit8.Text);
     form1.Q0:=StrToFloat(form1.Edit5.Text);
     form1.CurrQ:=form1.Q0;
           Sigma:=StrToFloat(Edit1.text);
           K:=0.15e8;
           G:=(Sigma*K/form1.R0)*(power(form1.CurrQ,2/3)/(1-
form1.Q0/(form1.CurrQ+0.0000001)));
            form1.G:=G;
            Form7.Edit1.Text :=FloatToStr(G);
            form1.G:=StrToFloat(Edit1.Text);
 form1.b0:=form1.R0*power(form1.CurrQ,(-1/3));
 form1.a0:=form1.R0*power(form1.CurrQ,2/3);
 A0:=StrToFloat(Edit5.Text);
  CurrA:=A0;
   B0:=power(R0*R0*R0/A0,1/2);
  q0:=A0/B0;}
  Form1.Button5.Click;
  end;
  if Edit4.Text<>'' then
  begin
   inc(form9.CountI);
  form1.Fr:=StrToFloat(form1.Edit4.Text);
  form1.eta:=StrToFloat(Edit3.Text);
  form1.mu:=StrToFloat(Edit2.Text);
  Form1.Edit9 .Text:=FloatToStr(form1.eta/form1.mu);
 V:=StrToFloat(Edit1.Text);
 R0_3:=V/((4/3)*3.1415);
R0:=power(R0_3,1/3);
form1.R0:=StrToFloat(form1.Edit12.Text);
 form1.R0_3:=form1.R0*form1.R0*form1.R0;
form1.V:=(4/3)*3.1415*form1.R0_3;
Form1.Edit1.Text:=FloatToStr(form1.V);
 form1.d:=StrToFloat(form1.Edit10.Text);
 form1.d_:=StrToFloat(form1.Edit11.Text);
     form1.G:=G;
   form1.G:=StrToFloat(Edit4.Text);
   form1.Teta0:=StrToFloat(Edit7.Text);
    form1.CurrTeta:=form1.Teta0;
    form1.h:=StrToFloat(form1.Edit8.Text);
     form1.Q0:=StrToFloat(form1.Edit5.Text);
  form1.CurrQ:=form1.Q0;
            Sigma:=StrToFloat(edit4.text);
            K:=0.15e8;
            G:=(Sigma*K/form1.R0)*(power(form1.CurrQ,2/3)/(1-
form1.Q0/(form1.CurrQ+0.0000001)));
```

```
            form1.G:=G;
            Form7.Edit4.Text:=FloatToStr(G);
            form1.G:=StrToFloat(Edit4.Text);
   form1.b0:=form1.R0*power(form1.CurrQ,(-1/3));
   form1.a0:=form1.R0*power(form1.CurrQ,2/3);
    Form1.Button5.Click;  end;
 if Edit5.Text<>'' then
 begin
  inc(form9.CountI);
   form1.Fr:=StrToFloat(form1.Edit4.Text);
 form1.eta:=StrToFloat(Edit3.Text);
 form1.mu:=StrToFloat(Edit2.Text);
 Form1.Edit9 .Text:=FloatToStr(form1.eta/form1.mu);
V:=StrToFloat(Edit1.Text);
R0_3:=V/((4/3)*3.1415);
R0:=power(R0_3,1/3);
form1.R0:=StrToFloat(form1.Edit12.Text);
 form1.R0_3:=form1.R0*form1.R0*form1.R0;
form1.V:=(4/3)*3.1415*form1.R0_3;
Form1.Edit1.Text:=FloatToStr(form1.V);
form1.d:=StrToFloat(form1.Edit10.Text);
 form1.d_:=StrToFloat(form1.Edit11.Text);
   form1.G:=G;
  form1.G:=StrToFloat(Edit5.Text);
  form1.Teta0:=StrToFloat(Edit7.Text);
   form1.CurrTeta:=form1.Teta0;
   form1.h:=StrToFloat(form1.Edit8.Text);
   form1.Q0:=StrToFloat(form1.Edit5.Text);
 form1.CurrQ:=form1.Q0;
   Sigma:=StrToFloat(edit5.text);
    K:=0.15e8;
     G:=(Sigma*K/form1.R0)*(power(form1.CurrQ,2/3)/(1-
form1.Q0/(form1.CurrQ+0.0000001)));
            form1.G:=G;
            Form7.Edit5.Text:=FloatToStr(G);
            form1.G:=StrToFloat(Edit5.Text);
 form1.b0:=form1.R0*power(form1.CurrQ,(-1/3));
 form1.a0:=form1.R0*power(form1.CurrQ,2/3);
  A0:=StrToFloat(Edit5.Text);
 CurrA:=A0;
  B0:=power(R0*R0*R0/A0,1/2);
 q0:=A0/B0;
 Form1.Button5.Click;
 end;
 if Edit6.Text<>'' then
 begin
  inc(form9.CountI);
   form1.Fr:=StrToFloat(form1.Edit4.Text);
 form1.eta:=StrToFloat(Edit3.Text);
 form1.mu:=StrToFloat(Edit2.Text);
```

```
  Form1.Edit9 .Text:=FloatToStr(form1.eta/form1.mu);
 V:=StrToFloat(Edit1.Text);
 R0_3:=V/((4/3)*3.1415);
R0:=power(R0_3,1/3);
 form1.R0:=StrToFloat(form1.Edit12.Text);
 form1.R0_3:=form1.R0*form1.R0*form1.R0;
form1.V:=(4/3)*3.1415*form1.R0_3;
Form1.Edit1.Text:=FloatToStr(form1.V);
 form1.d:=StrToFloat(form1.Edit10.Text);
 form1.d_:=StrToFloat(form1.Edit11.Text);
     form1.G:=G;
   form1.G:=StrToFloat(Edit6.Text);
   form1.Teta0:=StrToFloat(Edit7.Text);
    form1.CurrTeta:=form1.Teta0;
    form1.h:=StrToFloat(form1.Edit8.Text);
     form1.Q0:=StrToFloat(form1.Edit5.Text);
  form1.CurrQ:=form1.Q0;
              Sigma:=StrToFloat(edit6.text);
              K:=0.15e8;
              G:=(Sigma*K/form1.R0)*(power(form1.CurrQ,2/3)/(1-
form1.Q0/(form1.CurrQ+0.0000001)));
              form1.G:=G;
form1.b0:=form1.R0*power(form1.CurrQ,(-1/3));
 form1.a0:=form1.R0*power(form1.CurrQ,2/3);
  Form1.Button5.Click;end;
end;

Unit8
procedure TForm8.Button1Click(Sender: TObject);
begin
  if Edit1.Text<>'' then
  begin
  inc(form3.CountI);
  form1.Fr:=StrToFloat(form1.Edit4.Text);
  form1.eta:=StrToFloat(Edit3.Text);
  form1.mu:=StrToFloat(Edit2.Text);
  Form1.Edit9 .Text:=FloatToStr(form1.eta/form1.mu);
 V:=StrToFloat(Edit1.Text);
R0:=power(R0_3,1/3);
 form1.R0:=StrToFloat(form1.Edit12.Text);
 form1.R0_3:=form1.R0*form1.R0*form1.R0;
form1.V:=(4/3)*3.1415*form1.R0_3;
Form1.Edit1.Text:=FloatToStr(form1.V);
 form1.d:=StrToFloat(form1.Edit10.Text);
 form1.d_:=StrToFloat(form1.Edit11.Text);
       Sigma:=StrToFloat(edit7.text);
       K:=0.15e8;
       G:=(Sigma*K/form1.R0)*(power(form1.CurrQ,2/3)/(1-
form1.Q0/(form1.CurrQ+0.0000001)));
    form1.G:=G;
```

```
   form1.Teta0:=StrToFloat(Edit1.Text);
    form1.CurrTeta:=form1.Teta0;
    form1.h:=StrToFloat(form1.Edit8.Text);
     form1.Q0:=StrToFloat(form1.Edit5.Text);
  form1.CurrQ:=form1.Q0;
 form1.b0:=form1.R0*power(form1.CurrQ,(-1/3));
 form1.a0:=form1.R0*power(form1.CurrQ,2/3);
  Form1.Button5.Click;
  end;
  if Edit4.Text<>'' then
  begin
    inc(form3.CountI);
  form1.Fr:=StrToFloat(form1.Edit4.Text);
  form1.eta:=StrToFloat(Edit3.Text);
  form1.mu:=StrToFloat(Edit2.Text);
  Form1.Edit9 .Text:=FloatToStr(form1.eta/form1.mu);
end;
end.
```

Unit9

```
procedure TForm9.Formshow(Sender: TObject);
Var
      i:integer;
      SX,SIX,SIY:Real;
      StepX,StepY:Real;
      OldX,OldY,NewX,NewY:integer;
      MasX:array [0..99] of Real;
      MasY:array [1..4,0..99] of Real;
      BY:Real;
      SY,Sigma,KG:Real;
      index:integer;
      MaxY:integer;
      Z:integer;
      iii:integer;
      countL:integer;
    begin
    for i:=1 to 100 do
      MasX[i-1]:= strtofloat(form1.Memo1.Lines.Strings[i]);
      for i:=1 to Form1.Memo3.Lines.Count-1 do
      begin
           if (i<101) then MasY[1][i-1]:= strtofloat(form1.Memo3.Lines.Strings[i]);
           if (i>=101) and (i<201) then MasY[2][i-101]:=
strtofloat(form1.Memo3.Lines.Strings[i]);
           if (i>=201) and(i<301) then MasY[3][i-201]:=
strtofloat(form1.Memo3.Lines.Strings[i]);
           if (i>=301) and (i<401) then MasY[4][i-301]:=
strtofloat(form1.Memo3.Lines.Strings[i]);
      end;
       Image1.Canvas.MoveTo(100,370);
```

```
Image1.Canvas.LineTo(500,370);
Image1.Canvas.MoveTo(100,370);
Image1.Canvas.LineTo(100,0);
Image1.Canvas.Pen.Color:=clBlack;
SX:=MasX[99]-MasX[0];
SY:=MasY[1][99]-MasY[1][0];
if SY<MasY[2][99]-MasY[2][0] then SY:=MasY[2][99]-MasY[2][0];
if SY<MasY[3][99]-MasY[3][0] then SY:=MasY[3][99]-MasY[3][0];
if SY<MasY[4][99]-MasY[4][0] then SY:=MasY[4][99]-MasY[4][0];
index:=1;
MaxY:=round(MasY[1][99]);
while index<=CountI do
for index:=1 to 4 do
begin
NewX:=round(100+MasX[0]);
NewY:=round(370-MasY[index][0]);
Image1.Canvas.Pen.Width:=5;
Image1.Canvas.MoveTo(NewX,NewY);
Image1.Canvas.Pen.Color:=clRed;
Image1.Canvas.LineTo(NewX,NewY);
Image1.Canvas.TextOut(80,NewY,floattostr(round(MasY[index][0])));
Image1.Canvas.Pen.Width:=1;
Image1.Canvas.MoveTo(80,NewY);
Image1.Canvas.LineTo(100,NewY);
Image1.Canvas.TextOut(NewX,380,floattostr((MasX[0])));
Image1.Canvas.MoveTo(NewX,370);
Image1.Canvas.LineTo(NewX,390);
Image1.Canvas.MoveTo(NewX,NewY);
for i:=1 to 99 do
begin
Image1.Canvas.Pen.Width:=3;
    SIX:=MasX[i]-MasX[i-1];
    SIY:=MasY[index][i]-MasY[index][i-1];
    StepX:=(350*SIX)/SX;
    StepY:=(350*SIY)/SY;
    NewX:=Round(NewX+StepX);
    NewY:=Round(NewY-StepY);
    Image1.Canvas.LineTo(NewX,NewY);
    Image1.Canvas.Pen.Width:=3;
    Image1.Canvas.Pen.Color:=clRed;
    Image1.Canvas.LineTo(NewX,NewY);
    if i = 99 then
    begin
    Image1.Canvas.Pen.Width:=1;
    Image1.Canvas.TextOut(80,NewY,floattostr(round(MasY[index][i])));
    Z:=round(MasY[index][i]);
    Image1.Canvas.MoveTo(80,NewY);
    Image1.Canvas.LineTo(100,NewY);
    if MaxY>NewY then MaxY:=NewY;
    end;
```

```
        if i mod 10 = 0 then
        begin
        Image1.Canvas.Pen.Width:=1;
        Image1.Canvas.TextOut(NewX,380,floattostr(((MasX[i])*100000000)));
        Image1.Canvas.MoveTo(NewX,370);
        Image1.Canvas.LineTo(NewX,380);
        Image1.Canvas.MoveTo(NewX,NewY);
        Image1.Canvas.Pen.Color:=clBlack;
        end;
    end;
    if index=1 then
    Image1.Canvas.TextOut(NewX+10,NewY,'Sigma1 = ' +form7.edit1.text);
        if index=2 then
    Image1.Canvas.TextOut(NewX+10,NewY,'Sigma2 = ' +form7.edit4.text);
        if index=3 then
    Image1.Canvas.TextOut(NewX+10,NewY,'Sigma3 = ' +form7.edit5.text);

        if index=4 then
    Image1.Canvas.TextOut(NewX+10,NewY,'Sigma4 = ' +form7.edit6.text);
    inc(index);
    end;
        iii:=0;
        countL:=0;
    while iii<round(370-maxy) do
    begin
        iii:=iii+round((370-maxy)/20);
        Image1.Canvas.Pen.Width:=1;
        Z:=round((MaxY-iii)/MaxY);
        if countL mod 2 =0 then
        begin
        Image1.Canvas.MoveTo(90,370-iii);
        Image1.Canvas.LineTo(100,370-iii);
        end
        else
        begin
        Image1.Canvas.MoveTo(80,370-iii);
        Image1.Canvas.LineTo(100,370-iii);
        Image1.Canvas.TextOut(80,370-iii,floattostr(round((iii)*Z/(370-MaxY))));
        end;
         inc(countL);
    end;
end;

procedure TForm9.Save1Click(Sender: TObject);
begin
 if     SaveDialog1.Execute then
    Image1.Picture.SaveToFile(SaveDialog1.FileName+'.bmp');
end;

procedure TForm9.Print1Click(Sender: TObject);
```

```
var
 X1,X2,Y1,Y2:Integer;
 PointsX,PointsY:double;
 PrintDlg:TPrintDialog;
begin
 PrintDlg:=TPrintDialog.Create(Owner);
 if PrintDlg.Execute then
  begin
   Printer.BeginDoc;
   Printer.Title:='Results';
   Printer.Canvas.Refresh;
   PointsX:=GetDeviceCaps(Printer.Canvas.Handle,LOGPIXELSX)/100;
   PointsY:=GetDeviceCaps(Printer.Canvas.Handle,LOGPIXELSY)/100;
   X1:=50;
   Y1:=500;
   X2:=round(X1+Image1.Picture.Bitmap.Width*PointsX);
   Y2:=round(Y1+Image1.Picture.Bitmap.Height*PointsY);
   Printer.Canvas.CopyRect(Rect(X1,Y1,X2,Y2),Image1.Picture.Bitmap.Canvas,
          Rect(0,0,Image1.Picture.Bitmap.Width,Image1.Picture.Bitmap.Height));
   Printer.EndDoc;
  end;
 PrintDlg.Free;
end;
end.
```

Наукове видання

Shcherban V.Yu., Rezanova V.G., Demkivska T.I.

PROGRAMMING OF NUMERICAL METHODS AND EXAMPLES OF PRACTICAL APPLICATION

Щербань Володимир Юрійович
Резанова Вікторія Георгієвна
Демківська Тетяна Іванівна

ПРОГРАМУВАННЯ ЧИСЕЛЬНИХ МЕТОДІВ ТА ПРИКЛАДИ ПРАКТИЧНОГО ЗАСТОСУВАННЯ
(*англійською мовою*)

Підписано до друку 15.12.2021 р.
Формат 60x84/16. Папір офсетний.
Ум. друк. арк. 8,83.
Наклад 100 прим.

ФО-П Маслаков Руслан Олексійович
Свідоцтво про внесення суб'єкта видавничої справи
до державного реєстру видавців, виготівників
і розповсюджувачів видавничої продукції
ДК №4726 від 29.05.2014 р.
Тел. (095) 699-25-20.
E-mail: osvita2005@gmail.com.

ВД «Освіта України»™