The monograph presents methods for mathematical planning of experimental studies and optimization of the composition of multicomponent mixture systems. The results on the development of new synthetic fibrous materials by introducing nanofillers of different chemical nature and form into their structure are summarized. The software created by the authors for constructing an experimental plan in the studied area of the factor space and optimizing the composition of any types of mixture systems of three- and four-component heterogeneous compositions is considered in detail. Specific examples of the use of the software for establishing the composition-property relationship in order to obtain fibrous materials with predicted characteristics from nanofilled polymer mixtures are presented.

The monograph can be useful for teachers, scientists, postgraduates, students in the following specialties: computer science, chemical technology as well as for a wide range of specialists working on research and creation of new composite materials.

978-617-8543-18-1

Rezanova V.G., Rezanova N.M.

OPTIMIZATION OF THE COMPOSITION OF MULTI-COMPONENT SYSTEMS: RESEARCH AND SOFTWARE

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
KYIV NATIONAL UNIVERSITY OF TECHNOLOGIES AND DESIGN

Rezanova V.G., Rezanova N.M.

OPTIMIZATION OF THE COMPOSITION OF MULTI-COMPONENT SYSTEMS: RESEARCH AND SOFTWARE

Recommended by the Academic Council of the Kyiv National University of Technologies and Design (Protocol № 2 of April, 30, 2025)

Kyiv -2025

UDK 004.42:[544+001.894]

Recommended by the Academic Council of the Kyiv National University of Technologies and Design for a wide range of researchers, teachers and engineers (Protocol № 2 of April, 30, 2025)

Authors:

REZANOVA V. G. – Candidate of Technical Sciences, Associate Professor of the Department of Computer Science, Kyiv National University of Technologies and Design;
REZANOVA N. M. – Laureate of the State Prize of Ukraine in the field of science and technology, Candidate of Technical Sciences

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;
CHUPRYNKA V. I. - Doctor of Technical Sciences, Professor of the Department of Computer Science, Kyiv National University of Technologies and Design

Rezanova V.G., Rezanova N.M. Optimization of the composition of multi-component systems: research and software. Monograph. – K.: ArtEk, 2025. – 258 p.

ISBN 978-617-8543-18-1

The monograph presents methods for mathematical planning of experimental studies and optimization of the composition of multicomponent mixture systems. The results on the development of new synthetic fibrous materials by introducing nanofillers of different chemical nature and form into their structure are summarized. The software created by the authors for constructing an experimental plan in the studied area of the factor space and optimizing the composition of any types of mixture systems of three- and four-component heterogeneous compositions is considered in detail. Specific examples of the use of the software for establishing the composition-property relationship in order to obtain fibrous materials with predicted characteristics from nanofilled polymer mixtures are presented.

The monograph can be useful for teachers, scientists, postgraduates, students in the following specialties: computer science, chemical technology as well as for a wide range of specialists working on research and creation of new composite materials.

ISBN 978-617-8543-18-1 UDK 004.42:[544+001.894]

INTRODUCTION

The rapid development of science and technology in the world, the constantly growing needs of mankind for goods with improved properties, as well as limited natural resources have led to the search for methods for obtaining new substances and materials with a given set of indicators. In Ukraine, in 2017-2021, under the target scientific research program of the NAS of Ukraine "New functional substances and materials of chemical production", modern scientific approaches were developed to create non-traditional materials with improved functional characteristics for various areas of practical application and to establish ways to control such properties [1]. Within the framework of the program, fundamental principles for obtaining substances and materials of a wide range of purposes based on new energy-saving environmentally friendly technologies for the needs of various industries and the social sphere have been developed: energy saving, micro- and nanoelectronics, transport, aircraft construction, agro-industrial complex, light and food industry, household chemicals, environmental protection, etc.

Today, the improvement of the standard of living of society and its sustainable development is largely achieved thanks to scientific progress in chemical materials science, in particular in the creation of fundamentally new polymer composite materials. At the beginning of the 3rd millennium, composites have gained importance in a wide variety of areas of human activity, revolutionizing technology, everyday life and lifestyle. Their practical value is due to the nonlinearity and synergy of properties that provide an advantage over other materials, namely: high thermal and corrosion resistance, low weight in combination with improved mechanical performance and low cost. Their areas of application have expanded from household goods (fabrics, textiles, knitwear, packaging, biomedical products) to high-tech products (for aerospace and military equipment, microelectronics, energy complex, metallurgy, construction, healthcare, a new generation of adsorbents for environmental protection). The possibilities of giving polymer products the desired characteristics are virtually unlimited thanks to a wide range of methods for their modification.

One of the most effective is the introduction of various additives into composites, especially substances in the nanoscale. Natural or specially synthesized substances of different sizes, geometric structure and chemical nature are used as nanoadditives, which are selected taking into account the achievement of the desired characteristics of composites, their cost, the possibility of recycling, the impact on biodegradability, etc. A significant number of requirements for nanofillers are satisfied by natural layered aluminosilicates, silicas, carbon derivatives (nanofibers, nanotubes, fullerenes), metal nanoparticles (NPs), their oxides, etc. Their use allows you to regulate the characteristics of polymeric materials and give them a set of desired properties. Nanocomposites containing additives of natural or modified clay demonstrate a sharp improvement in strength and modulus of elasticity, heat and fire resistance, and gas permeability [2-4]. The introduction of carbon nanotubes (CNTs) expands the range of applications in a wide variety of areas: as reinforced and anticorrosion materials, solar cells, chemical sensors, adsorbents, products for shielding from electromagnetic and microwave radiation, etc. [5-8]. Due to the unique graphitic structure and extraordinary biological properties, CNTs are of increasing interest in biomedicine (drug delivery, bioimaging, biosensor materials and tissue engineering) [7]. Synthetic fibers and threads containing noble metal or metal oxide nanoparticles in their structure acquire antimicrobial, anti-allergic, sorption and antistatic properties and protect against UV radiation [9-12]. The simultaneous use of two different or bicomponent nanoadditives is more effective. Polyvinyl alcohol nanofibers containing Ag/TiO_2 nanoparticles exhibit antimicrobial and photocatalytic activity [13]. Polypropylene monofilaments with Ag/SiO_2 nanoadditives have, along with bactericidal properties, high mechanical and manipulation characteristics [14]. Modification of polymers with nanoadditives also allows solving environmental and social problems [15-17]. Composites for water purification [15], new environmentally friendly adsorbents for environmental restoration [16,17], and materials for biomedical purposes [18] have been created based on biopolymers. The development and implementation of new, so-called “green” technologies allows recycling and using secondary polymers [19-21].

In nanofilled composites of incompatible polymers, in addition to the concentration and chemical nature of additives, an important factor is their uneven distribution in the volumes of component phases, which significantly expands the possibilities of regulating heterogeneous morphology and makes mixed systems even more attractive. Under the condition of selective localization of highly dispersed electrically conductive nanomodifiers in the interphase layer, the content of the additive necessary to reduce the percolation threshold is significantly reduced [22,23]. At the same time, the formation of a percolation mesh structure by carbon nanotubes in the polymer matrix also contributes to a significant increase in the elastic modulus. The addition of organoclay to the polyamide/polylactide (PA/PL) mixture caused a change in the type of PA structure in the matrix - from droplet-matrix to interwoven, as a result, the heat resistance and plasticity of the composite material increased without a negative impact on its stiffness and strength [24]. The preferential localization of aluminum and titanium oxide nanoparticles at the phase interface in the polypropylene/copolyamide (PP/SPA) and polyethylene terephthalate (PET)/PP mixtures, respectively, resulted in an improvement in the performance characteristics of composite yarns – an increase in their strength and dimensional stability due to a decrease in the diameters of in situ formed PP and PET microfibrils [25,26]. The effectiveness of nanoadditives in polymer mixtures increases with the introduction of substances that affect interfacial phenomena [27-29]. The addition of compatibilizers in the PP/SPA/CNT and PET/PP/TiO_2 mixtures contributed to the improvement of their matrix-fibrillar structure – the average diameter of PP and PET fibrils decreased, and the uniformity of their distribution increased [27,28]. A significant increase in the tensile strength of biodegradable composites was achieved due to interfacial adhesion and the formation of a percolation network in the matrix with the simultaneous use of organomodified montmorillonite and multilayer CNTs [29].

Today, the number of created varieties of polymer composites exceeds the number of existing steels. At the same time, such materials are characterized by a longer service life of products, as well as a better price-quality ratio. The tendency to create new composites is constantly growing, despite the limited amount of natural raw materials, since only ~ 10% of petroleum products are spent on the production of all chemicals from oil, including monomers.

In chemical technology, the main criterion for testing theoretical hypotheses remains the results of experiments, which are laborious and long-term. Thus, the development of new nanocomposites requires research aimed at establishing physicochemical factors that determine the compatibility and segregation of components, the formation of a microheterogeneous structure and the relationship with the operational characteristics of products based on them. An important task is to minimize the transition time from laboratory experiments to industrial samples. An effective means of increasing the efficiency of scientific research in solving problems of calculation, analysis, optimization and prediction of chemical and technological processes is the method of mathematical modeling of the experiment. The mathematical model is a response function that connects the optimization parameter characterizing the results of the experiment with the parameters that vary during the experiments. Response surfaces in multicomponent systems are complex and can be adequately described only by polynomials of high degrees, which requires a significant number of experiments, since it is known that in a polynomial of degree n, coefficients are added from the number of components q. To accelerate research and increase the reliability of the results, we have developed computer programs that allow us to build plans for conducting experiments in the studied area of the factor space for any types of mixture systems and all possible combinations of ingredients in three- and four-component heterogeneous compositions [30-33]. With the help of the created programs, plans are built in an automated mode using three types of models of dependence of the output parameters on the content of components - incomplete cubic, cubic and quadratic, which establish the relationship between the content of ingredients and the properties of the system. Calculation of the coordinates of the points of the experimental plan is also carried out using software. To optimize the composition of the composition, software has been developed using the generalizing function of the Harrington criterion and using the penalty function method with the subsequent application of gradient descent with step fragmentation. Thus, the use of mathematical experimental planning using software will allow to accelerate the conduct of experiments dozens of times, sharply reduce their number and quickly identify the optimal variant of the studied process.

Further scientific research into heterogeneous multicomponent systems using the developed software will contribute to the development of Ukraine's chemical complex and the production of modern polymer composite materials, the production and use of which in various industries will increase the competitiveness of domestic products in the domestic and foreign markets, significantly reduce the country's dependence on imported chemical products, and solve environmental and social problems through the introduction of "green" technologies.

CHAPTER 1. COMPOSITE NANOFILLED SYNTHETIC FIBRE MATERIALS

Since ancient times, fibers, fibrous materials and products made from them have played an important role in people's lives. Until the beginning of the 20th century, the raw materials for fibrous materials were natural fibers - wool, cotton, flax, hemp, silk. With the expansion of requirements for the performance characteristics of such materials, especially for technical products, there was a need to create alternative fibers and threads. Starting from the middle of the last century, the development of the science of synthetic polymers, their ability to transition to a viscous-fluid state and the ability to longitudinally deform a liquid jet flowing from the spinneret opening, determined the emergence and existence of the field of technology of fibrous materials, including chemical fibers. During this period, a group of so-called "classical" or "traditional" fibers was formed: polyamide, polyester, polyolefin, polyacrylonitrile, polyvinyl chloride and polyvinyl alcohol. Today, traditional fibers are subjected to targeted modification in order to improve or give them fundamentally new functional characteristics. For this purpose, various methods of modification are used, among which the most common are physical (consists in reducing the dimensional characteristics of individual filaments to micro- and nanosizes) and composite, in which fibers are formed from binary mixtures of polymers or with the introduction of various additives (dyes, flame retardants, compatibilizers, substances in the nanoscale, etc.). Due to this, fundamentally new types of fibers and fibrous materials have appeared - high-strength, heat- and chemical-resistant, non-combustible, electrically conductive, sorption, ion-exchange.

Composite materials include materials that consist of two or more substances that structurally complement each other and have properties that are absent in each individual component. The performance of composites depends on many factors: the chemical nature of the matrix and filler, volume fraction, degree of dispersion, orientation, uniformity of the distribution of additive particles, the size and properties of the transition layer, etc. Among them, the size of nanoparticles plays a dominant role. In accordance with the terminology adopted by IUPAC (International Union of Pure and Applied Chemistry), objects with a size not exceeding 100 nanometers are considered nanoparticles. They can be of various shapes - plates, tubes, spheroids, rods, while at least in one dimension their size must be in the range from 1 to 100 nm. Filled composite fibers and filaments in which at least one of the components has the specified dimensions are called nanofibers.

Nanomaterials have always existed in nature in the form of composites filled with carbon black or natural clay, and have been used for many centuries. At the same time, they are new and little studied for materials science. The specificity of the characteristics of substances at the nanometer scale and their associated new unique properties are due to the fact that the dimensions of the structural elements of nanoobjects lie in the range (10-9÷10-7) m, have a complex internal organization, the ability to pack tightly, strong lateral (side) interactions, as well as a high surface area to volume ratio. The properties of nanocomposites are also largely determined by the size of the transition layer at the filler/polymer phase interface. The interface, which is around the nanoparticle, has a finite thickness, within which the system parameters differ sharply from similar characteristics in the polymer volume. The experimentally determined thickness of the interfacial layer ranges from 0.004 to 0.16 mm and depends on the degree of affinity between the surface of the NPs and the functional groups of the macromolecules of the polymer matrix [34]. The chemical nature of the additives and their concentration significantly affect the interfacial phenomena and properties of composite fibrous materials.

1.1. Synthetic classical fibers and threads filled with nanoadditives of various chemical nature

The increasing requirements for the quality and functional characteristics of fibers and threads lead to the search for methods of their modification in order to provide technical products and household goods made from them with a set of unique consumer properties. Modern textile materials for everyday and especially for special clothing must have high mechanical and hygienic indicators, as well as reliably protect a person from external negative factors (high and low temperatures, increased content of gases and emulsions of toxic substances, biological factors, electromagnetic radiation).

Nano-filled industrial synthetic fibers obtained by processing melts or polymer solutions have been produced for more than 20 years. Nanoparticles of various chemical nature, size and configuration are used for their modification: carbon derivatives, natural minerals, metals, metal oxides, etc. In this case, NPs with the desired size, shape and functional properties are selected from previously obtained ones or they are synthesized directly in the molding solution or on the surface of the finished fibers. Depending on the method of introducing nanoadditives, their content, dispersion and continuity of the structures that they form in the volume of the material, fibers with fundamentally new characteristics are obtained, which can be divided into bulk and surface. Bulk properties include mechanical, light, heat and electrical conductivity, density, etc. Surface properties are the sorption characteristics of fibers in relation to various substances (liquids, gases, ions, dyes), their catalytic activity, reflectivity, dyeing ability and other indicators that depend on the electronic structure of atoms located on the surface of the particles they form.

The unusual structure of natural aluminosilicates and their inherent properties provide broad opportunities for the creation of a range of multifunctional polymer materials. The basis of clays are silicon and alumina ions, which form, respectively, tetrahedral and octahedral two-dimensional networks interconnected into layers (plates) with dimensions of ~ (1000x1000x1) nm, which self-organize into packages with an interlayer space of up to 50 nm. The outer and inner surfaces of the plates are hydrophilic and polar, which promotes wetting and penetration into the space between the layers of both low- and high-molecular compounds that have polar groups in their structure. Due to this, layered silicates are the most effective modifiers for hydrophilic polymers [35]. The introduction of natural alumina particles into the structure of synthetic fibers from polar polymers provides high electrical and thermal conductivity, mechanical strength, chemical activity, protection against UV radiation and fire [3,4,36]. Thus, in polyamide fibers containing 5.0 wt. % of alumina nanoparticles, the tensile strength and bending strength increase by 40 and 60 %, respectively.

A more complex task is the modification of fibers based on non-polar or weakly polar polymers. In this case, the incompatibility of hydrophilic clays and hydrophobic polymers is the main problem, to solve which clays are pre-modified in various ways: by ion exchange of clay cations for organic cations; by adsorption on the surface of water-soluble polymers, alkyl ketones, methyl acrylate, surfactants; by grafting organosilanes to the clay surface with the formation of Si-O-Si bonds; by introducing organic molecules capable of Van der Waals or ion-dipole interaction with the clay surface, etc. [2,35]. The nature of the packing of modifier molecules in the interlayer space determines the distance between silicate plates, the organophilicity of clays and, as a result, the structure of nanocomposites when mixed with polymers. Filling polypropylene fibers with organomodified alumina allows to eliminate their significant disadvantage as textile threads, namely the ability to dye. Fibers containing 15.0 wt. % alumina are dyed with dyes of various classes to achieve deep tones, which significantly expands the areas of their application in the production of household materials.

The discovery of carbon nanotubes (CNTs) in 1991 led to significant progress in the field of nanotechnology and marked a new era in the material world, including in the field of chemical fibers. Single- and multi-walled CNTs are characterized by a complex of unique mechanical, electrical, thermal and chemical properties, as well as a high ability to transport electrons. The elastic modulus of carbon nanotubes approaches the values of this indicator for diamond (1.0 and 1.2 TPa, respectively), their strength is 100 times higher than the best samples of steel. They are also characterized by high electrical conductivity, thermal stability (up to 2800 ^{0} C in vacuum), thermal conductivity (approximately twice as high as that of diamond). The elasticity of multi-walled CNTs can reach 5000 GPa, they bend like a straw, but do not break and can straighten without damage [5]. Due to their ultra-high mechanical properties, single- and multi-walled carbon nanotubes are a particularly attractive reinforcing filler. The strength and Young's modulus of polypropylene fibers are increased by almost 3 times when 5.0 vol. % of CNTs are introduced into their structure, provided that they are additionally oriented [37]. Polyvinyl alcohol fibers filled with nanotubes are 120 times stronger than steel wire and 17 times lighter than Kevlar fiber. The introduction of (0.5÷3.0) wt. % of CNTs into the melt of polypropylene of different grades provides an increase in the tensile strength of monofilaments (P) and their dimensional stability (estimated by the value of the initial modulus E) in the entire range of additive concentrations [38]. The dependences of P and E on the CNT content are extreme: maximum values are achieved when 0.5 wt. % of the additive is introduced. The best mechanical characteristics are possessed by monofilaments formed from PP with lower viscosity, which may be associated with a thinner and more uniform dispersion of CNTs in the melt. Adding nanotubes to synthetic fibers in an amount of 5 to 20 wt. % provides them with electrical conductivity at the level of copper wire and chemical resistance to the action of many reagents [39]. To date, a significant problem in creating high-performance polymer/CNT composites is the difficulty of uniform dispersion and orientation of nanotubes in the matrix. The new method of “layer-by-layer planting” proposed by the authors [40] allowed the formation of polyvinyl alcohol fibers with adjustable CNT orientation, as a result of which the reinforcement effect increased sharply – the strength of the fibers increased from 50 to 1255 MPa, and their electrical conductivity also increased. The formation of a percolation network by carbon nanotubes in the polymer matrix provides an increase in thermal, optical, and electrical parameters even at an additive concentration of 0.0025 wt. % [6].

Effective modifiers are also nanoparticles of metals (Ag, Cu, Ti, Mn, Zn, Au, Pt, Pa) and metal oxides. Fibrous materials with additives of copper, nickel and silver NPs exhibit sorption and biocatalytic properties, with platinum inclusions they are catalytically active, and nickel-, iron- and cobalt-containing ones acquire magnetic characteristics [10,41-44]. The introduction of zinc oxide NPs in the form of nanorods into polypropylene fibers improves their mechanical performance [11]. Polyester textile threads filled with TiO_2, Al_2O_3, ZnO and MgO nanoparticles exhibit photocatalytic activity, protection from UV radiation, antistatic properties, and abrasion resistance [12,13]. Modification of synthetic fibers with TiO_2 and ZnO nanoparticles gives products made from them the ability to self-clean like plant leaves, insect wings, etc. The introduction of aluminum oxide nanoparticles

[25] and silicas with different specific surface areas [45] into their structure contributes to the improvement of the mechanical properties of polypropylene monofilaments. The use of combined nanoadditives enhances the modifying effect and expands the spectrum of operational characteristics of fibers and products based on them. The introduction of silver/silica [14], silver/alumina [46], and mixed oxide TiO_2/SiO_2 [47] into the structure of PP monofilaments gives them biological activity and improves mechanical properties. Nanosized silicon dioxide in the structure of synthetic fibers prevents pollution and promotes self-cleaning of products made from them, and the bifunctional additive TiO_2/SiO_2 makes it possible to create a new generation of effective nanofilled materials for cleaning technological environments, including the medical industry [48]. In terms of their sorption performance, they exceed ion exchange resins, while being 5 times cheaper than them. Such nanocomposites absorb a wide range of metal ions from water, destroy organic compounds, concentrate and separate radionuclides.

1.2. Nanofilled composite yarns and fine-fiber materials from melts of polymer blends

Polymer blending is a simple and affordable way to obtain new composite fibrous materials with predicted properties and is more effective than the synthesis of new monomers and polymers. Blends can be fully compatible or incompatible and partially compatible. In this case, various types of polymer dispersions occur - from simple binary to the formation of block copolymers, interpenetrating networks, microfibrillar or droplet structures, molecular composites, etc.

[3,49]. The formed types of phase morphologies determine the properties of such systems. Of particular interest are mixtures in which a component of the dispersed phase forms micro- or nanofibrils in the matrix of another. In the threads obtained from them, a self-reinforcing effect occurs, the degree of which can be regulated by changing the ratio of the sizes of the reinforcing fibers. By increasing the length or decreasing the diameter of the microfibrils, the mechanical properties of microfibrillar composites (*MFC*) can be significantly improved [50]. The process of obtaining *MFC* includes three main stages: extrusion mixing of melts of two polymers with different melting points (T_{mp}) and formation of an extrudate or monofilament; their cold drawing for longitudinal orientation and fibrillation of both phases; subsequent heat treatment at a temperature in the range between the T_{mp} of the mixture components, which ensures the formation of an isotropic matrix.

A schematic representation of the stages of the *MFC* production process is shown in the figure 1.1 [51]. Threads with a microfibrillar structure have a number of advantages over traditional ones: increased strength and resistance to deformation, relative ease of production and further processing, reduced weight, etc.

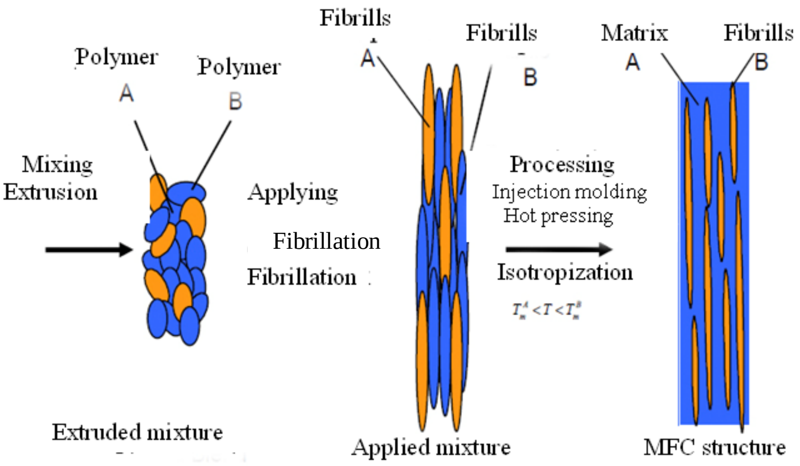


Fig. 1.1 – Scheme of formation of microfibrillar composites

Today, the *in situ* formation of micro- or nanofibrils of one component in the matrix of another has been implemented for many pairs of polymers by extrusion [3,26-28,52-60], blowing [61,62], uniaxial stretching [63] and 3D molding [1,64,65]. Regulation of microfibrillar morphology (reduction of fibril diameters, increase in their length and mass fraction) is achieved by introducing special substances into the mixture of incompatible polymers - compatibilizers [66], nanoadditives [25-27,47,57,60,65] or their compositions [28,67,68]. It is known that the properties of composite monofilaments largely depend on the type of structure formed by the polymer of the dispersed phase in the matrix. In this case, the formation of morphology is determined by the course of a number of microrheological processes that occur with droplets of the dispersed phase during the flow of the melt, namely: their dispersion, coalescence, deformation and migration. The degree of manifestation of each of them depends on the ratio of the main ingredients of the mixture, the content of the additive and its influence on the rheological properties of the components and the course of interfacial processes. The modifying effect of fillers on the structure of three- and four-component systems is manifested in a change in the ratio of viscoelastic properties of the ingredients, a decrease in the value of interfacial tension and an increase in the stability of liquid cylinders (fibrils). Nanoadditives in the melt of thermodynamically incompatible mixtures play a dual role. First, due to the compatibilizing effect in modified systems, the degree of dispersion of the fiber-forming polymer and the kinetic stability of the melts increase, and the processes of droplet aggregation are inhibited, which contributes to obtaining a finer morphology. Secondly, NPs give fibrous materials unique properties inherent in substances in the nanoscale.

1.2.1. Nanofilled composite threads with microfibrillar structure. One of the most studied systems for obtaining threads with increased initial modulus and tensile strength are blends of polyethylene terephthalate (PET) with polyolefins (PO) or with polyamides (PA), since polyester fibers are characterized by high resistance to deformation, which makes them an ideal reinforcing element. The formation of PET microfibrils in a PO and PA matrix allowed to significantly increase the dimensional stability of the threads and obtain a high-strength tire cord. Studies of the morphology and mechanical properties of composite threads based on PET/PP blends filled with titanium oxide nanoparticles showed that the morphometric characteristics of the fibrillated PET phase, namely their diameter and length and distribution uniformity, depend on the concentration and size of the filler nanoparticles [26,28]. At a titanium oxide NP content of 4.0 wt. % the average diameter of microfibrils decreases from 5.4 μm to 1.1 μm, and the range of diameters narrows from (2.0÷9.2) to (0.6÷4.5) μm compared to the original mixture. At the same time, their length also increases. The change in the structure of the threads caused an increase in the modulus and tensile strength by 1.4 and 1.3 times, respectively.

The possibility of controlling the process of PP microfibril formation in the SPA matrix by introducing into the melt of the PP/SPA mixture of 30/70 wt. % nanoparticles of oxides of various metals was shown by us in the works [25,69,70]. As can be seen from the microphotographs of cross-sections of extrudates of PP/SPA/aluminum oxide mixtures shown in Fig. 1.2, in the initial mixture PP is roughly dispersed in the SPA matrix. The introduction of (0.1÷3.0) wt. % Al_2O_3 nanoparticles into the system contributes to improving the compatibility of components at the phase interface and causes an increase in the degree of dispersion and uniformity of distribution of polymer particles of the dispersed phase in the dispersion medium. The modifying effect is achieved due to the compatibilizing (emulsifying) action of aluminum oxide nanoparticles, as evidenced by the decrease in the interfacial tension in nanofilled mixtures [70].

a)

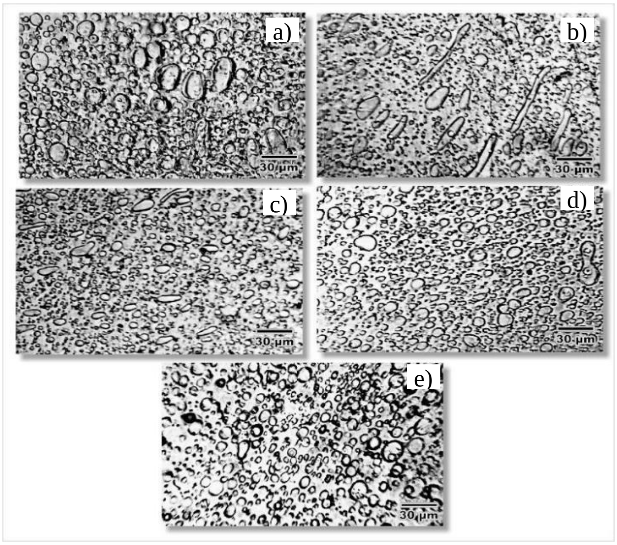


Fig. 1.2 – Microphotographs of cross-sections of extrudates of mixtures with different aluminum oxide contents, wt. %: a) 0; b) 0,1; c) 0,5; d) 1,0; e) 3,0

Microscopic studies of longitudinal sections of extrudates (Fig. 1.3) and residues of the dispersed phase after extraction of the matrix polymer (Fig. 1.4) indicate that aluminum oxide nanoparticles do not prevent droplets of the dispersed phase from deforming and merging with the formation of liquid jets (microfibrils) of PP in the SPA matrix.

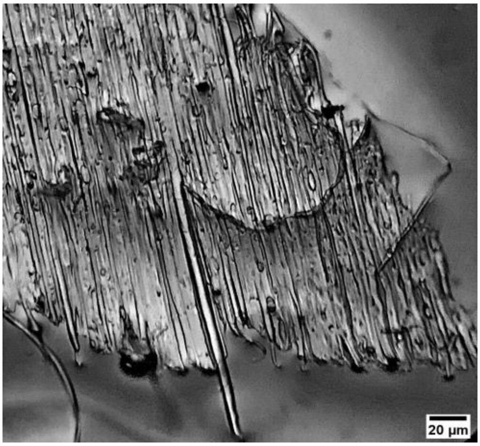
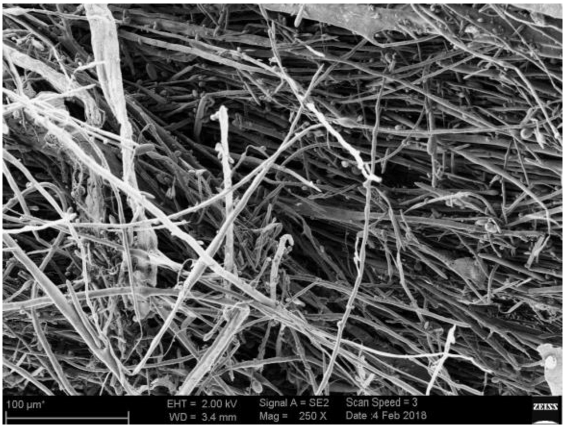


Fig. 1.3 – Micrograph of a longitudinal section of a PP/SPA/ Al_2O_3 extrudate with a nanoadditive content of 1.0 wt. %

During the treatment of extrudates with a solvent selective for PP, the copolyamide goes into solution, and the dispersed phase remains mainly in the form of a bundle of microfibrils (Fig. 1.4). Microscopic studies of the influence of the concentration of aluminum oxide NPs on the dimensional characteristics of different types of polypropylene structures indicate that, along with microfibrils, a small number of films and micron-sized particles are also formed. Microfibrils are the predominant type of structure in extrudates of the original and nanofilled mixtures. When aluminum oxide NPs are added, the diameter of the microfibrils decreases, and their mass fraction increases in the entire concentration range. At the same time, at a nanoadditive content of 1.0 wt. %, the average diameter of the microfibrils decreases to 2.2 μm (versus 4.0 μm for the original mixture), and their fraction increases to almost 95%. This is due to the increased resistance of nanofilled microfibrils of smaller diameters to decay, as evidenced by a decrease in the value of the instability coefficient and an increase in their lifetime [70].



Fug. 1.4 – Electron micrograph of dispersed phase (PP) structures after matrix polymer (CPA) extraction

The studies performed showed that the introduction of aluminum oxide NPs into the melt of the PP/SPA mixture not only does not complicate their processing, but even increases the stability of the formation and thermal orientation drawing of modified monofilaments. It is known that during the spinneret and thermal orientation drawing process, further deformation of the dispersed phase structures occurs, while the microfibrillar morphology in the monofilaments is preserved (Fig. 1.5) [59].

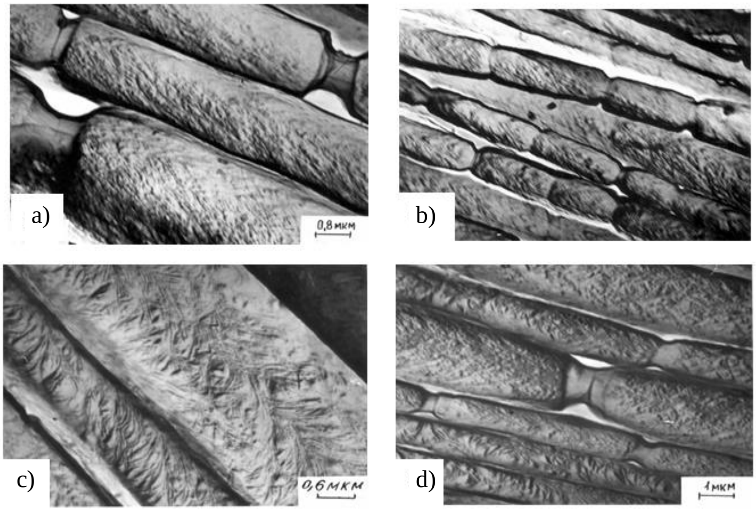


Fig. 1.5. Electron micrographs of polyoxymethylene microfibrils (at various magnifications) formed in situ in ethylene vinyl acetate copolymer

Important indicators of the threads, from the point of view of further processing and the quality of products based on them, are mechanical characteristics. The tensile strength and modulus of elasticity of composite monofilaments from nanofilled systems are improved, compared with the threads from the original mixture (Table 1.1). This is natural, since in the PP/SPA/Al_2O_3 mixture a matrix-fibrillar morphology is formed, that is, the effect of self-reinforcing the threads takes place. In this case, the modifying effect depends on the concentration of the nanoadditive: an increase in the content of aluminum oxide NPs from 0.1 to 1.0 wt. % is accompanied by an increase in the strength and dimensional stability of the threads, and at a concentration of 3.0 wt. % the values of P and E decrease. The degree of increase in the mechanical indicators of the monofilaments correlates with the morphology of the extrudates and the dimensional characteristics of PP microfibrils. Their minimum diameter and maximum proportion in the structure determine the highest values of strength and resistance to deformation of monofilaments formed from a mixture containing 1.0 wt. % alumina.

Table 1.1 – Effect of aluminum oxide nanoparticle content on mechanical properties of composite monothreads

Name of polymer, mixture	Content of Al_2O_3, wt. %	Extraction multiplicity	Strength, MPa	Elastic modulus MPa	Elongation, %
CPA	0	6,0	270	3240	13,7
PP	0	7,2	390	4970	8,9
PP/C{A	0	4,0	310	3870	14,6
PP/CPA	0,1	4,3	360	3910	14,0
PP/CPA	0,5	4,5	390	4100	13,8
PP/CPA	1,0	5,0	430	4520	12,1
PP/CPA	3,0	5,0	390	4150	11,9

The obtained result is consistent with our previous conclusion that the values of *P* and *E* reach maximum values when the entire polymer of the dispersed phase forms microfibrils [59].

Recent studies have shown that the most effective is the combined use of substances in the nanostate and traditional compatibilizers [27-29,71,72]. Thus, it was shown that for an incompatible PP/PA mixture, there was a synergistic effect on the morphology of the nanodispersed additive (hydrophobic silica NPs) and the compatibilizer (polypropylene with grafted maleic anhydride PPgMA): the addition of PPgMA provided a

12-fold reduction in the size of polyamide droplets, and when used simultaneously with a nanoadditive – 25-fold [71]. The introduction of polystyrene additives with grafted maleic anhydride (PSgMA) into the PS/PA/CNT mixture allowed to increase the uniformity in size and geometric shape of polystyrene droplets and the mechanical properties of composites [72]. Modification of the CNT surface with a surfactant ionic liquid promoted the formation of a percolation network structure by nanotubes in the polymer matrix, as a result of which the electrical characteristics of composites based on PS/butylene adipate and terephthalate copolymer mixtures were dramatically improved. In this case, double percolation occurred, and the formation of a network structure by nanotubes also caused a significant increase in the elastic modulus. Biodegradable biological materials with improved mechanical properties were obtained by simultaneously using multilayer CNTs and organomodified montmorillonite - a synergistic effect was achieved with a content of 0.5 wt. % of nanoadditives [29]. Simultaneous introduction of two compatibilizers into the PP/SPA mixture made it possible to implement a microfibrillar structure in compositions with a ratio of components corresponding to the phase change region (40/60 and 50/50 wt. %) [66]. The authors [28] showed that the simultaneous use of a nanodispersed additive (TiO_2) and a compatibilizer (PPgMA) in PP/PET blends is the most effective and provides maximum improvement in the mechanical properties of composite yarns by increasing the length and minimizing the diameter of PET fibrils in the polypropylene matrix.

Systematic studies on the possibility of controlling the process of microfibrillar structure formation in melts of thermodynamically incompatible polymer mixtures by introducing compatibilizer/nanoadditive compositions and establishing the structure–property relationship of fibrous materials have been conducted at KNUTD for many years [27,32,47,59,67,68]. Thus, carbon nanotubes and PPgMA compatibilizer were used to modify a PP/SPA mixture of 20/80 wt. %. The mechanical properties of monofilaments formed from the original polymers and modified mixtures are presented in Table 1.2 [27]. As can be seen from Table 1.2, the introduction of 20 wt. % of stronger PP into the copolyamide leads to an improvement in the mechanical performance of monothreads. The tensile strength and initial modulus of monofilaments, in the structure of which there is a nanofiller or compatibilizer, also increase. In all the studied mixtures, polypropylene forms *in situ* microfibrils in the SPA matrix and provides self-reinforcement of the threads.

Table 1.2 – The effect of modifier additives on the mechanical properties of monofilaments

Sample name	Linear density, text	Strength MPa	Elastic modulus MPa	Elonga-tion, %
PP	5,6	370	2600	9,4
CPA	7,3	210	3240	20,9
PP/CPA	8,1	260	3870	15,6
PP/СПА/PPgMA	9,1	320	3750	17,3
PP/СПА/CNT	11,0	340	4680	20,1
PP/CPA/CNT/ PPgMA	10,3	390	5110	19,8

The addition of compatibilizer and carbon nanotubes contributes to the reduction of microfibril diameters and the proportion of unwanted structures (particles, films), resulting in an increase in the strength and initial modulus of monofilaments. The maximum improvement in the mechanical properties of composite monofilaments occurs with the simultaneous addition of CNTs and PPgMA, which corresponds to the most perfect microfibrillar structure: the average diameter is 1.5 μm (versus 2.6 μm for the original mixture) and the proportion of films, the presence of which is known to worsen the mechanical properties of the filaments, is sharply reduced [59].

The possibility of controlling the process of self-reinforcing composite yarns from a PP/PVA blend by simultaneously introducing a nanofiller and a compatibilizer is shown in [67]. Quantitative microscopic studies of the effect of a nanodispersed silver/silica additive, a sodium oleate compatibilizer ($C_{18}H_{33}O_2Na$) or their combination on the microstructure of PP/PVA extrudates indicate that individual substances and their binary composition have an emulsifying effect on the melt and allow regulating its morphology. In modified compositions, the average diameter of microfibrils (đ) decreases and their mass fraction increases, and the number of other types of structures decreases (Table 1.3). In this case, the simultaneous use of a nanofiller and a compatibilizer is more effective.

Table 1.3 – The influence of additives silver/silica, sodium oleate or their compositions on the characteristics of structure formation processes in melts of PP/PVA blends

Name of the mixture, content of components, wt. %	Microfibrils		Content of structures of other types, wt. %	
	đ, μm	content, wt. %	Parti-cles	films
PP/PVA, 30/70	3,5	86,5	3,9	9,6
PP/PVA /Ag/SiO_2, 30/70/1	1,6	90,6	3,3	6,4
PP/PVA /$C_{18}H_{33}O_2Na$, 30/70/3	1,4	92,7	3,6	3,7
PP/PVA /Ag/SiO_2 /$C_{18}H_{33}O_2Na$ 30/70/1/3	1,1	97,9	1,2	0,9

As can be seen from Table 1.3, the diameter of microfibrils in the four-component mixture decreases by 3.2 times, while when adding 1.0 wt. % Ag/SiO_2 nanoparticles or 3.0 wt. % sodium oleate, *đ* decreases by 2.2 and 2.5 times, respectively. In the presence of two modifiers, migration processes are significantly slowed down, which leads to a sharp drop in the number of films.

Studies of the mechanical properties of composite monofilaments show that they also indirectly correlate with the microstructure formed by the polymer of the dispersed phase in the matrix (Table 1.3, 1.4). The highest indicators of strength and resistance to deformation are those of threads formed from a composition modified with a nanofiller and a compatibilizer simultaneously. In this case, polypropylene is present in the PVA matrix mainly in the form of thinner microfibrils, and the proportion of films is reduced by almost 10 times.

Table 1.4 – The effect of silver/silica, sodium oleate or their compositions on the mechanical characteristics of composite monofilaments from a PP/PVA blend

Name and composition of the mixture, wt %	Tex	Tensile strength, MPa	Initial modulus, MPa	Elongation, %
PP/PVA, 30/70	8,1	300	4200	9,3
PP/PVA /Ag/SiO_2, 30/70/1	7,4	390	4800	13,0
PP/PVA /$C_{18}H_{33}O_2Na$, 30/70/3	7,2	470	5300	8,5
PP/PVA /Ag/SiO_2 /$C_{18}H_{33}O_2Na$, 30/70/1/3	7,0	550	6400	8,0

Thus, the maximum self-reinforcing effect of composite filaments formed from compatibilized nanofilled incompatible polymer blends is the result of improving their matrix-fibrillar structure.

1.2.2. Nanofilled fine-fiber materials derived from microfibrillar composites. Today, there are a number of methods for producing fine-fiber materials with micro- and nano-sized diameters: aerodynamic spraying of the melt with a jet of compressed air [73-75], electroforming from a polymer melt or solution under the action of electrostatic forces [76-80], and processing of melts of mixtures of thermodynamically incompatible polymers into composites with a micro- and nanofibrillar structure [57-65,81-86].

Aerodynamic forming produces nonwoven materials (NM) with fiber diameters of (1.0-20.0) μm. With the maximum increase in air velocity during polymer melt blowing, NM were formed from nanofibers with an average diameter of ~ 500 nm [75]. Electroforming produces nanofiber sheets with individual filament diameters of 10 nm or more, but the use of this method is limited by low productivity and high toxicity of solvents. In order to give nonwoven materials new properties (for example, increasing filtration efficiency and reducing hydraulic resistance of filter materials), they are obtained on the basis of micro- and nano-sized fibers by combining blowing and electrospinning methods [76,79], but the production of such materials is complicated by the incompatibility of the forming speeds in both methods. In recent years, needle-free electrospinning technology has been developed, which can eliminate the shortcomings of traditional electroforming devices, such as low productivity, non-uniformity of sheets in thickness, limited size, and difficulty in cleaning a single needle [80]. The developed needle-free electrospinning apparatus can be used for the industrial production of nanofiber membranes of considerable width.

By processing melts of thermodynamically incompatible polymer mixtures for which microfibrillar morphology is realized, fine-fiber materials are obtained in the form of complex threads, staple fibers or nonwovens, in which individual filaments have micro- or nanosizes [57-65,81-87]. The structure of the composite monofilament or film, which comes out of the molding hole, is a continuous phase of the dispersion medium filled with thin jets (fibrils or microfibers). After extraction of the matrix from the composites with a solvent inert to the polymer of the dispersed phase, bundles of micro- and nanofibers or nonwoven webs from them remain. Fig. 1.6 shows a micrograph of polypropylene microfibrils with an average diameter of (1.5÷2.5) μm after dissolution of copolyamide from the composite strand [64].

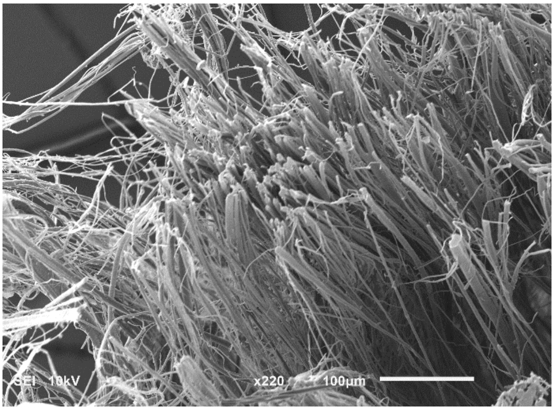


Fig. 1.6. Electronic microphotograph of PP microfibrils after extraction of SPA from the strand

By forming a jet on a capillary viscometer with subsequent thermal drawing from the melt of a polybutylene terephthalate/polypropylene (PBTE/PP) mixture, PBTE nanofibers with a diameter of 600 nm and a length of 100 μm were obtained [88]. By processing polyethylene terephthalate (PET)/PP [26] and polytetrafluoroethylene/polylactide (PTFE/PL) [86] compositions by extrusion, PET microfibrils with diameters of (2.0÷9.2) μm and PTFE nanofibers with diameters of (100÷500) nm were formed. Nonwoven material from polypropylene microfibers was obtained after extraction of the matrix polymer from composite films formed from the melt of a PP/SPA mixture on a worm press through a flat-slot head of the "fishtail" type [57]. The microfibers had diameters ranging from tenths of a micrometer to several micrometers, were of practically continuous length, and were oriented in the direction of extrusion. By processing mixtures consisting of two polymers of the dispersed phase and the matrix, a nonwoven material with a bimodal distribution of fibers by diameter was obtained [61]. Nonwoven fabrics were formed from mixtures in which the matrix polymers were polystyrene (PS) or polyethylene oxide (PEO), and the dispersed phase was polyethylene (PE) and polyamide 6, by the blowing method. After dissolving PS with tetrahydrofuran and PEO with water, a fine-fiber material with an average diameter of PA6 microfibers ~ 9.0 μm and PE nanofibers ~ 600 nm was obtained.

1.2.2.1. Nanofilled complex microfibrillar threads. Modification of the properties of synthetic fibers and threads by reducing the diameters of filaments to micro- and nano-sizes and introducing nano-additives into their structure is one of the most promising areas in the field of chemical fiber technology, as it allows significantly improving the quality of products and reducing the material intensity of production. Materials from ultrafine fibers retain all the positive properties inherent in products from traditional synthetic fibers: strength, high dimensional and wear resistance. At the same time, due to the very small diameter of individual filaments in textile products from them, many air voids can form. Thanks to them, free air exchange occurs between human skin and the external environment, i.e. such materials have better hygienic properties. Formation of complex microfibrillar fibers and threads by processing melts of polymer mixtures allows you to regulate their consumer characteristics both due to the properties inherent in nanofillers and due to their effect on the size of the filaments of the dispersed phase component in the matrix.

The dependence of the mechanical properties of complex yarns from polypropylene microfibrils on the content of nanoadditives and their chemical nature is given in Table 1.5, 1.6 and Fig. 1.7. Microfibrillar yarns were obtained by extraction of the matrix polymer from monofilaments formed from PP/SPA mixtures filled with aluminum oxide nanoparticles [25], as well as bicomponent nanoadditives TiO_2/SiO_2 [47], Ag/ SiO_2 and Ag/Al_2O_3 [87]. As can be seen from Table 1.5, the tensile strength, elastic modulus and elongation of complex yarns from the initial mixture are close to similar values for textile polypropylene yarns formed using traditional technology. The introduction of aluminum oxide nanoparticles into their structure leads to an increase in mechanical properties, the degree of increase of which, as for composite monofilaments, is determined by the content of the nanoadditive and correlates with the dimensional characteristics of PP microfibrils.

Table 1.5 – Effect of nanoscale alumina content on mechanical properties of complex threads

Contetn of Al_2O_3, wt. %	Strength, MPa	Elastic modulus, GPa	Elonga-tion, %	Maintaining strength, %	
				in a knot	in a loop
0	260	3,50	12.3	63	68
0.1	310	3,75	11.4	67	71
0.5	335	4,08	11.2	72	75
1.0	380	4,29	11.0	75	78
3.0	360	4,15	10.3	70	73

The minimum average diameter of microfibrils (2.2 μm) and their maximum proportion (94.9 wt. %) in the extrudate of the mixture containing 1.0 wt. % Al_2O_3 provided mono- and complex filaments with the highest strength and modulus of elasticity. In addition, microfibrillar filaments are characterized by improved elasticity, compared with textile polypropylene filaments, as evidenced by the values of strength retention in the loop and knot.

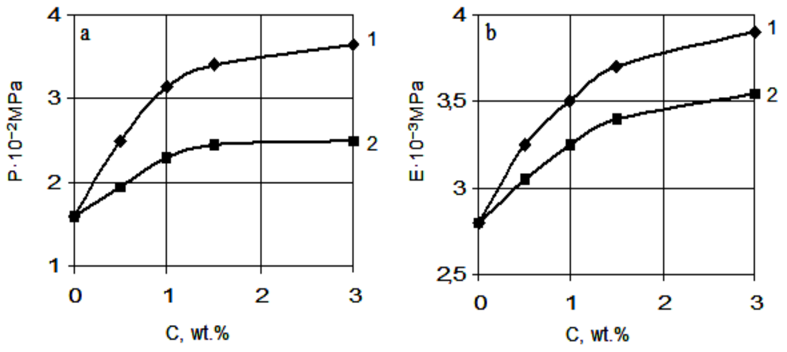
Table 1.6 presents the results of a study of the influence of the content of the mixed oxide TiO_2/SiO_2 on the properties of complex fibers from nanofilled PP microfibrils [47]. The data in the table indicate that the nature of the dependence of the mechanical properties of microfibrillar fibers on the content of the nanofiller is similar to that described for fibers modified with alumina.

Table 1.6 – Effect of TiO_2/SiO_2 nanoparticles concentration on the properties of complex threads

Content of TiO_2/SiO_2, wt. %	Strength, MPa	Initial modulus, GPa	Elon-gation, %	Specific surface area, m^2/g
0	160	2,8	13,3	84
0,5	190	3,5	11,8	135
1,0	240	3,8	12,6	190
3,0	220	3,4	11,7	210

Increasing the concentration of mixed oxide NPs to 1.0 wt. % leads to an increase in strength and initial modulus by 1.5 and 1.3 times, respectively, and its further increase is accompanied by some deterioration of the mechanical characteristics of the threads, which is associated with an increase in the diameters of microfibrils, as well as the number of films. The introduction of (0.5÷3.0) wt. % TiO_2/SiO_2 nanoparticles into the microfibril structure leads to an increase in their specific surface area in the entire concentration range.

Bifunctional inorganic substances in the nanostructure – silver/silica and silver/alumina also contribute to improving the quality of microfibrillar filaments: their tensile strength (*P*) and elastic modulus (*E*) increase (Fig. 1.7). At a concentration of nanofillers in the mixture of more than 1.5 wt. %, the growth rate slows down. The established dependence is natural and may be associated with the effect of filling with a high-modulus nanodispersed additive, as well as with a change in the morphology of complex filaments [87].



a) б)

Fig 1.7 – Effect of nanoadditive concentration on the strength (a) and modulus of elasticity (b) of microfibrillar PP threads: 1– Ag/SiO_2; 2 – Ag/Al_2O_3

The effect of modification with Ag/SiO_2 additive is more pronounced compared to Ag/Al_2O_3 nanoparticles, which is due to their higher specific surface area. It is known that silicas provide a significant improvement in the mechanical properties of filled compositions. In this case, the reinforcing effect of silica NPs correlates with the value of its specific surface area:

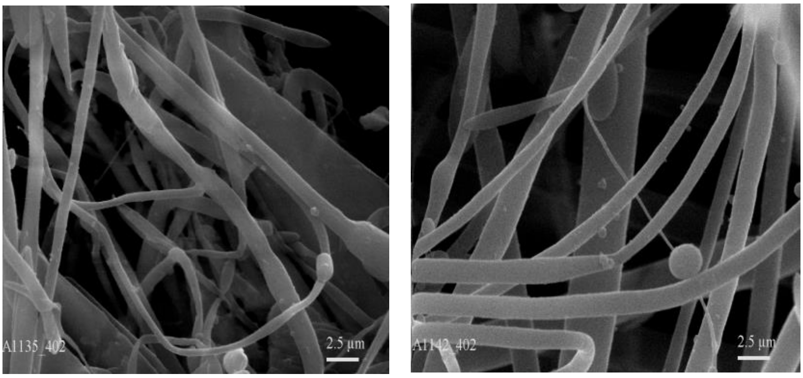
the modulus increases when $S_{sa} \geq 50$ m^2/g, and the degree of reinforcement increases with the increase of this indicator [89].

The possibility of improving the quality of complex yarns by simultaneously using two modifying additives of nanofiller and compatibilizer is shown in [68] on the example of complex yarns obtained by processing mixtures of PP/CPA/CNT and PP/CPA/CNT/compatibilizer. Ethylene copolymer with vinyl acetate or sodium oleate was used as a compatibilizer. Studies of the mechanical properties of yarns from PP microfibrils showed that adding 0.1 wt. % of carbon nanotubes to the mixture increases their strength and initial modulus (Table 1.7). From the electron micrographs shown in Fig. 1.8, it is clear that under the influence of the nanoadditive, the diameters of PP microfibrils decrease, and they acquire the correct cylindrical shape. The number of so-called “varicose” fibers, which are formed as a result of incomplete decomposition of liquid jets, also decreases. All this contributes to the improvement of the mechanical properties of the threads.

Table 1.7 – Effect of CNT and CNT/compatibilizer additives on the mechanical properties of complex threads

Additive		Tex	Strength, MPa	Initial modulus, GPa	Elonga-tion, %
name	content, wt. %				
без добавок		4,2	170	2,8	13,3
CNT	0,1	4,0	230	3,5	10,7
CNT/ CEVA	0,1/3,0	3,3	255	3,8	13.6
CTN/$C_{18}H_{33}O_2Na$	0,1/3,0	3,1	275	4,5	12,7

The introduction of a compatibilizer into the nanofilled mixture contributes to further improvement of the mechanical properties of the threads. At the same time, the modifying effect depends on the chemical nature of the compatibilizer. The binary additive CNT/sodium oleate is more effective than CNT/CEVA - the strength and initial modulus of the threads are higher by 12 and 25%, respectively. This is due to their different effects on the morphology of incompatible PP/SPA mixtures. In compositions with sodium oleate additives, PP microfibrils quantitatively predominate over other types of structures and have a smaller diameter, which is one of the reasons for the increase in the performance characteristics of complex threads.



a) b)

Fug. 1.7 – Electronic microphotographs of polypropylene microfibrils from PP/CPA/CNT blends with the following composition: 30/70/0 (a); 29.9/70/0.1 (b)

. The simultaneous use of a nanoadditive and a compatibilizer significantly improves the hygienic properties of modified microfibrillar threads - their hygroscopicity, determined by the value of equilibrium water absorption, increases by (15÷20) times [68]. This is due to changes in the pore structure and an increase in the specific surface area of microfibrils (Table 1.8). The values of the specific surface area (S_{sa}) of unmodified microfibrils, calculated from the drying thermograms of the water sorption-desorption process, exceed the S_{sa} of traditional polypropylene textile threads by several orders of magnitude.

Table 1.8 – Effect of CNT and CNT/compatibilizer additives on the specific surface area and porosity of PP microfibrils

Добавка		Об'єм пор, м^3/г·10^{-3}				Питома повер-хня, м^2/г
назва	вміст, мас. %	макро-пори	мікро-пори	ультрапори		
				полі-шар	моно-шар	
без добавок		1,4	1,04	0,42	0,58	84,0
ВНТ	0,1	4,5	6,0	0.53	0.52	180,0
ВНТ/СЕВА	0,1/3,0	3,9	7,3	0,43	0,49	190,0
ВНТ/ол.Na	0,1/3,0	2,7	8,2	0,42	0,59	220,0

The introduction of CNT additives causes an increase in the specific surface area by 2.1 times (Table 1.8). For microfibers containing binary CNT/compatibilizer additives in their structure, there is a further increase in S_{sa}. At the same time, the volume of micro- and macropores increases, and ultrapores almost do not change.

1.2.2.2. Nonwoven materials made of ultrafine nanofilled fibers. Fibrous materials with filament diameters of micro- and nanoscale dimensions demonstrate unique chemical, physical and mechanical properties, they are characterized by a very high surface to volume ratio, which ensures their wide application as highly efficient sorbents, precision purification filters, membranes for separating liquid and gaseous media, special and medical products, etc. A biodegradable fibrous material with individual filament diameters from 800 nm to 9 μm was obtained by electroforming, which is characterized by ultra-high hydrophobicity and the ability to sorb oil and oil products (their sorption reaches more than 100 grams per 1 gram of fiber) [78]. By the method of needle-free electrospinning from industrial polymers of polyvinyl alcohol, polyacrylonitrile (PAN) and mixtures of PAN with polyethylene oxide or polyethyleneimide, nanostructured membranes were obtained, which are one of the most promising materials for solving the global climate problem - reducing carbon dioxide emissions into the atmosphere by reducing its concentration in the production of energy enterprises and subsequent utilization [90]. The introduction of titanium dioxide nanoparticles into the structure of the filaments gives the membranes photocatalytic properties and expands the possibilities of their application for a wide range of environmental problems. By the method of extrusion from a mixture of PL/PVA of composition 40/60 wt. %, a biodegradable fibrous material for medical purposes from polylactide fibrils with sizes from 400 nm to 1 μm was obtained [91].

Microfiltration using polymeric fine-fiber materials is one of the simplest, most reliable and economically feasible methods of purifying drinking water, atmospheric air and technological gas and liquid media from mechanical contaminants of micron and submicron sizes, bacteria, microbes, etc. [92]. The main indicators characterizing the operational properties of filter materials (FM), namely the retention capacity (efficiency) and permeability (specific productivity) are determined by the size and shape of the elements from which they are made. The average pore diameter of the filter layers is the lower, the smaller the size of the structural elements, and their shape is the more uniform, the more geometrically uniform and correct the shape of the structures forming the filter layer. In nonwoven materials obtained by blowing or electroforming, the fibers have a uniform distribution in diameters, however, due to their chaotic (according to the law of chance) arrangement in the layer, there is a probability of the formation of a certain number of pores with diameters larger than the nominal.

The structural element of nonwovens obtained by extraction of matrix polymer from composite films formed from melts of the original and nanofilled PP/CPA mixtures on a worm press through a flat-slot head of the "fishtail" type were practically continuous PP microfibrils with diameters from tenths of a fraction to several micrometers [57,59,60]. The advantage of FMs obtained by extraction of matrix polymer from composite films with microfibrillar morphology is an ordered homogeneous structure - microfibrils in the filter layer are oriented in the direction of extrusion and are located parallel to each other. To improve the performance of FM, the structure of the filter layer was modified by adding nanofillers to the mixture: pyrogenic silica (SiO_2) and bifunctional substances based on it - silver/silica (Ag/SiO_2) and titanium oxide/silica (TiO_2/SiO_2) [57]. The introduction of all additives contributed to an increase in the mass fraction of microfibrils (W) (Fig. 1.8) and a decrease in their average diameter (d) (Fig. 1.9).

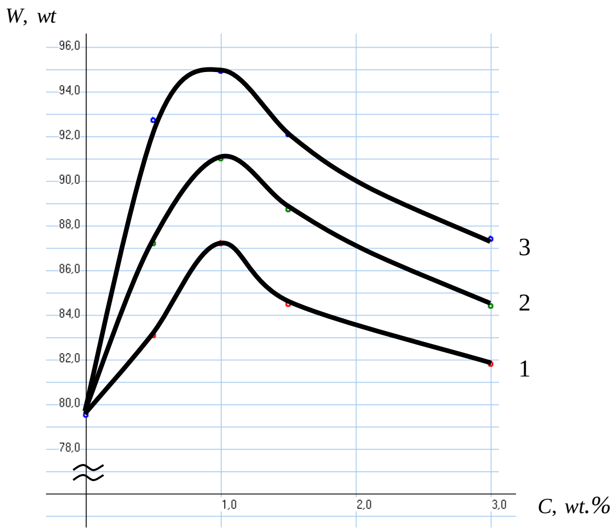


Fig. 1.8. The influence of the concentration and chemical nature of nanoadditives on the mass fraction of microfibrils in the extrudate: 1 – Ag/SiO_2; 2 – SiO_2; 3 – TiO_2/SiO_2

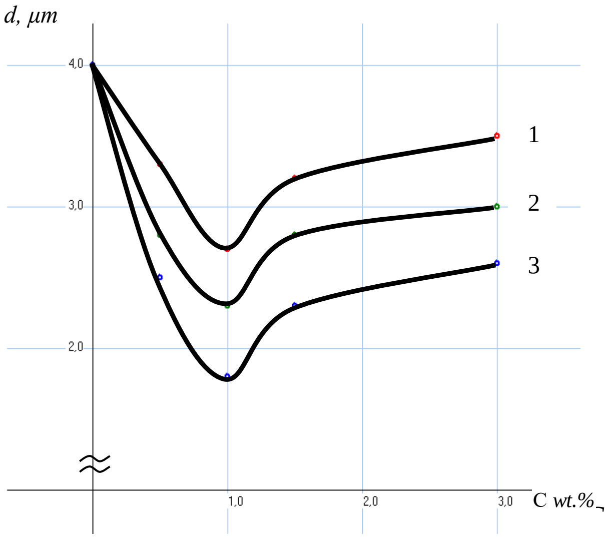


Fig. 1.9. Effect of concentration and chemical nature of nanoadditives on the average diameter of microfibrils: 1 – Ag/SiO_2; 2 – SiO_2; 3 – TiO_2/SiO_2

As can be seen from Fig. 1.8 and 1.9, the effect of nanoadditives on the morphology of the mixture depends on their chemical nature and concentration. Nanoparticles of the mixed oxide TiO_2/SiO_2 are more effective, compared to the original silica and silver/silica NPs: the diameters of the fibrils are the smallest, and their number in the extrudate is the largest, which is due to the high polarity of the oxide. The lack of interaction with the melt of non-polar polypropylene promotes the migration of NPs from the volume of the PP melt to the phase boundary of the components and localization in it. The preferential placement of nanoadditives in the transition layer most effectively reduces the value of the surface tension and contributes to the formation of a finer morphology [3,93].

The curves of dependences $W = f(C)$ and $d = f(C)$ have an extreme character – at the additive content of 1.0 wt. % the average diameter of microfibrils is minimal, and their mass fraction reaches maximum values for all investigated additives. The decrease in the dimensional characteristics of microfibrils is due to the compatibilizing effect of nanofillers. With an increase in the concentration of NPs of all additives > 1.0 wt. %, the structure coarsens. This may be due to the saturation of the interfacial zone with the modifier. A similar effect of reducing the surface activity of natural clay upon reaching a certain concentration was also observed by the authors [94,95]. For the natural rubber (NR)/PP mixture, the dimensional characteristics of the NR particles in the PP matrix decreased linearly only at a clay content of up to 5.0 wt. % [94]. The introduction of organomodified montmorillonite into the PP/polystyrene mixture in an amount of (0.2÷2.0) wt. %

contributes to a decrease in the dimensional characteristics of the microstructure in the entire concentration range. The best result is achieved at a clay content of 0.5 wt. %: the diameter of microfibrils decreases by 1.6 times and the uniformity of their distribution increases by ~ 5 times [95].

Analysis of the results of assessing the efficiency of atmospheric air filtration from mechanical particles with a size of (0.3÷1.0) μm by filter material from the initial and three-component mixtures containing 1.0 wt. % of nanofiller shows that the introduction of nanoadditives into the structure of polypropylene microfibrils provides an increase in the precision and efficiency of FM (Table 1.9).

Table 1.9 – The influence of the chemical nature of nanoadditives on the efficiency of atmospheric air purification and the performance of filter materials

Additive name	Efficiency, % (by particle size, μm)						Productivity* $dm^3/m^2\cdot$ hour
	0,3	0,4	0,5	0,6	0,8	1,0	
without additives	78,6	83,5	85,9	87,8	91,9	99,4	4050
SiO_2	99,8	100	100	100	100	100	10650
Ag/SiO_2	99,3	99,9	100	100	100	100	10840
TiO_2/SiO_2	99,9	100	100	100	100	100	12230

* at a pressure of $0.5 \cdot 10^5$ Pa

As can be seen from Table 1.9, the introduction of nanoadditives into the structure of the filter layer provides an increase in the retention capacity of PMs and their precision. At the same time, the values of the purification efficiency are indirectly correlated with the dimensional characteristics of the structural elements of the filter layer - the retention of particles with a size of 0.3 microns with maximum efficiency (99.9%) is demonstrated by PMs with mixed oxide additives. All modified filter materials retain mechanical impurities with a size of 0.5 microns and above with an efficiency of 100%, and from the original mixture only 85.9%. Such an improvement in one of the main indicators of filters is due, first of all, to an increase in the uniformity of the structure of the filter layer due to a decrease in the average diameter of microfibrils by almost 2 times and improvement of their shape. When cleaning media from mechanical impurities with a size of $\leq$ 1.0 μm through fibrous filter materials, in addition to the so-called “sieve” effect, a number of physicochemical processes play a significant role, namely: the contact effect, adsorption, Brownian diffusion [92]. Due to this, FM can retain particles with diameters 5 times smaller than the pore size. The decisive importance of the adsorption process is evidenced by a sharp increase in the specific surface area to 84 m^2/g for the original and to (190÷352) m^2/g for nanofilled PP microfibrils, compared to fibers formed using classical technology (Tables 1.6, 1.10, 1.11).

Table 1.10 – The influence of the chemical nature of nanoadditives on the specific surface area and hygroscopicity of polypropylene microfibrils

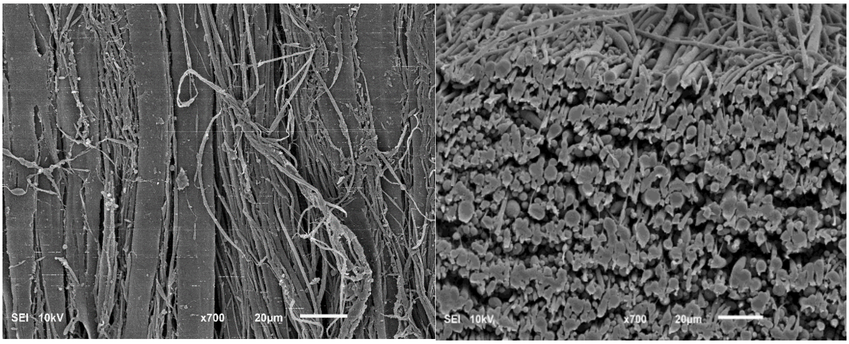
Additive name	Specific surface area, m^2/g	Hygroscopicity, %
without additives	84	0,17
SiO_2	244	0,35
Ag/SiO_2	230	0,31
TiO_2/SiO_2	190	0,48

Table 1.11 – Effect of silica concentration on the specific surface area of PP microfibrils

Silica content, wt. %	Specific surface area, m^2/g	Growth rate
0	84	0
0,5	197	2,3
1,0	244	2,9
3,0	307	3,7
5,0	352	4,2

The permeability of filters is determined by the pressure drop on both sides of the filter partition and the resistance of the material to the medium being cleaned. Studies of the performance of FM on distilled water showed its increase for samples in the structure of which there are nanoparticles of the original and modified silicas (Table 1.9). This is an unexpected result, since an increase in the precision and efficiency of filters of any class is usually accompanied by a decrease in their permeability. The increase in performance is obviously due to a decrease in the hydraulic resistance of the filter layer due to the better hydrophilicity of nanofilled PP microfibrils (Table 1.10). An additional factor that ensures the maximum performance of FM modified with a mixed oxide is the ability of materials with additives of TiO_2 nanoparticles to self-clean.

An effective method of regulating the structure and operational properties of filter materials, which allows to significantly expand the spectrum and areas of their application, is the use of the 3D molding method (*FDM* process) to obtain composite films with microfibrillar morphology [64,65]. Studies performed using a PP/CPA mixture have shown that when composite multilayer films are formed by FDM from strands with microfibrillar morphology, they retain the structure laid down during extrusion. Nonwoven filter material from such films can consist of several layers, in each of which polypropylene microfibrils are oriented in one direction and are located parallel to each other, and the layers are perpendicular to each other (Fig. 1.10). This provides the FM with increased mechanical performance and a uniform ordered morphology.



a) b)

Fig. 1.10 – Electronic microphotographs of the filter material: a) surface layer, b) cross section

In [64], it was shown that the filtration efficiency and precision of FM were increased by reducing the diameters of microfibrils in the strands, which was achieved by changing the size of the cells of the filtration meshes and the pressure before the die during processing on a single-screw extruder. The possibility of regulating the dimensional characteristics of PP microfibrils in the filter layer by changing the type of equipment for compounding the ingredients of the mixture (single-screw or twin-screw extruders) was also established

[65]. The introduction of zirconium dioxide (ZrO_2) nanoparticles into the system is an additional factor that provides regulation of the microfibrillar structure of composite films and filter materials based on them in the direction of reducing the diameters of microfibrils and narrowing their distribution. Thus, if the components are mixed on a twin-screw extruder and 2.5 wt. % of ZrO_2 nanoparticles are added to the mixture, the thinnest microfibrils are formed (the average diameter is 640 nm) with a narrow distribution in transverse dimensions. The results of evaluating the efficiency of atmospheric air purification from mechanical impurities with a size of (0.3÷1.0) microns show that it depends on the diameters of microfibrils and the number of layers (Table 1.12).

Table 1.12 – Efficiency of atmospheric air purification from mechanical impurities

Composition of the mixture for obtaining FM	Number of layers	Filtration efficiency, % (by particle size, μm)							
		0,3	0,4	0,5	0,6	0,7	0,8	0,9	1,0
PP/CPA	2	83,8	88,4	90,1	93,3	99,4	96,6	99,7	99,9
	4	95,2	97,2	97,2	99,1	99,8	99,8	99,9	100
	6	96,2	97,5	97,9	99,7	100	100	100	100
PP/CPA / ZrO_2	2	94,1	97,3	98,9	99,9	100	100	100	100
PP/CPA *	1	78,6	83,5	85,9	87,8	89,3	91,9	97,4	99,4

* FM from film obtained by extrusion method

The retention capacity of filters naturally increases with an increase in the number of layers, which is the result of an increase in the uniformity of the pore morphology of the material. The reduction of microfibril diameters to nanosizes in the filter layer obtained from the PP/CPA/ZrO_2 composition leads to a further improvement of one of the main characteristics of the PM - a two-layer filter provides the efficiency of gas medium purification at the level of materials without filler, which consist of 4-6 layers.

Conclusion

Today, nanofilled fibrous synthetic materials are industrially developed and widely used to provide products made from them with the desired consumer effects. The presence of substances in the nanoscale in the structure of fibers and threads helps to improve their mechanical properties, and in polymer mixtures enhances the self-reinforcing effect of products made from them.

In industry, nanofilled polymeric fibrous materials are successfully used to manufacture new types of filters capable of self-cleaning and preventing pollution. On their basis, a new generation of effective sorbents is created for cleaning technological environments, including the medical industry. In terms of their performance, they exceed ion-exchange resins, while being 5 times cheaper than them. Such nanocomposites absorb a wide range of metal ions from water, destroy organic compounds, concentrate and separate radionuclides.

Polymer modification with nanoadditives also allows solving environmental and social problems. Composites for water purification and new environmentally friendly adsorbents for environmental restoration have been created on the basis of biopolymers. The development and implementation of modern "green" technologies allows recycling and using secondary polymer resources.

Today, despite the large number of existing varieties of polymer composites, the trend towards their further improvement and the creation of new modern fibrous nanofilled materials is constantly growing. Analysis of the data presented in section 1 on the influence of nanofillers on the performance characteristics of fibrous materials shows that one of the determining factors is the content of modifying additives in the structure of materials. In the technology of multicomponent systems, the main criterion for testing theoretical hypotheses remains the results of experiments, which are laborious and long-term.

Based on this, an important task is to maximally reduce the transition time from laboratory experiments to industrial samples. An effective means of increasing the efficiency of scientific research is the creation of software for mathematical modeling of experiments, processing their results, and optimizing the composition of multicomponent mixture systems in order to obtain modern fibrous materials with improved properties.

CHAPTER 2. SOFTWARE FOR MATHEMATICAL EXPERIMENTAL PLANNING AND OPTIMIZATION OF THE COMPOSITION OF MULTICOMPONENT SYSTEMS

In chemical technology, most of the studied objects belong to the class of complex systems, which are characterized by a large number of interconnected parameters. The task of studying such systems is to establish the dependence between the input parameters - factors and output parameters - indicators of the quality of the system's functioning, as well as to determine the levels of factors that optimize its output parameters. Today, there are two approaches to solving the problems of identification and optimization of complex systems: deterministic and stochastic [96]. In the first method, before solving extreme problems, a comprehensive study of the mechanisms of the phenomenon is carried out, on the basis of which the system is given by a clearly deterministic model (usually in the form of a system of differential equations). In this case, the developed mathematical apparatus of modern control theory can be used to solve the optimization problem. However, such systems, due to the complexity of a comprehensive study of the mechanism of the phenomenon, are not amenable to a complete mathematical description in a reasonable time, which limits the application of the deterministic approach. In the absence of complete knowledge of the mechanism of phenomena, identification and optimization problems, i.e., finding optimal conditions for the course of processes or optimal selection of the composition of multicomponent systems, are solved using experimental and statistical methods.

In the case of experimental and statistical studies of the object, the relationship between the input and output parameters of the system is usually described by a polynomial. To estimate the coefficients of the polynomial that approximates the real dependence (response function ψ), it is necessary to have statistical data that characterize the state of the system during operation. This information can be obtained by passive or active experiment (setting up experiments at certain points $X_u = (x_{1u}, x_{2u}, ..., x_{ku})$ $(u = 1,2,...N)$ of the permissible region of the space of controlled input parameters. Provided that the influence of uncontrolled input parameters is insignificant compared to controlled ones, the system under study can be described by the following model:

$$\xi (y) = \varphi (X) + \acute{\varepsilon} \qquad (2.1)$$

Today, in experimental and statistical research, the method of mathematical planning of an experiment is widely used, the essence of which is to select the number of experiments and the conditions for their conduct, necessary and sufficient to solve a given problem with the required accuracy, methods for mathematical processing of their results and decision-making. In experimental planning, the experiment itself is considered as an object of research and optimization, in which optimal control of the experiment is carried out. Depending on the information about the system under study, the research strategy changes in the direction of its optimization for each specific stage. Experimental planning is a powerful tool in conducting research and optimizing complex systems, which allows you to significantly reduce the number of experiments, and, thus, material costs and terms of conducting experiments, makes it possible to obtain mathematical models and quantitative assessments of the influence of various factors on the processes under study. The use of experimental planning methods, in comparison with traditional methods, allows you to increase the efficiency of scientific research by up to 10 times. A mathematical model is a system of mathematical relationships - formulas, functions, equations that describe the object under study. Analytical recording of the property–composition dependence has a number of advantages over geometric methods of spatial representation of complex surfaces for multicomponent systems, namely: determination of property indicators directly by calculation, its versatility, the possibility of application in many fields of research (chemistry and chemical technology, metallurgy, the building materials industry, medicine, biology, agriculture, etc.). In addition, the problem is formalized, and the obtained dependences can be calculated using software.

2.1. Basic concepts of the mathematical design of experiments method

2.1.1. Factors, optimization parameters and models. During experiments, they usually deal with objects of research, which can be: technological processes, various compositions, products, etc. For them, input parameters are distinguished - controlled factors x_1, x_2, ..., x_p, corresponding to the effects on the system, and output (quantitative characteristics of the research goal) - optimization parameters (criteria) y_1, y_2, ..., y_l. In this case, the model of the research object can be represented as a cybernetic system with $k+n+l$ inputs and m outputs (Fig. 2.1) [96].

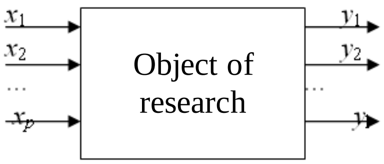


Fig. 2.1 – Model of the research object

Each of the output parameters depends on the state of the controlled part of the inputs, which is determined by the *k*-dimensional vector $X = (x_1, x_2, ..., x_k)$; the controlled uncontrolled part of the inputs, which is described by the *n*-dimensional vector $Z = (z_1, z_2 ... z_n)$; the uncontrolled part, which is determined by the *l*-dimensional vector $E = (e_1, e_2 ... e_l)$, and the outputs, i.e. the numerical characteristics of the research objectives, are the optimization parameters (criteria) $y_1, y_2, ..., y_l$; $y = F(X,Z,E)$. During the experiments, each factor can take one of several values, called levels. A fixed set of factor levels determines one of the possible states of the cybernetic system. At the same time, this set represents the conditions for conducting one of the possible experiments. Each fixed set of factor levels corresponds to a certain point in a multidimensional space, called the factor space. Experiments cannot be implemented at all points of the factor space, but only at those that belong to the admissible region of the factor space *G* (Fig. 2.2).

The system reacts differently to different sets of factor levels. At the same time, there is a certain relationship between the factor levels and the reaction (response) of the system. The function ψ, which connects the optimization parameter with the factors, is called the response function, and the geometric image corresponding to the response is the response surface (Fig. 2.3).

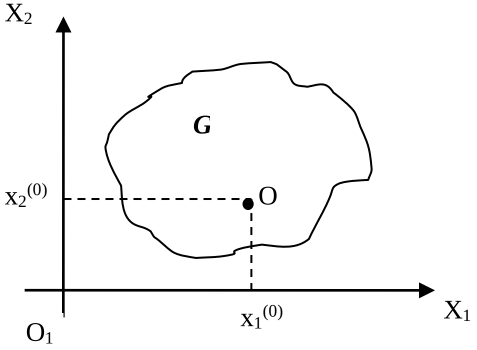


Fig. 2.2 Factor space

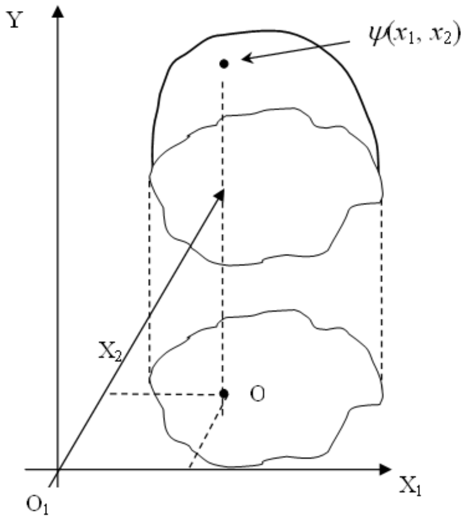


Fig. 2.3. Response surface

The form of the dependencies ψ_j for the system under study is unknown in advance, therefore it is necessary to obtain approximate equalities based on experimental data:

$$\hat{y}_j = \hat{\psi}_j(x_1, x_2, ..., x_k), \quad j = 1, 2, ..., l.$$

The experiment must be conducted in such a way that, provided that the minimum number of experiments is performed, varying the values of the independent variables according to specially formulated rules, a mathematical model of the system can be constructed and the optimal values of its properties can be found.

The selection of factors, optimization parameters and models takes place taking into account the purpose of the study and the existing conditions for conducting the experiment. Factors are variables that acquire a certain value at some point in time. They determine both the object itself and its state. There are quantitative and qualitative factors. Quantitative factors are variables that can be evaluated quantitatively, namely: measured, weighed, etc.; qualitative factors do not have a numerical assessment, but for them it is possible to construct a conditional scale that carries out coding, establishes a correspondence between the levels of the qualitative factor and the numbers of the natural series. Factors can also be controlled and uncontrolled. Controlled are such input variables, the values of which in the experiment are known at each point in time. Thus, when studying a technological process, all variables that determine the state of the process and the values of which can be estimated using appropriate measuring instruments are controlled. Controlled variables, in turn, can be divided into controlled and uncontrolled. Controlled factors are those whose values can be purposefully varied during the experiment. Factors for which such a change is impossible are called uncontrolled. These are input variables whose values cannot be estimated during the experiment, or those that have an impact on the results of the experiment, or even factors about the existence of which the experimenter has no information.

The characteristic of the goal of an experiment or research, given quantitatively, is called an optimization parameter (optimization criterion, objective function). Optimization parameters can be economic (profit, cost, profitability, experiment costs, etc.), technical and economic (productivity, stability, reliability, efficiency, etc.), technical and technological (product yield, physical, mechanical, physicochemical, medical and biological characteristics).

A number of requirements are put forward for the optimization parameter: effectiveness in terms of achieving the goal (i.e., the optimization parameter should evaluate the functioning of the system as a whole, and not its individual subsystems); universality (the ability to comprehensively characterize the object of study); quantitative expression in a single number; the presence of a physical essence; simplicity and accessibility of calculation. The number of values that an optimization parameter can take is called its definition domain; they can be continuous and discrete, limited and unlimited. The researcher must be able to determine the optimization parameter for any possible combination of selected levels of factors.

For planning experiments, models in the form of algebraic polynomials have found the greatest application. To choose a specific model, it is necessary to formulate certain requirements. These include adequacy (the ability of the model to predict the results of the experiment in a certain area with the required accuracy); meaningfulness (the model must well explain already known facts, identify new ones and predict the further behavior of the system); simplicity (the simpler the model, the better it is, other things being equal).

Depending on the problem statement, different models can be chosen. Explicit functional dependencies of the form are often used:

$$y = f(x_1, x_2, \dots x_p, \beta_1, \beta_2, \dots \beta_m, \varepsilon), \qquad (2.2)$$

where: f – some function called the regression function; $x_1, x_2, \dots x_p$ – independent variables (factors); $\beta_1, \beta_2, \dots \beta_m$ – dependence parameters; ε – random component. The latter is introduced into the model when the data show noticeable variability of a random nature. It is very often assumed that ε enters model (2.2) additively, then it takes the form:

$$y = f(x_1, x_2, \dots x_p, \beta_1, \beta_2, \dots \beta_m) + \varepsilon \qquad (2.3)$$

Relations (2.2), (2.3) are called regression models.

For independent factors $x_1, x_2, \dots x_p$, the researcher chooses certain values, and experimentally obtains the corresponding values y. Then (2.3) passes into a system of relations from which the parameters $\beta_1, \beta_2, \dots \beta_m$ are determined. Due to the presence of a random component, the

$\beta_1, \beta_2, ... \beta_m$ parameters can only be estimated (and not precisely determined). In this case, estimates $b_1, b_2, ... b_m$ of the corresponding parameters are obtained, and instead of model (2.3) in reality, an approximation $\hat{y}$ to it is operated:

$$\hat{y} = f(x_1, x_2, ... x_p, b_1, b_2, ... b_m).$$

If the function f is a polynomial, then $b_1, b_2, ... b_m$ are called regression coefficients, and the function takes the form:

$$\hat{y} = b_0 + \sum_i b_i x_i + \sum_{i,j} b_{ij} x_i x_j + ... \qquad (2.4)$$

2.2. Mathematical planning of an experiment

Solving problems using mathematical methods is carried out by formulating the problem, choosing a research method, a mathematical model and analyzing the result obtained. The mathematical formulation of the problem is presented in the form of numbers, geometric shapes, functions, systems of equations, etc.

The main stages of mathematical planning are as follows: setting the problem, defining the object and purpose of the research, studying objects, etc.; choosing the type of mathematical model (often several models are built and the best one is chosen); describing the transformation of input signals into output characteristics of the object (for example, using algebraic dependencies); studying the quality of the developed models [96].

After selecting the type of model, i.e. the type of dependence of y on x and writing the corresponding equation, in the area of the factor space allocated for research, an experiment is planned. Then, experiments are carried out to estimate the numerical values of the constants (coefficients) of this equation. Since the polynomial (2.4) has C^d_{k+d} coefficients that need to be determined, the experiment plan

$$D = \begin{pmatrix} x_{11} & x_{21} & & x_{k1} \\ x_{12} & x_{22} & & x_{k2} \\ & & & \\ x_{1N} & x_{2N} & & x_{kN} \end{pmatrix}$$

must contain at least C^l_{k+d} different experimental points X_u = (x_{1u}, x_{2u}, ..., x_{ku}), u = 1,2,...N.

2.2.1. Determination of regression coefficients by the least squares method. According to the results of the experiment on the object of study, a mathematical model of a certain form is obtained. In particular, it can be a regression model with a regression function in the form of a polynomial of the appropriate degree - the so-called polynomial regression model. The quality of the regression model's approximation to the real object depends not only on the experimental data, but also on the method of processing the results used to build the model. For this purpose, the least squares method (*LSM*) is often chosen. In this case, it is assumed that n experiments are performed, in each of which the vector of independent factors x = (x_1,…x_p) is given certain values. As a result, some values of the dependent variable y are obtained. Provided that x^i = (x^i_1,…, x^i_p) is a set of values of the dependent variables that were given to them in the i-th experiment, then y_i are the corresponding values of the dependent variable (i = 1,2,…, n). To estimate the parameter vector β = (β_1,…, β_m), we choose such a vector $b = (b_1,\ldots, b_m)$ for which the sum $S(\beta)$ (2.5) takes on a minimum value by $\beta \in R^m$:

$$S(\beta) = \sum_{i=1}^{n} \left[y_i - f(x^i;\beta) \right]^2 \qquad (2.5)$$

where: R^m – m-dimensional Euclidean space.

If the regression function f is differentiable with respect to the parameters $(\beta_1,\ldots,\beta_m)$, then the necessary condition for the minimum of $S(\beta)$ is that the equalities

$$\frac{\partial S(\beta)}{\partial \beta_j} = 0,\ j = 1,2,\ldots,m\,. \qquad (2.6)$$

System (2.6) consists of equations, the number of which is equal to the number of unknowns of the system – coefficients $b_1, b_2, \ldots, b_m$, and is called a system of normal equations or a normal system.

The solution to the problem of minimizing the function $S(\beta)$is given below for a special case of model (2.6) provided that $p = 1$, the vector of independent variables x is a scalar variable, and $m = 2$. In this case, instead of the notations β_1, β_2, the more common β_0, β_1 will be used for the dependence parameters. It is also assumed that the function f is linear in the parameters β_0, β_1, i.e. in the expression of the function f, the variable x is present only in power 1. Then the regression function f takes the form:

$$f(x) = \beta_0 + \beta_1 x, \qquad (2.7)$$

and, thus, the following partial case of model (2.6) will be investigated:

$$y = \beta_0 + \beta_1 x + \varepsilon, \qquad (2.8)$$

where: x and y – respectively, independent and dependent variables, β_0, β_1 – model parameters, ε – random component of the model

Equation (2.8) is called simple linear regression.

To estimate the parameters β_0, β_1 from experimental data, it is assumed that the independent variable x in the experiments takes the values $x_1,\dots, x_n$, and the dependent variable y – respectively, $y_1,\dots, y_n$. In this case, the problem of minimizing the function $S(\beta)$ takes the form:

$$S(\beta)= S(\beta_0, \beta_1) = \sum_{i=1}^{n}\left[y_i-(\beta_0+\beta_1 x_i)\right]^2 \to min, \quad (2.9)$$

where the minimum is taken for all values of β_0, β_1 for fixed $x_1,\dots, x_n$ and $y_1,\dots, y_n$. If the solution to problem (2.9) is denoted by (b_0, b_1), and the corresponding estimate of the regression function (2.7) is ŷ, then

$$\hat{y} = \hat{y}(x) = b_0 + b_1 x \quad (2.10).$$

Fig. 2.4 schematically depicts the regression line (2.10) and a set of experimental points (x_i, y_i), as well as vertical segments (deviations) connecting the indicated points and the line. These deviations are measured by the differences of the ordinates corresponding to the experimental points and the points of the approximating line for the values $x = x_1 ,\dots, x_n$, that is, by the algebraic values of the vertical segments shown in Fig. 2.4. In this case, the sum of the squares of the lengths of such segments will be the smallest possible. The value β_1 is the slope, and β_0 is the free member of the line (the segment on the ordinate axis at $x = 0$), and b_1, b_0 are their estimates from the experimental data. They are, respectively, the slope and the free member of the equation of the line (2.10).

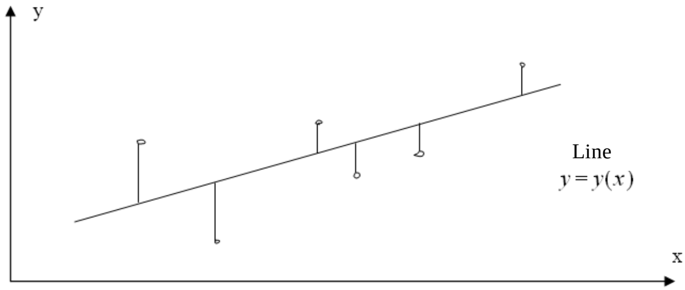


Fig. 2.4. Regression line with vertical deviations

To solve problem (2.9), calculate the partial derivatives with respect to β_0, β_1 of the function $S = S(\beta_0, \beta_1)$, which have the following form:

$$\partial S / \partial \beta_0 = -2 \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i),$$

$$\partial S / \partial \beta_1 = -2 \sum_{i=1}^{n} x_i (y_i - \beta_0 - \beta_1 x_i).$$

By equating the found derivatives to zero and performing appropriate simplifications, we obtain a system of two equations with unknown parameters β_0, β_1:

$$\beta_0 n + \beta_1 \Sigma x_i = \Sigma y_i,$$

$$\beta_0 \Sigma x_i + \beta_1 \Sigma x_i^2 = \Sigma x_i y_i, \qquad (2.11)$$

In equations (2.11), to simplify the notation, the summation indices are omitted (here and in similar situations, the sign Σ means summation over all possible values of the index, in this case from 1 to *n*). This system is a partial case of the normal equations (2.6). The solution of the normal system (2.11) is the solution to the minimization problem (2.9). The system of equations (2.11) is always consistent, regardless of whether its determinant is zero or not. The equality of zero of the specified determinant can occur only in the case when all observations are made at only one value of x. In this case, the specified system has many solutions, each of which can be found from the equation:

$$\beta_0 n + \beta_1 n\, x = \Sigma\, y_i, \qquad (2.12)$$

Provided that the main determinant of the system of equations (2.11) is not equal to zero, the system has a unique solution, for which the following notation is introduced:

$S_{x\,y} = \Sigma(x_i - \bar{x})(\,y_i - \bar{y}\,)$, $S_{x\,x} = \Sigma(x_i - \bar{x}\,)^2$, $S_{y\,y} = \Sigma(\,y_i - \bar{y}\,)^2$,

where summation indices are omitted.

In the case where $\bar{y} = (y_1 + \ldots + y_n)/n)$, та $\bar{x} = (x_1 + \ldots + x_n)/n$ are the arithmetic mean values of the independent and dependent variables, the solution of system (2.11) takes the form:

$$b_1 = S_{x\,y} \,/\, S_{x\,x}\,, \qquad (2.13)$$

$$b_0 = \bar{y} - b_1 \bar{x}\,. \qquad (2.14)$$

Thus, in the case of simple linear regression, the relationship model between the objective function y and the independent variable x is given by equality (2.10), in which the coefficients b_0, b_1 are determined by equations (2.13), (2.14).

Provided that certain probability assumptions about the nature of the sample data $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ are met, then the model (2.10) also has the corresponding properties of a probabilistic nature. This makes it possible to assess the quality of the constructed model, find confidence intervals for its values, and perform forecasting using regression analysis and planning of experiments.

2.2.2. *Model adequacy checking.* After determining the coefficients of the developed mathematical model (2.10), the hypothesis of the adequacy of the regression equation is tested, i.e., the possibility of using the obtained equation for further research is determined, or the need to build another model is determined. The procedures for this test are conventionally divided into analytical and graphical methods. For analytical testing of the adequacy of the model, the difference between the experimental value and the response value predicted by the regression equation at some points of the factor space, which can be selected from the points of the plan (for unsaturated plans), or from additional control points, is studied. Control points are usually chosen either in the area of greatest interest, or placed in such a way that observations in them can be used to construct a polynomial of higher degree.

The implementation of the analytical method involves making more than one observation at least at one of the points $\{x_j\}$. Provided that $x_1, x_2,\ldots, x_n$ are observation points, and $n > 1$, they are all considered different.

The dependent variable y, up to the random additive error ε, can be represented as a linear combination of factor variables (independent variables, regressors) $x_0, x_1,\ldots, x_{p-1}$:

$$y = \beta_0 x_0 + \ldots + \beta_{p-1} x_{p-1} + \varepsilon, \tag{2.15}$$

where: $\beta_0,\ldots, \beta_{p-1}$ – coefficients of the mathematical model

As a result, a sample of size n was made, which is a set of experimentally obtained n sets of numbers of the form:

($x_{i0},\ldots,x_{i,p-1}, y_i$), $i = 1,2,\ldots,n$, where: x_{ij} is the value of the jth regressor (j-th independent variable) at the i-th observation, y_i - is the corresponding value of the dependent variable y. The error value ε at the i-th observation is denoted by ε_i.

To check the adequacy of a linear model, a fairly common method is to compare estimates of error variances obtained, on the one hand, using this model, and on the other hand, independently. This is equivalent to testing some linear hypothesis by calculating and analyzing the corresponding Fisher's F-ratio.

At the first stage, the experimental data were marked with the letter xi for the i-th observation point (row vector) of the independent variable, i.e. $\mathbf{x}_i = (x_{i0}, \ldots, x_{i,p-1}), i = 1, 2, \ldots, n$. Since this method requires the presence of several observations for y at least at one of the points $\mathbf{x}_i$, it is assumed that this requirement is met, i.e. among the points xi there are some that are repeated. In this case, $\mathbf{x}_1, \mathbf{x}_1, \ldots, \mathbf{x}_m$ – are different observation points, and at least in one of them the number of observations is greater than 1. The specified F-statistic has the following form:

$$F = \frac{S_1^2}{S_2^2} \qquad (2.16)$$

де: $$S_1^2 = \frac{1}{m-p}\sum_{i=1}^{m} n_i\left(\hat{y}_i - \bar{y}_i\right)^2,$$

$$S_2^2 = \frac{1}{n-m}\sum_{i=1}^{m}\sum_{j=1}^{n_i}(y_{ij} - \bar{y}_i)^2$$

y_{i1}, …. y_{in_i}, i = 1,…., m – the values of the output variable observed at the point **x** = **x^i**; n_i - number of experiments at the i-th point

Provided that $m > p$, the relation of the form $\frac{S_1^2}{S_2^2}$ (a variant from the set of F-relations) has a Fisher distribution $F(m-p, \quad n-m)$ [33,97]. According to the general provisions, the hypothesis of the adequacy of the model is not accepted at the significance level α if the specified relation exceeds the quantile of the level (1-α) of the Fisher distribution. Otherwise, the hypothesis is accepted. To check the adequacy of the model by the described method, the software developed by us can be used [33].

2.2.3. Full and fractional factor experiments. In experimental studies, each of the different values that a variable X_i takes is called a level of that variable. The number of different levels of a factor X_i is denoted by S_i. An experiment in which the levels of each factor are combined with all levels of the other factors is called a full factor experiment (FFE). A full factor experiment is written as: $S_1 \times S_2 \times ... \times S_k$, since the number of different points or different experiments is $N_1 = S_1 \times S_2 \times ... \times S_k$. An experimental plan is called an incomplete or fractional factor plan if the number of different points is $N_1 < S_1 \times S_2 \times ... \times S_k$. Provided that in the response function

$$\eta = f(X_1, X_2, ..., X_k) \tag{2.17}$$

the number of different values that a variable X_i $(i = 1,2,...,k)$ can take in all experiments is two, i.e. S_i=2. In other words, the variable X_i in each experiment takes one of two possible values (X_{i1} and X_{i2}), or varies at two levels. If X_{i1} < X_{i2}, then X_{i2} is called the upper level of the factor X_i, and X_{i1} is called the lower level. To simplify the equations, coded variables are introduced: $$x_i = \frac{X_i - X_i^0}{S_i}, \qquad i = 1,2,...,k,$$

where: $$X_i^0 = \frac{X_{i1} + X_{i2}}{2} \quad i = 1,2,...,k;$$

$$S_i = \frac{X_{i2} - X_{i1}}{2} \quad i = 1,2,...,k.$$

The coded variable x_i ($i = 1,2,...,k$) in each experiment can take the values 1 or -1, which are its upper and lower levels. Without loss of generality, we can assume that expression (2.17) with variables $X_1, X_2, ..., X_k$ presented in coded variables form has the following form:

$$\eta = f(x_1, x_2, ..., x_k) \tag{2.18}$$

In the case when in expression (2.17) the number of independent variables $k = 2$, then $\eta = f(x_1, x_2)$. All possible combinations of levels of variable x_1 i x_2 in a full factor experiment 2^2 are presented in Table 2.1

Table 2.1 – Matrix of plan FFE 2^2

Experiment number	Matrix of independent variables				Research option	Observation
	x_0	x_1	x_2	$x_1 x_2$		
1	1	-1	-1	1	(1)	Y_1
2	1	1	-1	-1	a	Y_2
3	1	-1	1	-1	b	Y_3
4	1	1	1	1	ab	Y_4

In the table, the symbol (1) means that both factors are in the lower level; a – x_1 in the upper level; b – x_{21} in the upper level; ab – both in the upper level. This is a full factor experiment 2^2. Often the response function has the form:

$$\eta = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 x_2 \qquad (2.19)$$

A schematic representation of the FFE 2^2 is shown in Fig. 2.5.

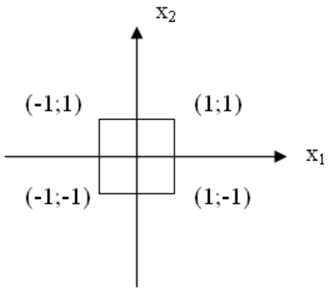


Fig. 2.5 – A schematic representation of the FFE 2^2

From Fig. 2.5 it is seen that the observations y_1, y_2, y_3, y_4 are made at the vertices of the square. The coefficients of equation (2.19) can be calculated by the method of least squares.

The response function of FFE 2^3 has the form: $\eta = f(x_1, x_2, x_3)$. All different combinations of variable levels are presented in Table 2.2.

Table 2.2 – Matrix of plan FFE 2^3

Matrix of independent variables								Research option	Observation
x_0	x_1	x_2	x_3	$x_1 x_2$	$x_1 x_3$	$x_2 x_3$	$x_1 x_2 x_3$		
1	-1	-1	-1	1	1	1	-1	(1)	y_1
1	1	-1	-1	-1	-1	1	1	A	y_2
1	-1	1	-1	-1	1	-1	1	B	y_3
1	1	1	-1	1	-1	-1	-1	ab	y_4
1	-1	-1	1	1	-1	-1	1	C	y_5
1	1	-1	1	-1	1	-1	-1	ac	y_6
1	-1	1	1	-1	-1	1	-1	bc	y_7
1	1	1	1	1	1	1	1	abc	y_8

The response function is calculated from the equation:

$$\eta = \beta_0 + \sum_{1 \le i \le 3} \beta_i x_i + \sum_{1 \le i < j \le 3} \beta_{ij} x_i x_j + \beta_{123} x_1 x_2 x_3 \quad (2.20)$$

The coefficients (2.20) are determined by the least squares method.

In a full factor experiment 2^k the number of experiments is $N = 2^k$. As the number of variables k increases, the number of experiments N increases rapidly, so for large values of k, the implementation of FFE 2^k becomes practically impossible. For FFE 2^k experiments the response function has the following form:

$$\eta = \beta_0 + \sum_{1 \le i \le k} \beta_i x_i + \sum_{1 \le i < j \le k} \beta_{ij} x_i x_j + \ldots + \beta_{12\ldots k} x_1 x_2 \ldots x_k \quad (2.21)$$

With the growth of N there is an increase in the number of interactions and their order in (2.21), but often in the specified equation the effects of high-order interactions can be neglected, or it is known a priori that some of them are absent. The number of experiments to find estimates of the unknown coefficients of such an equation can be significantly reduced. This is achieved by using fractional factor experiments. If in FFE 2^k observations are carried out at all vertices of the k -dimensional hypercube, then when using fractional plans - only at some of them.

Below is an example of constructing a fractional replica, in which the response function has the form:

$$\eta = \beta_0 + \sum_{1 \le i \le 3} \beta_i x_i \tag{2.22}$$

In this expression, the effects of pair and triple interactions are absent $\beta_{12} = \beta_{13} = \beta_{23} = \beta_{123} = 0$.

If FFE 2^3 is used to estimate the unknown coefficients, then N=8. However, the number of experiments can be reduced, since in (2.22) there are no interaction effects. For this purpose, a plan is constructed, the matrix of which has the form:

$$D = \begin{array}{c} \begin{array}{ccc} x_1 & x_2 & x_3 \end{array} \\ \begin{pmatrix} -1 & -1 & 1 \\ 1 & -1 & -1 \\ -1 & 1 & -1 \\ 1 & 1 & 1 \end{pmatrix} \\ \underbrace{\qquad\qquad}_{\text{матриця ПФЕ } 2^2} \end{array} \tag{2.23}$$

The matrix D is obtained from the matrix FFE 2^3 by deleting individual rows from it: (1; -1; 1), (-1; 1; 1), (-1; -1; -1), (1; 1; -1). The constructed fractional factor experiment (FFE) design (2.23) is a half-replica of FFE 2^3. For its recording, the notation is used: 2^{3-1}, where 2 is the number of levels; 3 is the number of variables; N = 23-1 is the number of experiments. The code designation of the half-replica: c; a; в; aвc. As can be seen from (2.23), the features of the design of the design are that the variable x_3 at the points of the design satisfies the so-called generating relation::

$$x_3 = x_1 x_2 \tag{2.24}$$

Using this equation, it is easy to construct (2.23) – first the FFE 2^2, and then the column vector x_3, which corresponds to (2.24).

2.3. Planning an experiment on composition–property diagrams

2.3.1. Simplex grid plans. In chemical technology, in particular in the technology of polymer composite materials, most of the objects under study belong to the class of complex experimental design systems, which are mixtures of q different components. The variables x_i $(i = 1,2,...,q)$ of such systems are the proportions (relative content) of the i-th components of the mixture and satisfy the following condition [96,113,114]:

$$\sum_{1 \le i \le q} x_i = 1, \ (x_i \ge 0) \tag{2.25}$$

The locus of points satisfying condition (2.25) is a (q-1)-dimensional regular simplex, which is a triangle for q=3, a tetrahedron for q=4, etc. Each point of such a simplex corresponds to a mixture of a certain composition, and, conversely, any combination of the relative contents of q components corresponds to a specific point of the simplex. Since when planning experiments and constructing composition–property diagrams one has to operate with the factor space in the form of simplexes, it is advisable to switch from ordinary Cartesian coordinates to a special simplex system, in which the relative contents of each component are plotted along the corresponding faces of the simplex [96,98]. At the vertices of the simplex each x_i = 1, and further - are determined by the lines (or surfaces) of the level parallel to the opposite side (or face) of the simplex. So, for example, for a three-component mixture, the simplex is an equilateral triangle x_1, x_2, x_3 (Fig. 2.6).

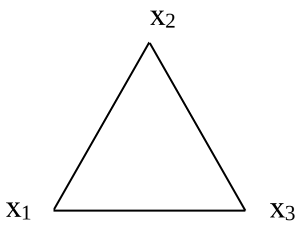


Рис. 2.6 – Симплексна система координат

The value of the variable x_1 at the vertex x_1 is equal to one, and on the side x_2x_3 it is zero.

The problem of constructing a mathematical model of composition–property can be solved by writing the desired function as a polynomial of degree n in canonical form:

$$\hat{y} = \sum_{1\le i\le q} \beta_i x_i + \sum_{m=2}^{n} \left\{ \sum_{1\le i\le j\le q} \beta_{ij}^{(m)} x_i x_j \left(x_i - x_j\right)^{m-2} \right\} + \sum_{m=3}^{n} \left\{ \sum_{1\le i_1\le i_2\le \ldots \le i_m\le q} \beta_s x_{i_1}^{s_1} x_{i_2}^{s_2} \ldots x_{i_m}^{s_m} \right\}$$

(2.26)

where: $s_1 + s_2 + \ldots + s_m = n$.

Polynomials of this form (so-called reduced polynomials) are obtained from ordinary polynomials of the corresponding degree taking into account the relation (2.25) and contain C_{q+n-1}^{n} coefficients. For example, a polynomial of the second degree, which in the general case is described by the equation:

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3 + b_{12} x_1 x_2 + b_{13} x_1 x_3 + b_{23} x_2 x_3 + b_{11} x_1^{2} + b_{22} x_1^{2} + b_{33} x_1^{2}$$

taking into account the ratio $x_1 + x_2 + x_3 = 1$ will take the form:

$$\hat{y} = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_{12} x_1 x_2 + \beta_{13} x_1 x_3 + \beta_{23} x_2 x_3 .$$

To estimate the coefficients of the reduced polynomial (2.26), plans were proposed that provide a uniform distribution of experimental points over a (q-1)-dimensional simplex. The points of such plans are the nodes of $\{q, n\}$-simplex grids, in which (n+1) equally spaced levels in the interval from 0 to 1 $\left(x_i = 0, \, {}^{1}\!/_{n}, {}^{2}\!/_{n}, \ldots, 1 \right)$ are used for each factor (component) and various combinations of them are taken. Thus, the number of such combinations C_{q+n-1}^{n} is equal to the number of coefficients in the reduced polynomial (2.26). The set of points $\left(x_{1u}, x_{2u}, \ldots, x_{qu}\right)$, $u = 1, 2, \ldots, N = C_{q+n-1}^{n}$, where

$x_{iu} = 0, \, {}^{1}\!/_{n}, \, {}^{2}\!/_{n}, \ldots, 1, \qquad \sum_{1 \le i \le q} x_{iu} = 1$ forms a saturated simplex-grid $\{q, n\}$- plan.

Examples of $\{q, n\}$-grids are shown in Fig. 2.7.

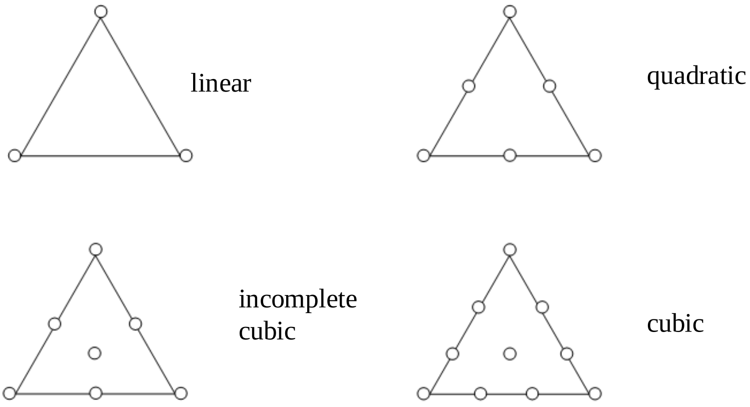


Fig. 2.7 – Types of $\{q, n\}$-grids

Each grid corresponds to a plan matrix:

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$ - for linear grid;

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1/2 \\ 0 & 1/2 & 1/2 \end{pmatrix}$$ - for quadratic grid;

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1/2 \\ 0 & 1/2 & 1/2 \\ 1/3 & 1/3 & 1/3 \end{pmatrix}$$ - for incomplete cubic grid;

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 2/3 & 1/3 & 0 \\ 1/3 & 2/3 & 0 \\ 2/3 & 0 & 1/3 \\ 1/3 & 0 & 2/3 \\ 0 & 2/3 & 1/3 \\ 0 & 1/3 & 2/3 \\ 1/3 & 1/3 & 1/3 \end{pmatrix}$$ - for cubic grid.

2.3.1.1. Planning with a preliminary transformation of the simplex sub-area. When solving *q*-component mixed problems, it is often necessary to investigate only a (*q*-1)-dimensional simplex subdomain of the full (*q*-1)-dimensional domain. The subdomain can be given by restrictions on the domain of change of all components, for example, $x_i \geq q_i$ (*i* = 1, 2, …, *q*). In this case, direct application of the methods described above is impossible, since the condition is violated, therefore, a transformation of the subdomain is first performed by transition to a new coordinate system $(z_1, z_2, ..., z_q)$ (Fig. 2.8) [96,113,114].

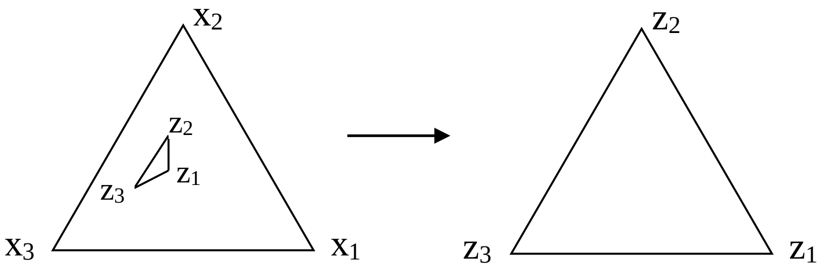


Fig. 2.8 – Transformation of a simplex sub-area

For the transformed sub-area, the equalities hold:

$$0 \le z_i \le 1,\ \mathrm{i} = 1, 2, \ldots, q;\ z_1^{(u)} + z_2^{(u)} + \ldots + z_q^{(u)} = 1, \quad (2.27)$$

where: u – any point of the sub-area.

The transformation dependence between the coordinate systems $(x_1, x_2, ..., x_q)$ and $(z_1, z_2, ..., z_q)$, which corresponds to condition (2.27), is given by the following matrix equation $X = AZ$, in expanded form:

$$\begin{vmatrix} x_1^{(u)} \\ x_2^{(u)} \\ \ldots \\ x_q^{(u)} \end{vmatrix} = \begin{vmatrix} x_1^{(1)} & x_1^{(2)} & \ldots & x_1^{(q)} \\ x_2^{(1)} & x_2^{(2)} & \ldots & x_2^{(q)} \\ \ldots & \ldots & \ldots & \ldots \\ x_q^{(1)} & x_q^{(2)} & & x_q^{(q)} \end{vmatrix} \cdot \begin{vmatrix} z_1^{(u)} \\ z_2^{(u)} \\ \ldots \\ z_q^{(u)} \end{vmatrix} \quad (2.28)$$

The elements of the matrix A are: – coordinates of the vertices of the simplex $x_i^{(u)}$ and $z_i^{(u)}$ – initial and new coordinates of the u-th transformed point. All plans that were used for the complete simplex can be constructed with respect to the new variables z, but the implementation of the experiment in such conditional plans is impossible. To conduct research, it is necessary to represent the experimental compositions of the systems in *x*-coordinates, that is, to make a transition according to the conditions (2.28).

2.3.2. Simplex-centroid plans. In simplex-lattice plans, experimental points are located mainly on the periphery of the simplex. As already noted, for a three-component composition, the simplex is an equilateral triangle, each vertex of which is an independent component of the mixture; the points contained on the edges of the triangle correspond to binary systems of pairs of ingredients, the points inside the simplex are the composition of the mixture from all three components. For a four-component system, the region of admissible variables has the form of a tetrahedron. Its faces correspond to simplices of ternary mixtures of three components, and the points inside are a mixture of four ingredients. In simplex-lattice plans, for constructing models of degree n, the experimental points are located in the simplex symmetrically, using for each component $x_i\,(i=\overline{1,q})$ q+1 equidistant levels ranging from 0 to 1: $x_i = 0;\ 1/n;2/n;\ldots;\ n/n=1$. All possible combinations of these levels are plans or simplex lattices. Such plans are considered fully saturated, i.e. the number of experiments in them is equal to the number of unknown coefficients of the corresponding model. In simplex-lattice plans, the experimental points are usually located on the periphery of the simplex. Some of these plans, for example, first- and second-order grids, do not contain any experimental points inside the studied region, i.e. those that correspond to the composition of all components. The polynomial used, adequately describing the results of experiments on the faces of the simplex, can give significant deviations for the central regions that correspond to mixtures of all q components of the studied system. Based on this, another arrangement of experimental points was proposed - simplex-centroid experimental design [96]. In simplex-lattice plans, the experiments are implemented in $N = 2^q - 1$ experimental points, q of which are points containing one non-zero component; C^2_q - points containing two non-zero components (binary mixtures); C^3_q – points containing three non-zero components (ternary mixtures), etc., and one point containing all the components of the mixture. The simplex-centroid plan contains points with coordinates (1,0,...,0); (1/2,1/2,0...,0); ...; (1/q, 1/q... 1/q), as well as all points that can be obtained by permuting their coordinates. Thus, the experimental points are placed at the vertices of the simplex, the midpoints of the sides, the centers of faces of different dimensions, and one point is in the center of the simplex.

Unlike simplex-grid plans, in which for a given q there is a set of $\{q, n\}$-grids (n = 1,2,...), there is a unique simplex-centroid plan for a fixed q. The approximating polynomial can be chosen as follows:

$$\hat{y} = \sum_{1 \le i \le q} \beta_i x_i + \sum_{1 \le i < j \le q}^{n} \beta_{ij} x_i x_j + \sum_{m=3}^{n} \beta_{ijk} x_i x_j x_k + \ldots + \beta_{12\ldots q} x_1 x_2 \ldots x_q. \tag{2.29}$$

It contains as many coefficients as there are points used in the simplex-centroid plan, i.e. these coefficients are uniquely determined by the responses at 2^q -1 points of such a plan..

2.4. Optimization of the composition of multicomponent systems

A significant number of experimental problems in chemistry and chemical technology are formulated as problems of determining the optimal process conditions, the optimal composition of the composition, etc. The research process is usually divided into separate stages. The information obtained after each stage determines the further strategy of the experiment. Thus, the possibility of optimal control of the experiment arises.

At the first stage of solving the optimization problem, it is necessary to clearly formulate it, as well as to make transformations and simplifications in order to bring it to a form convenient for further solution. The optimization problem of processes characterized by several responses is usually reduced to a single-criterion optimization problem with constraints in the form of equality or inequality. Depending on the form of the response surface and the nature of the constraints, it is proposed to use uncertain Lagrange multipliers, linear and nonlinear programming, ridge analysis, etc. for optimization [99]. The disadvantages of these methods of solving the optimization problem include the complexity of the calculation. In particular, provided that the response surface is described by second-order polynomials, solving the problem for a conditional extremum using uncertain Lagrange multipliers leads to the need to solve a system of nonlinear equations.

In the general case, the multi-criteria optimization problem is formulated as follows [100]:

$$\min_{\vec{x}} \{f_1(\vec{x}),\ \ f_2(\vec{x}),\ \ldots,\ f_k(\vec{x})\},\quad \vec{x} \in S \tag{2.30}$$

where: $f_i : R^n \to R$ − are $k\,(k \geq 2)$ of target functions

In this case, the target functions that are investigated for the maximum are transformed into functions that are investigated for the minimum:

$$\min(y) = -\max(y) \tag{2.31}$$

The solution vector $\vec{x} = (x_1, x_2, \ldots, x_n)^T$ belongs to the non-empty domain of definition *S*.

The solution of a multi-criteria optimization problem consists in finding a vector of variables that will satisfy the imposed constraints and optimize a vector function whose elements correspond to the objective functions. They are a mathematical description of the satisfaction criterion and, as a rule, can conflict with each other. Thus, the optimization problem is to find a solution at which the values of the objective functions would be acceptable for the formulation of the problem..

In the process of solving multi-criteria problems, a number of problems are solved [100]:

- the problem of normalization - individual criteria, as a rule, have different scales and units of measurement, which makes it impossible to directly compare them;

- the problem of taking into account the priority of criteria - they often have different significance, which is why it is necessary to find a mathematical definition of priority and the degree of its influence on solving the problem;

- the problem of determining the compromise region - arises when solving multi-dimensional nonlinear problems.

All decision-making problems are complex and multi-objective, since when choosing the best option, many different requirements must be taken into account, which may conflict with each other. Based on this, a multi-objective problem is often reduced to a single-objective problem, that is, one is formulated that includes one criterion, and one or more additional constraints are added to the original system of constraints.

There is no universal method for solving multi-criteria mathematical programming problems. The choice and correct use of any of the known methods is left to the decision-maker. The most common heuristic method for solving a particular multi-criteria problem is to reduce it to the solution of some scalar (single-criteria) problem, the objective function of which is most often a certain combination of existing criteria $f_1,\ f_2,\ ...,\ f_m$. This method is called scalarization of a multi-criteria problem. Depending on the method of combining several existing criteria into a single scalar one, one or another type of scalarization is obtained, which is chosen based on the essence of the problem being solved and some additional information about the advantages.

The simplest method of scalarization is based on the use of the so-called linear convolution of criteria:

$$F(x)=\sum_{i=1}^{m}\alpha_i\cdot f_i(x)\to \min$$

$$\alpha_i \geq 0,\ i = 1,\ \ldots,\ m,\ \ \sum_{i=1}^{m} \alpha_i = 1 \qquad (2.32)$$

In practice, the scalarization process begins with the selection of linear convolution coefficients, i.e. numbers $\alpha_i,\ i = 1, \ldots, m$. These numbers are interpreted as “importance coefficients” of the corresponding criteria, the more important of which is assigned a larger coefficient in the linear convolution of criteria, and the less important one is assigned a lower one. This method is convenient to use; it allows you to preserve the linearity of the output functions, i.e. in the case when the initial criteria are linear, the final criterion will also be linear.

Models describing a single-criteria problem are much simpler and can be solved by one of the known methods and used to optimize multicomponent systems. In order to determine the optimal composition of the mixture, it is necessary to solve the so-called conditional optimization problem, which is associated with optimization under variable constraints. These constraints reduce the size of the region in which the optimum is located. The optimization process becomes more complicated, since in the presence of constraints it is impossible to use the applied optimality conditions. In this case, even the basic conditions according to which the optimum should be achieved at a stationary point may be violated.

To move from a conditional optimization problem with constraints to an unconstrained problem, there are a number of methods: the method of indefinite Lagrange multipliers, the method of penalty functions, the method of barrier functions, etc. If the method of penalty functions is used, it is necessary that it “penalizes” the function *Z* for violating the constraints (i.e., increases its value). In this case, the minimum of the function *Z* will be inside the constraint region. There may be several penalty functions $P(x)$ that satisfy this condition. The minimization problem consists in minimizing the function $Z = f(x)$ under the constraints $c_j(x) > 0$, $j = 1, 2, ..., m$, then the function $P(x)$ takes the form:

$$P(x) = r \cdot \sum_{j=1}^{m} \frac{1}{c_j(x)}, \qquad (2.33)$$

where: r – quite a small value

By applying one of the above methods, we obtain an unconditional optimization problem, which is formulated as follows: find the minimum of the function *f(x)*, where $x \in R^n$ in the absence of restrictions on *x*, and *f(x)* is a scalar objective function, continuously differentiable [100,101].

When solving these problems, the researcher must take into account the following factors:

- the nature of the objective function of the problem being solved - single or multi-extreme;

- the possibility of obtaining information about the derivatives of the objective function during the optimization process;

- the presence of different approaches to organizing an iterative procedure for finding the optimum (methods based on the iterative movement of variables in a direction determined by one or another method).

Several methods can be used to perform unconditional optimization: direct search, first-order, second-order (Newton methods), random search, gradient, etc. In direct search methods for the minimum of the objective function (or zero-order methods), information is used only about the value of the function. Many of them do not have sufficient theoretical justification and are built on the basis of heuristic considerations. Random search methods implement an iterative process of moving optimization variables in space using random directions. The advantage of these methods is a large range of possible directions of movement. The gradient method with step splitting is most often used, since it is quite simple and is characterized by good convergence.

Thus, to optimize the content of ingredients in a multicomponent mixture, it is necessary to conduct multi-criteria optimization of the system taking into account several conflicting objective functions. To do this, the multi-criteria problem is reduced to a single-criteria problem, the conditional optimization problem is transformed into an unconditional optimization problem and it is solved by one of the specified methods.

Today, one of the most widely used methods for solving the problem of optimizing processes with a large number of responses is the general optimization criterion proposed by Harrington - the so-called generalized desirability function D [99]. To find it, the found response values are converted into a dimensionless desirability scale d. The construction of a desirability scale, which establishes the relationship between the response value y and the corresponding value d (partial desirability function), is fundamentally subjective, that is, one that reflects the researcher's attitude to individual responses.

It is convenient to create a desirability scale using the method of quantitative assessments with an interval of desirability values from zero to one, but other options are also possible. The value d = 0 (or D = 0) corresponds to an absolutely unsuitable response value, and d = 1 (D = 1) is the best response value, and its further improvement is either impossible or of no interest. Intermediate desirability values and the numerical assessments corresponding to them are given in Table 2.3.

Table 2.3 – Base estimations of the desirability scale

Quantitative assessment on a desirability scale	Desirability of response values
0,80 ÷ 1,00	very good
0,63 ÷ 0,80	good
0,37 ÷ 0,63	satisfactorily
0,20 ÷ 0,37	bad
0,00 ÷ 0,20	very bad

This choice of numerical estimates is explained by the convenience of calculations, since $d = 0.63 \approx 1 - 1/e$ and $d = 0.37 \approx 1/e$. The d scale constructed in accordance with Table 2.3 is a dimensionless scale, with the help of which any response can be transformed in such a way that it is interpreted in terms of usefulness or desirability for any specific application.

The simplest type of transformation is one in which there is an upper and/or lower specification limit, and these limits are unique and do not allow changes in the quality criterion. Outside these limits, the value d = 0.0, and between them d = 1. The partial desirability function under a one-sided constraint has the following form:

$$d = \begin{cases} 0, & y < y_{\min} \\ 1, & y \geq y_{\min} \end{cases} \tag{2.34}$$

Similarly, a partial desirability function is obtained if the specification imposes a constraint from above, and under a two-sided constraint, the desirability function takes the form:

$$d = \begin{cases} 0, & y < y_{\min}\ ma\ y > y_{\max} \\ 1, & y_{\min} \leq y \leq y_{\max} \end{cases} \tag{2.35}$$

It is always desirable that the response value is not only between the specification limits, but also at a certain distance from them in order to prevent possible random fluctuations. In addition, it is sometimes difficult to determine the exact limiting line between acceptable and unacceptable product quality indicators. In the general case, the conversion of y to d is carried out according to a more complex law. For a two-sided restriction of the form $y_{\min} \leq y \leq y_{\max}$, the conversion of the measured response y to the scale d is performed using the expression:

$$d = \exp\left(-\left(|y'|\right)^n\right) \tag{2.36}$$

where: n − positive number ($0 < n < \infty$), not necessarily an integer;

$$y' = \frac{2y - (y_{\max} + y_{\min})}{y_{\max} - y_{\min}} \tag{2.37}$$

In this case, the exponent of the power n can be calculated by assigning a value of d to some value of y (preferably in the interval $0{,}6 < d < 0{,}9$) using the formula:

$$n = \frac{\ln(\ln(1/d))}{\ln(|y'|)}. \qquad (2.38)$$

For one-sided constraints of the form $y \le y_{\max}$ or $y \ge y_{\min}$, more convenient form of converting y to d is another exponential dependence:

$$d = \exp[-\exp(-y')] \qquad (2.39)$$

where: y' – dimensionless value of the output variable, which is determined from the expression

$$y' = b_0 + b_1 y \qquad (2.40)$$

The coefficients b_0 and b_1 can be calculated by specifying the corresponding desirability values d for two values of the property y, preferably in the interval $0{,}2 < d < 0{,}8$. In practice, one-sided specification is most common.

Having several responses converted into a scale d, it is possible to combine from these different d some generalized desirability index D by means of arithmetic operations. In this case, if one of the responses is absolutely unsatisfactory, the generalized desirability function D should be equal to zero regardless of the levels of the different responses. A mathematical expression that meets these requirements is the geometric mean of the partial desirability functions, i.e.:

$$D = \sqrt[k]{d_1 d_2 \ldots d_k} \qquad (2.41)$$

where: k – number of optimization criteria

Provided that some one $d_i = 0$, then the corresponding D is also zero. Moreover, the generalized desirability function is most strongly influenced by the smallest values of d_i. At the same time, $D = 1$ only when all partial desirabilities $d_i = 1$

(i = 1, 2, ... k). It is also important that expression (2.41) allows us to apply to partial desirabilities and the generalized indicator a single method of specifying the basic desirability scale estimates given in Table 2.3, if $d_i = d_1 = d_2 =\dots d_k = 0{,}37$, then $D = 0.37$, etc. With the generalized desirability function, all computational operations can be performed, as with any system response, and D can be used as an optimization criterion in the study and optimization of the process. It should be borne in mind that the set of possible values of D is limited to $D \leq 1$. The most effective application of the generalized desirability function turned out to be in the development of recipes in the technology of obtaining new polymer materials.

2.5. Software for planning experiments, developing mathematical models, and optimizing the composition of multicomponent systems

2.5.1. Software for constructing an experimental design for ternary mixture systems. To build a work plan for conducting research on various three-component mixture systems of all possible ratios of components, we have developed software (software) [102,103]. The program allows you to solve one of the important problems that may arise during planning, namely, the uneven content of the mixture components (the concentration of one or two of them is less than the content of the others by an order of magnitude or more). The software was created in the *Builder* environment in the C++ language [104-106].

In order to optimize the composition of compositions that are mixtures of q different components, the simplex-grid method is used, since it is the most suitable for studying mixtures. The variables $x_i \quad (i = 1,2,\ldots,q)$ of such systems are the proportions (relative to the content) of the *i*-th components of the mixture and satisfy the condition (2.25).

When developing an experimental design, the factor space is operated in the form of simplexes, therefore, the created software provides for a transition from ordinary Cartesian coordinates to a special simplex system. The points that determine the relative content of each component are laid out along the corresponding faces of the simplex. At the vertices of the simplex, each $x_i = 1$, and then they are determined by the lines (or surfaces) of the level parallel to the opposite side (or face) of the simplex. For a three-component mixture on the plane, the simplex has the form of a triangle with vertices $x_1x_2x_3$ (Fig. 2.9).

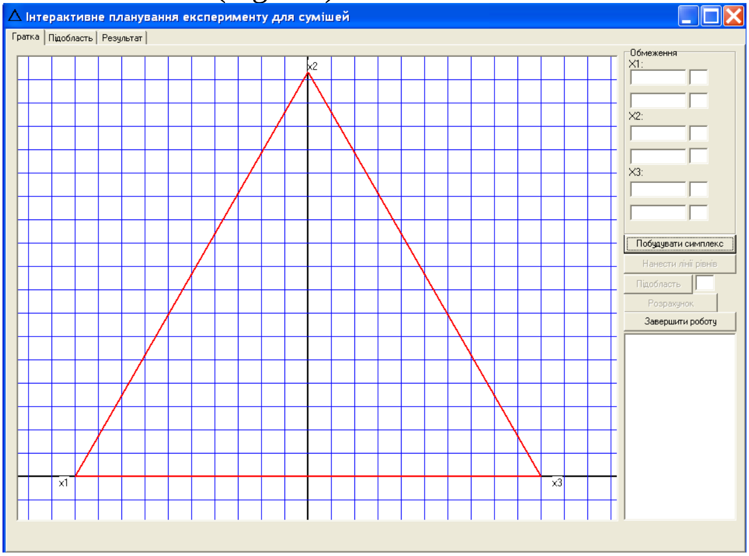


Fig. 2.9 – Simplex area for experiment planning

Each vertex of the simplex is an independent component of the mixture; the points forming the edges of the triangle correspond to binary systems of pairs of ingredients, the points in the middle of the simplex are a mixture of a mixture of all three components. The value of the content of the first component (x_1) at the vertex $x_1$is equal to one, and on the opposite side x_2x_3 is zero.

In *q*-component mixtures, the content of ingredients can vary from 0 to 1 or within this interval, which is determined by the requirements for the properties of the created compositions. In this case, it is necessary to investigate only the (*q*-1)-dimensional simplex subdomain of the full (*q*-1)-dimensional area. The subdomain is given by restrictions on the content of all components. The developed program allows you to automatically obtain a factor space for conducting an experiment for all possible combinations of the composition of the compositions, including those with uneven content of components. A limited area of irregular shape, which is a factor space for conducting an experiment, is obtained by introducing restrictions on the concentration of ingredients. For this purpose, the program provides an option to enter restrictions on the content of the components of the mixture (Fig. 2.10). The limited region of irregular shape, i.e. the factor space of the experiment for compositions with comparable ingredient contents, is shown in Fig. 2.11.

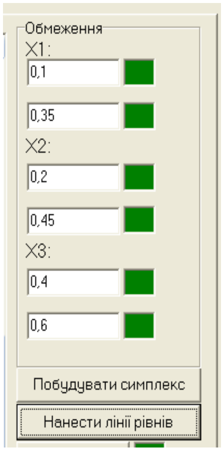
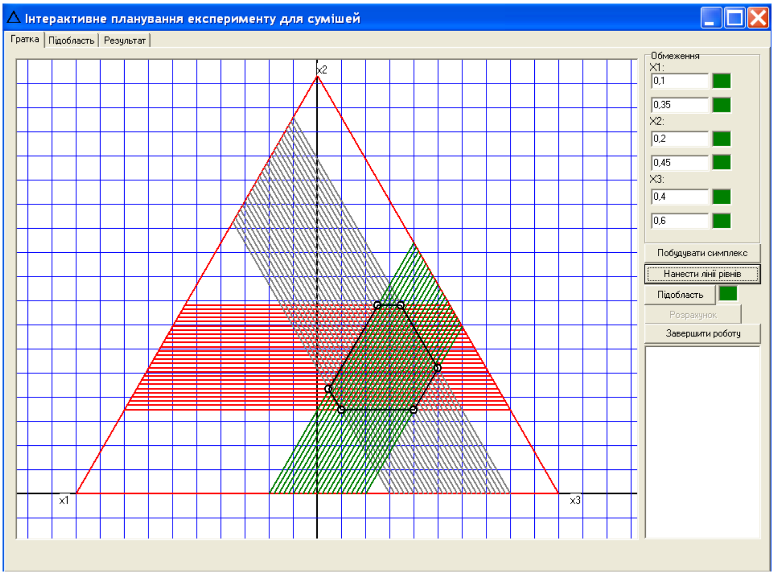

Обмеження
X1:
0,1
0,35
X2:
0,2
0,45
X3:
0,4
0,6

Fig. 2.10 – Section of the program working form with restrictions on the content of mixture components



Обмеження
X1:
0,1
0,35
X2:
0,2
0,45
X3:
0,4
0,6

Fig. 2.11 – Factor space for conducting an experiment

Planning experiments using the simplex-lattice method is carried out in a subregion "similar" to the original simplex, i.e. in the polygon it is necessary to select a triangular subregion. This triangle, firstly, must lie completely inside the "cut out" area and, secondly, most fully cover it. The program allows you to construct a region in the form of a triangle inside the found subregion. The user can interactively select a triangular region inside the found polygon, for which he must first click the "Subregion" button, and an enlarged subregion for conducting the experiment appears on the monitor screen (Fig. 2.12).

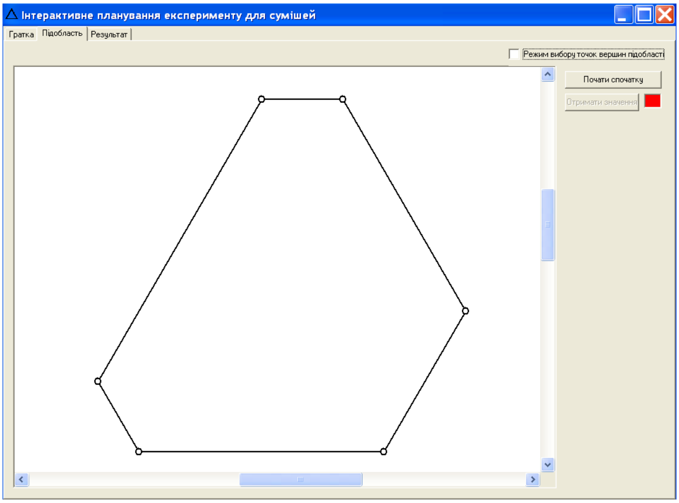
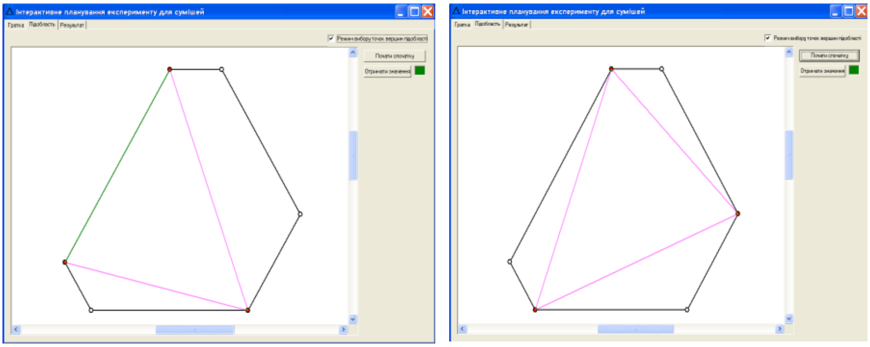

Інтерактивне планування експерименту для сумішей
Гратка
Підобласть
Результат
Режим вибору точок вершин підобласті
Почати спочатку
Отримати значення

Fig. 2.12 – The enlarged subregion obtained under the given constraints

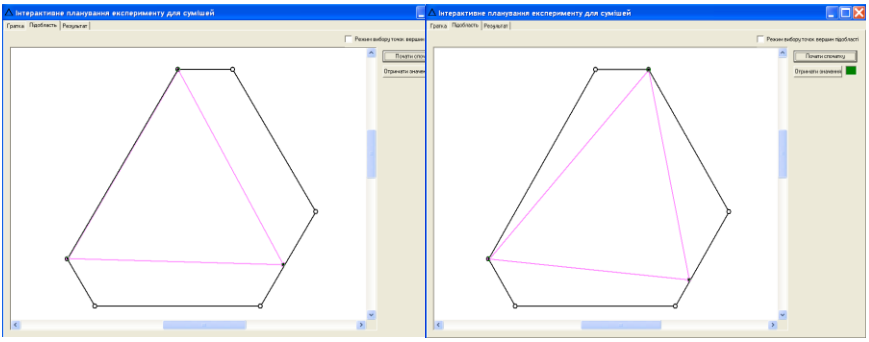
The triangular subregion can be selected in various ways (Fig. 2.13 a, b).


Інтерактивне планування експерименту для сумішей
Гратка
Підобласть
Результат
☑ Режим вибору точок вершин підобласті
Почати спочатку
Отримати значення

a) b)

Fig. 2.13 – Different options for choosing a triangular subregion

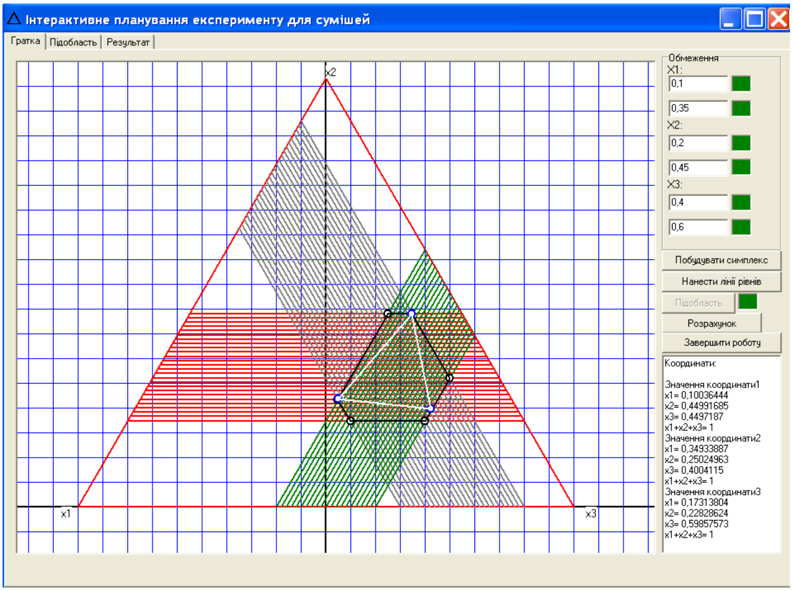
The determination of the vertex points of the triangle can be carried out in two modes: by selecting the option "subregion vertex point selection mode", or by canceling it. Fig. 2.14 (a, b) shows the subregions with the option canceled.


Інтерактивне планування експерименту для сумішей
Гратка
Підобласть
Результат
☐ Режим вибору точок вершин підобласті
Почати спочатку
Отримати значення

a) b)

Fig. 2.14 – Different options for selecting a triangular subregion with the "Subregion vertex selection mode" option unchecked

The researcher selects the most appropriate area of the triangular shape based on his empirical experience. The next option is to select a subarea that most fully covers the possible combinations of the ratios of the components in the composition. Then, by clicking the “Get values” button, the user sees on the screen that the selected triangle is inside the complete simplex, and in the window on the right - the coordinates of its vertices in the simplex system (Fig. 2.15).



```
X1:
0,1
0,35
X2:
0,2
0,45
X3:
0,4
0,6
```

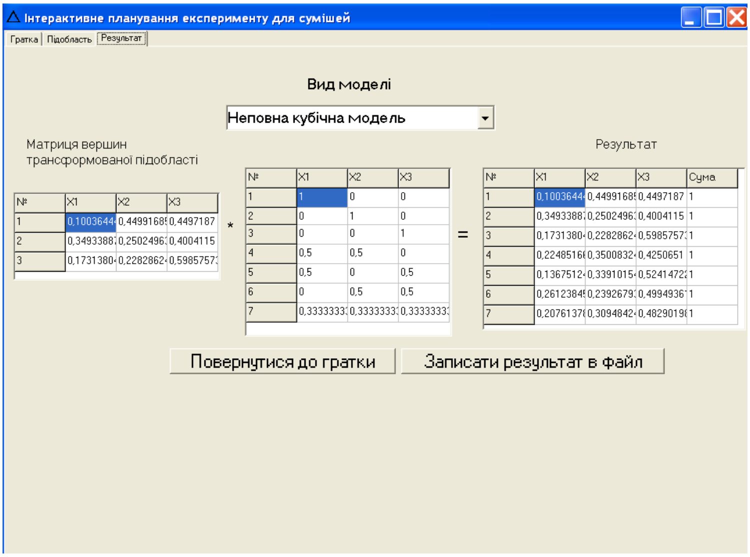
```
Координати:

Значення координати1
x1= 0,10036444
x2= 0,44991685
x3= 0,4497187
x1+x2+x3= 1
Значення координати2
x1= 0,34933887
x2= 0,25024963
x3= 0,4004115
x1+x2+x3= 1
Значення координати3
x1= 0,17313804
x2= 0,22828624
x3= 0,59857573
x1+x2+x3= 1
```

Fig. 2.15 – Constructed subdomain inside the full simplex

The properties of the system can be described by different models taking into account specific requirements for them - first of all, this is adequacy and simplicity. The created software provides the possibility of using three types of models: quadratic, incomplete cubic and cubic. The calculation of the coefficients of the equations is carried out according to the matrix relation $X = AZ$, where the matrix elements: A - coordinates of the vertices of the simplex, X and Z - matrices of plans: for the desired working and for the complete simplex, respectively.

By default, the program calculates the dependence of the output variables on the content of the mixture components according to the incomplete cubic model. By clicking the "Calculation" button, the user receives an experiment plan on the screen (Fig. 2.16, table "Result").



Матриця вершин трансформованої підобласті

№	X1	X2	X3
1	0,1003644	0,4499168	0,4497187
2	0,3493388	0,2502496	0,4004115
3	0,1731380	0,2282862	0,5985757

*

№	X1	X2	X3
1	1	0	0
2	0	1	0
3	0	0	1
4	0,5	0,5	0
5	0,5	0	0,5
6	0	0,5	0,5
7	0,3333333	0,3333333	0,3333333

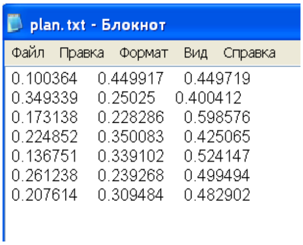
=

Результат

№	X1	X2	X3	Сума
1	0,1003644	0,4499168	0,4497187	1
2	0,3493388	0,2502496	0,4004115	1
3	0,1731380	0,2282862	0,5985757	1
4	0,2248516	0,3500832	0,4250651	1
5	0,1367512	0,3391015	0,5241472	1
6	0,2612384	0,2392679	0,4994936	1
7	0,2076137	0,3094842	0,4829019	1

Fig. 2.16 – Plan of experiment for the incomplete cubic model

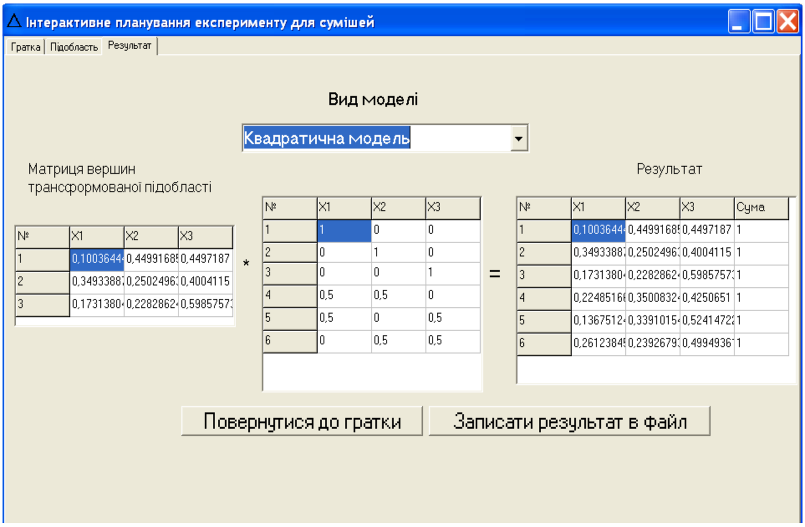
For convenience, the experiment plan can be saved to a file. The created file stores the experiment plan (Fig. 2.17).



```
0.100364   0.449917   0.449719
0.349339   0.25025   0.400412
0.173138   0.228286   0.598576
0.224852   0.350083   0.425065
0.136751   0.339102   0.524147
0.261238   0.239268   0.499494
0.207614   0.309484   0.482902
```

Fig. 2.17 – The result of the program - a saved experiment plan

As already mentioned, the program allows you to build experimental plans also using quadratic and cubic models. To do this, you need to return to the form with calculations (Fig. 2.16) and select the appropriate options on the form (Fig. 2.18, 2.19).


Інтерактивне планування експерименту для сумішей
Гратка
Підобласть
Результат
Повернутися до гратки
Записати результат в файл

Вид моделі: Квадратична модель

Матриця вершин трансформованої підобласті

№	X1	X2	X3
1	0,1003644	0,4499168	0,4497187
2	0,3493388	0,2502496	0,4004115
3	0,1731380	0,2282862	0,5985757

*

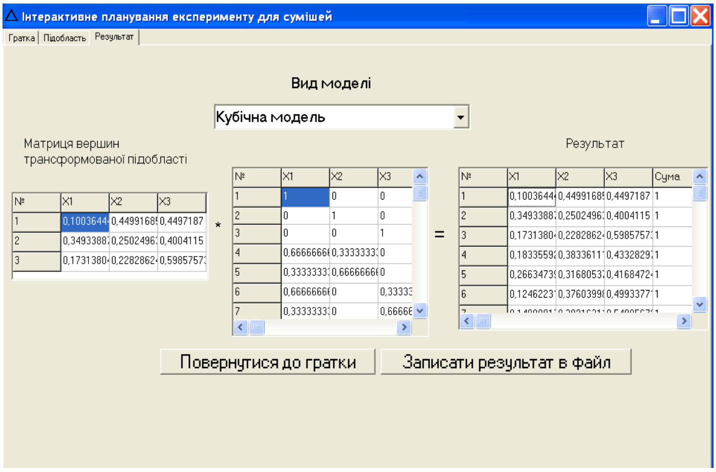
№	X1	X2	X3
1	1	0	0
2	0	1	0
3	0	0	1
4	0,5	0,5	0
5	0,5	0	0,5
6	0	0,5	0,5

=

Результат

№	X1	X2	X3	Сума
1	0,1003644	0,4499168	0,4497187	1
2	0,3493388	0,2502496	0,4004115	1
3	0,1731380	0,2282862	0,5985757	1
4	0,2248516	0,3500832	0,4250651	1
5	0,1367512	0,3391015	0,5241472	1
6	0,2612384	0,2392679	0,4994936	1

Fig. 2.18 – Plan of experiment for the quadratic model


Інтерактивне планування експерименту для сумішей
Гратка
Підобласть
Результат
Вид моделі
Кубічна модель
Повернутися до гратки
Записати результат в файл

Матриця вершин трансформованої підобласті

№	X1	X2	X3
1	0,1003644	0,4499168	0,4497187
2	0,3493388	0,2502496	0,4004115
3	0,1731380	0,2282862	0,5985757

*

№	X1	X2	X3
1	1	0	0
2	0	1	0
3	0	0	1
4	0,6666666	0,3333333	0
5	0,3333333	0,6666666	0
6	0,6666666	0	0,33333
7	0,3333333	0	0,66666

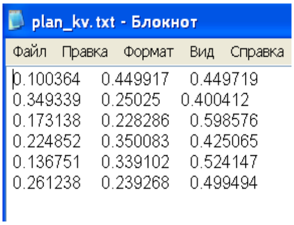
=

Результат

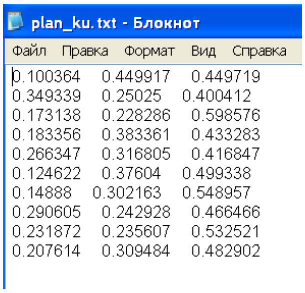
№	X1	X2	X3	Сума
1	0,1003644	0,4499168	0,4497187	1
2	0,3493388	0,2502496	0,4004115	1
3	0,1731380	0,2282862	0,5985757	1
4	0,1833559	0,3833611	0,4332829	1
5	0,2663473	0,3168053	0,4168472	1
6	0,1246223	0,3760399	0,4993377	1
7	[illegible]	[illegible]	[illegible]	1

Fig. 2.19 – Plan of experiment for the cubic model

Experiment plans for these models can also be written to a files (Fig. 2.20).



```
0.100364   0.449917   0.449719
0.349339   0.25025   0.400412
0.173138   0.228286   0.598576
0.224852   0.350083   0.425065
0.136751   0.339102   0.524147
0.261238   0.239268   0.499494
```



```
0.100364   0.449917   0.449719
0.349339   0.25025   0.400412
0.173138   0.228286   0.598576
0.183356   0.383361   0.433283
0.266347   0.316805   0.416847
0.124622   0.37604   0.499338
0.14888   0.302163   0.548957
0.290605   0.242928   0.466466
0.231872   0.235607   0.532521
0.207614   0.309484   0.482902
```

Fig. 2.20 – The result of the program - saved experiment plans

2.5.1.1. Experimental plan for ternary systems with incommensurable component contents. The developed software makes it possible to plan an experiment, in particular, for three-component compositions, in which one of the components is added in a much smaller amount compared to the other two ingredients (the difference can be 50÷1000 times). Such a task arises, in particular, when planning experimental studies on the influence of substances in the nanoscale on the properties of polymer compositions. When interactively planning an experiment in systems of such a composition, the factor space has a very small size (contracted into a strip or point), which makes it necessary to make an uneven increase in the planning area with the obligatory preservation of the correspondence of mathematical coordinates. The created program allows you to perform this operation [103].

The user begins work by entering restrictions on the content of the mixture components and clicks the "apply level lines" button (Fig. 2.21)

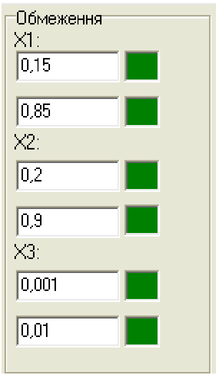
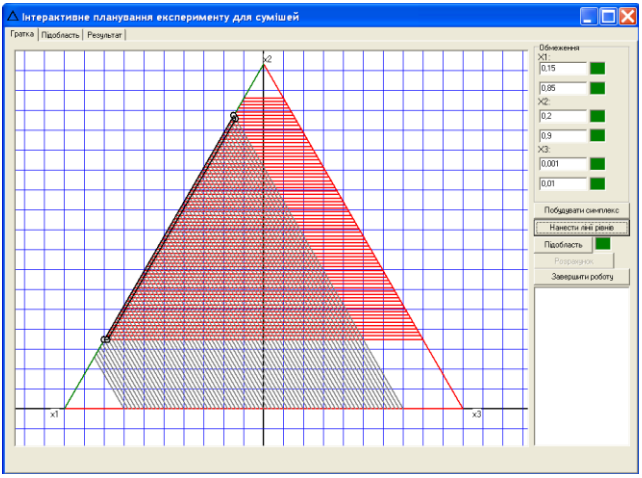

Обмеження
X1:
0,15 🟩
0,85 🟩
X2:
0,2 🟩
0,9 🟩
X3:
0,001 🟩
0,01 🟩

Fig. 2.21 – Section of the program working form with restrictions on the content of mixture components

The factor space for conducting the experiment appears on the monitor in the form of a strip (Fig. 2.22).



Обмеження
X1: 0,15 🟩 0,85 🟩
X2: 0,2 🟩 0,9 🟩
X3: 0,001 🟩 0,01 🟩

Fig. 2.22 – Factor space contracted into a strip

Further, when clicking the "subarea" button, in the previous version of the program, which did not provide for uneven enlargement of the subarea, the user would receive the following result (Fig. 2.23).

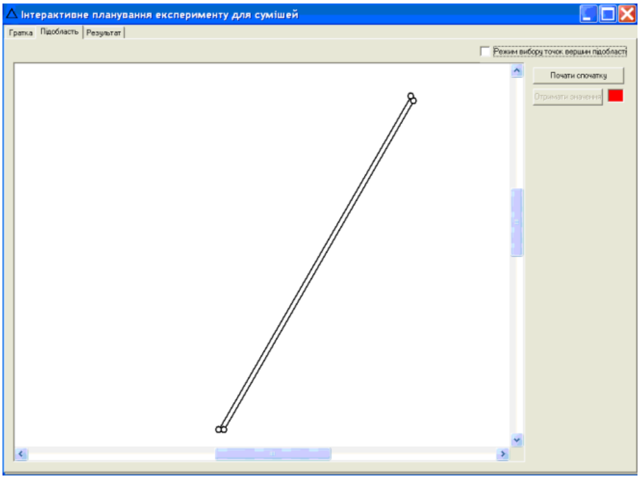

Інтерактивне планування експерименту для сумішей
Гратка
Підобласть
Результат
Режим вибору точок вершин підобласті
Почати спочатку
Отримати значення

Fig. 2.23 – Increased factor space compared to the previous version of the program

In this case, interactive planning is impossible due to the small size of the factor space.

The uneven increase in the planning area provided for in the developed program is achieved automatically by the algorithm built into it by introducing a coefficient that changes the size of the subarea in a certain direction to the appropriate size while maintaining the correspondence of mathematical coordinates (Fig. 2.24).

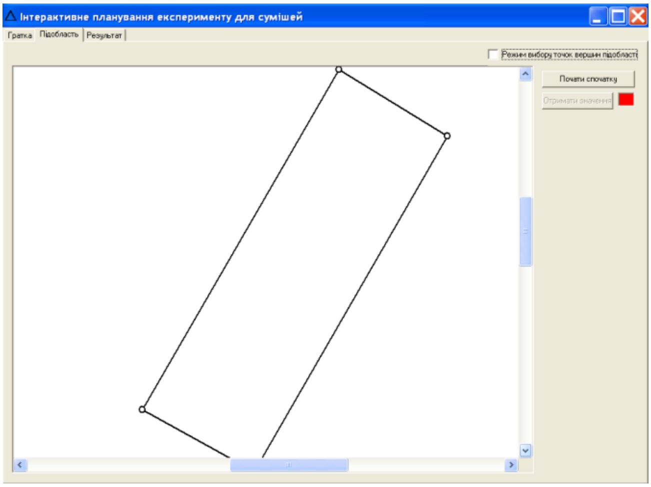

Інтерактивне планування експерименту для сумішей
Гратка
Підобласть
Результат
Режим вибору точок вершин підобласті
Почати спочатку
Отримати значення

Fig. 2.24 – Unevenly enlarged factor space

The user interactively selects three points within the subdomain and the factor space of the experiment appears on the monitor screen in the form of a triangle (Fig. 2.25).

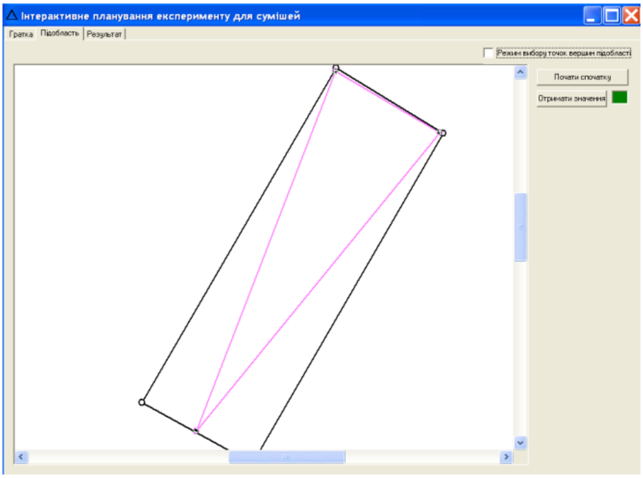
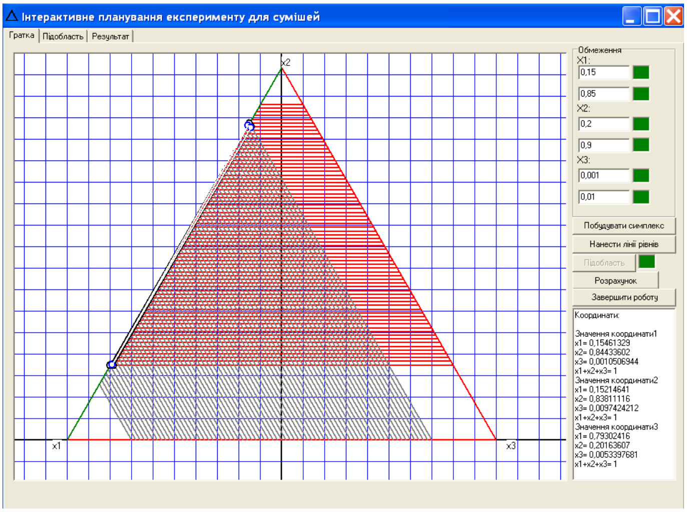

Інтерактивне планування експерименту для сумішей
Гратка
Підобласть
Результат
Режим вибору точок вершин підобласті
Почати спочатку
Отримати значення

Fig. 2.25 – Experiment planning area

By clicking the "Get values" button, the researcher receives on the screen the area of the experiment, which is located in the complete simplex, on which the triangle is not visually visible (Fig. 2.26).



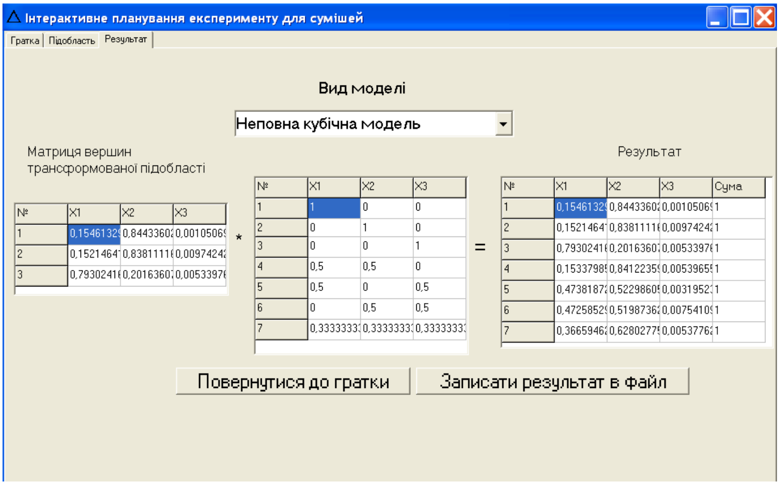
Обмеження
X1:
0,15 🟩
0,85 🟩
X2:
0,2 🟩
0,9 🟩
X3:
0,001 🟩
0,01 🟩

```
Координати:

Значення координати1
x1= 0,15461329
x2= 0,84433602
x3= 0,0010506944
x1+x2+x3= 1
Значення координати2
x1= 0,15214641
x2= 0,83811116
x3= 0,0097424212
x1+x2+x3= 1
Значення координати3
x1= 0,79302416
x2= 0,20163607
x3= 0,0053397681
x1+x2+x3= 1
```

Fig. 2.26 – The experimental area is located in the full simplex

The program by default outputs a plan matrix for an incomplete cubic model (Fig. 2.27).



Матриця вершин трансформованої підобласті

№	X1	X2	X3
1	0,15461329	0,84433602	0,00105069
2	0,15214647	0,83811116	0,00974242
3	0,79302416	0,20163607	0,00533976

*

№	X1	X2	X3
1	1	0	0
2	0	1	0
3	0	0	1
4	0,5	0,5	0
5	0,5	0	0,5
6	0	0,5	0,5
7	0,33333333	0,33333333	0,33333333

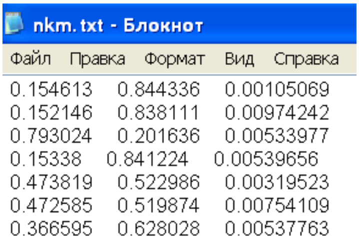
=

Результат

№	X1	X2	X3	Сума
1	0,15461329	0,84433602	0,00105069	1
2	0,15214647	0,83811116	0,00974242	1
3	0,79302416	0,20163607	0,00533976	1
4	0,15337985	0,84122359	0,00539655	1
5	0,47381872	0,52298605	0,00319523	1
6	0,47258529	0,51987362	0,00754109	1
7	0,36659462	0,62802775	0,00537762	1

Fig. 2.27 – Plan of experiment for an incomplete cubic model has been constructed

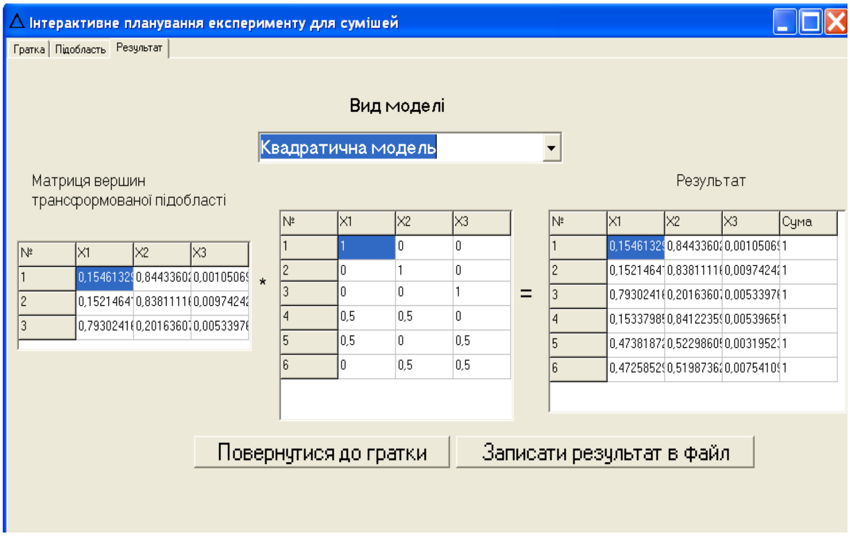
The experiment plan can be saved in a file (Fig. 2.28).



```
0.154613   0.844336   0.00105069
0.152146   0.838111   0.00974242
0.793024   0.201636   0.00533977
0.15338   0.841224   0.00539656
0.473819   0.522986   0.00319523
0.472585   0.519874   0.00754109
0.366595   0.628028   0.00537763
```

Fig. 2.28 – Saved experiment plan

Experiment plans for quadratic and cubic models are constructed similarly (Fig. 2.29, 2.30) and saved in files (Fig. 2.31).


Інтерактивне планування експерименту для сумішей
Гратка
Підобласть
Результат
Повернутися до гратки
Записати результат в файл

Вид моделі

Квадратична модель

Матриця вершин трансформованої підобласті

№	X1	X2	X3
1	0,1546132	0,8443360	0,0010506
2	0,1521464	0,8381111	0,0097424
3	0,7930241	0,2016360	0,0053397

*

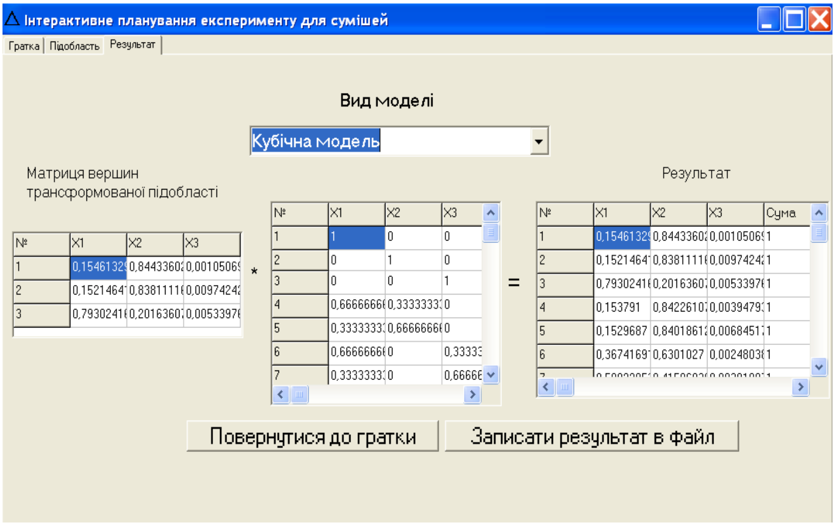
№	X1	X2	X3
1	1	0	0
2	0	1	0
3	0	0	1
4	0,5	0,5	0
5	0,5	0	0,5
6	0	0,5	0,5

=

Результат

№	X1	X2	X3	Сума
1	0,1546132	0,8443360	0,0010506	1
2	0,1521464	0,8381111	0,0097424	1
3	0,7930241	0,2016360	0,0053397	1
4	0,1533798	0,8412235	0,0053965	1
5	0,4738187	0,5229860	0,0031952	1
6	0,4725852	0,5198736	0,0075410	1

Fig. 2.29 – Plan of experiment for a quadratic model


Інтерактивне планування експерименту для сумішей
Гратка
Підобласть
Результат
Повернутися до гратки
Записати результат в файл

Вид моделі

Кубічна модель

Матриця вершин трансформованої підобласті

№	X1	X2	X3
1	0,1546132	0,8443360	0,0010506
2	0,1521464	0,8381111	0,0097424
3	0,7930241	0,2016360	0,0053397

*

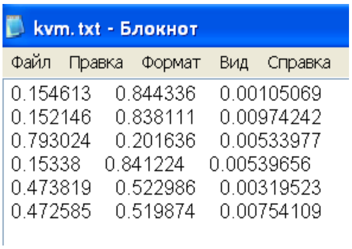
№	X1	X2	X3
1	1	0	0
2	0	1	0
3	0	0	1
4	0,6666666	0,3333333	0
5	0,3333333	0,6666666	0
6	0,6666666	0	0,33333
7	0,3333333	0	0,66666

=

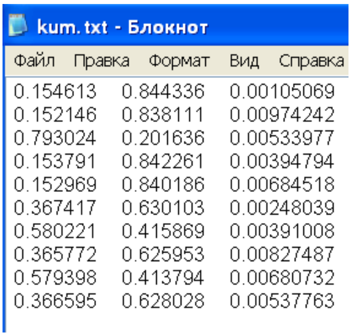
Результат

№	X1	X2	X3	Сума
1	0,1546132	0,8443360	0,0010506	1
2	0,1521464	0,8381111	0,0097424	1
3	0,7930241	0,2016360	0,0053397	1
4	0,153791	0,8422610	0,0039479	1
5	0,1529687	0,8401861	0,0068451	1
6	0,3674169	0,6301027	0,0024803	1
7	[illegible]	[illegible]	[illegible]	1

Fig. 2.30 – Plan of experiment for a cubic model



```
0.154613   0.844336   0.00105069
0.152146   0.838111   0.00974242
0.793024   0.201636   0.00533977
0.15338   0.841224   0.00539656
0.473819   0.522986   0.00319523
0.472585   0.519874   0.00754109
```



```
0.154613   0.844336   0.00105069
0.152146   0.838111   0.00974242
0.793024   0.201636   0.00533977
0.153791   0.842261   0.00394794
0.152969   0.840186   0.00684518
0.367417   0.630103   0.00248039
0.580221   0.415869   0.00391008
0.365772   0.625953   0.00827487
0.579398   0.413794   0.00680732
0.366595   0.628028   0.00537763
```

a) b)

Fig. 2.31 – Saved plans of experiments for quadratic (a) and cubic (b) models

Thus, the developed software has a user-friendly interface and does not require additional knowledge of computer technology. The software can be used for automated interactive planning of an experiment in the process of conducting scientific research on mixture systems (polymers modified with inorganic and organic additives, nanofilled polymer dispersions, fiberglass, polymer concretes, etc.) in research laboratories and at enterprises of various industries. The program allows you to plan an experiment for all possible ratios of ingredients in three-component compositions, including solving one of the important problems that may arise during planning, namely: the uneven content of mixture components, in which the concentration of one or two of them is less than the number of others by at least an order of magnitude.

Thus, in C++ software has been developed that allows interactively building an experiment plan for various ternary mixtures using three types of models of dependence of the output parameters on the content of the components - incomplete cubic, cubic and quadratic. The program allows solving the problem of experiment planning for compositions in which the content of one or two components differs from the others by hundreds and thousands of times. This is achieved thanks to the algorithm built into the software, which ensures an uneven increase in the area of the factor space with the obligatory preservation of the correspondence of mathematical coordinates.

2.5.2. Software for constructing an experimental design for four-component mixture systems. In order to automate the process of experimental research on optimizing the composition of multicomponent mixtures, we developed software for building a work plan of experiments for all combinations of ingredient ratios in four-component systems [31]. The program was created in the Delphi environment [107-109]. The software was developed on the basis of the simplex-lattice method, while the ratio of ingredients in the compositions satisfies condition (2.25), which determines the region of admissible variables, the so-called simplexes [96]. As already noted, for a four-component system it has the form of a tetrahedron, the faces of which correspond to the simplexes of three-component mixtures, and the points inside are four-component ones. To build models in simplex-lattice plans, experimental points are symmetrically located mainly on the periphery of the simplex. To take into account the results of experiments inside the simplex when developing the software, we used simplex-centroid plans, which contain points with coordinates: $(1;0;...;0)$; $(1/2;1/2;0;...;0)$; …; $(1/q;1/q;...;1/q)$, as well as all points that can be obtained by permuting these coordinates [150]. In this case, the experimental points are located at the vertices of the simplex, the midpoints of the sides, the centers of the faces of different dimensions, one point is in the center of the simplex. In this case, from the 2^{q-1} experimental data, q points have one non-zero component; C_q^2; C_q^3; C_q^4 □ - two, three and four non-zero ones, respectively, and one point contains all the components.

To construct the experimental plan, we took a conditional mixture of two polymers (*A*, *B*) and two modifying additives (*c*, *d*), the relative concentrations of which were x_1, x_2, x_3, x_4, respectively. Two-sided restrictions were imposed on the content of individual ingredients of the system:

$$0 \le a_i \le x_i \le b_i \le 1, \quad i = \overline{1,q} \tag{2.45}$$

where: a_i, b_i – upper and lower limits of each component, which must not be equal to each other.

The development of plan of experiment that meets some optimality criterion begins with determining the coordinates of candidate points, namely: polygon vertices, edge midpoints, face centers, and the common centroid. For this purpose, the software uses the McLean–Anderson method [99], according to which all possible combinations of the lower and upper levels a_i and b_i are selected and for each component, skipping the content of one of them. For the four-component mixture under study, one of the options may be $a_1;b_2;–;b_4$, while the total number of combinations (at q=4) is 32. In the created software, to select all possible combinations of ingredients in the mixture, the researcher performs the corresponding procedure: procedure convert (a, b: vector1; var x1, x2, x3, x4: vector). The input of this procedure is given by restrictions on the content of each of the components of the mixture. They are specified on the form, and the program reads the data recorded in the Edit component. The variable a takes the values of the lower levels, and the variable b takes the values of the upper levels of the content for each ingredient of the mixture. The output of the procedure is four one-dimensional arrays *x1, x2, x3, x4*, the elements of which are the values of the coordinates of the vertices of the polyhedron corresponding to the content of the components of the mixture.

The resulting polyhedron has faces of the first and second orders. Faces of the first order are edges that have two identical coordinates, and faces of the second order are edges that have one identical coordinate. The program performs the procedure: procedure grani (x1, x2, x3, x4: vector; var ox1, ox2, ox3, ox4: vector), during which the points are compared, and those of them that have one identical coordinate form a face. In this case, the vertices that are repeated are excluded. At the output, we have four one-dimensional arrays *ox1, ox2, ox3, ox4*, which are the coordinates of the centers of the selected faces. The dimension of the resulting polyhedron is always q-1.

Further, among the obtained combinations, it is necessary to select those for which the sum of the concentrations is less than one, and add the content of the component that was omitted. To do this, the procedure is performed: procedure rebra (x1, x2, x3, x4: vector; var dx1, dx2, dx3, dx4: vector), at the input of which the coordinates of the vertices of the polyhedron *x1, x2, x3, x4* are given. During the procedure, all points are compared with each other. Points with two identical coordinates are searched for. These points form the edges of the polyhedron. At the output of the procedure, we have one-dimensional arrays *dx1, dx2, dx3, dx4,* the elements of which correspond to the coordinates of the centers of the edges.

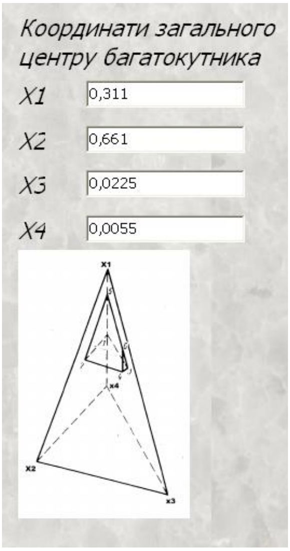
Variants with added components that satisfy conditions (2.25) and (2.45) represent the vertices of the desired polyhedron, which in the studied simplex forms an octagon. The resulting polyhedron has faces of the first and second orders: the first order is edges that have two identical coordinates, and the second is one coinciding coordinate. In this case, the vertices that are repeated are automatically excluded. The dimension of the resulting polyhedron is *q-1*.

The next step is to select the *r*-dimensional faces, or hyperfaces of the polyhedron, which are within $1 \le r \le q-2$. At $r=1$ it be an edge, at $r=2$ is a face, at $r=3$ is a hyperface. A face with dimension *r* is formed by a group of vertices that have the same coordinates in the number $q-r-1$. In the four-component system, a three-dimensional polyhedron is formed. Its edges have vertices with two identical coordinates ($4-1-1=2$), and the faces have vertices with one identical coordinate ($4-2-1=1$). In this case, the maximum number of vertices with the same coordinates $q-r-1$ is selected, because they form the *r*-dimensional face. The upper limit of the total number of *r*-dimensional faces is calculated by the formula:

$$\sum_{q-r-1=1}^{q-2} C_q^{q-r-1} \cdot 2^{q-r-1} \qquad (2.46)$$

In each of the selected faces, the coordinates of the centers (centroids) are determined as the average value of the coordinates of the vertices that form the corresponding face. When executing the procedure: procedure centr(x1, x2, x3, x4: vector; var cx1, cx2, cx3, cx4: real) the values of the coordinates of the candidate points for the plan are input to it.

Next, the coordinates of the common center (centroid) of the polyhedron are calculated as the average value of the coordinates of all vertices. The output is the coordinates of the common center of the polyhedron *cx1, cx2, cx3, cx4* (Fig.2.32).


Координати загального центру багатокутника

X1	0,311
X2	0,661
X3	0,0225
X4	0,0055

Fig. 2.32 – Programmatic determination of the common center of a polyhedron (centroid)

As a result, 27 candidate points for the experimental design are obtained.

As a response function that connects the initial parameters with the factors that change during the experiments, $y = \varphi(x_1, x_2, x_3, x_4)$ we chose an incomplete cubic model, which has the following form:

$$\hat{y} = \sum_{1 \le i \le q} \beta_i x_i + \sum_{1 \le i < j \le q} \beta_{ij} x_i x_j + \sum_{1 \le i < j < k \le q} \beta_{ijk} x_i x_j x_k \qquad (2.47)$$

To determine the numerical values of the coefficients of the polynomial (2.47), it is sufficient to have 14 points of the plan [96]. In order to select specific points for conducting experiments, the developed software uses a method of drawing up a plan containing a given number of experiments. It consists in the fact that the specified points must be maximally distant from each other in the factor space allocated on the simplex by restrictions. For this, the distance between all candidate points and the center of the octahedron (d_{mn}) is calculated by the formula:

$$d_{mn} = \left[\sum_{1 \le i \le q} \left(\frac{x_{mi} - x_{ni}}{b_i - a_i} \right)^2 \right]^{\frac{1}{2}}, \qquad (2.48)$$

where: m and n – furst and second points, i – component number

When executing the procedure: procedure vids_centr (x1,x2,x3,x4:vector; cx1,cx2,cx3,cx4: Real; a,b:vector1; var dc:vector) the distance from the candidate points in the plan x_1, x_2, x_3, x_4 to other points cx_1, cx_2, cx_3, cx_4 is determined by the formula (2.48). The input parameters for it are the coordinates of the candidate points in the plan *x1, x2, x3, x4* and the coordinates of another point *cx1, cx2, cx3, cx4* to which the distance needs to be found (in particular, this may be a point − the common center of the figure). In addition, the values of the restrictions on the content of the mixture components (arrays *a*, *b*) are passed to the procedure. The result of the procedure is a one-dimensional array *dc* containing the distances from each point to the center (or another point).

Procedure execution: procedure max_d (x1,x2,x3,x4,dc:vector; var max:integer) determines the number of the array element that has the maximum value of this distance. The input parameters of the procedure are one-dimensional arrays *x1, x2, x3, x4,* corresponding to the coordinates of the polyhedron, and a one-dimensional array *dc,* the elements of which are the distances from the points of the polyhedron to its center.

To select the points for conducting experiments, the procedure is performed: procedure vibir_tochok (tx1, tx2, tx3, tx4, dc: vector; dn: Real; var px1,px2,px3,px4: vector). In the software, the distances ($d_{i-1}; d_{i-5}$) are calculated between each of the obtained points and the remaining points according to the formula (2.48). Then the researcher selects the normalized distance (d'_{mn}), the value of which affects the number of points in the plan. It should be selected smaller when a larger number of points is required, and larger if their number is small enough. The normalized distance was selected, guided by the condition:

$$d_{ц}^{cp} \le d'_{mn} \le (2d_{ц}^{cp})^{\frac{1}{2}}, \qquad (2.49)$$

where: $d_{ц}^{ср}$ − average distance of a point from the center

The software assumes: $d_{ц}^{ср}$ =0.7424, and the normalized distance $d_{mn}^{'}$=1.0019. The coordinates of the points of the polyhedron *tx1, tx2, tx3, tx4* are given as parameters to the procedure input, the vector of distances from which to the center dc and the normalized distance dn are selected from condition (2.49). The procedure determines two points that are at the greatest distance from the center and from these points to the remaining candidate points. Points for which the distances to the two already selected points of the plan are less than the normalized one are included in the plan, and the rest are filtered out. Candidate points are arranged in order of decreasing distance from them to the center of the polyhedron. The first points in the arrays are those that are located at the maximum distance from the center of the figure. Then points that have a distance to the two selected points less than the normalized one are discarded. The procedure outputs four one-dimensional arrays *px, px2, px3, px4* containing the coordinates of the points included in the plan. If there are not enough points in the plan to build the model, it is necessary to reduce the selected norm, and if there are too many points, then you can either increase the normalized distance, or repeat all the above actions for the candidate points, not taking into account those that are already included in the plan. For the selected example, together with the two points that have already been selected for the experimental plan, we obtained 15 points, but only 14 are needed, so from these points we discarded the one that has the smallest distance to the common center of the polyhedron.

Thus, software has been developed using the simplex-centroid method according to the McLean–Anderson algorithm, which allows obtaining an experimental plan for studying a four-component system, which contains 14 necessary and sufficient points.

2.5.2.1. Computer-aided planning of experiments and optimization of composition composition to obtain microfibrillar filaments with improved properties. As shown in section 1.2, reducing the diameters of individual filaments to micro- and nano-sizes and introducing substances in their structure in the nanoscale is an effective method of modifying synthetic fibers and threads. Adding special substances to the mixture of incompatible polymers - compatibilizers [66], nanoadditives [47,59,60,69] or their compositions [67,68] allows you to control the process of *in situ* formation of fibrils of one component in the matrix of the other. Thus, introducing nanoparticles of the original [59,60] and modified silica [47] into the melt of a PP/CPA mixture allows you to obtain complex threads from nanofilled PP microfibrils with a high specific surface area and improved mechanical properties. Simultaneous addition of carbon nanotubes and sodium oleate (compatibilizer) to the PP/SPA mixture is more effective than individual substances [68]. Polyethylene terephthalate (PET) fibrils in a PP matrix with maximum length and minimum diameter were obtained by modifying a PET/PP blend using grafted maleic anhydride and TiO_2 nanoparticles [28].

When choosing the composition of the composition for obtaining fine-fiber materials, it is important to combine their desired indicators with the maximum content of the dispersed phase component, since the technology for their production from melts of polymer mixtures involves the extraction of the matrix polymer from a composite monofilament or film [59]. This is due to the fact that increasing the concentration of the dispersed phase polymer is a prerequisite for improving the economic performance of production and reducing the environmental load on the environment.

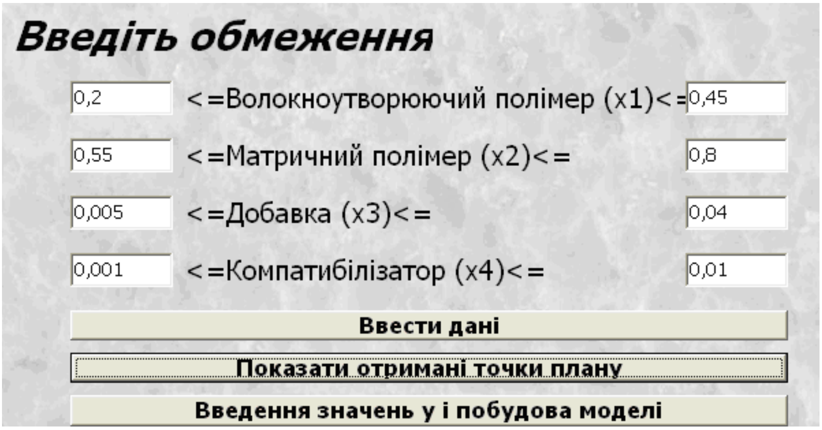
Based on this, we conducted research on optimizing the composition of the nanofilled compatibilized polypropylene/copolyamide blend with the maximum possible PP content to obtain complex microfibrillar yarns with predetermined characteristics. To reduce the time spent on studying the four-component PP/CPA/silica/siloxane composition, the experimental plan and the creation of a mathematical model were carried out using the developed software [31]. As an equation that establishes the relationship between the content of the components of the studied system and the properties of microfibrillar yarns, the program provides an incomplete third-order polynomial. To estimate the numerical values of its coefficients, an experimental plan was drawn up in the studied region of the factor space. The input variables were: x_1, x_2, x_3, x_4 - relative concentrations of PP, SPA, nanoadditive and compatibilizer, respectively. The following restrictions were imposed on the concentrations of the ingredients of the mixed composition:

$$0{,}2 \leq x_1 \leq 0{,}45; \; 0{,}55 \leq x_2 \leq 0{,}80;$$

$$0,005 \le x_3 \le 0,040; \; 0,001 \le x_4 \le 0,010 \qquad (2.50)$$

In this case, the condition (2.25) must be met. The following initial parameters were selected: y_1 – average diameter of PP microfibrils; y_2 – strength of complex microfibrillar threads at break; y_3 – hygroscopicity of threads.

In the created program, restrictions are introduced on the content of each of the components of the mixture – arrays *a* and *b* (Fig. 2.33).



Введіть обмеження

0,2	<=Волокноутворюючий полімер (x1)<=	0,45
0,55	<=Матричний полімер (x2)<=	0,8
0,005	<=Добавка (x3)<=	0,04
0,001	<=Компатибілізатор (x4)<=	0,01

Ввести дані
Показати отримані точки плану
Введення значень y і побудова моделі

Fig. 2.33 – Introduction of restrictions on the content of mixture components

Next, the program performs the following actions step by step, according to the described algorithm:

- determines the coordinates of the vertices of the polyhedron;
- selects the *r*-dimensional faces of the polyhedron ($1 \le r \le q-2$) and determines the coordinates of their centroids;

- calculates the coordinates of the common center of the polyhedron;

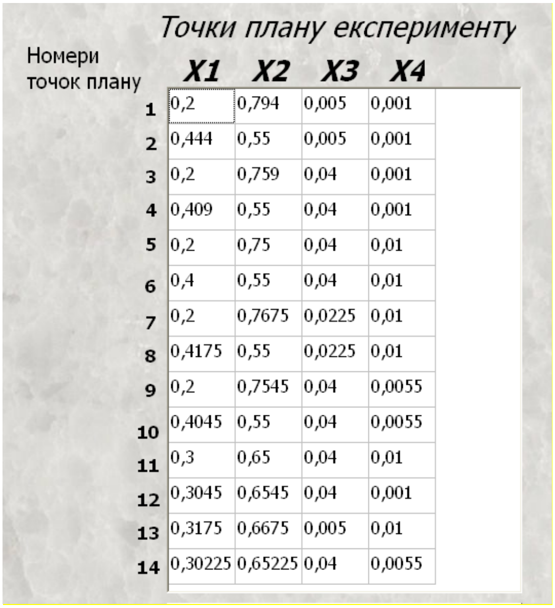
- finds the distance from the candidate points in the plan to the common center and determines the two points that lie at the greatest distance from the center;

- eliminates points for which the distance to the two selected ones is less than the normalized one (for the system under study, the following normalized distance was chosen: d'_{mn} =1.0019);

- determines the coordinates of the points that entered the plan.

Thus, in a few fractions of a second, the program creates an experimental plan for studying the composition of PP/CPA/silica/siloxane according to the McLean–Anderson algorithm, which contains 14 required points (Fig. 2.34).

Experimental studies were conducted using a thermodynamically incompatible PP/SPA mixture, in which the dispersed phase was isotactic polypropylene, and the dispersion medium was alcohol-soluble copolyamide. Pyrogenic silica (SiO_2) with a specific surface area of 324 m^2/g was chosen as the nanofiller, and an organosilicon substance (polyethylsiloxane) was chosen as the compatibilizer. The components were mixed in a worm-disc extruder.



Точки плану експерименту

Номери точок плану	X1	X2	X3	X4
1	0,2	0,794	0,005	0,001
2	0,444	0,55	0,005	0,001
3	0,2	0,759	0,04	0,001
4	0,409	0,55	0,04	0,001
5	0,2	0,75	0,04	0,01
6	0,4	0,55	0,04	0,01
7	0,2	0,7675	0,0225	0,01
8	0,4175	0,55	0,0225	0,01
9	0,2	0,7545	0,04	0,0055
10	0,4045	0,55	0,04	0,0055
11	0,3	0,65	0,04	0,01
12	0,3045	0,6545	0,04	0,001
13	0,3175	0,6675	0,005	0,01
14	0,30225	0,65225	0,04	0,0055

Fig. 2.34 – Plan of experiment for studying the composition of PP/CPA/silica/siloxane

The modifying additives were previously introduced into the PP melt, and the resulting granules were mixed with the matrix polymer (CPA). Composite monofilaments were formed on a laboratory stand at a temperature of 190 ^{0}C, with a draw ratio of 1000 %, and their thermoorientation drawing was carried out at a temperature of 150 ^{0}C with a multiplicity of 5. Complex threads from nanofilled PP microfibrils were obtained by extracting the matrix polymer from composite threads with an aqueous solution of ethyl alcohol. The strength of complex threads at break was determined using a KT 7010 AZ brand tearing machine. The hygroscopicity of the threads was estimated by the weight method at an air humidity of 98 %. The processes of PP structure formation in the matrix were studied using an MBD-15 optical microscope, determining the average diameter of microfibrils in the bundle after extraction of CPA from the composite extrudate.

Experimental studies carried out in accordance with the developed plan showed that for all compositions a microfibrillar structure is realized. The ratio of siloxane and silica significantly affects the formation of the morphology of the PP/SPA mixture (the average diameter of microfibrils varies from 1.6 to 7.1 μm). All modified systems are stably processed into composite monofilaments. After extraction of the matrix polymer from them, complex polypropylene microfibrillar filaments were obtained, the properties of which are given in Table. 2.5.

Table 2.5 – Effect of mixture composition on the average diameter of PP microfibrils and on threads properties

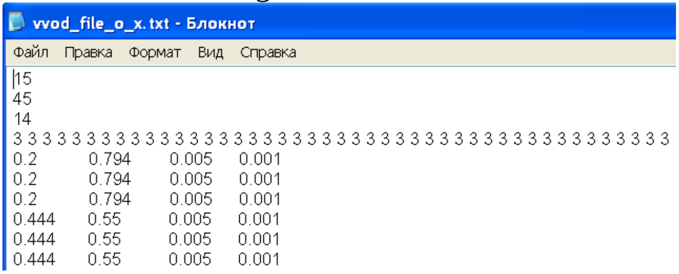
N of point of plan	Average diameter of microfibrils, μm	Threads strength, MPa	Hygroscopicity of threads, %
1	1,6	250	0,51
2	4,4	280	0,43
3	3,3	345	0,69
4	7,1	410	0,63
5	3,4	365	0,37
6	6,2	325	0,31
7	3,5	380	0,40
8	5,4	355	0,29
9	2,7	420	0,53
10	6,5	400	0,48
11	4,2	410	0,34
12	3,7	445	0,68
13	4,4	390	0,59
14	3,2	470	0,73

Based on the data in Table 2.5, the coefficients of the polynomial (2.47) were calculated by the least squares method in matrix form. The calculations were performed using a previously created program in the Object Pascal language [33]. As a result, a system of equations (2.51) was obtained, which is a mathematical model describing the process under study:

$$\begin{cases} \hat{y} = 3.46x_1 + 5.24x_2 + 4.17x_3 - 5.33x_4 + 2.78x_1x_2 + 6.37x_1x_3 - 46.83x_1x_4 + 7.89x_2x_3 - \\ \quad - 24.61x_2x_4 - 49.05x_3x_4 - 161.35x_1x_2x_3 - 0.97x_1x_2x_4 + 4.94x_1x_3x_4 + 3995x_2x_3x_4 \\ \hat{y} = 359.9x_1 + 340.4x_2 + 408.8x_3 - 263.9\beta_4x_4 + 457.7x_1x_2 548.6x_1x_3 - 2878x_1x_4 - 1128\beta_{23}x_2x_3 - \\ \quad - 1034\beta_{23}x_2x_4 - 2311x_3x_4 - 5155x_1x_2x_3 - 446.1x_1x_2x_4 + 248.4x_1x_3x_4 - 71890x_2x_3x_4 \\ \hat{y} = 0.85x_1 + 0.79x_2 + 1.01x_3 - 3.69x_4 + 0.86x_1x_2 + 1.58x_1x_3 - 19.72x_1x_4 - 8.00x_2x_3 - \\ \quad - 9.78x_2x_3 - 16.87x_2x_3 - 40.47x_1x_2x_3 - 9.79x_1x_2x_4 + 1.34x_1x_3x_4 - 116.3x_2x_3x_4 \end{cases} \tag{2.51}$$

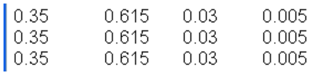
After determining the coefficients of the regression equation, a statistical analysis of the results was performed - the equations were checked for adequacy, i.e. the ability of the model to predict the results of research in a certain area with the required accuracy [97]. The adequacy of the model was checked using software developed by us earlier, which uses a fairly common method, which consists in comparing the estimates of the error variances between the response values calculated by the regression equation at some points of the factor space, on the one hand, and on the other hand, obtained independently [33]. This is equivalent to testing some linear hypothesis by calculating and analyzing the corresponding Fisher *F*-ratio. This method requires the presence of several observations for y at least at one of the points $\mathbf{x}_i$. For the created model, 15 different points were determined, each of which is repeated three times (i.e., a total of 45 observations).

The input file of the observation points, which contains the values of x_i, is shown in Fig. 2.35.



```
15
45
14
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
0.2      0.794    0.005   0.001
0.2      0.794    0.005   0.001
0.2      0.794    0.005   0.001
0.444    0.55     0.005   0.001
0.444    0.55     0.005   0.001
0.444    0.55     0.005   0.001
```

…   …   …

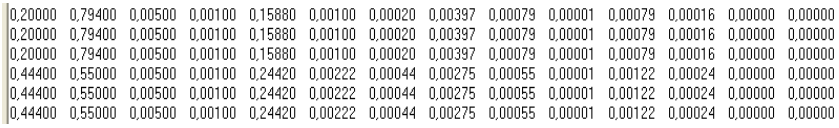


```
0.35      0.615   0.03     0.005
0.35      0.615   0.03     0.005
0.35      0.615   0.03     0.005
```

Fig. 2.35 – File *x.txt* – input data of observation points

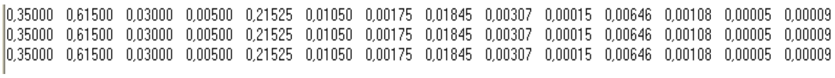
(In this case, Figure 2.35, as well as the following Figures 2.36, 2.37 and 2.38, for better understanding, data for the first, second and last points are presented.)

After entering the data x_i, the plan matrix for the developed model is programmatically generated (Fig. 2.36). For the convenience of the user, it is displayed in the form window using software created in the C++ language using modern programming methods [104,110].



```
0,20000  0,79400  0,00500  0,00100  0,15880  0,00100  0,00020  0,00397  0,00079  0,00001  0,00079  0,00016  0,00000  0,00000
0,20000  0,79400  0,00500  0,00100  0,15880  0,00100  0,00020  0,00397  0,00079  0,00001  0,00079  0,00016  0,00000  0,00000
0,20000  0,79400  0,00500  0,00100  0,15880  0,00100  0,00020  0,00397  0,00079  0,00001  0,00079  0,00016  0,00000  0,00000
0,44400  0,55000  0,00500  0,00100  0,24420  0,00222  0,00044  0,00275  0,00055  0,00001  0,00122  0,00024  0,00000  0,00000
0,44400  0,55000  0,00500  0,00100  0,24420  0,00222  0,00044  0,00275  0,00055  0,00001  0,00122  0,00024  0,00000  0,00000
0,44400  0,55000  0,00500  0,00100  0,24420  0,00222  0,00044  0,00275  0,00055  0,00001  0,00122  0,00024  0,00000  0,00000
```

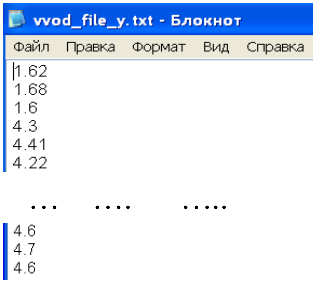
…   …   …



```
0,35000  0,61500  0,03000  0,00500  0,21525  0,01050  0,00175  0,01845  0,00307  0,00015  0,00646  0,00108  0,00005  0,00009
0,35000  0,61500  0,03000  0,00500  0,21525  0,01050  0,00175  0,01845  0,00307  0,00015  0,00646  0,00108  0,00005  0,00009
0,35000  0,61500  0,03000  0,00500  0,21525  0,01050  0,00175  0,01845  0,00307  0,00015  0,00646  0,00108  0,00005  0,00009
```

Fig.2.36 – Programmatically generated plan matrix

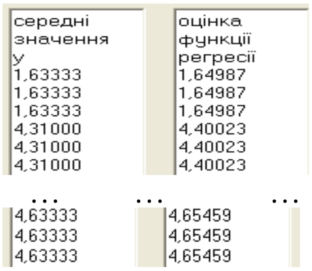
Thus, for the variable y_1, the experimental observation data are shown in Fig. 2.37.



```
1.62
1.68
1.6
4.3
4.41
4.22
  …    ….    …..
4.6
4.7
4.6
```

Fig. 2.37 – File *y.txt* – experimental observation data y_1

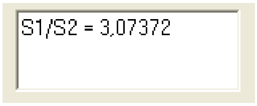
The average values calculated by the software for each observation point and the corresponding estimates of the regression function for the variable $y_1$of the model (2.51) are presented in Fig. 2.38.

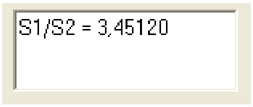


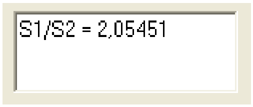
```
середні       оцінка
значення      функції
y             регресії
1,63333       1,64987
1,63333       1,64987
1,63333       1,64987
4,31000       4,40023
4,31000       4,40023
4,31000       4,40023
  …        …        …
4,63333       4,65459
4,63333       4,65459
4,63333       4,65459
```

Fig. 2.38 – The values of the regression function estimates and the average values obtained in the created software

The next step of the software is to determine Fisher's *F*-ratio using formula (2.16) for all output variables of the model (2.51). The obtained values are shown in Fig. 2.39.


S1/S2 = 3,07372


S1/S2 = 3,45120


S1/S2 = 2,05451

a) b) c)

Fig. 2.39 – *F*F-ratio obtained in the software application for y_1 (a), y_2 (b) та y_3 (c)

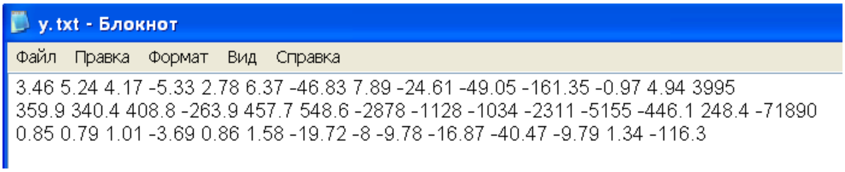
According to the general provisions, the hypothesis of the adequacy of the model $\hat{y}$ is not accepted at the significance level α if the ratio (2.16) exceeds the level quantile (1 – α) of the Fisher distribution, and in other cases it is accepted. Provided that $m > p$, the ratio $\frac{S_1^2}{S_2^2}$ has the form of the Fisher distribution [97]. The specified check is implemented by the software.

The results obtained indicate that the developed mathematical model is adequate: for the significance level α = 0.05 = $F(m-p,\quad n-m)$ = F(15–14, 45–15) = F(1, 16)=4.17, i.e. for all y from model (2.51) the calculated dispersion ratio $\frac{S_1^2}{S_2^2}$ is less than the value $F(m-p,\quad n-m)$.

The optimal content of ingredients in the studied four-component mixture was determined by the multi-criteria optimization method using software developed by us [111,112]. Multi-criteria optimization is the process of simultaneous optimization of several conflicting objective functions in a certain domain of definition. In the general case, the multi-criteria optimization problem is described by expression (2.30), while the objective functions that are investigated at the maximum are transformed into functions that are investigated at the minimum by formula (2.31) [100]. For a nano-filled compatibilized polypropylene/copolyamide mixture, the multi-criteria optimization problem has the following form:

$$\begin{gathered}
y_1 = 3.46x_1 + 5.24x_2 + 4.17x_3 - 5.33x_4 + 2.78x_1x_2 + 6.37x_1x_3 - 46.83x_1x_4 + 7.89x_2x_3 - \\
- 24.61x_2x_4 - 49.05x_3x_4 - 161.35x_1x_2x_3 - 0.97x_1x_2x_4 + 4.94x_1x_3x_4 + 3995x_2x_3x_4 \rightarrow \min \\
y_2 = 359.9x_1 + 340.4x_2 + 408.8x_3 - 263.9\beta_4x_4 + 457.7x_1x_2 548.6x_1x_3 - 2878x_1x_4 - 1128\beta_{23}x_2x_3 - \\
- 1034\beta_{23}x_2x_4 - 2311x_3x_4 - 5155x_1x_2x_3 - 446.1x_1x_2x_4 + 248.4x_1x_3x_4 - 71890x_2x_3x_4 \rightarrow \max \\
y_3 = 0.85x_1 + 0.79x_2 + 1.01x_3 - 3.69x_4 + 0.86x_1x_2 + 1.58x_1x_3 - 19.72x_1x_4 - 8.00x_2x_3 - \\
- 9.78x_2x_3 - 16.87x_2x_3 - 40.47x_1x_2x_3 - 9.79x_1x_2x_4 + 1.34x_1x_3x_4 - 116.3x_2x_3x_4 \rightarrow \max \\
0.2 \le x_1 \le 0.45 \qquad (2.52) \\
0.55 \le x_2 \le 0.8 \\
0.005 \le x_3 \le 0.04 \\
0.001 \le x_4 \le 0.01 \\
x_1 + x_2 + x_3 + x_4 = 1
\end{gathered}$$

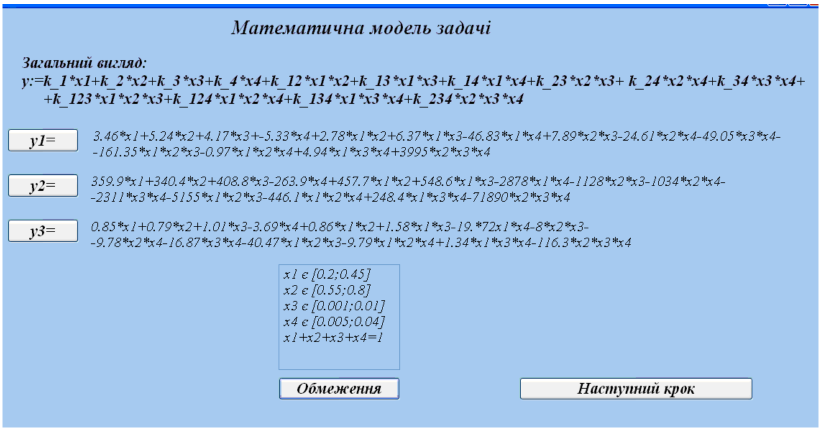
To solve this problem, we used the scalarization method, that is, we converted it to the solution of some scalar (single-criteria) problem. Scalarization was performed by the linear convolution method, using software [111]. The coefficients of the problem variables (y_1, y_2, y_3) are read from the file *y.txt* (Fig. 2.40).



```
3.46 5.24 4.17 -5.33 2.78 6.37 -46.83 7.89 -24.61 -49.05 -161.35 -0.97 4.94 3995
359.9 340.4 408.8 -263.9 457.7 548.6 -2878 -1128 -1034 -2311 -5155 -446.1 248.4 -71890
0.85 0.79 1.01 -3.69 0.86 1.58 -19.72 -8 -9.78 -16.87 -40.47 -9.79 1.34 -116.3
```

Fig.2.40 – File *y.txt* – coefficients of the mathematical model

Fig. 2.41 shows the initial data of the problem.



Загальний вигляд:

```
y:=k_1*x1+k_2*x2+k_3*x3+k_4*x4+k_12*x1*x2+k_13*x1*x3+k_14*x1*x4+k_23*x2*x3+ k_24*x2*x4+k_34*x3*x4+
    +k_123*x1*x2*x3+k_124*x1*x2*x4+k_134*x1*x3*x4+k_234*x2*x3*x4
```

```
y1=  3.46*x1+5.24*x2+4.17*x3+-5.33*x4+2.78*x1*x2+6.37*x1*x3-46.83*x1*x4+7.89*x2*x3-24.61*x2*x4-49.05*x3*x4-
     -161.35*x1*x2*x3-0.97*x1*x2*x4+4.94*x1*x3*x4+3995*x2*x3*x4

y2=  359.9*x1+340.4*x2+408.8*x3-263.9*x4+457.7*x1*x2+548.6*x1*x3-2878*x1*x4-1128*x2*x3-1034*x2*x4-
     -2311*x3*x4-5155*x1*x2*x3-446.1*x1*x2*x4+248.4*x1*x3*x4-71890*x2*x3*x4

y3=  0.85*x1+0.79*x2+1.01*x3-3.69*x4+0.86*x1*x2+1.58*x1*x3-19.*72x1*x4-8*x2*x3-
     -9.78*x2*x4-16.87*x3*x4-40.47*x1*x2*x3-9.79*x1*x2*x4+1.34*x1*x3*x4-116.3*x2*x3*x4
```

```
x1 є [0.2;0.45]
x2 є [0.55;0.8]
x3 є [0.001;0.01]
x4 є [0.005;0.04]
x1+x2+x3+x4=1
```

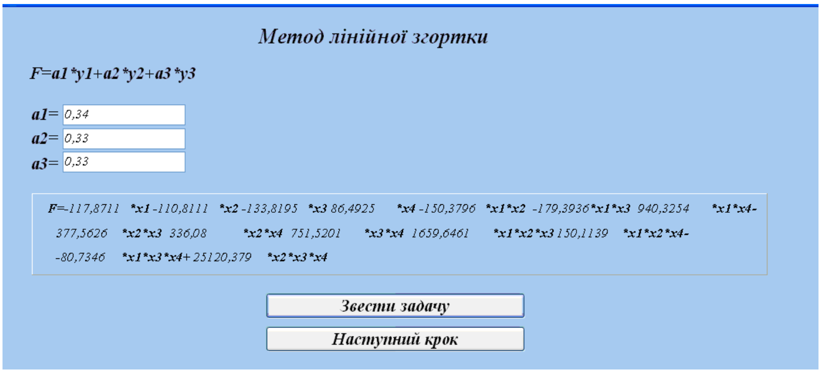
Fig. 2.41 – Initial data of the multi-criteria problem

The convolution weights, which determine the degree of importance of each criterion: $\alpha_1 = 0.34$; $\alpha_2 = 0.33$; $\alpha_3 = 0.33$ - are specified on the software form.

In the software, the minimization of the objective functions that are investigated to the maximum is carried out according to the formula: $\min(y) = -\max(y)$. Next, the linear combination of the objective functions is minimized, that is, the following problem is solved:

$$F = \alpha_1 \cdot y_1 + \alpha_2 \cdot y_2 + \alpha_3 \cdot y_3 \to \min \qquad (2.53)$$

By clicking the “reduce problem” button, the form (Fig. 2.42) displays the single-criteria problem that was obtained as a result of the calculations.


Метод лінійної згортки

*F=a1*y1+a2*y2+a3*y3*

a1= 0,34
a2= 0,33
a3= 0,33

```
F=-117,8711 *x1 -110,8111 *x2 -133,8195 *x3 86,4925 *x4 -150,3796 *x1*x2 -179,3936*x1*x3 940,3254 *x1*x4-
377,5626 *x2*x3 336,08 *x2*x4 751,5201 *x3*x4 1659,6461 *x1*x2*x3 150,1139 *x1*x2*x4-
-80,7346 *x1*x3*x4+ 25120,379 *x2*x3*x4
```

Fig.2.42 – The optimization problem is transformed into a single-criteria one

Thus, using software, a mathematical model was created in the form of a single-criteria problem (2.54), which determines the influence of the nanoadditive and compatibilizer on the dimensional characteristics of polypropylene microfibrils and the properties of complex threads.

$$F=-117.87\cdot x_1-110.81\cdot x_2-133.82\cdot x_3+86.49\cdot x_4-150.38\cdot x_1\cdot x_2-$$
$$-179.39\cdot x_1\cdot x_3+940.33\cdot x_1\cdot x_4-377.56\cdot x_2\cdot x_3+$$
$$+336.08\cdot x_2\cdot x_4+751.52\cdot x_3\cdot x_4+1659.65\cdot x_1\cdot x_2\cdot x_3+150.11\cdot x_1\cdot x_2\cdot x_4-$$
$$-80.73\cdot x_1\cdot x_3\cdot x_4+25120.38\cdot x_2\cdot x_2\cdot x_4\rightarrow \min$$

$$\begin{gathered}0.2\le x_1\le 0.45\\ 0.55\le x_2\le 0.8\\ 0.005\le x_3\le 0.04\\ 0.001\le x_4\le 0.01\\ x_1+x_2+x_3+x_4=1\end{gathered}\qquad (2.54)$$

The developed model is much simpler than the multi-criteria optimization problem and can be solved by one of the known methods and used to optimize the four-component composition.

In order to determine the optimal composition of the studied mixture, the so-called conditional optimization problem was solved, which is associated with optimization under constraints on the variables. To move from the conditional optimization problem of the studied four-component mixture with constraints to the problem without constraints, the penalty function method was used [100,101], in which by $P(x)$ the function *Z* will be "penalized" if the constraints are violated (i.e., its value is increased), while the minimum of the function *Z* will be located inside the constraint region. Under constraints $c_j(x) > 0$, $j = 1, 2, ..., m$, function $P(x)$ is written by equation (2.33). The minimization problem for the polypropylene/copolyamide/silica/siloxane system is to minimize function $Z = f(x)$ under constraints $c_j(x) > 0$, $j = 1, 2, ..., m$, then function Z will have the following form:

$$Z = \varphi(x,r) = f(x) + r \cdot \sum_{j=1}^{m} \frac{1}{c_j(x)} \qquad (2.55)$$

Provided that *x* has admissible values, i.e. values for which $c_j(x) > 0$, the function *Z* will take values that are larger than the corresponding ones, and the difference can be reduced by r. In the case when x has admissible indices, at the same time approaching the boundary of the constraint region, and at least one of the functions $c_j(x)$ is close to zero, the values of the functions $P(x)$ and *Z* will be quite large, i.e. the influence of $P(x)$ function is manifested in the formation of a "crest with sharp edges” along the boundary of the constraint region. Provided that the search starts from an admissible point and the unconstrained function $\varphi(x,r)$ is minimized, the minimum will, of course, be reached inside the admissible region for problems with constraints. Since *r* is a sufficiently small value, to reduce the influence of $P(x)$ at the minimum point, it is necessary to make the minimum point of $\varphi(x,r)$ the unconstrained function coincide with the minimum point of the problems with constraints.

To solve problem (2.54), the software creates an unconstrained function using a penalty:

$$F=-117.87\cdot x_1-110.81\cdot x_2-133.82\cdot x_3+86.49\cdot x_4-150.38\cdot x_1\cdot x_2-$$
$$-179.39\cdot x_1\cdot x_3+940.33\cdot x_1\cdot x_4-377.56\cdot x_2\cdot x_3+$$
$$+336.08\cdot x_2\cdot x_4+751.52\cdot x_3\cdot x_4+1659.65\cdot x_1\cdot x_2\cdot x_3+150.11\cdot x_1\cdot x_2\cdot x_4-$$
$$-80.73\cdot x_1\cdot x_3\cdot x_4+25120.38\cdot x_2\cdot x_2\cdot x_4+$$
$$+r\cdot\left(\frac{1}{x_1-0.2}+\frac{1}{0.45-x_1}+\frac{1}{x_2-0.55}+\frac{1}{0.8-x_2}+\frac{1}{x_3-0.005}+\frac{1}{0.04-x_3}+\right.$$
$$\left.+\frac{1}{x_4-0.001}+\frac{1}{0.01-x_4}+(1-x_1-x_2-x_3-x_4)^2\right)\to\min$$

(2.56)

The closer to the minimum the penalty is under the condition $r\to 0$, the smaller the gradient of the function will be. The search ends under the condition $r_n\le\varepsilon$, where ε - is a given sufficiently small number. As a result of applying the penalty function method, we obtained an unconditional optimization problem.

To solve the optimization problem of the four-component composition (2.56), the gradient method with step splitting was used [101]. It is assumed that the functions *f(x)*, ∇f exist and are continuous. The method is based on an iterative procedure, which is defined by the formula:

$$x^{(k+1)} = x^{(k)} + \lambda_k \cdot S_k\,, \tag{2.57}$$

where: λ_k – step size,

S_k – vector in the direction $x^{(k+1)} - x^{(k)}$

Gradient methods differ only in the way they determine λ_k, and S_k are usually found by solving the optimization problem *f(x)* in the direction of S_k. The direction S_k depends on how the function *f(x)* is approximated. To do this, a sequence of points $\left\{x^{(k)}\right\}$, *k*=0,1,… is constructed that satisfy the following condition:

$$f\left(x^{(k+1)}\right) < f\left(x^{(k)}\right),\ k=0,1,\ldots. \tag{2.58}$$

Sequence points $\{x_k\}$ are calculated according to the following rule:

$$x^{k+1} = x^k - \lambda_k \cdot grad\ f\left(x_k\right),\ k=0,1,\ldots \tag{2.59}$$

The step size λ_0 is not changed as long as the function decreases at the points of the sequence. The condition for the end of the calculations is the fulfillment of the inequalities (the gradient $grad\ f\left(x^{(k)}\right)$ is close to zero):

$$\left|\frac{df\,(x^{(k)})}{dx^{(i)}}\right| \le \varepsilon,\ i = 1, 2, ..., n \tag{2.60}$$

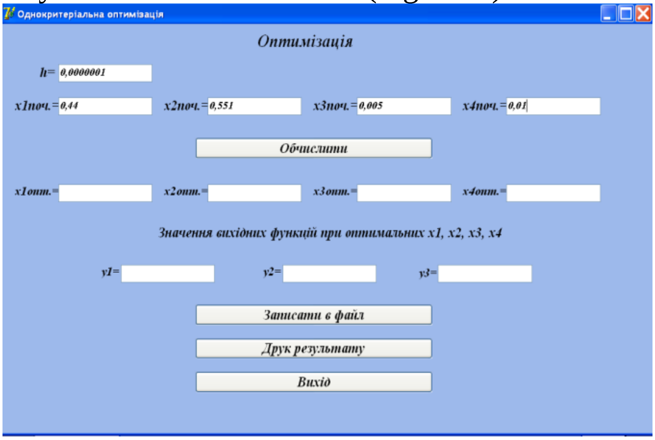
or

$$\left\|grad\ f\left(x^{(k)}\right)\right\| = \sqrt{\sum_{i=1}^{n}\left[\frac{df\,(x^{(k)})}{dx_i}\right]} \le \varepsilon\,, \tag{2.61}$$

where: ε – given a fairly small number

If the decrease condition is not met, the step size is usually reduced by half ($\lambda_k = \frac{\lambda_k}{2}$) until the inequality $f\left(x^{(k+1)}\right) < f\left(x^{(k)}\right)$ is met and the calculations are continued.

Calculations to determine the optimal content of ingredients in the studied mixture were performed using software [112]. The researcher begins work with the program by specifying on the form the starting point: $x^{(0)} = \begin{pmatrix} 0.44 \\ 0.551 \\ 0.005 \\ 0.001 \end{pmatrix}$, initial values of variables, step size $\lambda_0 = 0.0000001$ and a sufficiently small number $\varepsilon = 0.01$ (Fig. 2.43).



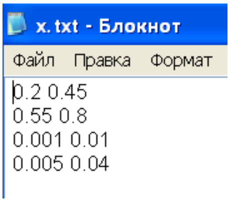
h= 0,0000001			
x1поч.= 0,44	x2поч.= 0,551	x3поч.= 0,005	x4поч.= 0,01
x1опт.=	x2опт.=	x3опт.=	x4опт.=

Значення вихідних функцій при оптимальних x1, x2, x3, x4

y1=	y2=	y3=

Fig. 2.43 – Form “Single-criteria optimization” – entering initial values

Constraints on the problem variables are read from the file x.txt (Fig. 2.44).



```
0.2 0.45
0.55 0.8
0.001 0.01
0.005 0.04
```

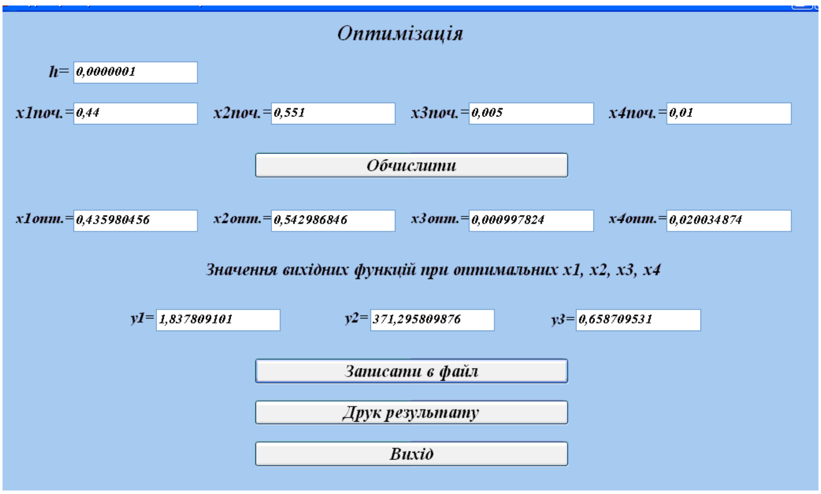
Fig. 2.44 – File x.txt for entering constraints on problem variables

In this case, the program performs the following steps of the algorithm:

- finds partial derivatives at the point $x^{(0)}$;
- checks the stopping condition at $grad\ f\left(x^{(k)}\right)$;
- calculates the value of the function at the initial point $x^{(0)}$, $F(x^{(0)})$;
- takes a step along the antigradient direction $x^{(1)} = x^{(0)} - \lambda_0 \cdot grad\ \ f(x^{(0)})$;
- calculates the value of the function at the point $x^{(1)}$. $F(x^{(1)})$.
- since $F(x^{(1)}) > F(x^{(0)})$, the step size decreases: $\lambda_1 = \dfrac{0.0000001}{2} = 0.00000005 \cdot$
- repeats the described operations until $grad\ f\left(x^{(k)}\right) < \varepsilon$.

At the last step of the algorithm we obtain the following values: $x^{(n)} = \begin{pmatrix} 0.43598 \\ 0.54298 \\ 0.00099 \\ 0.02003 \end{pmatrix}$.

At the same time, the optimal values of the problem variables appear in the corresponding fields (Fig. 2.45).



Оптимізація

h= 0,0000001

x1поч.= 0,44 x2поч.= 0,551 x3поч.= 0,005 x4поч.= 0,01

x1опт.= 0,435980456 x2опт.= 0,542986846 x3опт.= 0,000997824 x4опт.= 0,020034874

Значення вихідних функцій при оптимальних x1, x2, x3, x4

y1= 1,837809101 y2= 371,295809876 y3= 0,658709531

Fig. 2.45 – Form "Single-criteria optimization" – calculation results

Thus, using the developed software, the values of the variables x_1, x_2, x_3, x_4 are calculated, which are the optimal contents of the ingredients of the studied four-component mixture, and the initial parameters y_1, y_2, y_3, which characterize the dimensional characteristics of PP microfibrils and the properties of polypropylene complex threads based on them.

The optimal composition of the PP/SPA/silica/siloxane composition for the formation of monofilaments, calculated using computer programs at all stages of the study, is as follows, wt. %: polypropylene - 43.6; copolyamide - 54.3; silica - 0.1; siloxane - 2.0. It was established that the simultaneous introduction of nanosized silica and organosilicon liquid into the melt of the PP/SPA mixture in an amount of 1.9 and 0.1 wt. %, respectively, made it possible to implement microfibrillar morphology in the four-component composition. At the same time, the polymer content of the dispersed phase in it is almost 1.5 times higher than in the unfilled one. Increasing the concentration of the fiber-forming polymer in the composition is one of the prerequisites for increasing economic indicators and environmental safety of the production of fine-fiber materials by processing polymer mixtures. Studies of the properties of complex microfibrillar threads formed from a composition of optimal composition have shown their significant improvement. Thus, the breaking strength is at the level of the best samples of traditional textile PP threads. The introduction of siloxane into the composition provides a significant increase in the resistance of the studied threads to self-erasure (1027 versus 516 thousand cycles for textile threads). Modified complex threads are also characterized by improved hygienic properties - their hygroscopicity is 17 times higher than that of conventional textile threads.

Conclusion

To study four-component compositions and establish the relationship between the content of ingredients and the properties of products obtained from them, several software programs have been developed that allow you to build an experimental plan, develop mathematical models, check their adequacy and optimize the composition of the mixture. The experimental plan for the influence of the ratio of ingredients in a four-component heterogeneous system is created using software using the simplex-centroid method. In this case, the placement of candidate points in the simplex, which is a tetrahedron, is carried out according to the McLean-Anderson algorithm, and the necessary and sufficient number of plan points is 14. By calculating the coordinates of the points of the experimental plan, the content of which is subject to two-sided restrictions, a mathematical model of the process under study is obtained in the form of a system of regression equations. The model is used to find the optimal composition by the method of multi-criteria optimization. For this, the multi-criteria problem is converted to a solution by the method of single-criteria linear convolution. The transition from a conditional optimization problem with constraints to an unconstrained problem is carried out using the penalty function method. The optimal values of the composition ingredient content and the initial parameters characterizing the properties of products based on it are determined using the gradient method with step splitting.

The developed software was used, in particular, to optimize the composition of the polypropylene/copolyamide mixture, which contained silica as a nanofiller and organosilicon as a compatibilizer. It was found that the combined action of both modifying additives with a total content of 2.0 wt. % allows to implement the process of forming PP microfibrils in the SPA matrix and to achieve an increase in the concentration of the dispersed phase component to almost 45 wt. %, which is a prerequisite for improving the economic and environmental performance of production. Complex polypropylene threads obtained from a composition with an optimal composition are characterized by increased strength, resistance to self-erasing and hygroscopicity.

Thus, the developed programs for mathematical planning and analysis of experiments in the study of three- and four-component compositions can be used to study any mixture systems and will help accelerate the implementation of research and obtain products with the best performance from them.

In conclusion, a few words about the prospects for further development of the polymer composites industry and software for their creation. Despite the fact that various types of composites have been used by mankind since ancient times, the goal was usually to overcome some of the shortcomings of one of the components, for example, increasing the strength of clay bricks by adding straw. Today, with a scientifically sound composition of the mixture, materials with completely new properties or with significantly improved indicators are created. In recent years, polymer composites, including nano-filled ones, have played an increasingly important role, the total production volume of which is of the same order as the production of all metals. At the same time, the number of varieties of polymer materials exceeds the number of different types of steel. The variety of polymer mixtures and composites will further strengthen this trend in the future. The main reason for the growing interest in such materials in the world is due to the combination of low cost and small mass with excellent properties. The main problem when using polymer compositions, from the point of view of ecology, is the complexity of utilization and return to secondary processing of production waste. The solution to these problems can be the search for new types of biotechnology for the production of both traditional and new types of monomers and polymers (including fiber-forming ones). As an example of the implementation of fundamentally new technologies, polylactide fibers, films and nanofilled plastics obtained on the basis of natural polysaccharides can be cited. At the same time, there are no complex environmental problems due to the non-toxicity of the initial and finished products and the possibility of their recycling, assimilation and biodegradation in the environment.

The current state and prospects for the development of polymer composite materials, including nanofilled fibrous ones, are considered, indicating that the advantages of polymer mixtures and composites are a prerequisite for their further widespread use in various industries, as well as in everyday life and, most importantly, in medicine. The research and creation of new types of polymer composites will be greatly facilitated by the widespread use of mathematical modeling methods using software.

REFERENCES

1. Нові функціональні речовини і матеріали хімічного виробництва: зб. матеріалів цільової програми наукових досліджень НАН України / НАН України. Київ: Академперіодика, 2021. 332 с.

2. Hassan T., Salam A., Khan A. Functional nanocomposites and their potential applications: a review. *J. of Polym. Research*. 2021. Vol. 28, № 2.

3. Utraki L.A., Wilkie Ch.A. Polymer blends handbook.: monograph. London: Springer New York Heidelberg Dordrecht, 2014. 2373 p.

4. Saharudin M.S., Hasbi S., Nazri M.N.A., Inam F. A review of recent developments in mechanical properties of polymer–clay nanocomposites. In book: Advances in Manufacturing Engineering. Springer Singapore: Sept. 2020. P. 107-129

5. Nurazzi N.M., Asyraf M.R.M., Khalina A., Abdullah N., Sabaruddin F.A., Kamarudin S.H., Ahmad S., Mahat A.M., Lee Ch.L., Aisyah H.A., Norrrahim M.N.F., Ilyas R. A., Harussani M. M., Ishak M. R., Sapuan S. M. Fabrication, Functionalization, and Application of Carbon Nanotube-Reinforced Polymer Composite: An Overview. *Polymer*. 2021. №13. P.1047- 1091.

6. Morsi M.A., Rajeh A., Al-Muntaser A.A. Reinforcement of the optical, thermal and electrical properties of PEO based on MWCNTs/Au hybrid fillers: Nanodielectric materials for organoelectronic devices. *Compos. Part B Eng.* 2019. P. 106957.

7. Huang B. Carbon nanotubes and their polymeric composites: the applications in tissue engineering. *Biomanufacturing Rtviews*. 2020. Vol 5. №3.

8. Chen J., Liu B., Gao X., Xu D. A review of the interfacial characteristics of polymer nanocomposites containing carbon nanotubes. *RSC Adv.* 2018. Vol.8. P. 28048-28085.

9. Abdelgawad A.M., Hudson S.M., Rojas O.J. Antimicrobial wound dressing nanofiber mats from multicomponent (chitosan/silver-NPs/polyvinyl alcohol) systems. *Carbohydr Polym.* 2014. Vol.100. P.166-178.

10. Bhandari V., Jose S., Badanayak P., Sankaran A., Anandan V. Antimicrobial Finishing of Metals, Metal Oxides, and Metal

Composites on Textiles: A Systematic Review. *Ind. Eng. Chem. Res.* 2022. Vol. 61, №1. P.86-101.

11. Fakoori E., Karami, H. Preparation and Characterization of ZnO-PP Nanocomposite Fibers and Non-Woven Fabrics. *J. Text. Inst.* 2018. Vol.109, №9. P.1152-1158.

12. Biomedical Applications of Polymeric Nanofibers // by editors R. Jayakumar, V. N. Shantikumar, monograph. New York: Springer, 2012. 283 p.

13. Kanjwal M.A., Barakat N.M., Shceikh F.A., Balk W., Khil M.S., Kim H.Y. Efect of silver Content and Morphology on the catalic Activity of Silver-grafted Titanium Oxide Nanostructure. *Fibers and Polymers*. 2010. Vol.11, №5. P.700-709.

14. Дзюбенко Л.С., Сап'яненко О.О., Горбик П.П., Плаван В.П., Резанова Н.М., Лутковський Р.А., Вільцанюк О.А. Властивості шовного матеріалу з поліпропілену модифікованого частинками нанорозмірного срібла та кремнезему. *Наносистеми, наноматеріали, нанотехнології*. 2018. т.16, №2. С.347-362.

15. Mukhopadhyay R., Bhaduri D., Sarkar B. Clay–polymer nanocomposites: progress and challenges for use in sustainable water treatment. *J. Hazardous Materials*. 2020. Vol.383. P.121-125.

16. Mar Orta M., Martín J., Luis Santos J., Aparicio I., Medina-Carrasco S., Alonso E. Biopolymer-clay nanocomposites as novel and ecofriendly adsorbents for environmental remediation. *Applied Clay Science*.2020. Vol. 198. P. 105838.

17. Tskhe Y., Buzgo M., Simaite A. Electorospun nanofibers with photocatalytic particles for carbon sorption. *Nanotech / Biotech France 2021 and joint virtual conferences*: Book of abstracts International Conference, 23-25 June. 2021 P.21.

18. Murugesan S., Murugesan S., Scheibel T. Copolymer/clay nanocomposites for biomedical applications. *Advanced Functional Materials*. 2020. Vol.30, №17. P. 1908101.

19. Felton G.P. Biodegradable polymers: Processing, Degradation, and Applications: monograph: Nova Science Publishers, 2011. 700 p .

20. Chu C.C. Advancement in Biodegradation Study and Applications. *Biodegradable Polymers*. 2015. Vol.1. P.22-29.

21. Lendlein A., Sisson A.L. Handbook of Biodegradable

Polymers: Synthesis, Characterization and Applications: monograph: Wiley-VCH, 2011. 391 p.

22. Abbasi M.A., Javadi A., Nazockdast H., Fathi A., Altstaedt V. Effect of Dispersion and Selective Localization of Carbon Nanotubes on Rheology and Electrical Conductivity of Polyamide 6 (PA6), Polypropylene (PP), and PA6/PP Nanocomposites. *J. Polym. Sci. Part B Polym. Phys*. 2015. Vol.53, №5. P.368-378.

23. Soares da Silva J.P., Soares B. G., Silva A.A. , Livi S. Double Percolation of Melt-Mixed PS/PBAT Blends Loaded With Carbon Nanotube: Effect of Molding Temperature and the Non-covalent Functionalization of the Filler by Ionic Liquid. *Front. Mater.* Aug. 2019.p. 191.

24. Nuzzo A., Coiai S., Carroccio S.C., Dintcheva N.T., Gambarotti C. Filippone G. Heat-Resistant Fully Bio-Based Nanocomposite Blends Based on Poly(lactic Acid). *Macromol. Mater. Eng.* 2014. Vol.299, №1. P.31-40.

25. Plavan V.P., Rezanova V.G., Budash Yu.O., Ishchenko O.V., Rezanova N.M. Influence of aluminum oxide nanperticles on formation of the structure and mechanical properties of microfibrillar composites. *Mechanics of Composite Materials*. 2020. Vol.56, № 3. P.1-14.

26. Li W., Schlarb A.K., Evstatiev M. Study of PET/PP/TiO_2 microfibrillar-structured composites: Part 1. Preparation, morphology and dynamic mechanical analysis of fibrillized blends. *J. Appl. Polym. Sci.* 2009. №113. P. 1471-1479.

27. Резанова Н.М., Савченко Б.М., Плаван В.П., Булах В.Ю., Сова Н.В. Закономірності одержання нанонаповнених полімерних матеріалів з матрично-фібрилярною структурою. *Наносистеми, наноматеріали, нанотехнології*. 2017. т.15, №3. С.559-571.

28. Li W., Karger-Koksis J., Schlarb A.K. Dispersion of TiO_2 Particles in PET/PP/TiO_2 and PET/PP/PP-g-MA/TiO_2 Composites Prepared with Different Blending Procedure. *Macromol. Mater. Eng.* 2009. №294. P.582-589.

29. Zhu B., Bai T., Wang P., Wang Y., Liu Ch., Shen Ch. Selective dispersion of carbon nanotubes and nanoclay in biodegradable poly(ε-caprolactone)/poly(lactic acid) blends with

improved toughness, strength and thermal stabilityInt. *J Biol Macromol.* 2019.

30. Rezanova V.G., Rezanova N. M., Viltsanyuk O.O. Software for planning and simulation in research of nano-filled three-component systems. *Nanosistemi, Nanomateriali, Nanotehnologii*. 2022. Vol.20, №2. P.423-435.

31. Rezanova V.G., Rezanova N.M. Mathematical Modelling and Software Development to Optimize the Composition of Four-Component Nanofilled Systems. *Nanosistemi, Nanomateriali, Nanotehnologii*. 2020. Vol.18, №4. P.863-874.

32. Резанова В.Г. Резанова Н.М. Програмне забезпечення для дослідження полімерних систем : монографія. Київ: АртЕк, 2020. 358 с.

33. Щербань В.Ю., Краснитський С.М., Резанова В.Г. Математичні моделі в САПР. Обрані розділи та приклади застосування: монографія. Київ: КНУТД, 2011. 219 с.

34. Pukanszky B. Interfaces and interphases in multicomponent materials: past, present, future. *European Polymer Journal*. 2005. Vol.41, №4. P.645-662.

35. Utracki L.A. Clay-Containing nanocomposites: monograph. UK.: Rapza Technology limited, 2004. Vol.2. 325 p.

36. Clay Nanoparticles. Flame retardant potential of clay nanoparticles / by edit. A. Kausar. monograph: Elsevier Ltd., 2020. P. 169-184.

37. Sfiligoj Smole M., Stana Kleinschek K. Nanofilled polypropylene fibres: in book Nanofibers and nanotechnology in textiles / by edit. P.J. Brown, K.S. Stevens. monograph, North America: Wootheat Publishing, 2007. 530 p.

38. 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.

39. Fornes T.D., Baur J.W., Sabba Y., Thomas E.L. Morphology and properties of melt-spun polycarbonate fibers containing single- and multiwall carbon nanotubes. *Polymer*. 2006. Vol.47, №5. P.1704-1714.

40. Wu M.L., Chen Yu., Zhang L., Zhan H., Qiang L., Wang J.N. High-Performance Carbon Nanotube/Polymer Composite Fiber

from Layer-by-Layer Deposition. *ACS Appl. Mater. Interfaces*. 2016. Vol.8, №12. P.8137-8144.

41.Vimbela G.V., Ngo S.M., Fraze C., Yang L., Stout D.A. Antibacterial properties and toxicity from metallic nanomaterials *Int. Journ. of Nanomedicine,* 2017. Vol. 12. P. 3941-3965.

42. Anthony L.A. Science and thechnology of polymer nanomer nanofibers: monograph. Hoboken, New Jersey. USA: John Wiley & Sons Inc., 2008. 424 p.

43. Yeo S.Y., Lee H.J., Jeong S.H. Preparation of nanocomposite fibers for permanent antibacterial effect. *Journal of Materials Science*. 2008. Vol.38, №10. P.125-132.

44. Патент України № UA 32794, 2008. МКП D01/10. Антимікробна шовна хірургічна нитка з наночастинками срібла і міді. О.Ф. Петренко, М.В. Косінов, В.Г. Каплуненко. Бюл. 10.

45. Мельник И.А., Цебренко М.В. Закономерности формования модифицированных полипропиленовых волокон. *Хим. волокна.* 2008. №5. С. 15-18.

46. Мельник І.А., Резанова В.Г., Цебренко М.В., Резанова Н.М., Готфрід А.О., Вільцанюк О.А. Поліпропіленові хірургічні мононитки з антимікробними властивостями. *Вісник КНУТД.* 2013. № 2. С.79-86.

47. Rezanova N.M., Plavan V.P., Rezanova V.G., Bohatyryov V.M. Regularities of producing of nano-filled polypropylene microfibers. *Vlakna a Textil*. 2016. №2. P.3-8.

48. Suresh G.A. Processing and Properties of Nanocomposites: monograph. Singapure: Toh Tuck Link World Scientific Publishing Co., 2007. 548 p.

49. Harrats C., Thomas S., Groeninckx G. Micro- and nanostructured multiphase polymer blend systems phase morphology and interfaces: monograph. Boca Raton, London New York: Taylor & Francis Group Inc., 2006. 473 p.

50. Nicolov N., Evstatiev M., Fakirov S. Morphology of microfibrillar reinforced composites PET/PA 6 blend. *Polymer*. 1996. №37. P.4455-4463.

51. Fakirov S., Bhattacharyya D., Lin R. J. T., Fuchs C., Friedrich K. Contribution of coalescence to microfibril formation in polymer blends during cold drawing. *J. Macromol. Sci. Part B*. 2007, №46. P.183-194.

52. Huang M., Schlarb A.K. Polypropylene/poly(ethylene terephthalate) microfibrillar reinforced composites manufactured by fused filament fabricatio. *J. Appl. Polym. Sci.* 2021. Vol. 138, № 23. P. e50557.

53. Shen J., Huang W., Zuo S. In-situ fiberized poly(ethylene terephthalate) as a reinforcement to poly(propylene) matrix. *Macromol. Mater. and Eng.* 2003. № 288. P.658-664.

54. Taepaiboon P., Junkasem J., Dangtungee R., Amornsakchai T., Supaphol P. In Situ microfibrillar-reinforced composites of isotactic polypropylene/recycled poly(ethylene terephthalate) system and effect of compatibilizer. *J. Appl. Polym. Sci.* 2006. №102. P.1173-1181.

55. Jayanarayanan K., Thomas S., Joseph K. Morphology, static and dynamic mechanical properties of in situ microfibrillar composites based on polypropylene/poly(ethylene terephthalate) blends. *Composites. Part A.* 2008. № 39. P.164-175.

56. Thomas S., Mishra R., Kalarikka N. Micro and nano fibrillar composites (mfcs and nfcs) from polymer blends: monograph: Woodhead Publishing, 2017. 372 p.

57. Rezanova N.M, Rezanova V.G., Plavan V.P., Viltsaniuk O. O. Polypropylene fine-fiber filter materials modified with nano-additives. *Functional Materials*. 2019. Vol.26, №2. P.389-396.

58. Tran N.H.A., Brünig H., Boldt R., Heinrich G. Morphology Development from Rod-like to Nanofibrillar Structures of Dispersed Poly (Lactic Acid) Phase in a Binary Blend with Poly (Vinyl Alcohol) Matrix along the Spinline. *Polymer*. 2014. Vol.55, №24. P.6354-6363.

59. Резанова Н.М., Будаш Ю.О., Плаван В.П. Інноваційні технології хімічних волокон. навч. посіб. Київ: КНУТД, 2017. 240 с.

60. Tsebrenko M.V., Rezanova V.G., Tsebrenko I.O. Features of obtaining of polypropylene microfibers with nanosize fillers. *J. of Mater. Sci. and Eng.* 2010. Vol.4, №6. P.36-44.

61. Jin K., Eyer S., Dean W., Kitto D., Bates F.S., Ellison C.J. Bimodal nano- and micro-fiber nonwovens by melt blowing immiscible ternary polymer blends. *Industrial and Eng. Chem. Res.* 2020. Vol. 59, № 12, № P. 5238–5246.

62. Ellison C.J., Phatak A., Giles D.W., Bates F.S. Melt blown nanofibers: Fiber diameter distributions and onset of fiber breakup. *Polymer*. 2007. Vol.48, № 11. P. 3306-3316.

63. 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.

64. 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 Inter. J. of Advan. Manufact. Techn*. 2019. №101. P.2681-2688.

65. Beloshenko V., Chishko V., Plavan V., Rezanova N., Savchenko B., Sova N., Vozniak I. Production of Filter Material from Polypropylene/Copolyamide Blend by Material Extrusion-Based Additive Manufacturing: Role of Production Conditions and ZrO_2 Nanoparticles. *3D Printing and Additive Manufacturing*. 2021.Vol.8, №4. P.253-262.

66. Rezanova V., Tsebrenko M. Influence of binary additives of compatibilizers on the micro- and macrorheological properties of melts of polypropylene-copolyamide mixtures. *J. of Eng. Phys. and Thermophys*. 2009. Vol.81, №4. P. 766-773.

67. Резанова Н.М., Плаван В.П., Дзюбенко Л.С., Сап'яненко О.О., Горбик П.П., Коршун А.В. Структуроутворення у компатибілізованих нанонаповнених розтопах поліпропілен/пластифікований полівініловий спирт. *Наносистеми, наноматеріали, нанотехнології*. 2018. т.16, №1. С.55- 70.

68. Rezanova N.M., Meĺnik I.A., Tsebrenko M.V., Korshun A.V. Preparation of Nano-Filled Polypropylene Microfibers. *Fibre Chem*. 2014. №46. P.21-27.

69. Rezanova, N.M., Budash, Yu.O., Plavan, V.P., Bessarabov, V.I. Formation of microfibrillar structure of polypropylene/copolyamide blends in the presence of nanoparticles of metal oxides. *Voprosy khimii i khimicheskoi tekhnologii*. 2021. № 1. P.71-78.

70. Резанова Н.М., Будаш Ю.О., Плаван В.П., Коршун А.В., Пристинський С. В. Регулювання стійкості рідких мікроструменів поліпропілену в матриці співполіаміду за рахунок нанодобавок. *Технології та інжиніринг.* 2021. №2. С.60-69.

71. Sangroniz L., Palacios J. K., Fernandez M., Eguiazabal J.I., Santamaria A., Muller A.J., *European Polym. J*. 2016. Vol.10, №83. P.537-549.

72. Azubuike L., Sundararaj U. Interface Strengthening of PS/aPA Polymer Blend Nanocomposites via In Situ Compatibilization: Enhancement of Electrical and Rheological Properties. *Materials*. 2021. Vol. 14, №17. P. 4813.

73. Yesil Y., Bhat G.S Structure and mechanical properties of polyethylene melt blown nonwovens. *Int. J. Sci. Tech*. 2016. Vol.28, №10. P.780-793.

74 . Jin K., Banerji D., Bates F.S., Ellison C.J. Mechanically robust and recyclable cross-linked fibers from melt blown anthracene-functionalized commodity polymers. *ACS Appl. Mater. Interfaces*. 2019. Vol.11, №13. P.12863-12870.

75. Ellison C.J., Phatak A., Giles D.W., Bates F.S. Melt blown nanofibers: Fiber diameter distributions and onset of fiber breakup. *Polymer*. 2007. Vol.48, № 11. P. 3306-3316.

76. Mei Y., Wang Z., Li X. Improving filtration performance of electrospun nanofiber Mats by a bimodal method. *J. Appl. Polym. Sci*. 2013**.** Vol.128, № 2. P. 1089-1099.

77. Luo C.J., Stoyanov S.D., Stride E., Pelan E., Edirisinghe M. Electrospinning versus fiber production methods: from specifics to technological convergence. *Chem. Soc. Rev*. 2012. Vol.41, №13. P.4708-4735.

78. Li H., Li Y., Wang W. Needleless melt-electrospinning of biodegradable poly(lactic acid) ultrafine fibers for removal of oil from water. *Polymer*. 2017. Vol. 9., № 2. P.3-12.

79. Erben J., Pilarova K., Sanetrnik F., Chvojka J., Jencova V., Blazkova L., Havlicek J., Novak O., Mikes P., Proseeka E., Lukas D., Kuzelova-Kostakova E. The combination of melt blown and electrospinning for bone Tissue Engineering. *Mater. Lett*. 2015. Vol.143. P.172-176.

80. Wei L., Yu H., Qin X. Experimental investigation of process parameters for the filtration property of nanofiber membrane fabricated by needleless electrospinning apparatus. *J of Ind. Textiles*, Jan. 2020**.**

81. Voznyak Yu, Morawiec J., Galeski A. Ductility of polylactide composites reinforced with poly (butylene succinate) nanofibers. *Composites Part A*. 2016. Vol.90. P.218-224.

82. Yousfi M., Dadouche T., Chomat D., Samuel C., Soulestin J., Lacrampe M.-F., Krawczak P. Development of nanofibrillar morphologies in poly(L-lactide)/poly(amide) blends: role of the matrix elasticity and identification of the critical shear rate for the nodular/fibrillar transition. *RSC Adv*. 2018. Vol.8. P. 22023-22041.

83. Vozniak I., Hosseinnezhad R., Morawiec J., Galeski A. Nanofibrillar green composites of polylactide/polyhydroxyalkanoate produced in situ due to shear induced crystallization. *Polymer*. 2019. Vol.11. P.1811-1825.

84. Kimble L.D., Bhattacharyya D., Fakirov S. Biodegradable microfibrillar polymer-polymer composites from poly(L-lactic acid)/poly(glycolic acid). *eXPRESS Polym. Let*. 2015. Vol.9, №3. P.300-307.

85. Hosseinnezhad R., Vozniak I., Morawiec J., Galeski A., Dutkiewicz S. In situ generation of sustainable PLA-based nanocomposites by shear induced crystallization of nanofibrillar inclusions. *RSC Adv*. 2019. Vol.9. P.30370-30392.

86. Doan V.A., Nobukava S., Yamaguchi M. Localization of nanofibers on polymer surface using interface transfer technique. *Composites Part B*. 2012. Vol. 43, №3. P.1218-1223.

87. Rezanova N.M., Rezanova V.G., Plavan V.P., Viltsaniuk O.O. The Influence of Nano-Additives on the Formation of Matrix-Fibrillar Structure in the Polymer Mixture Melts and on the properties of Complex threads. *Vlakna a Textil*. 2017. №2. P.37-42.

88. Doan V.A., Yamaguchi M. Interphase transfer of nanofillers and functional liquid between immiscible polymer pairs. *Recent Res. Devel. Mat. Sci*. 2013. № 10. P.59-88.

89. Manson J., Sperling H. Polymer blends and composites: monograph. New York: Plenum Press, 1976. 440 p.

90. Tskhe Y., Buzgo M., Simaite A. Electorospun nanofibers with photocatalytic particles fo carbon sorption. *Nanotech / Biotech*

France 2021 and joint virtual conferences: Book of abstracts International Conference, 23-25 June. 2021 P.77.

91. Shields R.J., Bhattacharyya D., Fakirov S. Fibrillar polymer-polymer composites: morphology, properties and application. *J. of Mater. Sci.* 2008. №43. P.6758-6770.

92. Filters and Filtration Handbook / ed. by Ch. Dickenson. monograph. Oxford: Elsevier Advanced Technology, 1992. 780 p.

93. Polymer Blends / ed. by D.R. Paul, C.B.Bucknall. monograph. New York: John Wiley & Sons Inc., 2000. Vol.1. 618 p.

94. Chandran N, Chandran S, Maria H.J., Thomas S. Compatibilizing action and localization of clay in a polypropylene/natural rubber (PP/NR) blend. *RSC Adv.* 2015. Vol.105, №5. P. 86265-86273.

95. Budash Y., Rezanova N., Plavan V., Rezanova V. Thermally and organomodified montmorillonite as effective regulators of the structure formation process in polypropylene/polystyrene blends. *Polym. and Polym. Composites.* 2022. Vol. 30. P. 1–8.

96. Handbook of Design and Analysis of Experiments. Edited by A. Dean, M. Morris, J. Stufken, D. Bingham // CRC Press Taylor & Francis Group, Boca Raton, 2015. – 946 p.

97. N. R. Draper, H. Smith Applied Regression Analysis. - John Wiley & Sons, 1998. - 736 p.

98. Thomas B. Barker, Andrew Milivojevich Quality by Experimental Design. 4th Edition // Chapman & Hall, 2016. – 754 p.

99. S. Dutta Optimization in Chemical Engineering 1st Edition // Cambridge University Press, 2016. – 380 p.

100. M. Bierlaire Optimization: Principles and Algorithms, Second edition // EPFL Press, 2018. - 738 p

101. Резанова В. Г. Резанова Н. М. Програмне забезпечення для дослідження полімерних систем : монографія. Київ: АртЕк, 2020. 358 с.

102. Резанова В.Г., Резанова Н.М., Куценко С.І. Свідоцтво про реєстрацію авторського права на твір № 107120. Комп’ютерна програма «АІПЕСС». Автоматизоване інтерактивне планування експерименту для сумішевих систем, дата реєстрації 09.08.2021.

103. Резанова В.Г., Резанова Н.М., Куценко С.І. Свідоцтво про реєстрацію авторського права на твір № 107121. Комп'ютерна програма «АІПЕССу». Автоматизоване інтерактивне планування експерименту для сумішевих систем – удосконалена, дата реєстрації 09.08.2021.

104. Stroustrup B. Programming: Principles and Practice Using C++: monograph. 2nd Edition: Addison-Wesley Professional, 2014. 1312 p.

105. Meyers S. Effective modern C ++: O'Reilly Media, 2014. 334 p.

106. Schildt H. C++. Basic course . M.: Williams: 2018, p. 624 p.

107. Stroustrup B. The C++ Programming Language Fourth Edition. Addison-Wesley, 2013. – 1366 p.

108. Васильєв О. Програмування на С++ в прикладах і задачах. К.: Ліра-К, 2020. – 382 с.

109. S. B. Lippman, J. Lajoie, E. Barbara My C++ Primer, Addison-Wesley: 2012. – 976 p.

110. B. Stroustrup Tour of C++ Addison-Wesley Professional , 2023. – 320 p.

111. Резанова В.Г. Перетворення задачі оптимізації при дослідженні чотирикомпонентних сумішей полімерів. *Вісник* КНУТД. 2016. №2. С.40-47.

112. Резанова В.Г. Оптимізація складу чотирикомпонентних сумішей полімерів із застосуванням методу штрафних функцій. Вісник КНУТД. 2016. №3. С.59-67.

113. Shcherban V. Yu., Rezanova V. G., Demkivska T. I. Programming of numerical methods and examples of practical application: monography. – Kyiv: Education of Ukraine, 2021. –150 p.

114. Резанова В.Г., Резанова Н.М. Програмне забезпечення для оптимізації складу багатокомпонентних сумішей. - К.:АртЕк. - 2022. 315 с.

ADDITION 1

PROGRAM LISTING

Basic procedures and functions for implementing interactive experiment planning for a ternary mixture

```
double minmax(double mas[], int len, bool findmax)
{
  double a;
  a = mas[0];
  for(int i = 0; i < len; i++)
  {
    if(findmax == false)
    {
      if(mas[i] < a) a = mas[i];
    }
    else
    {
      if(mas[i] > a) a = mas[i];
    }
  }
  return a;
}
void DrawGraph(TImage *image)
{
   x0 = floor(image->Width/2);
   y0 = floor(image->Height/2);
   image->Canvas->Pen->Color = clBlack;
   image->Canvas->Pen->Width = 2;

   image->Canvas->MoveTo(x0,0);
   image->Canvas->LineTo(x0,image->Height);
   image->Canvas->MoveTo(0,y0);
   image->Canvas->LineTo(image->Width, y0);
   image->Canvas->Pen->Color = clBlue;
   image->Canvas->Pen->Width = 1;
   for(int i = 1; i < 200; i++)
            {
              image->Canvas->MoveTo(x0+i*mashx,0);
              image->Canvas->LineTo(x0+i*mashx,image->Height);
              image->Canvas->MoveTo(0,y0+i*mashy);
              image->Canvas->LineTo(image->Width, y0+i*mashy);
            }
            for(int i = 1; i < 200; i++)
            {
              image->Canvas->MoveTo(x0-i*mashx,0);
              image->Canvas->LineTo(x0-i*mashx,image->Height);
              image->Canvas->MoveTo(0,y0-i*mashy);
              image->Canvas->LineTo(image->Width, y0-i*mashy);
            }
               image->Canvas->Pen->Color = clRed;
            image->Canvas->Pen->Width = 2;
            image->Canvas->MoveTo(x0-a/2*mashx,y0);      //      X1
image->Canvas->LineTo(x0+a/2*mashx,y0);  //  X3  (отложили  5
вправо)
            image->Canvas->LineTo(x0,y0-(sqrt(3)/2 * a)*mashy);
            image->Canvas->LineTo(x0-a/2*mashx,y0); // Замкнули
            //X1,X2,X3
            image->Canvas->TextOutA(x0-a/2*mashx-20,y0,"x1");
            image->Canvas->TextOutA(x0,y0-(sqrt(3)/2  *  a)*mashy-
15,"x2");
            image->Canvas->TextOutA(x0+a/2*mashx+15,y0,"x3");

            image->Canvas->Pen->Color = clBlack;
            image->Canvas->Pen->Width = 2;
         }
         void ClearGraph(TImage *image)
         {
            image->Canvas->Pen->Mode=pmCopy;
            image->Canvas->Pen->Color = clWhite;
            image->Canvas->MoveTo(0,0);
            image->Canvas->FillRect(Rect(0,0,image->Width, image-
>Height));
         }
         void __fastcall TForm1::ButtonDrawClick(TObject *Sender)
         {
            ClearGraph(Image1);
```

```
    DrawGraph(Image1);
           GroupBox1->Visible=true;
         Form1->ScrollBox1->HorzScrollBar->Position=1111;
      Form1->ScrollBox1->VertScrollBar->Position=961;
       Form1->ScrollBox1->Height/2;
}
double max(double x, double y)
{
    if (x < y) {
        return y;
    }
    return x;
}
double min(double x, double y)
{
    if (x > y) {
        return y;
    }
    return x;
}
bool thc(double x, double y, double z, double w, double a, double b)
{
    double k, c,res;
     bool flag=false;
    if (z == x) {
        return (a == x && b >= min(y, w) && x <= max(y, w));
    }
    k = (w - y) / (z - x);
    c = y - k * x;
     res= a * k + c;
       flag=floor(b*10000000) == floor(res*10000000);
    return flag;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
 ClearGraph(Image1);
    DrawGraph(Image1);
```

```
      Button4->Enabled=true;
        double kx1,ky1,kx2,ky2,kx3,ky3;
        kx1 =(double) StrToFloat(Form1->Edit1->Text);
        ky1 =(double) StrToFloat(Form1->Edit2->Text);
        kx2 = (double)StrToFloat(Form1->Edit3->Text);
        ky2 = (double)StrToFloat(Form1->Edit4->Text);
        kx3 =(double) StrToFloat(Form1->Edit5->Text);
        ky3 = (double)StrToFloat(Form1->Edit6->Text);
         float r=0.0001;
        float rr=0.01f;
        float rrr=0.001f;
        float rrrr=0.0001f;
        double t=0.0001;
        double tt=0.01;
        double ttt=0.001;
        double tttt=0.0001;
        float k;
        int l=r;
        //-------------------------------------
        DrawRegions(kx1,ky1,kx2,ky2,kx3,ky3);
//=========================================
       double eps=0.000;
       if(kx1==0)
        {
        kx1=kx1+eps;
        }
        if(kx2==0)
        {
        kx2=kx2+eps;
        }
        if(kx3==0)
        {
        kx3=kx3+eps;
        }
        if(kx1==1)
        {
        kx1=kx1-eps;
        }
```

```
if(kx2==1)
{
kx2=kx2-eps;
}
if(kx3==1)
{
kx3=kx3-eps;
}
float test;
test=100*kx1;
float test2=0.00002,test3;
test2=100*test2;
Kfx1 = kx1;
Kfy1 = ky1;
Kfx2 = kx2;
Kfy2 = ky2;
Kfx3 = kx3;
Kfy3 = ky3;
double ky;
double kx;
kx=((a/2)*mashx);
ky=(sqrt(3)/2 * a)*mashy;
// Регион X2:
double RegX2[5],RegY2[5];
RegX2[0] = x0-(kx*(1-ky2));
RegY2[0] = y0-(ky*(ky2));
RegX2[1] = x0+(kx*(1-ky2));
RegY2[1] = y0-(ky*(ky2));
RegX2[2] = x0+(kx*(1-kx2));
RegY2[2] = y0-(ky*(kx2));
RegX2[3] = x0-(kx*(1-kx2));
RegY2[3] = y0-(ky*(kx2));
RegX2[4] = RegX2[0]; RegY2[4] = RegY2[0];
// Регион X1:
double RegX1[5],RegY1[5];
RegX1[0] = x0-(kx*kx1);
RegY1[0] = y0-(ky*(1-kx1));
RegX1[1] = x0-(kx*ky1);
```

```
	RegY1[1] = y0-(ky*(1-ky1));
	if(ky1>0.50) RegX1[2] = x0-((kx/0.5)*fabs((0.5-ky1)));
	else if(ky1<0.51) RegX1[2] = x0+((kx/0.5)*(0.5-ky1));
	RegY1[2] = y0;
	if(kx1>0.5) RegX1[3] = x0-((kx/0.5)*fabs((0.5-kx1)));
	else if(kx1<0.51) RegX1[3] = x0+((kx/0.5)*(0.5-kx1));
	RegY1[3] = y0;
	RegX1[4] = RegX1[0]; RegY1[4] = RegY1[0];
	// Регион X3:
	double RegX3[5],RegY3[5];
	RegX3[0] = x0+(kx*(kx3));
	RegY3[0] = y0-(ky*(1-kx3));
	RegX3[1] = x0+(kx*(ky3));
	RegY3[1] = y0-(ky*(1-ky3));
	if(ky3>0.5) RegX3[2] = x0+((kx/0.5)*fabs(0.5-ky3));
	else  if(ky3<0.51)  RegX3[2]  =  x0-((kx/0.5)*fabs((0.5-
ky3)));

	RegY3[2] = y0;
	if(kx3>0.5) RegX3[3] = x0+((kx/0.5)*fabs(0.5-kx3));
	else  if(kx3<0.51)  RegX3[3]  =  x0-((kx/0.5)*fabs((0.5-
kx3)));

	RegY3[3] = y0;
	RegX3[4] = RegX3[0]; RegY3[4] = RegY3[0];
	//------------------------------------------------------
	// Малюємо точки і лінії регіонів X1,X2,X3:
	//DrawFigure(RegX2,RegY2,4,clBlack,clGreen);
	//DrawFigure(RegX1,RegY1,4,clBlack,clGreen);
	//DrawFigure(RegX3,RegY3,4,clBlack,clGreen);

//========================================
		double
peretinX1[10],peretinY1[10],peretinX2[10],peretinY2[10],
	peretinX3[10],peretinY3[10],
peretinX4[10],peretinY4[10];
	int ff1 = 0, ff2 = 0, ff3 = 0,ff4=0;
	NullMas(peretinX1,10);
```

```
                NullMas(peretinY1,10);
                NullMas(peretinX2,10);
                NullMas(peretinY2,10);
                NullMas(peretinX3,10);
                NullMas(peretinY3,10);
                  NullMas(peretinX4,10);
                NullMas(peretinY4,10);
                  bool Fl=false;
            CrossTwoPoligon2(5,RegX2,RegY2,5,RegX1,RegY1,ff1,per
etinX1,peretinY1);
CrossTwoPoligon2(5,RegX3,RegY3,5,RegX2,RegY2,ff2,peretinX2,
peretinY2);
            CrossTwoPoligon2(5,RegX1,RegY1,5,RegX3,RegY3,ff3,per
etinX3,peretinY3);
                 double SumaPeretX[30],SumaPeretY[30];
                  NullMas(SumaPeretX,30);
                 NullMas(SumaPeretY,30);
                 int countsuma=0;
                    for (int i=0;i<ff3;i++){
               bool
InFigure=thc(RegX2[0],RegY2[0],RegX2[1],RegY2[1],peretinX3[i],
peretinY3[i]);
               bool flag1=false;
               bool flag2=false;
               flag1=peretinX3[i]>=min(RegX2[0], RegX2[1]);
               flag2=peretinX3[i]<=max(RegX2[0], RegX2[1]);
               double test=0;
               test=max(RegX2[0],RegX2[1]);
               if(InFigure &&flag1  && flag2){
              //  Form1->Image1->Canvas->Pen->Color=clBlack;
               //            Form1->Image1->Canvas->Ellipse(peretinX3[i]-
5,peretinY3[i]-5,peretinX3[i]+5,peretinY3[i]+5);
                SumaPeretX[countsuma]=peretinX3[i];
                SumaPeretY[countsuma]=peretinY3[i];
                countsuma++;
                 peretinX3[i]=0;
                  peretinY3[i]=0;
                } }
                for (int i=0;i<ff3;i++){
               bool
InFigure=thc(RegX2[1],RegY2[1],RegX2[2],RegY2[2],peretinX3[i],
peretinY3[i]);
               if(InFigure  &&  peretinX3[i]>=min(RegX2[1],  RegX2[2])
&& peretinX3[i]<=max(RegX2[1], RegX2[2])){
               // Form1->Image1->Canvas->Pen->Color=clGreen;
              //              Form1->Image1->Canvas->Ellipse(peretinX3[i]-
5,peretinY3[i]-5,peretinX3[i]+5,peretinY3[i]+5);
               SumaPeretX[countsuma]=peretinX3[i];
                SumaPeretY[countsuma]=peretinY3[i];
                countsuma++;
                 peretinX3[i]=0;
                  peretinY3[i]=0;} }
                for (int i=0;i<ff3;i++){
               bool
InFigure=thc(RegX2[2],RegY2[2],RegX2[3],RegY2[3],peretinX3[i],
peretinY3[i]);
               if(InFigure  &&  peretinX3[i]>=min(RegX2[2],  RegX2[3])
&& peretinX3[i]<=max(RegX2[2], RegX2[3])){
              //  Form1->Image1->Canvas->Pen->Color=clBlue;
              //              Form1->Image1->Canvas->Ellipse(peretinX3[i]-
5,peretinY3[i]-5,peretinX3[i]+5,peretinY3[i]+5);
               SumaPeretX[countsuma]=peretinX3[i];
                SumaPeretY[countsuma]=peretinY3[i];
                countsuma++;
                peretinX3[i]=0;
                  peretinY3[i]=0; } }
                for (int i=0;i<ff3;i++){
               bool
InFigure=thc(RegX2[3],RegY2[3],RegX2[4],RegY2[4],peretinX3[i],
peretinY3[i]);
               if(InFigure  &&  peretinX3[i]>=min(RegX2[3],  RegX2[4])
&& peretinX3[i]<=max(RegX2[3], RegX2[4])){
               // Form1->Image1->Canvas->Pen->Color=clYellow;
              //              Form1->Image1->Canvas->Ellipse(peretinX3[i]-
5,peretinY3[i]-5,peretinX3[i]+5,peretinY3[i]+5);
               SumaPeretX[countsuma]=peretinX3[i];
```

```
                SumaPeretY[countsuma]=peretinY3[i];
                countsuma++;
                peretinX3[i]=0;
                  peretinY3[i]=0; } }
                //DrawFigure(peretinX3,peretinY3,ff3,clRed,clBlue);
               double vx[5],vy[5]; bool Fl2[5];
               int kk=0;
              for(int i=0;i<5;i++)    {
              Fl2[i]=0                ;}
                 for (int j=0;j<ff3;j++) {
              for (int i=0;i<4;i++){

Fl2[i]=PointCrossTwoLine2(RegX2[i],RegY2[i],RegX2[i+1],RegY2
[i+1],peretinX3[j],peretinY3[j],peretinX3[j],peretinY3[j]+10000,vx[
kk],vy[kk]);
              if (Fl2[i]){
              kk++;
              }
              }  if(kk==1){
                 // Form1->Image1->Canvas->Pen->Color=clWhite;
               SumaPeretX[countsuma]=peretinX3[j];
                SumaPeretY[countsuma]=peretinY3[j];
                countsuma++;
               //              Form1->Image1->Canvas->Ellipse(peretinX3[j]-
5,peretinY3[j]-5,peretinX3[j]+5,peretinY3[j]+5);
              }
              for(int d=0;d<5;d++)    {
              Fl2[d]=0               ;}
              kk=0;
               }
                for (int i=0;i<ff1;i++){
               bool
InFigure=thc(RegX3[0],RegY3[0],RegX3[1],RegY3[1],peretinX1[i],
peretinY1[i]);
               bool flag1=false;
               bool flag2=false;
               flag1=peretinX1[i]>=min(RegX3[0], RegX3[1]);
               flag2=peretinX1[i]<=max(RegX3[0], RegX3[1]);
```

```
            double test=0;
            test=max(RegX3[0],RegX3[1]);
            if(InFigure &&flag1  && flag2){
           //  Form1->Image1->Canvas->Pen->Color=clBlack;
           //            Form1->Image1->Canvas->Ellipse(peretinX1[i]-
5,peretinY1[i]-5,peretinX1[i]+5,peretinY1[i]+5);
             SumaPeretX[countsuma]=peretinX1[i];
             SumaPeretY[countsuma]=peretinY1[i];
             countsuma++;
              peretinX1[i]=0;
               peretinY1[i]=0;
             } }
             for (int i=0;i<ff1;i++){
            bool
InFigure=thc(RegX3[1],RegY3[1],RegX3[2],RegY3[2],peretinX1[i],
peretinY1[i]);
            if(InFigure  &&  peretinX1[i]>=min(RegX3[1],  RegX3[2])
&& peretinX1[i]<=max(RegX3[1], RegX3[2])){
            // Form1->Image1->Canvas->Pen->Color=clGreen;
            //           Form1->Image1->Canvas->Ellipse(peretinX1[i]-
5,peretinY1[i]-5,peretinX1[i]+5,peretinY1[i]+5);
            SumaPeretX[countsuma]=peretinX1[i];
             SumaPeretY[countsuma]=peretinY1[i];
             countsuma++;
              peretinX1[i]=0;
               peretinY1[i]=0;} }
             for (int i=0;i<ff1;i++){
            bool
InFigure=thc(RegX3[2],RegY3[2],RegX3[3],RegY3[3],peretinX1[i],
peretinY1[i]);
            if(InFigure  &&  peretinX1[i]>=min(RegX3[2],  RegX3[3])
&& peretinX1[i]<=max(RegX3[2], RegX3[3])){
           //  Form1->Image1->Canvas->Pen->Color=clBlue;
           //            Form1->Image1->Canvas->Ellipse(peretinX1[i]-
5,peretinY1[i]-5,peretinX1[i]+5,peretinY1[i]+5);
            SumaPeretX[countsuma]=peretinX1[i];
             SumaPeretY[countsuma]=peretinY1[i];
             countsuma++;
```

```
                peretinX1[i]=0;
                  peretinY1[i]=0; } }
                for (int i=0;i<ff1;i++){
               bool
InFigure=thc(RegX3[3],RegY3[3],RegX3[4],RegY3[4],peretinX1[i],
peretinY1[i]);
               if(InFigure  &&  peretinX1[i]>=min(RegX3[3],  RegX3[4])
&& peretinX1[i]<=max(RegX3[3], RegX3[4])){
              //  Form1->Image1->Canvas->Pen->Color=clYellow;
              //              Form1->Image1->Canvas->Ellipse(peretinX1[i]-
5,peretinY1[i]-5,peretinX1[i]+5,peretinY1[i]+5);
                SumaPeretX[countsuma]=peretinX1[i];
                SumaPeretY[countsuma]=peretinY1[i];
                countsuma++;
                peretinX1[i]=0;
                  peretinY1[i]=0; } }
                //DrawFigure(peretinX3,peretinY3,ff3,clRed,clBlue);
             /*  double vx[5],vy[5]; bool Fl2[5];
               int kk=0;       */

              for(int i=0;i<5;i++)    {
              Fl2[i]=0                ;}
                  for (int j=0;j<ff1;j++) {
              for (int i=0;i<4;i++){
            Fl2[i]=PointCrossTwoLine2(RegX3[i],RegY3[i],RegX3[i+1
],RegY3[i+1],peretinX1[j],peretinY1[j],peretinX1[j]-
10000,peretinY1[j],vx[kk],vy[kk]);
              if (Fl2[i]){

              kk++;
              }
              }  if(kk==1){
               //   Form1->Image1->Canvas->Pen->Color=clWhite;
                  SumaPeretX[countsuma]=peretinX1[j];
                SumaPeretY[countsuma]=peretinY1[j];
                countsuma++;
              //              Form1->Image1->Canvas->Ellipse(peretinX1[j]-
5,peretinY1[j]-5,peretinX1[j]+5,peretinY1[j]+5);
```

```
            }
            for(int d=0;d<5;d++)    {
            Fl2[d]=0              ;}
            kk=0;
             }
             //DrawFigure(peretinX1,peretinY1,ff1,clWhite,clRed);
             //
CrossTwoPoligon2(5,RegX3,RegY3,ff2,peretinX2,peretinY2,ff4,per
etinX4,peretinY4);
              // DrawFigure(peretinX4,peretinY4,ff4,clWhite,clRed);
            //  PointsPeretin(RegX1,RegY1,peretinX2, peretinY2, ff2);

           // DrawFigure(peretinX2,peretinY2,ff2,clRed,clRed);
               for (int i=0;i<ff2;i++){
             bool
InFigure=thc(RegX1[0],RegY1[0],RegX1[1],RegY1[1],peretinX2[i],
peretinY2[i]);
             if(InFigure  &&  peretinX2[i]>=min(RegX1[0],  RegX1[1])
&& peretinX2[i]<=max(RegX1[0], RegX1[1])){
             // Form1->Image1->Canvas->Pen->Color=clBlack;
             //             Form1->Image1->Canvas->Ellipse(peretinX2[i]-
5,peretinY2[i]-5,peretinX2[i]+5,peretinY2[i]+5);
               SumaPeretX[countsuma]=peretinX2[i];
              SumaPeretY[countsuma]=peretinY2[i];
              countsuma++;
               peretinX2[i]=0;
                peretinY2[i]=0;
              } }
              for (int i=0;i<ff2;i++){
             bool
InFigure=thc(RegX1[1],RegY1[1],RegX1[2],RegY1[2],peretinX2[i],
peretinY2[i]);
             if(InFigure  &&  peretinX2[i]>=min(RegX1[1],  RegX1[2])
&& peretinX2[i]<=max(RegX1[1], RegX1[2])){
            //  Form1->Image1->Canvas->Pen->Color=clGreen;
            //              Form1->Image1->Canvas->Ellipse(peretinX2[i]-
5,peretinY2[i]-5,peretinX2[i]+5,peretinY2[i]+5);
              SumaPeretX[countsuma]=peretinX2[i];
```

```
               SumaPeretY[countsuma]=peretinY2[i];
               countsuma++;
                peretinX2[i]=0;
                 peretinY2[i]=0;} }
               for (int i=0;i<ff2;i++){
              bool
InFigure=thc(RegX1[2],RegY1[2],RegX1[3],RegY1[3],peretinX2[i],
peretinY2[i]);
              if(InFigure  &&  peretinX2[i]>=min(RegX1[2],  RegX1[3])
&& peretinX2[i]<=max(RegX1[2], RegX1[3])){
             //  Form1->Image1->Canvas->Pen->Color=clBlue;
             //             Form1->Image1->Canvas->Ellipse(peretinX2[i]-
5,peretinY2[i]-5,peretinX2[i]+5,peretinY2[i]+5);
               SumaPeretX[countsuma]=peretinX2[i];
               SumaPeretY[countsuma]=peretinY2[i];
               countsuma++;
               peretinX2[i]=0;
                 peretinY2[i]=0; } }
               for (int i=0;i<ff2;i++){
              bool
InFigure=thc(RegX1[3],RegY1[3],RegX1[4],RegY1[4],peretinX2[i],
peretinY2[i]);
              if(InFigure  &&  peretinX2[i]>=min(RegX1[3],  RegX1[4])
&& peretinX2[i]<=max(RegX1[3], RegX1[4])){
              // Form1->Image1->Canvas->Pen->Color=clYellow;
             //             Form1->Image1->Canvas->Ellipse(peretinX2[i]-
5,peretinY2[i]-5,peretinX2[i]+5,peretinY2[i]+5);
              SumaPeretX[countsuma]=peretinX2[i];
               SumaPeretY[countsuma]=peretinY2[i];
               countsuma++;
               peretinX2[i]=0;
                 peretinY2[i]=0; } }
               //DrawFigure(peretinX3,peretinY3,ff3,clRed,clBlue);
             // double vx[5],vy[5]; bool Fl2[5];
            //  int kk=0;

             for(int i=0;i<5;i++)    {
             Fl2[i]=0               ;}
```

```
                for (int j=0;j<ff2;j++) {
            for (int i=0;i<4;i++){
          l2[i]=PointCrossTwoLine2(RegX1[i],RegY1[i],RegX1[i+1],
RegY1[i+1],peretinX2[j],peretinY2[j],peretinX2[j]+10000,peretinY2
[j],vx[kk],vy[kk]);
            if (Fl2[i]){

            kk++;
            }
            }  if(kk==1){
             //   Form1->Image1->Canvas->Pen->Color=clWhite;
                SumaPeretX[countsuma]=peretinX2[j];
              SumaPeretY[countsuma]=peretinY2[j];
              countsuma++;
             //              Form1->Image1->Canvas->Ellipse(peretinX2[j]-
5,peretinY2[j]-5,peretinX2[j]+5,peretinY2[j]+5);
            }
            for(int d=0;d<5;d++)    {
            Fl2[d]=0               ;}
            kk=0;
             }
              /*  for (int j=0;j<ff1;j++) {
            for (int i=0;i<4;i++){
          Fl2[i]=PointCrossTwoLine2(RegX3[i],RegY3[i],RegX3[i+1
],RegY3[i+1],peretinX1[j],peretinY1[j],peretinX1[j]+10000,peretinY
1[j],vx[kk],vy[kk]);
            if (Fl2[i]){
            kk++;
            }
            }
            if(kk<1){
              peretinX1[j]=0;
              peretinY1[j]=0;
            }
            kk=0;
            }   */
          CrossTwoPoligon2(5,RegX3,RegY3,ff2,peretinX2,peretinY
2,ff4,peretinX4,peretinY4);
```

```
        //=========================================
          /*    PointsPeretin(RegX3,RegY3,peretinX1,    peretinY1,
ff1);
           PointsPeretin(RegX1,RegY1,peretinX2, peretinY2, ff2);
           PointsPeretin(RegX2,RegY2,peretinX3, peretinY3, ff3);
          */
           double SumaPeretX2[30],SumaPeretY2[30];
            NullMas(SumaPeretX2,30);                        double
SumaPeretX3[30],SumaPeretY3[30];   NullMas(SumaPeretX3,30);
           NullMas(SumaPeretY2,30);
           NullMas(SumaPeretY3,30);
          /*  double SumaPeretX[30],SumaPeretY[30];
            NullMas(SumaPeretX,30);
           NullMas(SumaPeretY,30);
           double SumaPeretX2[30],SumaPeretY2[30];
           double SumaPeretX3[30],SumaPeretY3[30];
           int d=0;
           NullMas(SumaPeretX,30);
           NullMas(SumaPeretY,30);
            NullMas(SumaPeretX2,30);
           NullMas(SumaPeretY2,30);
             NullMas(SumaPeretX3,30);
           NullMas(SumaPeretY3,30);
           PointsUnite(peretinX1,peretinY1,peretinX2,peretinY2,
           peretinX3,peretinY3,SumaPeretX,SumaPeretY,d);  */
            NullMas(FX,10);
           NullMas(FY,10);
                FN=0;
               DelRepeatPoints(SumaPeretX,SumaPeretY,10);
                Path(SumaPeretX,SumaPeretY,countsuma);
        PointsUnite2(SumaPeretX,SumaPeretY,SumaPeretX2,Suma
PeretY2,FN);
            DelRepeatPoints(SumaPeretX2,SumaPeretY2,10);
           int FN2=0;
           for(int i=0; i<FN; i++){
           if (SumaPeretX2[i]!=0 && SumaPeretY2!=0)
           {
           SumaPeretX3[FN2]=SumaPeretX2[i];
```

```
               SumaPeretY3[FN2]=SumaPeretY2[i];
               FN2++;
               }  }
               FN=FN2;
               Path(SumaPeretX3,SumaPeretY3,FN);
                //
PointsUnite2(SumaPeretX2,SumaPeretY2,SumaPeretX3,SumaPeret
Y3,FN2);
          DrawFigure(SumaPeretX3,SumaPeretY3,FN2,clBlack,clBla
ck);
              if (FN2==3){
              double q,qq,qqq;
              int e,ee,eee;
              bool r=false,rr=false,rrr=false;
              q=(SumaPeretX3[0]);
              qq=(SumaPeretX3[1]);
              qqq=(SumaPeretX3[2]);
              e=(SumaPeretY3[0]*100000);
              ee=(SumaPeretY3[1]*100000);
              eee=(SumaPeretY3[2]*100000);
                 if (r && rr && rrr)
              {
              FN2=1;
              }
              }
               if (FN2==0)
               {
               Edit13->Color=clRed;
               Button4->Enabled=false;    Error2=false;
               Error1=true;
               Button11->Visible=true;
               }
               else if (FN2==1 || FN2==2)
               {
               Edit13->Color=clRed;
               Button4->Enabled=false;
               Error1=false;
               Error2=true;   Button11->Visible=true;
```

```
            } else
            {
             Edit13->Color=clGreen;
            Button4->Enabled=true;
            Error1=false;
            Error2=false;  Button11->Visible=false;
            }
            // FN = countsuma;
             for(int i = 0; i < FN2; i++)
             {
                FX[i] = SumaPeretX3[i];
                FY[i] = SumaPeretY3[i];
             }   }
           /* for (int i=0;i<d;i++){
            if (1390!=floor(SumaPeretY[i]))
            { SumaPeretY[i]=0;
            SumaPeretX[i]=0;}
            }   */
           /* for (int i=0;i<5;i++){
                    bool
InFigure=thc(RegX3[3],RegY3[3],RegX3[4],RegY3[4],SumaPeretX
[i],SumaPeretY[i]);
                /*   if(InFigure)  {
                    if  (((RegX3[0]<=  SumaPeretX[i])&&  (RegX3[1]<=
SumaPeretX[i]))||
                    ((RegX3[0]>=    SumaPeretX[i])&&    (RegX3[1]>=
SumaPeretX[i]))) {
                      SumaPeretX[i]=0;
                    }
                   } */
                /*   if (!InFigure) { SumaPeretX[i]=0;
                   SumaPeretY[i]=0;
                   } }  */
             /*        bool Fl,FL;
                   double Xv,Yv;
                   int counter=0,num=0,dd=0;
                   double MaxX3,MinX3,MaxY3,MinY3;
                   MaxX3=MaxMin(4,RegX3,1);
```

```
                    MaxY3=MaxMin(4,RegY3,1);
                    MinX3=MaxMin(4,RegX3,-1);
                    MinY3=MaxMin(4,RegY3,-1);
          Fl=PointCrossTwoLine2(RegX3[3],RegY3[3],RegX3[4],Re
gY3[4],SumaPeretX[2],SumaPeretY[2],SumaPeretX[2]+1000,Suma
PeretY[2],Xv,Yv);
               for (int i=0;i<d;i++) {
              FL=(MaxX3<SumaPeretX[i] || MinX3>SumaPeretX[i]) ||
(MaxY3<SumaPeretY[i] || MinY3>SumaPeretY[i]);
                  if (FL) { SumaPeretX[i]=0;
                    SumaPeretY[i]=0;
                    counter++;
                    }  }
                    num=d-counter;
                    counter=0;
                      for (int i=0;i<d;i++) {
              FL=RegX2[3]>SumaPeretX[i]                    ||
RegX2[2]<SumaPeretX[i]    ||    RegY2[0]>SumaPeretY[i]    ||
RegY2[3]<SumaPeretY[i];
                  if (FL) { SumaPeretX[i]=0;
                    SumaPeretY[i]=0;
                    counter++;
                    }  }
                    counter=0;
                    num=d-counter;
                     for (int i=0;i<d;i++) {
              FL=RegX1[1]>SumaPeretX[i]                    ||
RegX1[3]<SumaPeretX[i]    ||    RegY1[0]>SumaPeretY[i]    ||
RegY1[3]<SumaPeretY[i];
                  if (FL) { SumaPeretX[i]=0;
                    SumaPeretY[i]=0;
                    counter++;
                    }  }
                    num=d-counter;
                    for (int i=0;i<d;i++){
                    if (SumaPeretX[i]!=0 && SumaPeretY[i]!=0){
                      SumaPeretX2[dd]=SumaPeretX[i];
                      SumaPeretY2[dd]=SumaPeretY[i];
                  dd++;
                }
                }
                int mm=0;
               for (int i=0;i<4;i++) {
                 for (int j=0;j<dd;j++)
             {
         Fl=PointCrossTwoLine2(RegX3[i],RegY3[i],RegX3[i+1],Re
gY3[i+1],SumaPeretX2[j],SumaPeretY2[j],SumaPeretX2[j]+1000,S
umaPeretY2[j],Xv,Yv);
              bool
InFigure=thc(RegX3[i],RegY3[i],RegX3[i+1],RegY3[i+1],SumaPer
etX2[j],SumaPeretY2[j]);
              bool
InFigure2=thc(RegX1[i],RegY1[i],RegX1[i+1],RegY1[i+1],SumaPe
retX2[j],SumaPeretY2[j]);
             if(InFigure==true || InFigure2==true)
             {
                SumaPeretX3[mm]=SumaPeretX2[j];
              SumaPeretY3[mm]=SumaPeretY2[j];
               mm++;
             }
             /*   if (Fl && !InFigure)
               {
               SumaPeretX2[j]=0;
               SumaPeretY2[j]=0;
                //mm++;
               }    */
              //}

             //   }
             //    Path(SumaPeretX2,SumaPeretY2,dd);
           // if (RegX3[1]<SumaPeretX[14] &&
         //---------------------------------------------------------------------
         void   DrawRegions(double   kx1,   double   ky1,double
kx2,double ky2, double kx3, double ky3)
         {
           double eps = 100; // Кількість частин
```

```
          double kxx1,kyy1,kxx3,kyy3;
         /*  kxx1 = 1 - kx1;
          kyy1 = 1 - ky1;
          kxx3 = 1 - kx3;
          kyy3 = 1 - ky3;  */
          //-----------------
          kx1=(kx1*eps);
          ky1=(ky1*eps);
          kx2=(kx2*eps)+1;
          ky2=(ky2*eps)+1;
          kx3=(kx3*eps);
          ky3=(ky3*eps);
          double ky;
          double kx;
          kx=((a/2)*mashx);
          ky=(sqrt(3)/2 * a)*mashy;
          //-----------------------
          // Для X1,X2,X3:
          //-----------------------------------------------------
          // X2:
          Form1->Image1->Canvas->Pen->Color = clRed;
          for(double i=kx2;i<=ky2;i++) {
            Form1->Image1->Canvas->MoveTo(x0-
(kx/eps)*(eps+1-i),y0-((ky/eps)*(i-1)));
            Form1->Image1->Canvas-
>LineTo(x0+(kx/eps)*(eps+1-i),y0-((ky/eps)*(i-1)));
          }
          //X1:
          Form1->Image1->Canvas->Pen->Color = clGray;
          for(double i=kx1;i<=(ky1);i++) {
           if(kx1<(eps/2+2))
           {
            Form1->Image1->Canvas->MoveTo(x0-(kx/eps)*(i),y0-
((ky/eps)*(eps-i)));
            Form1->Image1->Canvas-
>LineTo(x0+((kx/(eps/2))*((eps/2+1)-i-1)),y0);
           }
           if(kx1>(eps/2+1))
```

```
            {
             Form1->Image1->Canvas->MoveTo(x0-(kx/eps)*(i),y0-
((ky/eps)*(eps-i)));
             Form1->Image1->Canvas->LineTo(x0-
((kx/(eps/2))*fabs(((eps/2+1)-i-1))),y0);
            }
           }
           //X3:
           Form1->Image1->Canvas->Pen->Color = clGreen;
           for(double i=kx3;i<=(ky3);i++) {
            if(kx3<(eps/2+2))
            {
             Form1->Image1->Canvas->MoveTo(x0-(kx/eps)*(-i),y0-
((ky/eps)*(eps-i)));
             Form1->Image1->Canvas->LineTo(x0-
((kx/(eps/2))*((eps/2+1)-i-1)),y0);
            }
            if(kx3>(eps/2+1))
            {
            Form1->Image1->Canvas->MoveTo(x0-(kx/eps)*(-i),y0-
((ky/eps)*(eps-i)));
            Form1->Image1->Canvas-
>LineTo(x0+((kx/(eps/2))*fabs(((eps/2+1)-i-1))),y0);
            }
           }
          //--------------------------------------------------------
        }
        bool PointCrossTwoLine(float Xa,float Ya,float Xb,float
Yb,float Xc,float Yc,float Xd,float Yd,float &X0,float &Y0)
        {
             float
A1,B1,C1,A2,B2,C2,Ra,Rb,Rd,Rc,Rab,Rcd,D1,D2,D0,x0,y0;
             int i;
             bool B;
             B=False;
             A1=Yb-Ya;
             B1=Xa-Xb;
             C1=Ya*Xb-Xa*Yb;
             A2=Yd-Yc;
             B2=Xc-Xd;
             C2=Yc*Xd-Xc*Yd;
             Ra=A2*Xa+B2*Ya+C2;
             Rb=A2*Xb+B2*Yb+C2;
             Rc=A1*Xc+B1*Yc+C1;
             Rd=A1*Xd+B1*Yd+C1;
             Rab=Ra*Rb;
             Rcd=Rc*Rd;
             if (Ra==0 && Rcd<0)
             {X0=Xa;Y0=Ya;B=true;}
              else if (Rb==0 && Rcd<0)
              {X0=Xb;Y0=Yb;B=true;}
               else if (Rc==0 && Rab<0)
               {X0=Xc;Y0=Yc;B=true;}
                else if (Rd==0 && Rab<0)
                {X0=Xd;Y0=Yd;B=true;}
                 else if (Ra==0 && Rc==0 && Rb!=0 &&
Rd!=0)
                 {X0=Xc;Y0=Yc;B=true;}
                  else if (Rb==0 && Rc==0 && Ra!=0 &&
Rd!=0)
                  {X0=Xc;Y0=Yc;B=true;}
                   else if (Ra==0 && Rd==0 && Rb!=0 &&
Rc!=0)
                   {X0=Xd;Y0=Yd;B=true;}
                    else if (Rb==0 && Rd==0 && Ra!=0 &&
Rc!=0)
                    {X0=Xd;Y0=Yd;B=true;}
             else
             {
             B=((Rcd<0)&&(Rab<0));
             if (B)
             {
             D0=A1*B2-A2*B1;
             D1=C2*B1-C1*B2;
             D2=C1*A2-C2*A1;
             X0=D1/D0;
```

```
            Y0=D2/D0;
            }
            }
            return B;
        }
        bool PointCrossTwoLine2(double Xa,double Ya,double
Xb,double Yb,double Xc,double Yc,double Xd,double Yd,double
&X0,double &Y0)
        {
            double
A1,B1,C1,A2,B2,C2,Ra,Rb,Rd,Rc,Rab,Rcd,D1,D2,D0,x0,y0;
            int i;
            bool B;
            B=False;
            A1=Yb-Ya;
            B1=Xa-Xb;
            C1=Ya*Xb-Xa*Yb;
            A2=Yd-Yc;
            B2=Xc-Xd;
            C2=Yc*Xd-Xc*Yd;
            Ra=A2*Xa+B2*Ya+C2;
            Rb=A2*Xb+B2*Yb+C2;
            Rc=A1*Xc+B1*Yc+C1;
            Rd=A1*Xd+B1*Yd+C1;
            Ra=floor(Ra);
            Rb=floor(Rb);
            Rc=floor(Rc);
            Rd=floor(Rd);
            Rab=Ra*Rb;
            Rcd=Rc*Rd;
            Rab=floor(Rab);
            Rcd=floor(Rcd);
            if(Ra == -1) Ra = 0;
            if(Rb == -1) Rb = 0;
            if(Rc == -1) Rc = 0;
            if(Rd == -1) Rd = 0;
            if(Rab == -1) Rab = 0;
            if(Rcd == -1) Rcd = 0;
            if (Ra==0 && Rcd<0)
            {X0=Xa;Y0=Ya;B=true;}
              else if (Rb==0 && Rcd<0)
              {X0=Xb;Y0=Yb;B=true;}
                else if (Rc==0 && Rab<0)
                {X0=Xc;Y0=Yc;B=true;}
                  else if (Rd==0 && Rab<0)
                  {X0=Xd;Y0=Yd;B=true;}
                    else if (Ra==0 && Rc==0 && Rb!=0 &&
Rd!=0)
                    {X0=Xc;Y0=Yc;B=true;}
                      else if (Rb==0 && Rc==0 && Ra!=0 &&
Rd!=0)
                      {X0=Xc;Y0=Yc;B=true;}
                        else if (Ra==0 && Rd==0 && Rb!=0 &&
Rc!=0)
                        {X0=Xd;Y0=Yd;B=true;}
                          else if (Rb==0 && Rd==0 && Ra!=0 &&
Rc!=0)
                          {X0=Xd;Y0=Yd;B=true;}
            else
            {
            B=((Rcd<0)&&(Rab<0));
            if (B)
            {
            D0=A1*B2-A2*B1;
            D1=C2*B1-C1*B2;
            D2=C1*A2-C2*A1;
            X0=D1/D0;
            Y0=D2/D0;
            }
            }
            return B;
        }
        bool cross(double Xa,double Ya,double Xb,double
Yb,double Xc,double Yc,double Xd,double Yd,double &X0,double
&Y0)
        {
            double
A1,B1,C1,A2,B2,C2,Ra,Rb,Rd,Rc,Rab,Rcd,D1,D2,D0,x0,y0;
            int i;
            bool B;
            B=False;
            A1=Ya-Yb;
            B1=Xb-Xa;
            C1=Xa*Yb-Xb*Ya;
            A2=Yc-Yd;
            B2=Xd-Xc;
            C2=Xc*Yd-Xd*Yc;
            Ra=A2*Xa+B2*Ya+C2;
            Rb=A2*Xb+B2*Yb+C2;
            Rc=A1*Xc+B1*Yc+C1;
            Rd=A1*Xd+B1*Yd+C1;


            D0=A1*B2-A2*B1;
            D1=C2*B1-C1*B2;
            D2=C1*A2-C2*A1;
            X0=D1/D0;
            Y0=D2/D0;
            return B;
        }
        bool CrossTwoPoligon (int KilksPointPol1,float
XPol1[],float YPol1[],
        int KilksPointPol2,float XPol2[],float YPol2[],int &Vt,float
Xv0[],float Yv0[])
        {
         bool Fl=false;
         float Xv,Yv;
         for (int k=0;k<KilksPointPol1-1;k++)
            for (int m=0;m<KilksPointPol2-1;m++)
            {
        Fl=PointCrossTwoLine(XPol1[m],YPol1[m],XPol1[m+1],Y
Pol1[m+1],XPol2[k],YPol2[k],XPol2[k+1],YPol2[k+1],Xv,Yv);
                  if (Fl)
            { if (Vt==0)
            {  Xv0[Vt] = Xv;
               Yv0[Vt] = Yv;
               Vt++;
            }
             if(Xv!=Xv0[Vt-1]||Yv!=Yv0[Vt-1])
              { Xv0[Vt] = Xv;
               Yv0[Vt] = Yv;
               Vt++;
              }
            }
           }if (Vt>=5)
           Fl=true;
           return Fl;
       }
       bool     CrossTwoPoligon2     (int     KilksPointPol1,double
XPol1[],double YPol1[],
       int    KilksPointPol2,double    XPol2[],double    YPol2[],int
&Vt,double Xv0[],double Yv0[])
       {
        bool Fl=false;
        double Xv,Yv;
        for (int k=0;k<KilksPointPol1-1;k++)
           for (int m=0;m<KilksPointPol2-1;m++)
           {
       Fl=PointCrossTwoLine2(XPol1[m],YPol1[m],XPol1[m+1],
YPol1[m+1],XPol2[k],YPol2[k],XPol2[k+1],YPol2[k+1],Xv,Yv);
              if (Fl)
            { if (Vt==0)
            {  Xv0[Vt] = Xv;
               Yv0[Vt] = Yv;
               Vt++;
            }
             if(Xv!=Xv0[Vt-1]||Yv!=Yv0[Vt-1])
              { Xv0[Vt] = Xv;
               Yv0[Vt] = Yv;
               Vt++;
              }
            }
```

```
      }if (Vt>=5)
      Fl=true;
      return Fl;
}
void NullMas(double mas[], int n)
{
  for(int i = 0; i < n; i++)
  {
    mas[i] = 0;
  }
}
void DelRepeatPoints(double masX[],double masY[], int n)
{
  for(int i = 0; i < n; i++)
  {
    for(int j = i+1; j < n; j++)
    {
       if((masX[i] == masX[j]) && (masY[i] == masY[j]))
       {
         masX[j] = 0;
         masY[j] = 0;
       }
    }
  }
  for(int i = 0; i < n-1; i++)
  {
     if(masX[i] == 0 && masY[i] == 0)
     {
       if(masX[i+1] != 0 && masY[i+1] != 0)
       {
        masX[i] = masX[i+1];
        masY[i] = masY[i+1];
        masX[i+1] = 0;
        masY[i+1] = 0;
       }
     }
  }
}
```

```
          void DrawFigure(double pointX[],double pointY[],int n,
          TColor colline,TColor colpoint)
          {
             Form1->Image1->Canvas-
>MoveTo(pointX[0],pointY[0]);
             for(int i = 0; i < n; i++)
             {
               Form1->Image1->Canvas->Pen->Color = colpoint;
               Form1->Image1->Canvas->Ellipse(pointX[i]-
5,pointY[i]-5,pointX[i]+5,pointY[i]+5);
               //Form1->Image1->Canvas->Ellipse(pointX[i]-
(5+i)*2,pointY[i]-(5+i)*2,pointX[i]+(5+i)*2,pointY[i]+(5+i)*2);
               Form1->Image1->Canvas->Pen->Color = colline;
               Form1->Image1->Canvas-
>LineTo(pointX[i],pointY[i]);
             }
             Form1->Image1->Canvas->LineTo(pointX[0],pointY[0]);
          }
          void Path(double X[], double Y[], int n)
          {
             double Ycdet2;
             double Otbor1[10],Otbor2[10],Otbor1Y[10],Otbor2Y[10];
             NullMas(Otbor1,10);
             NullMas(Otbor2,10);
             NullMas(Otbor1Y,10);
             NullMas(Otbor2Y,10);
             ParamDet3(n,X,Y,Ycdet2);
             int g1=0,g2=0;
             for(int i=0;i<n;i++)
             {
              if (Ycdet2>=Y[i])
              {
                Otbor1[g1]=X[i];
                Otbor1Y[g1]=Y[i];
                g1++;
              }
              else
              {
```

```
    Otbor2[g2]=X[i];
    Otbor2Y[g2]=Y[i];
    g2++;
  }
}
double tempG=0;
double tempGG=0;
int fl=0;
while (true)
{
   fl = 1;
   for(int i = 0; i < g1 - 1; i++)
   {
       if ((Otbor1[i] < Otbor1[i+1]) && Otbor1[i]!=0)
       {
                   double temp = Otbor1[i];
                   Otbor1[i] = Otbor1[i + 1];
                   Otbor1[i + 1] = temp;
                   double temp2 = Otbor1Y[i];
                   Otbor1Y[i] = Otbor1Y[i + 1];
                   Otbor1Y[i + 1] = temp2;
                   fl = 0;
        }
    }
    if (fl == 1) break;
}
fl=0;
while (true)
{
    fl = 1;
    for(int i = 0; i < g2 - 1; i++)
    {
        if(Otbor2[i] < Otbor2[i+1] && Otbor2[i]!=0)
        {
           double temp = Otbor2[i];
           Otbor2[i] = Otbor2[i + 1];
           Otbor2[i + 1] = temp;
           double temp2 = Otbor2Y[i];
```

```
                    Otbor2Y[i] = Otbor2Y[i + 1];
                    Otbor2Y[i + 1] = temp2;
                    fl = 0;
                  }
               }
               if (fl == 1) break;
             }
             for(int i=0;i<g1;i++)
             {
                 X[i]=Otbor1[i];
                 Y[i]=Otbor1Y[i];
             }
             for(int i=g1,j=g2-1;i<g1+g2;i++,j--)
             {
                 X[i]=Otbor2[j];
                 Y[i]=Otbor2Y[j];
             }
           }
           void PointsUnite(double peretinX1[],double
peretinY1[],double peretinX2[],double peretinY2[],
           double peretinX3[],double peretinY3[],double
SumaPeretX[],double SumaPeretY[], int &d)
           {
             for(int i=0;i<10;i++)
             {
              if(peretinX3[i]!=0 && peretinY3[i]!=0)
              {
               SumaPeretX[d]=peretinX3[i];
               SumaPeretY[d]=peretinY3[i];
               d++;
              }
             }
             for(int i=0;i<10;i++)
             {
              if(peretinX2[i]!=0 && peretinY2[i]!=0)
              {
               SumaPeretX[d]=peretinX2[i];
               SumaPeretY[d]=peretinY2[i];
```

```
            d++;
           }
          }
          for(int i=0;i<10;i++)
          {
           if(peretinX1[i]!=0 && peretinY1[i]!=0)
           {
            SumaPeretX[d]=peretinX1[i];
            SumaPeretY[d]=peretinY1[i];
            d++;
           }
          }
         }
         void PointsUnite2(double peretinX3[],double
peretinY3[],double SumaPeretX[],double SumaPeretY[], int &d)
         {
          for(int i=0;i<30;i++)
          {
           if(peretinX3[i]!=0 && peretinY3[i]!=0)
           {
            SumaPeretX[d]=peretinX3[i];
            SumaPeretY[d]=peretinY3[i];
            d++;
           }
          }
         }
         void PointsPeretin(double RegX1[],double RegY1[], double
peretinX2[], double peretinY2[], int ff2)
         {
           bool fak=false;
           int kl=0;
           double ho2=0,hoho2=0;
           for(int j=0;j<ff2;j++){
           for(int i=0;i<4;i++)
           {
fak=PointCrossTwoLine2(RegX1[i],RegY1[i],RegX1[i+1],RegY1[i
+1],peretinX2[j],peretinY2[j],peretinX2[j],peretinY2[j]+5000,ho2,ho
ho2);
```

```
      if (fak==true)
      {kl++;  }
    if (kl>1){
    fak =false;
    kl=0;
    peretinX2[j]=0;
     peretinY2[j]=0;
    break;
     }
   }
   if(kl==0)
   {
     peretinX2[j]=0;
     peretinY2[j]=0;
   }
   if(kl==1)
    {
    kl=0;
    }
   } kl=0;
}
void __fastcall TForm1::Button2Click(TObject *Sender)
{
  /* Form1->PageControl1->ActivePage=TabSheet2;
   ClearGraph(Image2);
   //DrawGraph(Image2);
      Form1->ScrollBox2->Width/2;
      Form1->ScrollBox2->Height/2;
   double XcE,YcE,XcIm3,YcIm3;
   XcE=Image2->Width/2;
   YcE=Image2->Height/2;
   KilksPointDet = 0;
   for(int i = 0; i < 10; i++)
   {
     selectedpoints[i] = false;
   }
   ParamDet();
   ParamModeli();
```

```
             BuildIm2(XcE,YcE);  */
             Form2->Close();
          }
          //-------------------------------------------------------------------
void ParamDet()
          {
              MaxX=MaxMin(FN,FX,1);
              MaxY=MaxMin(FN,FY,1);
              MinX=MaxMin(FN,FX,-1);
              MinY=MaxMin(FN,FY,-1);
              DlDet=MaxX-MinX;
              ShDet=MaxY-MinY;
              XcDet=(MaxX+MinX)/2;
              YcDet=(MaxY+MinY)/2;
          }
          void ParamDet2(int KilkT, double X2[], double Y2[], double
&DlDet2, double &ShDet2)
          {
              double MaxX2=MaxMin(KilkT,X2,1);
              double MaxY2=MaxMin(KilkT,Y2,1);
              double MinX2=MaxMin(KilkT,X2,-1);
              double MinY2=MaxMin(KilkT,Y2,-1);
              DlDet2=MaxX2-MinX2;
              ShDet2=MaxY2-MinY2;
          }
          void ParamDet3(int KilkT, double X2[], double Y2[],double
&YcDet)
          {
              double MaxY2=MaxMin(KilkT,Y2,1);
              double MinY2=MaxMin(KilkT,Y2,-1);
              YcDet=(MaxY2+MinY2)/2;
          }
          void ParamDet4(int KilkT, double X2[], double Y2[],double
&YcDet,double &XcDet,double &DL,double &Sh)
          {
              double MaxY2=MaxMin(KilkT,Y2,1);
              double MinY2=MaxMin(KilkT,Y2,-1);
              double MaxX2=MaxMin(KilkT,X2,1);
```

```
      double MinX2=MaxMin(KilkT,X2,-1);
      YcDet=(MaxY2+MinY2)/2;
      XcDet=(MaxX2+MinX2)/2;
      DL=MaxX2-MinX2;
      Sh=MaxY2-MinY2;
  }
  double MaxMin(int n,double Z[],int p)
  {
      int i;
      double q;
      q=Z[0];
      for (i=1;i<n;i++)
         if (p*q<p*Z[i]) q=Z[i];
        return q;
  }
  void ParamModeli()
  {
    double Xmax,Ymax,Xmin,Ymin;
    Xmax=MaxX; Xmin=MinX;
    Ymax=MaxY; Ymin=MinY;
    DlMod=Xmax-Xmin;
    ShMod=Ymax-Ymin;
    XcMod=(Xmax+Xmin)/2;
    YcMod=(Ymax+Ymin)/2;
  }
  void BuildIm2(double XcE,double YcE)
  {
     int i,j;
      mx=(Form1->ScrollBox2->Width)/DlMod;
      my=(Form1->ScrollBox2->Height)/ShMod;
      mxyIm2=mx;
      if (my<mx)mxyIm2=my;

      GraphIm2(FN,FX,FY,XcMod,YcMod,      XcE,      YcE,
mxyIm2, 0, 2);

       // for (j=0;j<FN; j++)
```

```
            //Elipse(FX[j],FY[j],2,XcMod,YcMod,    XcE,    YcE,
mxyIm2);
        }
        void BuildIm3(double XcE,double YcE)
        {
           int i,j;
            mx=(Form1->ScrollBox2->Width)/DlMod;
            my=(Form1->ScrollBox2->Height)/ShMod;
            mxyIm2=mx;
            if (my<mx)mxyIm2=my;

            GraphIm3(FN,FX,FY,XcMod,YcMod,       XcE,       YcE,
mxyIm2, 0, 2);

           // for (j=0;j<FN; j++)
            //Elipse(FX[j],FY[j],2,XcMod,YcMod,    XcE,    YcE,
mxyIm2);
        }
        void GraphIm2(int n, double X[], double Y[], double Xcf,
double Ycf,
                double Xce, double Yce, double mxy,int q, int p)
        {
             int j;
              ky = 1;
              kx = 1;
              double Xr[300],Yr[300];
              NullMas(Xr,300);
              NullMas(Yr,300);
        //========================================
              double** figura;
              double** matrObert;
              figura = new double*[n];
              matrObert = new double*[n];
              for(int i = 0; i < n; i++)
              {
                 figura[i] = new double[3];
                 matrObert[i] = new double[3];
              }
```

```
 for(int i = 0; i < n; i++)
 {
     figura[i][0] = X[i];
     figura[i][1] = Y[i];
     figura[i][2] = 1;
 }
  double DlDet3, ShDet3,mashtY3,mashtX3 ;
 ParamDet2(n, X, Y, DlDet3, ShDet3);
 int kut=60;
 mashtY3 = DlDet3/ShDet3;
 mashtX3 = ShDet3/DlDet3;

   /* for(int i = 0; i < n; i++)
 {
     figura[i][0] = Xr[i];
     figura[i][1] = Yr[i];
     figura[i][2] = 1;
 }  */
 double alpha = 120 * M_PI / 180; // В радіанах.
 matrObert = OberD(alpha,Xcf,Ycf);
 figura = MultipleMatrix(figura, matrObert, n);

    for(int i = 0; i < n; i++)
 {
    X[i] = figura[i][0];
    Y[i] = figura[i][1];
 }  ParamDet2(n, X, Y, DlDet3, ShDet3);
 mashtY3 = DlDet3/ShDet3;
 mashtX3 = ShDet3/DlDet3;
 if (mashtY3 > 20)
{
ky = int(mashtY3)/3;
    // ky=5;
}
 for(j=0;j<n;j++)
  {
   Xr[j]=(X[j]-Xcf)*mxy/1.2*kx+Xce;
   Yr[j]=(Y[j]-Ycf)*mxy/1.2*ky+Yce;
```

```
    }
/*   for(int i = 0; i < n; i++)
   {
      Xr[i] = figura[i][0];
      Yr[i] = figura[i][1];
   }  */
   /*double DlDet2, ShDet2;
   ParamDet2(n, X, Y, DlDet2, ShDet2);
   if (ShDet2==0) {
     ShDet2=10;
   }
   mashtY = DlDet2/ShDet2;
   mashtX = ShDet2/DlDet2;
   if (floor(mashtX)==floor(mashtX3))
   {
   vxod=true;
   alpha = kut * M_PI / 180; // В радіанах.
   matrObert = OberD(alpha,Xcf,Ycf);
   figura = MultipleMatrix(figura, matrObert, n);
   for(int i = 0; i < n; i++)
   {
      X[i] = figura[i][0];
      Y[i] = figura[i][1];
   }
   double DlDet2, ShDet2;
   ParamDet2(n, X, Y, DlDet2, ShDet2);
   mashtY = DlDet2/ShDet2;
   mashtX = ShDet2/DlDet2;
   }
   if(mashtY > 20)
  {
  ky = int(mashtY)/3;
     // ky=5;
  }
   if(mashtX > 20) {kx = 20;
                 }
   for(j=0;j<n;j++)
    {
```

```
                 Xr[j]=(X[j]-Xcf)*mxy/1.2*kx+Xce;
                 Yr[j]=(Y[j]-Ycf)*mxy/1.2*ky+Yce;
                }

//=========================================
                for(int i = 0; i < n; i++)
                {
                    figura[i][0] = Xr[i];
                    figura[i][1] = Yr[i];
                    figura[i][2] = 1;
                }   double alpha2 ;
                if (vxod){
                 alpha2 = -(2*kut) * M_PI / 180; // В радіанах.
                 }
                else
                {   alpha2 = -(kut) * M_PI / 180; // В радіанах.
                }
                matrObert = OberD(alpha2,Xcf,Ycf);

                figura = MultipleMatrix(figura, matrObert, n);
                for(int i = 0; i < n; i++)
                {
                  Xr[i] = figura[i][0];
                  Yr[i] = figura[i][1];
                }
//=========================================
                 for(j=0;j<n;j++)
                 {
                  Xr2[j]=Xr[j];
                  Yr2[j]=Yr[j];
                 }
               Form1->Image2->Canvas->Pen->Width=p;
               Form1->Image2->Canvas->Pen->Mode=pmCopy;
                switch(q)
                 {
                  case                    1:Form1->Image2->Canvas->Pen-
>Color=clRed;break;
```

```
                case                    2:Form1->Image2->Canvas->Pen-
>Color=clBlue;break;
                case                    3:Form1->Image2->Canvas->Pen-
>Color=clGreen;break;
                case                    4:Form1->Image2->Canvas->Pen-
>Color=clGray;break;
                default:Form1->Image2->Canvas->Pen-
>Color=clBlack;
               }
              for(j=0;j<n;j++)
              {
               if                    (j==0)Form1->Image2->Canvas-
>MoveTo(Xr[j],Yr[j]);
               else Form1->Image2->Canvas->LineTo(Xr[j],Yr[j]);
              }
              Form1->Image2->Canvas->LineTo(Xr[0],Yr[0]);

              for (j=0;j<FN; j++)
              Form1->Image2->Canvas->Ellipse(Xr2[j]-5,Yr2[j]-
5,Xr2[j]+5,Yr2[j]+5);
         }
         double angle( int x1, int y1, int x2, int y2)
         {
            return                                              acos(
(x1*x2+y1*y2)/(sqrt((double)x1*x1+y1*y1)*sqrt((double)x2*x2+y2
*y2)));
         }
         void GraphIm3(int n, double X[], double Y[], double Xcf,
double Ycf,
                   double Xce, double Yce, double mxy,int q, int p)
         {
             int j;
              ky = 1;
              kx = 1;
              double Xr[300],Yr[300];
              NullMas(Xr,300);
              NullMas(Yr,300);
```

```
//==========================================================
            double** figura;
            double** matrObert;
            figura = new double*[n];
            matrObert = new double*[n];

            for(int i = 0; i < n; i++)
            {
               figura[i] = new double[3];
               matrObert[i] = new double[3];
            }
            for(int i = 0; i < n; i++)
            {
                figura[i][0] = X[i];
                figura[i][1] = Y[i];
                figura[i][2] = 1;
            }
            bool ok=false;
            int kil=0;
            double DlDet3, ShDet3,mashtY3,mashtX3 ;
            int kut=60;
             double alpha;
            while (!ok)
             {
                 ParamDet2(n, X, Y, DlDet3, ShDet3);
             mashtY3 = DlDet3/ShDet3;
            mashtX3 = ShDet3/DlDet3;
            if (mashtY3>5)
            {
            ky=int(mashtY3)/3;
            ok=true;
            break;
            }
            if (kil==3){
            ok=true;
            break;
            }
            kil++;
             alpha = kut * M_PI / 180; // В радіанах.
            matrObert = OberD(alpha,Xcf,Ycf);
             figura = MultipleMatrix(figura, matrObert, n);
               for(int i = 0; i < n; i++)
            {
               X[i] = figura[i][0];
               Y[i] = figura[i][1];
            }
            }
            for(j=0;j<n;j++)
             {
               Xr[j]=(X[j]-Xcf)*mxy/1.2*kx+Xce;
               Yr[j]=(Y[j]-Ycf)*mxy/1.2*ky+Yce;
             }
                 for(int i = 0; i < n; i++)
            {
                figura[i][0] = Xr[i];
                figura[i][1] = Yr[i];
                figura[i][2] = 1;
            }
             double alpha2=0;
             for(int i=0;i<kil;i++){

            alpha2 =- kut* M_PI / 180;
             matrObert = OberD(alpha2,Xcf,Ycf);

            figura    =    MultipleMatrix(figura,    matrObert,    n);}
kutpov=kil;
            for(int i = 0; i < n; i++)
            {
               Xr[i] = figura[i][0];
               Yr[i] = figura[i][1];
            }
                 /* matrObert = OberD(alpha,Xcf,Ycf);

            figura = MultipleMatrix(figura, matrObert, n);
            for(int i = 0; i < n; i++)
```

```
 {
   X[i] = figura[i][0];
   Y[i] = figura[i][1];
 }
 double DlDet2, ShDet2;
 ParamDet2(n, X, Y, DlDet2, ShDet2);
 if (ShDet2==0) {
  ShDet2=10;
 }
 mashtY = DlDet2/ShDet2;
 mashtX = ShDet2/DlDet2;
 if (floor(mashtX)==floor(mashtX3))
 {
 vxod=true;
 alpha = kut * M_PI / 180; // В радіанах.
 matrObert = OberD(alpha,Xcf,Ycf);
 figura = MultipleMatrix(figura, matrObert, n);
 for(int i = 0; i < n; i++)
 {
   X[i] = figura[i][0];
   Y[i] = figura[i][1];
 }
 double DlDet2, ShDet2;
 ParamDet2(n, X, Y, DlDet2, ShDet2);

 mashtY = DlDet2/ShDet2;
 mashtX = ShDet2/DlDet2;
 }
 if(mashtY > 20)
{
ky = int(mashtY)/3;
   // ky=5;
}
 if(mashtX > 20) {kx = 20;
                 }
 for(j=0;j<n;j++)
  {
   Xr[j]=(X[j]-Xcf)*mxy/1.2*kx+Xce;
```

```
              Yr[j]=(Y[j]-Ycf)*mxy/1.2*ky+Yce;
             }
         //========================================
             for(int i = 0; i < n; i++)
             {
                 figura[i][0] = Xr[i];
                 figura[i][1] = Yr[i];
                 figura[i][2] = 1;
             }    double alpha2 ;
             if (vxod){
              alpha2 = -(2*kut) * M_PI / 180; // В радіанах.
              }
             else
             {    alpha2 = -(kut) * M_PI / 180; // В радіанах.
             }
             matrObert = OberD(alpha2,Xcf,Ycf);

             figura = MultipleMatrix(figura, matrObert, n);   */
           /*   for(int i = 0; i < n; i++)
             {
                Xr[i] = figura[i][0];
                Yr[i] = figura[i][1];
             } */
         //========================================
for(j=0;j<n;j++)
             {
              Xr2[j]=Xr[j];
              Yr2[j]=Yr[j];
             }
             double Xfind,Yfind,Dlfind,Shfind;
            ParamDet4(n,Xr,Yr,Yfind,Xfind,Dlfind,Shfind);
            Form1->ScrollBox2->HorzScrollBar-
>Position=(Xfind+(Dlfind/2))*0.66;//1111;
            Form1->ScrollBox2->VertScrollBar-
>Position=(Yfind+(Shfind/4))*0.66;//961;
             Form1->Image2->Canvas->Pen->Width=p;
             Form1->Image2->Canvas->Pen->Mode=pmCopy;
              switch(q)
```

```
                {
                 case                  1:Form1->Image2->Canvas->Pen-
>Color=clRed;break;
                 case                  2:Form1->Image2->Canvas->Pen-
>Color=clBlue;break;
                 case                  3:Form1->Image2->Canvas->Pen-
>Color=clGreen;break;
                 case                  4:Form1->Image2->Canvas->Pen-
>Color=clGray;break;
                 default:Form1->Image2->Canvas->Pen-
>Color=clBlack;
                }

               for(j=0;j<n;j++)
               {
                if                    (j==0)Form1->Image2->Canvas-
>MoveTo(Xr[j],Yr[j]);
                else Form1->Image2->Canvas->LineTo(Xr[j],Yr[j]);
               }
               Form1->Image2->Canvas->LineTo(Xr[0],Yr[0]);

               for (j=0;j<FN; j++)
               Form1->Image2->Canvas->Ellipse(Xr2[j]-5,Yr2[j]-
5,Xr2[j]+5,Yr2[j]+5);
          }

           bool pnpoly(int npol, double xp[], double yp[], double x,
double y)
           {
             bool c = false;
             for (int i = 0, j = npol - 1; i < npol; j = i++)
             {
              if ((((yp[i] <= y) && (y < yp[j])) || ((yp[j] <= y) && (y <
yp[i]))) &&
               (((yp[j] - yp[i]) != 0) && (x > ((xp[j] - xp[i]) * (y -
yp[i]) / (yp[j] - yp[i]) + xp[i]))))
                 c = !c;
             }
```

```
      return c;
     }
    void      __fastcall      TForm1::Image2MouseDown(TObject
*Sender,
       TMouseButton Button, TShiftState Shift, int X, int Y)
    {
     int Xr,Yr;
     Image2->Canvas->Pen->Mode=pmXor;
     Image2->Canvas->Pen->Color=clGreen;
     Image2->Canvas->Pen->Width=2;

     double FX2[10],FY2[10];
     for(int i = 0; i < 10; i++)
     {
         FX2[i] = 0; FY2[i] = 0;
         FX2[i] =Xr2[i]; //floor(Xr2[i]);
         FY2[i] =Yr2[i]; //floor(Yr2[i]);
     }
     if(Button==mbLeft && KilksPointDet < 3)
      {
       if(pointselectmode == false)
       {
        // Перевірка, що точка знаходиться у фігурі.
        bool InFigure = pnpoly(FN,FX2,FY2, X, Y);
        if(InFigure == true)
        {
         Xd[indexPoints]=X; Xt[indexPoints]=X;
         Yd[indexPoints]=Y; Yt[indexPoints]=Y;

         if (indexPoints==0){
         Image2->Canvas-
>MoveTo(Xt[indexPoints],Yt[indexPoints]);
          Form1->Image2->Canvas->Ellipse(Xt[indexPoints]-
5,Yt[indexPoints]-5,Xt[indexPoints]+5,Yt[indexPoints]+5);
           }
         else
          {Image2->Canvas-
>LineTo(Xt[indexPoints],Yt[indexPoints]);          Form1->Image2-
>Canvas->Ellipse(Xt[indexPoints]-5,Yt[indexPoints]-
5,Xt[indexPoints]+5,Yt[indexPoints]+5);  }
         indexPoints++;
         }
       }
       else if(pointselectmode == true)
       {
         // Знаходимо найближчу точку фігури.
         double dist[10];
         for(int i = 0; i < FN; i++)
         {
            dist[i] = sqrt(pow(FX2[i]-X,2)+pow(FY2[i]-Y,2));
         }
         double min = dist[0];
         int index = 0;
         for(int i = 1; i < FN; i++)
         {
            if(dist[i] < min && dist[i] != 0)
            {
              min = dist[i];
              index = i;
            }
         }
          if(selectedpoints[index] == false)
         {
         // Замалюємо точку:
         Form1->Image2->Canvas->Brush->Color = clRed;
         Form1->Image2->Canvas->Brush->Style = bsSolid;
         Form1->Image2->Canvas-
>FloodFill(FX2[index],FY2[index],clBlack,fsBorder);
         Form1->Image2->Canvas->Brush->Color = clWhite;
         Xd[indexPoints]=FX2[index];
Xt[indexPoints]=FX2[index];
         Yd[indexPoints]=FY2[index];
Yt[indexPoints]=FY2[index];

         if                          (indexPoints==0)Image2->Canvas-
>MoveTo(Xt[indexPoints],Yt[indexPoints]);
```

```
                else                                    Image2->Canvas-
>LineTo(Xt[indexPoints],Yt[indexPoints]);
                indexPoints++;
                selectedpoints[index] = true;
                }
              }
             }
            else if(Button==mbRight)
              {
                if(pointselectmode == true && indexPoints>0)
                {
                  //Form1->Image2->Canvas->Brush->Color = clWhite;
                  //Form1->Image2->Canvas->Brush->Style = bsSolid;
                  //Form1->Image2->Canvas-
>FloodFill(Xt[indexPoints],Yt[indexPoints],clBlack,fsBorder);

                  //Image2->Canvas-
>MoveTo(Xt[indexPoints],Yt[indexPoints]);
                  indexPoints--;
                  Image2->Canvas-
>LineTo(Xt[indexPoints],Yt[indexPoints]);
                }
                if(pointselectmode == false && indexPoints>0)
                {
                 indexPoints--;
                 Image2->Canvas-
>LineTo(Xt[indexPoints],Yt[indexPoints]);
                }
              }
           if(indexPoints > 2)
              {
                Xd[indexPoints]=Xd[0];
                Yd[indexPoints]=Yd[0];
                Image2->Canvas->LineTo(Xt[0],Yt[0]);
                KilksPointDet=indexPoints;
                indexPoints=0;
                Button5->Enabled=true;
                Edit14->Color=clGreen;
              }
          }
          //--------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
          {
             if (L11){
             ShowMessage("Error!!!");
             }
             else if (L111)
             {
             ShowMessage("Error!!! ");
             }
          /*    Form1->PageControl1->ActivePage=TabSheet1;
              //Обернена функція:
              double Xg[300],Yg[300];
              NullMas(Xg,300);
              NullMas(Yg,300);
              double XcE=Image2->Width/2;
              double YcE=Image2->Height/2;
          //==========================================
              double** figura;
              double** matrObert;
              figura = new double*[KilksPointDet];
              matrObert = new double*[KilksPointDet];
              int kut=0;
              if (vxod)
              {kut=120;
              }
              else
              {kut=60;
              }
              for(int i = 0; i < KilksPointDet; i++)
              {
                  figura[i] = new double[3];
                  matrObert[i] = new double[3];
              }
              for(int i = 0; i < KilksPointDet; i++)
              {
                  figura[i][0] = Xd[i];
                  figura[i][1] = Yd[i];
                  figura[i][2] = 1;
              }
              double alpha2 = kut * M_PI / 180; // В радіанах.
              matrObert = OberD(alpha2,XcMod,YcMod);

              figura      =      MultipleMatrix(figura,      matrObert,
KilksPointDet);
              for(int i = 0; i < KilksPointDet; i++)
              {
                 Xd[i] = figura[i][0];
                 Yd[i] = figura[i][1];
              }
          //==========================================
              for(int j=0;j<KilksPointDet;j++)
              {
                 Xg[j]=((Xd[j]-XcE)/(mxyIm2*kx/1.2))+XcMod;
                 Yg[j]=((Yd[j]-YcE)/(mxyIm2*ky/1.2))+YcMod;
              }
          //==========================================
              for(int i = 0; i < KilksPointDet; i++)
              {
                  figura[i][0] = Xg[i];
                  figura[i][1] = Yg[i];
                  figura[i][2] = 1;
              }
              double alpha3 = -kut * M_PI / 180; // В радіанах.
              matrObert = OberD(alpha3,XcMod,YcMod);

              figura      =      MultipleMatrix(figura,      matrObert,
KilksPointDet);
              for(int i = 0; i < KilksPointDet; i++)
              {
                 Xg[i] = figura[i][0];
                 Yg[i] = figura[i][1];
              }
          //==========================================
             DrawFigure(Xg,Yg,KilksPointDet,clWhite,clBlue);
//==================================================
              double Xg2[300], Yg2[300];
              NullMas(Xg2,300);
              NullMas(Yg2,300);
              kx=((a/2)*mashx);
              ky=(sqrt(3)/2 * a)*mashy;

              double X1,X2,X3;
              double Y1,Y2,Y3;
              X1 = x0-a/2*mashx;
              Y1 = y0;
              X2 = x0;
              Y2 = y0-(sqrt(3)/2 * a)*mashy;
              X3 = x0+a/2*mashx;
              Y3 = y0;
             // double MatrOb[7][3];
              double** MatrOb;
              double** MatrOb2;
              double** MatrOb3;
              double** base;
              MatrOb = new double*[7];
              MatrOb2 = new double*[3];
              MatrOb3 = new double*[3];
              base = new double*[3];
             // matrObert = new double*[KilksPointDet];
               for(int i = 0; i < 7; i++)
              {
                 MatrOb[i] = new double[3];
                // matrObert[i] = new double[3];
              }
               for(int i = 0; i < 3; i++)
              {
                 base[i] = new double[3];
                 MatrOb2[i] = new double[7];
                 MatrOb3[i] = new double[7];
                // matrObert[i] = new double[3];
              }
```

```
        MatrOb [0][0]=1;
         MatrOb[0][1]=0;
         MatrOb[0][2]=0;
         MatrOb[1][0]=0;
         MatrOb[1][1]=1;
         MatrOb[1][2]=0;
         MatrOb[2][0]=0;
         MatrOb[2][1]=0;
         MatrOb[2][2]=1;
         MatrOb[3][0]=double(1.0/2);
         MatrOb[3][1]=double(1.0/2);
         MatrOb[3][2]=0;
         MatrOb[4][0]=double(1.0/2);
         MatrOb[4][1]=0;
         MatrOb[4][2]=double(1.0/2);
         MatrOb[5][0]=0;
         MatrOb[5][1]=double(1.0/2);
         MatrOb[5][2]=double(1.0/2);
         MatrOb[6][0]=double(1.0/3);
         MatrOb[6][1]=double(1.0/3);
         MatrOb[6][2]=double(1.0/3);
        double kx1[5];
        double kx2[5];
        double kx3[5];
        NullMas(kx1,5);
        NullMas(kx2,5);
        NullMas(kx3,5);
          for(int i=0;i<7;i++)
       {
         for(int j = 0; j < 3; j++)
         {  StringGrid1->Cells[0][i+1]=i+1;
          StringGrid2->Cells[0][i+1]=i+1;
         //                                    StringGrid1-
>Cells[j+1][i+1]=FloatToStr(MatrOb2[i][j]);
          StringGrid1-
>Cells[j+1][i+1]=FloatToStr(MatrOb[i][j]);
        //   StringGrid2->Cells[4][i+1]=FloatToStr(suma[i]);
        }
```

```
   }
     for(int i = 0; i < KilksPointDet; i++)
     {
Coef(X1,Y1,X2,Y2,X3,Y3,Xg[i],Yg[i],kx1[i],kx2[i],kx3[i]);
     }
//=========================================
     Memo1->Text = "Координати: ";
     Memo1->Lines->Add("");
     double base2[3][3];
     for (int i=0;i<3;i++){
     base[0][i]=kx1[i];
     base[1][i]=kx2[i];
     base[2][i]=kx3[i];
     }
      for (int i=0;i<3;i++)
       {
       for (int j=0;j<3;j++)
       {
       base2[i][j]=base[i][j];
       }}
     /* base[0][0]=1;
      base[0][1]=2;
      base[0][2]=3;
      base[1][0]=4;
      base[1][1]=5;
      base[1][2]=6;
      base[2][0]=4;
      base[2][1]=5;
      base[2][2]=4;   */
  /*   double res[3][7];    double res2[3][7];

   double op;
 /*  op=determinant(base,3);
   for(int i=0;i<3;i++)
{
    for ( int j=0;j<3;j++)
    {
                if((i+j)%2==0)
                {
                base2[i][j]=minor(i,j,base,3); //funn1k(Bee,n);
                }
                else
                {
                 base2[i][j]=-(minor(i,j,base,3));
                }
    }
}
for(int i=0;i<3;i++)
{
        for(int j=0;j<3;j++)
        {
        base[i][j]=base2[i][j];

        }
}
for(int i=0;i<3;i++)
{
        for(int j=0;j<3;j++)
        {

        base2[i][j]=base[j][i];
        }
}
for(int i=0;i<3;i++)
{
        for(int j=0;j<3;j++)
        {
        base2[i][j]=base2[i][j]/op;
        }
}
for(int i=0;i<3;i++)
{
        for(int j=0;j<3;j++)
        {
        base[i][j]=base2[i][j];
        }
```

```
}   */
     /*  double suma[7];
      for (int i=0;i<7;i++)
      {
      for (int j=0;j<3;j++)
      {
      MatrOb2[j][i]=MatrOb[i][j];
      }}
     /* for (int i=0;i<3;i++)
      {
      for (int j=0;j<7;j++)
      {
      =res[i][j];
      }} */
    /*   MatrOb3=MultipleMatrix2(base,MatrOb2,3);
       for (int i=0;i<7;i++)
      {
      for (int j=0;j<3;j++)
      {
      MatrOb[i][j]=MatrOb3[j][i];
      }}
      suma[0]=0;
      suma[1]=0;
      suma[2]=0;
      suma[3]=0;
      suma[4]=0;
      suma[5]=0;
      suma[6]=0;
        for (int i=0;i<7;i++)
      {
      for (int j=0;j<3;j++)
      {
      suma[i]+=MatrOb[i][j];
      }}
        /* for (int i=0;i<3;i++)
      {
      for (int j=0;j<7;j++)
      {
```

```
                res2[i][j]=MatrOb3[i][j];
                }}
               /*  for (int i=0;i<7;i++)
                {
                for (int j=0;j<3;j++)
                {
                MatrOb[j][i]=res[i][j];
                }}  */
               /*  for(int i=0;i<7;i++)
             {
               for(int j = 0; j < 3; j++)
               {  StringGrid1->Cells[0][i+1]=i+1;
                StringGrid2->Cells[0][i+1]=i+1;
               //                                          StringGrid1-
>Cells[j+1][i+1]=FloatToStr(MatrOb2[i][j]);
                 StringGrid2-
>Cells[j+1][i+1]=FloatToStr(MatrOb[i][j]);
                 StringGrid2->Cells[4][i+1]=FloatToStr(suma[i]);
              }
             }
                 for(int i=0;i<3;i++)
             {
               for(int j = 0; j < 3; j++)
               {  StringGrid3->Cells[0][i+1]=i+1;

                 StringGrid3->Cells[j+1][i+1]=FloatToStr(base[i][j]);
               }
             }
                 Memo1->Lines->Add("");
              for(int i = 0; i < KilksPointDet; i++)
              {
              Memo1->Text     =     Memo1->Text     +     "Значення
координати" + FloatToStrF(i+1, ffGeneral, 6, 6)+" ";
              Memo1->Text = Memo1->Text + "x1="+"";
              Memo1->Text     =     Memo1->Text     +     "     "     +
FloatToStrF(kx3[i], ffGeneral, 10, 10)+" ";
              Memo1->Lines->Add("");
              Memo1->Text = Memo1->Text + "x2="+"";
```

```
            Memo1->Text = Memo1->Text + " " +
FloatToStrF(kx2[i], ffGeneral, 10, 10)+" ";
            Memo1->Lines->Add("");
            Memo1->Text = Memo1->Text + "x3="+"";
            Memo1->Text = Memo1->Text + " " +
FloatToStrF(kx1[i], ffGeneral, 10, 10)+" ";
            Memo1->Lines->Add("");
             Memo1->Text = Memo1->Text + "x1+x2+x3="+"";
            Memo1->Text = Memo1->Text + " " +
FloatToStrF(kx1[i]+kx2[i]+kx3[i], ffGeneral, 8, 8);
            Memo1->Lines->Add("");
            }

//========================================
        }
        //------------------------------------------------------------------
        void Coef(double x1,double y1,double x2,double y2,double
x3,double y3,double x,double y,
        double &kx1, double &kx2, double &kx3)
        {
           double a,b,c,s,sum,sum2;
           s = double(0.5 *(double(fabs((double(x2 - x1)) *
(double(y3 - y1)) - (double(x3 - x1)) * (double(y2 - y1))))));
           a = double(0.5 *(double(fabs((double(x2 - x1)) *
(double(y - y1)) - (double(x - x1)) * (double(y2 - y1))))));
           b = double(0.5 *(double(fabs((double(x - x1)) *
(double(y3 - y1)) - (double(x3 - x1)) * (double(y - y1))))));
           c = double(0.5 *(double(fabs((double(x2 - x)) *
(double(y3 - y)) - (double(x3 - x)) * (double(y2 - y))))));
           kx1 = a/s;
           kx2 = b/s;
           kx3 = c/s;
           sum=kx1+kx2;
           sum2=sum+kx3;
        }
        //------------------------------------------------------------------
        double** Obert(double alpha)
        {
            double **MatrObert;
            MatrObert = new double*[3];
            for(int i = 0; i < 3; i++)
               MatrObert[i] = new double[3];
            for(int i = 0; i < 3; i++)
            {
               for(int j = 0; j < 3; j++)
               {
                 MatrObert[i][j] = 0;
               }
               MatrObert[i][i] = 1;
            }
            MatrObert[0][0] = cos(alpha);
            MatrObert[0][1] = sin(alpha);
            MatrObert[1][0] = -sin(alpha);
            MatrObert[1][1] = cos(alpha);
            return MatrObert;
          }
          double** MultipleMatrix(double** figura, double **matr,
int N1)
          {
                 double **result;
                 result = new double*[N1];
                 for(int i = 0; i < N1; i++)
                    result[i] = new double[3];
                 for(int i = 0; i < N1; i++)
                 {
                          for(int j = 0; j < 3; j++)
                          {
                         result[i][j] = 0;
                                   for(int k = 0; k < 3; k++)
                                   {
                                            result[i][j] += figura[i][k] *
matr[k][j];
                                   }
                          }
                 }
                 return result;
```

```
        }
        double** MultipleMatrix2(double** figura, double **matr,
int N1=3)
        {
                double **result;
                result = new double*[N1];
                for(int i = 0; i < N1; i++)
                   result[i] = new double[7];

                for(int i = 0; i < N1; i++)
                {
                        for(int j = 0; j < 7; j++)
                        {
                       result[i][j] = 0;
                                for(int k = 0; k < 3; k++)
                                {
                                        result[i][j]  +=  figura[i][k]  *
matr[k][j];
                                }
                        }
                }
                return result;
        }
        //--------------------------------------------------------------------
        double **OberD(double Alfa,double a1,double a2)
        {
              double **MatrPer;
              MatrPer=new double *[3];
              for (int i=0;i<3;i++)
              {
               MatrPer[i]=new double[3];
              }
              for (int i=0;i<3;i++)
              {
                    for (int j=0;j<3;j++)
                    {
                     MatrPer[i][j]=0;
                    }
              }
              MatrPer[2][2]=1;
              MatrPer[0][0]=cos(Alfa);
              MatrPer[1][1]=cos(Alfa);
              MatrPer[0][1]=sin(Alfa);
              MatrPer[1][0]=-sin(Alfa);
              MatrPer[2][0]=(-a1*cos(Alfa))+(a2*sin(Alfa))+a1;
              MatrPer[2][1]=(-a1*sin(Alfa))-(a2*cos(Alfa))+a2;
              return MatrPer;
        }
        //---------------------------------------------------------------------------
        void    __fastcall    TForm1::PointSelectModeClick(TObject
*Sender)
        {
          pointselectmode = !pointselectmode;
        }
        //---------------------------------------------------------------------------
          double **Gomo(double k,double a1,double a2)
        {
              double **MatrPer;
              MatrPer=new double *[3];
              for (int i=0;i<3;i++)
              {
               MatrPer[i]=new double[3];
              }
              for (int i=0;i<3;i++)
              {
                    for (int j=0;j<3;j++)
                    {
                     MatrPer[i][j]=0;
                    }
              }
              MatrPer[2][2]=1;
              MatrPer[0][0]=k;
              MatrPer[1][1]=k;
              MatrPer[2][0]=(1-k)*a1;
              MatrPer[2][1]=(1-k)*a2;
              return MatrPer;
```

```
}
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{  BitBtn1->Enabled=false;
Button12->Visible=true;
  Form1->PageControl1->ActivePage=TabSheet3;
  StringGrid1->Cells[0][0]="№";
  StringGrid1->Cells[1][0]="X1";
  StringGrid1->Cells[2][0]="X2";
  StringGrid1->Cells[3][0]="X3";
  StringGrid1->Cells[4][0]="Сума";
  StringGrid2->Cells[0][0]="№";
  StringGrid3->Cells[0][0]="№";
        StringGrid2->Cells[1][0]="X1";
  StringGrid2->Cells[2][0]="X2";
  StringGrid2->Cells[3][0]="X3";
  StringGrid2->Cells[4][0]="Сума";
  StringGrid3->Cells[1][0]="X1";
  StringGrid3->Cells[2][0]="X2";
  StringGrid3->Cells[3][0]="X3";
 // StringGrid1->Cells[4][0]="Сума";
}
//---------------------------------------------------------------------------
 double determinant(double **x,int n)
{
            int i, j;
            double det=0;
            int e, f, g, h;
            if(n == 1)
            {
                        return x[0][0];
            }
            else if (n == 2)
            {
                        return (x[0][0]*x[1][1])-(x[0][1]*x[1][0]);
            }
            else if (n >= 3)
            {
                        double **c;
```

```
                    c = new double *[n - 1];
                    for (i = 0; i < n; i++)
                            c[i] = new double[n - 1];

                    for (j = 0; j < n; j++)
                    {
                            e = 0;
                            for (g = 1; g < n; g++)
                            {
                                    f = 0;
                                    for(h=0;h<n;h++)
                                            if (h != j)
                                            {
                                            c[e][f] = x[g][h];
                                                            f++;
                                            }
                                    e++;
                            }
                            det     +=      pow(-1,     j     +
2)*x[0][j]*determinant(c,n-1);
                    }
                    return det;
            }
        return det;
    }
    double minor(int i2, int j2, double **x, int n)
    {
    double jj = 0;
    double **kk;
    kk = new double *[n];
    for (int i = 0; i < n; i++)
    kk[i] = new double[n];
    int i3=0,j3=0;
    for (int i=0;i<n;i++)
        {
        for (int j=0;j<n;j++)
        {
        if(i!=i2 && j!=j2)
        {
        kk[i3][j3]=x[i][j];
                j3++;
                if (j3==n-1)
                {
                j3=0;i3++;
                }
              }
        }      }
    jj = determinant(kk,n-1);
    return jj;
    }
    void __fastcall TForm1::Button4Click(TObject *Sender)
    {
    Form1->PageControl1->ActivePage=TabSheet2;
          ClearGraph(Image2);
          Button5->Enabled=false;
          Edit14->Color=clRed;
          BitBtn1->Enabled=false;
       Button4->Enabled=false;
       //DrawGraph(Image2);
          // Form1->ScrollBox2->Width/2;
          // Form1->ScrollBox2->Height/2;
       double XcE,YcE,XcIm3,YcIm3;
       XcE=Image2->Width/2;
       YcE=Image2->Height/2;
       KilksPointDet = 0;
       for(int i = 0; i < 10; i++)
       {
         selectedpoints[i] = false;
       }
       ParamDet();
       ParamModeli();
       BuildIm3(XcE,YcE);
    }
    //-------------------------------------------------------------------
void __fastcall TForm1::Button5Click(TObject *Sender)
    {  Button4->Enabled=false;
```

```
        BitBtn1->Enabled=true;

        Form1->PageControl1->ActivePage=TabSheet1;
            //Обернена функція:
            double Xg[300],Yg[300];
            NullMas(Xg,300);
            NullMas(Yg,300);
            double XcE=Image2->Width/2;
            double YcE=Image2->Height/2;
        //=======================================
            double** figura;
            double** matrObert;
            figura = new double*[KilksPointDet];
            matrObert = new double*[KilksPointDet];
            for(int i = 0; i < KilksPointDet; i++)
            {
              figura[i] = new double[3];
              matrObert[i] = new double[3];
            }
            int kut=60;
             double alpha2=0;
              for(int i = 0; i < KilksPointDet; i++)
            {
                  figura[i][0] = Xd[i];
                  figura[i][1] = Yd[i];
                  figura[i][2] = 1;
            }
               for(int i=0;i<kutpov;i++){
              alpha2 = kut* M_PI / 180;
                matrObert = OberD(alpha2,XcMod,YcMod);
              figura      =      MultipleMatrix(figura,      matrObert,
KilksPointDet);}
              for(int i = 0; i < KilksPointDet; i++)
              {
                Xd[i] = figura[i][0];
                Yd[i] = figura[i][1];
              }
             for(int j=0;j<KilksPointDet;j++)
```

```
{
    Xg[j]=((Xd[j]-XcE)/(mxyIm2*kx/1.2))+XcMod;
    Yg[j]=((Yd[j]-YcE)/(mxyIm2*ky/1.2))+YcMod;
}
 for(int i = 0; i < KilksPointDet; i++)
{
      figura[i][0] = Xg[i];
      figura[i][1] = Yg[i];
      figura[i][2] = 1;
}
   for(int i=0;i<kutpov;i++){

  alpha2 = -kut* M_PI / 180;
    matrObert = OberD(alpha2,XcMod,YcMod);
  figura      =      MultipleMatrix(figura,      matrObert,
KilksPointDet);}
  for(int i = 0; i < KilksPointDet; i++)
  {
    Xd[i] = figura[i][0];
    Yd[i] = figura[i][1];
  }
   for(int i = 0; i < KilksPointDet; i++)
{
     Xg[i] = figura[i][0];
     Yg[i] = figura[i][1];
}
/*if (vxod)
{kut=120;
}
else
{kut=60;
}
for(int i = 0; i < KilksPointDet; i++)
{
  figura[i] = new double[3];
  matrObert[i] = new double[3];
}
for(int i = 0; i < KilksPointDet; i++)
```

```
            {
                  figura[i][0] = Xd[i];
                  figura[i][1] = Yd[i];
                  figura[i][2] = 1;
            }
            double alpha2 = kut * M_PI / 180; // В радіанах.
            matrObert = OberD(alpha2,XcMod,YcMod);

            figura       =       MultipleMatrix(figura,       matrObert,
KilksPointDet);
            for(int i = 0; i < KilksPointDet; i++)
            {
                 Xd[i] = figura[i][0];
                 Yd[i] = figura[i][1];
            }
        //==========================================
            for(int j=0;j<KilksPointDet;j++)
            {
                Xg[j]=((Xd[j]-XcE)/(mxyIm2*kx/1.2))+XcMod;
                Yg[j]=((Yd[j]-YcE)/(mxyIm2*ky/1.2))+YcMod;
            }
        //==========================================
            for(int i = 0; i < KilksPointDet; i++)
            {
                  figura[i][0] = Xg[i];
                  figura[i][1] = Yg[i];
                  figura[i][2] = 1;
            }
            double alpha3 = -kut * M_PI / 180;
            matrObert = OberD(alpha3,XcMod,YcMod);

            figura       =       MultipleMatrix(figura,       matrObert,
KilksPointDet);
            for(int i = 0; i < KilksPointDet; i++)
            {
                 Xg[i] = figura[i][0];
                 Yg[i] = figura[i][1];
            }     */
```

```
//============================================
   DrawFigure(Xg,Yg,KilksPointDet,clWhite,clBlue);
    double Xg2[300], Yg2[300];
    NullMas(Xg2,300);
    NullMas(Yg2,300);
    kx=((a/2)*mashx);
    ky=(sqrt(3)/2 * a)*mashy;
    double X1,X2,X3;
    double Y1,Y2,Y3;
    X1 = x0-a/2*mashx;
    Y1 = y0;
    X2 = x0;
    Y2 = y0-(sqrt(3)/2 * a)*mashy;
    X3 = x0+a/2*mashx;
    Y3 = y0;
   // double MatrOb[7][3];
    double** MatrOb;
    double** MatrOb2;
    double** MatrOb3;
    double** base;
    MatrOb = new double*[7];
    MatrOb2 = new double*[3];
    MatrOb3 = new double*[3];
    base = new double*[3];
   // matrObert = new double*[KilksPointDet];
     for(int i = 0; i < 7; i++)
    {
       MatrOb[i] = new double[3];
      // matrObert[i] = new double[3];
    }
     for(int i = 0; i < 3; i++)
    {
       base[i] = new double[3];
       MatrOb2[i] = new double[7];
       MatrOb3[i] = new double[7];
      // matrObert[i] = new double[3];
    }
    MatrOb [0][0]=1;
```

```
                 MatrOb[0][1]=0;
                 MatrOb[0][2]=0;
                 MatrOb[1][0]=0;
                 MatrOb[1][1]=1;
                 MatrOb[1][2]=0;
                 MatrOb[2][0]=0;
                 MatrOb[2][1]=0;
                 MatrOb[2][2]=1;
                 MatrOb[3][0]=double(1.0/2);
                 MatrOb[3][1]=double(1.0/2);
                 MatrOb[3][2]=0;
                 MatrOb[4][0]=double(1.0/2);
                 MatrOb[4][1]=0;
                 MatrOb[4][2]=double(1.0/2);
                 MatrOb[5][0]=0;
                 MatrOb[5][1]=double(1.0/2);
                 MatrOb[5][2]=double(1.0/2);
                 MatrOb[6][0]=double(1.0/3);
                 MatrOb[6][1]=double(1.0/3);
                 MatrOb[6][2]=double(1.0/3);
                 double test[3][7];
                double kx1[5];
                double kx2[5];
                double kx3[5];
                NullMas(kx1,5);
                NullMas(kx2,5);
                NullMas(kx3,5);
                   for(int i=0;i<7;i++)
               {
                 for(int j = 0; j < 3; j++)
                 {  StringGrid1->Cells[0][i+1]=i+1;
                  StringGrid2->Cells[0][i+1]=i+1;
                 //                                                StringGrid1-
>Cells[j+1][i+1]=FloatToStr(MatrOb2[i][j]);
                   StringGrid1-
>Cells[j+1][i+1]=FloatToStr(MatrOb[i][j]);
                //   StringGrid2->Cells[4][i+1]=FloatToStr(suma[i]);
                 }
```

```
              }
               for(int i = 0; i < KilksPointDet; i++)
               {
          Coef(X1,Y1,X2,Y2,X3,Y3,Xg[i],Yg[i],kx1[i],kx2[i],kx3[i]);
               }
//==========================================================
               Memo1->Text = "Координати: ";
               Memo1->Lines->Add("");
               double base2[3][3];
               for (int i=0;i<3;i++){
                base[2][i]= kx1[i];
               base[0][i]=  kx3[i];
               base[1][i]=kx2[i];
               }
                for (int i=0;i<3;i++)
                 {
                 for (int j=0;j<3;j++)
                 {
                 base2[i][j]=base[i][j];
                 }}
               /* base[0][0]=1;
                base[0][1]=2;
                base[0][2]=3;
                base[1][0]=4;
                base[1][1]=5;
                base[1][2]=6;
                base[2][0]=4;
                base[2][1]=5;
                base[2][2]=4;   */
               double res[3][7];    double res2[3][7];
             double op;
           /*  op=determinant(base,3);
             for(int i=0;i<3;i++)
          {
              for ( int j=0;j<3;j++)
              {
                           if((i+j)%2==0)
                           {
```

```
                base2[i][j]=minor(i,j,base,3); //funn1k(Bee,n);
                }
                else
                {
                 base2[i][j]=-(minor(i,j,base,3));
                }
    }
  }
  for(int i=0;i<3;i++)
  {
        for(int j=0;j<3;j++)
        {
        base[i][j]=base2[i][j];
        }
  }
  for(int i=0;i<3;i++)
  {
        for(int j=0;j<3;j++)
        {
        base2[i][j]=base[j][i];
        }
  }
  for(int i=0;i<3;i++)
  {
        for(int j=0;j<3;j++)
        {
        base2[i][j]=base2[i][j]/op;
        }
  }
  for(int i=0;i<3;i++)
  {
        for(int j=0;j<3;j++)
        {
        base[i][j]=base2[i][j];
        }
  }   */
        double suma[7];
       for (int i=0;i<7;i++)
```

```
 {
 for (int j=0;j<3;j++)
 {
 MatrOb2[j][i]=MatrOb[i][j];
 }}
 for (int i=0;i<7;i++)
 {
 for (int j=0;j<3;j++)
 {
 test[j][i]=MatrOb2[j][i];
 }}
/* for (int i=0;i<3;i++)
 {
 for (int j=0;j<7;j++)
 {
 =res[i][j];
 }} */
 MatrOb3=MultipleMatrix2(base,MatrOb2,3);
  for (int i=0;i<7;i++)
 {
 for (int j=0;j<3;j++)
 {
 MatrOb[i][j]=MatrOb3[j][i];
 }}
 suma[0]=0;
 suma[1]=0;
 suma[2]=0;
 suma[3]=0;
 suma[4]=0;
 suma[5]=0;
 suma[6]=0;
   for (int i=0;i<7;i++)
 {
 for (int j=0;j<3;j++)
 {
 suma[i]+=MatrOb[i][j];
 }}
   /* for (int i=0;i<3;i++)
```

```
          {
          for (int j=0;j<7;j++)
          {
          res2[i][j]=MatrOb3[i][j];
          }}
         /*  for (int i=0;i<7;i++)
          {
          for (int j=0;j<3;j++)
          {
          MatrOb[j][i]=res[i][j];
          }}  */
           for(int i=0;i<7;i++)
       {
         for(int j = 0; j < 3; j++)
         {  StringGrid1->Cells[0][i+1]=i+1;
          StringGrid2->Cells[0][i+1]=i+1;
         //                                              StringGrid1-
>Cells[j+1][i+1]=FloatToStr(MatrOb2[i][j]);
           StringGrid2-
>Cells[j+1][i+1]=FloatToStrF(MatrOb[i][j], ffGeneral, 8, 8);
           StringGrid2->Cells[4][i+1]=FloatToStr(suma[i]);
        }
       }
           for(int i=0;i<3;i++)
       {
         for(int j = 0; j < 3; j++)
         {  StringGrid3->Cells[0][i+1]=i+1;

           StringGrid3->Cells[j+1][i+1]=FloatToStrF(base[j][i],
ffGeneral, 8, 8);
         }
       }
           Memo1->Lines->Add("");
        for(int i = 0; i < KilksPointDet; i++)
        {
        Memo1->Text     =     Memo1->Text     +     "Значення
координати" + FloatToStrF(i+1, ffGeneral, 6, 6)+" ";
        Memo1->Lines->Add("");
```

```
              Memo1->Text = Memo1->Text + "x1="+"";
              Memo1->Text    =    Memo1->Text    +    "    "    +
FloatToStrF(kx3[i], ffGeneral, 8, 8)+" ";
              Memo1->Lines->Add("");
              Memo1->Text = Memo1->Text + "x2="+"";
              Memo1->Text    =    Memo1->Text    +    "    "    +
FloatToStrF(kx2[i], ffGeneral, 8, 8)+" ";
              Memo1->Lines->Add("");
              Memo1->Text = Memo1->Text + "x3="+"";
              Memo1->Text    =    Memo1->Text    +    "    "    +
FloatToStrF(kx1[i], ffGeneral, 8, 8)+" ";
              Memo1->Lines->Add("");
               Memo1->Text = Memo1->Text + "x1+x2+x3="+"";
              Memo1->Text    =    Memo1->Text    +    "    "    +
FloatToStrF(kx1[i]+kx2[i]+kx3[i], ffGeneral, 8, 8);
              Memo1->Lines->Add("");
              }
             }
          void __fastcall TForm1::Edit1Exit(TObject *Sender)
          {
            Button1->Enabled=false;
           if(!TryStrToFloat(Edit1->Text,Lb1)){
           // ShowMessage("Error!!! ");
            Edit7->Color=clRed;
            L11=true;
            L1=false;
             Button3->Visible=true;
            //Lb1= double(0.01);
           // Form1->Edit1->Text=Lb1;
           }
           else {
              Lb1 =(double) StrToFloat(Form1->Edit1->Text);
              if (Lb1<=0 || Lb1>=1){
              L11=false;
              L1=false;
              Edit7->Color=clRed;
              Button3->Visible=true;
              L111=true;
```

```
    //  Lb1= double(0.01);
    // ShowMessage("Error!!! ");
  //   Form1->Edit1->Text=double(0.01);
     }
     else
     {
     L11=false;
     L111=false;
     L1=true;
     Edit7->Color=clGreen;
    // Edit2->Enabled=true;
     Button3->Visible=false;
       if (Lb1>=Ub1 )
   {
    Edit8->Color=clRed;
    U1=false;
     U11=false;
     Edit8->Color=clRed;   Button6->Visible=true;
     U111=false;
     U1111=true;
   }
   else {
   Edit8->Color=clGreen;
    U11=false;
     U111=false;
     U1=true;
     Edit8->Color=clGreen;      Button6->Visible=false;
     U1111=false;
   }
     }
   }
   if (L1 && L2 && L3 && U1 && U2 && U3){
   Button1->Enabled=true;
   }
   else{
     Button1->Enabled=false;
   }
}
```

```
//-------------------------------------------------------------------
void __fastcall TForm1::Edit2Exit(TObject *Sender)
{    Button1->Enabled=false;
   if(!TryStrToFloat(Edit2->Text,Ub1)){
 // ShowMessage("Error!!! ");
  Edit8->Color=clRed;
  U11=true;
  //Lb1= double(0.01);
 // Form1->Edit1->Text=Lb1;
 U1=false;
 U1111=false;
 Button6->Visible=true;
 }
 else {
     Ub1 =(double) StrToFloat(Form1->Edit2->Text);
     if (Ub1<=0 || Ub1>=1){
     U11=false;
     Edit8->Color=clRed;  Button6->Visible=true;
     U111=true;
     U1=false;
       U1111=false;
     }
     else
     {
     U11=false;
     U111=false;
     U1=true;
     U1111=false;
     Edit8->Color=clGreen; Button6->Visible=false;
       if (Lb1>=Ub1 )
   {
    Edit8->Color=clRed;
    U1=false;
     U11=false;
     Edit8->Color=clRed;   Button6->Visible=true;
     U111=false;
     U1111=true;
   }
```

```
            else {
            Edit8->Color=clGreen;
             U11=false;
              U111=false;
              U1=true;
              Edit8->Color=clGreen;       Button6->Visible=false;
              U1111=false;
            }
              }
            }
             if (L1 && L2 && L3 && U1 && U2 && U3){
            Button1->Enabled=true;
            }
            else{
              Button1->Enabled=false;
            }
         }
         //------------------------------------------------------------------
void __fastcall TForm1::Button6Click(TObject *Sender)
         {
          if (U11){
            ShowMessage("Error!!!!!!");
            }
            else if (U111)
            {
            ShowMessage("Error!!! ");
            }
            else if (U1111){
            ShowMessage("Error!!! є");
            }
         }
         //------------------------------------------------------------------
void __fastcall TForm1::Edit3Exit(TObject *Sender)
         {   Button1->Enabled=false;
         if(!TryStrToFloat(Edit3->Text,Lb2)){

           Edit9->Color=clRed;
           L22=true;
```

```
   L2=false;    Button7->Visible=true;
}
else {
   Lb2 =(double) StrToFloat(Form1->Edit3->Text);
   if (Lb2<=0 || Lb2>=1){
   L22=false;
   Edit9->Color=clRed;   Button7->Visible=true;
   L222=true;
     L2=false;
   }
   else
   {
   L22=false;
   L222=false;
   L2=true;
   Edit9->Color=clGreen;
     Edit2->Enabled=true;  Button7->Visible=false;
        if(Lb2>=Ub2){
  Edit10->Color=clRed; Button8->Visible=true;
  U2=false;
   U22=false;
   Edit10->Color=clRed;
   U222=false;
   U2222=true;
  }
  else {
  Edit10->Color=clGreen;  Button8->Visible=false;
   U22=false;
   U222=false;
   U2=true;
   Edit10->Color=clGreen;
    U2222=false;
  }
   }
  }
   if (L1 && L2 && L3 && U1 && U2 && U3){
  Button1->Enabled=true;
  }
```

```
            else{
              Button1->Enabled=false;
            }
         }
         //----------------------------------------------------------------------
void __fastcall TForm1::Button7Click(TObject *Sender)
         {
            if (L22){
            ShowMessage("Error!!! ");
            }
            else if (L222)
            {
            ShowMessage("Error!!! ");
            }
         }
         //----------------------------------------------------------------------
void __fastcall TForm1::Button8Click(TObject *Sender)
         {
         if (U22){
            ShowMessage("Error!!! ");
            }
            else if (U222)
            {
            ShowMessage("Error!!! ");
            }
             else if (U2222){
            ShowMessage("Error!!! ");
            }
         }
         //----------------------------------------------------------------------
void __fastcall TForm1::Edit4Exit(TObject *Sender)
         {   Button1->Enabled=false;
          if(!TryStrToFloat(Edit4->Text,Ub2)){
          // ShowMessage("Error!!! ");
           Edit10->Color=clRed;
           U22=true;
           //Lb1= double(0.01);
          // Form1->Edit1->Text=Lb1;
```

```
  U2=false;
  U2222=false;  Button8->Visible=true;
}
else {
   Ub2 =(double) StrToFloat(Form1->Edit4->Text);
   if (Ub2<=0 || Ub2>=1){
   U22=false;
   Edit10->Color=clRed;
   U222=true;
   U2=false;
   U2222=false;  Button8->Visible=true;
   }
   else
   {
   U22=false;
   U222=false;
   U2=true;
   Edit10->Color=clGreen; Button8->Visible=false;
     if(Lb2>=Ub2){
  Edit10->Color=clRed; Button8->Visible=true;
  U2=false;
   U22=false;
   Edit10->Color=clRed;
   U222=false;
   U2222=true;
  }
  else {
  Edit10->Color=clGreen;  Button8->Visible=false;
   U22=false;
   U222=false;
   U2=true;
   Edit10->Color=clGreen;
    U2222=false;
  }
   }
  }
   if (L1 && L2 && L3 && U1 && U2 && U3){
  Button1->Enabled=true;
```

```
   }
   else{
     Button1->Enabled=false;
   }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button10Click(TObject *Sender)
{
 if (L33){
   ShowMessage("Error!!!!");
   }
   else if (L333)
   {
   ShowMessage("Error!!!");
   }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::EError(TObject *Sender)
{
  if (U33){
   ShowMessage("Error!!! ");
   }
   else if (U333)
   {
   ShowMessage("Error!!!)");
   }
    else if (U3333){
   ShowMessage("Error!!! ");
   }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit5Exit(TObject *Sender)
{     Button1->Enabled=false;
if(!TryStrToFloat(Edit5->Text,Lb3)){
  Edit11->Color=clRed;
  L33=true;
     L3=false;   Button10->Visible=true;
 }
```

```
else {
   Lb3 =(double) StrToFloat(Form1->Edit5->Text);
   if (Lb3<=0 || Lb3>=1){
   L33=false;
   Edit11->Color=clRed;  Button10->Visible=true;
   L333=true;
   L3=false;
   }
   else
   {
   L33=false;
   L333=false;
   L3=true;
   Edit11->Color=clGreen;Button10->Visible=false;
     Edit2->Enabled=true;        if(Lb3>=Ub3){
  Edit12->Color=clRed;    Button9->Visible=true;
  U3=false;
   U33=false;
   Edit12->Color=clRed;
   U333=false;
   U3333=true;
  }
  else {
  Edit12->Color=clGreen;      Button9->Visible=false;
   U33=false;
   U333=false;
   U3=true;
   Edit12->Color=clGreen;
   U3333=false;
  }
   }
  }
   if (L1 && L2 && L3 && U1 && U2 && U3){
  Button1->Enabled=true;
  }
  else{
   Button1->Enabled=false;
  }
```

```
          }
          //-----------------------------------------------------------------------
void __fastcall TForm1::Edit6Exit(TObject *Sender)
          {   Button1->Enabled=false;
          if(!TryStrToFloat(Edit6->Text,Ub3)){
           // ShowMessage("Error!!! ");
            Edit12->Color=clRed;
            U33=true;
            //Lb1= double(0.01);
           // Form1->Edit1->Text=Lb1;
                U3=false; U3333=false;   Button9->Visible=true;
           }
           else {
              Ub3 =(double) StrToFloat(Form1->Edit6->Text);
              if (Ub3<=0 || Ub3>=1){
              U33=false;
              Edit12->Color=clRed;    Button9->Visible=true;
              U333=true;
              U3=false;  U3333=false;
              }
              else
              {  U3333=false;
              U33=false;
              U333=false;
              U3=true;
              Edit12->Color=clGreen;  Button9->Visible=false;
                if(Lb3>=Ub3){
             Edit12->Color=clRed;    Button9->Visible=true;
             U3=false;
              U33=false;
              Edit12->Color=clRed;
              U333=false;
              U3333=true;
            }
            else {
            Edit12->Color=clGreen;      Button9->Visible=false;
             U33=false;
              U333=false;
```

```
      U3=true;
      Edit12->Color=clGreen;
      U3333=false;
     }
      }
     }
      if (L1 && L2 && L3 && U1 && U2 && U3){
     Button1->Enabled=true;
     }
     else{
      Button1->Enabled=false;
     }
   }
   //---------------------------------------------------------------------------
void __fastcall TForm1::ErError2(TObject *Sender)
   {
   if(Error1){
    ShowMessage("Немає точок перетину");
   } else if (Error2){
         ShowMessage("Точок перетину не достатньо для
побудови підобласті (1 або 2 точки)");
   }
   }
   //---------------------------------------------------------------------------
void __fastcall TForm1::Edit6Change(TObject *Sender)
   {
   Button4->Enabled=false;
   }
   //---------------------------------------------------------------------------
void __fastcall TForm1::Edit5Change(TObject *Sender)
   {
   Button4->Enabled=false;
   }
   //---------------------------------------------------------------------------
   void __fastcall TForm1::Edit4Change(TObject *Sender)
   {
   Button4->Enabled=false;
   }
```

```
          //-----------------------------------------------------------------
void __fastcall TForm1::Edit3Change(TObject *Sender)
          {
          Button4->Enabled=false;
          }
          //-----------------------------------------------------------------
void __fastcall TForm1::Edit2Change(TObject *Sender)
          {
          Button4->Enabled=false;
          }
          //-----------------------------------------------------------------
void __fastcall TForm1::Edit1Change(TObject *Sender)
          {
          Button4->Enabled=false;
          }
          //-----------------------------------------------------------------
void __fastcall TForm1::Button12Click(TObject *Sender)
          {
          Form1->PageControl1->ActivePage=TabSheet3;
          }
          //-----------------------------------------------------------------
void __fastcall TForm1::Button15Click(TObject *Sender)
          { Button1Click(Form1); indexPoints=0;
            Button4Click(Form1);    Button5->Enabled=false;
              Edit14->Color=clRed;
          }
          //-----------------------------------------------------------------
          void __fastcall TForm1::Button13Click(TObject *Sender)
          {
          Form1->PageControl1->ActivePage=TabSheet1;
          }
          //-----------------------------------------------------------------
```

ADDITION 2

PROGRAM LISTING

Basic procedures and functions for constructing an experiment plan for a four-component mixture

```
procedure convert(a, b: vector1; var x1, x2, x3, x4: vector);
var
  i, j, g, n: Integer;
  k1, k2: mas;
begin
  k1[1, 1] := a[1];
  k1[1, 2] := a[2];
  k1[1, 3] := a[3];
  k1[1, 4] := 1 - (k1[1, 2] + k1[1, 1] + k1[1, 3]);
  k1[2, 1] := b[1];
  k1[2, 2] := a[2];
  k1[2, 3] := a[3];
  k1[2, 4] := 1 - (k1[2, 2] + k1[2, 1] + k1[2, 3]);
  k1[3, 1] := a[1];
  k1[3, 2] := b[2];
  k1[3, 3] := a[3];
  k1[3, 4] := 1 - (k1[3, 2] + k1[3, 1] + k1[3, 3]);
  k1[4, 1] := b[1];
  k1[4, 2] := b[2];
  k1[4, 3] := a[3];
  k1[4, 4] := 1 - (k1[4, 2] + k1[4, 1] + k1[4, 3]);
  k1[5, 1] := a[1];
  k1[5, 2] := a[2];
  k1[5, 3] := b[3];
  k1[5, 4] := 1 - (k1[5, 2] + k1[5, 1] + k1[5, 3]);
  k1[6, 1] := b[1];
  k1[6, 2] := a[2];
  k1[6, 3] := b[3];
  k1[6, 4] := 1 - (k1[6, 2] + k1[6, 1] + k1[6, 3]);
  k1[7, 1] := a[1];
  k1[7, 2] := b[2];
  k1[7, 3] := b[3];
  k1[7, 4] := 1 - (k1[7, 2] + k1[7, 1] + k1[7, 3]);
```

```
k1[8, 1] := b[1];
k1[8, 2] := b[2];
k1[8, 3] := b[3];
k1[8, 4] := 1 - (k1[8, 2] + k1[8, 1] + k1[8, 3]);
k1[9, 1] := a[1];
k1[9, 2] := a[2];
k1[9, 4] := a[4];
k1[9, 3] := 1 - (k1[9, 2] + k1[9, 1] + k1[9, 4]);
k1[10, 1] := b[1];
k1[10, 2] := a[2];
k1[10, 4] := a[4];
k1[10, 3] := 1 - (k1[10, 2] + k1[10, 1] + k1[10, 4]);
k1[11, 1] := a[1];
k1[11, 2] := b[2];
k1[11, 4] := a[4];
k1[11, 3] := 1 - (k1[11, 2] + k1[11, 1] + k1[11, 4]);
k1[12, 1] := b[1];
k1[12, 2] := b[2];
k1[12, 4] := a[4];
k1[12, 3] := 1 - (k1[12, 2] + k1[12, 1] + k1[12, 4]);
k1[13, 1] := a[1];
k1[13, 2] := a[2];
k1[13, 4] := b[4];
k1[13, 3] := 1 - (k1[13, 2] + k1[13, 1] + k1[13, 4]);
k1[14, 1] := b[1];
k1[14, 2] := a[2];
k1[14, 4] := b[4];
k1[14, 3] := 1 - (k1[14, 2] + k1[14, 1] + k1[14, 4]);
k1[15, 1] := a[1];
k1[15, 2] := b[2];
k1[15, 4] := b[4];
k1[15, 3] := 1 - (k1[15, 2] + k1[15, 1] + k1[15, 4]);
k1[16, 1] := b[1];
k1[16, 2] := b[2];
k1[16, 4] := b[4];
k1[16, 3] := 1 - (k1[16, 2] + k1[16, 1] + k1[16, 4]);
k1[17, 1] := a[1];
k1[17, 3] := a[3];
```

```
k1[17, 4] := a[4];
k1[17, 2] := 1 - (k1[17, 4] + k1[17, 1] + k1[17, 3]);
k1[18, 1] := b[1];
k1[18, 3] := a[3];
k1[18, 4] := a[4];
k1[18, 2] := 1 - (k1[18, 4] + k1[18, 1] + k1[18, 3]);
k1[19, 1] := a[1];
k1[19, 3] := b[3];
k1[19, 4] := a[4];
k1[19, 2] := 1 - (k1[19, 4] + k1[19, 1] + k1[19, 3]);
k1[20, 1] := b[1];
k1[20, 3] := b[3];
k1[20, 4] := a[4];
k1[20, 2] := 1 - (k1[20, 4] + k1[20, 1] + k1[20, 3]);
k1[21, 1] := a[1];
k1[21, 3] := a[3];
k1[21, 4] := b[4];
k1[21, 2] := 1 - (k1[21, 4] + k1[21, 1] + k1[21, 3]);
k1[22, 1] := b[1];
k1[22, 3] := a[3];
k1[22, 4] := b[4];
k1[22, 2] := 1 - (k1[22, 4] + k1[22, 1] + k1[22, 3]);
k1[23, 1] := a[1];
k1[23, 3] := b[3];
k1[23, 4] := b[4];
k1[23, 2] := 1 - (k1[23, 4] + k1[23, 1] + k1[23, 3]);
k1[24, 1] := b[1];
k1[24, 3] := b[3];
k1[24, 4] := b[4];
k1[24, 2] := 1 - (k1[24, 4] + k1[24, 1] + k1[24, 3]);
k1[25, 2] := a[2];
k1[25, 3] := a[3];
k1[25, 4] := a[4];
k1[25, 1] := 1 - (k1[25, 2] + k1[25, 4] + k1[25, 3]);
k1[26, 2] := b[2];
k1[26, 3] := a[3];
k1[26, 4] := a[4];
k1[26, 1] := 1 - (k1[26, 2] + k1[26, 4] + k1[26, 3]);
```

```
    k1[27, 2] := a[2];
    k1[27, 3] := b[3];
    k1[27, 4] := a[4];
    k1[27, 1] := 1 - (k1[27, 2] + k1[27, 4] + k1[27, 3]);
    k1[28, 2] := b[2];
    k1[28, 3] := b[3];
    k1[28, 4] := a[4];
    k1[28, 1] := 1 - (k1[28, 2] + k1[28, 4] + k1[28, 3]);
    k1[29, 2] := a[2];
    k1[29, 3] := a[3];
    k1[29, 4] := b[4];
    k1[29, 1] := 1 - (k1[29, 2] + k1[29, 4] + k1[29, 3]);
    k1[30, 2] := b[2];
    k1[30, 3] := a[3];
    k1[30, 4] := b[4];
    k1[30, 1] := 1 - (k1[30, 2] + k1[30, 4] + k1[30, 3]);
    k1[31, 2] := a[2];
    k1[31, 3] := b[3];
    k1[31, 4] := b[4];
    k1[31, 1] := 1 - (k1[31, 2] + k1[31, 4] + k1[31, 3]);
    k1[32, 2] := b[2];
    k1[32, 3] := b[3];
    k1[32, 4] := b[4];
    k1[32, 1] := 1 - (k1[32, 2] + k1[32, 4] + k1[32, 3]);
    j := 0;
    for i := 1 to 32 do
    begin
     if (k1[i, 1] >= a[1]) and (k1[i, 1] <= b[1]) and (k1[i, 2] >=
a[2]) and (k1[i, 2] <= b[2]) and (k1[i, 3] >= a[3]) and (k1[i, 3] <=
b[3]) and (k1[i, 4] >= a[4]) and (k1[i, 4] <= b[4]) then
     begin
      j := j + 1;
      k2[j, 1] := k1[i, 1];
      k2[j, 2] := k1[i, 2];
      k2[j, 3] := k1[i, 3];
      k2[j, 4] := k1[i, 4];
     end;
     for n := j + 1 to 32 do
```

```
      begin
        k2[n, 1] := 0;
        k2[n, 2] := 0;
        k2[n, 3] := 0;
        k2[n, 4] := 0;
      end;
    end;
    for g := 1 to 32 do
    begin
      x1[g] := k2[g, 1];
      x2[g] := k2[g, 2];
      x3[g] := k2[g, 3];
      x4[g] := k2[g, 4];
    end;
  end;
  procedure grani(x1, x2, x3, x4: vector; var ox1, ox2, ox3, ox4: vector);
  type
    PMyList = ^AList;
    AList = record
      R1: real;
      R2: real;
      R3: real;
      R4: real;
    end;
  var
    MyList: TList;
    ARecord: PMyList;
    i, i1, i2, i3, j, k, prov_sovp, count: Integer;
    sumx1, sumx2, sumx3, sumx4, serx1, serx2, serx3, serx4: Real;
  begin
    MyList := TList.create;
    sumx1 := 0;
    serx1 := 0;
    count := 0;
    sumx2 := 0;
    serx2 := 0;
```

```
        sumx3 := 0;
        serx3 := 0;
        sumx4 := 0;
        serx4 := 0;
        for i := 1 to 32 do
        begin
          prov_sovp := 0;
          if i <> 1 then
          begin
            for k := 1 to i - 1 do
            begin
              if (x1[i] = x1[k]) then
                prov_sovp := 1;
            end;
          end;
          if ((i = 1) or (prov_sovp = 0)) then
          begin
            if ((x1[i] <> 0) and (x2[i] <> 0) and (x3[i] <> 0) and
(x4[i] <> 0)) then
            begin
              for j := i to 32 do
              begin
                if (x1[i] = x1[j + 1]) then
                begin
                  sumx1 := sumx1 + x1[j + 1];
                  sumx2 := sumx2 + x2[j + 1];
                  sumx3 := sumx3 + x3[j + 1];
                  sumx4 := sumx4 + x4[j + 1];
                  count := count + 1;
                end;
              end;
              if (count <> 0) then
              begin
                sumx1 := sumx1 + x1[i];
                sumx2 := sumx2 + x2[i];
                sumx3 := sumx3 + x3[i];
                sumx4 := sumx4 + x4[i];
                serx1 := sumx1 / (count + 1);
                ox1[i] := serx1;
                serx2 := sumx2 / (count + 1);
                ox2[i] := serx2;
                serx3 := sumx3 / (count + 1);
                ox3[i] := serx3;
                serx4 := sumx4 / (count + 1);
                ox4[i] := serx4;
                New(ARecord);
                ARecord^.R1 := ox1[i];
                ARecord^.R2 := ox2[i];
                ARecord^.R3 := ox3[i];
                ARecord^.R4 := ox4[i];
                MyList.Add(ARecord);
              end;
              sumx1 := 0;
              serx1 := 0;
              count := 0;
              sumx2 := 0;
              serx2 := 0;
              sumx3 := 0;
              serx3 := 0;
              sumx4 := 0;
              serx4 := 0;
            end;
          end;
        end;
        for i1 := 1 to 32 do
        begin
          prov_sovp := 0;
          if i1 <> 1 then
          begin
            for k := 1 to i1 - 1 do
            begin
              if (x2[i1] = x2[k]) then
                prov_sovp := 1;
            end;
          end;
          if ((i1 = 1) or (prov_sovp = 0)) then
```

```
        begin

         if ((x1[i1] <> 0) and (x2[i1] <> 0) and (x3[i1] <> 0) and
(x4[i1] <> 0)) then
         begin
          for j := i1 to 32 do
          begin
           if (x2[i1] = x2[j + 1]) then
           begin
            sumx1 := sumx1 + x1[j + 1];
            sumx2 := sumx2 + x2[j + 1];
            sumx3 := sumx3 + x3[j + 1];
            sumx4 := sumx4 + x4[j + 1];
            count := count + 1;
           end;
          end;
          if (count <> 0) then
          begin
           sumx1 := sumx1 + x1[i1];
           sumx2 := sumx2 + x2[i1];
           sumx3 := sumx3 + x3[i1];
           sumx4 := sumx4 + x4[i1];
           serx1 := sumx1 / (count + 1);
           ox1[i1] := serx1;
           serx2 := sumx2 / (count + 1);
           ox2[i1] := serx2;
           serx3 := sumx3 / (count + 1);
           ox3[i1] := serx3;
           serx4 := sumx4 / (count + 1);
           ox4[i1] := serx4;
           New(ARecord);
           ARecord^.R1 := ox1[i1];
           ARecord^.R2 := ox2[i1];
           ARecord^.R3 := ox3[i1];
           ARecord^.R4 := ox4[i1];
           MyList.Add(ARecord);
          end;
          sumx1 := 0;
          serx1 := 0;
          count := 0;
          sumx2 := 0;
          serx2 := 0;
          sumx3 := 0;
          serx3 := 0;
          sumx4 := 0;
          serx4 := 0;
        end;
      end;
    end;
    for i2 := 1 to 32 do
    begin
      prov_sovp := 0;
      if i2 <> 1 then
      begin
        for k := 1 to i2 - 1 do
        begin
          if (x3[i2] = x3[k]) then
            prov_sovp := 1;
        end;
      end;
      if ((i2 = 1) or (prov_sovp = 0)) then
      begin
        if ((x1[i2] <> 0) and (x2[i2] <> 0) and (x3[i2] <> 0) and
(x4[i2] <> 0)) then
        begin
          for j := i2 to 32 do
          begin
            if (x3[i2] = x3[j + 1]) then
            begin
              sumx1 := sumx1 + x1[j + 1];
              sumx2 := sumx2 + x2[j + 1];
              sumx3 := sumx3 + x3[j + 1];
              sumx4 := sumx4 + x4[j + 1];
              count := count + 1;
            end;
          end;
          if (count <> 0) then
          begin
            sumx1 := sumx1 + x1[i2];
            sumx2 := sumx2 + x2[i2];
            sumx3 := sumx3 + x3[i2];
            sumx4 := sumx4 + x4[i2];
            serx1 := sumx1 / (count + 1);
            ox1[i2] := serx1;
            serx2 := sumx2 / (count + 1);
            ox2[i2] := serx2;
            serx3 := sumx3 / (count + 1);
            ox3[i2] := serx3;
            serx4 := sumx4 / (count + 1);
            ox4[i2] := serx4;
            New(ARecord);
            ARecord^.R1 := ox1[i2];
            ARecord^.R2 := ox2[i2];
            ARecord^.R3 := ox3[i2];
            ARecord^.R4 := ox4[i2];
            MyList.Add(ARecord);
          end;
          sumx1 := 0;
          serx1 := 0;
          count := 0;
          sumx2 := 0;
          serx2 := 0;
          sumx3 := 0;
          serx3 := 0;
          sumx4 := 0;
          serx4 := 0;
        end;
      end;
    end;
    for i3 := 1 to 32 do
    begin
      prov_sovp := 0;
      if i3 <> 1 then
      begin
```

```
          for k := 1 to i3 - 1 do
          begin
            if (x4[i3] = x4[k]) then
              prov_sovp := 1;
          end;
        end;
        if ((i3 = 1) or (prov_sovp = 0)) then
        begin
          if ((x1[i3] <> 0) and (x2[i3] <> 0) and (x3[i3] <> 0) and
(x4[i3] <> 0)) then
          begin
            for j := i3 to 32 do
            begin
              if (x4[i3] = x4[j + 1]) then
              begin
                sumx1 := sumx1 + x1[j + 1];
                sumx2 := sumx2 + x2[j + 1];
                sumx3 := sumx3 + x3[j + 1];
                sumx4 := sumx4 + x4[j + 1];
                count := count + 1;
              end;
            end;
            if (count <> 0) then
            begin
              sumx1 := sumx1 + x1[i3];
              sumx2 := sumx2 + x2[i3];
              sumx3 := sumx3 + x3[i3];
              sumx4 := sumx4 + x4[i3];
              serx1 := sumx1 / (count + 1);
              ox1[i3] := serx1;
              serx2 := sumx2 / (count + 1);
              ox2[i3] := serx2;
              serx3 := sumx3 / (count + 1);
              ox3[i3] := serx3;
              serx4 := sumx4 / (count + 1);
              ox4[i3] := serx4;
              New(ARecord);
              ARecord^.R1 := ox1[i3];
              ARecord^.R2 := ox2[i3];
              ARecord^.R3 := ox3[i3];
              ARecord^.R4 := ox4[i3];
              MyList.Add(ARecord);
            end;
            sumx1 := 0;
            serx1 := 0;
            count := 0;
            sumx2 := 0;
            serx2 := 0;
            sumx3 := 0;
            serx3 := 0;
            sumx4 := 0;
            serx4 := 0;
          end;
        end;
      end;
      for j := 0 to (MyList.Count - 1) do
      begin
        ARecord := MyList.Items[j];
        ox1[j+1]:= ARecord^.R1;
         ox2[j+1]:= ARecord^.R2;
          ox3[j+1]:= ARecord^.R3;
           ox4[j+1]:= ARecord^.R4;
      end;
    end;
    procedure rebra(x1, x2, x3, x4: vector; var dx1, dx2, dx3,
dx4: vector);
type
      PMyList = ^AList;
      AList = record
        R1: real;
        R2: real;
        R3: real;
        R4: real;
      end;
var
      MyList: TList;
```

```
       ARecord: PMyList;
       i, i1, i2, i3, j, count: Integer;
       sumx1, sumx2, sumx3, sumx4, serx1, serx2, serx3, serx4:
Real;
      begin
       MyList := TList.create;
       sumx1 := 0;
       serx1 := 0;
       count := 0;
       sumx2 := 0;
       serx2 := 0;
       sumx3 := 0;
       serx3 := 0;
       sumx4 := 0;
       serx4 := 0;
       for i := 1 to 32 do
       begin
         if ((x1[i] <> 0) and (x2[i] <> 0) and (x3[i] <> 0) and
(x4[i] <> 0)) then
         begin
          for j := i to 32 do
          begin
           if (x1[i] = x1[j + 1])then  begin
            if (x2[i]=x2[j+1]) then
             begin
             sumx1 := sumx1 + x1[j + 1];
             sumx2 := sumx2 + x2[j + 1];
             sumx3 := sumx3 + x3[j + 1];
             sumx4 := sumx4 + x4[j + 1];
             count := count + 1;
             end;
            if (x3[i]=x3[j+1]) then
             begin
             sumx1 := sumx1 + x1[j + 1];
             sumx2 := sumx2 + x2[j + 1];
             sumx3 := sumx3 + x3[j + 1];
             sumx4 := sumx4 + x4[j + 1];
             count := count + 1;
```

```
    end;
   if (x4[i]=x4[j+1]) then
    begin
    sumx1 := sumx1 + x1[j + 1];
    sumx2 := sumx2 + x2[j + 1];
    sumx3 := sumx3 + x3[j + 1];
    sumx4 := sumx4 + x4[j + 1];
    count := count + 1;
    end;
  end;
if (count <> 0) then
begin
  sumx1 := sumx1 + x1[i];
  sumx2 := sumx2 + x2[i];
  sumx3 := sumx3 + x3[i];
  sumx4 := sumx4 + x4[i];
  serx1 := sumx1 / (count + 1);
  dx1[i] := serx1;
  serx2 := sumx2 / (count + 1);
  dx2[i] := serx2;
  serx3 := sumx3 / (count + 1);
  dx3[i] := serx3;
  serx4 := sumx4 / (count + 1);
  dx4[i] := serx4;
  New(ARecord);
  ARecord^.R1 := dx1[i];
  ARecord^.R2 := dx2[i];
  ARecord^.R3 := dx3[i];
  ARecord^.R4 := dx4[i];
  MyList.Add(ARecord);
end;
sumx1 := 0;
serx1 := 0;
count := 0;
sumx2 := 0;
serx2 := 0;
sumx3 := 0;
serx3 := 0;
```

```
              sumx4 := 0;
              serx4 := 0;
             end;
             end;
         end;
         for i1 := 1 to 32 do
         begin
             if ((x1[i1] <> 0) and (x2[i1] <> 0) and (x3[i1] <> 0) and
(x4[i1] <> 0)) then
             begin
               for j := i1 to 32 do
               begin
                 if (x2[i1] = x2[j + 1]) then
                 begin
                   if (x3[i1]=x3[j+1])then
                   begin
                   sumx1 := sumx1 + x1[j + 1];
                   sumx2 := sumx2 + x2[j + 1];
                   sumx3 := sumx3 + x3[j + 1];
                   sumx4 := sumx4 + x4[j + 1];
                   count := count + 1;
                 end;
                   if (x4[i1]=x4[j+1])then
                   begin
                   sumx1 := sumx1 + x1[j + 1];
                   sumx2 := sumx2 + x2[j + 1];
                   sumx3 := sumx3 + x3[j + 1];
                   sumx4 := sumx4 + x4[j + 1];
                   count := count + 1;
                 end;
               end;
               if (count <> 0) then
               begin
                 sumx1 := sumx1 + x1[i1];
                 sumx2 := sumx2 + x2[i1];
                 sumx3 := sumx3 + x3[i1];
                 sumx4 := sumx4 + x4[i1];
                 serx1 := sumx1 / (count + 1);
```

```
          dx1[i1] := serx1;
          serx2 := sumx2 / (count + 1);
          dx2[i1] := serx2;
          serx3 := sumx3 / (count + 1);
          dx3[i1] := serx3;
          serx4 := sumx4 / (count + 1);
          dx4[i1] := serx4;
          New(ARecord);
          ARecord^.R1 := dx1[i1];
          ARecord^.R2 := dx2[i1];
          ARecord^.R3 := dx3[i1];
          ARecord^.R4 := dx4[i1];
          MyList.Add(ARecord);
         end;
         sumx1 := 0;
         serx1 := 0;
         count := 0;
         sumx2 := 0;
         serx2 := 0;
         sumx3 := 0;
         serx3 := 0;
         sumx4 := 0;
         serx4 := 0;
        end;
       end;
      end;
      for i2 := 1 to 32 do
      begin
        if ((x1[i2] <> 0) and (x2[i2] <> 0) and (x3[i2] <> 0) and
(x4[i2] <> 0)) then
        begin
         for j := i2 to 32 do
         begin
          if (x3[i2] = x3[j + 1]) then
          begin
           if (x4[i2]=x4[j+1]) then
           begin
            sumx1:= sumx1 + x1[j + 1];
            sumx2:= sumx2 + x2[j + 1];
            sumx3:= sumx3 + x3[j + 1];
            sumx4:= sumx4 + x4[j + 1];
            count:= count + 1;
          end;
         end;
         if (count <> 0) then
         begin
          sumx1 := sumx1 + x1[i2];
          sumx2 := sumx2 + x2[i2];
          sumx3 := sumx3 + x3[i2];
          sumx4 := sumx4 + x4[i2];
          serx1 := sumx1 / (count + 1);
          dx1[i2] := serx1;
          serx2 := sumx2 / (count + 1);
          dx2[i2] := serx2;
          serx3 := sumx3 / (count + 1);
          dx3[i2] := serx3;
          serx4 := sumx4 / (count + 1);
          dx4[i2] := serx4;
          New(ARecord);
          ARecord^.R1 := dx1[i2];
          ARecord^.R2 := dx2[i2];
          ARecord^.R3 := dx3[i2];
          ARecord^.R4 := dx4[i2];
          MyList.Add(ARecord);
         end;
         sumx1 := 0;
         serx1 := 0;
         count := 0;
         sumx2 := 0;
         serx2 := 0;
         sumx3 := 0;
         serx3 := 0;
         sumx4 := 0;
         serx4 := 0;
        end;
       end;
```

```
   end;
   for j := 0 to (MyList.Count - 1) do
   begin
     ARecord := MyList.Items[j];
      dx1[j+1]:= ARecord^.R1;
      dx2[j+1]:= ARecord^.R2;
       dx3[j+1]:= ARecord^.R3;
        dx4[j+1]:= ARecord^.R4;
   end;
 end;
 procedure centr(x1, x2, x3, x4: vector; var cx1, cx2, cx3,
cx4:real);
 var
   i:Integer;
   count,sum1,sum2,sum3,sum4:Real;
 begin
   count:=0; sum1:=0; sum2:=0; sum3:=0; sum4:=0;
   for i:=1 to 32 do begin
     if ((x1[i] <> 0) and (x2[i] <> 0) and (x3[i] <> 0) and (x4[i]
<> 0)) then
       begin
        count:=count+1;
         sum1:=sum1+x1[i];
         sum2:=sum2+x2[i];
         sum3:=sum3+x3[i];
         sum4:=sum4+x4[i];
       end;
     if count <> 0 then
       begin
        cx1:=sum1/count;
        cx2:=sum2/count;
        cx3:=sum3/count;
        cx4:=sum4/count;
       end;
    end;
 end;
 procedure vids_centr (x1,x2,x3,x4:vector;
cx1,cx2,cx3,cx4:Real; a,b:vector1; var dc:vector);
```

```
            var i,n:Integer;
            begin
              for i:=1 to 32 do
              begin
               if ((x1[i] <> 0) and (x2[i] <> 0) and (x3[i] <> 0) and (x4[i]
<> 0)) then
                 begin
                   dc[i]:=Sqrt(Sqr((x1[i]-cx1)/(b[1]-a[1]))+Sqr((x2[i]-
cx2)/(b[2]-a[2]))+Sqr((x3[i]-cx3)/(b[3]-a[3]))+Sqr((x4[i]-cx4)/(b[4]-
a[4])));
                 end;
                for n:=i+1 to 32 do begin
                dc[n]:=0;
                end;
              end;
            end;
            procedure convert2 (x1, x2, x3, x4, dcv, dx1, dx2, dx3, dx4,
dcr, ox1, ox2, ox3, ox4, dcg:vector; var tx1, tx2, tx3, tx4, dc:vector);
             type
             PMyList = ^AList;
             AList = record
               R1: real;     R2: real;    R3: real; R4: real;  R5: Real;
             end;
           var i,j:integer;
             MyList: TList;
             ARecord: PMyList;
            begin
             MyList := TList.create;
             for i:=1 to 32 do
             begin
               if ((x1[i] <> 0) and (x2[i] <> 0) and (x3[i] <> 0) and (x4[i]
<> 0) and (dcv[i]<>0)) then
               begin
                     New(ARecord);
                     ARecord^.R1 := x1[i];
                     ARecord^.R2 := x2[i];
                     ARecord^.R3 := x3[i];
                     ARecord^.R4 := x4[i];
```

```
                  ARecord^.R5 := dcv[i];
                  MyList.Add(ARecord);
            end;
            if ((dx1[i] <> 0) and (dx2[i] <> 0) and (dx3[i] <> 0) and
(dx4[i] <> 0)and (dcr[i]<>0)) then
            begin
                  New(ARecord);
                  ARecord^.R1 := dx1[i];
                  ARecord^.R2 := dx2[i];
                  ARecord^.R3 := dx3[i];
                  ARecord^.R4 := dx4[i];
                  ARecord^.R5 := dcr[i];
                  MyList.Add(ARecord);
            end;
            if ((ox1[i] <> 0) and (ox2[i] <> 0) and (ox3[i] <> 0) and
(ox4[i] <> 0)and (dcr[i]<>0)) then
            begin
                  New(ARecord);
                  ARecord^.R1 := ox1[i];
                  ARecord^.R2 := ox2[i];
                  ARecord^.R3 := ox3[i];
                  ARecord^.R4 := ox4[i];
                  ARecord^.R5 := dcg[i];
                  MyList.Add(ARecord);
            end;
          end;
             for j := 0 to (MyList.Count-1) do
          begin
            ARecord := MyList.Items[j];
           tx1[j+1]:= ARecord^.R1;
             tx2[j+1]:= ARecord^.R2;
              tx3[j+1]:= ARecord^.R3;
               tx4[j+1]:= ARecord^.R4;
                dc[j+1]:= ARecord^.R5;
          end;
         end;
          procedure norm (dc:vector; var dn:real);
          var i,count:Integer;
```

```
  sum,dn1,dn2:Real;
begin
  count:=0;
  for i:=1 to 32 do
  begin
   if (dc[i]<>0) then
   begin
    count:=count+1;
    sum:=sum+dc[i];
   end;
  end;
  dn1:=sum/count;
  dn2:=sqrt(2*dn1);
  dn:=(dn1+dn2)/2;
end;
procedure max_d (x1,x2,x3,x4,dc:vector; var max:integer);
var i:integer;
begin
 for i:=1 to 32 do
  begin
    if ((x1[i] <> 0) and (x2[i] <> 0) and (x3[i] <> 0) and
(x4[i] <> 0) and (dc[i]<>0)) then
    begin
     if dc[i] > dc[max] then
    max := i;
    end;
  end;
  end;
procedure vibir_tochok (tx1,tx2,tx3,tx4,dc:vector; dn:Real;
var px1,px2,px3,px4:vector);
 type
PMyList = ^AList;
AList = record
  R1: real;     R2: real;    R3: real;     R4: real;
end;
var k1,i1,k,j,i,max:Integer;
  dc1,dc2:vector;
  dcc,dc11,dc22,sx1,sx2,sx3,sx4:Real;
```

```
        MyList: TList;
        ARecord: PMyList;
        begin
          MyList := TList.create;
         j:=1;
          max_d(tx1,tx2,tx3,tx4,dc,max);
          px1[j]:=tx1[max];  px2[j]:=tx2[max];  px3[j]:=tx3[max];
px4[j]:=tx4[max];
          tx1[max]:=0;  tx2[max]:=0;  tx3[max]:=0;   tx4[max]:=0;
dc[max]:=0;
          max_d(tx1,tx2,tx3,tx4,dc,max);
          px1[j+1]:=tx1[max];                  px2[j+1]:=tx2[max];
px3[j+1]:=tx3[max]; px4[j+1]:=tx4[max];
          tx1[max]:=0;  tx2[max]:=0;  tx3[max]:=0;   tx4[max]:=0;
dc[max]:=0;
         for i:=1 to 32 do begin
          if ((tx1[i] <> 0) and (tx2[i] <> 0) and (tx3[i] <> 0) and
(tx4[i] <> 0) and (dc[i]<>0)) then
          begin
          vids_centr
(tx1,tx2,tx3,tx4,px1[1],px2[1],px3[1],px4[1],a,b,dc1);
          vids_centr
(tx1,tx2,tx3,tx4,px1[2],px2[2],px3[2],px4[2],a,b,dc2);
          end;
         end;
        for k1:=31 downto 1 do
        for k := 1 to k1 do
         begin
          if (dc[k]<dc[k+1]) then
           begin
            sx1:=tx1[k+1];      sx2:=tx2[k+1];      sx3:=tx3[k+1];
sx4:=tx4[k+1]; dc11:=dc1[k+1]; dc22:=dc2[k+1];  dcc:=dc[k+1];
            tx1[k+1]:=tx1[k];  tx2[k+1]:=tx2[k];  tx3[k+1]:=tx3[k];
tx4[k+1]:=      tx4[k];      dc1[k+1]:=dc1[k];      dc2[k+1]:=dc2[k];
dc[k+1]:=dc[k];
            tx1[k]:=sx1;  tx2[k]:=sx2;  tx3[k]:=sx3;  tx4[k]:=sx4;
dc1[k]:=dc11; dc2[k]:=dc22;  dc[k]:=dcc;
           end;
```

```
  end;
  for i1:=1 to 32 do begin
    if ((dc1[i1]>1.0019) and (dc2[i1]>1.0019)) then begin
    New(ARecord);
        ARecord^.R1 := tx1[i1];
        ARecord^.R2 := tx2[i1];
        ARecord^.R3 := tx3[i1];
        ARecord^.R4 := tx4[i1];
        MyList.Add(ARecord);
    end;
    end;
 for j := 0 to (MyList.Count-1) do
 begin
   ARecord := MyList.Items[j];
  px1[j+3]:= ARecord^.R1;
    px2[j+3]:= ARecord^.R2;
     px3[j+3]:= ARecord^.R3;
      px4[j+3]:= ARecord^.R4;
 end;
   for j := 4 to 14 do
 begin
  px1[j]:= px1[j];
    px2[j]:=px2[j];
     px3[j]:=px3[j];
      px4[j]:=px4[j];
 end;
   for j := 15 to 32 do
 begin
  px1[j]:= 0;
    px2[j]:=0;
     px3[j]:=0;
      px4[j]:=0;
 end;
 end;
procedure TForm1.Button1Click(Sender: TObject);
begin
  a[1] := StrToFloat(Edit1.Text);
  a[2] := StrToFloat(Edit3.Text);
  a[3] := StrToFloat(Edit5.Text);
  a[4] := StrToFloat(Edit7.Text);
  b[1] := StrToFloat(Edit2.Text);
  b[2] := StrToFloat(Edit4.Text);
  b[3] := StrToFloat(Edit6.Text);
  b[4] := StrToFloat(Edit8.Text);
 convert(a, b, x1, x2, x3, x4);
 grANI(x1, x2, x3, x4, ox1, ox2, ox3, ox4);
 rebra(x1, x2, x3, x4, dx1, dx2, dx3, dx4);
 centr(x1, x2, x3, x4, cx1, cx2, cx3, cx4);
 vids_centr(x1,x2,x3,x4,cx1,cx2,cx3,cx4,a,b,dcv);
 vids_centr(dx1,dx2,dx3,dx4,cx1,cx2,cx3,cx4,a,b,dcr);
 vids_centr(ox1,ox2,ox3,ox4,cx1,cx2,cx3,cx4,a,b,dcg);
convert2(x1,x2,x3,x4,dcv,dx1,dx2,dx3,dx4,dcr,ox1,ox2,ox3,
ox4,dcg,tx1,tx2,tx3,tx4,dc);
 norm (dc,dn);
 vibir_tochok (tx1,tx2,tx3,tx4,dc,dn,px1,px2,px3,px4);
 end;
procedure TForm1.Button2Click(Sender: TObject);
 var j:Integer;
begin
  form2.Show;
  for j:=0 to 31 do begin
  form2.StringGrid1.Cells[0, j] := floattostr(px1[j+1]);
  form2.StringGrid1.Cells[1, j] := floattostr(px2[j+1]);
  form2.StringGrid1.Cells[2, j] := floattostr(px3[j+1]);
  form2.StringGrid1.Cells[3, j] := floattostr(px4[j+1]);
  end;
  Form2.edt1.Text:=FloatToStr(cx1);
  Form2.edt2.Text:=FloatToStr(cx2);
  Form2.edt3.Text:=FloatToStr(cx3);
  Form2.edt4.Text:=FloatToStr(cx4);
end;
end.
```

CONTENTS

Резанова Вікторія Георгіївна,

Резанова Наталія Михайлівна

ОПТИМІЗАЦІЯ СКЛАДУ БАГАТОКОМПОНЕНТНИХ СИСТЕМ: ДОСЛІДЖЕННЯ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Редактор Резанова В.Г.

Дизайн та верстка авторські

Формат 60*84/16
Папір офсетний 80гр/м2. Друк цифровий. Гарнітура Times New Roman
Умовн.-друк. арк. 16 Обл.- вид. арк. 5,9
Замовлення № 0520-0070
Підписано до друку 15.05.2025 р.

ТОВ «Видавничий дім «АртЕк»
04050, м. Київ, вул. Юрія Іллєнка, буд. 63
Тел.. 099 552 15 04 ph-artek@ukr.net
www.book-on-demand.com.ua
Свідоцтво про внесення суб'єкта видавничої прави
ДК №4779 від 15.10.14р.