

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE  
NATIONAL TECHNICAL UNIVERSITY OF UKRAINE  
“IGOR SIKORSKY KYIV POLYTECHNIC INSTITUTE”

# **BASICS OF WOLFRAM MATHEMATICA LABORATORY GUIDE**

## **Tutorial**

Recommended by Methodical council of “Igor Sikorsky Kyiv Polytechnic Institute”  
as a tutorial for bachelors  
according to the educational program “Electronic components and systems”  
specialty 171 Electronics

Compiler: K. S. Klen

Electronic online educational publication

Kyiv  
“Igor Sikorsky Kyiv Polytechnic Institute”  
2022

Reviewer *Naida S.A.*, doctor of technical sciences, professor  
“Igor Sikorsky Kyiv Polytechnic Institute”

Responsible editor *Yamnenko Y.S.*, doctor of technical sciences, professor

*The stamp was provided by Methodical council of Igor Sikorsky Kyiv Polytechnic Institute  
(protocol № 3 from 01.12.2022)  
at request of the Academic council of the faculty  
(protocol № 09/2022-2 from 26.09.2022)*

In the preparation of bachelors in the specialty 171 Electronics, the educational program «Electronic Systems», one of the important disciplines that is a component of the training cycle is the discipline «Basics of Wolfram Mathematica». The purpose of developing a computer workshop is to give students a thorough knowledge of the basic constants, operators and functions of Mathematica, the rules of compiling programs and the purpose of Mathematica extension packages. As a result of studying the material of the computer workshop, the student should gain the ability to do engineering calculations; use visualization tools, user interface and graphing; to make calculation programs. The computer workshop contains theoretical information and tasks for up to 8 laboratory classes and a list of recommended literature.

Register № 22/23-145. Volume 2,3 author's sheet

National technical university of Ukraine  
“Igor Sikorsky Kyiv Polytechnic Institute”  
Peremohy Ave., 37, Kyiv, 03056

<https://kpi.ua>

Certificate of inclusion in the State Register of publishers, manufacturers  
and distributors of publishing products DK № 5354 dated 25.05.2017

© Igor Sikorsky Kyiv Polytechnic Institute, 2022

## CONTENT

<b>Introduction</b> .....	4
<b>Laboratory class № 1.</b> Basics of working with the Mathematica package.....	5
<b>Laboratory class № 2.</b> Working with vectors and matrices.....	10
<b>Laboratory class № 3.</b> Graphical functions of the Mathematica system.....	15
<b>Laboratory class № 4.</b> Functions for solving algebraic equations and systems of equations in Mathematica.....	20
<b>Laboratory class № 5.</b> Functions of mathematical analysis.....	27
<b>Laboratory class № 6.</b> Computer interpolation technologies in Mathematica.....	34
<b>Laboratory class № 7.</b> Fundamentals of programming in Mathematica.....	37
<b>Laboratory class № 8.</b> I / O graphics.....	43
<b>Literature</b> .....	50

## **Introduction**

The Mathematica software package is a system of computer algebra designed to process mathematical formulas. Other systems of computer algebra include programs Maxima, Maple, MuPAD, Reduce. The main task of these programs is to automate algebraic transformations and processing of symbolic expressions. The first version of the Mathematica package was developed in 1988, version 11.0.1 of the package was released in September 2016.

The computer workshop discusses the main features of the Mathematica system for processing data arrays, plotting graphs, solving linear and transcendental equations, solving problems of mathematical analysis, data approximation, basics of programming and creating a user interface.

## Laboratory class № 1. Basics of working with the Mathematica package

**Objective:** study the basic features of the Mathematica system.

### Brief theoretical information

#### Arithmetic operations

Operation	Arithmetic operator	Abbreviated form	Function
Addition	+	+=	Plus[x1, x2,...xn]
Subtraction	-	-=	-
Multiplication	*	*=	Times[x1, x2,...xn]
Division	/	/=	Divide[x1, x2]
Exponentiation	^	-	-

#### Named constants

E – number e;

Pi – number  $\pi$ ;

I – imaginary unit  $\sqrt{-1}$ ;

Infinity – imaginary infinity  $+\infty$ , with negative infinity put the sign -;

Degree – the number of radians in degrees  $\pi/180$ ;

EulerGamma – Euler constant 0,577216;

GoldenRatio – constant of the golden section  $\frac{1+\sqrt{5}}{2}$ ;

Catalan – Catalan constant 0.915966.

#### Built-in mathematical functions

##### Functions for determining divisors, the least multiple of integers

Divisors[n] – gives a list of the integers that divide n;

ExtendedGCD[n,m] – gives the extended greatest common divisor of the integers n and m;

GCD[n1,n2...] – gives the greatest common divisor of the n1, n2...;

LCM[n1,n2...] – gives the least common multiple of the n1, n2...

Mod[x,y] – gives the remainder on division of x by y.

##### Functions of rounding real numbers

Round[x] – gives the integer closest to x;

Floor[x] – gives the greatest integer less than or equal to x;

Ceiling[x] – gives the smallest integer greater than or equal to x;

Quotient[x,y] – returns a rounded integer x/y, less or equal to x/y.

##### Calculation of factorials

Factorial[n] – gives the value of n!;

Factorial2[n] – gives the value of  $n! = n*(n-2)*(n-4)...$

##### Obtaining prime numbers

Prime[n] – gives the n-th prime number;

PrimePi[n] – gives the number of primes  $Pi(x)$  less than or equal to x.

##### Elementary functions

*Power and logarithmic functions*

$\sqrt{z} \rightarrow \text{Sqrt}[z]$ ;

$z^a \rightarrow \text{Power}[z,a];$

$e^z \rightarrow \text{Exp}[z];$

$\ln(z) \rightarrow \text{Log}[z];$

$\log_a(z) \rightarrow \text{Log}[a,z];$

*Trigonometric functions*

$\sin(z) \rightarrow \text{Sin}[z];$

$\cos(z) \rightarrow \text{Cos}[z];$

$\text{tg}(z) \rightarrow \text{Tan}[z];$

$\text{ctg}(z) \rightarrow \text{Cot}[z];$

$\text{csc}(z) \rightarrow \text{Csc}[z];$

$\text{sec}(z) \rightarrow \text{Sec}[z].$

*Inverse trigonometric functions*

$\text{arc sin}(z) \rightarrow \text{ArcSin}[z];$

$\text{arc cos}(z) \rightarrow \text{ArcCos}[z];$

$\text{arcctg}(z) \rightarrow \text{ArcCot}[z];$

$\text{arc csc}(z) \rightarrow \text{ArcCsc}[z];$

$\text{arcsec}(z) \rightarrow \text{ArcSec}[z].$

*Hyperbolic functions*

$\sinh(z) \rightarrow \text{Sinh}[z];$

$\cosh(z) \rightarrow \text{Cosh}[z];$

$\text{tgh}(z) \rightarrow \text{Tanh}[z];$

$\text{ctgh}(z) \rightarrow \text{Coth}[z];$

$\text{csc h}(z) \rightarrow \text{Csch}[z];$

$\text{sec h}(z) \rightarrow \text{Sech}[z].$

*Inverse hyperbolic functions*

$\text{arc sinh}(z) \rightarrow \text{ArcSinh}[z];$

$\text{arc cosh}(z) \rightarrow \text{ArcCosh}[z];$

$\text{arcctgh}(z) \rightarrow \text{ArcCoth}[z];$

$\text{arc csc h}(z) \rightarrow \text{ArcCsch}[z];$

$\text{arc sec h}(z) \rightarrow \text{ArcSech}[z].$

### **Arithmetic operations with real numbers**

Real numbers in the system Mathematica are presented in the usual or standart form. When representing a number in the usual form, the whole part of the number is separated from the fractional part by point: 1.35, 0.24. Thus 0 integers it is possible not to write and instead of 0.25 to use marking - .25. Numbers written in this form, call numbers with a fixed point.

A point at the end of a number is an indication that the number is real. For example, the number 131. – real, number 2/5 – is rational, and the number 2./5 – real.

When representing a number in standart form, the number is written in the form of a mantis with whole and fractional part and order in the form of a number degree: 5.\*10^-3, 3.335\*10^-6. You can use a space instead of a multiplication sign. Arithmetic operations on real numbers give an approximate result. The Mathematica system operates with

$2.22507^{-308} \dots 1.79769 \cdot 10^{308}$ . To increase the accuracy of real numbers can be represented in rational using the functions:

Rationalize[z] – converts number z to a nearby rational;

Rationalize[z,dz] – converts number z to a nearby rational with accuracy dz;

### **Arithmetic operations with complex numbers**

The complex number is represented as follows:

$$z = \text{Re}(z) + I * \text{Im}(z).$$

Functions for performing operations on complex numbers:

Abs[z] – gives the absolute value of the complex number z;

Arg[z] – gives the argument of the complex number z;

Conjugate[z] – gives the complex conjugate of the complex number z;

### **Expressions of their transformation and calculation**

#### **Substitutions**

Substitutions are a mathematical apparatus designed to calculate the functions at numerically specified values of the argument. They allow you to tabulate the values of functions. The Mathematica system uses the symbol «/.»

f(x) /. x->a;

f(x,y,...) /. {x->a,y->b,...};

{f1(x),f2(x),...} /. x->a;

{f1(x,y,..),f2(x,y,...),...} /. {x->a,y->b,...};

f(x) /. x->{x0,x1,...}.

Substitution expressions have the following meaning:

f(x) /. x->a – substitutes in the expression f(x) the value of x=a.

f(x,y,...) /. {x->a,y->b,...} – substitutes in the expression f(x,y,...) the values x=a, y=b,...

{f1(x),f2(x),...} /. x->a – substitutes in the expressions f1(x), f2(x),... the value x=a.

{f1(x,y,..),f2(x,y,...),...} /. {x->a,y->b,...} – substitutes in the expressions f1(x,y,...), f2(x,y,...),... the values x=a, y=b,...

f(x) /. x->{x0,x1,...} – tabulation of the function f(x).

#### **Convert expressions**

Expression conversion functions:

Simplify[f] – simplifies the expression f;

FullSimplify[f] - simplifies the expression f, which contains special functions;

Expand[f] – expands out products and positive integer powers in f;

Collect[f,x] – collects together terms involving the same powers of expression f matching x;

TrigExpand[f] – expands out trigonometric functions;

Factor[f] – factors a polynomial over the integers.

#### **Function Expand**

Modifications of the function Expand:

Expand[f] – expands out products and positive integer powers in f;

ExpandAll[f] – expands out all products and integer powers in any part of the expression f;

ExpandNumerator[f] - expands out products and powers that appear in the numerator of the expression f;

ExpandDenominator[f] - expands out products and powers that appear as denominators in the expression f;

PowerExpand[f] - expands all powers of products and powers of function f;

ComplexExpand[f] – expands the expression f assuming that all variables are real;

ComplexExpand[f, {x1,x2,...}] - expands the expression f assuming that variables matching any of the x are complex ;

FunctionExpand[f] – tries to expand out special and certain other functions in the expression f, when possible reducing compound arguments to simpler ones.

### **Function Collect**

Modifications of the function Collect:

Collect[f,x] – collects together terms involving the same powers of expression f matching x;

Collect[f, {x1,x2,...}] – collects together terms that involve the same powers of of expression f matching x1,x2...

### **Function Factor**

Modifications of the function Factor:

Factor[f] – factors the function f over the integers;

FactorList[f] – gives a list of the factors of the function f, together with their exponents;

FactorTerms[f] – pulls out any overall numerical factor in the expression f;

FactorTermsList[f] – gives a list in which the first element is the overall numerical factor in the expression f, and the second element is the polynomial with the overall factor removed;

FactorInteger[f] – gives a list of the prime factors of the integer f, together with their exponents.

Functions of transformation of trigonometric expressions

TrigReduce[f] – rewrites products and powers of trigonometric functions in the expression f in terms of trigonometric functions with combined arguments.;

TrigExpand[f] – expands out trigonometric functions in the expression f;

TrigFactor[f] – factors trigonometric functions in the expression f;

TrigToExp[f] – converts trigonometric functions to exponentials.;

ExpToTrig [f] – converts exponentials in expression to trigonometric functions.

### **Work program**

1. Calculate the factorial of the number 32.

2. Find the sum of fractions  $\frac{2}{3} + \frac{7}{92} + \frac{3}{31}$  in rational number format.

3. Find the sum of fractions  $\frac{2}{3} + \frac{7}{92} + \frac{3}{31}$  in real number format.



4. Convert the real number, which was calculated in task 3, into a rational number with an accuracy of  $10^{-4}$ .
5. Find the modulus of the number  $(e^{5i} + 32 \cdot \cos(44 + 6i) - \sinh(33))$ .
6. Write the analytical expression of the function  $2 \sin(1/2(x + y)) \cos(1/2(x - y))$  and calculate the value of the function at  $x=5, y=2$ .
7. Write the analytical expression of the function  $1/2(1+1/i)e^{ix} + (1-1/i)e^{ix}$  and calculate the value of the function at  $x=1..10$  with step 1.
8. Collect the expression  $2x^5 - 6x^4 + 2x^3 + 11x^4 - 5 + 7x^2 + 6x^5 - 3$ .
9. Divide  $213x^6 + 213x^5 + 155x^4 - 58x^3 - 46x^2 + 12x + 12$  by  $x^2 + x + 1$ .
10. Simplify the expression  $(x + t - \tau)^2(t - \tau)$ .
11. Represents the expression  $(x + y - 1)^2 + (x - \tau)^3 + 10 \sin(y) \cdot (x - \tau)$  in powers of  $x$ .

### Control questions

1. List the main features of the Mathematica system.
2. Describe the main data types of Mathematica.
3. Specify the form of writing rational and fractional numbers and describe the features of their use.
4. List the main functions designed to simplify mathematical expressions.
5. Give examples of named constants used in Mathematica.
6. Specify a command to get online help for a specific object.
7. Describe the features of working in interactive mode.
8. Specify the purpose of the Collect [] function.
9. Name the function that should be used to simplify mathematical expressions.
10. Specify the operator for substituting the numerical values of the argument in the analytical function.

## Laboratory class № 2. Working with vectors and matrices

**Objective:** study of the functions of the Mathematica system, designed to work with vectors and matrices.

### Brief theoretical information

Vectors and matrices in the Mathematica system are lists. A list is a collection of data bounded by curly brackets. The vector is one-dimensional, the matrix is a two-dimensional list. Elements of vectors and matrices can be real and imaginary numbers, functions, mathematical expressions.

Vector and matrix creation is done with the following functions.

Array [f, n]- generates a list of length n , with elements f [1], f [2],..., f [n];

Array [f, n<sub>1</sub>, n<sub>2</sub>]- generates a list of length n<sub>1</sub>, start with element f [n<sub>2</sub>], and n<sub>2</sub> can be a number, function, expression;

Array [f, { n<sub>1</sub>, n<sub>2</sub>}] generates an n<sub>1</sub> x n<sub>2</sub> array of nested lists, with elements f (n<sub>1</sub>, n<sub>2</sub>);

Array [f, n<sub>1</sub>, n<sub>2</sub>, h] - generates a list of length n<sub>1</sub>, start with element f (n<sub>2</sub>), uses head

h.

### Determining the structure of a vector or matrix

The following functions are used to determine the structure of a vector or matrix:

VectorQ [V] - gives True if expression V is a vector, and gives False otherwise;

MatrixQ [M] - gives True if expression M is a matrix, and gives False otherwise;

Length [V] - gives the number of elements in a vector V;

Length [M] - gives the number of elements in a matrix M;

MemberQ [V, n] - returns True if an element of V matches n, and False otherwise;

FreeQ [V, n] - yields True if no subexpression in V matches n, and yields False otherwise;

FreeQ [M, n] - yields True if no subexpression in M matches n, and yields False otherwise;

Dimensions [V] - gives a list of the dimensions of V;

Dimensions [M] - gives a list of the dimensions of M (the number of rows and the number of columns);

Position [V, n] - gives a list of the positions at which objects matching n appear in vector V;

Count [V, n] - gives the number of elements in V, that match n;

TensorRank [V] - gives the rank of vector V, if V is a tensor;

TensorRank [M] gives the rank of matrix M, if M is a tensor.

### Transformation and creation of vectors and matrices

The Mathematica system has rich features for converting and creating new vectors and matrices (and lists of any level). The following functions can be used for this purpose:

Drop [V, n] - gives V with its first n elements dropped;

Drop [V,-n] - gives V with its last n elements dropped;

Drop [V, {n}] – gives V with its n-th element dropped.;

Drop [V, {m, n}]- gives V with elements m through n dropped;

Last [f] - gives the last element in vector (matrix) f;

Rest [V] - gives vector V with the first element removed;

Take [V, n]- gives the first n elements of vector V;

Take [V,-n]- gives the last n elements of vector V;  
 Take [V, {m, n}] - gives elements m through n of vector V;  
 Append [V, a] - gives vector V with a appended;  
 Prepend [V,-a] - gives vector V with a prepended ;  
 Insert [V, a, n] - inserts a at position n in V, the position is counted from the start;  
 Insert [V, a,-n] - inserts a at position n in V, the position is counted from the end;  
 Delete [V, n] - deletes the element at position n in vector V;  
 {Delete [f, n<sub>1</sub>], Delete [f, n<sub>2</sub>], Delete [f, n<sub>3</sub>], ...} - deletes the elements at positions n<sub>i</sub>  
 in vector (matrix) and creates new vector (matrix).

The creation of new vectors and matrices is also possible by changing the location of the vector or matrix, which uses the following functions.

Flatten [M] - flattens out nested lists;  
 Flatten [M, n] - flattens to level n matrix M;  
 Sort [f] - sorts the elements of vector (matrix) f into canonical order;  
 Reverse [f] - reverses the order of the elements in vector (matrix) f;  
 RotateLeft [f] - cycles the elements in vector (matrix) f one position to the left;  
 RotateLeft [f, n] - cycles the elements in vector (matrix) f n positions to the left;  
 RotateRight [f] - cycles the elements in vector (matrix) f one position to the right;  
 RotateRight [f, n] - cycles the elements in vector (matrix) f n position to the right;  
 Transpose [M] - transposes the first two levels in matrix M.

Representation of vectors and matrices in tabular form is possible using the functions TableForm and MatrixForm, what look like:

```
TableForm[f]
Out[n] // MatrixForm
```

where f – name of the vector or matrix;

n - the number of the line in which the vector or matrix is located;

% - is used if the representation function % // MatrixForm of the vector or matrix in tabular form is located after the vector (matrix).

You can also create a vector or matrix using a function List:

List [a, b, c,... ] — creates vector {a,b,c,...};

List [{a, b, c,..}, {d, e, f, ..}, {g, h, k, ..}] – creates matrix {{a, b, c,..}, {d, e, f, ..}, {g, h, k, ..}}.

### Creation of vectors and matrices using the Range function

The Range function is used to create numerical lists and has the following modifications:

Range [n<sub>max</sub>] - generates the list {1, 2, ..., n<sub>max</sub>};

Range [n<sub>min</sub>, n<sub>max</sub>] - generates the list {n<sub>min</sub>, n<sub>max</sub>};

Range [n<sub>min</sub>, n<sub>max</sub>, dn] - generates the list from n<sub>min</sub> to n<sub>max</sub> uses step dn.

### Creation of vectors and matrices using Table functions

To create vectors and matrices, you can use the Table function, which has the form:

Table [f, { n<sub>max</sub> }] - generates a list of n<sub>max</sub> copies of f;

Table [f,{1, n<sub>max</sub>}] - generates a list of the values of f from 1 to n<sub>max</sub>;

Table [f, {n, n<sub>min</sub>, n<sub>max</sub>}] — generates a list of the values of f starts with n=n<sub>min</sub> to n<sub>max</sub>;

Table [f, {n, n<sub>min</sub>, n<sub>max</sub>, dn}]— generates a list of the values of f starts with n=n<sub>min</sub> to n<sub>max</sub> uses steps dn.

## **Selection elements of vector and matrix**

The following methods of selecting of elements of vectors and matrices are implemented in the Mathematica system:

- use of double square brackets;
- use of the Part function;
- use the Select function.

### **Use of double square brackets**

In this case, the expression that separates the elements of the vector or matrix is represented as:

$f[[n]]$   $f[[n_1, n_2, \dots]]$ ,

where  $f$  - name of the vector or matrix;

$n$  - the selected element;

$n_i$  -  $i$ -th element from the set of selected elements.

### **Select elements of the vector and matrix using the Part function**

The Part function is represented as follows:

$\{\text{Part}[f, n_1], \text{Part}[f, n_2], \dots\}$ ,

where  $f$  – name of vector;

$n_i$  -  $i$ -th element of vector  $f$ .

If you select elements of the matrix using the Part function, this function is represented as follow:

$\{\text{Part}[f, n_1, m_1], \text{Part}[f, n_2, m_2], \dots\}$ ,

where  $n_i$  -  $i$ -th row element of matrix;

$m_i$  -  $i$ -th column element of matrix.

In the case of selecting elements from complex elements of a vector or matrix, the Part function is represented as follows:

$\text{Part}[f, n, m, l]$ ,

where  $f$  - name of vector or matrix;

$n$  - number element of vector  $f$ ;

$m$  - expression level ( $m = 1$  in the case of a vector,  $m = 2$  in the case of a matrix);

$l$  - element number in a vector or matrix.

Output of elements of vectors and matrices is carried out by means of functions MatrixForm and TableForm.

### **Combining vectors and matrices**

The combination of vectors and matrices is carried out using the following functions:

Union [F] - gives a sorted version of a list F, in which all duplicated elements have been dropped;

Union [f<sub>1</sub>, f<sub>2</sub>, ...] - combines f<sub>1</sub>, f<sub>2</sub> removing repeating elements of vectors and matrices;

Join [f<sub>1</sub>, f<sub>2</sub>... ] - combines f<sub>1</sub>, f<sub>2</sub> ... in a single chain (concatenation);

Complement [f<sub>1</sub>, f<sub>2</sub>, ... ] - gives the elements in  $f$  that are not in any of the f<sub>1</sub>, f<sub>2</sub>, ...;

Intersection [f<sub>1</sub>, f<sub>2</sub>, ...] - gives a sorted list of the elements common to all the lists.

### **Mathematical operations on vectors and matrices**

The simplest arithmetic operations on vectors and matrices will be considered in the following example.

Given vectors V1, V2 and matrices M1, M2:

$$V1 = \{1, 3, 5, 2, 6, 4\};$$

$$V2 = \{2, 7, 5, 8, 1, 3\};$$

$$M1 = \{\{1, 2, 3\}, \{6, 5, 4\}, \{1, 3, 5\}\};$$

$$M2 = \{\{3, 2, 1\}, \{4, 5, 6\}, \{5, 3, 1\}\}.$$

Tasks:

- add, subtract, multiply and divide the vector V1 and the matrix M1 by a number 3;

- square vector V2 and matrix M2;

- calculate square root of vector V1 and matrix M1;

- calculate  $e^{V1}$ ,  $\sin(V2)$ ,  $\ln(M1)$ ,  $\cosh(M1)$ .

Other functions designed to work with vectors and matrices are listed below:

Det [M] - gives the determinant of the square matrix;

IdentityMatrix [M] - gives the identity matrix: matrix with the diagonal elements equal 1, and others elements equals 0;

Transpose [M] - transposes the first two levels in M;

Inverse [M] - gives the inverse of a square matrix;

Tr [M] - finds the trace of the matrix M (sum of the diagonal elements);

LinearSolve [M, b] - returns the vector of unknowns of the matrix equation  $M * x = b$ , where M - matrix of coefficients of the system of equations, x - vector of unknowns, b - vector of free terms;

Eigensystem[M] - gives a list of the eigenvalues and eigenvectors of the square matrix M;

Eigenvalues [M] - gives a list of the eigenvalues of the square matrix M;

Eigenvectors[M] - gives a list of the eigenvectors of the square matrix M;

PseudoInverse[M] - finds the pseudoinverse of a rectangular matrix M.

### Work program

1. Create matrix 4x4:

$$z = \begin{pmatrix} N+4 & N/2 & N\%2 & 0 \\ N/4 & N+3 & N\%3 & N\%4 \\ N\%2 & N\%3 & N+2 & N/6 \\ 0 & N\%4 & N/4 & N+1 \end{pmatrix},$$

where N – number of variant, % - the sign of the operation is the remainder of the division.

2. Represent the matrix z in tabular and matrix form.

3. Use the Insert [], Delete[] i Position[] functions to replace the zero elements of the matrix with the row number in which they are located.

4. Find the determinant of the matrix z.

5. Transpose the matrix z.

6. Find the eigenvalues of the matrix z.

7. Create from the first row and third column of the matrix z vector q.

8. Remove from the vector q 3 and 8 elements.

9. Combine the matrix z and the vector q.

10. Sort the elements of the formed list.

### **Control questions**

1. Specify the structure of the Mathematica system, which is used to create vectors and matrices.
2. List the main functions for creating vectors and matrices.
3. Name the function used to represent matrices in tabular form.
4. Specify the functions that are used to select parts of vectors and matrices.
5. Specify the procedure for referring to a single element of a vector or matrix in the Mathematica system.
6. List the basic mathematical operations on vectors and matrices.
7. Specify a function for combining vectors and matrices.
8. Specify a function for sorting vectors and matrices.
9. Specify the functions intended for deleting elements of vectors and matrices.
10. Specify a function for inserting new elements into vectors and matrices.

## Laboratory class № 3. Graphical functions of the Mathematica system

**Objective:** study of graphical functions of Mathematica system.

### Brief theoretical information

#### Two-dimensional graphics

##### Plot function

The Plot function allows you to build graphically defined graphs in two-dimensional space in a rectangular coordinate system. Several functions can be displayed on one graph. By default, the grid is displayed on the screen.

Plot function recording format

`Plot[f, {x, xmin, xmax}];`

`Plot[{f1, f2, ...}, {x, xmin, xmax}],`

where  $f$  – function, the graph of which is built,

$f_i$  –  $i$ -th function, the graph of which is built,  $i=1,2,\dots$

$x$  – function argument,

$x_{min}$ ,  $x_{max}$  – argument change interval  $x$ .

Function Plot options

The options of the Plot function are set as follows:

Option name -> option value.

The main options of the Plot function are:

- setting the scale along the axis:

`PlotRange -> {ymin, ymax}` – sets the y-axis scale from  $y_{min}$  to  $y_{max}$  with automatic step selection;

`PlotRange->{{ xmin, xmax}, {ymin, ymax}}` - sets the scale on the y-axis from  $y_{min}$  to  $y_{max}$  and on the x-axis from  $x_{min}$  to  $x_{max}$  with automatic step selection;

- definition of the axis name:

`AxesLabel -> {"Tx", "Ty"}` – sets the inscriptions  $T_x$  and  $T_y$  on the x and y axes, respectively;

- determining the name of the plot:

`PlotLabel -> "T"` – sets the name of the plot;

- choice of graphics style:

`Axes -> None` – the schedule is built without axes.

##### ListPlot function

Used to plot graphs given as an array of points.

List format of the ListPlot function

`ListPlot[{y1, y2, ...}];`

`ListPlot[{x1, y1}, {x2, y2}, ...],`

where  $y_i$  –  $i$ -th value of function  $y(x)$ ,

$x_i$  -  $i$ -th the value of the function  $y(x)$  argument.

Purpose of the ListPlot function

`ListPlot[{y1, y2, ...}]` – plots the point values of the function  $y(x)$  with the notation of the number of points on the x-axis.

`ListPlot[{x1, y1}, {x2, y2}, ...]` – plots a list of points with specified x and y coordinates.

The ListPlot function has two options:

- definition of the axis name:

AxesLabel -> {"Tx", "Ty"} – sets the inscriptions Tx and Ty on the x and y axes, respectively;

- determining the size of points:

PlotStyle -> PointSize [d] - sets the diameter of the point equal to d.

### **Show function**

Used to plot point and analytical graphs in one plane.

Show recording format

Show[r1,r2],

where r1, r2 – are variables used to denote graphs

### **Choice of graphic style**

The PlotStyle option allows you to select the color of the lines and their thickness.

PlotStyle option directives

- line color:

PlotStyle -> {GrayLevel[k1], GrayLevel[k2], ...},

where k1, k2, ... - the color codes of the lines in shades of gray of the corresponding functions are selected from the range 0..1;

PlotStyle -> {Hue[c1], Hue[c2], ...},

where c1, c2, ... - tabular color codes of the lines of the corresponding functions, selected from the range 0..1;

PlotStyle -> { RGBColor[r1,g1,b1], Hue[r2,g2,b2], ... },

where r1, g1, b1 ... - the brightness of the red, green and blue color components are selected from the range 0..1;

- line thickness:

PlotStyle -> Thickness[d] – sets the thickness of the lines of the graph as a fraction of its full width;

PlotStyle -> AbsoluteThickness[d] – sets the thickness of the graph lines in pixels;

- dashing style:

PlotStyle -> Dashing[{d1, d2, ...}] – sets the stroke length of the graph lines, where di is specified as a fraction of the width of the graph line;

PlotStyle -> AbsoluteDashing[d1] - sets the stroke length of the lines of the graph, where di is specified in pixels;

- point graph:

PlotStyle -> PointSize[d] – graph in the form of circles with a diameter of d, which are measured in fractions of the total width of the graph;

PlotStyle -> AbsolutePointSize[d] – graph in the form of circles with a diameter of d, which are measured in pixels.

### **Graphs of special types**

#### **Graphing functions on a logarithmic scale**

LogPlot [f, {x, xmin, xmax}] - plots a linear-logarithmic graph of the function f with a logarithmic scale on the y-axis in the range xmin..xmax.

LogLinearPlot[f, {x, xmin, xmax}] – plots a logarithmic-linear graph of the function f with a logarithmic scale on the x-axis in the range xmin..xmax.



LogLogPlot[f, {x, xmin, xmax}] – plots a graph of the function f with a logarithmic scale on two axes in the range xmin..xmax.

Functions

LogListPlot[{x1,y1},{x2,y2},...]

LogLinearListPlot[{x1,y1},{x2,y2},...]

LogLogListPlot[{x1,y1},{x2,y2},...]

similar to the previous three and are used to construct scatter plots.

**The function of plotting graphs in the polar coordinate system**

PolarPlot[f, {t, tmin, tmax}] – plots the position of the end of the vector f when the angle t changes from tmin to tmax.

**Chart construction function**

BarChart[{c1,c2,...}] – makes a bar chart with bar lengths.

PieChart[{c1,c2,...}] – makes a pie chart with sector angle proportional to list.

The PieChart function uses a number of options, the description of which is available with the Options[PieChart] command.

**Functions of three-dimensional graphics**

Plot3D[f, {x, xmin, xmax}, {y, ymin, ymax}] – plots the function  $f = f(x, y)$ . Plot3D options are available with the Options[Plot3D] command.

When constructing graphs expressed in 3 arguments, one argument must be expressed in others. For example, the graph of the sphere:  $x^2+y^2+z^2=R^2$ , where R-radius, will be converted to  $z = \sqrt{R^2 - x^2 - y^2}$ .

ParametricPlot3D[{f1,f2,f3}, {t1,t1min,t1max}, {t2,t2min,t2max}] – plots a three-dimensional graph of a parametrically given function  $z(t1, t2) = f(x(t1, t2), y(t1, t2))$ . Plot3D options are available with the Options[ParametricPlot3D] command.

**Work program**

1. Plot graphs of continuous functions in one coordinate plane:

$$y_1 = \frac{1}{N} \cdot x^2;$$

$$y_2 = \frac{N}{3} \cdot x \cdot e^{-x};$$

$$y_3 = N \cdot \ln x,$$

where N – number of variant.

Range of plotting:  $x = 1..10$ .

Name of graphs: "Functions  $y_1=...., y_2=...., y_3=....$ ".

Axis captions: «x», «y(x)».

Line styles are given in table 1.

Table 1. Line styles

Number of variant	Type of lines (dashed-d, solid-s)	Colour	Length of bar, dashed line or pixels	Thickness of line or pixels
1	s-s-s	blue-red-green		1-3-6
2	d-s-s	blue-purple-green	5	1-2-3

3	s-d-s	blue-red-grey	6	2-4-3
4	d-d-s	blue-red-green	3-6	8-6-4
5	s-s-d	blue-brown-green	8	5-6-3
6	d-s-d	brown-black-grey	5-8	7-2-5
7	s-d-d	purple-red-black	3-7	4-3-6
8	d-d-d	blue-red-green	1-3-6	2-9-3
9	s-s-s	blue-purple-green		4-8-6
10	d-s-s	blue-red-grey	8	1-8-3
11	s-d-s	blue-red-green	10	2-3-1
12	d-d-s	blue-brown-green	3-7	2-6-5
13	s-s-d	blue-black-grey	5	2-6-4
14	d-s-d	blue-red-black	1-4	2-3-6
15	s-d-d	blue-black-purple	4-6	1-6-3

2. Plot graphs of point functions in one coordinate plane:

$$y_1 = N\sqrt{x};$$

$$y_2 = \frac{3}{N} \cdot x \cdot \lg x;$$

$$y_3 = N \cdot \sin(x).$$

Range of plotting:  $x = 1..10$  with a step  $\Delta x=1$ .

Axis captions: «x», «y(x)».

The size of the points of the graphs  $y_1 - ((N\%4)*0.01)$ ,  $y_2 - ((N\%5)*0.02)$ ,  $y_3 - ((N\%3)*0.03)$ , where operation % - the remainder of the division.

3. Plot graphs of point and continuous functions of item 1,2 in one coordinate plane.

4. Construct a logarithmic graph of the function

$$y = N \cdot \ln x.$$

Range of change  $x = 1.10000$ .

Variants for which the condition  $N\% 3 = 0$  is fulfilled: on the x-axis - logarithmic scale, y - logarithmic scale.

$N\%3=1$  : no oci x – logarithmic scale, y – linear scale.

$N\%3=2$  : no oci x – linear scale, y – logarithmic scale.

5. Plot a diagram.

For even variants - circular, odd - in the form of columns.

Data:  $\{N\%3, (N+20)\%7, (N+11)\%2, (N+38)\%22, (N+12)\%5, (N+30)\%8\}$ .

6. Plot the graph of the function in the polar coordinate system

$$y_2 = \frac{N}{3} e^{-Nx}.$$

7. Plot a three-dimensional graph of the surface

$$\frac{3y^2}{N} + \frac{2x^2}{N} - \frac{4z^2}{N} = 1.$$

### Control questions

1. List the main functions for plotting.
2. Name the main options of the function Plot [].
3. Specify the function that must be used to plot analytical and point functions on a single graph canvas.
4. Describe the possible recording formats of the ListPlot [] function.
5. List the parameters of this PlotStyle [].

6. List the possible parameters that specify the type of graph line.
7. List the functions for constructing special graphs.
8. Name the functions used to build three-dimensional graphs.
9. Name the functions used to build diagrams.
10. Name the functions used to build graphs in a logarithmic scale.

## Laboratory class № 4. Functions for solving algebraic equations and systems of equations in Mathematica

**Objective:** get skills in solving algebraic equations and systems of equations in Mathematica.

### Brief theoretical information

#### Analytical methods for solving algebraic and transcendental equations

##### Solve function

To solve the equations in analytical form, the Solve function is used, the recording format of which is as follows:

`Solve[f,x],`

where  $f$  – equation, which is written in any form,

$x$  – variable name.

The equation symbol "==" is used to write the equation, for example  $ax^2 + bx + c == 0$ . Pay attention that the Solve function does not always form a solution in the most compact form, so after using this function sometimes you need to use Simplify, Expand or FullSimplify.

##### Roots function

This function is designed to determine the roots of a polynomial, it has the form:

`Roots[f, x],`

where  $f$  – polynomial, the roots of which must be found (can be represented as an equation),  $x$  – polynomial argument.

The result of applying the Roots [f, x] function is the real and complex roots of the equation  $f(x) = 0$ . In this case, the solution can be obtained in analytical and numerical form. The solution in the analytical form in the general case can be obtained for a polynomial not higher than the fourth degree. The solution of  $f(x) = 0$  does not exist if  $f(x)$  – is a polynomial of the fifth and higher degree. However, if the polynomial can be factorized, then the function Roots [f, x] will find all the roots of the corresponding equation.

If the coefficients of the polynomial are given in the form of numbers, then the function Roots [f, x] gives a solution in the form of an exact or approximate value of the roots. The exact value of the roots is represented by numbers in a rational form, the approximate - in the form of real numbers.

Applying the Roots [f, x] function to transcendental equations gives an erroneous result, so its use is irrational for this type of equation.

#### Numerical methods for solving algebraic and transcendental equations

There are a large number of numerical methods for solving algebraic and transcendental equations. The algorithm of any of these methods is a set of conditions for choosing the initial approximation, the calculated ratios and signs of the end of the computational process.

The Mathematica system has many built-in functions for solving algebraic and transcendental equations in numerical form. The main ones are: NSolve, NRroots, FindRoot. Consider in detail these functions and give examples.

##### NSolve function

The NSolve function represents as:

NSolve[f, x],

where f – equation, x – the required unknown.

The result of this function is the roots of the equation. Roots can be real and complex numbers. can solve all equations solved by the Solve function. Its difference is only in the form of answers.

### **NRroots function**

The NRroots function represents as:

NRroots[f, x],

where f – equation, x – the required unknown.

The result of using this function is the roots of the polynomial  $f(x) = 0$ .

### **FindRoot function**

The FindRoot function represents as:

FindRoot[f, {x, x<sub>0</sub>}],

where f – equation, x – the required unknown (root (roots) of the equation), x<sub>0</sub> – initial approximation.

The FindRoot[f, {x, x<sub>0</sub>}] finds the root of the equation  $f(x) = 0$  from the range of x values close to x<sub>0</sub>. The following method of determining the roots of algebraic and transcendental equations using a FindRoot[f, {x, x<sub>0</sub>}] function is recommended:

1. Determining the isolation region of the desired root and selecting the value of the approximation x<sub>0</sub>.
2. Input of the equation  $f(x) = 0$  with assigning it a unique name.
3. Enter the FindRoot[f, {x, x<sub>0</sub>}] function with the selected value of x<sub>0</sub>.
4. Get a solution by pressing <Shift>+<Enter>.

### **Methods for solving systems of equations in the system Mathematica**

The Mathematica system has rich possibilities for solving systems of algebraic equations. Built-in functions allow solving systems of linear and nonlinear equations in analytical and numerical form. Give the opportunity to check the reliability of the results quite effectively and in an original way. During operation, the system issues comments that allow the user to make decisions about the answers received. The main functions for solving systems of equations are: Solve[F,X], Solve[F,X,Y], N[Solve[F,X]], FindRoot[F,X]. Consider these functions, describe the technology of their implementation, give examples and problems for independent solution.

### **Solve[F,X] function**

The Solve[F, X] function allows solving systems of linear and nonlinear equations in analytical form. It is represents as follows:

Solve[{f<sub>1</sub>,f<sub>2</sub>,...}, {x<sub>1</sub>,x<sub>2</sub>,...}]

where f<sub>i</sub> – i-th equation presented in any form, x<sub>i</sub> – i-th unknown.

Equations f<sub>1</sub>, f<sub>2</sub>, ... an also be represented by a unifying sign &&. Examples of function Solve[F,X] representation:

Solve[{a\*x^2+y==b,x+2\*y==a+b},{x,y}]

Solve[a\*x^2+y==b&&x+2\*y==a+b,{x,y}]

When entering equations, the multiplication sign (\*) can be replaced by pressing the <Space> key. When solving practical problems, it is convenient, and in some cases even advisable, to enter equations separately from the Solve function, assigning them names, which are then entered into the Solve function instead of equations. Example:

$$f_1 = a \cdot x^2 + y = b; f_2 = x + 2 \cdot y = a + b$$

Solve[{f1, f2}, {x, y}]

or

Solve[f1 && f2, {x, y}]

This form of notation simplifies the verification of the solution of the system of equations.

### *Systems of equations*

Methods for solving equations:

1. Entering equations with a unique name, which is specified by the assignment sign (=).
2. Write the function Solve[{f1, f2, ...}, {x, y, ...}] also Solve [f1 && f2 && ..., {x, y, ...}].
3. Checking the validity of the solution of the system of equations.

### *Systems of nonlinear algebraic equations*

The method of solving systems of nonlinear equations is the same as linear ones.

### **Solve function [F, X, Y]**

The Solve function [F, X, Y], as well as the Solve function [F, X], allows to solve systems of linear and nonlinear equations in an analytical form, but only with restriction: solutions are carried out on variable X and are excluded for variables Y. For example, the function Solve [{x + 2 \* y == 3, 2 \* x + y ^ 2 + b == 7}, x, y] will determine X and exclude from the solution Y.

### **NSolve function [F, X]**

The NSolve function [F, X] allows solving systems of linear and nonlinear equations in numerical form. It is recorded as follows:

Solve [{f1, f2, ...}, {x1, x2, ...}],

where fi is the i-th equation represented in an arbitrary form, xi is the i-th unknown.

Equations f1, f2, ... can also be represented by the unifying sign &&. The method of solving systems of equations using the function NSolve [F, X] is almost no different from the technology of solving using the function Solve {F, X}.

### **FindRoot function [F, {X, x0}]**

The FindRoot function [F, {X, x0}] solves systems of linear and nonlinear equations by numerical iteration methods. To implement it, you need to know the initial approximations of the unknown. The function looks like:

FindRoot [{f1, f2, ...}, {x1, x10}, {x2, x20}, ...],

Where fi is the i-th equation represented in an arbitrary form, xi is the i-th unknown, xi0 is the initial approximation of the i-th unknown.

Equations f1, f2, ... can also be represented by the unifying sign &&.

The technique of solving equation systems using the FindRoot function [F, {X, x0}] differs significantly from the technology of solving equations using the NSolve function [F, X].

The difference is the need to determine the initial approximations.

### **Eliminate function [F, x]**

The Eliminate function [F, x] is designed to reduce the number of system equations by excluding the specified variables x. It looks like:

Eliminate [{f1, f2, ...}, {x1, x2, ...}],

where fi is the i-th equation, presented in any form,

xi is the i-th unknown to be excluded.

Equations f1, f2, ... can also be represented by the joining sign &&.

This function transforms the original system of equations so that the number of equations and variables is reduced. The limit is one equation with one unknown.

### Matrix methods for solving systems of linear equations

The system of linear algebraic equations can be represented as follows:  $A * X = B$ , where A is a matrix of coefficients, X is a vector of unknowns, B is a vector of free members (right parts) of the system of equations. The method of solving equations in the Mathematica system is simple and consists of the following:

1. Introduction of a matrix of coefficients with the assignment of a name, such as A.
2. Enter a vector of unknowns named X.
3. Introduction of a vector of free members named B.
4. Creating the expression  $z = A. X == B$ .
5. Introduction of the Solve function [z, X].

In addition to the above, there are the following two matrix methods for solving systems of algebraic equations in the Mathematica system.

Method 1. Determination of the vector of unknown X by the formula:  $X = A^{-1}B$ .

The multiplication operation is written by the Dot function, and the matrix inversion operation is written by the Inverse function. Then the decision is written as follows:

$$X := \text{Dot} [\text{Inverse} [A], B]$$

Method 2. Using the LinearSolve function.

The LinearSolve function is written as follows:

$$X := \text{LinearSolve} [A, B]$$

Listing 14 shows an example of solving a system of linear equations

$$\begin{cases} a_1x_1 + 2x_2 - 3x_3 = b_1; \\ -7x_1 + cx_2 + x_3 = b_2; \\ x_1 + x_2 + dx_3 = b_3, \end{cases}$$

in two ways.

### Special cases of solving systems of equations

A system of equations can have a number of solutions equal to the number of unknowns, such a system is called compatible. If the number of equations is infinitely large, then the system is called compatible and indefinite. If the system has no solution, it is called incompatible.

### Work program

1. Findsolution of the transcendental equation in analytical and numerical form in accordance with the option shown in table 1.

Table 1. Transcendental equation

Var.	Equation	Var.	Equation	Var.	Equation
1.11	$3x^2 - e^x = 0$	4.14	$9 - x^2 = e^x$	7.17	$x \cdot \text{tg}(x/2) = \sin(x/2)$
2.12	$\lg x = x - 5$	5.15	$\sin(2x) = 6 - x^2$	8.18	$2^x = x + 5$
3.13	$2^x = \lg x + x^5 + 40$	6.16	$\ln(2 + x) = 0.4x^3$	9.19	$\log_3(5x - 6) = 2\sqrt{10 - 3x}$
10.20	$\log_2 x = x^2 - x$				

2. Find the roots of the polynomial equation in analytical and numerical form according to the option shown in table 2.

Table 2. Polynomial equation

Var.	Equation	Var.	Equation	Var.	Equation
1.11	$x^4 + x^2 + 1 = 0$	4.14	$x^5 - 3x^2 + x - 12 = 0$	7.17	$x^7 - 3x^5 + 2x - 4 = 0$
2.12	$x^5 - 4x^3 + 2x^2 + 1 = 0$	5.15	$x^4 - 6x^3 + 3x - 2 = 0$	8.18	$x^5 - 8x^4 + 4x^2 - 6 = 0$
3.13	$x^6 - 4x^2 + 3x^3 + x - 25 = 0$	6.16	$x^8 - 1 = 0$	9.19	$x^4 - 6x^3 + 2x - 8 = 0$
				10.20	$x^5 - 3x^4 + 3x^3 - 1 = 0$

3. Solve the system of linear equations according to the variant specified in table. 3 using the functions Solve [F, X], Solve [F, X, Y], NSolve [F, X] and the matrix method. Explain the features of using these functions.

Table 3. System of linear equations

Var.	System of equations	Var.	System of equations
1.11	$\begin{cases} 1.5x_1 - 0.8x_2 + 4.25x_3 = 5.1; \\ 1.2x_1 + 7.18x_2 - 3.2x_3 = 4.2; \\ 0.5x_1 - 1.5x_2 + 7.1x_3 = -1.2. \end{cases}$	2.12	$\begin{cases} 6.7x_1 - 0.6x_2 + 0.83x_3 = 6.8; \\ 0.8x_1 + 1.1x_2 + 7.2x_3 = 5.2; \\ 1.2x_1 + 5.4x_2 - 0.54x_3 = -3.2. \end{cases}$
3.13	$\begin{cases} -1.32x_1 + 2.15x_2 + 7.6x_3 = -1.4; \\ 2.62x_1 + 6.1x_2 - 4.12x_3 = 5.6; \\ 8.3x_1 - 2.84x_2 - 1.5x_3 = -6.5. \end{cases}$	4.14	$\begin{cases} 0.51x_1 - 10x_2 - 3.62x_3 = -2.05; \\ 3.09x_1 + 1.23x_2 - 4.64x_3 = -5.6; \\ 3.2x_1 - 2.31x_2 - 8.4x_3 = 6.1. \end{cases}$
5.15	$\begin{cases} 7.12x_1 - 6.66x_2 + 2.6x_3 = -3.1; \\ -1.7x_1 + 6.5x_2 - 0.87x_3 = 2.85; \\ 0.65x_1 + 0.87x_2 - 8.7x_3 = 5.56. \end{cases}$	6.16	$\begin{cases} 6.4x_1 - 0.73x_2 + 2.1x_3 = 3.8; \\ -1.07x_1 + 3.8x_2 - 1.5x_3 = -1.2; \\ 2.7x_1 - 3.1x_2 + 4.2x_3 = -7.5. \end{cases}$
7.17	$\begin{cases} 9.21x_1 - 1.84x_2 + 0.7x_3 = -3.2; \\ -6.17x_1 + 8.5x_2 - 2.87x_3 = -3.75; \\ 0.7x_1 + 0.87x_2 - 8.7x_3 = 2.64. \end{cases}$	8.18	$\begin{cases} 4.3x_1 - 1.2x_2 + 10.3x_3 = 4.2; \\ 0.21x_1 + 6.2x_2 + 3.54x_3 = 5.1; \\ -0.31x_1 - 0.52x_2 + 3.6x_3 = -2.1. \end{cases}$
9.19	$\begin{cases} 6.9x_1 - 2.3x_2 + 1.21x_3 = 3.1; \\ x_1 + 2.3x_2 - 3.4x_3 = -2.3; \\ 0.21x_1 - 0.43x_2 + 6.3x_3 = 3.6. \end{cases}$	10.20	$\begin{cases} 12.4x_1 - 0.56x_2 + 4.2x_3 = 6.3; \\ -0.65x_1 + 4.4x_2 + 1.5x_3 = 1.5; \\ 1.5x_1 + 2.1x_2 - 2.8x_3 = 1.7. \end{cases}$

4. Solve the system of nonlinear equations according to the variant specified in table. 4 using the FindRoot function [F, {X, x0}].

Table 4. System of linear equations

Var.	System of equations	Initial approximations
1.11	$\begin{cases} \sin(x_1 + x_2) - 1.2x_1 = 0.1; \\ x_1^2 + x_2^2 = 1. \end{cases}$	$x_1 = 0.74, x_2 = 0.67.$



2.12	$\begin{cases} tg(x_1x_2 + 0.2) = x_1^2; \\ 0.6x_1^2 + 2x_2^2 = 1. \end{cases}$	$x_1 = 0.88, x_2 = 0.52.$
3.13	$\begin{cases} \sin x_1 + 2\sin x_2 = 1; \\ 2\sin 3x_1^2 + 3\sin 3x_2^2 = 0.3. \end{cases}$	$x_1 = 1.08, x_2 = 0.06.$
4.14	$\begin{cases} tg(x_1 - x_2) - 4x_1 = 0; \\ x_1^2 + 2x_2^2 = 1. \end{cases}$	$x_1 = -0.5, x_2 = 0.6.$
5.15	$\begin{cases} x_1^4 + x_2^2 - 3 = 0; \\ x_1^3 + x_2^3 - 4 = 0. \end{cases}$	$x_1 = 0.95, x_2 = 1.4.$
6.16	$\begin{cases} \sin x_1 - x_2 = 1.3; \\ \cos x_2 - x_1 = -0.82. \end{cases}$	$x_1 = 1.8, x_2 = -0.35.$
7.17	$\begin{cases} x_1^3 + x_2^3 - 6x_1 + 3 = 0; \\ x_1^3 + x_2^3 - 6x_2 = 2. \end{cases}$	$x_1 = 0.52, x_2 = -0.37.$
8.18	$\begin{cases} x_2 + e^{x_1 - x_2} = 0; \\ x_1 + e^{x_1 + x_2} = 0. \end{cases}$	$x_1 = -0.7, x_2 = -0.35.$
9.19	$\begin{cases} \sin(x_2 + 1) - x_1 = 1.2; \\ 2x_2 + \cos x_1 = 2. \end{cases}$	$x_1 = 0.1, x_2 = -0.1.$
10.20	$\begin{cases} \cos(x_2 - 1) + x_1 = 0.5; \\ x_2 - \cos x_1 = 3. \end{cases}$	$x_1 = 0, x_2 = \pi.$

5. Using the function Eliminate [F, x] from the system of linear equations according to table. 5 express the analytical expression of the specified variable.

Table 5. System of linear equations

Var.	System of equations	Variable	Var.	System of equations	Variable
1.11	$\begin{cases} 1.2x_1 - 1.06x_2 - 6.7x_3 + 5.3x_4 = 2.12; \\ 4.2x_1 - 6.3x_2 - 0.9x_3 - 1.7x_4 = -1.1; \\ 0.6x_1 + 6.8x_2 + 0.82x_3 + 1.3x_4 = 0.83. \end{cases}$	$x_1$	2.12	$\begin{cases} 9.7x_1 + 0.35x_2 - 1.84x_3 + 0.6x_4 = 2.15; \\ 4.64x_1 - 7.1x_2 - 4.3x_3 + 2.2x_4 = 1.5; \\ 0.32x_1 + 0.348x_2 - 3.3x_3 - 1.7x_4 = -3.1. \end{cases}$	$x_3$
3.13	$\begin{cases} 6.5x_1 - 2.34x_2 + 1.4x_3 - 1.5x_4 = 2.8; \\ 0.5x_1 + 7.3x_2 - 2.4x_3 + 2.7x_4 = -3.8; \\ 8.6x_1 + 0.34x_2 - 6.4x_3 - 1.7x_4 = 0.64. \end{cases}$	$x_2$	4.14	$\begin{cases} 2.8x_1 + 4.3x_2 - 3.7x_3 + 0.8x_4 = 5.1; \\ -0.45x_1 - 8.24x_2 + 4.8x_3 + 5.2x_4 = 5.4; \\ 0.54x_1 + 2.3x_2 + 3.7x_3 - 1.7x_4 = 1.54. \end{cases}$	$x_1$
5.15	$\begin{cases} 6x_1 + 0.13x_2 - 0.67x_3 - 0.8x_4 = 1.9; \\ 3.8x_1 + 1.25x_2 - 4.3x_3 + 3.2x_4 = 6.4; \\ 0.38x_1 - 0.64x_2 + 3.2x_3 + 2.3x_4 = 5.4. \end{cases}$	$x_3$	6.16	$\begin{cases} 1.5x_1 - 2.6x_2 + 7x_3 - 0.8x_4 = -11.2; \\ 6.6x_1 + 1.3x_2 - 1.24x_3 + 0.9x_4 = 5.3; \\ 0.85x_1 - 8.4x_2 + 4.7x_3 + 2.7x_4 = 1.6. \end{cases}$	$x_2$

7.17	$\begin{cases} 6.2x_1 - 0.52x_2 + 2.3x_3 - 8.7x_4 = -1.8; \\ -4.2x_1 + 3.4x_2 - 0.5x_3 - 5.4x_4 = 0.7; \\ 0.2x_1 + 0.8x_2 + 3.6x_3 + 9.1x_4 = 3.2. \end{cases}$	$x_1$	8.18	$\begin{cases} 0.63x_1 - 0.54x_2 + 1.7x_3 - 4.7x_4 = 3.6; \\ 0.65x_1 + 4.4x_2 + 0.15x_3 + 2.3x_4 = 2.3; \\ 1.5x_1 + 0.2x_2 + 4.1x_3 - 6.7x_4 = 2.8. \end{cases}$	$x_3$
9.19	$\begin{cases} 8.4x_1 - 0.25x_2 + 3.1x_3 + 5.2x_4 = -5.7; \\ -0.3x_1 + 6.1x_2 - 1.54x_3 - 6.9x_4 = 3.3; \\ -6.8x_1 + 1.2x_2 - 7x_3 - 8.5x_4 = 4.5. \end{cases}$	$x_2$	10.20	$\begin{cases} 12x_1 + 4.2x_2 - 0.8x_3 - 3.5x_4 = -5.4; \\ -4.1x_1 + 2.2x_2 - 0.16x_3 - 4.5x_4 = 1.6; \\ -1.6x_1 - 4.3x_2 + 8.4x_3 + 3.1x_4 = 12.2. \end{cases}$	$x_1$

### Control questions

1. List the functions designed to solve equations.
2. List the functions used to solve polynomial equations.
3. List the functions used to solve transcendental equations.
4. Specify the features of the use of functions that are used to obtain analytical and numerical solutions of equations.
5. Specify the function used to reduce the system equations.
6. List the modifications of the Solve [] function designed to solve systems of equations, and list the features of their use.
7. Describe the method of solving the system of equations by the matrix method.
8. Describe special cases of solving systems of equations.
9. Describe the method of localizing the roots of the equation.
10. Specify the value of the initial approximation parameter in the FindRoot [] function.

## Laboratory class № 5. Functions of mathematical analysis

**Goal:** study of the functions of mathematical analysis implemented in the program Mathematica.

### Brief theoretical information

#### Calculation of sums and products of series

The calculation of the sums of series can be carried out in analytical or numerical form. The function is used to calculate the amounts in analytical form Sum. The following Sum recording formats exist:

- Sum [f<sub>i</sub>, {i, imax}];
- Sum [f<sub>i</sub>, {i, imin, imax}];
- Sum [f<sub>i</sub>, {i, imin, imax, Δi}];
- Sum [f<sub>i</sub>, j, .. {i, imin, imax}, {j, jmin, jmax}],

where f is the summation element,

and, j - summation variables,

imin, imax - summation elements,

Δi is the step of changing the argument and.

If it is necessary to calculate the sum of the members of the series represented by the analytical function, from 1 to n, you must use the function Sum [f<sub>i</sub>, {i, imax}].

Function Sum [f<sub>i</sub>, {i, imin, imax}] calculates the sum of the values of the function f in the range of values of the argument imin ..imax with step 1. When using the function Sum [f<sub>i</sub>, {i, imin, imax, Δi}] step of changing the argument is given by the parameter Δi.

Function Sum [f<sub>i</sub>, j, .. {i, imin, imax}, {j, jmin, jmax}] calculates the sum of several variables.

Numerical calculation of sums is performed by the NSum function, which has the same modifications as the Sum function.

The product is calculated similarly to summation. To do this, use the functions:

- Product - for calculating products in analytical and numerical form;
- NProduct - to calculate products only in numerical form.

#### Calculation of the function boundary

Calculating the function boundary in the system Mathematica is done using the Limit function. The syntax of its record is as follows:

Limit [f(x), x-> x0].

The Limit function has a Direction option. The Direction option indicates the direction of approach to the border. Her record has two options:

Direction -> +1;

Direction -> -1.

A value of +1 indicates approaching the border on the left side, -1 - on the right side.

#### Schedule functions in power series

To decompose into a power series in the system Mathematica uses the following functions:

Series [f, {x, x0, n}] - decomposes the function f around the point x = x0 using n members of the series;

Series [f(x, y), {x, x0, nx}, {y, y0, ny}] - decomposes the function f into two variables x and y around the point (x0, y0) with the number of members nx and ny, respectively .

#### Calculation of derivative functions

The calculation of derivatives is carried out using the following functions:

D [f, x];

D [f, {x, n}];

D [f, x1, x2,...];

Dt [f, x];

Dt [f];

Derivative [n1, n2, ..] [f] - is a derivative of f [{x1, x2,...}] taken ni times xi.

Where f is the differentiated function,

x is the variable of differentiation,

x1, x2, .. - differentiation variables,

n is the order of the derivative.

### Methods for calculating integrals

#### Analytical methods

The integral in analytical form is calculated using the following built-in functions:

- *Integrate* [f(x),x] - calculates the indefinite integral of the function by the argument x;f(x)

- - calculates the definite integral of the function *Integrate* [f(x), {x, xH, xK}] f(x) on the variable with the lower and upper limits of integration. The limits of integration can be symbolic variables, numbers and even functions;xxHxK

- - Calculates the definite integral of the function of many variables with integration limits,,, *Integrate* [f(x, y, ...), {x, xH, xK}, {y, yH, yK, ...}] x, y, ... xHxKyHyK

#### Numerical Methods

Calculation of integrals in numerical form is necessary in the following cases:

- the original is not expressed through elementary functions;

- subintegral function is given in the form of a table;

- the analytical expression of the original is too complex.

The format of the NIntegrate function is as follows:

*NIntegrate*(f(x), {x, xH, xK}),,

where f(x) is a subintegral function;

x - argument of subintegral function;

xH, xK - lower and upper limits of integration.

The method of using the NIntegrate function does not differ from the method of calculating a definite integral in analytical form.

#### Calculation of multiple integrals

The calculation of multiple integrals in the Mathematica system is carried out by repeatedly using the Integrate function in analytical integration or NIntegrate in numerical integration.

#### Calculation of improper integrals

The Mathematica system allows you to compute integrals with infinite boundaries. The same functions are used as in the case of calculating integrals with finite limits. The infinity value is denoted by either the  $\infty$  symbol or the Infinity service constant.

#### Tabular integration

The subintegral function f(x) can be given in the form of a table. This is often necessary when conducting experimental research. In such cases, the calculation of the integral can be performed by the formulas of rectangles, trapezoids or parabolas. The solution can also be obtained by interpolating the function f(x) with its subsequent integration. Mathematica after version 6.0 changed the built-in ListIntegrate function to the integrated form Integrate [Interpolation []], which has the following features of use:

- *Integrate*[Interpolation;[{y1, y2, ..., yn}]] [x], {x, x1, x2}

- *Integrate*[Interpolation;[{y1, y2, ..., yn}, InterpolationOrder  $\rightarrow$  k]] [x], {x, x1, x2}

- *Integrate*[Interpolation [data] [x], {x, Min [xc], Max [xc]}];

The functions use the following notation:

-  $y_i$  - the value of the function  $y = f(x)$  in the  $i$ -node,  $i = 1, 2, \dots, n$ ;  $x_i$

-  $x$  is the argument of the function  $y = f(x)$ .

-  $x_1, x_2$  - final values of the argument;

-  $k$  is the interpolation order.

-;data =  $\{\{x_1, y_1\}, \dots, \{x_n, y_n\}\}$

-;  $x_c = \text{data}[[\text{All}, 1]]$

Using the function for tabular integration, you can skip writing the built-in character *InterpolationOrder*, then the default interpolation order will be  $k = 3$ .

### **Solving differential equations in Mathematica**

Mathematica system allows to solve in analytical and numerical form linear and nonlinear differential equations and systems. The solution of differential equations and systems is carried out with the help of built-in functions.

#### **Analytical methods**

Analytical methods for solving differential equations in the Mathematica system are implemented using two built-in functions:

*DSolve* [ $f, y[x], x$ ];

*DSolve* [ $\{f_1, f_2, \dots\}, \{y[x_1], y[x_2], \dots\}, \{x_1, x_2, \dots\}$ ].

Consider in detail these functions and give examples.

Function *DSolve* [ $f, y[x], x$ ] designed to solve the differential equation  $f$  regarding the function  $y(x)$  with an argument  $x$ . The function gives the general solution of the equation with the integrating constants that are denoted  $c[i]$ .

#### **Solution of differential equations under known initial conditions**

The following modification of the *DSolve* function is used to obtain the solution of differential equations under the known initial conditions:

*DSolve* [ $f(x, x_0), y[x], x$ ],,

where  $f(x, x_0)$  - differential equation in conjunction with the initial conditions;

$y(x)$  - desired function;

$x$  - independent variable.

#### **Solution of systems of differential equations in analytical form**

Systems of differential equations in analytical form are also solved using the built-in function *DSolve*, which in this case has the form:

*DSolve* [ $\{f_1, f_2, \dots\}, \{y_1(x), y_2(x), \dots\}, x$ ]

where  $f_i$  - the  $i$ -th equation of the system;

$y_i(x)$  - and the sought-after unknown;

$x$  - independent variable.

#### **Numerical methods for solving differential equations**

Numerical methods for solving differential equations in the Mathematica system are implemented using the following two built-in functions:

*NDSolve* [ $f, y[x], \{x, x_{\min}, x_{\max}\}$ ]

*NDSolve* [ $\{f_1, f_2, \dots, y_1(x_0), y_2(x_0), \dots\}, \{y_1[x], y_2[x], \dots\}, \{x, x_{\min}, x_{\max}\}$ ]

where  $f$  - differential equation and initial conditions;

$f_i$  - the  $i$ -th equation of the system of differential equations;

$y[x]$  - desired function;

$y_i[x]$ - the i-th desired function of the system of differential equations;

$y_i(x_0)$  - i-th initial condition;

$x_{\min}, x_{\max}$  - minimum and maximum value of the independent variable;

$x$  - the argument of the desired function.

The numerical solution functions of differential equations and systems have the `StartingStepSize` option, which determines the value of the initial integration step.

Numerical methods are most often used in cases where the equation in analytical form is not solved by the system or has no analytical solution. These are most nonlinear equations. Next, the built-in functions, methods of their implementation and examples are given in detail.

*Function* `NDSolve` [ $f, y[x], \{x, x_{\min}, x_{\max}\}$ ]

This function solves the n-th order differential equation by calculating the required function  $y(x)$  in the range of the independent variable  $x$  from  $x_{\min}$  to  $x_{\max}$ . The solution can be obtained in the form of a table or graph.

*Function* `NDSolve` [ $\{f_1, f_2, \dots, y_1(x_0), y_2(x_0), \dots\}, \{y_1[x], y_2[x], \dots\}, \{x, x_{\min}, x_{\max}\}$ ]

This function solves a system of n-th order differential equations by calculating the required functions  $y_1[x], y_2[x], \dots$  in the range of the independent variable  $x$  from  $x_{\min}$  to  $x_{\max}$ . The solution can be obtained in the form of tables or graphs.

Mathematica, like any other computer algebra system, is not ideal for solving differential equations. The obtained solution rarely coincides with the answer available in mathematical reference books. It is not uncommon for a built-in function to give no solution or to be erroneous, although the equation is quite simple.

When solving differential equations by numerical methods, unacceptably large errors can occur due to methodological errors and errors in choosing the integration step. It is always necessary to remember that at computer technologies of the decision of differential equations check of reliability of the received results is necessary.

### Work program

1. Calculate the sum of the series with a step  $\Delta n = 1$  according to the option specified in table. 1.

Table 1. The sum of the series

Var.	Number	Var.	Number	Var.	Number	Var.	Number
1.11	$\sum_{k=1}^n 2k - 1$	4.14	$\sum_{k=0}^{\infty} aq^k$	7.17	$\sum_{k=1}^{\infty} \frac{1}{k2^k}$	10.20	$\sum_{k=1}^{n-1} \sin\left(\frac{\pi k}{n}\right)$
2.12	$\sum_{k=1}^n k!k$	5.15	$\sum_{k=1}^{\infty} \frac{1}{(2k-1)^2}$	8.18	$\sum_{k=1}^{\infty} \frac{x^k}{k}$		
3.13	$\sum_{k=1}^n \frac{k^2 + k - 1}{(k+2)!}$	6.16	$\sum_{k=1}^{\infty} \frac{1}{4k^2 - 1}$	9.19	$\sum_{k=1}^{\infty} \frac{1}{x^k k}$		

2. Calculate the product of a series with a step  $\Delta n = 1$  according to the option specified in table. 2.

Table 2. The product of the series

Var.	Number	Var.	Number	Var.	Number	Var.	Number
1.11	$\prod_{k=2}^{\infty} \left(1 - \frac{1}{(2k-1)^2}\right)$	4.14	$\prod_{k=2}^{\infty} \left(1 - \frac{1}{k^2}\right)$	7.17	$x \prod_{k=1}^{\infty} \left(1 + \frac{x^2}{(k\pi)^2}\right)$	10.20	$x \prod_{k=1}^{\infty} \left(1 - \frac{x^2}{(k\pi)^2}\right)$
2.12	$\prod_{k=0}^{\infty} (1 + x^{2^k})$	5.15	$\prod_{k=1}^{\infty} \left(1 - \frac{1}{(2k)^2}\right)$	8.18	$2^{n-1} \prod_{k=0}^{n-1} \sin\left(x + \frac{k\pi}{n}\right)$		
3.13	$\prod_{k=1}^{\infty} \left(1 + \frac{(-1)^{k+1}}{2k-1}\right)$	6.16	$\prod_{k=1}^{\infty} \left(1 - \frac{4x^2}{\pi^2(2k+1)^2}\right)$	9.19	$x \prod_{k=1}^{\infty} \cos\left(\frac{x}{2^k}\right)$		

3. Calculate the boundary of the function according to the option shown in table 3.

Table 3. The boundary of the function

Var.	Border	Var.	Border	Var.	Border	Var.	Border
1.11	$\lim_{x \rightarrow \infty} (1+x)^{\frac{1}{x}}$	4.14	$\lim_{x \rightarrow 1} (1-x)^{\frac{1}{1-x}}$	7.17	$\lim_{x \rightarrow 0} \left(\frac{1-a^x}{ax}\right)$	10.20	$\lim_{x \rightarrow 2} \left(\frac{n^2 - n^x}{x-2}\right)$
2.12	$\lim_{x \rightarrow 0} \left(\frac{a^x + b^x}{2}\right)^{\frac{1}{x}}$	5.15	$\lim_{x \rightarrow \infty} (1-x)^{\frac{1}{1-x}}$	8.18	$\lim_{x \rightarrow \infty} \left(2a \frac{1-e^{-ax}}{2-e^{-ax}}\right)$		
3.13	$\lim_{x \rightarrow \infty} \left(1 + \frac{1}{x}\right)^{\frac{1}{x}}$	6.16	$\lim_{x \rightarrow 0} \left(\frac{1-e^{ax}}{\ln(1-x)}\right)$	9.19	$\lim_{x \rightarrow a} \left(\frac{x-a}{\ln(x-a+1)}\right)$		

4. Decompose the function into a power series and calculate its value at the point  $x_0$  according to the option given in table. 4.

Table 4. Power series of the function

Var.	Function	Membership	Schedule point	Var.	Function	Membership	Schedule point
1.11	$\arcsin(5x)$	5	$x_0 = 0.3$	6.16	$x/(1-x^2)$	7	$x_0 = 0.5$
2.12	$\ln(2x+1)$	7	$x_0 = 4$	7.17	$\ln((1+x)/(1-x))$	7	$x_0 = 0.9$
3.13	$\sin(3x^2)$	3	$x_0 = \pi/3$	8.18	$(1+x)^m$	8	$x_0 = 3, m = 5$
4.14	$\sin(x)/x$	5	$x_0 = \pi/4$	9.19	$\ln(x + \sqrt{x^2+1})$	8	$x_0 = 0.8$
5.15	$1 - e^x/x$	5	$x_0 = 10$	10.20	$\operatorname{tg}(x)$	6	$x_0 = 7\pi/8$

5. Calculate the derivative of the function at point  $x$ , according to the option shown in table. 5.

Table 5. Analytical expression of the function

Var.	Function	$x_1-x_2$	Var.	Equation	$x$	Var.	Equation	$x$
------	----------	-----------	------	----------	-----	------	----------	-----

1.11	$(\sin ax + \cos ax)^2$	0..2.5	4.14	$(x-1)/(x+1)$	-9 ..- 6	7.17	$x \cos x$	0.. $\pi$ / 4
2.12	$ax^3 + bx - c$	1.5..3.4	5.15	$\sqrt{a+bx^2}$	2.3- 12.2	8.18	$\sin(x)^2 \cos(x)^4$	0.. $\pi$ / 12
3.13	$xe^{-ax}$	-3 ..- 1	6.16	$\sin(ax)/(x+b)$	8.4- 10	9.19	$x^2 \arcsin(x)$	0.. $\pi$ / 6
						10.20	$x^2 \ln x$	0..e

6. Calculate the integral of the function in analytical form and find its value within the given limits  $x_1$ - $x_2$  integration in accordance with the option specified in table. 5.

7. Calculate the value of the integral by the numerical method according to the variant specified in table. 5.

8. Solve the differential equation by analytical and numerical methods in accordance with the option specified in table. 6.

Table 6. Differential equation

Var.	Function	Var.	Equation	Var.	Equation
1.11	$y'' - 0.5y' - 0.5y = e^{0.5x}$ $y(0) = 1, y'(1) = 0$	4.14	$(1+x)y'' + y' = 0$ $y(0) = -2, y'(0) = 0$	7.17	$8y'' + 4y' + 4y = 0$ $y(0) = 1, y'(0) = 3$
2.12	$y'' - 3y' + 2y = x^2 + 3x$ $y(0) = -1, y'(0) = 3$	5.15	$y' = 0.5xy$ $y(0) = 1$	8.18	$y'' - 3y' = x^2 + 3x$ $y(0) = 1, y'(0) = 3$
3.13	$y' = -2y/(100+x)$ $y(0) = 0$	6.16	$8y'' + 2y' - 3y = 0$ $y(0) = -1, y'(0) = 1$	9.19	$y'' - 2y' + y = xe^x$ $y(0) = 1, y'(0) = 1$
				10.20	$y' = -xy/(1+x^2)$ $y(0) = 0$

9. Solve the differential equation by the analytical method under the known initial conditions in accordance with the option specified in table. 7.

Table 7. Differential equation

Var.	Function	Var.	Equation	Var.	Equation
1.11	$y' - \frac{4}{x}y = x\sqrt{y}, y(1) = 1$	4.14	$y'' + y = 4xe^x,$ $y(0) = -2, y'(0) = 0$	7.17	$y'' + 9y = 6\cos(3x),$ $y(0) = 1, y'(0) = 3$
2.12	$y'' + 3y = -2y,$ $y(0) = -1, y'(0) = 3$	5.15	$y'' + y = \sin(x),$ $y(0) = 1, y'(0) = 2$	8.18	$y'' - y = e^x,$ $y(0) = 0, y'(0) = -1$
3.13	$y''' = -8y,$ $y(0) = 0, y'(0) = 1, y''(0) = 1$	6.16	$y'' + 4y' + 4y = 8xe^{2x},$ $y(0) = -1, y'(0) = 1$	9.19	$y'' + 2y' - 3y = x^2e^x,$ $y(0) = 1, y'(0) = 1$
				10.20	$y'' - 4y' + 5y = 0,$ $y(0) = 0, y'(0) = 1$

10. Solve the system of differential equations by the analytical method according to the variant specified in table. 8.

Table 8. Differential equation

Var.	Function	Var.	Equation	Var.	Equation
1.11	$\begin{cases} x' = y + 2e^t; \\ y' = x + t^2. \end{cases}$	4.14	$\begin{cases} x' = 2x - 4y + 4e^{-2t}; \\ y' = 2x - 2y. \end{cases}$	7.17	$\begin{cases} x' = 5x - 3y + 2e^{3t}; \\ y' = x + y + 5e^{-t}. \end{cases}$



2.12	$\begin{cases} x' = y - 5\cos(t); \\ y' = 2x + y. \end{cases}$	5.15	$\begin{cases} x' = 2x - 4y + 4e^{-2t}; \\ y' = 2x - 2y. \end{cases}$	8.18	$\begin{cases} x' = 2x + y + e^t; \\ y' = -2x + 2t. \end{cases}$
3.13	$\begin{cases} x' = 3x + 2y + 4e^{5t}; \\ y' = x + 2y. \end{cases}$	6.16	$\begin{cases} x' = 2y - x + 1; \\ y' = 3y - 2x. \end{cases}$	9.19	$\begin{cases} x' = x + 2y; \\ y' = x - 5\sin(t). \end{cases}$
				10.20	$\begin{cases} x' = 2x - 4y; \\ y' = x - 3y + 3e^t. \end{cases}$

### Control questions

1. Specify the name of the function used to calculate the sum of the series and list its parameters.
2. Describe the purpose and parameters of the Limit [] function.
3. List the parameters of the Series [] function.
4. List the functions of integration and differentiation of functions.
5. Describe the procedure for calculating the integrals of functions given in the table.
6. List the modifications of the functions designed to solve differential equations.
7. List the modifications of functions designed to solve systems of differential equations.
8. Describe the features of solving differential equations with known and unknown initial conditions.
9. Describe the features of solving differential equations by numerical and analytical methods.
10. Describe the method of solving the differential equation using the NDSolve function.

## Laboratory class № 6. Computer interpolation technologies in Mathematica environment

**Objective:** study of interpolation functions of the program Mathematica.

### Brief theoretical information

#### Interpolation, accurate in nodes

In Mathematica, interpolation, accurate in nodes, can be implemented by the following methods:

- universal;
- using the universal functions `InterpolatingPolynomial` and `Interpolation`.

#### Universal method

The universal method requires the solution of systems of algebraic equations, which were obtained on the basis of the data of the function, which is presented in the form of a table or matrix.  $y = f(x)$

The problem of interpolation, which is exact in nodes, can also be solved using the matrix method of solving a system of linear equations. Checking the correctness of the solution of the problem is performed by tabulating the obtained formula and comparing the results with the original data, for example using a function where `Table[f(x), {x, xH, xK, h}]`  $f(x)$  – tab function  $x$  – function argument, the initial and final value of the argument, the step of the table, if, then it can be neglected.  $f(x)$   $x_H, x_K - h - h = 1$

Calculate the absolute  $\varepsilon$  and relative  $\delta$  RMS interpolation error by the following formulas:

$$\varepsilon = \sqrt{\frac{\sum_{i=1}^n \Delta_i^2}{n}}, \delta = \frac{\varepsilon}{y_{\min}} \cdot 100\%, \quad (1)$$

where  $\Delta_i = y(x_i) - \varphi(x_i)$  is the difference between the values of the tabulated function  $y(x_i)$  and the interpolation function  $\varphi(x_i)$ ,  
 $n$  is the number of values of the tabulated function,  
 $y_{\min}$  - the minimum value of the function  $y(x)$ .

#### Function `InterpolatingPolynomial`

To interpolate polynomial functions use a function that has the following format: `InterpolatingPolynomial`

`InterpolatingPolynomial[z, x],,`

where  $z$  – source data matrix,

$x$  – function argument  $z$ .

#### Function `Interpolation[data]`

This function allows you to solve the problem of interpolation over data (data) in the range of arguments given by this data. The approximation function is unknown to the user. The data are given in the form of a matrix of the function  $y = f(x)$ . When you enter this function, Mathematica does not issue an interpolation function, but a new function:

`InterpolatingFunction[{{xH, xK}}, <>],`

where  $x_H, x_K$  – the range of arguments of the interpolation function.

If you now enter the value of the argument from the range  $x_H - x_K$ , the answer will be the value of the function at a given value of the argument.

### **Interpolation by nonlinear functions**

If the interpolation function is nonlinear, then two methods are used to determine its coefficients by the exact method in nodes:

- creation and solution of a system of nonlinear equations;
- linearization of nonlinear interpolation function by coordinate transformation.

### **Interpolation approximated in nodes**

Interpolation, approximated in nodes (approximation), is carried out by the criterion of minimum standard error (least squares method). Implemented by the Mathematica system using the function *Fit*. Function *Fit* looks like:

$Fit[\{\{M\}\}, \{XX\}, x]$ ,

where  $M$  - is the matrix of the original data,

$XX$  - list of basic variables,

$x$  - is the argument of the function.

Method of interpolation using a function  $Fit[\{\{M\}\}, \{XX\}, x]$  is implemented by performing the following actions:

1. Introduction of a matrix of initial data with assignment to it of the unique name, for example,  $M$ .
2. Introduction of basic variables  $X$ .
3. Enter the function  $Fit[\{\{M\}\}, \{XX\}, x]$ .

### **Padé approximation**

The Padé approximation is used to interpolate a function given analytically by a fractional-rational function.

The Padé function has the form:

$PadéApproximant[f(x), \{x, r, \{n_1, n_2\}\}]$ ;

where  $f(x)$  –function, given in analytical form,

$x$  –argument of the function  $f(x)$ ,

$r$  –point near which the approximation function is valid,

$n_1$  –degree of the polynomial of the numerator,

$n_2$  –degree of the polynomial of the denominator.

### **Spline interpolation**

The spline interpolation implementation function is located in the *SplineFit* subpackage of the *NumericalMath* package. Cubic spline interpolation provides high accuracy of the mathematical model. Before use, you must connect this package with the following line: `<< Splines` a6o Needs["Splines`"]`. The function has the following format:  $SplineFit[F, type]$ ,

Where  $F$  is the data presented as a matrix,

$type$  - type of approximation, by default - approximation by cubic splines (Cubic). Approximation with Bezier and CompositeBezier splines is also possible.

## Work program

1. Tabulate functions  $y_1(x)$  with a step  $\Delta x_1$ ,  $y_2(x)$  with a step  $\Delta x_2$  in a given range according to the option given in table. 1.

Table 1. Function variants

Variant	Function	$\Delta x$	Range	Variant	Function	$\Delta x$	Range
1,11	$y_1 = 0,1x^5 - x^4 - 2x^2 + 3x + 5$	2	-5:11	6,16	$y_1 = 0,01x^7 - x^5 + 3x^3 + 4x^2 + 2$	1	0:10
	$y_2 = \ln(5x)$	1	1:10		$y_2 = -1/3x^2$	1	1:10
2,12	$y_1 = 0,5x^4 - 2x^3 + 6x - 3$	1	-3:5	7,17	$y_1 = -0,06x^6 + x^5 - 4x^3 + x^2 + 1$	2	0:20
	$y_2 = e^{-5x}$	5	0:30		$y_2 = \cosh(3x)$	1	-3:3
3, 13	$y_1 = 0,1x^6 - 3x^5 + 60x - 6$	1	-3:4	8,18	$y_1 = -0,06x^6 + x^5 - 4x^3 + x^2 + 1$	0,5	-3:3
	$y_2 = 10e^{-x^2}$	1	-5:5		$y_2 = \sinh(3x)$	1	-3:3
4,14	$y_1 = -0,1x^5 - 3x^4 + 20x^2 - 6$	1	-3:3	9,19	$y_1 = x^8 - x^3 - 1000x^2 + x + 5$	0,5	-3:3
	$y_2 = \sin(3x)$	1	-5:5		$y_2 = \ln(5x) + x$	1	1:10
5,15	$y_1 = x^4 - 15x^3 + 10x^2 + 5x + 12$	2	-5:15	10,20	$y_1 = -x^4 + 3x^3 + 4x^2 + 5x + 5$	0,5	-2:4
	$y_2 = \cos(3x)$	1	-5:5		$y_2 = 10e^{-x^2} + x^2$	0,5	0:4

2. Using arrays of tabulated values to interpolate functions:
- InterpolatingPolynomial;
  - $Fit[\{\{M\}\}, \{\{X\}\}, x];$
  - $Pade[f(x), \{x, r, n1, n2\}];$
  - SplineFit[F, type].
3. Calculate the relative interpolation error using each of the functions and enter the data in the table.
4. Draw conclusions about the peculiarities of the use of interpolation functions.

## Control questions

1. Describe the features of using interpolation methods that are exact in nodes and interpolation that is approximate in nodes.
2. Describe the universal method of interpolation of functions, exact in nodes.
3. Indicate how the accuracy of interpolation methods is evaluated.
4. Specify the features of using the function InterpolatingPolynomial[[]].
5. Specify the features of using the function Interpolation[[]].
6. Describe the method of interpolation by the method of linearization of nonlinear functions.
7. Specify the functions that are used for interpolation approximated in nodes.
8. Name the methods of interpolation by nonlinear functions.
9. List the parameters of the approximation function.
10. Indicate what in practice is the relationship between the degree of the interpolation polynomial  $n$  and the number of data points of the function  $N$ .

## Laboratory class № 7. Fundamentals of programming in Mathematica

**Objective:** to gain programming skills in the Mathematica environment.

### Brief theoretical information

#### Mathematica as a programming system

In the Mathematica system it is possible to solve many problems without the use of programming. However, all system tools (alphabet, letters, numbers, operators, and special characters) are part of an ultra-high-level object-oriented programming language. According to its capabilities in performing mathematical and scientific and technical calculations, this language has significant advantages over conventional programming languages - Pascal, C, C ++.

The Mathematica system contains a large number of functions, many of which implement mathematical transformations and modern computational methods, both numerical and analytical. The programming language of the Mathematica system is a default interpreter and is not intended for creating executable files. However, individual expressions can be compiled using the Compile function, which is useful when you need to increase the speed of calculations. The language of the Mathematica system allows you to implement most types of programming: functional, structural, object-oriented, mathematical, logical, recursive, etc.

The core of the Mathematica system has a functional structure. The language of the system allows you to break programs into separate modules (blocks), procedures and functions with local variables. Object-oriented programming is based on a generalized concept of an object. In the Mathematica system, objects can be mathematical expressions, input and output data, graphs and drawings, sounds, etc. Three main properties are closely connected with the concept of object: encapsulation, inheritance and polyformism. All of them are inherent in the objects of the Mathematica system and do not require special tools for their implementation. Encapsulation means combining in one object both data and methods of their processing. Inheritance means that each object derived from other objects inherits their properties. Polyformism - a property that allows you to transmit a number of message objects, which will be processed by each object according to its individual characteristics.

The programming language of Mathematica is specially designed to implement any of these approaches to programming, as well as a number of others, such as recurrent programming, using which the next step of the calculation is based on the data obtained in the previous steps. Possibly recursive programming is when the function in the general case repeatedly addresses itself. Mathematica language tools allow you to implement elements of visual-oriented programming. Mathematica allows you to create palettes and panels with various buttons that allow you to control the program or enter new program objects.

#### Fundamentals of functional programming

The principle of functional programming involves the use of only functions when writing. At the same time repeated embedding of functions in each other is possible. In some cases, especially during symbolic transformations, there is a mutual recursion of functions, accompanied by almost unlimited deepening of recursion and increasing complexity of the expressions processed by the system. The concept of a function is associated with the mandatory return of some value in response to a call to a function.

Returning functions to some values allows you to use them along with operators to compose mathematical expressions. Functions are divided into internal and user-defined functions.

### User functions

The process of creating a function in Mathematica is similar to other programming languages. For example, the function for reducing  $x$  to the power of  $n$  could be defined as follows:

```
powerxn [ x , n ] := x ^ n
```

However, this feature is inoperable. The reason for this is that in Mathematica the symbols  $x$  and  $n$  are ordinary symbols and cannot accept formal parameters. For implementation it is necessary to use variables-samples having after their names underscores. Samples can be formal parameters of functions and perceive actual values and parameters. Thus, it would be correct to write the user function in the form:

```
powerxn [ x_ , n_ ] := x ^ n
```

Variables of the list of parameters, after the name of which there is a sign " $_$ ", are local in the body of the function or procedure with parameters. In their place the actual value of the corresponding parameter is substituted.

Note that the variable  $t$  in the function  $f$  is global, which explains the result of the last operation. The use of global variables in the body of the function is quite possible, but creates a so-called side effect, in this case changes the value of the global variable  $t$ . To eliminate side effects, it is necessary to use samples and other special ways to create functions.

Function parameters can be lists, provided they can be combined. Once created, user functions can be used according to the same rules as built-in functions.

### Clean features

Sometimes it may be necessary to use a function only at the time of its creation. This function is represented only by an expression without a name, which led to its name. To create such an object is a built-in function `Function`, which is used in the form:

`Function [body]` – creates a pure function with the body;

`Function [{ x }, body]` – creates a pure function of the parameter  $x$  with the body;

`Function [{ x1, x2, ... }, body]` – creates a pure function of a number of parameters  $x_1, x_2, \dots$  with body.

To calculate the function thus created, a list of parameters is written in square brackets after it.

### Anonymous functions

The so-called anonymous functions have an extremely compact form of setting functions. They have neither a name nor a common definition, they are written in expressions of a special kind. In this expression instead of variables use designations  $\#$  (for one variable),  $\# 1, \# 2, \dots$  for a number of variables. End the body of the function with the symbol  $\&$ . If it is necessary to calculate the value of the function, then after writing it in square brackets indicate a list of actual parameters.

### Superposition of functions

Functional programming involves the use of superposition of functions. Functions are used for its implementation:

`Nest[expr, x, n]` – applies an expression (function) to a given argument  $x$   $n$  times.

`NestList[f, x, n]` – returns the list of uses of the function  $f$  to the specified argument  $x$   $n + 1$  times.

Fold [f, x, list] – returns the next item in FoldList[f, x, list].

FoldList [f, x, {a, b, ...}] – returns  $\{x, f[x, a], f[f[x, a], b], j\}$ .

ComposeList[{f1, f2, ...}, x] – generates a list in the form {x, a[x], a[a[x]], ...}.

### Functions FixedPoint and Catch

In functional programming, instead of the cycles described below, you can use the following function:

FixedPoint [f, expr] – calculates expr and applies the expression f to it until the result is repeated.

FixedPoint [ f , expr , SameTest\_ > comp ] - calculates expr and applies the expression f to it until the next two results are true.

Another feature of this kind - Catch []:

Catch [expr]– calculates expr, before the first execution of the Throw [value] function, then returns value.

Catch [expr, form] – calculates expr before the first execution of the Throw function [value, tag], then returns value.

Catch[expr, form, f ] – returns f [value, tag] instead of value.

### Implementation of recursive and recurrent algorithms

An important place in solving many mathematical problems is the implementation of recursive and recurrent algorithms. Consider a typical example of the implementation of an iterative recurrent algorithm for calculating the square root of the expression f (x) at the initial value of  $x_0 = a$ , according to the following formulas of Newton's method:

$$x_0 = a \quad x_n = x_{n-1} - f(x_{n-1}) / f'(x_{n-1}).$$

This function can be written as follows:

**newtoniter[f\_, x0\_, n\_ ] := Nest[ ( # - f [#] / f'[#] ) &, N[x0], n]**

Recursive algorithms use calculations in which the body of the function refers to itself. Mathematica allows this possibility.

### Fundamentals of procedural programming

The basis of procedural programming is the concept of procedure as a complete software module and typical controls: cycles, conditional and unconditional transitions, etc. Although, the programming of Mathematica systems in this case remains functional, because the elements of procedural programming are given in the form of functions. However, it is possible to use traditional programming tools - conditional expressions, loops, procedures, etc.

Procedures are completely independent software modules, have their own identifier and perform some sequence of operations.

To create full-fledged procedures and functions, the Block structure is used in two modifications:

Block[{x, y, ...}, procedure] – procedure with declaration of the list of local variables x,y,...

Block[{x = x0, y=y0, ...}, procedure] – procedure with declaration of the list of local variables x, y,... with initial values.

ignored outside the block. If the variable u has not been defined before use in the function, it remains undefined. And if it had some value before (for example, 123456 in our case), then after leaving the procedure it will have this value.

### Organization of cycles

#### Type cycles Do

Cycles of this type have several modifications:

Do[expr, {imax}] – calculates the expression expr imax times.

Do[expr, {i, imax}] – calculates the expression expr with the variable and, which alternately takes values from 1 to imax in increments 1.

Do[expr, {i, imin, imax}] – calculates the expression expr with the variable and, which alternately takes values from imin to imax in increments 1.

Do[expr, {i, imin, imax, di}] – calculates the expression expr with the variable and, which alternately takes values from imin to imax in increments di.

Do[expr, {i, imin, imax}, {j, jmin, jmax}, ...] – calculates the expression expr using a series of nested loops with variables j, i, etc.

### **Type cycles For**

Another type of For loop is implemented by a function:

For [start, test, incr, body]

In it the variable of a cycle at first is appropriated value start, then cyclically changes from this value to value body with a step incr; and so on as long as the test condition is true.

When the condition becomes false, the cycle ends.

### **Type cycles While**

Cycle recording form: While [test, expr] In this type of loop, the value of expr is calculated as long as the test condition is true.

The device of local variables in this type of cycles is not used.

### **Directives for interrupting and continuing cycles**

The following function directives can be used in these loop types and in other structures:

Abort [ ] – stops calculating with the message \$ Aborted.

Break [ ] – exits the body of the loop or the level of nesting of the program containing the operator (loops such as Do, For and While or the body of the operator - Switch switch). The operator returns a Null value.

Continue [ ] – sets the transition to the next step of the current cycle Do, For or While.

Interrupt [ ] – interrupts calculations with the possibility of updating them using a dialog box.

Return [ ] – aborts execution with Null return.

Return [ expr ] – interrupts execution with the return of the value of the expression expr.

### **Conditional expressions and unconditional transitions**

#### **Function If**

As in most programming languages, conditional expressions are specified using the operator or the If function. The Mathematica system has the following modifications to the If function:

If [condition, t, f] – returns the result t if the condition calculation is true, and f if the result is false.

If [condition, t, f, u] - returns the result u if the result of calculating the condition is neither true nor false.

#### **Functions - switches**

To organize several branches in the system Mathematica use operators - switches Which i Switch :

Which [ test<sub>1</sub> , value<sub>1</sub> , test<sub>2</sub> , value<sub>2</sub> , ...] - calculates each test condition in order and returns the value<sub>i</sub> of the first test<sub>i</sub> condition that was true.



Switch [ expr , form<sub>1</sub> , value<sub>1</sub> , form<sub>2</sub> , value<sub>2</sub> , ...] - calculates the value of the condition expr, then compares it sequentially with each expression form<sub>i</sub> and returns the value value<sub>i</sub>, the first match of the expression form<sub>i</sub> with the condition expr.

### Unconditional transitions

The unconditional transition operator Goto [tag] creates a transition to the place of the program marked with the label Label [tag]. Forms Goto [expr] and Label [expr] are also possible, where expr is a specific expression.

### Work program

1. Write a program to solve the problem according to the option given in table.1.

option.	Task
1	For a given number k to form a list of elements and k, where and is an integer in the range 1-10. Plot a graph based on the values in the list. Calculate the mean and standard deviation of all array elements.
2	Two numbers and j are introduced. Of the two numbers, choose the largest and use it to create a list. Let the larger number be the number j. Then the first element of the list is equal to the number 1-i / j. The variable j is assigned a new value $j = j (1-i / j)$ . Find the largest number of the two and and j. Let the larger number be the number j. Then the second element of the list is equal to the number 1-i / j, etc. Write a program that forms 10 list items.
3	Calculate the integral of a given function in the range $x = 0-10$ , with a step $\Delta x = 0.1$ by the trapezoidal method.
4	Construct Pascal's triangle.
5	Given an integer matrix of size M x N. Find the number of its rows and columns, all elements of which are different from each other.
6	Given a matrix of size M x N. Print the number of its row that contains the maximum number of identical elements.
7	Given a matrix of size M x N. Find the element that is the maximum in its row and the minimum in its column. If such an element missing output 0.
8	Given a set of N real numbers. Check if they form ascending sequence. If form to deduce 1, differently - 0.
9	Write a program that finds the decomposition of a natural number into prime factors. The result of the program is the number of prime factors N and their values in descending order of value.
10	Write a program that cyclically shifts list items to k positions. If k - positive shift is carried out to the right, otherwise - left.
11	Write a program that changes the order of the elements list on the reverse.

12	Write a program that sums the elements of the matrix from line n1 to line n2, starting with column k1, ending with column k2.
13	Write a program that removes from the matrix a row and a column that contains an element of value k.
14	Find the positions of the elements of the array, the difference between the values of which is minimal.
15	Given an integer array. Find the largest number of it identical elements.

### Control questions

1. Name the capabilities of the programming language of the Mathematica system.
2. Describe the structure of the software environment of the Mathematica system.
3. Specify the features of creating user functions.
4. Note the features of the functional programming style.
5. Note the features of the procedural programming style.
6. List the functions for creating loops.
7. Name the functions for creating conditional expressions and unconditional transitions.
8. Specify the purpose of the FixedPoint [] and Cath [] functions.
9. Specify the functions of organizing loops that have local variables - iterators, and which are global.
10. Define the term "pure function".

## Laboratory class № 8. Graphical input and output objects

**Objective:** gaining skills in creating graphical I / O objects Mathematica.

### Brief theoretical information

I/O in Mathematica is organized using the FrontEnd interface processor. Additionally, the system implements a number of additional I / O functions:

Input[] – stops the system and returns the value of the expression that will be entered in the dialog box (used to organize the dialog input);

Input[«comment»] – works similarly to the previous function, in addition displays in the dialog box "comment";

InputString[ ] – reads data and saves it in the form of a character string;

InputString[«комента»] – works similarly to the previous function, in addition displays in the dialog box "comment".

StylePrint[expr] – creates a new cell in the current document with the default style and enters the expression expr;

StylePrint[expr, «style»] – creates a new style cell in the current document and enters the expression expr;

Print[expr] – displays the value of the expression expr, together with the function Input [] can be used to organize a dialogue;

Print[«prompt», expr] – displays a text comment in quotation marks, and then the value of the expression expr.

These functions are enough to organize the simplest dialogue with the program

When the Input [] function is executed, a dialog box appears in the center of the screen. The window displays the query, which is specified in quotation marks as a parameter of the Input [] function. After entering the desired value (in the general example, this is the radius of the circle), the Input [] function returns the entered value and it is assigned to the variable R.

The Print [] function then displays the calculated value of the circle length with a short comment.

### Data output format

Mathematica implements a number of functions to customize the presentation format. The following functions are most often used:

AccountingForm[expr] – derives all the numbers contained in the expression expr, in the accounting form of presentation;

CForm[expr] – outputs the expression expr in the language format C;

EngineeringForm[expr] – derives the real numbers of the expression expr in engineering form;

FortranForm[expr] – outputs the expression expr in the form adopted for the Fortran language;

FullForm[expr] – outputs the full form of the expression expr without using special syntax;

InputForm[expr] – outputs the expression expr in input form;

NumberForm[expr, n] – outputs the expression expr in the form of a real number to the nearest n digits;

OutputForm[expr] – outputs expr at once in the standard initial form of the Mathematica system;

ScientificForm[expr] – outputs the expression expr in a scientific format;  
 TeXForm[expr] – outputs the expression expr in the form of the TeX language, which focuses on the layout of texts with mathematical formulas;  
 TextForm[expr] – outputs the expression expr in plain text format;  
 TreeForm[expr] – outputs the expression expr with a demonstration of different levels of expression.

## GUI elements

Mathematica tools allow you to create objects GUI (Graphic User Interface) laptops, which makes the latter much clearer and easier to use.

### Single-coordinate sliders

Sliders allow you to smoothly or discretely change the value of a particular variable. Listing 4 shows the slider repeating of three sliders with different parameter values.

Slider [x] is a slider with an x value from 0 to 1.

Slider [x, {xmin, xmax}] is a slider with an x value from xmin to xmax.

Slider [x, {xmin, xmax, dx}] - slider with value x from xmin to xmax with step dx.

Slider [x, {{e1, e2,...}}] - is a slider in which the same intervals correspond to sequential settings.

Slider [x, {{{e1, w1}, {e2, w2},...}}] - slider uses relative width intervals  $w_i$  for  $e_i$ .

### Two-coordinate sliders

To construct surfaces that describe the functions of two variables use two-coordinate sliders. They are created by the Slider2D function.

Slider2D [x, y] - 2D slider with changing values of x and y from 0 to 1.

Slider2D [Dynamic [pt]] - 2D slider with changing values of dynamic variable pt.

Slider2D [pt, {min, max}] - 2D slider with change of pt values from min to max. Slider2D

[pt, {min, max, d}] - 2D slider with change of pt values from min to max with step d.

Slider2D [pt, {{xmin, ymin}, {xmax, ymax}}] - 2D slider with change of pt values from xmin to xmax in x coordinate and from ymin to ymax in y coordinate.

Slider2D [pt, {{xmin, ymin}, {xmax, ymax}, {dx, dy}}] - 2D slider with change of pt values from xmin to xmax on x coordinate and from ymin to ymax on y coordinate with steps dx and dy respectively.

The engine of such a slider can move the mouse in any direction. The slider returns two numeric values depending on the position of the engine. These values can be used to calculate the functions of two variables and plot graphs of surfaces, three-dimensional figures, parametrically given graphs, etc.


### CheckBox graphical user interface element

It is often necessary to perform calculations provided that some options are set, for which it is convenient to use the CheckBox element, which is created by the CheckBox function [].

Checkbox[x] – displays a checkbox for x that displays  when x True, and  when x False.

Checkbox [Dynamic [x]] - displays a checkbox for the dynamic variable x, which changes the value of x when you click on the checkbox.

Checkbox[x, {val1, val2}] - displays a checkbox for x that display  when xval2, and  when x val1.

Checkbox[x, {val1, val2, val3, ...}] - displays a checkbox that switches between values *vali* and displays  for *vali* at *i*.

### Locator

A locator is an object that has the shape of a dot in a graphics window and returns its coordinates. This object is represented by a painted circle in the middle of the intersection, which has a bright dot in the middle. The locator is moved with the mouse. To create it, use the function.

Locator [{x, y}] - displays the locator in the position {x, y}.

Locator [Dynamic [pos]] - displays the locator with a dynamic change in the value of position, the value changes when you change the position of the locator.

Locator [{x, y}, obj] - Displays obj as a locator object. Locator [{x, y}, None] - Displays nothing visual as a locator object.

Locators are often used to plot points.

### Mouse control functions

Sometimes it is necessary to determine the coordinates of the mouse cursor, which uses the MousePosition function.

The coordinates of the mouse are displayed in the format of integers. Adding MousePosition to the Dynamic parameter list allows you to display the mouse cursor coordinates as a list of current coordinates.

The Opener [x] or Opener [Dynamic [x]] function responds to each mouse click on a black triangle. However, if you click on it, a list will appear.

Toggler function [x]:

Toggler [x] - Displays a cyclic switch that switches the value of x between True and False.

Toggler [Dynamic [x]] - Displays a cyclic switch that switches the value of the dynamic variable x between True and False.

Toggler [x, {val1, val2, ...}] - displays a cyclic switch that switches the value of x between a sequence of values of *vali*.

Toggler [x, {val1\_> pict1, val2\_> pict2, ...}] - displays a cyclic switch that switches the value of x between a sequence of *vali* values, where *vali* values are displayed via *picti*.

Toggler [x, vlist, dpict] - displays *dpict* if x is not equal to any of the values *valid* from the *vlist*.

AutoAction-> True toggles the value when dragging the mouse through the result.

### Button with the inscription

Often some actions must be performed when pressing the button with the inscription. To create such a button use the function Button [label, action], where the label-name of the button, action-action performed after pressing the button. In the list of its parameters indicate the line with the name of the button and the expression that is performed by conditions for activating the mouse button.

### Manipulator

A manipulator is an object similar to a slider, but has greater functionality and visual capabilities. The manipulator is created using the function: Manipulator [x] - displays the manipulator, which sets x from 0 to 1.

Manipulator [Dynamic [x]] - displays a manipulator that sets the dynamic variable x from 0 to 1.

Manipulator [x, {xmin, xmax}] - displays a manipulator that sets x from *xmin* to *xmax*.

Manipulator [ $x$ , { $x_{min}$ ,  $x_{max}$ ,  $dx$ }] - displays a manipulator that sets  $x$  from  $x_{min}$  to  $x_{max}$  in steps of  $dx$ .

A distinctive feature of the manipulator is a characteristic button in the form of a gray rectangle with a "+" in it. When you activate this button with the mouse, a panel with slider controls appears under the manipulator slider. By means of bodies (buttons) of management of the manipulator it is possible to start it and to provide automatic movement of the slider engine, it is possible to change the direction and speed of movement of the slider, to make its stop. As the slider approaches the start and end points, the engine has a characteristic shadow.

### **Rotator of the radius vector**

In some cases, for example, when plotting functions in the polar coordinate system, you need an object that specifies the angle of rotation of the radius vector. Such object – turning angle setter – is setting with function `DynamicModule[{ $x$  = RandomReal[{0,50}], {Experimental`AngularSlider[Dynamic@ $x$ ], Dynamic@ $x$ }}`. In fig. 6 shows an example of the application of this function.

The task is a graphic object in the form of a circle, inside which is placed a radius vector. It can rotate in one direction or another with the mouse, which leads to a change in the angle set by the unit. In the example shown in Fig. 6, projections of the end of the radius vector on the coordinate axis are constructed. These projections are known to have sinusoidal functions. The angle corresponding to the current position of the radius vector is indicated by dots on the graphs of sinusoidal functions.

### **Drop-down menu**

The drop-down menu is another widely used object for building a graphical interface. It is created by function `ActionMenu[name, {lbl1:>act1, lbl2:>act2, ...}]`. The parameters of the function are a line with an inscription on the button and a list of names of menu items and actions performed when activating the corresponding menu items.

### **Data entry panel**

The expression input panel is used for interactive input of an arbitrary expression, for example, to build its graph. It is entered by the Panel function.

`Panel[expr]` – shows a panel containing *expr*.

`Panel[expr, title]` - shows a panel called *title*.

`Panel[expr, title, pos]` – places the title in a position defined by *pos*.

`Panel[expr, {title1, title2, ...}, {pos1, ...}]`- places the title *title<sub>i</sub>* in the *pos<sub>i</sub>* position.

`Panel[]`.

### **Radiobuttons and settings menu**

The `RadioButton [ $x$ ,  $val$ ]` function creates a so-called circle radio button. The result of the function can be used for software input of an action.

Another function in a number of recording formats specifies the construction of the settings menu has the following recording formats:

`SetterBar[ $x$ , {val1, val2, ...}]`- displays the settings menu  $x$  with settings *val<sub>i</sub>*.

`SetterBar[Dynamic[ $x$ ], {val1, val2, ...}]`- displays the settings menu for the dynamically variable  $x$  with settings *val<sub>i</sub>*.

`SetterBar[ $x$ , {val1->lbl1, val2->lbl2, ...}]`- displays the settings menu  $x$  with settings *val<sub>i</sub>*, where the name *val<sub>i</sub>* defined as *lbl<sub>i</sub>*.

Another option for building a settings menu is set by the function:

Setter[*x*, *val*]- displays the setting at which the variable *x* takes the value *val*. The button is called *val*, when you press it the variable *x* takes the value *val*.

Setter[Dynamic[*x*], *val*]- displays the setting at which the dynamic variable *x* takes the value *val*. The button is called *val*, when you click on it, the dynamic variable *x* takes the value of *val*.

Setter [*x*, *val*, *label*] - displays the setting at which the variable *x* takes the value *val*. The button is called the *label*.

Setter[*x*, {*val1*, *val2*,...}, *label*]- displays the setting at which the variable *x* takes the value *val<sub>i</sub>*. The button is called the *label*.

### **Color slider**

To change the color of GUI objects, it is convenient to use the slider, which is set by the following function: ColorSlider[*color*], ColorSlider[Dynamic[*color*]], ColorSlider[[]].

The color of the first square is the current color. To the left of it is the color selection panel. To select a color, simply place the mouse cursor on the desired color of the panel and press the left mouse button. The color of the control square will be similar to the selected one.

### **Starter mechanism**

The Trigger function simulates the operation of the trigger. It has several forms of writing:

Trigger[Dynamic[*u*]]- displays the trigger, at startup of which the variable *u*

varies from 0 to 1. Trigger[Dynamic[*u*], {*umin*, *umax*}] - displays the trigger, at start of which the variable *u* changes from *umin* to *umax*.

Trigger [Dynamic [*u*], {*umin*, *umax*, *du*}] - displays the trigger, when started the variable *u* changes from *umin* to *umax* in increments of *du*.

Trigger [Dynamic [*u*], {*umin*, *umax*}, *ups*] - displays the trigger, when started the variable *u* changes from *umin* to *umax* with a step *ups* every second.

When you press the start button (large triangle), the value of the dynamic variable *x* begins to change from 0 to 1 within a few seconds. Then this change stops. The panel also has buttons to stop variables and return to 0 (Reset).

### **Functions for marking additional information on graphs**

Function ClickPane[*image*, *func*],  
ClickPane[*image*, {{*xmin*, *ymin*}, {*xmax*, *ymax*}}, *func*] is used to indicate a point in the image *image* using the expression *func*. Most often, the ClickPane function is used to select with the mouse cursor a certain place in the image, in which you need to click to place a graphic object, such as a point, arrowhead, circle, etc.

The Tooltip [*expr*, *label*] function outputs the expression *expr* and replaces it with the label if the mouse cursor is set to the expression *expr*.

To recognize a graph with the Tooltip [] function, you must hover the mouse cursor over one of the curves.

### **PopupWindow**

The function PopupWindow [] provides control over the mouse cursor placed in it object. If the cursor is placed on the object, it is modified and a panel with the specified message appears. For example, in Fig. 18 object is a circle, which at the time of hovering the mouse cursor changes color. Then, after the panel labeled "This is a disk" appears, the circle restores color.

### **Display the menu and select its position**

The MenuView function [] provides the creation of menus with drop-down items under numbers that correspond to the value of the variable n.

### Display of menus with attachments and navigation between them

The TabView function [] displays a menu with attachments that you can switch between by left-clicking. An image is created in the object window, which is created by a cellular automaton using the CellularAutomaticc [] function with different values of the r (rule) parameter. Possible values of r are selected from the list by activating the corresponding tab with the mouse.

### Slide menu

Slide menus are used to create presentations. Use the SlideView [] function to create a slide menu. When activating the buttons with triangle images, you can select a symbol from the list. Switching can go in any direction, as well as immediately at the beginning or end of the list.

### Creating windows of interactive dialogue with the user

Mathematica provides the ability to create dialog boxes based on the GUIKit extension package. You need to connect the extension first GUIKit` using the function Needs["GUIKit`"].

### Work program

Write a program that demonstrates the ability to work with elements of the graphical interface in accordance with the option specified in table. 1.

Table 1

Variant	GUI elements
1,11	Slider single-coordinate, drop-down menu, function PopupWindow[]
2,12	Two-coordinate slider, data entry panel, function MenuView[]
3,13	Element CheckBox, function RadioButton[],function TabView[]
4,14	Locator, function SetterBar[],slide menu
5,15	Funky MousePosition[],function Setter[],user dialog box
6,16	Function Input[],function Print[],locator
7,17	Function Opener[],color slider, element CheckBox
8,18	Button, trigger mechanism, two-coordinate slider
9,19	Manipulator, function ClickPane[],function MousePosition[]
10,20	The slider is single-coordinate, function Tooltip[],button

### Control questions

1. List the I / O functions of the Mathematica system.
2. Name the output formats supported by Mathematica.
3. List the types of sliders and name their functional purpose.
4. Name the elements of the graphical interface, designed to choose one of the alternatives.
5. Name the functions of working with the mouse.
6. List the functions for creating marks on graphs.



7. Name the functions for color selection.
8. Specify the functions that are used to create the menu.
9. Describe the principle of using the SlaidMenu [] function.
10. Explain the functional purpose of the trigger function.

## Literature

1. Mathematica 5. Tutorial. System of symbolic, graphic and numerical calculations / Shmidsky J.K. - Dialectics 2004 - 592 p.
2. Chen K., Jiblin P., Irving A. Matlab in mathematical research: Per. from English - M.: Mir, 2001. 346 p.
3. Differential Equations with Mathematica, Third Edition / Brian R. Hunt, Ronald L. Lipsman, John E. Osborn, Donald A. Outing, Jonathan Rosenberg - 2009 John Wiley & Sons, 271 pp.
4. A Physicist's Guide to Mathematica, Second Edition / Patrick T. Tam – 2008 Academic Press, 728 pp.
5. Computer Solutions in Physics: With Applications in Astrophysics, Biophysics, Differential Equations, and Engineering / Steve VanWyk - World Scientific 2008 - 282 pp.
6. Mathematica by Example, Fourth Edition / Martha L. Abell, James P. Braselton Publisher: Academic Press 2008 - 576 pp.
7. Mathematica DeMYSTiFied / Jim Hoste - McGraw-Hill Professional 2008 – 320 pp.
8. Mathematica Navigator: Mathematics, Statistics and Graphics, Third Edition / Heikki Ruskeepaa Academic Press 2009 - 1136 pp.