MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

NATIONAL TECHNICAL UNIVERSITY
"KHARKIV POLYTECHNICAL INSTITUTE"

APPLIED METHODS OF CALCULATIONS ON MOTOR VEHICLES

Methodological instructive regulations
for laboratory work

specialty
J8 Automobile transport
G11 Mechanical engineering
full-time education

Kharkiv
NTU "KhPI"
2025

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

NATIONAL TECHNICAL UNIVERSITY
"KHARKIV POLYTECHNICAL INSTITUTE"

APPLIED METHODS OF CALCULATIONS ON MOTOR VEHICLES

Methodological instructive regulations
for laboratory work
specialtys
J8 Automobile transport
G11 Mechanical engineering
full-time education

Approved
editorial and publishing university council,
protocol No. __ dated ___________

Kharkiv
NTU "KhPI"
2025

Applied methods of calculations on motor vehicles. Methodological instructive regulations for laboratory workfrom the course "Applied methods of calculations on motor vehicles" for students of specialtys J8 Automobile transport G11 Mechanical engineering of full-time / O.Rebrov, V. Shevtsov, A. Kozhushko, A. Mamontov. – Kharkiv: NTU "KhPI", 2025.– 55 p.

Author: O. Rebrov
V. Shevtsov
A. Kozhushko
A. Mamontov

Reviewer:

Department of Car and Tractor Industry

CONTAINS

INTRODUCTION

At present, the problems of constructing and studying mathematical models of systems for various purposes and signals of various kinds occupy one of the leading places in modern science. The penetration of mathematical methods into engineering, natural sciences and humanities follows the path of mathematical design and modeling of the corresponding objects, and the ever-increasing capabilities of computer technology ensure the great success of this scientific direction.

An important aspect of modeling is that the model must be an adequate substitute for the real situation, a real system. The model reduces the redundancy of information storage, automatically taking into account the values that correspond to the real object and characterize it, presented in the form of input and output values and state characteristics..

The MATLAB system provides an intelligent environment for organizing computations for hundreds of thousands of professionals in engineering and scientific research. The MATLAB system is a leader in providing a convenient, integrated programming language for implementing a wide variety of mathematical algorithms and mathematical modeling tasks. The package also includes Simulink, a visual modeling tool that allows you to build and explore mathematical models without programming.

MATLAB programming offers the greatest flexibility, richness, and convenience for solving and exploring mathematical problems. In presenting the material, preference has been given to the simplest language constructs, by studying which you can create a wide variety of non-trivial mathematical algorithms.

LABORATORY WORK №1

Programming in MATLAB

Objective: To become familiar with the interface and capabilities of the MATLAB package. To master the techniques for using the basic arithmetic functions of the MATLAB package.

Tasks: complete all the examples given in the methodological guide, complete the independent work tasks.

Information material. Today, the MATLAB package is a powerful and universal tool for solving problems in various fields of human activity.

The working environment of the MATLAB package has a convenient interface for accessing many MATLAB utilities. When you start MATLAB, the workspace appears on the screen (Fig. 1.1).

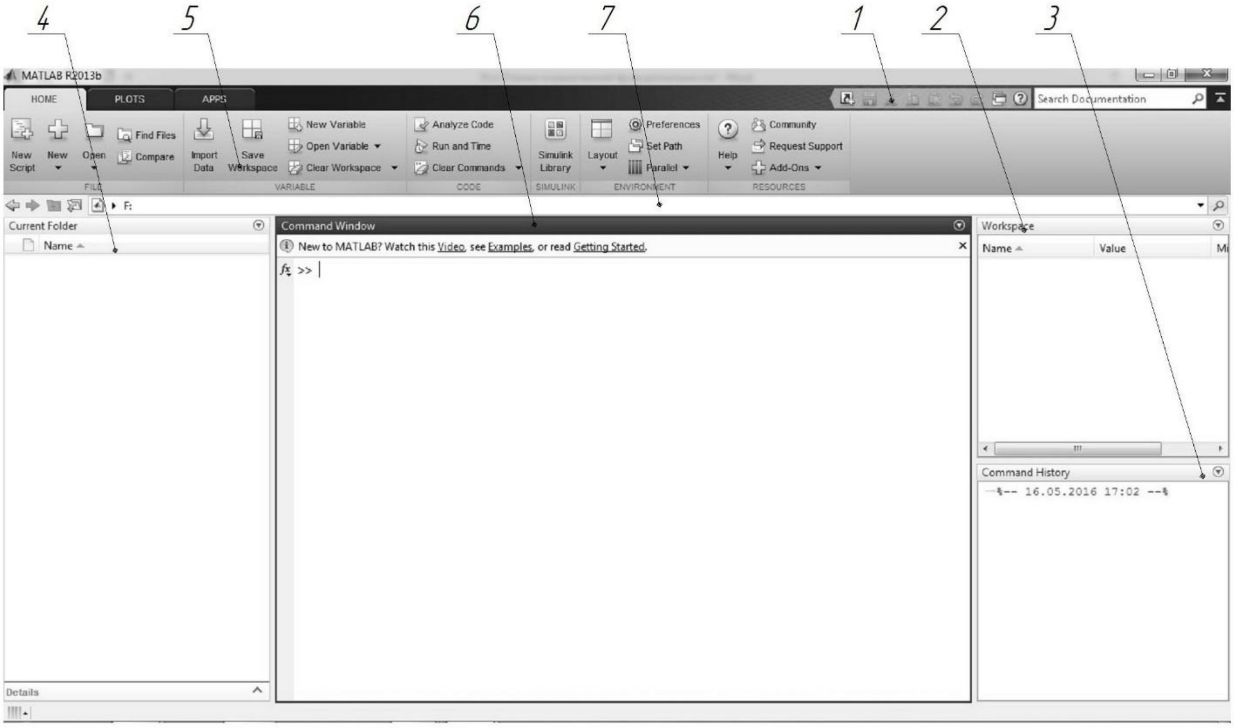


Figure 1.1 - Workspace of the MATLAB package:

1 - Menu; 2 - Workspace tabbed window; 3 - Command history tabbed window; 4 - Current folder tabbed window; 5 - Toolbar; 6 - Command window command window; 7 - Status bar

The Workspace tab (position 2) provides access to various Toolbox modules and the contents of the Workspace. The Command History and Current Directory tabs (items 3,4) are used to view and recall previously entered commands and to set the current directory. The single-line Command window (position 6) is used to enter commands.

All commands described in this lab should be typed at the command line. The >> symbol itself indicates a command line prompt and is not required in the examples. It is convenient to use the scroll bars or keys to navigate around the workspace <Home>, <End> to move left or right and <PageUp>, <PageDown> to move up or down. If the command line with the cursor disappears after moving around the command window workspace, you need to press <Enter>.

Note: If some of the windows described above are missing from the MATLAB environment, select the corresponding items from the Desktop menu: Workspace, Command History, Current Directory, Command Window.

Arithmetic calculations. If you type "1+2" in the command line and press <Enter>, the following is displayed in the MATLAB command window (Fig. 1.2):

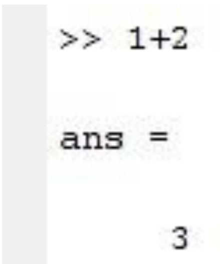


Figure 1.2 – Calculation result in the Command Window field

The MATLAB program first calculated the sum "1+2", then wrote the result into the special variable ans and displayed its value. If you need to continue working with the expression, for example, calculate "(1+2)/4.5", then you need to use the already existing result, which is stored in the variable ans (Fig. 1.3):

```
>> ans/4.5

ans =

    0.6667
```

Figure 1.3 – Calculation result using previous results

Output format of calculation results. The required result output format is determined by the user from the MATLAB menu. Choose Preferences from the File menu. To set the output format, ensure that the Command window is selected from the list in the left panel. The format assignment is made from the Numeric format list in the Text display panel. The output format can also be set directly on the command line, an example is given in Table 1.

Table 1 – Calculation result output format

Format name	Translation result

long	`>> format long`
	`>> 10000000/3`
	
	`ans =`
	
	`   3.333333333333334e+006`

bank	`>> format bank`
	`>> 10000000/3`
	
	`ans =`
	
	`   3333333.33`

short	`>> format short`
	`>> 10000000/3`
	
	`ans =`
	
	`  3.3333e+006`

Procedure:

1.1 Calculate the square root (each line of input ends with pressing the "Enter" key).

Enter the data in the form presented in table 1.4.

Table 1.4 – Initial data for calculation.

`>> Q=144;` `>> sqrt(Q)`	`>> Q=144;` `>> Q^(1/2)`

1.2 Calculate the sine of $-\pi$ this π (with a step $\frac{\pi}{12}$). Enter the data as shown in Fig. 1.4:

```
>> x=-pi:(pi/12):pi;
>> sin(x)
```

Figure 1.4 – Calculating the sine of a variable

The calculation results will appear on the screen.

1.3 Calculate function y^3-2y^2+26 within the range of 2.1 to 3.5 with a step of 0.1. Enter the data as shown in Fig. 1.5:

```
>> y=2.1:0.1:3.5;
>> (y.^3)-2*(y.^2)+26
```

Figure 1.5 – Finding the result of a function of a variable

1.4 Find the meanings of the expressions presented in Table 1.5 on your own:

Table 1.5 – Expressions for independent work.

1/2sin(0.5)	23·34/28
1/4cos(0.7)	sin(π)+1
2tan(0.3)	cos(2π)-1
0.72-3.41/2	|1-1/sin(π/3)|
4tan(0.8)	cos(-3π)+1
$\frac{\left(12\frac{1}{6}-6\frac{1}{27}-5,25\right)13,5+0,111}{0,02}$	$\frac{\left(6,6-3\frac{3}{14}\right)5\frac{5}{6}}{(21-1,25)/2,5}$

1.5. Independently calculate the functions presented in Table 1.6:

Table 1.6 – Functions for independent calculation

$2\cos(x)$	within the range of $-\pi$ this π with a step $\frac{\pi}{7}$
$\cos(x)-\tan(x)$	within the range of $-\pi$ this π with a step $\frac{\pi}{8}$
$(x-3)^2-2(x-2)$	within the range of -10 to 15 with a step of 0.25
x^3-3x+8	within the range of 0 to 2 with a step of 0.2
$\frac{x^2}{5}-x^3+2x$	within the range of 1 to 5 with a step of 0.4

Control questions

1. How are real numbers represented in calculations in the MATLAB system?

2. How do you change the format in which real numbers are displayed in the Command window?

3. How do you declare variables in the MATLAB language?

4. How do you make the result of the actions written in the next line: a) appear in the command window; b) appear on the screen?

5. What is the role of the system variable ans?

6. How do you return a previously typed command to the command line?

LABORATORY WORK №2

2-D Graphing in MATLAB

Objective: to familiarize yourself with the MATLAB function graphing system.

Tasks: to complete all the examples given in the methodological instructions, to complete the tasks for independent work.

Information material. MATLAB has well-developed graphical data visualization capabilities. Let's first consider the construction of a simple graph of a function of one variable, using the example of the function $y(x) = e^{-x}\sin(10x)$ in the range from 0 to 1 with a step of 0.05.

Deriving a function as a graph consists of three steps:

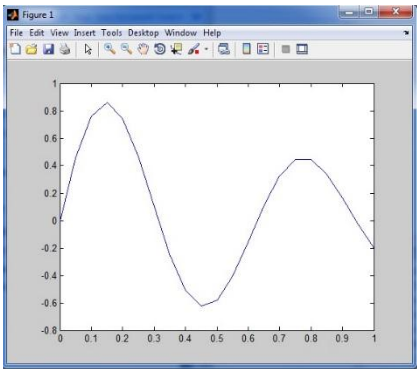
1. Enter the vector of argument values x.

2. Vector calculation y in function $y(x)$.

3. Calling the plot command to build a graph.

It is better to end the commands for inputting a vector and calculating a function with a semicolon to plot their value in the command window (it is not necessary to put a semicolon after the plot command, because it does not output anything in the command window).

```
>> x=0:0.05:1;
>> y=exp(-x).*sin(10*x);
>> plot(x,y)
```



and b

Figure 2.1 – Function implementation $y(x) = e^{-x}\sin(10x)$:
and– in the MATLAB command window; b – function graph

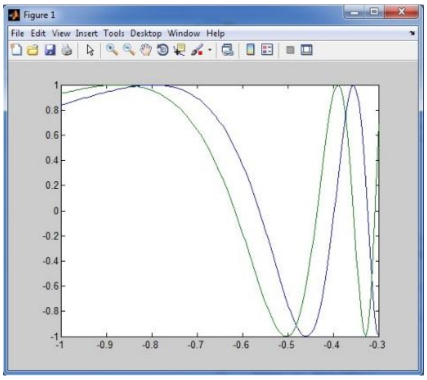
After executing the commands, the Figure 1 window with the function graph will appear on the screen (Fig. 2.1 b). The window contains a menu, a toolbar and a graph area. In the following, commands specifically designed for graphing will be described. Now we are interested in the principle of graphing and some of the simplest possibilities for visualizing functions.

To graph a function in the MATLAB working environment, two vectors of the same dimension must be defined, for example x and y. The corresponding array x contains the values of the arguments, and y - the value of the function from these arguments. The command plot connects points with coordinates (x(i), y(i)) straight lines, automatically scaling the axes for optimal placement of the graph in the window. When plotting graphs, it is convenient to position the main MATLAB window and the window with the graph next to each other on the screen so that they do not overlap.

It is convenient to compare several functions by plotting their graphs on the same axes. For example, let's plot the graphs of the functions on the area from -1 to -0.3 $f(x)=\sin\left(\frac{1}{x^2}\right)$, $f(x)=\sin\left(\frac{1.2}{x^2}\right)$ using the following command sequence:

```
>> x=[-1:0.005:-0.3];
>> f=sin(x.^-2);
>> g=sin(1.2*x.^-2);
>> plot(x,f,x,g)
```

a



b

Figure 2.2 – Function implementation $f(x)=\sin\left(\frac{1}{x^2}\right)$, $f(x)=\sin\left(\frac{1.2}{x^2}\right)$:

A – in the MATLAB command window; b – function graphs

_Note 1._Using a plot with an argument - a vector - results in the construction of a "vector graph", ie the dependence of the values of the vector elements on their number. The plot argument can also be a matrix, in which case the column graphs are plotted on a single axis coordinate.

Sometimes it is necessary to compare the type of change of two functions with very different values. The graph of a function with small values almost merges with the abscissa axis and it is impossible to determine its shape. In this situation, the function plotyy, which displays graphs in a window with two vertical axes that have the appropriate scale. For example, let's compare two functions: $f(x) = x^{-3}$ and $f(x) = 1000 \cdot (x+0.5)^{-4}$.

When performing this example, note that the color of the graph matches the color of the corresponding ordinate axis.

The following functions are used to plot graphs on logarithmic and semi-logarithmic scales:

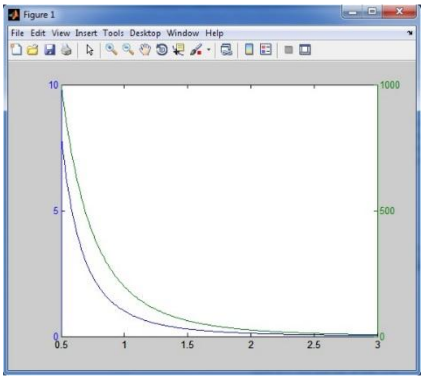
– loglog (logarithmic scale on both axes);

– semilogx (logarithmic scale on the abscissa axis);

– semilogy (logarithmic scale on the ordinate axis).

```
>> x=[0.5:0.01:3];
>> f=x.^-3;
>> F=1000*(x+0.5).^-4;
>> plotyy(x,f,x,F)
```

a



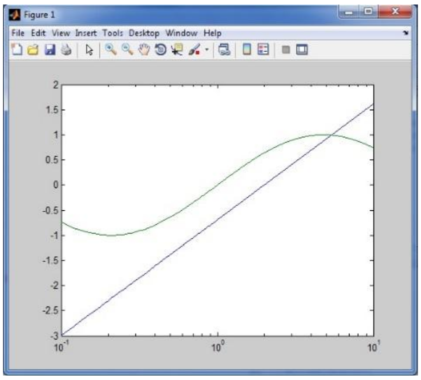
b

Figure 2.3 – Function implementation $f(x) = x^{-3}$:
A – in the MATLAB command window; b – function graphs

Arguments loglog, semilogx, semilogy are given as a pair of vectors of abscissa and ordinate values, in the same way as plot. Let's build, for example, graphs of functions $f(x) = \ln(0.5x)$ and $g(x) = \sin(\ln(x))$ on the section from 0.1 to 10 on a logarithmic scale (Fig. 2.4) along the x axis:

```
>> x=[0.1:0.01:10];
>> f=log(0.5*x);
>> g=sin(log(x));
>> semilogx(x,f,x,g)
```

a



b

Figure 2.4 – Function implementation $f(x) = \ln(0.5x)$, $g(x) = \sin(\ln(x))$:
A – in the MATLAB command window; b - function graphs.

The constructed graphs of functions should be as easy to understand as possible. It is often necessary to apply markers, change the color of the line, and when preparing for printing, set the type of lines (solid, dotted, dash-dotted). Table 2.1 shows the change of color, marker type and line in MATLAB.

Table 2.1 – Changing the color, marker type, and line in MATLAB

Color		Marker type		Line type	
1	2	3	4	5	6
yes	yellow	.	dot	-	solid
m	pink	at	circle	:	dotted
c	blue	x	cross	-.	dash-dotted
r	red	+	plus sign	--	dashed
g	green	*	star		
b	blue	s	square		
w	white	d	rhombus		
k	black	v	triangle with vertex down		
		^	triangle with vertex up		
		<	triangle with vertex left		
		>	triangle with right vertex		
		p	five-pointed star		
		h	six-pointed star		

If, for example, you need to plot the first graph with a red dotted marker without a line, and the second with a black dotted line, then you should use the command plot(x, f, 'r.', x, g, 'k:').

The ease of use of graphs largely depends on additional design elements: a coordinate grid, axis labels, title, and legend. The grid is applied with the command grid on, axis signature is achieved by commands xlabel, ylabel, the title is set by the command title. Having multiple graphs on the same axes requires placing a legend by command legend with information. Let's plot, for example, the change in daily temperature:

```
>> time=[0 4 7 9 10 11 12 13 13.5 14 14.5 15 16 17 18 20 22];
>> temp1=[14 15 14 16 18 17 20 22 24 28 25 20 16 13 13 14 13];
>> temp2=[12 13 13 14 16 18 20 20 23 25 25 20 16 12 12 11 10];
>> plot(time, temp1, 'ro-', time, temp2, 'go-')
>> grid on
>> title('Суточная температура')
>> xlabel('Время (час.)')
>> ylabel('Температура (C)')
>> legend('10 мая, 11 мая')
```

Figure 2.5 – Implementation of the example in the MATLAB command window

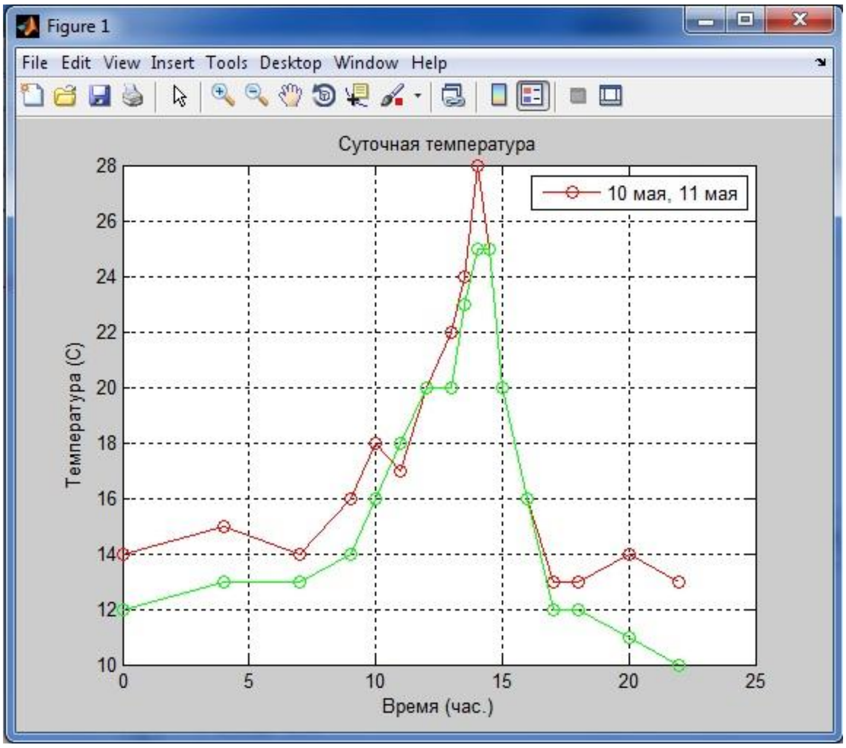


Figure 2.6 – Daily temperature change graphs

Do it yourself:

2.1. Plot the graph of a function $f(x)=\frac{tg(x)}{x}$ in the interval $-100 \le x \le 100$ with step 2.

2.2. Plot the graph of a function $f(x)=\frac{\sin(x)\cdot\cos(x)}{x}$ in the interval $-5 \le x \le 5$ in steps of 1 and 0.01.

2.3. Compare the nature of the change in functions $f(x)=\frac{\cos(x)}{6}+1$ and $g(x)=\frac{\sin(x)}{6}+10$ in the interval $-\pi \le x \le \pi$ with the number of points $n=100$.

2.4. Construct graphs of functions $f(x)=2\cdot\ln(x-1)$ and $g(x)=\sin(\ln(x^2))$ on the area from 0.1 to 10 on a logarithmic scale along the x-axis.

2.5. Construct a graph on the area from 0 to 10 (with a step of 0.1) using the random number generation of the fifth line of the rand(101) function.

Review questions

1. How to graph a function of two independent variables?

2. Is it possible to plot a graph of one vector relative to another?

3. How do I change the curve type, for example, from solid to dotted?

4. How to label the coordinate axes of a graph?

5. What functions can be used to plot a graph on logarithmic and semi-logarithmic scales?

LABORATORY WORK №3

Plotting 3-D graphs in MATLAB

Purpose of work:to get acquainted with the MATLAB 3-D function graphing system.

Task:complete all the examples given in the methodological instructions, complete the tasks for independent work.

Information material Plotting a function of two variables in MATLAB on a rectangular domain of variables involves two preliminary steps:

1. Dividing the definition area with a rectangular grid.

2. Calculating function values at the intersection points of grid lines.

Let's plot the function graph. $Z(X,Y) = X^2 + Y^2$ in the area of definition in the form of a square X belongs to [0, 1], Y – [0, 1]. It is necessary to divide the square with a uniform grid (for example, with a step of 0.2) and calculate the values of the functions at the nodes marked with dots.

It is convenient to use two two-dimensional arrays X and Y, with dimensions of six by six, to store information about the coordinates of the nodes. The array X consists of identical rows, in which the coordinates X1, X2, ..., X6 are recorded, and the array y contains identical columns with Y1, Y2, ..., Y6. We write the value of the function at the nodes of the grid in the array Z of the same dimension (6x6), and to calculate the matrix Z we use the expression for the function, but with element-wise matrix operations. Then, for example, Z (3,4) will be exactly equal to the value of the function Z (X, Y) at the point (X3, Y4). To generate the arrays of the grid X and Y by the coordinates of the nodes, MATLAB provides the meshgrid function, to construct a graph in the form of a wireframe surface – the mesh function. The following operators lead to the appearance on the screen of a window with a graph of the function (the semicolon at the end of the operators is not put in order to control the generation of the arrays):

```
>> [X,Y]=meshgrid(0:0.2:1,0:0.2:1)

X =

         0    0.2000    0.4000    0.6000    0.8000    1.0000
         0    0.2000    0.4000    0.6000    0.8000    1.0000
         0    0.2000    0.4000    0.6000    0.8000    1.0000
         0    0.2000    0.4000    0.6000    0.8000    1.0000
         0    0.2000    0.4000    0.6000    0.8000    1.0000
         0    0.2000    0.4000    0.6000    0.8000    1.0000


Y =

         0         0         0         0         0         0
    0.2000    0.2000    0.2000    0.2000    0.2000    0.2000
    0.4000    0.4000    0.4000    0.4000    0.4000    0.4000
    0.6000    0.6000    0.6000    0.6000    0.6000    0.6000
    0.8000    0.8000    0.8000    0.8000    0.8000    0.8000
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000

>> Z=X.^2+Y.^2

Z =
         0    0.0400    0.1600    0.3600    0.6400    1.0000
    0.0400    0.0800    0.2000    0.4000    0.6800    1.0400
    0.1600    0.2000    0.3200    0.5200    0.8000    1.1600
    0.3600    0.4000    0.5200    0.7200    1.0000    1.3600
    0.6400    0.6800    0.8000    1.0000    1.2800    1.6400
    1.0000    1.0400    1.1600    1.3600    1.6400    2.0000

>> mesh(X,Y,Z)
```
Figure 3.1 – Implementation of the meshgrid function in the MATLAB command window

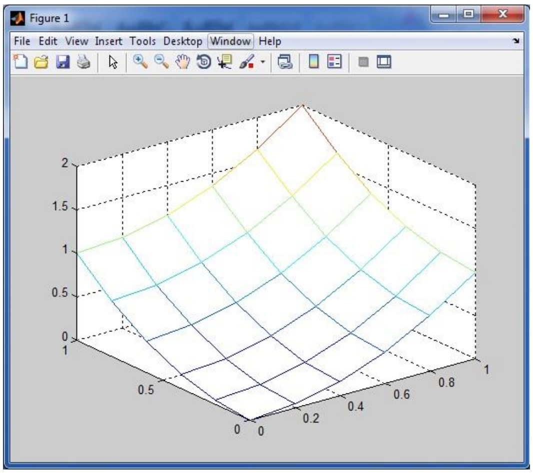


Figure 3.2 – Function surface $Z(X,Y) = X^2 + Y^2$

Let's consider the main possibilities for visualizing functions of two variables, using the example of plotting a function graph. $Z(X,Y) = 4\sin(2\pi X)\cdot\cos(1.5\pi Y)\cdot(1-X^2)\cdot Y\cdot(1-Y)$ in the rectangular region of definition X belongs to [-1; 1], Y to [0, 1].

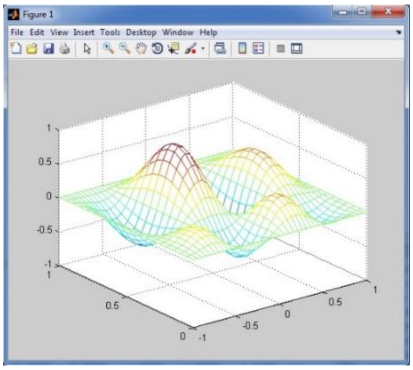
```
>> [X,Y]=meshgrid(-1:0.05:1,0:0.05:1);
>> Z=4*sin(2*pi*X).*cos(1.5*pi*Y).*(1-X.^2).*Y.*(1-Y);
```

Figure 3.3 - Implementation in the MATLAB command window

The surf function builds a wireframe surface of the graph of a function and fills each cell of the surface with a specific color, depending on the values of the function at the points corresponding to the corners of the cell. The color is constant within each cell.

The shading flat command allows you to remove the frame lines. The shading interp command is intended to obtain a surface smoothly filled with color, depending on the values of the function.

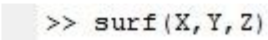
```
>> mesh(X,Y,Z)
```

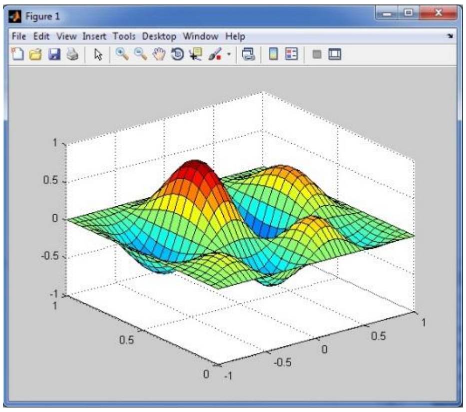


a *b*

Figure 3.4 -– Mesh function implementation
a– in the MATLAB command window; b – graph of the function surface

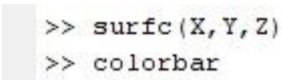
With shading faceted, you can return to a surface with wireframe lines. The three-dimensional plots obtained using the commands described above are convenient for getting an idea of the shape of the surface, but they are difficult to judge the value of the function. MATLAB defines the colorbar command, which displays a bar next to the plot that establishes a correspondence between the color and the value of the function.

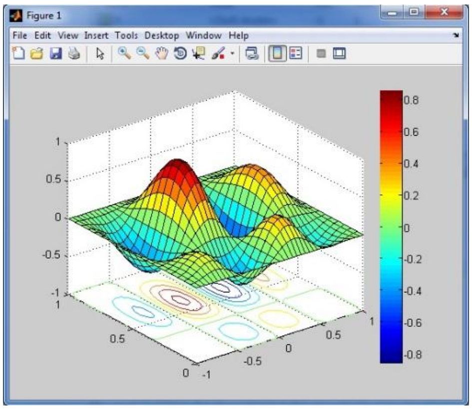




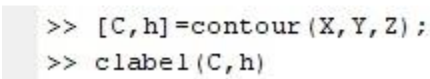
a *b*

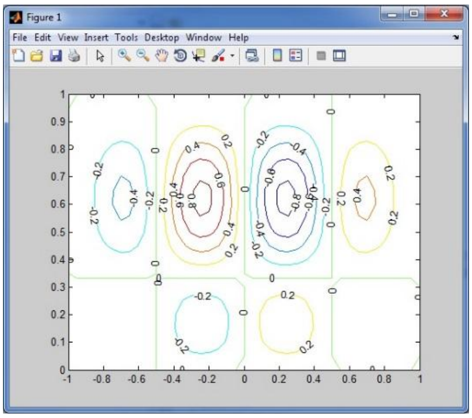
Figure 3.5 – Implementation of the surf function
a – in the MATLAB command window; b – graph of the wireframe surface of the function





a *b*





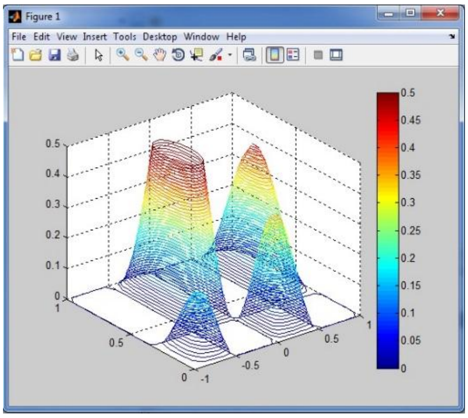
c *d*

Figure 3.6 – Implementation of the surfc and contour functions
a, *c*– in the MATLAB command window; b, d – graph of the wireframe surface of functions and level lines, respectively

Using a colored surface, it is difficult to draw conclusions about the value of a function at a particular point in the xy plane. The meshc or surfc commands allow you to get a more accurate idea of the behavior of the function. These commands build a mesh surface or a colored mesh surface and place the function level lines (constant function value lines) on the xy plane.

MATLAB allows you to construct a surface consisting of level lines using the contour3 function. This can be useful in the same way as the mesh, surf, meshc, and surfc functions described above with three arguments. In this case, the number of level lines is chosen automatically. It is possible to specify either the number of level lines or a vector whose elements are equal to the values of the function that will appear as level lines as the fourth argument in contour3. Specifying a vector (the fourth argument levels) is convenient when you need to investigate the behavior of a function in some region of its values (a slice of the function). For example, construct a surface consisting of level lines corresponding to the values of the function from 0 to 0.5 with a step of 0.01:

```
>> levels=[0:0.01:0.5];
>> contour3(X,Y,Z,levels)
>> colorbar
```



a *b*

Figure 3.7 – Implementation of the contour3 function
a–in the MATLAB command window; b – graph of the wireframe surface of the function

To display several graphs of functions of one variable on the same axes, the capabilities of the plot, plotyy, semilogx, semilogy, loglog functions were used. They allow you to display graphs of several functions by specifying the corresponding vector arguments in pairs, for example plot (x, f, x, g). However, they cannot be used to combine three-dimensional graphs. To combine such graphs, the hold on command is intended, which must be set before constructing the graph. In the following example, combining two graphs (a plane and a cone) leads to their intersection. The cone is given parametrically by the following dependencies: $x(u,v) = 0.3 \cdot u \cdot \cos v$; $y(u,v) = 0.3 \cdot u \cdot \sin v$; $z(u,v) = 0.6 \cdot u$; $u, v \in [-2\pi,\ 2\pi]$ To graphically display a cone, first you need to generate a column vector and a row vector using a colon, containing the values of the parameters in the given interval (it is important that u is a column vector and v is a row vector):

```
>> u=[-2*pi:0.1*pi:2*pi]';
>> v=[-2*pi:0.1*pi:2*pi];
```

Figure 3.8 – Function implementation in the MATLAB command window

Next, matrices X, Y are formed, containing the values of the functions $x(u,v)$ and $y(u,v)$ at points corresponding to the parameter values. The formation of matrices is performed using the outer product of vectors.

```
>> a=[1;2;3];
>> b=[5;6;7];
>> C=a*b'

C =

     5     6     7
    10    12    14
    15    18    21
```

Figure 3.9 - Function implementation in the MATLAB command window

Let's form the matrices X, Y for the graphic representation of the cone:

```
>> X=0.3*u*cos(v);
>> Y=0.3*u*sin(v);
```

Figure 3.10 – Function implementation in the MATLAB command window

The matrix Z must be the same size as the matrices X and Y. In addition, it must contain values that correspond to the parameter values. If the function $z(u,v)$ included the product of u and v, then the matrix Z could be filled in a similar way with the matrices X and Y using the outer product. On the other hand, the function $z(u,v)$ can be represented as $z(u,v) = 0.6 \cdot u \cdot g(v)$, where $g(v) \equiv 1$ Therefore, to calculate Z, we can apply the outer product of the vectors u and $g(v)$, where the row vector $g(v)$ has the same dimension as v, but consists of units:

```
>> Z=0.6*u*ones(size(v));
```

Figure 3.11 – Function implementation in the MATLAB command window

All the necessary matrices for the cone mapping are created. The plane mapping is performed as follows:

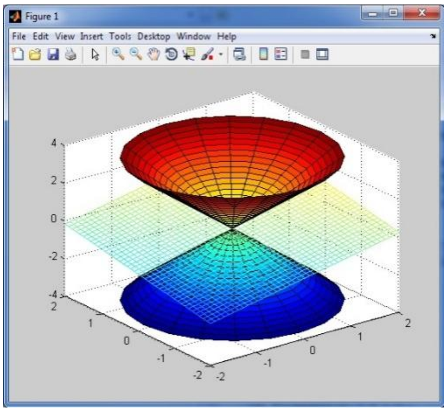
```
>> [X1,Y1]=meshgrid(-2:0.1:2);
>> Z1=0.5*X1+0.4*Y1;
```

Figure 3.12 – Function implementation in the MATLAB command window

Now it is not difficult to write the complete sequence of commands for constructing the intersection of a cone and a plane:

```
>> u=[-2*pi:0.1*pi:2*pi]';
>> v=[-2*pi:0.1*pi:2*pi];
>> X=0.3*u*cos(v);
>> Y=0.3*u*sin(v);
>> Z=0.6*u*ones(size(v));
>> surf(X,Y,Z)
>> [X1,Y1]=meshgrid(-2:0.1:2);
>> Z1=0.5*X1+0.4*Y1;
>> hold on
>> mesh(X1,Y1,Z1)
>> hidden off
```

a



b

Figure 3.7 – Implementation of the program for constructing a cone and a plane:
a–in the MATLAB command window; b – graph of the intersection of function surfaces

Work order.

Do it yourself.

3.1. Construct a three-dimensional spiral using functions such as $x(t) = 2 \cdot \sin(t)$ and $y(t) = \cos(t)$ in the range $0 \le t \le 9 \cdot \pi$ with a step of 0.01; 0.1 and 1 (plot3).

3.2. Construct a surface $f(x, y) = e^{-x^2 - y^2}$ in ranges $-1 \le x \le 1$ and $-2 \le y \le 2$ two different functions.

3.3. Construct a surface $Z = \dfrac{a \cdot \sin(f(x, y))}{f(x, y)}$ in meaning $f(x, y) = \sqrt{x^2 + y^2} + 0.001$ in the range $-15 \le x, y \le 15$ in increments of 0.2. Vary the coefficient a: 5, 20, 0.5.

3.4. Construct a conical surface using the following equations $X = Z \cdot \cos(T) \cdot tg\dfrac{\pi}{6}; Y = Z \cdot \sin(T) \cdot tg\dfrac{\pi}{6}$ at the same time $0 \le T \le 2\pi$, and also $0 \le Z \le 10$. Cut off part of the conical surface.

Review questions

1. How to convert a column to a vector and vice versa?

2. What components can be used to construct a 3-D surface in MATLAB?

3. What functions can be used to construct a 3-D surface in MATLAB?

4. How to build straight lines?

5. Which MATLAB function displays a column next to the graph that establishes a correspondence between the color and the value of the function?

LABORATORY WORK №4

Conditional and loop operators

Purpose of work:to familiarize yourself with the conditional and loop operators of the MATLAB system.

Task:complete all the examples given in the methodological instructions, complete the tasks for independent work.

Information material. The if conditional statement allows you to implement logic in MATLAB. Below is an example of an implementation of the sign() function (using an M-file), which returns +1 if the number is greater than zero, -1 if the number is less than zero, and 0 if the number is equal to zero:

```
x=input('Введите x:');
if x>0
    disp(1);
end
if x<0
    disp(-1)
end
if x==0
    disp(0);
end
```

Figure 4.1 – Function implementation in the MATLAB command window

When implementing assembly conditions in MATLAB, the logical operators AND, OR, and NOT are used.

For example, you can check that the variable x falls in the range from -5 to 5, but does not belong to the range from 10 to 21. Obviously, this can be implemented as follows:

```
x=input('Введите x: ');
if (x>=-5 && x<=5) && (x<10 || x>21)
    disp ('x належить [-5;5], та входить в [10;21]');
else
     disp ('x або не входить в [-5;5], або в [10;21]');
end
```

Figure 4.2 – Function implementation in the MATLAB command window

The switch conditional operator implements an algorithm that checks whether a variable is equal to a constant value. Let's consider an example of how this operator works to convert lowercase Latin letters to uppercase:

```
ch='a';
switch ch
    case 'a', ch='A';
    case 'b', ch='B';
    case 'c', ch='C';
    case 'd', ch='D';
    ...
    case 'z', ch='Z';
end
disp(ch);
```

Figure 4.3 - Function implementation in the MATLAB command window

The while loop operator, with an indefinite number of iterations, repeatedly executes an instruction or group of instructions as long as the controlling equation is true. Let's consider an example of the while loop for calculating the sum of a series $S = \sum_{i=1}^{20} i$:

```
S=0;  %початкове значення суми
i=1;  %лічильник суми
while i<=20  %цикл працює поки i<=20
    S=S+i;  %розраховується сума
    i=i+1; %лічильник збільшується на 1
    disp(S);
end
```

Figure 4.4 – Function implementation in the MATLAB command window

The for loop operator with a specified number of iterations. Let's consider an example of implementing an algorithm for finding the maximum value of an element in a vector:

```
a=[3 6 5 3 6 9 5 3 1 0];
m=a(1);
for i=1:length(a)
    if m<a(i)
        m=a(i);
    end
end
disp(m);
```

Figure 4.5 – Function implementation in the MATLAB command window

Procedure for performing the work. Do it yourself:

4.1. Check whether the variable x belongs to the range from 0 to 2. This is achieved by fulfilling one of two conditions: $x < 0$ or $x > 2$.

4.2. Calculate the sum of the series $\sum_{x=0.5}^{5} 2x^2 - 5x + 1$? Write a program that would display the number of iterations of the sum of a series on the working window.

4.3. Calculate the sum of the elements of the second line of the rand(4) function using the for operator.

4.4. Select the value of the torque T for the coupling, if its diameter is given and the dependence is known (table 4.1):

Table 4.1 – Initial data for calculation.

Version	Diameter, mm	Torque, Nm
1	250…280	500
2	290…310	1000
3	320…350	1200
4	360…380	1400
5	390	1600

4.5. Write a program that finds all three significant figures, the sum of the squares of the first two digits is equal to the sum of the cubes of the last two digits?

LABORATORY WORK №5

Getting to know the Simulink environment

Purpose of work:studying the principles and gaining practical skills in modeling elementary structural diagrams in the Simulink environment of the MATLAB package.

Task:

1. Learn how to create a new model and set basic parameters.
2. Get acquainted with the main blocks of the Sources, Sinks, Math operation sections of the Simulink environment using the example of performing laboratory work.
3. Complete the tasks for independent work.

Information material The simplest and most convenient tool for modeling dynamic systems in the MATLAB package is SIMULINK. It is supplied as a separate application and contains libraries of various blocks from which a structural diagram of the model is built in the working window.

To run the program, you must first start the MATLAB package. After opening the main MATLAB window, you must start the Simulink program. This can be done in one of three ways:

• Click the (Simulink) button on the MATLAB command window toolbar.

• In the command line of the main MATLAB window, type Simulink and press the Enter key on the keyboard.

• Execute the Open command in the File menu and open the model file (mdl - file).

The latter option is convenient to use for launching an already ready and debugged model, when you only need to perform calculations and do not need to add new blocks to the model.

To create a model in the SIMULINK environment, you must sequentially perform a number of actions.Create a new model file using the **File/New/Model** command:

1. Place the blocks in the model window. To do this, open the appropriate section of the library (for example, Sources). Then, by pointing the cursor at the required block and pressing the left mouse button, "drag" the block into the created window. The mouse button must be kept pressed.

2. Next, if necessary, you need to change the block parameters set by the program by default. To do this, double-click the left mouse button, pointing the cursor at the block image. The window for editing the parameters of this block will open. When setting numerous parameters, keep in mind that a period, not a comma, should be used as a decimal separator.

3. After installing all the blocks from the necessary libraries on the diagram, you need to connect the circuit elements. To connect the blocks, you need to point the cursor to the "output" of the block, and then, pressing and, without releasing the left mouse button, draw a line to the input of another block. Then release the button. In case of correct connection, the arrow image at the input of the block changes color. To create a branch point in the connecting line, you need to move the cursor to the intended node and, by pressing the right mouse button, draw a line. To delete a line, you need to select the line (in the same way as for a block), and then press the Delete key on the keyboard.

4. After compiling the calculation scheme, it is necessary to save it as a file on disk by selecting the File / Save As ... menu item in the scheme window and specifying the folder and file name. It should be borne in mind that the file name should not exceed 32 characters, must begin with a letter and cannot contain Cyrillic characters and special characters. The same requirement applies to the file path (to the folders in which the file is stored). When editing the scheme further, you can use the Fille / Save menu item. When the SIMULINK program is

restarted, the scheme is loaded using the File / Open ... menu in the library browser window or from the main MATLAB window.

To improve the visibility of the model, it is convenient to use text labels. To create a label, you need to point the mouse at the place of the label and double-click the left mouse button. After that, a rectangular frame with an input cursor will appear. In a similar way, you can change the labels for model blocks. Before performing calculations, you must first set the calculation parameters. The calculation parameters are set in the control panel of the Simulation / Parameters menu.

The calculation settings window has 4 tabs:

• Solver (Calculation) - Set the parameters for calculating the model.

• Workspace I/O (Workspace Input/Output) - Sets the parameters for data exchange with the MATLAB workspace.

• Diagnostics - Select diagnostic mode options.

• Advanced - Set advanced options.

The model calculation parameters are set using the controls located on the Solver tab. These controls are divided into three groups: Simulation time (simulation interval or, in other words, calculation time), Solver options (calculation parameters), Output options (output parameters).

Work order.Create a model to calculate the expression:$\frac{(25^3+8)\cdot\cos(30^{\circ})}{2+0,5\cdot\sqrt{71}-tg(45^{\circ})}$. To create simple constants, use the Constant block in the Sources section. Drag this block into the model field and set the required value in the parameters window (the parameters window opens by double-clicking on the block image). Such constants will be the numbers 25, 3, 8, 30, 2, 0.5, 71.45.

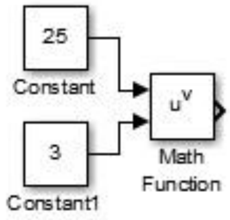


Figure 5.1 – Implementation in the model using blocksSIMULINK

For exponentiation, the Math Function block in the Math Operations section is used (note that the upper input is used for the number to be exponentiated, and the lower input is used for the exponent value). To select the exponentiation function, select the pow parameter in the options window. For summing values, the Sum block in the Math Operations section is used, and for product, the Product block in the Math Operations section is used.

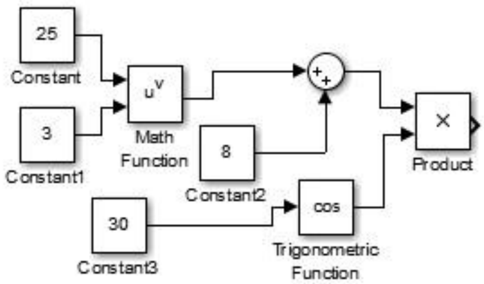


Figure 5.2 – Implementation in the model using blocksSIMULINK

To find the trigonometric function, use the Trigonometric Function block of the Math Operations section, which is input with a constant value. To find the cosine, select it from the cos list in the Trigonometric Function block parameters window. Combining them, we get:

The denominator of the expression is also constructed by analogy. To find the root, the Sqrt block of the Math Operations section is used. The tangent of the angle is found similarly to the cosine. To find the difference, you can use the Subtract block of the Math Operations section. Combining these, we get:

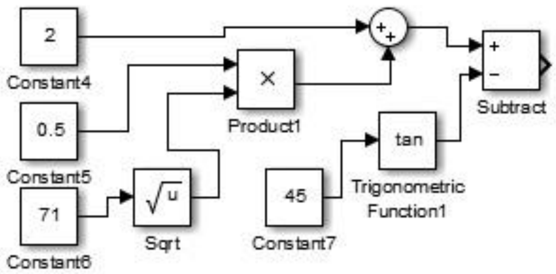


Figure 5.3 – Implementation in the model using blocksSIMULINK

To calculate the private expression (division operation), we use the Divide block of the Math Operations section. The numerator is supplied to the upper input, and the denominator to the lower. To output the numerical result, we use the Display block of the Sinks section. We connect the blocks and run the model. As you can see, the result of the calculation will be the number 525. The Display block can also be installed at any stage to determine the intermediate results of the calculation.

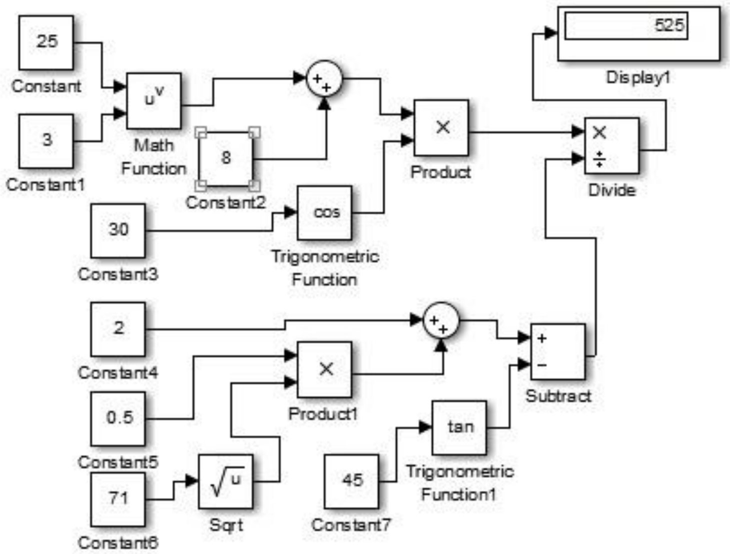


Figure 5.4 – Implementation in the model using blocksSIMULINK

Do it yourself.

Make a flowchart for the expressions (Table 5.1).

Table 5.1 – Initial data for calculation.

$tg(30+\frac{\sin(45)+\sqrt{10}\cdot(10-2{,}5)}{1{,}5\cdot\sqrt{27}+\cos(\sqrt{5+8})})$	$\frac{\cos(2{,}5+\sqrt{76})}{\sin(60)}+tg(\cos(\sqrt{5}+8{,}5))-\frac{ctg(20)}{25\cdot(\sqrt{43}+5)}$

Test questions

1. What is Simulink?
2. What are the main elements of the library browser window?
3. How are models created in Simulink?
4. Basic operations on blocks?
5. What are the main modeling parameters?
6. What blocks were used in the work? What functions do they perform?

LABORATORY WORK №6

Working with AC signal sources

Purpose of work: Learn the rules for creating systems with variable signal sources and displaying results in the form of graphs.

Task:

- Get acquainted with the types of AC signal sources and their main parameters.

- Complete tasks for independent work.

Information material. If it is necessary to create circuits with a variable signal, use the standard blocks located in the Sources section. The Sine Wave block is used to create a sinusoidal signal with a given frequency, amplitude, phase and offset. Two algorithms can be used to form the output signal by the block. The type of algorithm is determined by the Sine Type parameter (signal formation method):

• Time-based - Based on the current time.

• Sample-based - By the size of the model time step.

The output signal of the source in this mode corresponds to the expression:

y = Amplitude * sin (frequency * time + phase) + bias.

Parameters:

1. Amplitude - Amplitude.

2. Bias - The constant component of the signal.

3. Frequency (rads / sec) - Frequency (rads / sec).

4. Phase (rads) - Initial phase (rads).

5. Sample time - Model time step. Used to coordinate the operation of the source and other components of the model in time. The parameter can take the following values: 0 (default) - Used when modeling continuous systems.> 0 (positive value) - Used when modeling discrete systems. In this case, the model time step can be interpreted as a quantization step in the time of the output signal. -

1 - The model time step is set to the same as in the previous block, i.e. the block from which the signal comes to this block.

The source of linearly varying influence Ramp forms a linear signal of the form y = Slope * time + Initial value. parameters:

1. Slope - The rate of change of the output signal.

2. Start time - Start time of signal formation.

3. Initial value - Initial signal level at the output of the block.

The Step signal generator generates a step signal. Parameters:

Step time - Time of signal drop (s).

Initial value - Initial value of the signal.

Final value - The final value of the signal.

The difference can be either in the larger direction (the final value is greater than the initial value) or in the smaller direction (the final value is less than the initial value). The values of the initial and final levels can be not only positive, but also negative.

The Pulse Generator pulse signal source is used to generate rectangular pulses. Parameters:

Pulse Type - Signal generation method. Can take two values:

- Time-based - Based on the current time.

- Sample-based - By the size of the model time and the number of calculation steps.

Amplitude - Amplitude.

Period - Period. Given in seconds for Time-based Pulse Type or in model time steps for Sample-based Pulse Type.

Pulse width - Pulse width. Given in % relative to the period for Time-based Pulse Type or in model time steps for Sample-based Pulse Type.

Phase delay - Phase delay. Given in seconds for Time-based Pulse Type or in model time steps for Sample-based Pulse Type.

Sample time - Model time step. Available for Sample-based Pulse Type.

Workflow. Let's build a circuit using all the listed sources of variable signals. After creating a new model, we transfer all the necessary blocks from the sections to the model field. The Product, Sum, and Add blocks are located in the Math Operations section. The Scope block is used to display the simulation results on the screen, and the abscissa is the model time scale. The Scope block is located in the Sinks section. Let's connect the blocks as shown in Figure 6.1.

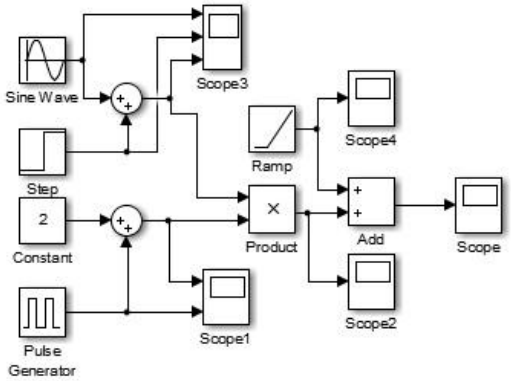


Figure 6.1 – Implementation in the model using blocksSIMULINK

We will set the following initial parameters for the Sine Wave block: Amplitude - 10; Bias - 0; Frequency - 2; Phase - 10. After starting the model, a visual representation of this signal source will be presented in the first window of the Scope 3 block (to increase the number of displayed signals, click the Parameters button in the Scope block parameters window and in the General tab opposite Number of axes enter the number of required inputs).

We will set the following initial parameters for the Step block: Step time - 5; Initial value - 0; Final value - 10. The image of this signal will be presented in the second window to the Scope 3 blocks. As can be seen from the model, the sum of these two signals will be presented in the third window of the Scope 3 block.

We will set the following output parameters for the Pulse Generator block: Amplitude - 1; Period - 2; Pulse Width - 50. The image of this signal will be presented in the second window of the Scope 1 block. The first window of the block will present the sum of the signal of the Pulse Generator block and the

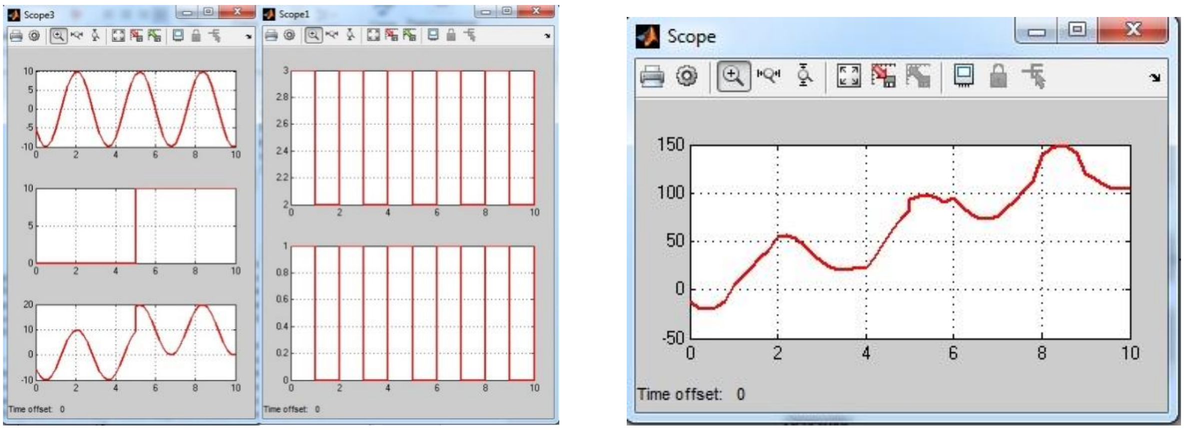
Constant block (the value of the Constant block is 2). Click on the Parameters button and in the General tab opposite Number of axes enter the number of required inputs).

We will set the following initial parameters for the Step block: Step time - 5; Initial value - 0; Final value - 10. The image of this signal will be presented in the second window to the Scope 3 blocks. As can be seen from the model, the sum of these two signals will be presented in the third window of the Scope 3 block.

We will set the following output parameters for the Pulse Generator block: Amplitude - 1; Period - 2; Pulse Width - 50. The image of this signal will be presented in the second window of the Scope 1 block. The first window of the block will present the sum of the signal of the Pulse Generator block and the Constant block (the value of the Constant block is 2)

We will set the following output data for the Ramp block: Slope - 10; Initial output - 5. The image of this signal will be presented in the Scope 4 block window. The Scope 2 block window will present an image of the product of the sum of the Sine Wave and Step blocks and the sum of the Constant and Pulse Generator. The Scope block window will present a final image of the signal simulated in this model.

We will set the following output data for the Ramp block: Slope - 10; Initial output - 5. The image of this signal will be presented in the Scope 4 block window. The Scope 2 block window will present an image of the product of the sum of the Sine Wave and Step blocks and the sum of the Constant and Pulse Generator. The Scope block window will present a final image of the signal simulated in this model.



a *b*

Figure 6.2 – Results of implementing the model using blocksSIMULINK:
a– graphs of input signals Sine Wave, Step and their sum; b – graph of the overall simulation result

Do it yourself.

Make a block diagram containing the sources of the alternating signal:

$$\frac{Ramp}{Sine\ Wave1} + Sine\ Wave \cdot Step1 + \frac{Pulse\ Generator}{Sine\ Wave2} - Step$$

with input data: Sine Wave, Step, Pulse Generation, Ramp (Table 6.1):

Table 6.1 – Initial data for calculation.

Blocks	Block parameters			
	Amplitude	Bias	Frequency	Phase
Sine Wave	11	0	2	2
Sine Wave1	10	5	5	0
Sine Wave2	8	-5	4	0
	Amplitude	Period	Pulse Width	Phase delay
Pulse Generator	-7	10	20	0
	Step Time		Initial Value	Final Value
Step	2		-3	1
Step1	3		1	4
	Slope		Start time	Initial output
Ramp	1		5	5

Check questions:

1. What blocks are called AC signal sources and why?

2. What are the main parameters of these blocks?

3. What is used to display the simulation result when using variable signal blocks?

4. What mathematical operations can be performed on these blocks?

5. How do you change the amplitude of a sinusoidal signal, and what block is used to create such a signal?

6. What is the SampleTime parameter, and what does it affect?

LABORATORY WORK №7

Graphs and tools for displaying simulation results

Purpose of work: Get acquainted with the types of graphs, setting their basic parameters and designing the signals displayed in them, as well as other means of displaying simulation results.

Task:

- Get acquainted with the types of graphs and their main parameters.
- Get acquainted with the blocks for outputting numerical modeling results.
- Complete tasks for independent work.

Information material The result output blocks are located in the Sinks section.

The Scope block plots the studied signals as a function of time. It allows you to observe signal changes during the simulation process.

The toolbar contains 11 buttons:

1. Print - print the contents of the oscilloscope window.
2. Parameters - access to the parameters settings window.
3. Zoom - zoom in on both axes.
4. Zoom X-axis - zoom in on the horizontal axis.
5. Zoom Y-axis - zoom in on the vertical axis.
6. Autoscale - automatic scaling on both axes.
7. Save current axes settings - save the current window settings.
8. Restore saved axes settings - restore previously saved window settings.
9. Floating scope - putting the oscilloscope into "free" mode.
10. Lock / Unlock axes selection - lock / unlock the connection between the current coordinate system of the window and the displayed signal. The tools become available after enabling the Floating scope mode.
11. Signal selection - selection of signals to display. The tool becomes available after enabling Floating scope mode.

There are several ways to resize displayed graphs:

1. Click the appropriate button on the toolbar or click once with the left mouse button in the desired location on the graph. The zoom will increase by 2.5 times.

2. Click the appropriate button on the toolbar or, by clicking the left mouse button, use the dynamic frame or segment to specify the area of the graph for the enlarged image.

Right-click in the graph window and select Axes properties from the context menu. The graph properties window will open, where you can specify the limit values of the vertical axis using the Y-min and Y-max parameters. In the same window, you can specify the graph title (Title) by replacing the expression% <SignalLabel> in the input line.

The block parameters are set in the 'Scope' parameters window, which is opened using the (Parameters) tool in the toolbar. The parameters window has two tabs:

General - general settings.

Data history - parameters for saving signals in the MATLAB workspace.

The Floating Scope block is essentially a regular Scope oscilloscope, put into "free" mode. In this mode, the oscilloscope block has no inputs, and selecting the entire signal is done using the Signal selection tool on the toolbar. To select signals, perform the following steps:

1. Select the coordinate system in which the graph will be displayed. This is achieved by a single left-click inside the desired system. The selected coordinate system will be highlighted in blue around the perimeter.

2. Use the tool to open the Signal Selector dialog box.

3. Check the names of the blocks whose output signals you want to investigate.

After the calculation is completed, the selected signals will be displayed in the Floating Scope block window.

The Display block displays the value as a number.

parameters:

Format - data display format. The Format parameter can have the following values:

1. short - 5 significant decimal digits.
2. long - 15 significant decimal digits.
3. short_e -5 significant decimal digits and 3 powers of ten symbols.
4. long_e - 15 significant decimal digits and 3 powers of ten symbols.
5. bank - "money" format. A format with a fixed point and two decimal digits in the fractional part of the number.

Decimation - the multiplicity of displaying the input signal. When Decimation = 1, every value of the input signal is displayed, when Decimation = 2, every second value is displayed, when Decimation = 3, every third value, etc.

Sample time - model time step. Determines the discreteness of data display.

Floating display (check box) - puts the block in "free" mode. In this mode, the input port of the block is absent, and the signal for display is selected by clicking the left mouse button on the corresponding communication line. In this mode, the calculation parameter Signal storage reuse should be set to off (Advanced tab in the Simulation parameters dialog box).

XY Graph block Plots a graph of one signal as a function of another (a graph of the form Y (X)).

parameters:

x-min - Minimum signal value on the X axis.

x-max - Maximum signal value on the X axis

y-min - Minimum signal value on the Y axis.

y-max - Maximum signal value on the Y axis

Sample time - model time step.

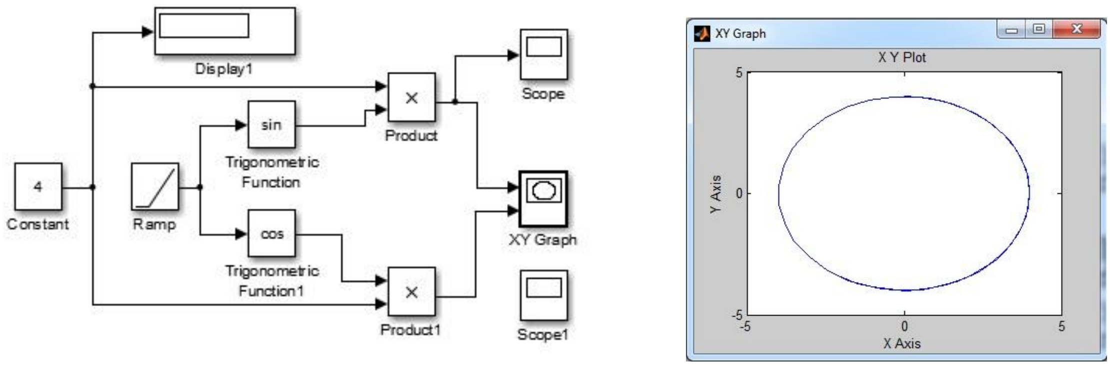
The block has two inputs. The upper input is intended for supplying a signal, which is the argument (X), the lower one is for supplying the function values (Y).

The graphic device can also be used to construct time dependences. To do this, apply a time signal from the output of the Clock block to the first input.

Work order.

In the course of the work, it is proposed to create a model with a variable signal at the output, using the Ramp block.

Build a diagram:



and *b*

Figure 7.1 – Implementation of a model with a variable signal at the output:
and– SIMULINK block diagram; b – simulation result graph

In the Constant block, set the value to 4. Leave the Ramp block unchanged. In the XY Graph block, set the minimum values x, y = -5, the maximum x, y = 5. When selecting the value to display in Floating Scope1, select Product1. If constructed correctly, the following graph should appear in the XY Graph window:

Display the value of the Constant block on the Display1 block. Display the value of the results of the product of the Constant block and the sine function from the signal of the Ramp block, and the product of the cosine function from the signal of the Ramp block, respectively, on the Scope and Scope1 blocks.

Do it yourself.

Draw a flowchart corresponding to the following formula:

$$Pulse\ Generator^{Ramp} - 3 \cdot Sine\ Wave + Step - \frac{Pulse\ Generator}{Sine\ Wave^2}$$

Display the result using Scope and XY Graph on the X axis - Sine Wave, on Y - the result of the calculation of the entire block diagram. Output data: Sine

Wave: Amplitude - 5; Bias - 0; Frequency - 2; Phase - 1. Step: Step time - 4; Initial value - 2; Final value - (-2). Ramp: Slope - 8; Start time - 2. Pulse Generator: Amplitude - 3; Period - 10; Pulse Width - 20%. XY Graph - Xmin, Ymin - (-25), Xmax, Ymax - 25. To raise to a power, the Math Function block is used, in the parameters of which in the Function list we select pow. The lower input of the block corresponds to the power, and the upper one to the signal being raised to a power.

Check questions:

1. Which blocks display a graph of the result of a signal change over time?
2. How can I change the display style of the graph's appearance?
3. Is it possible to plot a signal change over time using the XY Graph block?
4. How to change the display form of the Display block result?

LABORATORY WORK №8

Getting to know the Ports & Subsystems blocks

Purpose of work: Get acquainted with the types and main parameters of the modeling blocks of the Ports & Subsystems section.

Task:

- Get acquainted with the main parameters of the Subsystem and Atomic Subsystem blocks.

- Get acquainted with the basic parameters and application of the block with the Enabled Subsystem control signal.

- Complete tasks for independent work.

Information material. A subsystem is a fragment of a Simulink model, designed as a separate block. The number of subsystems in a model is not limited, in addition, subsystems can include other subsystems. The level of nesting of subsystems in each other is also not limited.

The subsystem is connected to the model (or to the top-level subsystem of the hierarchy) using input (Inport block of the Sources library) and output (Outport block of the Sinks library) ports. Adding an input or output port to a subsystem results in the appearance of a port label on the subsystem image, through which external signals are transmitted inside the subsystem or output to the main model. Renaming the Inport or Outport blocks allows you to change the port labels displayed on the subsystem icon from the standard ones (In and Out) to those required by the user. Subsystems can be virtual (Subsystem) and monolithic (Atomic Subsystem). The difference between these types of subsystems is the order of execution of the blocks during the calculation. If the subsystem is virtual, Simulink ignores the presence of boundaries separating such a subsystem from the model when determining the order of block calculation. In other words, in a virtual system, the output signals of several blocks can first be calculated, then the calculation of the blocks in the main model can be performed, and then the calculation of the blocks included in the subsystem can be performed again. A monolithic subsystem is considered a single (indivisible) block and Simulink performs the calculation of all blocks in such a subsystem without switching to the calculations of other blocks in the main model. There are two ways to create a subsystem in the model:

1. Copy the desired subsystem from the Subsystem library into the model.

2. Select the desired fragment of the model with the mouse and execute the Create Subsystem command from the Edit menu of the model window. The selected fragment will be placed in the subsystem, and the inputs and outputs of the subsystem will be provided with the appropriate ports. This method allows you to create a virtual unmanaged subsystem. In the future, if necessary, you can make the subsystem monolithic by changing its parameters, or managed by adding a control element from the desired subsystem located in the library. You can cancel the grouping of blocks into a subsystem with the Undo command.

Access to the subsystem parameters window is via the Edit menu using the Block Parameters ... command. Parameters:

1. Show port labels - Show port labels.

2. Treat as atomic unit (checkbox) - Consider the subsystem as monolithic. Thus, the blocks of the virtual and monolithic subsystems are the same block, differing in the value of this parameter.

3. Access - Availability of the subsystem for changes. Select from the list:

• ReadWrite - The user can open and modify the subsystem.

• ReadOnly - The user can open the subsystem for viewing only.

• NoReadOrWrite - The user cannot open and modify the subsystem.

4. Name of error callback function - The name of the function used to handle errors that occur in this subsystem.

The Subsystem (or Atomic Subsystem) block in the library contains input and output ports and a communication line between them.

The Enabled Subsystem (hereinafter referred to as the E-subsystem) is activated when there is a positive signal at the control input. If the input signal is vector, then the subsystem is activated if at least one element takes a positive value. The value of the output signal in the event that the system is blocked is determined by the settings of the output ports of the subsystem (Outport blocks). In the event that the Output when disabled parameter (type of signal at the output of the subsystem) of the output port has the value held, then the output signal of the subsystem is equal to the last value calculated by it, if this parameter has the value reset, then the output signal of the subsystem is equal to the value set by the Initial output parameter (initial value).

The properties of the E-subsystem are determined by the parameters of the Enable block, which can be located anywhere in this subsystem. Its parameters are listed below. Parameters:

1. States when enabling - State when starting. The parameter sets the state of the subsystem at each start. It is selected from the list:

- held - Use the previous state (the last state when the system was active).
- reset - Use the initial (initial) state.

2. Show output port (checkbox) - Show output port. When the check box is checked, an additional output port appears on the Enable block icon, the signal from which can be used to control blocks within the subsystem.

Work order.

Let's solve the system of equations using the Subsystem block. To solve, we express the variable from each equation:

$$\begin{cases} 2.5 \cdot x - 30 \cdot y + 5.4 \cdot z = -86.73 \\ -1.3 \cdot y + 8 \cdot x - 0.5 \cdot z = 8.96 \\ -0.8 \cdot z + 0.2 \cdot x + 50 \cdot y = 141.26 \end{cases}$$

and

$$\begin{cases} x = \dfrac{-86.73 + 30 \cdot y - 5.4 \cdot z}{2.5} \\ y = \dfrac{8.96 - 8 \cdot x + 0.5 \cdot z}{-1.3} \\ z = \dfrac{141.26 - 0.2 \cdot x - 50 \cdot y}{-0.8} \end{cases}$$

b

Figure 8.1 – Solving the system of equations:

and– system of equations; b – expressed variables

Let's make a block diagram for these equations. We have three variables - we use a separate Subsystem block for each, we will call each block x, y, z, respectively. As can be seen from the equation, to find each variable, you need to know the other two variables. That is, for the model inside each Subsystem block, you need two inputs corresponding to the variables in the equations (Fig. 8.2).

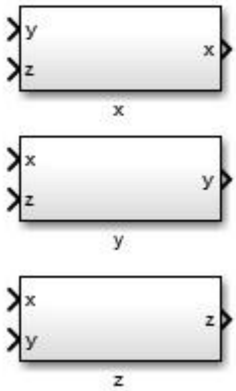


Figure 8.2 – Subsystem blocks for each variable.

To do this, in the Subsystem block, we will add another In block and rename both input blocks to y and z for the x block, to x and z for the y block, and x and y for the z block. The entire system will be self-contained, that is, the selection of variables will be carried out by calculating the blocks in parallel inside and outside our model. Recall that the use of the Atomic Subsystem block is unacceptable in this case, since such parallel calculation is impossible in them. We collect each equation for determining the variable in its Subsystem block. Equation for determining x in the Subsystem block with the name x.

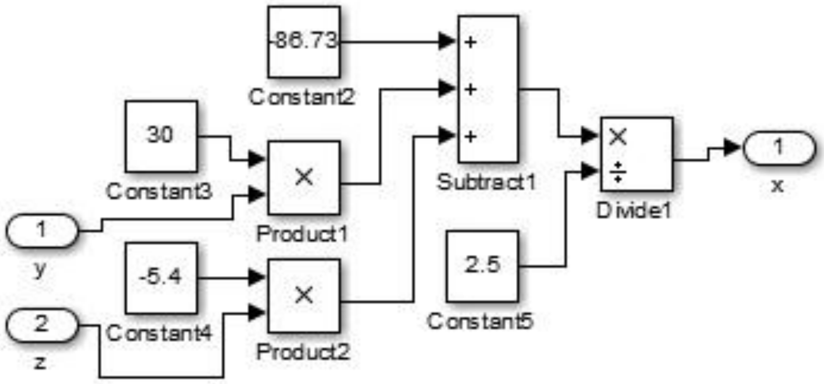


Figure 8.3 – Model for determining the variable x.

Similarly, we type equations for other variables in the y and z blocks. We connect the corresponding inputs and outputs and add the Display blocks.

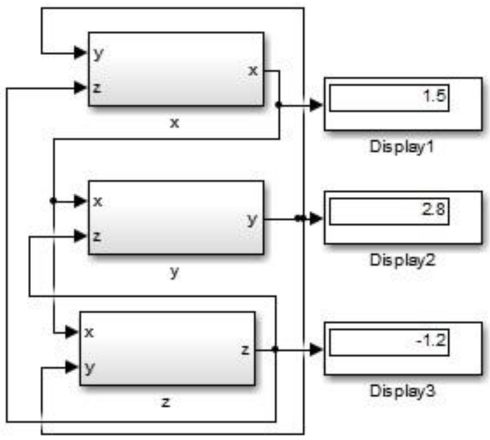


Figure 8.4 – Variable definition model with output

We build a block diagram using the Enabled Subsystem block. As a control signal, we use the Pulse Generator block with the output data: Amplitude - 1, Period - 2, Pulse Width - 20%. As an input signal, we use the Ramp block with the output data: Slope - 20, Initial output - 0. In the Enabled Subsystem parameters, use output 1 with the Held parameter, and 2 with the Reset parameter. To output 2, apply the product of the input signal and 5. The general scheme is shown in Fig. 8.6.

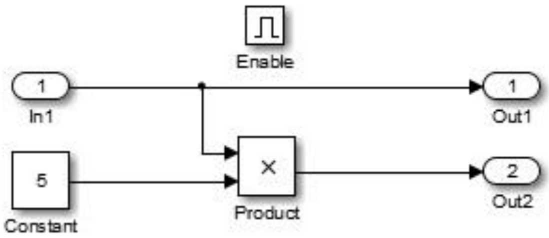


Figure 8.5 – Model of signal change in the middle of the Enabled Subsystem block

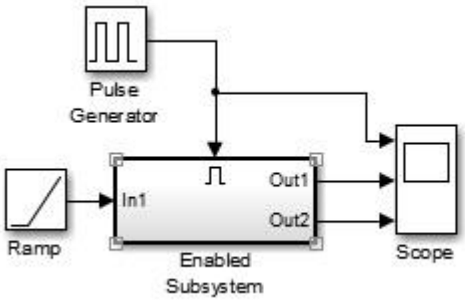


Figure 8.6 – General view of the model using the Enabled Subsystem block

In the example, the States when enabling parameter of the Enable block has the value held. The Output when disabled parameter of the first output port of the subsystem has the value held, and the second - reset. As can be seen from the timing diagrams, when the subsystem is turned off, the signal of the second output port is compared to the initial value (zero), and the signal of the first output port is equal to the last calculated value at the moment the subsystem is active.

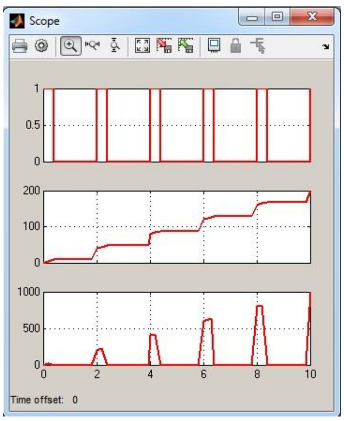


Figure 8.7 – Simulation result using the Enabled Subsystem block

Do it yourself.

Build a block diagram using Subsystem with an integrated Atomic Subsystem block for the system of equations:

$$\begin{cases} \sin(30)\cdot x + \frac{1}{5}\cdot y + 5.5\cdot z - k = 10.3098 \\ tg(45)\cdot z - x - 20\cdot y + 1.2\cdot k = -195.3506 \\ z + \frac{2}{9}\cdot x - \cos(60)\cdot y + 5\cdot k = 15.6352 \\ -14\cdot z + 4.5\cdot x + \frac{y}{3} + k = -8.6667 \end{cases}$$

Test questions

1. What are subsystem blocks used for?

2. Can a subsystem be used within a subsystem?

3. What is a controlled subsystem? What is used to control it?

4. What is the difference between a Subsystem and an Atomic Subsystem?

5. Is it possible to make an Enabled Subsystem a Subsystem (according to the principle of operation)?

LABORATORY WORK №9

Subsystem blocks of conditional operators If Action Subsystem, For Iterator Subsystem, While Iterator Subsystem.

Purpose of work: Get acquainted with the main blocks of conditional operators, the conditions for their use, and the principles of building a model using these blocks.

Task:

- Get acquainted with the conditional operator blocks If Action Subsystem, For Iterator Subsystem, While Iterator Subsystem

- Complete tasks for independent work.

Informational material.

The If conditional operator block provides the generation of control signals for the If Action subsystems. parameters:

1. Number of inputs - Number of inputs.

2. If expression - Conditional expression. A conditional expression can include the following characters: <. <=, ==, ~ =,>,> =, &, |, [], and also a unary minus. If the written conditional expression is true, then a control signal is generated at the output port of the If block.

3. Elseif expressions - One or a list of alternative conditional expressions separated by commas, to be evaluated if the conditional expression If expression is false. Each conditional expression recorded in the list Elseif expressions corresponds to an output Elseif port on which a control signal is formed if the corresponding conditional expression is true. In this case, the algorithm for calculating alternative conditional expressions is such that if one of the alternative conditional expressions is true, then the following expressions in the list are not checked. An alternative conditional expression can include the same characters as the If expression.

4. Show else condition (check box) - Show Else port. A control signal is generated on the Else port if the conditional expression and all alternative conditional expressions are false.

The block icon displays the conditional expressions written in its parameters. Adding each new alternative conditional expression results in a new Elseif output port.

If the input signals of the block are scalars, then to denote them in expressions, the notation of the form u1, u2, u3, etc. is used. If the input signals are vector, then to denote the elements of the vector, expressions of the form u1 (1), u1 (2), u2 (1), u2 (2), etc. are used.

Conditionally driven Action Subsystem

The subsystem is designed to work under the control of If or Switch Case blocks. In the first case, it is called If Action Subsystem, and in the second, Switch Case Action Subsystem.

The subsystem parameters are determined by the settings of its output ports, as well as by the settings of the Action Port block, the presence of which in the subsystem turns it into an Action Subsystem.

The block has one setting:

States when execution is resumed - The state of the system subsystem when the next execution is resumed. The parameter value is selected from the list:

held - Use the previous state (the last state when the system was active).

reset - Use the initial (initial) state.

The parameter in question has the same effect on the behavior of the subsystem as the States when enabling parameter of the Enable block.

Managed Subsystem For Iterator Subsystem

A controlled subsystem For Iterator Subsystem is a subsystem that is executed repeatedly during one simulation cycle. The number of iterations must be known in advance and can be given by an external signal source or by a block parameter. The main properties of the subsystem are set by the For Iterator iteration block. The block is analogous to the For loop operator in the C programming language.

The For block can be located anywhere in the subsystem. Its parameters are listed below. Parameters:

1. States when starting - State of the subsystem at the next startup. The parameter value is selected from the list:

- held - Use the previous state (the last state when the system was active).
- reset - Use the initial (initial) state.

2. Source of number of iterations (checkmark) - The source specifies the number of iterations.

- internal - internal.
- external - External.

3. Number of iterations - Number of iterations. The parameter is available only when selecting an internal source of the number of iterations.

4. Show iteration number port - Display on the block icon the output port from which the iteration number signal is taken.

5. Output data type - Data type of the port output signal. The parameter value is selected from the list: int32, int16, int8 and double.

Managed While Iterator Subsystem

A controlled subsystem While Iterator Subsystem is a subsystem that is executed repeatedly during one simulation cycle. The number of iterations is not known in advance. The loop is terminated if the value of the logical signal at the control input of the subsystem becomes FALSE. The main properties of the subsystem are set by the iteration block While Iterator. The block is an analogue of the while (do-while) loop operator in the C programming language.

The properties of the While Iterator Subsystem are determined by the parameters of the While Iterator block. Its parameters are listed below.

parameters:

1. Maximum number of iterations - Maximum number of iterations. If the parameter value is -1 (minus one), then the number of iterations is not limited.

2. While loop type (check box) - Loop type. Select from the list:

- while - While loop.
- do-while - Do-while loop.

3. States when starting - The state of the system subsystem at the next startup. The parameter value is selected from the list:

- held - Use the previous state (the last state when the system was active).
- reset - Use the initial (initial) state.

4. Show iteration number port - Display on the block icon the output port from which the iteration number signal is taken.

5. Output data type - Data type of the port output signal. The parameter value is selected from the list: int32, int16, int8 and double.

The IC input port allows you to set the initial value of the signal that stops the execution of the while loop. If you use a do-while program, the subsystem will be executed at least once (since the condition is checked at the end of the loop).

Work order.

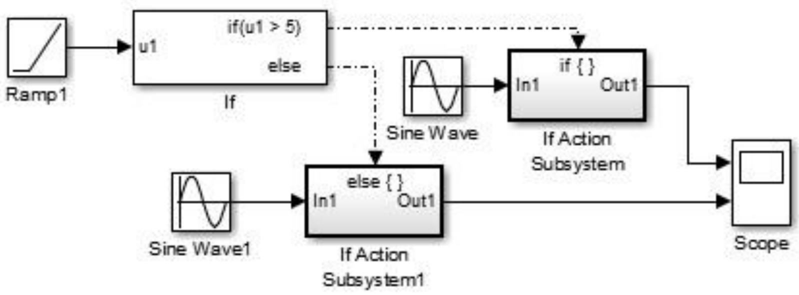


Figure 9.1 – General view of the model using conditional operator blocks

Let's create a block diagram using conditional programming blocks. As can be seen in the model, the If block is used to specify the conditions under which the If Action Subsystem and If Action Subsystem1 blocks pass the signal. The input signal of the If block is a Ramp block with standard initial parameters. While the

Ramp value is less than 5, the signal passes to the If Action Subsystem block, if the If condition is less than 5, the signal passes to the If Action Subsystem1 block. Parameters of the Sine Wave1 block: Amplitude - 15; Bias - 0; Frequency - 2. Parameters of the Sine Wave block: Amplitude - 10; Bias - 0; Frequency - 3. At the output, we get two signals:

As you can see, the If Action Subsystem1 block works first, then the If Action Subsystem.

Let's create a flowchart for calculating , where x changes from 1 to 10 using the conditional statement blocks For Iterator Subsystem, While Iterator Subsystem.$\sum x^2$

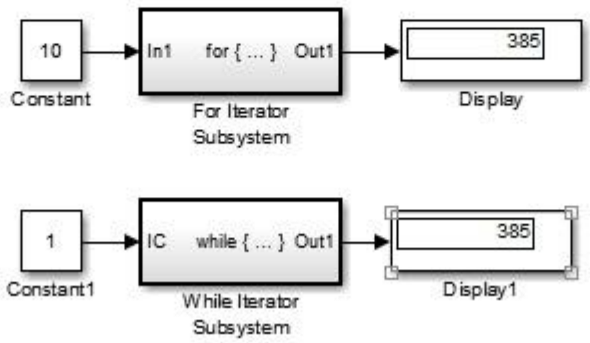


Figure 9.2 – Result of calculating the For Iterator Subsystem, While Iterator Subsystem blocks

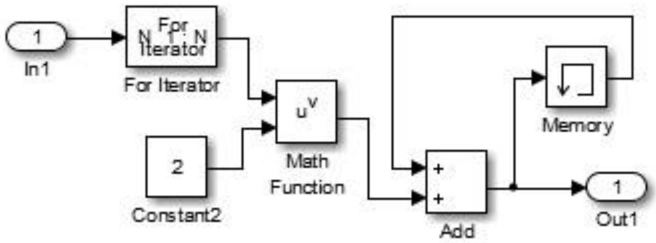


Figure 9.3 -Model of calculating the result inside the For Iterator Subsystem block

As you can see, the answer to this problem is 385. As mentioned above, the number of iterations in the first part of our scheme, namely in the For Iterator Subsystem block, is specified using a block inside the For Iterator subsystem. In our case, we need to calculate the sum of the squares of the first 10 digits, so we set the number of iterations to 10.

Each number is squared and summed to the previous one, this can be achieved using the Memory block of the Discrete section. This block remembers the previous value and outputs it again to the output, in our case for cyclic summing. The While Iterator Subsystem block works similarly.

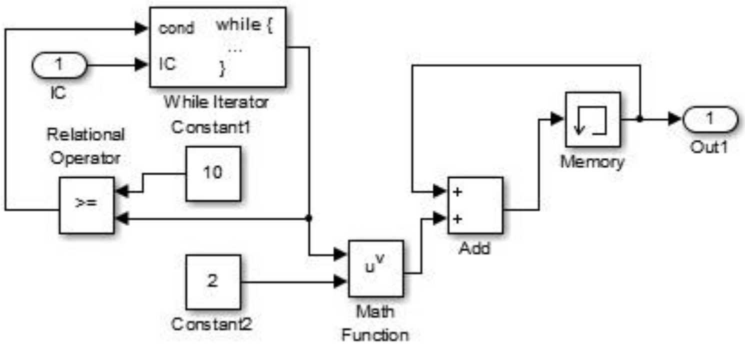


Figure 9.4 - Result calculation model in the middle of the While Iterator Subsystem block

As mentioned above, the Cond input receives the signal of the end of the loop execution. The summation loop itself is similar to the For Iterator Subsystem block. The difference is the presence of the Relation Operator block, which compares the condition for terminating the loop calculation, and if the condition is met, sends a False signal to the Cond input and the calculations are terminated.

Do it yourself:

9.1. Build a flowchart with a conditional operator If Action Subsystem, which will have 4 conditions: x less than 0; x greater than 0 but less than 5, x greater than 5 but less than 10, and x greater than 10. Apply Ramp, Sine Wave, Step, and Pulse Generator signals with standard parameters to the input. Output all results to one Scape block.

9.2. Create a For Iterator Subsystem, While Iterator Subsystem flowchart to calculate the accumulation of the sum of values with a step of 10. The loop execution stops when the sum reaches the value 100.

Check questions:

1. What are conditional and loop operators?

2. What blocks are used when building a circuit with the conditional If operator?

3. Which block parameters are the main ones for the For Iterator Subsystem block?

4. Which block parameters are the main ones for the While Iterator Subsystem block?

LITERATURE

1. Gaev E. O. Universal mathematical package MATLAB and typical problems of computational mathematics. Textbook / E. O. Gaev, B. M. Nesterenko– K.: NAU, 2004. – 176 p.

2.Lazarev Yu. F.MATLAB and Dynamic Systems Modeling. Tutorial /Yu. F. Lazarev– Kyiv: NTUU "KPI", 2009. – 60 p.

Educational edition

Methodological guidelines
to perform laboratory work
"Numerical and applied methods of vehicle calculations"
from the courses "Numerical methods of calculations of cars and tractors",
"Applied calculation methods for motor vehicles"

for students of specialties
J8 Automobile transport
G11 Mechanical engineering

Compilers:
REBROV Oleksiy
SHEVTSOV Vadim
KOZHUSHKO Andriy Pavlovych
MAMONTOV Anatolii

Responsible for the release Rebrov O.Yu.
The work was recommended for publication by Sivykh D.G.
Computer layout Shevtsov V.M.
Editor Rebrov O.Yu.

Plan 2025, item 340

Submitted for printing on 10/20/25.
Times New Roman font.

Publishing Center of NTU "KhPI".
Certificate of state registration of the State Enterprise No. 5478 dated 08/21/2017.
61002, Kharkiv, Kyrpichova St., 2