Noncommercial Joint-Stock Company

ALMATY UNIVERSITY OF POWER ENGENEERING AND TELECOMMUNICATIONS

Automation and Control Department

SOFTWARE FOR AUTOMATION SYSTEMS

Methodical guidelines to laboratory works for students of specialty 05070200 - Automation and control

Almaty 2019

COMPILED BY: Ibrayeva L.K., Abzhanova L.K., Ilyasov A.Z. Software for automation systems. Methodical guidelines to laboratory works for students of specialty 5B070200 - Automation and control. - Almaty: AUPET, 2019. – 47 p.

Methodical instructions intended to performance of works with use of various tools of programming and dynamic modeling system of MatLab and contain descriptions to 6 laboratory works.

Methodical instructions are used at performance of laboratory works on discipline "Modeling and identification of control objects".

Ill - 17, table -20, bibliography – 10.

Reviewer: c.t.s., docent – G.D. Musapirova

Printed according to the plan for publications of NC JSC "Almaty university of power engineering and telecommunications" for 2019 y.

Introduction

The discipline "Software for automation systems" is studied by students of the “Automation and Control” specialty in the 2nd course in the composition of the basic disciplines (elective course), the volume of 3ECTS credits.

At present, for all specialties of technical profile, including future specialists (bachelors) in the field of automation and control, knowledge of modern computer technologies is needed, which is widely used in the design of automated control systems, modeling of control objects and development of monitoring systems of various technological processes.

The proposed methodical instructions for fulfilling the laboratory works contain 9 themes.

The subject of laboratory work is firstlydevoted to the practical study of the basics of work in the universal specialized system MATLAB - the set of applications for solving problems of technical calculations, modeling and control. Of course, any of these applications can be studied only after the initial study of work in MATLAB, which is dedicated to the first 3 laboratory works.

In the other three laboratory works, students are introduced to the basic concepts of dynamic systems and to programming methods. To solve such problems, various tools of MATLABareused. The obtained knowledge will help students in the future when using specialized applications MATLAB in the relevant disciplines of the specialty, as well as in the fulfilling the diploma.

Laboratory classes can cover only the main range of discipline issues. In accordance with the credit technology of training, students must study part of the material on their own at the IWS (independent work of the student) or at the IWST (independent work of the student under the guidance of the tutor). In the working program of discipline the subject of IWS is given.

The task options for the student is given by the tutor at the 1st lesson and remains unchanged when performing all subsequent works.

The referencesare given at the end of the guidelines.

Reports on laboratory works should be executed and issued in accordance with the requirements of the “NJSC Almaty University of Power Engineering and Communications”Corporate Standard56023-1910-04-2014 “Educational and methodical works. General requirements for the construction, presentation, design and content of educational and training works”.

1 Laboratory work №1. Introduction to MATLAB. The usual calculations

The goal of the work: to study the MATLABenvironment, data representation, simple numerical expressions and mathematical formulas.

1.1 The task for the laboratory work

In the process of executing the laboratory work the student must:
- examine the purpose and components of the MATLABuser interface;
- learn the MATLABobjects;
- learn the rules of data presentation;
- prepare a report on the work.

1.2 User interface

The MATLABsystem has a multi-window user interface, which contains a number of means of direct access to various components of the system (figure 1.1).

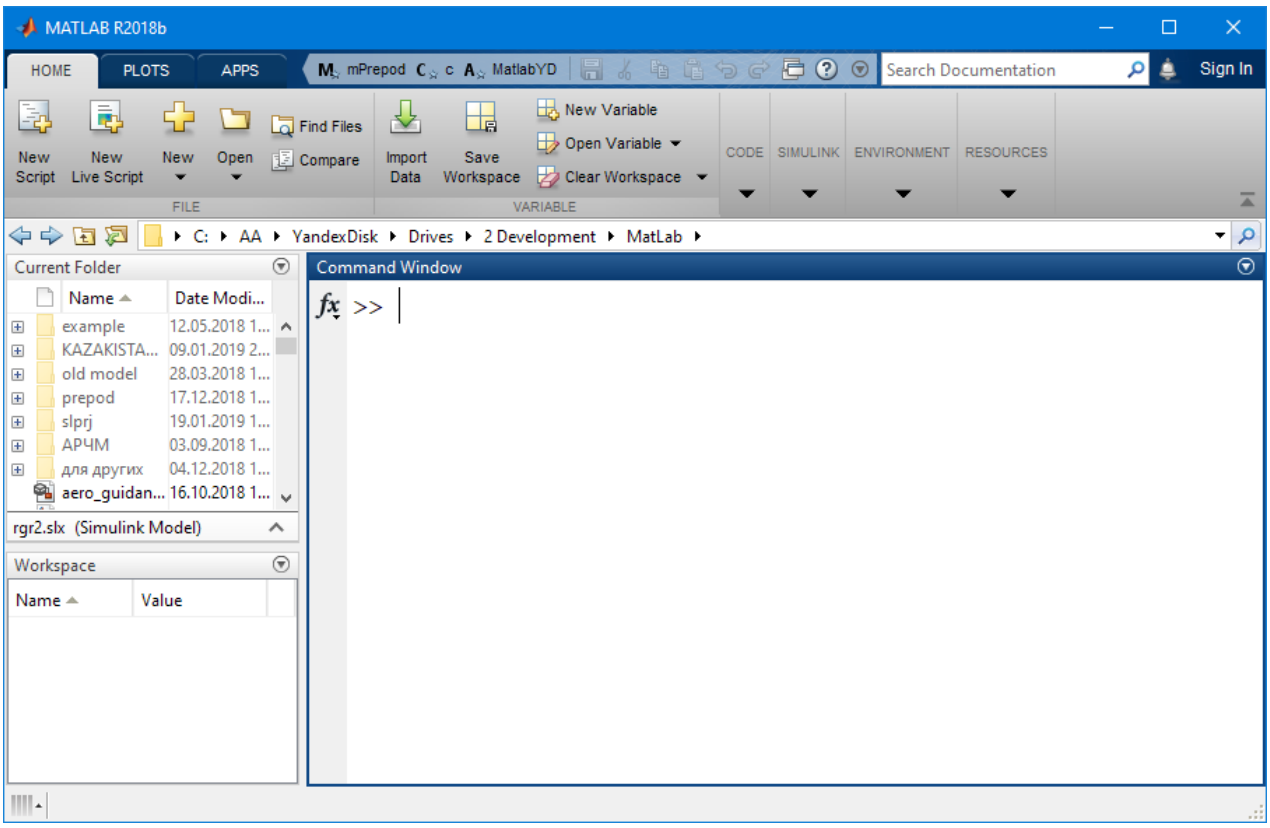
The main part of the system window is occupied by the *Command Window*, in which the input line is located, beginning with special characters ">>" (the symbol is entered automatically). This line contains commands be executed by the system.

When you first start up MATLAB, the workspace is empty. In the bottom left corner of the window contains the *Command History* window, which simply gives a chronological list of all MATLABcommands that you used, and the *Current Directory* window which shows you the contents and location of the directory you are currently working in. If necessary, you can run commands again by double-clicking the command in the command history window. Note that the current directory (or folder: folder and directory mean the same thing) is also displayed in the top right corner next to the main menu. If you like to see a demo of MATLAByou can type *demo* after the prompt.

The window of the MATLABworkspace -*Workspace* is shown in figure 1.2 and contains a list of variables (named arrays) accumulated in memory during operation, extension of the list of variables when accessing functions, executing M-files and loading saved variables.

Like all windows of MATLABdesktop, the workspace window is accompanied by a context menu that includes the following options:

1) OpenSelection.
2) GraphSelection.
3) SelectAll.
4) Import Data.
5) Save SelectionAs.
6) SaveWorkspace As.
7) DeleteSelection.
8)- DeleteWorkspace.



.

Figure 1.1 - MATLABwindow that contains the *Command Window*, the current *Workspace* and *Current Folder*of workspace

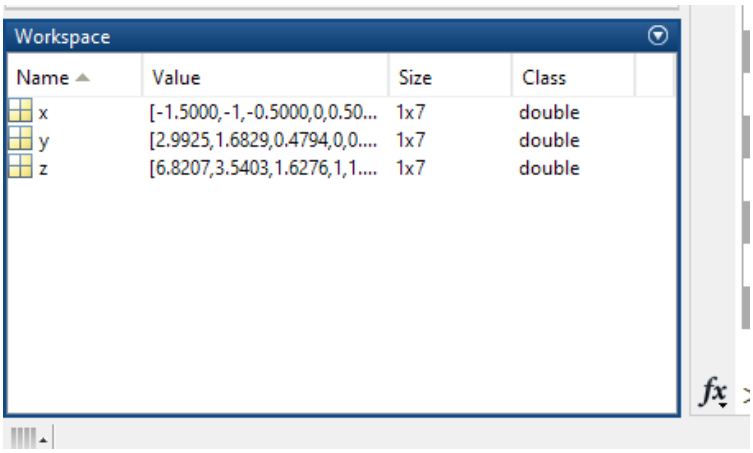


Figure 1.2 - *Workspace* window

MATLABcontains a powerful *help* subsystem:

- *Help* menu – *Help Window* or the corresponding button on the toolbar, and *help win*command;

- use *help win topic* to display the help window for individual topics;

- *help ops*displays a list of the MATLABoperators.

Another useful MATLABcommand is the *look for* command. You can use it to search for a command by keyword.

Managing the workspace.The contents of the workspace are saved between the execution of individual commands. Thus, the results of one task can affect the next. To avoid this possibility, it is recommended to give a clear command to clean the workspace at the beginning of each new independent calculation.

All variables created in the MATLABsession are stored in the MATLABworkspace. So that this workspace is not destroyed when you exit MATLAB, use *File–Save Workspace As* ... - all variables in the workspace will be saved in the current directory in the file with the extension.*mat*. These.*mat* files are binary files, which mean you can't edit or read them. You can reload the file at any time later using the command: *File-Import Data*. This method saves all variables in the workspace. You can select any number of variables.

The *dir* command gives you a list of all the files that are in the current directory.

1.3 Data representation in MATLAB

The main concept of all mathematical systems is *mathematical expression*. Mathematical expressions are based on numbers, constants, variables, operators, functions, and various special characters.

*Number*is a simple object of MATLABlanguage that represents quantitative data. Numbers are used in the conventional representation of them. The representation of numbers in MATLABfollows the rules of programming languages.

A constant is a predefined numeric or symbolic value represented by a unique name. There are some standard constants in MATLAB(table 1.1).

Table 1.1 – The predefined constant values

Variable name	Explanationofthevariable
pi	The number π=3,14159…
i,j	The imaginary unit, $\sqrt{-1}$.
inf	The infinity, ∞
NaN	Not a number

A character constant is any sequence of characters enclosed in an apostrophe, such as ' *Hello*!'.

Avariable is an object with a name that stores some data. Depending on this data, variables can be numeric or symbolic, vector or matrix.

The variable name must start with a letter, can contain letters, numbers, and an underscore. The name must not be the same as the names of other variables, functions, and procedures in the system. In MATLABuppercase and lowercase letters are different.

A special command is used to remove the variable definition:

- *clear*-deletes all variables;
- *clear x*-deletesvariable *x*;

- *clear x, y*-deletes*x* and *y* variables.

An operator is a special designation for a specific operation on data – *operands*. For example, arithmetic operators are signs of sum (+), subtraction (–), multiplication (*), division (/), exponentiation (^). Operators are used in conjunction with operands. For example, in the expression 2+3, the sign "+" is an addition operator, and the numbers 2 and 3 are operands.

An assignment operator is used to set a variable to a certain value:

Variable_Name = Value;

To display the value of a variable, enter its name in the command line and press *Enter*.A variable must have a value before it can be used. After you enter a variable, it is stored in the *Workspace* window. By double-clicking on the variable name in this window, you can view information about it (dimension, type).

If you do not assign an expression to a variable, the response is passed to a variable named ans (answer), which you can use later. The sign"; " suppresses the output of the result on the screen(not required).

Example of command input and result output (>>- system prompt, after this character the command is typed):

*>> 25*625*
ans =
15625

If you want to place multiple expressions on the same line, you can use the (,), (;) characters.

The character (%) is used for comments.

The *who* command displays all the names of the used variables.

The *whos* command displays more information about the variables.

Type the expression at the command prompt and press *Enter*for calculating the values of arithmetic expressions.Before you can calculate the value of a mathematical expression,you must determine the value of each variable included in it. A calculated expression can contain any number of variables, operators, and functions.

Arithmetic calculations in MATLABfollow the order accepted in mathematics. Use parentheses to change the order of actions.

A distinctive feature of MATLABis that if you enter any element correctly, this element is automatically stored in memory until the next change in the *Workspace*.

Format commands. By default, MATLABrepresents numbers with four decimal places. This is the so-called *short* format. If you need more precision, you must use the *format* command. The *formatlong*reflects 16 digits after the decimal point. *Formatbank* rounds a number to two decimal places.

Large numbers are displayed using an exponential representation. The *shorte* (or *short e*) *format* displays a number in exponential form with four decimal places and an exponent. The *longeformat* displays a number in exponential form with 16 decimal places and an exponent. Below are examples of working with formats:

Example 1.1 shows how to work with formats.

Example 1.1

```
>> format shorte
>> 5/3
ans =
  1.6667e+00

>> format longe
>> 5/3
ans =
   1.666666666666667e+00

>> format short
>> 5/3
ans =
1.6667

>>formatbank
>> 5/3
ans =
       1.67
```

Example 1.2.
Calculate the value of the expression:

$$z = \frac{x^2 + y}{3 - |sinx|} + 2,$$

where n *x= 25, y=3,6*.
The order of the input in *Command Window*:

```
>>x=25;
>>y=3.6;
>>z=(x^2+y)/(3-abs(sin(x)))+2
z =
 221.2040
```

The result is z=221.2040.

1.4 The order of fulfilling the laboratory work

1.4.1 Run MATLAB.Examine the user interface components.

1.4.2 Configure the MATLABWindow using the *Layout-Default*(figure 1.5) menu. Setting up MATLAB's windows can also be done using the button in the form of "arrows" (in the upper right corner).

1.4.3 Make settings (font, color, etc.) using the "*Preferences*" button.

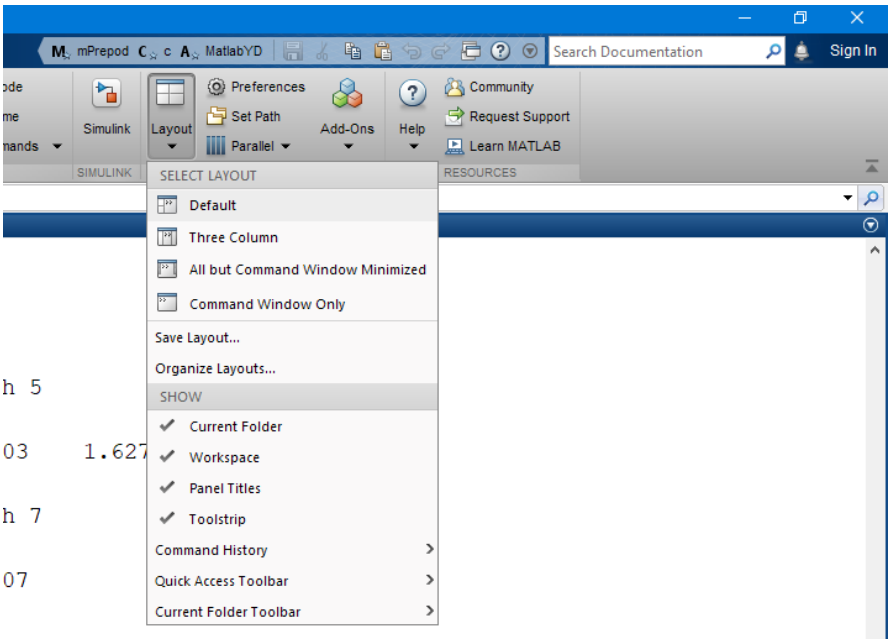


Figure 1.5 –MATLABwindow settings

1.4.4 Change the working (current)folder using the command

>> cd 'path to folder'

(for example,*>>cd 'c:\Users'*)or by the means of the current folder.

1.4.5Execute the operations with numbers:

- enter two numbers;

-addthese numbers;

- get the product of the result on any other number;

- assign the result to variable *a*;

- calculate the arithmetic expressions (by the option – table 1.2, 2^{nd} column);

- set the variable *x* and calculate the value of the given function (by the option – table 1.2, 3^{n} column); for another argument value, do the same action by placing a character «;» at the end of the expression;

- change the values of some arguments and calculate the value of the function without typing the expression again.

1.4.7 Review the list of used variables.

1.4.7 Receive full information about the variables.

1.4.8 Perform the actions to manage the variable: change its value; remove it from the workspace, etc.

1.4.9 Save the session, close MATLAB.

1.4.10 Download MATLAB, load your saved session.

Table 1.1 – Assignement options

№	Expression	Function	Interval [a,b] andsteph
1	$\frac{\left(12\frac{1}{6}-6\frac{1}{27}-5{,}25\right)13{,}5+0{,}111}{0{,}02}$	$f(x)=\frac{x^2}{1+0.25\sqrt{x}}$	[1,1..3,1], h =0.2
2	$\frac{\left(1\frac{1}{12}+2\frac{5}{32}+\frac{1}{24}\right):9{,}6+2{,}13}{0{,}00004}$	$f(x)=\frac{x^2+3}{2x+24}$	[1..5], h =0.5
3	$\frac{\left(6{,}6-3\frac{3}{14}\right)5\frac{5}{6}}{(21-1.25):2{,}5}$	$f(x)=\frac{x^3}{3x^2+\sqrt{x}}$	[0..3], h =0.1
4	$\frac{2.625-\frac{2}{3}\cdot 2\frac{5}{14}}{\left(3\frac{1}{12}+4.375\right):19\frac{8}{9}}$	$f(x)=\frac{x^3+3}{x+0.25\sqrt{x}}$	[0.5..5], h =0.2
5	$\frac{0{,}134+0{,}05}{18\frac{1}{6}-1\frac{11}{14}-\frac{2}{15}\cdot 2\frac{6}{7}}$	$f(x)=\frac{x^2}{x-0.5x+\sqrt{x}}$	[1..6], h =0.3
6	$\frac{\left(58\frac{4}{15}-56\frac{7}{24}\right):0{,}8+2\frac{1}{9}\cdot 0{,}225}{8{,}75\cdot 0{,}6}$	$f(x)=\frac{x^2+2x}{x^3-10x}$	[0..10], h =0.8
7	$\frac{\left(\frac{0.216}{0.15}+0.56\right):0{,}5}{\left(7.7:24.75+\frac{2}{15}\right)4.5}$	$f(x)=\frac{4}{x^3-0.25x}$	[1..8], h =0.6
8	$\frac{1\frac{4}{11}\cdot 0.22:0.3-0.96}{\left(0.2-\frac{3}{40}\right)1{,}6}$	$f(x)=\frac{10x^3+x}{5x+x^2}$	[0..6], h =0.4
9	$\frac{\left(\frac{3}{5}+0{,}425-0{,}005\right):0{,}12}{30{,}5+\frac{1}{6}+3\frac{1}{3}}$	$f(x)=\frac{x^2}{5x+0.25\sqrt{x}}$	[0..10], h =0.7
10	$\frac{3\frac{1}{3}+2{,}5}{2{,}5-1\frac{1}{3}}\div\left(\frac{0{,}05}{\frac{1}{7}-0{,}125}+5{,}7\right)$	$f(x)=\frac{10}{x+x^3}$	[1..8], h =0.5
11	$\frac{0{,}725+0{,}42}{0{,}128-6{,}25-(0{,}0345/0{,}12)}\cdot 0{,}25$	$f(x)=\frac{x^3}{7+5\sqrt{x}}$	[0..10], h =0.5

Continuation of the table 1.1

№	Expression	Function	Interval [a,b] andsteph
12	$\frac{\left(4{,}5 \cdot 1\frac{2}{3} - 6{,}75\right) \cdot 0{,}6}{\left(3{,}333 \cdot 0{,}3 + 0{,}222 \cdot \frac{4}{9}\right) \cdot 2\frac{2}{3}}$	$f(x) = \frac{x^2}{5x + 0.25}$	[1..9], h =0.4
13	$\frac{\left(5\frac{4}{45} - 4\frac{1}{6}\right) : 5\frac{8}{15}}{\left(4\frac{2}{3} + 0{,}75\right) \cdot 3\frac{9}{13}}$	$f(x) = \frac{x^2}{7 + 0.5x}$	[0..5], h =0.1
14	$\frac{\left(3\frac{4}{17} - 4\frac{1}{6}\right) : 5\frac{2}{3}}{\left(2\frac{2}{3} + 2{,}5\right) \cdot 3\frac{1}{5}} + \frac{2}{5}$	$f(x) = \frac{x^3 + x}{2x + 0.5}$	[0..6], h =0.2
15	$\frac{\left(12\frac{3}{8} + 45\frac{1}{24}\right) + 5\frac{2}{7} \cdot 0{,}5}{0{,}75 \cdot 0{,}6}$	$f(x) = \frac{x^2}{x - 0.5x^2}$	[1..10], h =0.3
16	$\frac{\left(4\frac{4}{7} - 10\frac{7}{33}\right) : \frac{2}{17} + 2\frac{1}{9} \cdot 0{,}225}{0{,}6\frac{3}{8}}$	$f(x) = \frac{10 + x}{x^2 + 4}$	[1.1..2], h =0.05
17	$\frac{(68{,}023 - 66{,}028) : 6\frac{1}{9} + \frac{7}{40} \cdot 4{,}5}{0{,}042 + 0{,}086}$	$f(x) = \frac{x}{x^3 + 0.47\sqrt{x}}$	[0.5..10], h =0.7
18	$\frac{(1{,}88 + 2{,}127) \cdot 0{,}01875}{0{,}625 - \frac{13}{18}/3{,}13} + 8{,}29$	$f(x) = \frac{x^2}{2x}$	[-3..3], h =0.2
19	$\frac{\frac{3}{0{,}4} - 0{,}009 : (0{,}15 : 2{,}5)}{0{,}32 \cdot 6 + 0{,}033 - (5{,}3 - 3{,}88)}$	$f(x) = \frac{x^2 + x}{0.25\sqrt{x}}$	[-5..5], h =0.5
20	$\frac{(34{,}06 - 33{,}81) \cdot 4}{6{,}84/(28{,}57 - 25{,}15)} + 1{,}33/\frac{4}{21}$	$f(x) = \frac{10x^2}{x + 5}$	[-5..5], h =0.4

1.5 Report on laboratory work

The report on laboratory workcontain:
- the task option;
- the screen shorts of work with comments;
- the screen shorts of all executed tasks.

1.6 Control questions

1.6.1 Explain the purpose of the MATLABsystem.
1.6.2 What is the Current Folder window?
1.6.3 What is Command History used for?
1.6.4 What is the purpose of the Workspace window?
1.6.5 How can display on the screen the specific elements of user interface?
1.6.6 How can I enter a variable into the command window?
1.6.7 List the rules for naming variables.
1.6.8 Name the reserved variables MATLAB.

1.6.9 Explain the rules for performing arithmetic operations.

1.6.10 How does the output of the result suppress on the screen?

2 Laboratory work №2. Working with matrices and vectors

*The goal of the work:*to study the work with matrices and vectors, as well as the use of mathematical functions in engineering calculations.

2.1 The task for laboratory work

In the process of executing the laboratory work the student must:

- explore mathematical functions in MATLAB;
- study various operations with vectors, matrices, complex variables;
- learn how to apply the studied functions for solution of engineering problems.

2.2 Array operations. MATLABmathematical functions

The storage structure of all data in MATLABis a *matrix*. A matrix variable in MATLABcan have any number of rows and columns. A *scalar* variable is also stored as a matrix with one row and one column. A *matrix* variable can be any variable, that is, it can be *a scalar, a vector* or *a matrix*.

There are two main types of vectors in MATLAB: *row vector* and *column vector*. A row vector stores its numbers "horizontally" and a column vector stores its numbers "vertically". These arrays are usually enclosed in square brackets.

Creating vectors and matrices. Vector is a one-dimensional array of numbers. A row vector is created by enclosing a set of elements in square brackets using a space or a comma between the elements. The column vector is created similarly, only the elements are separated by a semicolon. All indexes of the vector are numbered starting from 1.

Matrix is a two-dimensional array of numbers. In MATLAB, a matrix is created by entering each line as a sequence of numbers separated by a comma or space; the end of the line is indicated by a semicolon.

MATLABcan be used as a simple "pocket calculator" for matrices: you can quickly and easily multiply, add or subtract them. This is a very handy tool for checking matrix calculations.

MATLABallows you to create some matrices automatically, without typing each element:

- *zeros (m, n)* creates a matrix of dimension *mxn* with zero elements;
- *ones (m,n)* creates a matrix of dimension *mxn* with elements equal to one;
- *eye (m,n)* creates a diagonal matrix of dimension *mxn*;
- *rand (m,n)* creates a matrix of dimension *mxn*, the elements of which are random numbers between 0 and 1.

Mathematical functions. MATLABoffers many predefined mathematical functions for technical computing. You can use *help elfun* and *help specfun* to view a list of built-in elementary and special functions of MATLAB. All of the standard mathematical functions (often called *the elementary functions)* are available in MATLABusing their usual mathematical names (figure 2.1).

Table 2.1 – List of commonly used functions, where variables can be numbers, vectors and matrices

cos(x)	Cosine	abs(x)	Absolute value
sin(x)	Sine	sign(x)	Signum function
tan(x)	Tangent	max(x)	Maximum value
acos(x)	Arc cosine	min(x)	Minimum value
asin(x)	Arc sine	ceil(x)	Rount towards +∞
atan(x)	Arc tangent	floor(x)	Rount towards -∞
exp(x)	Exponential	round(x)	Rount to nearest integer
sqrt(x)	Square root	rem(x)	Remainder after division
log(x)	Natural logarithm	angle(x)	Phase angle
log10(x)	Common logarithm	conj(x)	Comples conjugate

Array arithmetic operations. In MATLABall arithmetic operations:+,−, * and ˆ can be applied to matrices. There are also arithmetic operations that allow calculations to be performed *element-by-element*. The list of such operations is given in the table 2.2.

Table 2.2 - Element-by-element operations with arrays

Notation	Purpose
.*	Element-by-element multiplication
./	Element-by-element division
.^	Element-by-element exponentiation

These operations – "*operations with dot*" - are performed as follows.If *A* and *B* are two matrices of the same size with elements $A = [a_{ij}]$ and $B = [b_{ij}]$, then the command>> *C=A.*B* produces another matrix *C* of the same size with elements $c_{ij} = a_{ij} * b_{ij}$.

To raise a scalar number to a power, the sign ^ is used, for example, *10^2*.If this operation should be applied to each element of the matrix, it should be used .^

Actions with matrices. Basic operators with matrices are performed in MATLABas well as matrix calculations in mathematics (addition, subtraction, transpose of matrix, matrix multiplication by number, product of two matrices, determination of the degree of the matrix), considering the rules of mathematics on the dimension of matrices to which these operations are applied.

Anotheroperators:

- *inv(A)* -computes the inversion of matrix;

- *det (A)* computes the determinant of a matrix;

- very original in MATLABare two unknowns in mathematics functions of dividing matrices. Thus, we introduce the concepts of division of matrices from left to right (‘/’) and right to left (‘\’).

B/A operation is equivalent to *B*inv(A); A\B* operation equals to *inv(A)*B.* So, in matrices slashes refer to inverses: if the slash is a backslash (\) then the inverse is taken of the preceding matrix; if it is a normal slash (/) then the inverse is taken of the following matrix;

In many engineering applications we need to solve matrix-vector equations. This is very easy in MATLAB, as MATLABwas especially designed for matrices.

For example, if the matrix equation (system of linear algebraic equations)*Ax=b* is given, where *A* is the matrix of the system, *b* is the vector of the right parts of the system, *x* is the vector of unknown variables, then to solve this system it is enough to enter the command *x=inv(A)*b* in the MATLABCommand Window. The components of the matrix *A* and the vector of the right parts *b*, of course, must be entered into the MATLABCommand Window.

Operations with vectors and matrices as arrays of data(processing of measurement data).Suppose we have a certain dependence *y(x)*which is given by the number of points. It can be specified as a matrix containing two rows - the values of *x* and values of *y*, call this matrix *xydata*.

Consider basic tools of data processing:

- *size(xydata)* - specifies the number of rows and columns of the matrix *xydata*;

- *max(v)* - returns the value of the maximum element value of the vector*v*;

- *min(v)* - returns the value of the minimum element value of the vector*v*;

- *sort(v)* -generates a vector whose elements are allocated in the ascending order of their values;

- *sum(v)* - calculates the sum of the elements of the vector*v*;

- *prod (v)* - calculates the product of all elements of a vector*v*;

- *diff (v)* -creates a vector whose elements are the difference between adjacent elements of a vector *v*; the size of the vector is one less than the size of the vector*v;*

- *cumsum(v)* - generates a vector, every element of which is the sum of all previous elements of a vector*v;*

- *cumprod(v)* -generates a vector, every element of which is the product of all the previous elements of a vector *v;*

- *mean(v)* -determines the average value of the elements of the vector *v;*

- *std(v)* - determines the standard deviation of the elements of the vector *v.*

These same functions *sum, max,min,sort,prod,diff,cumsum, cummprod, mean, std* can be applied to matrices. In this case, these operations are performed not in relation to the rows of the matrix, and with respect to each of the columns of the matrix (except for functions of *size*).

Functionsof a complex variable. Almost all elementary mathematical functions are evaluated at a complex value of the argument and get the complex

result value. Thanks to this feature, function *sqrt*, unlike other programming languages, calculates the square root of negative argument, and the function *abs* computes the modulus of a complex number.

In MATLABthere are several additional functions that are only designed for complex argument:

- *real(z)* - specifies the real part of complex argumentz;
- *imag(y)* - specifies the imaginary part of complex argumentz;
- *angle(z)* -calculates the value of the argument of a complex number z (in radians from $-\pi$ to π);
- *conj(z)* - produces a number, complex conjugate relative to z.

These functions allow you to build calculations with real numbers, the result of which is complex (for example, to find complex roots of quadratic equations).

The function tabulation. Tabulating a function is the calculation of all its values for each argument value in a specified range. To specify a range of numbers, you must write the name of the variable, put an assignment sign, and then after a colon write the initial value, step, and final value:

Variable_name=Initial value:Step:final value;

If the step is 1, you can specify only the initial and final values, instead of specifying it. After entering the range of values of the function argument, the function itself is set.

Since the argument has multiple values, the multiplication, division, and exponentiation operations must be performed *element-by-ele*ment. To do this, use the operations "with dot": ".* " ,"./ " ,".^» are used.

Example 2.1. Calculate values of function:

$y = 2xsin(x); z = 3x^2 + cos(x)$

for $x \in [-1{,}5; 1{,}5]$ *with step 0,5.*

The order of the input in *Command Wind*ow:

>>x=-1.5:0.5:1.5

*>>y=2*x.*sin(x)*

*>>z=3*x.^2+cos(x)*

2.3 The order of fulfilling the laboratory work

2.3.1 Run MATLAB.

2.3.2 Execute the following operations with vectors:

- enter any vector *v*. Type *n= size (v)* - this is an example of the MATLABfunction. Explain what this function defines;
- enter a column vector with three rows and one column;

- enter two vectors with three elements; apply the following operations to these vectors: +, -, ' , *;

- transpose column vector to row vector and vice versa using ' operation;

- form a column vector whose elements are the first 10 numbers using the command (1:10)'.

2.3.3 In MATLABis actively used sign «:»:

- enter h=10:2: 20 (without square brackets!). view the result;

- create the same vector in a different way;

- create a vector using the": "sign, but with a negative increment;

- create a vector using the": "sign, but without the increment.

2.3.4 Perform matrix operations:

- enter two 5x5 matrices *A* and *B* in the command window;

- find the inverse matrix to matrix *A*;

- find the product of the resulting matrix and matrix *B*;

- find the product of matrix *A* and the inversion of matrix *B*, name the result variable *C*;

- find the inversion of the matrix *C*, write the result to the variable *ans*, apply the inversion to this variable, compare the results of the last two actions;

- calculate the second degree of the matrix *A*, find the product of the matrix *A* on itself, and compare the results of the last two operations;

- calculate the negative second degree of the matrix *A*, find the product of the inverse matrix itself. Compare the results;

- calculate *sin*(*A*); explain the result;

- find the product of each element of matrix *A* for each element of matrix *B*;

- create a matrix *A* of dimension 2x2 and a matrix B of dimension 2x3;

- type *C=[A B]* and *D=[A,B];* type *E=[AB;B A]*. Explain the result;

- type *E=[A;B]*. Why can't this operation be performed?

- rewrite the first column of the matrix *A* in the variable *A2:A2 =A(1: 2,1*) - this operation can be performed by the command *A2 = A (:, 1).*Execute the command *A (2,:).* Explain the results.

2.3.5 Enter two square matrices *A*and*B* in the MATLABwindow. Calculate:

- product A*B;

- product B*A;

- sum of A+B;

- product 3*A;

- enter A=2. What will change?

2.3.6 Create a matrix of dimension 5x5, the elements of which are random numbers.

2.3.7 Create a diagonal matrix of dimension 4x4.

2.3.8 Create a matrix of dimension 10x10 whose elements are random numbers between 0 and 10. Replace all elements in the first row and first column on 1.

2.3.9 Create matrix *A*, transpose it in the matrix *B.*

2.3.10 Give examples of using the functions *zeros(m,n), ones(m,n), eye(m,n) and rand (m,n).*

2.3.11 Solution of system of linear algebraic equations:

- form a system of linear equations;
- enter the matrix of the system and vector of right parts in the Command Window;
- check that the determinant of the matrix is not equal to zero;
- solve this system in three ways: using the operation of dividingof a vector by a matrix; finding the inverse matrix A^{-1}; using the procedure*inv;* compare the results.

2.3.12 We need to calculate of the value of the function *y = a*exp(-k*x)*sin(x)* at values of the argument *x*from *0* to *10* with step *1* (*a* and *k* are chosen at their discretion).

2.3.13 Enter the vector *v* in the Command Window. Execute the following operations:

- find the size of *v;*
- determine the maximum and minimum elements of a vector;
- determine the sum and product of the elements of the vector.

2.3.14 Enter the matrix of 7x7 in the Command Window. Apply the operations of the previous paragraph to this matrix.

2.3.15 Enter in Command Window the complex numbers *u* and *z;* execute the following actions:

- using function "*disp*" reflect the sum of the complex numbers *u* and *z;*
- reflect the real, imaginary parts and arguments of these complex numbers;
- form two quadratic equations with positive and negative discriminant; solve these equations using Command Window;
- find the complex conjugate relatives to *u* and *z.*

2.4 Report on laboratory work

The report on laboratory work contains the screen shorts of the all executed tasks with comments.

2.5 Control questions

2.5.1 How are the complex variables in Command Window of MATLABentered?

2.5.2 How do the minimum, maximum or average value of the data arrayfind?

2.5.3 What is the difference between ordinary sign of the arithmetic operations from the operations with dot?

2.5.4 How does a system of linear algebraic equationssolvein MATLAB?

2.5.5 How does the inverse matrixfind?

2.5.6 How does the determinant of matrixfind?

2.5.7 How the elementary functions are applied to matrices and vectors?

2.5.8 How can we find the real, imaginary parts and arguments of complex numbers?

2.5.9 What data processing functions do you know? What is the difference in the application of these functions for matrices and vectors

2.5.10 Explain the concepts of division of matrices from left to right and right to left.

3 Laboratory work №3. Introduction to MATLABgraphics

The goal of the work: to learn the rules of creation of 2- and 3-dimenshional graphics in MATLAB.

3.1The task for laboratory work

In the process of executing the laboratory work the student must:

- learn how to create two-dimensional graphics;
- to acquire the skills of design charts;
- learn how to create two-dimensional graphics and design them.

3.2 MATLABgraphics

The graphics in MATLABhave the following features:

- high level, that is, does not require detailed knowledge of the graphics subsystem;
- object-oriented, that is, each object on the figure has properties that can be changed;
- access to the chart is possible both through viewing objects and using the built-in functions.

Two-dimensional graphics. MATLABhas a large number of functions associated with graphical output. The main function of ensuring the plotting on the display screen is a following function:

$$plot(x_1, y_1, s_1, x_2, y_2, s_2 \ldots),$$

where x_1, y_1 - the set of vectors whose elements are arrays of values of argument and function corresponding to the first curve;

x_2, y_2 - arrays of values of argument and function corresponding to the second curve and so on;

s_1, s_2, …- contain three special symbols that define type of the line connecting the points of the graph; type of chart data points and line color. This variables are symbolic, the indication of it is not necessarily. By default: type of line - straight, type of point – pixel.

The following additional parameters can be set for each chart in the plotting command (table 3.1).The parameters can be combined; in this case the line type is specified first, and then the marker type.

Graphics in MATLAB are always displayed in a separate window, which is called the *figure*. If you immediately draw another graph in the shape window, the old graph is removed from the graphics window.

Table 3.1 –Line properties

Line color		Marker type		Line type	
y	yellow	.	point	-	solid
m	pink	o	circle	:	dotted
c	cyan	x	x-mark	-.	dashdot
r	red	+	plus	--	dashed
g	green	*	star	(none)	non-line
b	blue	s	square		
w	white	d	diamond		
k	black	v	triangle		

Example 3.1

>> x = -pi : .01 : pi;

>> y = sin(x);

>> z = cos(x);

>>plot(x, y, 'r-.', x, z, 'm:')

To reflect the grid of coordinate lines on the chart, write *grid* after calling the *plot* function. The grid has an integer increment. The title of the chart is displayed by using the procedure of *title ('text').*Explanations along the horizontal axis is function *xlabel('x')* and along the vertical axis is *ylabel ('y')*.

This form of figures completely meets the requirements of engineering graphics.

Special graphics:

- if the function arguments are not defined when plotting the graph, then the system uses the vector element number as an argument;
- often the view is used in the form of bar charts - the function *bar*;
- to plot a function graph with discrete arguments, use the *stem* procedure;
- a graph of the function with discrete argument values we can get using the procedure *stem*;
- another useful function for engineers is the construction of a histogram of a given vector - *hist (y, x),* where *y* is the vector of the we are building; *x* is the vector that defines the intervals of the first vector.

This function counts the number of elements of the vector *y* whose values fall inside the range specified by the vector *x* and builds a bar graph of the counted numbers of elements of the vector *y* as a function specified by the ranges of vector *x*;

- procedure *subplot (m,n,p)* allows you to build in one graphic window, but on separate graphic fields multiple charts; here *m* specifies how many parts is divided into the graphic window vertically, *n* specifies how many parts is split the graphics window horizontally; *p* is the number of a sub windows in which to build the chart(example 3.2);
- *gtext (x,y,'text')* - to place text in the plot field, the beginning of text is placed in the point with coordinates *x* and *y* (here you can also use the menu);
- *figure* creates a new graphic window, leaving the previous (example 3.3);
- *hold on* is used to display several graphs in one window(example 3.4);
- *hold off disables* the mode set by the previous command;
- functions in the polar coordinate system. These functions are constructed similarly to graphs of functions in the Cartesian system. The *polar* command (example 3.5) is used to build such a graph.

Example 3.2

```
>>x = -pi : .01: pi;
>> y = sin(x);
>> z = cos(x);
>> subplot(211); plot(x, y, 'c')
>>subplot(212); plot(x, z, 'g')
```

Example 3.3

```
>>x = -pi : .01: pi;
>> y = sin(x);
>> z = cos(x);
>>plot(x, y)
>> figure
>>plot(x, z, 'r')
```

Example3.4

```
>>x = -pi : .01: pi;
>> y = sin(x);
>> z = cos(x);
>> plot(x, y)
>> hold on
>>plot (x, z)
```

Example 3.5

>>phi = 0 : .01: 15;
>> r = phi;
>>polar (phi, r, ‘g’)

Three-dimensional graphs is used to construct complex surfaces represented by a function of two variables *z=f(x,y).* Some of the used functions are listed in table 3.2. Other graphical functions can be found in the MATLABhelp system.

Table 3.2- Functions for construction of surfaces

Function	Use
mesh, surf	Buildingsurfaces
meshc, surfc	Builds a surface and contour diagram under it
meshz	The surface on the "pedestal"
surfl	Thehighlightedsurface
contour	Counter scheme
plot3	Three-dimensional line (parametric definition)
comet3	Movement on three-dimensional line

Examples of surface construction are given below.

Example 3.6.

[X,Y]=meshgrid(-5:0.1:5);
*Z=X.*sin(X+Y);*
meshc(X,Y,Z)

The first line specifies the location of the grid of the future surface with an interval of change *x* and *y* from -5 to 5 in increments of 0.1. If the *x* and *y* ranges are different, the functions are passed in two ranges. The second line defines an expression to calculate the *z*-values in the grid nodes. The third command plots a surface graph.

Example 3.7.

>> [X, Y] = meshgrid (-8: .5: 8);
>>R = sqrt (X.^2 + Y.^2) + eps;
>> Z = sin(R)./R;
>> surfl(Z)

You can change the function *z(x,y)* and get new graphic. Using the context menu, you can execute any command, including those that are not related to graphics.

3.3 The order of fulfilling the laboratory work

3.3.1 Run MATLAB.

3.3.2 Plot a graph of a given (by options in table 3.3, *Function 1* column) function in a given interval with a given step; place the chart title, axis labels on the chart.

3.3.3 Plot the same function as a bar chart.

3.3.4 Build a histogram of random variables on the interval [-2,9;2,9] with step 0,1.

3.3.5 Set the maximum and minimum ambient temperature in the vectors *"maxtemp"* and *"mintemp"* for the first ten days (vector *"date"*). Execute these steps:

- plot the graph of maximum temperature;
- plot the graph of minimum temperature on the second figure;
- build both graphs on the same figure;
- add title on the chart (use context menu);
- add labels to the axes;
- on the maximum temperature chart, change the line style, color, marker properties, etc.
- follow the steps in the previous paragraph for the minimum temperature graph;
- add a *legend* to the graph.

3.3.6 Execute the example 3.6. Repeat this example for another surface *z(x,y)* by your choice. Using the shape menu, make some changes on the chart (line type, color, etc.).

3.3.7 Create three-dimensional graphics, using the "peaks" function:

- close the current figure window;
- type *sampleplot = peaks(25)* ; a sampleplot matrix is created whose elements are the values of the *peaks* function;
- type *surf (sampleplot)* to create a 3D- graph of this functionvalues. The *z*-axis of the coordinate is directed vertically;
- change the background color of the chart: *Edit-Figure Properties*;
- click *Style*, select the desired color;
- select the *Title* and set it;
- click on the constructed surface, select *Edit - Current Object Property - Color* and change the color.

3.3.8 You can view the chart from different angles:

- select *Tools-Rotate 3D*, or click on the rotation button (far right next to the magnifying glasses);
- click anywhere in the figure and hold the cursor;

- move the cursor slightly. So you can control the point of view.

3.3.9 Build 2D-graphic of function specified in the interval *[a,b*] with increments of *h*(by options in the table 3.3, *Function 2* column). Formatthis charts.

3.3.10 Build a 3D graph of the function (by options in the table 3.4) for the arguments *x* and *y* given in the interval *[-2π,2π].*Format the graphics.

3.3.11 Build graphs of functions in the polar coordinate system and functions specified parametrically (options in the table 3.5).

Table 3.3 - Options of the tasks for building a two-dimensional graph

№	Function 1	Function 2	a	b	h
1	$y = sin(x)$	$z = exp(x+3)/5000 - 1$	-2π	2π	$\pi/20$
2	$y = cos(x)$	$z = 0.00025e^3-x - 0.6$	-2π	2π	$\pi/20$
3	$y = \|tg(x)\| + 0.1$	$z = (1+x)^6$	-2π	2π	$\pi/20$
4	$y = (x^2-1)/15$	$z = 1+sin(x)$	-2π	2π	$\pi/20$
5	$y = (x^3-2)/15$	$z = 5cos(x)$	-2π	2π	$\pi/20$
6	$y = x^2 - 10$	$z = 0.025exp(-1.2x)$	-5	5	1
7	$y = 3sin(x)$	$z=0.015x^3$	-5	5	1
8	$y = 4sin(x)$	$z = 0.05x^2$	1	10	1
9	$y = 6sin(x)$	$z = 0.01x^3$	-10	10	1
10	$y = 2+cos(x)$	$z = - 0.05(x^2 + 10cos(x))$	-8	8	1
11	$y = sin^2(x/3)$	$z = 0.01(x^2 - 40sin(x))$	-8	8	1
12	$y = cos^3(x)$	$z = sin(x) + sin(2x)$	$-\pi$	π	$\pi/8$
13	$y = 0.5x + cos^2(x)$	$z = sin^2(x) + cos(x)$	$-\pi$	π	$\pi/8$
14	$y = sin(x) + cos^2(2x)$	$z = x(0.5 + x)exp(0.1x)$	$-\pi$	π	$\pi/8$
15	$y = \|sin(x)\|exp(x/2)$	$z = 5x - x^{1.5}+sin(x)$	0	5	0.5
16	$y=sin(x)+cos(x)$	$Y=1-2cos(x)$	$-\pi$	π	$\pi/8$
17	$Y=cos^2(x)$	$Y=xsin(x)$	-2π	2π	$\pi/20$
18	$Y=(x-2)/(x+2)$	$Y=cosec(x)$	-5	5	1
19	$Y=1/2(x+\|x\|)$	$Y=x^2+1/x$	-8	8	1
20	$Y=101/x$	$Y=10/(x^2+1)$	-10	10	1

Table 3.4–Options of the tasks for building a three-dimensional graph

№	Function	№	Function
1	$z=sin(x)cos(y)$	1	$z=sin(x)cos(y)$
2	$z=sin(x)cos(y)$	2	$z=sin(x)cos(y)$
3	$z=sin(x/2)cos(y)$	3	$z=sin(x/2)cos(y)$

№	Function	№	Function
4	*z=sin(2x)cos(y)*	*4*	*z=sin(2x)cos(y)*
5	*z = sin(x)cos(y/2)*	*5*	*z = sin(x)cos(y/2)*
6	*z = sin(x/2)cos(2y)*	*6*	*z = sin(x/2)cos(2y)*
7	*z = sin(2x)cos(2y)*	***7***	*z = sin(2x)cos(2y)*
8	*z = (1+sin(x)/x)(sin(y)/y)*	*8*	*z = (1+sin(x)/x)(sin(y)/y)*
9	*z = (sin(x)/x)cos(y)*	*9*	*z = (sin(x)/x)cos(y)*
10	*z = (sin(x)/x)|cos(y)|*	*10*	*z = (sin(x)/x)|cos(y)|*

Table 3.5 – Options for assignments for graphing functions in polar coordinates and functions defined parametrically

№	Functions in polar coordinates	№	Functions defined parametrically
1	$r=\varphi/2$	11	$X=t^3\ y=t^2$
2	$r=e^{\varphi}$	12	$X=10\cos(t)\ y=\sin(t)$
3	$r=\pi/\varphi$	13	$X=10\cos^3 t\quad y=10\sin^3 t$
4	$r=2\cos(\varphi)$	14	$X=a(\cos t+t\sin t)\ y=a(\sin t-t\cos t)$
5	$r=1/\sin(\varphi)$	15	$X=at/(1+t^3)\ y=at^2/(1+t^3)$
6	$r=\sec^2\varphi/2$	16	$X=2_t+2_{-t}\ y=2_t-2_{-t}$
7	$r=10\sin(3\varphi)$	17	$X=2\cos^2 t\ y=2\sin^2 t$
8	$r=1/\sin(\varphi)$	18	$X=t-t^2\ y=t^2-t^3$
9	$r=a(1+\cos(\varphi))$	19	$X=a(2\cos t-\cos(2t))\ y=a(2\sin t-\sin(2t))$
10	$r^2=a^2\cos(2\varphi)$	20	$X=a/\mathrm{sqrt}(1+t^2)\ y=at/(\mathrm{sqrt}(1+t^2)$

3.4 Report on laboratory work

The report on laboratory work contains the screen shorts of the all executed tasks and explanations.

3.5 Control questions

3.5.1 How can you build the function graphic?
3.5.2 How can you set the title of graphic and the labels of axis?
3.5.3 How can set the color, font and font size in graphic?
3.5.4 How can you set the interval of argument changing to build the graphic?
3.5.5 How can you set the axis properties?
3.5.6How can you build several graphs on one figure??
3.5.7 What is the difference between a bar chart and a histogram?
3.5.8 What is necessary to build a three-dimensional graph??
3.5.9 What is the difference between two-dimensional and three-dimensional graphs?

3.5.10 How can you rotate a three-dimensional graph?

4 Laboratory work №4. Basics of Simulink

*The goalof the work:*to study the basic methods of work in Simulink.

4.1 The task for laboratory work

In the process of executing the laboratory work the student must to:
- learn the purpose and methods of work in Simulink;
-study of the procedure of creating of block-diagram;
- create of block-diagrams for the considered tasks;
- execute analysis of modeling; make conclusion.

4.1 Description of Simulink

Simulink is a graphical extension of MATLAB for the modeling and simulation of systems. In *Simulink*, systems are presented as *block- diagrams*.

Simulink is started from the MATLAB command prompt by entering the command *simulink*. Alternatively, you can click on the "*Simulink Library Browser*" button at the top of the MATLAB command window. As result, the Simulink *Library Browser window* should now appear on the screen (figure 4.1). Most of the blocks needed for modeling basic systems can be found in the subfolders of the main "*Simulink*" folder (opened by clicking on the "+" in front of "Simulink").

There are two main classes of elements in Simulink: *blocks* and *lines*. Blocks are used to generate, modify, combine, output, and display signals. Lines are used to transfer signals from one block to another.

Blocks. This folder contains the common block classes available for use:

1)*Continuous*: Linear, continuous-time system elements (integrators, transfer functions, state-space models, etc.).

2)*Discrete*: Linear, discrete-time system elements (integrators, transfer functions, StateSpace models, etc.).

3)*Functions & Tables*: User-defined functions and tables for interpolating function values.

4)*Math*: Mathematical operators (sum, gain, dot product, etc.).

5)*Nonlinear*: Nonlinear operators (coulomb/viscous friction, switches, relays, etc.).

6)*Signals & Systems*: Blocks for controlling/monitoring signal(s) and for creating subsystems.

7)*Sinks*: Used to output or display signals (displays, scopes, graphs, etc.).

8)*Sources*: Used to generate various signals (step, ramp, sinusoidal, etc.).

Blocks have zero to several input terminals and zero to several output terminals. Unused input terminals are indicated by a small open triangle. Unused output terminals are indicated by a small triangular point.

Lines. Lines transmit signals in the direction indicated by the arrow. Lines must always transmit signals from the output terminal of one block to the input terminal of another block.

Lines can never transmit a signal to another line; for this purpose, lines must be combined with another block, such as an added.

Signals can be scalar or vector. The lines used to transmit scalar and vector signals are identical. The type of signal transmitted by the line is determined by the blocks at both ends of the line.

The search for the necessary blocks is performed by typing the block name in the same window in the search field "Enter search item".

4.2Example of a creating a block-diagram

Create a block- diagram to solve the following differential equation:

$$x' = \sin(t)\,, x(0) = 0.$$

4.2.1 Open a new model window of *untitled* and save it: *New-Model.*

4.2.2 Drag and drop the following blocks from the *Library Browser* window and place them in the model window: *Sine Wave, Integrator, and Scope*.

4.2.3 Simulink allows you to configure the blocks in the model to more accurately reflect the characteristics of the analyzed system. Setting is performed by double-clicking on the block.

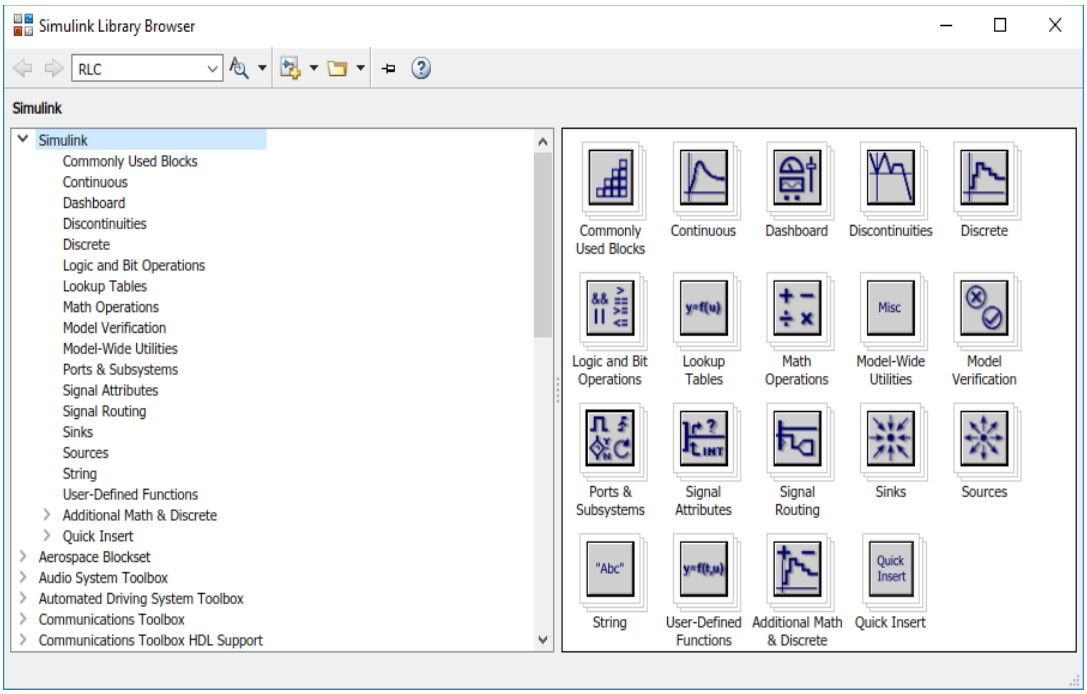


Figure 4.1 – Simulink Library Browser Window

Click the *Sine Wave* block. In the window that appears, set the amplitude, frequency, and phase of the sinusoidal input signal. The *Sample time* value specifies the time interval between sequential readings of the signal. Setting this value to 0 indicates a continuous sampling of the signal.

The initial condition of the equation is entered in the settings window of *Integrator* block.

4.2.4 Change the size of the blocks, perform some formatting operations..

4.2.5 Connect the blocks. Add an equation expression to the free space of the model window.

4.2.6 Adjust the simulation time. Run the model.

4.2.7 Configure parameters of the *Scope* block, for example: *Autoscale, Axes properties, Scope parameters*. View the result (double click on the *Scope* block).

4.2.8 Make the following changes to the model block-diagramm; after making the changes, run the simulation and examine the results:

- display both input and output signal graphs in the *Scope* window: select the *Parameters* tool in the Scope window and set the *Number* of axes parameter to 2. In this case, the *Scope* block will have additional inputs, connect one of the inputs to the input signal;

- to display graphs of input and output signals in the same coordinate axes, use a *Mux* block (multiplexer) that combines multiple signals into a vector;

- use the graph plotter *XYGraph* to plot a trajectory in phase space.

4.3 The order of fulfilling the laboratory work

4.2.1 Learn the components of the Simulink *Library Browser* and formatting operators:

- create diagrams to reflect different signals from the *Sinks* group in the *Scope* window;

- run these simple models; view the results;

- perform settings of different blocks;

- create diagrams to perform operations on inputsignals: multiplication by number, finding absolute value, addition, subtraction, division, ratio operations, logical operations, mathematical functions;

- perform formatting operations(using the context menu): change the font, color of blocks and lines, background, screen color;

- change the position of blocks and displaying them: flip block, rotate block, show/hide shadow, and show/hide port labels.

4.3.2 For learn of basic Simulink blocks, create a block-diagram for computation of the values of all the following functions:

1) $y = (1 + 2\sin(2t))^2$.

2) $y = \sqrt{1 + 0.5\sin(2t)}$.

3) $y = t * e^{-t^2} * \cos(2\pi t)$.

4) $y = 3.5 + 0.3t - 0.06t^2 - \sqrt{e^{-2t} + t^2}$.

5) $y = \min(5t, 100 - 2t^2$.

6) $y = \sqrt{|-100 + 20t|}$.

Reflect the results in the blocks: *Scope, Display, To Workspace, To File.*

4.3.3 Create the block-diagram for the solution of the differential equation of the second order (figure 4.2):

$$y'' + \sin(t) = 0, y'(0)=0, y(0)=0.5.$$

4.3.4 Add the right part to the equation, here is the function e^{-t}:

$$y'' + \sin(t) = e^{-t}, y'(0)=0, y(0)=0.5.$$

Modify the diagram in figure 4.2:

- add the *Sum* block: its input is supplied by input signal (figure 4.3);
- select the entire diagram and create a *Subsystem* (use the context menu) ; name the subsystem;

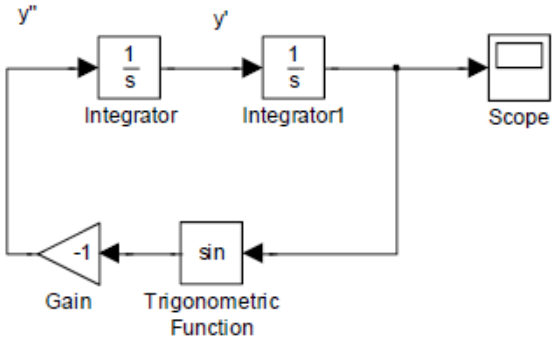


Figure 4.2 - Block-diagram for solution of the differential equation

- *In* and *Out* blocks are required for input and output signals (*Ports&Subsystems* library section);
- place the *Scope* block on the output;
- place the *Fcn* block at the input of the subsystem block;
- at the input of the Fcn block, place the block *Clock*;
- in the *Fcn* block, type the expression of input signal;
- run model. Analyze the results.

4.3.5 Execute the task by options.

4.3.6 Examine the purpose of the *Switch* block.

4.3.7 Create a block-diagram for the following task:

- set the simulation time *T*;
- place two *Fcn* blocks in which two different expressions are calculated (you can use the expression from p. 4.3.5 as one of them);

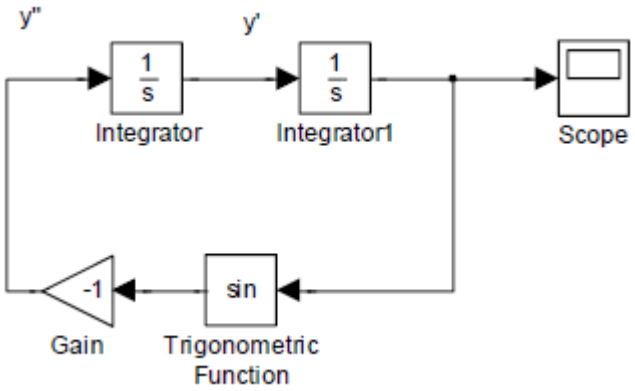


Figure 4.3 - Block-diagram for solution of the differential equationwith right part

- reflect on the *Scope* block at different times graphs of different expressions: up to the time *T*/2 – graph of the first expression, after the time *T/2*-graph of the second expression; to do this, use the *Switch* block;
- format the chart, place comments on it;
- run the chart; format the result in the viewing window as well.

Table 4.1 – Assignment options

№	Equation
1	$(1+x^2)y''-2xy'=0,\quad y=0, y'=3\ when x=0$
2	$1+y'^2=2yy'', y=1,\quad y'=1\ when x=1$
3	$yy''+y'^2=y'^3, y=1, y'=1\ when x=0$
4	$xy''=y',\quad y=0,\ y'=0\ when x=0$
№	Equation
5	$y''y^3=1, y=1,\quad y'=1\ when x=\frac{1}{2}$
6	$yy''-2xy'=0,\ y=0,\quad y'=3\ when\ x=0$
7	$xy''sqrt(1+y'^2)=0,\ y=0,\ x=1\quad y=1\ when x=e^2$
8	$y''(1+lnx)+\frac{1}{x\cdot y'}=2+lnx,\quad y=\frac{1}{2}, y'=1\ when x=1$
9	$y''=\frac{y'}{x\cdot\left(1+ln\left(\frac{y'}{x}\right)\right)},\quad y=\frac{1}{2},\quad y'=1\ when x=1$
10	$y''-y'^2+y'(y-1)=0,\quad y=2,\quad y'=2\ when x=0$
11	$3y'y''=y+y'^3+1,\quad y=-2,\ y'=0\ when x=0$
12	$y^2+y'^2-2yy'=0,\quad y=1, y'=1\ when x=0$
13	$y'y''+y'^2+yy''=0,\quad y=1, y'=-1\ when x=0$
14	$2y'+(y'^2-6x)y''=0,\quad y=0,\quad y'=2\ when x=2$
15	$y'y^2+yy''-y'^2=0,\quad y=1,\quad y'=2\ when x=0$
16	$2yy''-3y'^2=4y^2,\quad y=1,\quad y'=0\ when x=0$
17	$2yy''+y^2-y'^2=0,\quad y=1,\quad y'=1\ when x=0$
18	$y''=y'^2-y,\quad y=-\frac{1}{4},\quad y'=\frac{1}{2}\ when x=1$

19	$1 + yy'' + y'^2 = 0\,, \quad y = 0\,, y' = 1 \;\; when x = 1$
20	$(1 + yy')y'' = \left(1 + y'^2\right)y'\,, \quad y = 1, \quad y' = 1 \;\; when x = 0$

4.4 Report on laboratory work

The report on laboratory work contains block diagrams of the examples given in the description of laboratory work, and also tasks on option

4.5 Control questions

4.5.1 The purpose of Simulink.
4.5.2 Explain the procedure of creating of model block-diagram.
4.5.3 List of main parts of Library Browser of Simulink.
4.5.4 What blocks are used for inputs?
4.5.5 What blocks are used for outputs?
4.5.6 How is simulation timeset?
4.5.7 How is modelexecuted?
4.5.8 What is the Subsystem block for?
4.5.9 Give blocks and lines formatting and results viewing tools.
4.5.10 Explain the purpose and use of *Fcn, Mux, Gain, Switch, In, Out* blocks.

5 Laboratory work №5. Programming in MATLAB

The goal of the work: learn the types of program files in MATLAB and master working with them.

5.1 The task for laboratory work

In the process of executing the laboratory work the student must to:
- learn of *m*-filestypes;
- understandthe difference between function-files and script-files;
- get the skills to create *m*-files;
- learn relational and logical operators.

5.2 Types of M-files

We have used Matlabenvironment as a calculator. However, MATLAB is also a powerful programming language, as well as an interactive computational environment. Matlaballows to write series of commands into a file and executing the file as complete unit. Matlaballows to write two kinds of program files: scripts and function-files. These files have extension *.m* and are called *m* files.

Scripts are program files that contain a sequence of commands that must be executed together. Scripts do not have input variables and do not return any output. Theyworkwith the data in the command window.

All script files are M-files, but not all M-files are script files.

A *function* is a group of statements that together perform a task. In Matlab, functions are defined in separate files. Functions files are also program files with *.m* extension. You can use the Matlab editor or any other text editor to create your *.m* files. Function M-files must always start with the *function* statement and the name of the function file must always be the same as the name of the function.

Functions operate on variables within their own workspace, which is also called the *local* workspace, separate from the workspace you access at the Matlabcomm and prompt which is called the *base* workspace. Functions can accept more than one input arguments (*in*) and may return more than one output arguments *out*).

Syntax of a function statement is:

$function\ [out_1,out_2, ..., out_N] = myfun(in_1,in_2,in_3, ..., in_N),$

where *myfun*is name of function.

The first line of a function starts with the keyword *function*. It gives the name of the function and list of arguments. Immediately after this line you can place the comment lines (explanations to the code and algorithm of the program that begin and end with the %symbol)that appear when you type *helpmyfun*.

For a function, there is a concept of a local and global (or basic) workspace that is accessible from the MATLAB command line. A function can have more than one input variable (*in*) and return more than one output variable (*out*).

Example:

```
function f=myfunction(x)
f=log(x)/sin(x)
```

In the considered function named *myfunction*, the input argument is *x*, and the result of calculations is the output value *f*. The function file should be saved under the same name as the first line of the file. You can call the created function on the command line or in another function. For the above example, when the function is called, the result is reflected in the variable *f*:

```
>>myfunction(2)
f =
0.7623
```

5.3Logical expressions

Relational operators. Relational operators can work on both scalar and non-scalar data. Relational operators for arrays are performed element-by-element comparisons between two arrays and return a logical array of the same size, with elements equals 1 (*true*) when the relation is true and elements equals0 (*false*) in other case.

Table 5.1 -The relational operators available in MATLAB

Notation	Equal to	Less than or equal to	Less than	Greater than or equal to	Greater than	Not equal to
Operator	==	<=	<	>=	>	~=

Note that the "is equal to" operator consists of two equal signs and not a single "=" sign as you might expect.

Apart from the above-mentioned relational operators, MATLAB provides the following commands/functions used for the same purpose:

- *eq(a, b)* - tests whether *a* is equal to *b*;
- *ge(a, b)* - tests whether *a* is greater than or equal to *b*;
- *gt(a, b)* - tests whether *a* is greater than *b*;
- *le(a, b)* - tests whether *a* is less than or equal to *b*;
- *lt(a, b)* - tests whether *a* is less than *b*;
- *ne(a, b)* - tests whether *a* is not equal to *b*;
- *isequal*- tests arrays for equality;
- *isequaln* - tests arrays for equality, treating NaN values as equal.

Logical Operators. In MATLAB a logical expression has two possible values that are not "true" or "false" but numeric, i.e. 1- if the expression is true and 0 - if it is false

MatLab uses three logical operators. These are *AND (&), OR(|), NOT(~)*(table 5.2).

For & (and) to give a true result both expressions either side of the & must be true.

For | (or) to give a true result only one of the expressions either side of the | needs to be true.

The ~ (not) operator changes a logical expression from 0 to 1 and vice versa.

The ~ (not) operator changes a logical expression from 0 to 1 and vice versa. So result = ~(q<>== etc.

Table5.2 -Logicalrelationships

Operator	Condition	1st method	2nd method
Logical AND	x<3 AND y=4	and (x<3, y==4)	(x<3) & (y==4)
Logical OR	x=1 OR x=2	or (x==1, x==2)	(x==1) | (x==2)
NegationNOT	a≠1.9	not (a==1.9)	~(a==1.9)

As with relational operations, logical operations can be applied to vectors and matrices.

5.4 The order of fulfilling the laboratory work

5.4.1 Create a script file where executed following operations:
- product two matrices *A* and *B*;
- product matrix *A* on vector *v*;
- find matrix *C* which inverse to *B*;
- find a determinant of matrix *A*.

5.4.2 Create a function file to convert temperatures given in degrees Fahrenheit into degrees Celsius, according to the formula:

$$t_{Cel} = (t_{Fahr} - 32) \cdot 5/9;$$

- using this function finds the boiling point of water in Celsius (it is 212°in Fahrenheit);
- apply this function to any vector;
- apply this function to any matrix;
- explain the results.

5.4.3 Create the function files with more than one input variable for findingthe volume of a cuboid with the lengths *len*, breadth *br* and depth *dep*as the input variables.

5.4.4 Modify the previous function so that it calculates both the volume and the surface area of the cube; add comments; perform calculations and get results for multiple values of the input variables.

5.4.5Develop a flowchart of the algorithm to calculate the roots of the square equation. Create a file-function to implement this algorithm. Use this function to calculate the roots of the self-selected square equation. In the text of the function, type the command of checking of the equation discriminate.

5.4.6 Create a function that computes:
- the volume of a sphere with a certain radius;
- the surface area of the sphere;
- save function;
- compute the volumes and the surface areas of the spheres for several values of spheres radii.

5.4.7 In Command Window enter the scalar values *q, w, e*. Type:
w< e; q==e; (e>0) | (q<0); result = ~(q<0); (e>0) | (q<0).
Are the results as expected? Explain them.

5.4.8 Create the vectors *x* and *y*of 3 sizes. Type:
- z= (x < y);
- try z = ~(x < y);
- is the answer as you expected? Explain the result.

5.5 Report on laboratory work

The report on laboratory work contains the screen shorts of the all executed tasks and explanations. The student must demonstrate the work of the programs to the teacher.

5.6 Control questions

5.6.1 What is m-file?
5.6.2 What is the script-file?
5.6.3 What is function-file?
5.6.4 How can you run m-file?
5.6.5 Explain the difference between script and function-file.
5.6.6 What doesextension have m-files?
5.6.7 How can you edit m-file?
5.6.8 How can you set the comment in a m-file?
5.6.9 How can you view comments on an m-file when you use it?
5.6.10 What is an algorithm of a given problem? What is a flowchart?

6 Laboratory work № 6. MATLAB loops operators

The goal of the work: to master the technique of constructing branching algorithms and loops operators in MATLAB.

6.1 The task for laboratory work

In the process of executing the laboratory work the student must:
- learn *for* loops;
- learn the algorithms of using the decision making *if* operator;
- study the *while* loops;
- study the *switch* operator;
- understand the difference in the use of the above operators.

6.2 Control operators

Control operators are an important aspect in programming. These operators give choice and direction to computational processes. These operators include the conditional statements *if*, *while*, *switch*, and *for*. It should be noted that in the MATLAB there is no unconditional operator "*label*" (existing in other programming languages), which makes it difficult to move to the next or previous operator.

A kind of control structure designed to organize multiple execution of a set of instructions is called a *cycle*. A sequence of instructions designed for multiple execution is called a *loop body*. A single execution of the loop body is called *an*

iteration. The expression that determines whether the iteration will be executed again or the loop will end is called the *exit condition* or the *end condition* of the loop (or the *continuation condition*, depending on how its truth is interpreted — as a sign of the need to complete or continue the loop).

All loops with the *while, switch,* and *for* statements end with the *end* statement.

6.2.1 *"for"* loop.

This loop is designed to perform a certain number of repetitive actions. Its structure is as follows:

forx =initialvalue: step: final value
MATLAB statements
end

The counter can take not only integer values, but also real ones.

And also, the "*for*" loop can be used to calculate the accumulation of the amount (product). To do this, at the beginning of the loop, the variable in which the sum (product) is accumulated is assigned zero (or one for the product), then in the loop the next *i*-th element is added (multiplied) with the previous one.

"*For*" loops can be nested within each other, and nested loop variables must be different.

6.2.2 Decision making.

Decision making structures require that the programmer should specify one or more conditions to be evaluated or tested by the program, along with a statement or statements to be executed, if the condition is determined to be *true*, and optionally, other statements to be executed, if the condition is determined to be *false*.

MATLAB provides following types of decision-making statements:

1) if ... end statement consists of an *if* statement and a Boolean expression followed by one or more statements.

Syntax:

if <expression>
% statement(s)will execute if the Boolean expression is true
< statement(s)>
end

If the *expression* is *true*, the block of statements inside the *if* statement is executed. If the *expression* is *false*, the first statement after the *end* command is executed.

2) if...else...end statement. The *if* statement can be followed by an optional *else* statement, the operators following it are executed if the expression is false.

Syntax:

if <expression>
% statement(s)that are executed if the Boolean expression is true
<statement(s)>
else
<statement(s)>
% statement(s) that are executed if the Boolean expression is false
end

3) if...elseif...elseif...else...end statements. The *if* statement can be followed by one (or more) optional *elseif...* and *else* statements; useful to test various conditions.

Syntax:

if <expression 1>
% statements that are executed if condition 1 is true
<statement(s)>
elseif<expression 2>
% statements that are executed if condition 2 is true
<statement(s)>
elseif <expression 3>
% statements that are executed if condition 3 is true
<statement(s)>
else
% statements that are executed if none of the conditions are true
<statement(s)>
end

4) The nested if statements. You can use one *if* or *else if* statement inside another *if* or *else if* statement.

Syntax:

if < expression 1>
% statements that are executed if condition 1 is true
    if < expression 2>
    % statements that are executed if condition 2 is true
    end
end

6.2.4 The *while* loop.

The while loop is used when the number of necessary iterations in the loop body is unknown. It runs as long as the loop condition is satisfied:

while loop conditions
    MATLAB commands
end

The loop is also called the loop with a precondition. The following relationship operators are used to set the conditions(table 4.1, laboratory work №4).

6.2.5The *switch*operator.

The *switch*operator has the following structure:

Switch<expression, scalar or the symbols string>
case<value 1>
<operator 1>
case<value 2>
<operator 2>
…
otherwise
<operators>
end

The *switch*operator performs branching depending on the values in the lines *case*, and the computation of this value is in the line *switch*. There can be many fixed values, in this case we write the word *case* before each value. If the values calculated in the *switch* string do not match any fixed value, the statements following the word *otherwise* are executed.

6.3 Examples of using control operators

6.3.1An example of a program with a *for* loop.

Task 1. Plot a graph of the family of curves *z*, which is given by a function that depends on the variable *y*:

$$Z = 5 \cdot (y - y^2).$$

The variable y changes in the interval [1, 10] with step 1.

Program text:

>>y=[1:1:10];
$Z = 5 \cdot (y - y^2)$ *;*
hold on
plot(y,z)
end

The result of running the program is shown in figure 6.1.

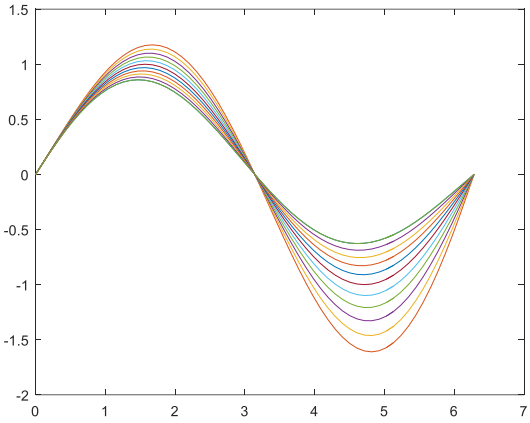


Figure 6.1 – Graphs of a given family of curves

Task 2. Calculate the sum $s = \sum_{k=1}^{10} \frac{1}{k!}$.

The text of file-function:

```
function sum
S=0;
for k=1:10
    S=S+1/factorial(k);
end
S
```

Run the program, and then the command window will answer:

```
S=
   1.7183
```

6.3.2 Example of using the conditional transition operator *if*.

Task. Calculate the values of the function f(x) depending on the value of the variable x by the following formula:

$$f(x) = \begin{cases} \dfrac{x}{(5x-1)}, when \;\; x < -4, \\ x^2 - x, \qquad when \;\; x \geq -4. \end{cases}$$

The text of the program is written to the m-file:

```
function f = myfunction( x )
if x<-4
   f1=x/(5*x-1)
else f2=x^2-x
end
```

After saving the program, you must type the name of the *m*-file in the command window and set the value of the variable in brackets (below the program is run for two values of the variable *x*):

```
>>myfunctions(-1)
f2 =
    2

>>myfunction(-5)
f1 =
   0.1923
```

6.3.3. An example of using a *while* statement.

Task. Calculate the sum of the series (series expansion function sin x) with an accuracy of 10^{-10} (that is, the sum considers only those members of the series that do not exceed 10^{-10}):

$$S(x) = \sum_{k=0}^{\infty}(-1)^k \frac{x^{2k+1}}{(2k+1)!}.$$

Text of *m*-file:

```
function y=mysinus(x)
s=0; k=0;
while abs(x.^(2*k+1)/factorial(2*k+1)) > 1.e-10
s=s+(-1)^k*(x.^(2*k+1)/factorial(2*k+1));
k=k+1;
end
s
```

Recall that the sign ";" after the operator suppresses the output of the result on the screen. In order not to display intermediate calculations, it is necessary to use this sign.

Program call:

```
>>mysinus(5)
```

As a result, the answer will be:

```
s =
   0.8415
```

6.3.4 Example of the use of the fork switch.

Task. Enter the value of a positive integer n. Compare this value with three fixed values of the variable *x*: -1; 0; 1.If the value of n is equal to one of the values of *x*, the case statements corresponding to these values will be executed. If the value of *n* does not match these values, statements after the *otherwise* keyword will be executed.

```
n = input ('Enter a number: ');
switch n
   case -1
disp('negative one')
   case 0
disp('zero')
   case 1
disp('positive one')
   otherwise
disp('other value')
end
```

6.4The order of fulfilling of the laboratory work

6.4.1 Learn the loop operators.

6.4.2 Create a flowchart of the comparison algorithms:

- two elements *a* and *b* using the "*if... end*" operator;
- two elements *a* and *b* using the "*if*...else...end" operator;
- create a script file to solve these problems.

You must place comments for the function and the output variables.

6.4.3Create a flowchart of the algorithm to compare these numbers sequentially with several others using the "*if...elseif...elseif...else...end*" statement. Create a function-file to solve this problem, provide it with comments.

6.4.4 Execute the examples of6.4 p.

6.4.3 According to option, create a file-function to calculate the sum of the series using different operators: *if* and *for* (table 6.1); *while* (table 6.2); *switch* (table 6.3). Separate *m*-filesare created for each operator.

Table6.1 – Options for "*for"*loop

Option №	Function
1	$Z = 4 \cdot (1 - y^2),\quad where\ y_0 = 1.5,\quad \Delta y = 0.5,\quad y_k = 4$
2	$Z = (x + 1)^3 - 2,\quad where\ x_0 = 0.5,\quad \Delta x = 0.5,\quad x_k = 4$
3	$Z = 1 + 2 \cdot x - x^2,\quad where\ x_0 = 0,\quad \Delta x = 0.5,\quad x_k = 4.5$
4	$Z = \sqrt{\lvert x + 1\rvert},\quad where\ x_0 = -4,\quad \Delta x = 1,\quad x_k = 4$
5	$Z = \sqrt{4 \cdot x^2 + 1},\quad where\ x_0 = 0,\quad \Delta x = 0.2,\quad x_k = 2.4$
6	$Z = e^{x^2} - 1,\quad where\ x_0 = -2,\quad \Delta x = 0.2,\quad x_k = 0$
7	$Z = 2 \cdot \sqrt{x + 1} \cdot cos(x),\quad where\ x_0 = 0,\quad \Delta x = 0.25,\quad x_k = 3.5$
8	$Z = ln(x) + 5 \cdot x,\quad where\ x_0 = 0.5,\quad \Delta x = 0.5,\quad x_k = 6$

Continuation of the table 6.1

9	$Z = 6 \cdot (1 - y^4), \quad where\ y_0 = 1, \quad \Delta y = 0.5, \quad y_k = 4.5$
10	$Z = (x+1)^3 - 4 \cdot x, \quad where\ x_0 = 0, \quad \Delta x = 0.5, \quad x_k = 4$
11	$Z = 4 + 2 \cdot x - 3 \cdot x^2, \quad where\ x_0 = 0, \quad \Delta x = 0.5, \quad x_k = 4.5$
12	$Z = \sqrt{\lvert 2 \cdot x + 1 \rvert}, \quad where\ x_0 = -4, \quad \Delta x = 1, \quad x_k = 4$
13	$Z = \sqrt{\frac{(4 \cdot x^2 + 3}{x}}, \quad where\ x_0 = 0, \quad \Delta x = 0.2, \quad x_k = 2.4$
14	$Z = 2 \cdot x \cdot e^{x^2} - 1, \quad where\ x_0 = -2, \quad \Delta x = 0.2, \quad x_k = 0$
15	$Z = 2 \cdot \sqrt{x+1} \cdot sin(2x), \quad where\ x_0 = 0, \quad \Delta x = 0.25, \quad x_k = 3.5$
16	$Z = ln(2x) + 5 \cdot x, \quad where\ x_0 = 0.5, \quad \Delta x = 0.5, \quad x_k = 6$
17	$Z = \frac{x}{\sqrt[3]{x^2 - 4}}, \quad where\ x_0 = 0, \quad \Delta x = 0.2, \quad y_k = 3$
18	$Z = x(x-1)^2(x-2)^3, \quad where\ x_0 = 0, \quad \Delta y = 0.25, \quad y_k = 4$
19	$Z = 2e^{x^2 - 4x}, \quad where\ x_0 = 1, \quad \Delta x = 0.5, \quad x_k = 5$
20	$Z = 2x^3 + 3x^2 - 12x + 5, \quad where\ x_0 = 0, \quad \Delta x = 0.4, \quad y_k = 4$

Table6.2 – Options for *if* statement

№	Function	№	Function
1	$f(x) = \begin{cases} 4 \cdot \sin(x) - 1, when\ x < 1 \\ ln(x) + 5, \quad when\ x \geq 1 \end{cases}$	7	$f(x) = \begin{cases} \sqrt{4 \cdot x^4 + 3}, & when\ x < 2 \\ e^{2 \cdot x} - 0.5, & when\ x \geq 2 \end{cases}$
2	$f(x) = \begin{cases} 25 - x, & when\ x < 0 \\ \cos^2(x) - x, & when\ x \geq 0 \end{cases}$	8	$f(x) = \begin{cases} \frac{(3 \cdot x^3 + 5)}{(3 \cdot x + 1)}, & when\ x < -1 \\ ln\lvert x + 4 \rvert, & when\ x \geq -1 \end{cases}$
3	$f(x) = \begin{cases} \sqrt{4 \cdot x^2 + 1}, & when\ x < 2 \\ e^x - 1, & when\ x \geq 0 \end{cases}$	9	$f(x) = \begin{cases} \frac{2.5 \cdot x}{(6 \cdot x + 4)}, & when\ x < -2 \\ \frac{(x^2 - 2)}{5}, & when\ x \geq -2 \end{cases}$
4	$f(x) = \begin{cases} \frac{(x^2 + 5)}{(x+1)}, & when\ x < -1 \\ ln\lvert x + 2 \rvert, & when\ x \geq -1 \end{cases}$	10	$f(x) = \begin{cases} -2x + 6, & when\ x < 10 \\ \cos^2(3x) - 2x, when\ x \geq 10 \end{cases}$
5	$f(x) = \begin{cases} \frac{x}{(3x+2)}, when\ x < -2 \\ x^2 - 2, \quad when\ x \geq -2 \end{cases}$	11	$f(x) = \begin{cases} \frac{\sqrt{(4 \cdot x^2 + 3}}{8}, & when\ x < 3 \\ 3 \cdot x \cdot e^{5 \cdot x} + 1, when\ x \geq 3 \end{cases}$
6	$f(x) = \begin{cases} -x + 5, & when\ x < 10 \\ \cos^2(x) + 2x, & when\ x \geq 10 \end{cases}$	12	$f(x) = \begin{cases} 3 - 5 \cdot x \cdot \cos(x), & when\ x < 5 \\ \frac{(x^2 + 2)}{6 \cdot x}, & when\ x \geq 5 \end{cases}$
13	$f(x) = \begin{cases} \sqrt{4 \cdot x^2 + 3}, & when\ x < 3 \\ e^x + 1, & when\ x \geq 3 \end{cases}$	17	$f(x) = \begin{cases} 7 \cdot \cos(x) + 2, & when\ x < 1 \\ 2 + (x-1)^3, & when\ x \geq 1 \end{cases}$
14	$f(x) = \begin{cases} 1 - \cos(x), when\ x < 5 \\ x^2 + 2, \quad when\ x \geq 5 \end{cases}$	18	$f(x) = \begin{cases} \sin\left(x - \frac{\pi}{4}\right), & when\ x < 0 \\ 2 + x - x^2, & when\ x \geq 0 \end{cases}$

Continuation of the table 6.2

№	Function	№	Function
15	$f(x)=\begin{cases}6\cdot cos(x)+1, when\ \ x<3\\ ln(2x)+15, \quad when\ \ x\geq 3\end{cases}$	19	$f(x)=\begin{cases}\dfrac{(2x-3)}{(3x+2)}, & when\ \ x<-1\\ ln\lvert x+6\rvert+5, & when\ \ x\geq -1\end{cases}$
16	$f(x)=\begin{cases}\dfrac{30-8\cdot x}{1.5}, & when\ \ x<0\\ \cos^2(x)-6\cdot x\,, when\ \ x\geq 0\end{cases}$	20	$f(x)=\begin{cases}2x^2-x^4, & when\ \ x<2\\ \mathrm{tg}^2(x)\,, & when\ \ x\geq 2\end{cases}$

Table6.3 – Options for *while* loop

Option №	Series	Accuracy and *x* variable values
1	$\sum_{n=0}^{\infty}\frac{x^n}{n}$	x=[0..10]; Accuracy10^{-10}
2	$\sum_{n=0}^{\infty}(-1)^{n-1}\frac{x^n}{n}$	x=[-5..5]ж Accuracy10^{-10}
3	$\sum_{n=0}^{\infty}\frac{x^{2n-1}}{2n-1}$	x=[-2..10]; Accuracy10^{-10}
4	$\sum_{n=0}^{\infty}(-1)^{n-1}\frac{x^{2n-1}}{2n-1}$	x=[-7..7]; Accuracy10^{-10}
5	$\sum_{n=0}^{\infty}(n+1)x^n$	x=[0..15]; Accuracy10^{-10}
6	$\sum_{n=0}^{\infty}(-1)^{n-1}(2n-1)x^{2n-2}$	x=[5..20]; Accuracy10^{-10}
7	$\sum_{n=0}^{\infty}n(n+1)x^{(n-1)}$	x=[0..10]; Accuracy10^{-10}
8	$\sum_{n=0}^{\infty}\frac{n}{x^n}$	x=[1..15]; Accuracy10^{-10}
9	$\sum_{n=0}^{\infty}\frac{x^{4n-3}}{(4n-3)}$	x=[-6..5]; Accuracy10^{-10}
10	$\sum_{n=0}^{\infty}\frac{(-1)^{n-1}}{(2n-1)3^{n-1}}$	x=[5..25]; Accuracy10^{-9}
11	$\sum_{n=0}^{\infty}\frac{2n-1}{2^n}$	x=[0..15]; Accuracy10^{-8}
12	$\sum_{n=0}^{\infty}\frac{(1+n)}{3^n}$	x=[2..20]; Accuracy10^{-9}

Continuation of the table 6.3

Option №	Series	Accuracy and *x* variable values
13	$\sum_{n=0}^{\infty} \frac{(-1)^n - n}{3^n}$	x=[-10..10]; Accuracy10^{-8}
14	$\sum_{n=1}^{\infty} \frac{(1+n)}{3^n}$	x=[-15..1]; Accuracy10^{-9}
15	$\sum_{n=1}^{\infty} \frac{n}{(2n+1)5^n}$	x=[0..50]; Accuracy10^{-8}
16	$\sum_{n=1}^{\infty} \frac{1}{n(n+1)}$	x=[50..100]; Accuracy10^{-9}
17	$\sum_{n=1}^{\infty} (\sqrt[2n+1]{x} - \sqrt[2n-1]{x}$	x=[1..15]; Accuracy10^{-10}
18	$\sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n}$	x=[0..10] Accuracy10^{-8}
19	$\sum_{n=1}^{\infty} \frac{1}{n!\,2^n}$	x=[-5..5]; Accuracy10^{-9}
20	$\sum_{n=1}^{\infty} \frac{(-1)^n}{n!}$	x=[-1..15]; Accuracy10^{-8}

Table6.4 – Options for *switch* operator

№	Function	№	Finction
1	$f(x) = \begin{cases} 4 \cdot \sin(x) - 1, when\ x = 1 \\ ln(x) + 5, \quad when\ x = 2 \end{cases}$	10	$f(x) = \begin{cases} \sqrt{4 \cdot x^4 + 3}, & when\ x = 2 \\ e^{2 \cdot x} - 0.5, & when\ x = 3 \end{cases}$
2	$f(x) = \begin{cases} 25 - x, & when\ x = 0 \\ \cos^2(x) - x, & when\ x = 5 \end{cases}$	11	$f(x) = \begin{cases} \frac{(3 \cdot x^3 + 5)}{(3 \cdot x + 1)}, & when\ x = -1 \\ ln\lvert x + 4\rvert, & when\ x = 1 \end{cases}$
3	$f(x) = \begin{cases} \frac{(x^2 + 5)}{(x + 1)}, & when\ x = -1 \\ ln\lvert x + 2\rvert, & when\ x = 1 \end{cases}$	12	$f(x) = \begin{cases} -2x + 6, & when\ x = 10 \\ \cos^2(3x) - 2x, when\ x = -10 \end{cases}$
4	$f(x) = \begin{cases} \frac{x}{(3x + 2)}, when\ x = -2 \\ x^2 - 2, \quad when\ x = 2 \end{cases}$	13	$f(x) = \begin{cases} \frac{\sqrt{(4 \cdot x^2 + 3}}{8}, & when\ x = 3 \\ 3 \cdot x \cdot e^{5 \cdot x} + 1, when\ x = -3 \end{cases}$
5	$f(x) = \begin{cases} -x + 5, & when\ x = -10 \\ \cos^2(x) + 2x, & when\ x = 10 \end{cases}$	14	$f(x) = \begin{cases} 3 - 5 \cdot x \cdot \cos(x), & when\ x = 5 \\ \frac{(x^2 + 2)}{6 \cdot x}, & when\ x = -5 \end{cases}$

Continuation of the table 6.4

№	Function	№	Finction
6	$f(x) = \begin{cases} \sqrt{4 \cdot x^2 + 3}, & when\ x = 3 \\ e^x + 1, & when\ x = -3 \end{cases}$	15	$f(x) = \begin{cases} 7 \cdot \cos(x) + 2, & when\ x = 1 \\ 2 + (x-1)^3, & when\ x = -1 \end{cases}$
7	$f(x) = \begin{cases} 1 - \cos(x), when\ x = 5 \\ x^2 + 2, & when\ x = -5 \end{cases}$	26	$f(x) = \begin{cases} \sin\left(x - \frac{\pi}{4}\right), & when\ x = 0 \\ 2 + x - x^2, & when\ x = 10 \end{cases}$
8	$f(x) = \begin{cases} 6 \cdot cos(x) + 1, when\ x = 3 \\ ln(2x) + 15, & when\ x = -3 \end{cases}$	17	$f(x) = \begin{cases} \frac{(2x-3)}{(3x+2)}, & when\ x = -1 \\ ln\lvert x + 6\rvert + 5, & when\ x = 1 \end{cases}$
9	$f(x) = \begin{cases} \frac{30 - 8 \cdot x}{1.5}, & when\ x = 0 \\ \cos^2(x) - 6 \cdot x, when\ x = 1 \end{cases}$	18	$f(x) = \begin{cases} 2x^2 - x^4, & when\ x = 2 \\ \mathrm{tg}^2(x), & when\ x = -2 \end{cases}$

6.5 Report on laboratory work

The report on laboratory work contains the screen shorts of the all executed tasks and explanations. The student must demonstrate the work of the programs to the teacher.

6.6 Control questions

6.6.1 What are the control operators?
6.6.2 Define the loop.
6.6.3 Define the iteration.
6.6.4 What is the loop body?
6.6.5 Explain the *if* statement.
6.6.6 What kinds of *if* statement do you know?
6.6.7 Explain the operation of the *for* loop.
6.6.8 Explain the operation of the *while* loop.
6.6.9 Explain the operation of the *switch* statement.
6.6.10 What is the difference between a cycle with a precondition and a cycle with a given number of repetitions?

References

1 Hunt Brian R., A Guide to Matlab : for Beginners and Experienced Users: updated for Matlab 8 and Simulink 8 / R. Hunt Brian , L. Lipsman Roland , M. Rosenberg Jonathan; All of the University of Maryland, College Park. - 3-edition. - United Kingdom: Cambridge University Press, 2014. –P. 317.

2 Matlab: Офиц. учебный курс Кембриджского университета / пер.с англ. - М. : Триумф, 2008. – 352с.

3 Васильев А.Н., Matlab. Практический подход: Самоучитель / А.Н. Васильев. - СПб.: Наука и Техника, 2012. – 448с.

4 Гайдук А.Р., Теория автоматического управления в примерах и задачах с решениями в MATLAB / А.Р. Гайдук, В.Е. Беляев, Т.А. Пьявченко. - 2-е изд.испр. - М.: Горячая линия-Телеком, 2011. – 464с.

5 Дьяконов В.П., MATLAB и SIMULINK для радиоинженеров / В.П. Дьяконов. – М.: ДМК Пресс, 2013. – 976с.

6 Кетков Ю., Matlab 7: Программирование, численные методы / Ю. Кетков , А. Кетков , М. Шульц. - СПб. : БХВ-Петербург, 2005. – 742с.

7 Кудинов Ю.И., Теория автоматического управления (с использованием MATLAB-SIMULINK): учеб.пособие / Ю.И. Кудинов, Ф.Ф. Пащенко. - СПб. : Лань, 2016. – 256с. - (Учебники для вузов. Специальная литература)

8 Мансурова М.Е., Основы программирования в Matlab: учеб.пособие / М.Е. Мансурова, К. Дуйсебекова; КазНУим.Аль-Фараби. - Алматы: Қазақуниверситеті, 2010. – 149с.

9 Шампайн Л.Ф., Решение обыкновенных дифференциальных уравнений с использованием Matlab: учеб.пособие / Л.Ф. Шампайн, И. Гладвел , С. Томпсон; пер.с англ. И.А.Макарова. - СПб.: Лань, 2009. – 304с. - (Учебники для вузов. Специальная литература).

10 http://matlab.exponenta.ru/simulink/book2/15.php

Content

Main plan 2019, pos.112

Lida Kuandykovna Ibrayeva
Laulasyn Kosylganovna Abzhanova
Azamat Zamirovich Ilyasov

SOFTWARE FOR AUTOMATION SYSTEMS

Methodical guidelines to laboratory works for students of specialty
05070200 - Automation and Control

Editor Kurmanbekova M.D.
Specialist for standardization Mukhametsaryeva G.I.

Signed for printing __.__.__.

Format60x84 1/16
Printing paper №1

Circulation 30 copies

Order . Price 2090tg.

Copying Bureau of the Noncommercial Joint-Stock Company
"Almaty University of Power Engineering and Telecommunications"
050013 Almaty, 126/1, Baitursynovstr.