

MINISTRY OF SCIENCE AND HIGHER EDUCATION
OF THE RUSSIAN FEDERATION

FEDERAL STATE AUTONOMOUS EDUCATIONAL
INSTITUTION OF HIGHER EDUCATION
"NORTH CAUCASUS FEDERAL UNIVERSITY"

SUGGESTIONS FOR TEACHING
for the performance of practical work
discipline "Intelligent Control System"
for students of the field of study

11.03.02 Infocommunication Technologies and Communication Systems,
Specialization "Intelligent Data Processing in Infocommunication Systems and Networks"

Stavropol, 2026

Table of Contents

Introduction.....	5
PracticalLesson No1. Classification of knowledge. Subject area research.....	7
Practical Lesson No2. Identifying Knowledge in Artificial Intelligence Systems. Fuzzy Logic. Forming Belonging Functions in the Fuzzy Logic Toolbox Software Environment	16
Practical Lesson No3. Building Models in Artificial Intelligence Systems (Prolog Declarative Language)	23
Practical Lesson No4. Products in Artificial Intelligence Systems	35
Practical Lesson No5. Frame models of knowledge representation.....	41
Practical Lesson No6. Neural Networks in Artificial Intelligence Systems. Approximation of Neural Network Functions	45
References	57

Introduction

The guidelines are devoted to the study of the theoretical foundations and practical tools used in intelligent control systems in the direction of 11.03.02 in the study of the discipline "Intelligent Control Systems".

Practical training is one of the key forms of training aimed at developing students' independence, acquiring skills and forming general and professional competencies. The performance of practical tasks contributes to an in-depth understanding, assimilation and consolidation of the material of the professional module, the development of thinking and creative potential. The purpose of practical classes in academic disciplines is to consolidate the theoretical material studied by students and develop skills for independent professional and research activities. The tasks of practical classes are due to the need for students to obtain knowledge, skills, and abilities in accordance with the requirements of the Federal State Educational Standards of Higher Education, on the basis of which the relevant professional competencies are formed. The leading didactic goal of practical classes is the formation of skills necessary in subsequent educational activities in general professional, professional disciplines. Methodological instructions for the implementation of work in practical classes are developed according to the sections of the work program. Preparation for practical work consists in independent study of theory according to the recommended literature provided for by the work program.

To effectively complete tasks, students must know theoretical materials and be able to apply this knowledge to acquire skills in performing practical tasks.

Conditions and procedure for performing work:

- 1 To study the methodological recommendations for the implementation of practical classes.
- 2 Answer the list of questions necessary to complete the tasks.
- 3 Study the content of the tasks and start performing.
- 4 Get advice on the performance of the work from the teacher.
- 5 The work is evaluated as a whole, and a grade is given based on the results of the work.

The defense is carried out by means of an individual conversation or the completion of a test task.

The process of performing practical tasks is aimed at the formation of the competence of PC-1, PC-2.

Practical Lesson №1. Classification knowledge. Subject area research

The purpose of the lesson: to study the given subject area and build a model of knowledge in the form of a graph.

Brief theoretical information

The main purpose of knowledge representation in an intelligent information system is to organize it in such a way that it is possible to easily access knowledge for effective decision-making, recognition of objects and situations, inference and other cognitive functions.

In the most general case, a semantic network is an oriented graph, the vertices of which are generalized concepts, events, properties of objects or actions. The vertices of the graph are connected by an arc if the corresponding information units are in some kind of interaction. Arcs, therefore, characterize relations, connections between objects and phenomena.

Semantic models of knowledge representation are based on a construct called a semantic network. The semantic approach is used in natural language recognition systems, in various domain-oriented research systems, and in question-and-answer systems.

In general, the term "semantic" means "semantic", because semantics is a science that establishes relations between objects and the symbols that denote them, i.e. determines the semantic meaning of signs [1].

In the semantic network, arcs can be defined by various methods, the choice of a particular method depends on the type of knowledge being represented. Often the arcs used to represent the hierarchy include arcs of the "set" type, "subset", "element". If the semantic network is used as a
Models of knowledge representation in the description of natural languages (translators) use arcs such as "object", "agent", or "recipient".

Concepts in a semantic network can be objects (abstract or concrete), and relations can be connections like "is", "has part", "belongs". For a network to be semantic, the presence of three types of relations is mandatory: "property – value", "class – class element", "example of a class element".

The main characteristics of semantic networks as models of knowledge representation are:

- objects are described in natural language;
- all knowledge is accumulated in a relatively homogeneous structure of memory;
- Unified relationships between objects are defined on networks, and unified inference methods correspond to them;
- Methods of inference in accordance with requests determine the areas of semantic knowledge related to the task, formulating an act of understanding the request and a certain chain of conclusions corresponding to the solution of the problem.

There are various classifications of semantic networks. By the number of types of relationships:

- homogeneous (there is only one type of relationship in the network);
 - heterogeneous (there are different types of relations in the network).
- By types of relations:
- binary (in this case, two objects are connected by relations);
 - paired (there are special relations that connect more than two concepts).

The following relationships are most commonly used in semantic networks:

- "part-whole" relations ("element - set", "class - subclass", and etc.);
- quantitative ("less", "greater", "equal", etc.);
- spatial ("close to", "far from", "above", "behind", "under", etc.);
- functional connections (usually defined by the verbs "produces", "influences", etc.);

- temporary ("earlier", "later", "during", etc.);
- attributive connections ("to have a meaning", "to have a property", etc.);
- logical connections ("and", "or", "not"), etc.

When solving problems in semantic networks, the problem of finding a solution in the knowledge base is reduced to the problem of finding a fragment of the network corresponding to a certain subnetwork that relates to the question posed [2].

Methodical instructions for work

To build a knowledge representation model in the form of a graph, you need to perform the following steps sequentially:

- 1 Determine the target actions of the task (which are solutions).
- 2 Determine the intermediate actions or the chain of actions between the initial state and the final state (between what is available and the target action).
- 3 Anticipate the conditions for each action in which it is possible to perform it purposefully and in a way that can be performed.
- 4 Determine the order of actions.
- 5 Add specific facts based on the stated objective.
- 6 Transform the received order of actions and the corresponding facts, conditions and actions.
- 7 To check the correctness of the construction, write down the chains, clearly tracing the connections between them. Such a set of steps assumes movement in the construction of the model from the result to the initial state, but it is also possible to move from the initial state to the result (steps 1 and 2).
- 8 Assign designations to facts F, rules P, actions D.
- 9 Build a domain graph.

Example

To build a model for the presentation of knowledge in the subject area "Restaurant" (visiting a restaurant).

To build a knowledge representation model, you need to complete the following steps:

1 Determine the target actions of the task (which are solutions).

2 Determine the intermediate actions or the chain of actions between the initial state and the final state (between what is available and the target action).

3 Anticipate the conditions for each action in which it is possible to perform it. Determine the general order of performing actions.

4 Add specifics, if necessary, based on the set dachas.

5 Transform the received procedure and the corresponding conditions in the product.

6 To check the correctness of the product structure, write down the product chains, clearly tracing the links between them.

This set of steps involves movement in the construction of a production model from the result to the initial state, but it is also possible to move from the initial state to the result (steps 1 and 2).

Let us consider the solution of the problem for all the above points. there.

1 A mandatory action performed in restaurants is the consumption of food and its payment. This means that there are already two target actions "eat food" and "pay", which are interrelated and follow each other.

2 Before eating anything in a restaurant, you need to come there, wait for the waiter and place an order. In addition, you need to choose which restaurant to go to. So, the chain of intermediate actions: "choosing a restaurant and going there", "making an order to the waiter".

3 Before going to a restaurant, you need to make sure that there is the necessary amount of money. The choice of a restaurant can be determined by many reasons, let's choose a territorial feature - to which one we are closer to the one we are going. Different people work in different restaurants, so depending on the choice of restaurant, the waiters will be different.

In addition, different restaurants specialize in different cuisines, so the ordered dishes will differ in different restaurants. This means that first there are actions that allow you to choose a restaurant, then characterizing restaurants, and only after ordering food and paying for the order.

4 Let's say that the problem will consider two restaurants: "Delicious Food" and "Yummy". The first is a pub and orders are brought faster than in the second, the second is a pizzeria. In the first one there is a waiter Sergey, and in the second one there is a waitress Marina. Peter is a client.

5 The above can be transformed into the following sentences like "If, then":

If the subject is hungry and the subject has a sufficient amount of money, then the subject can go to a restaurant.

If the subject is closer to the restaurant "Vkusnaya eda" than to the restaurant "Vkus-nyatina", and the subject can go to the restaurant, then the subject goes to the restaurant "Vkusnaya eda".

If the subject is closer to the restaurant "Vkusnyatina" than to the restaurant "Tasty Food", and the subject can go to the restaurant, then the subject goes to the restaurant "Vkusnyati-na".

If the subject goes to the restaurant "Vkusnyatina", and the waiter Marina works at the restaurant "Vkusnyatina", then Marina takes the order from the subject.

If the subject goes to the restaurant "Tasty Food" and the waiter Sergey works at the restaurant "Delicious Food", then Sergey takes the order from the subject.

If the subject has chosen the dishes, and Marina accepts the order from the subject, then the order will be brought in 20 minutes.

If the subject has chosen the dishes, and Sergey accepts the order from the subject, then the order will be brought in 10 minutes.

If the order is brought in 20 minutes or the order is brought in 10 minutes, then the subject can eat.

If the subject can eat, then after eating, the subject is obliged to pay for the order. Let us introduce notations for facts (F), actions (E) and products (P),

then:

Subject = Peter;

F1 = the subject wants to eat;

F2 = the subject has a sufficient amount of money;

F3 = the subject is closer to the restaurant "Vkusnaya eda" than to "Vkusnyatina"; F4= the waiter Marina works in the restaurant "Vkusnyatina";

F5 = in the restaurant "Vkusnaya Eda" there is a waiter Sergey;

F6 = the subject has chosen the dishes;

D1= the subject can go to the restaurant;

D2=the subject goes to the restaurant

"Vkusnaya eda"; D3=the subject goes to the restaurant "Vkusnyatina"; D4= Marina takes the order from the subject; D5=Sergey accepts the order from the subject; D6=the order will be brought in 20 minutes.

D7=the order will be delivered in 10 minutes.

D8=after the meal, the subject must pay for the order.

For products, we will set the priority (in parentheses before the comma, the higher the priority, the earlier the rule is checked).

P1(4, F1 and F2)= D1;

P2(5, F3 and D1)= D2;

P3(4, not F3 and D1)= D3;

P4(3, D3 and F4)= D4;

P5(3, D2 and F5)= D5;

P6(2, D4)= D6;

P7(2, D5)= D7;

P8(1, D6 or D7) = D8;

6 To show the relationship between products, let's build a graph (Figure 1).

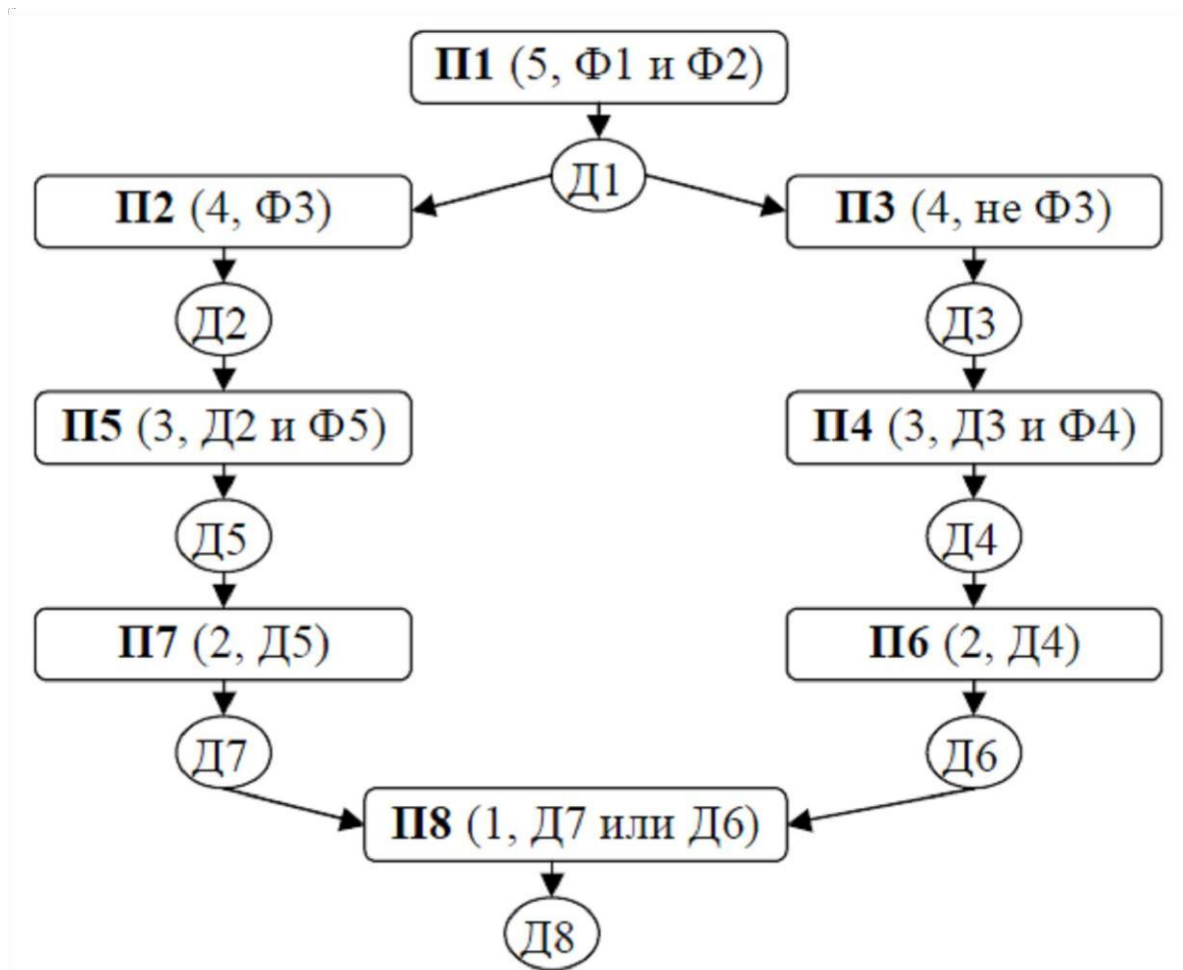


Figure 1 – Knowledge model graph

Task

Complete the tasks according to the specified option.

1 Variant: to build a model of knowledge representation in the subject area "Railway" (ticket sales).

2 Option: to build a model of knowledge representation in the subject area "Shopping Center" (organization).

3 Variant: to build a model of knowledge representation in the subject area "Gas station" (customer service).

4 Option: to build a model of knowledge representation in the subject area "Computer Networks" (organization).

5 Option: to build a model of knowledge representation in the subject area "University" (educational process).

6 Variant: to build a model of knowledge representation in the subject area "Computer Security" (means and methods of its provision).

7 Option: to build a model of knowledge representation in the subject area "Computer Security" (threats).

8 Variant: to build a model of knowledge representation in the subject area "Internet café" (organization and maintenance).

9 Variant: to build a model of knowledge representation in the subject area "Development of information systems" (conducting an information project).

10 Option: to build a model of knowledge representation in the subject area "Travel Agency" (work with clients).

11 Variant: to build a model of knowledge representation in the subject area "Kitchen" (cooking).

12 Variant: to build a model of knowledge representation in the subject area "Hospital" (reception of patients).

13 Variant: to build a model of knowledge representation in the subject area "Film distribution" (assortment and work with customers).

14 Option: to build a model of knowledge representation in the subject area "Car Rental" (assortment and work with customers).

15 Option: to build a model of knowledge representation in the subject area "Operating systems" (functioning).

16 Option: to build a model of knowledge representation in the subject area "Information Systems" (types and functioning).

17 Variant: to build a model of knowledge representation in the subject area "Enterprise" (structure and functioning).

Control questions and tasks

1 What is a semantic web?

2 What can be a concept in a semantic network?

3 List the main characteristics of semantic networks.

4 List the steps involved in building a semantic network.

Practical Lesson No2. Identifying Knowledge in Artificial Intelligence Systems. Fuzzy Logic. Forming Belonging Functions in the Fuzzy Logic Toolbox Software Environment

The purpose of the lesson: to get acquainted with the ways and means of describing odd sets and products in the odd output system in the interactive mode of using the graphic tools of the Fuzzy Logic Toolbox package (in MATLAB R2008b).

Brief theoretical information

To consider the results of the development and functioning of odd logic systems, we will use the graphical tools of the Fuzzy Logic Toolbox package. The same tools are used in the development of odd logic systems as a graphical object-oriented language for automatic programming.

These products include:

- FIS Editor (FIS) – Membership Function Editor (MFE)
- Rule Editor;
- Rule Viewer
- Sur-face Viewer. To describe odd utterances, non-odd linguistics are used.

variables (LP).

LP is a named variable that takes its values from a set of linguistic terms, i.e. symbolic values. For an odd LP, a term set is given as an odd set. This process is called fuzzification.

Fuzzification is one of the problematic tasks of describing an odd conclusion and reflects the author's individual empirical knowledge. Odd

Statements in the conditional part of an odd product can be composite, connected by the copula "AND" and/or "OR". In the calculation of utterances, these conjunctions are realized by logical or arithmetic operations of intersection or unification, respectively.

When obtaining a result for each rule, it is necessary to give an estimate of the degree of its truth. This assessment depends on the degree of truth of the statements of the conditional part of the rule, the degree of truth of the relation underlying the rule between the initial statements (premise) and the conclusion, i.e. the degree of truth of the implication, and the degree of truth of the statement relative to the value of the term set of possible results given in the rule. Obtaining an estimate of the degree of truth of the conclusion obtained by the rule is called activation.

If it is necessary to obtain an even quantitative value of the result, it can be obtained on the basis of the term-set function of the result in various ways according to algorithms named after their authors (Mamdani, Sugeno, Tsukamoto, etc.), which determines the type of the odd inference system. This operation is called defuzzification.

The Accessory Function Editor in graphical mode allows you to set and change the affiliation function of any fuzzy inference (SNS) LP terms [3].

Methodical instructions for work

To fuzzify the linguistic variable SNV, select its image – a named rectangle in the upper left part of the editor window.

In the editor window, graphs of the ownership functions for all values of the selected LP are displayed (by default for three values).

Three fields are used to describe the function of belonging to each LP value: Name, Type and Params. The described function is highlighted by clicking on its graph. In the Name field, the value of LP is set. In the Type field, by selecting the menu item, the name of the required function belongs to the

(one of the 11 built-in). In the Params input field, you specify the necessary parameters of the membership function, which determine the position of its modal values on the numeric scale, the range of which is specified in the Range and Display range input fields.

These operations are performed on all values from the term sets of the START linguistic variables.

Adding a new LP value with a built-in affiliation function is done by using the Edit > Add MF main menu command.

Deleting an unnecessary LP value is done by pressing the De-lete key, after selecting the graph of the function belonging to this value.

Example

The Fuzzy command (function) starts the main interface program of the Fuzzy Logic package from the command line mode – the Fuzzy Inference System Editor (FIS Editor, FIS Editor).

The view of the window that opens in this case is shown in Figure 2.

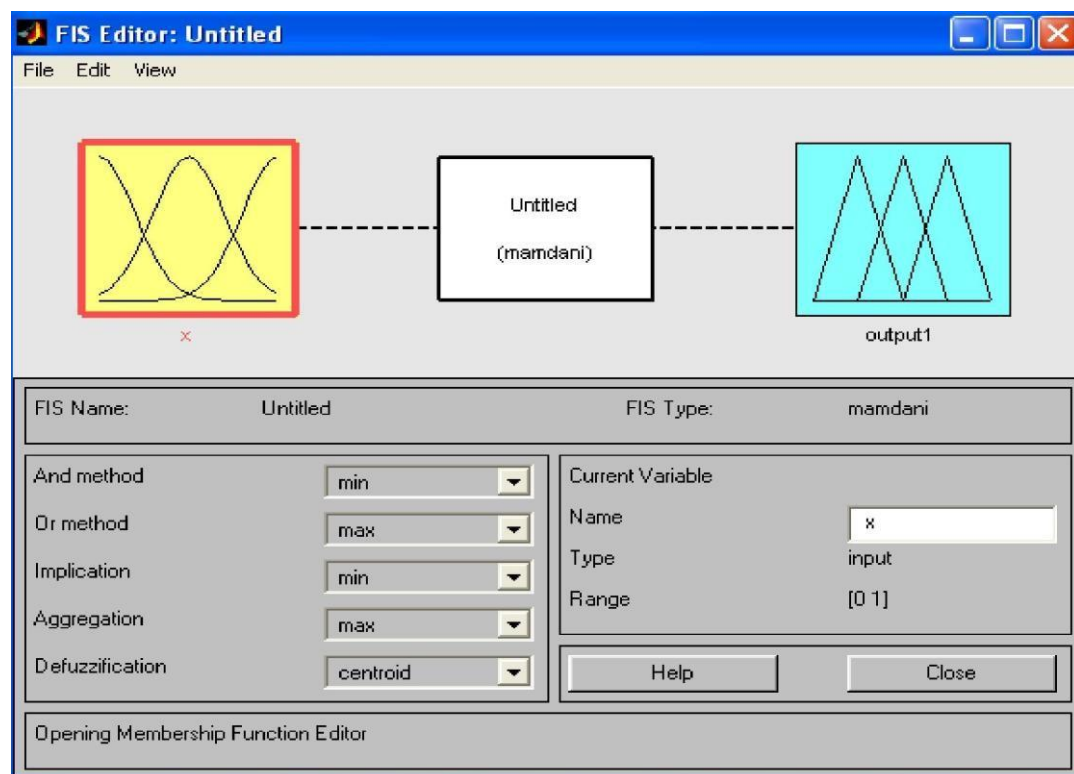


Figure 2 – Fis Editor window view

The main menu of the editor contains the following items:

File – work with model files (their creation, saving, reading and printing);

Edit – editing operations (adding and excluding input and output variables);

View – go to additional tools.

It is necessary to develop a fuzzy system that displays the relationship between the variables x and y , given using Table 1, the data presented in the table reflect the dependence $y = x^2$.

Table 1 – Values of x and y

x	-1	- 0.6	0	0.4	1
y	1	0.36	0	0.16	1

The required actions should be displayed in the following paragraphs.

1 In the File menu item, select the New Sugeno FIS option, and in the block displayed as a white square, Untitled2 (sugeno) will appear in the upper part of the editor window.

2 Left-click on the block called inputl. Then, on the right side of the editor, in the field entitled Name, instead of inputl, enter the designation of the argument, i.e. x . If you now click the mouse once somewhere (outside the editor blocks), the name of the marked block will change to x ; the same is achieved by pressing the Enter key after typing.

3 Double-click on this block. In this case, the Membership Function Editor window will open (see Figure 3). You need to enter the Edit menu item of this editor and select the Add Membership Functions option in it. This will bring up a dialog box (Figure 4) that allows you to specify the MF type and Number of MFs of the membership functions (in this

In this case, everything refers to the input signal, i.e., to the variable x). It is necessary to

Select the Gaussian Membership Function (Gaussmf) and set the number of Gaussian functions to five, based on the number of argument values in Table 1. Then you must confirm the entry of the information by clicking OK, after which you will return to the Membership Function Editor window.

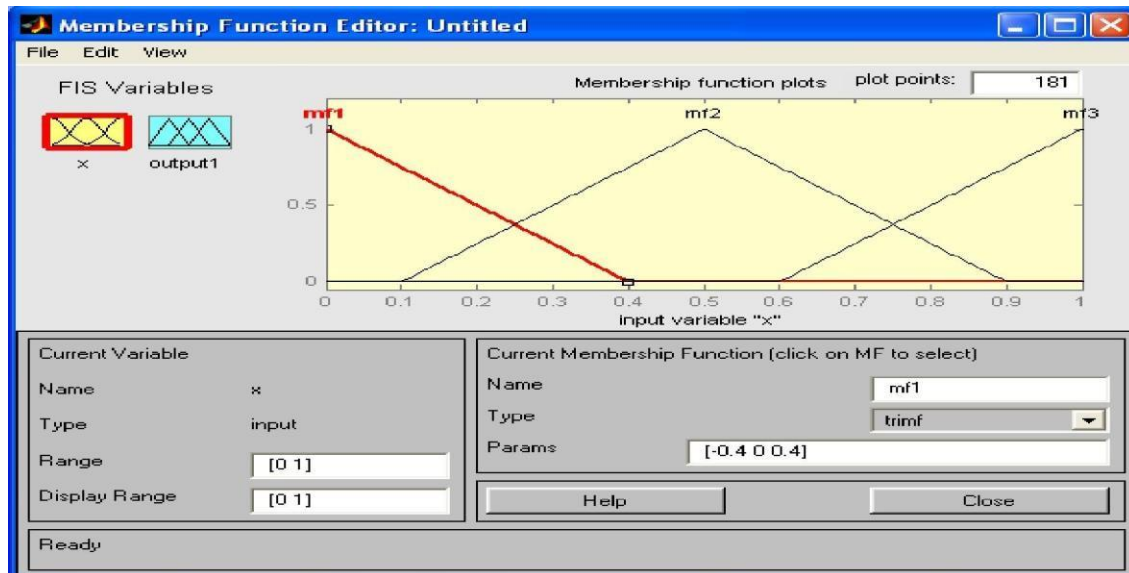


Figure 3 – The Accessory Function Editor window

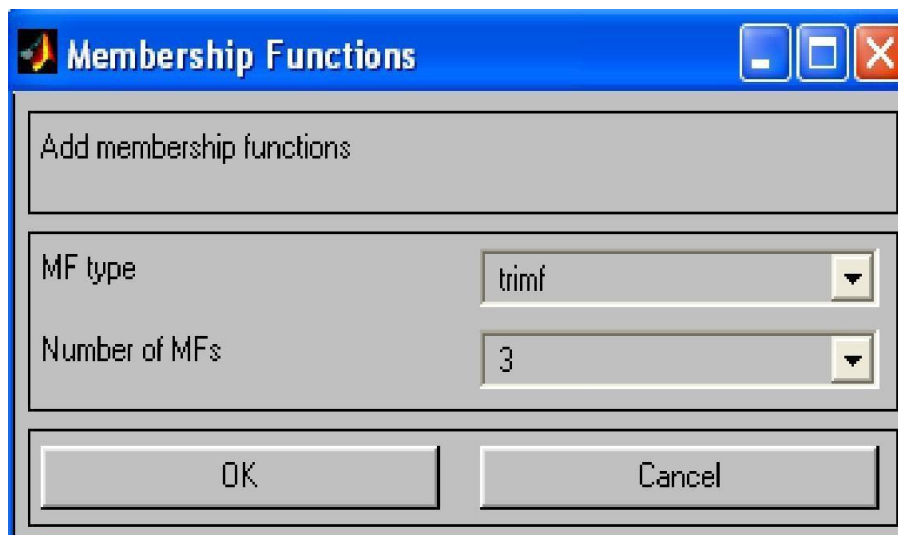


Figure 4 – Dialog box for setting the type and number of membership functions

4 In the Range field, you need to set the range of change x from -1 to $+1$, i.e. the range corresponding to Table 1. Then left-click somewhere in the editor field (or press the Enter key). After that, the range in the Display Range field will change accordingly.

5 Let's turn to the graphs of the given membership functions. To successfully solve the problem, it is necessary that the ordinates of the maximums of these functions coincide with the specified values of the argument x . For the left, central and right functions, this condition is met, but the other two must be "moved" along the axis of the abscissa. "Moving" is done very simply: move the cursor to the desired curve and click with the left mouse button. The curve is selected by turning red, after which it can be moved in the right direction with the help of a cursor (a more precise setting) can be made by changing the numerical values in the Params field – in this case, each belonging function corresponds to two parameters, the first determining the range of the curve, and the second determining the position of its center). For the selected curve, you can also change the name in the Name field (by pressing the Enter key to enter each name).

Next, you need to give all five curves new names, for example:

- leftmost – b_n ,
- next – n ,
- central – z ,
- Next to it on the right is P , the leftmost is Br .

Click the Close button and exit the Belong Function Editor, returning to the FIS Editor window.

6 Left-click once on the blue quad (block) entitled output1 (output 1). In the Name box, replace the name output1 with y (as in step 2).

7 Double-click on the selected block and go to the program – the owner function re-editor. In the Edit menu item, select the Add option

MFs. The view dialog box that appears now allows you to specify only linear or constant as membership functions, depending on which Sugeno algorithm (1st or 0th order) you want to choose. In the problem under consideration, you need to select constant membership functions with a total number of 4 (based on the number of different values y in Table 1). You need to confirm the entered data by clicking OK, after which you will return to the Ownership Function Editor window.

8 In the Range, you do not need to change the default change of $[0, 1]$. You need to change the names of the membership functions (their graphs are not given when using the Sugeno algorithm for output variables), for example, setting them as the corresponding numeric values y .

9 Double-click with the left mouse button on the middle (white) block, and a window of another program will open - the Rule Editor. It is necessary to enter the appropriate rules. When entering each rule, it is necessary to indicate the correspondence between each function of the ownership of the argument x and the numerical value of y .

The same should be done for other values, as a result, a set of five rules will be formed. The construction of the system is completed at this point.

Task

For the resulting variant (see practical task No1, pp. 9-10) and the developed PrO graph, form linguistic variables and obtain belonging functions for them.

Control questions and tasks

- 1 What is a linguistic variable?
- 2 What is fuzzification?
- 3 What is activation?
- 4 How can a clear quantitative value of the result be obtained?

Practical Lesson No3. Building Models in Artificial Intelligence Systems (Prolog Declarative Language)

The purpose of the lesson: to learn the visual development environment Visual Prolog. Create a project and run it.

Brief theoretical information

The prologue was developed at the University of Marseille in France by Alain Colmeroy and other members of the "artificial intelligence group" in 1973. This team of enthusiasts developed a program designed to prove theorems. It was written in Fortran, and was used in the construction of natural language processing systems. The program worked quite slowly and was called Prolog (from Programmation en Logique in French). The program served as a prototype for Prolog.

The Prolog logical programming language is declarative, which means that a program written in it looks like a set of logical descriptions that define the purpose for which it was written.

Prolog is based on a branch of mathematical logic called *predicate calculus*. Its basis is the procedure for proving theorems by *the method of solution for Horn disjuncts*.

Predicate logic is the simplest way to explain how thinking "works."

In fact, Prolog is not so much a programming language as a tool for describing data and the logic of their processing. Since the language is not algorithmic, the Prolog program does not contain explicit algorithmic constructions such as conditional or cyclic operators, i.e. it is not a program in the classical sense. Written in Prolog, it is a model of a fragment of the subject domain, the problem of which the problem is aimed at.

The solution of the problem itself is written not in terms of a computer, as is customary in imperative languages, but in terms of the subject area of the problem to be solved, in fact, in the spirit of object-oriented programming.

Prolog includes an inference mechanism that is based on sample matching, using query responses, it retrieves stored (known) information [4].

Methodical instructions for work

In order to start Visual Prolog, you need to perform the following actions: Start | Programs | Visual Prolog 5.2 | Visual Prolog 32. This opens the main window, which is called the Task window (Figure 5).

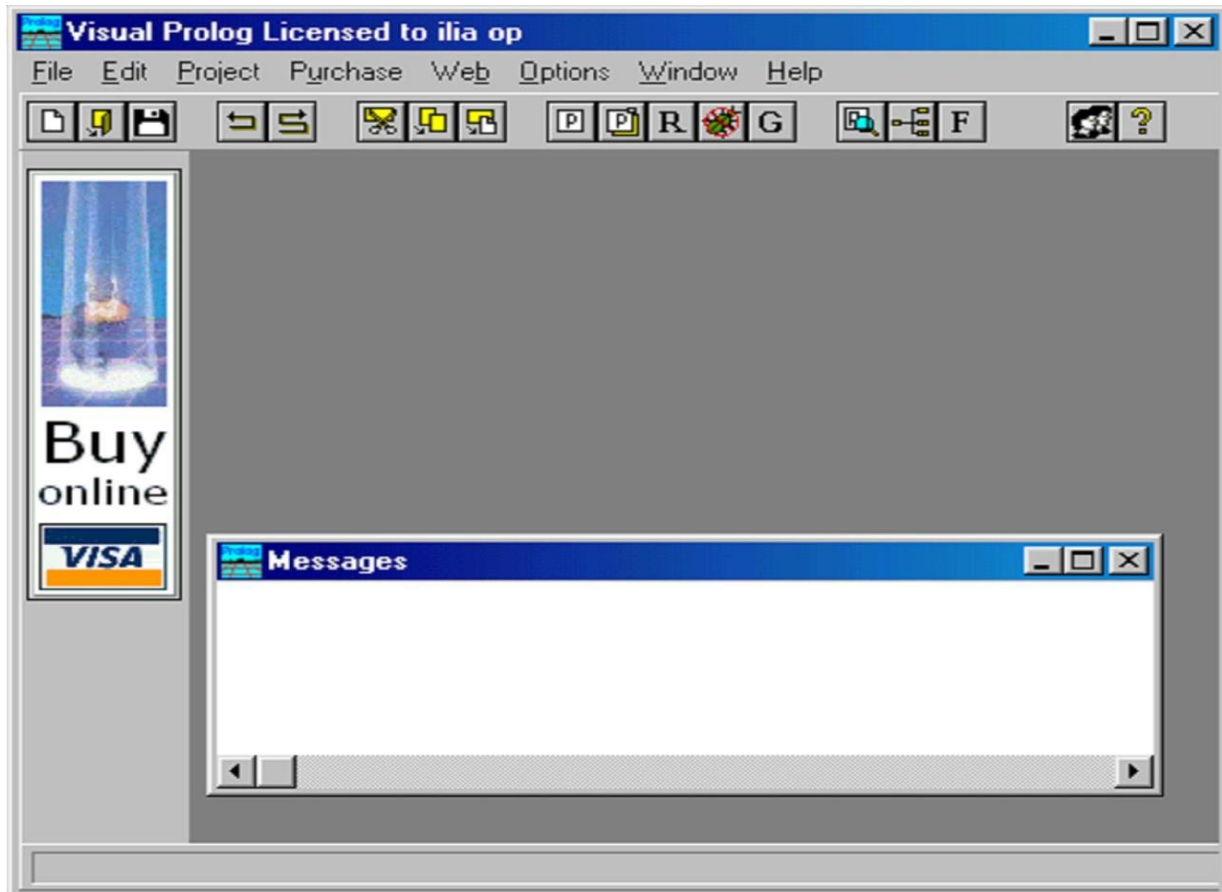


Figure 5 – Task window

Usually, the File, Edit, Project, Options, Help, and Window menus are available in this window, but when you activate some other windows, additional options may appear in the menu.

Frequently used menu commands can also be executed using the toolbar buttons (Figure 6).



Figure 6 – Toolbar

Each of the icons on the toolbar performs the same function as the corresponding menu command.

At the bottom of the Task window, there is a hint bar. It is divided into two parts.

The left margin is used to display contextual information, such as toolbar tooltips, information about the current control in the dialog editor, and so on.

The rightmost field is used by the make facility to display the current resource's generation/compilation/linking states.

To create a project, you need to define some (undefined) options of the Visual Prolog compiler. To do this, you need to do the following:

- 1 Start the Visual Prolog Visual Development Environment. The first time you start VDE(), the project will not load and the window shown in Figure 4 will be displayed. You will also be informed that an initialization file for Visual Prolog VDE has been created by default.

- 2 Create a new project.

Select the Project | New Project command by activating the Positive Expert dialog box.

3 Define the base directory and project name.

The name in the Project Name field should be defined as "Test". Next, you need to click inside the Name of. VPR File field. Also check the Multiprogrammer Mode checkbox and click inside the Name of.PRJ File field. The name of the project file Test.prj will appear (Figure 7).

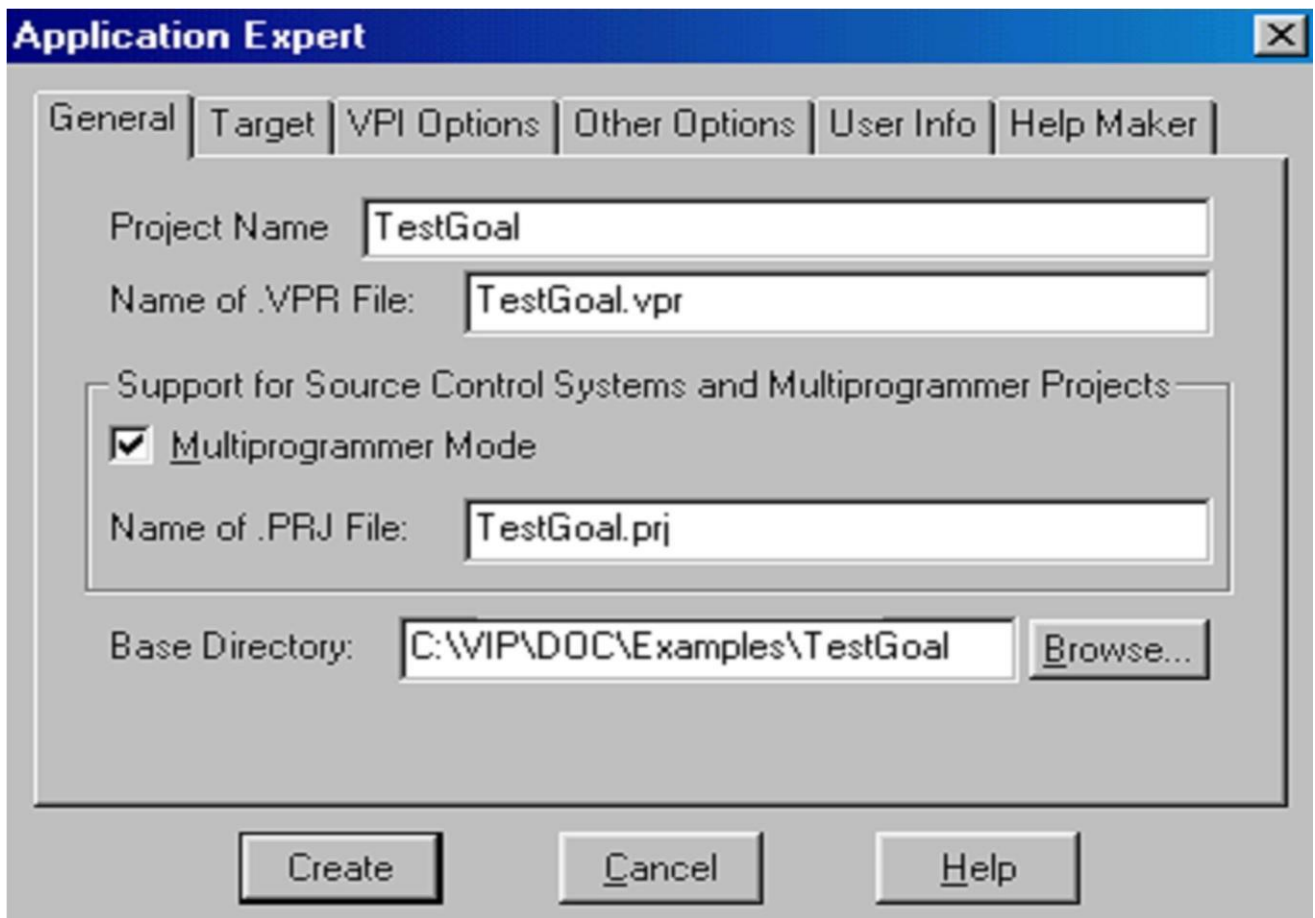


Figure 7 – General settings of the Application Expert dialog box

Define the goal of the project. On the Tooling tab, it is recommended to select the parameters marked in Figure 8.

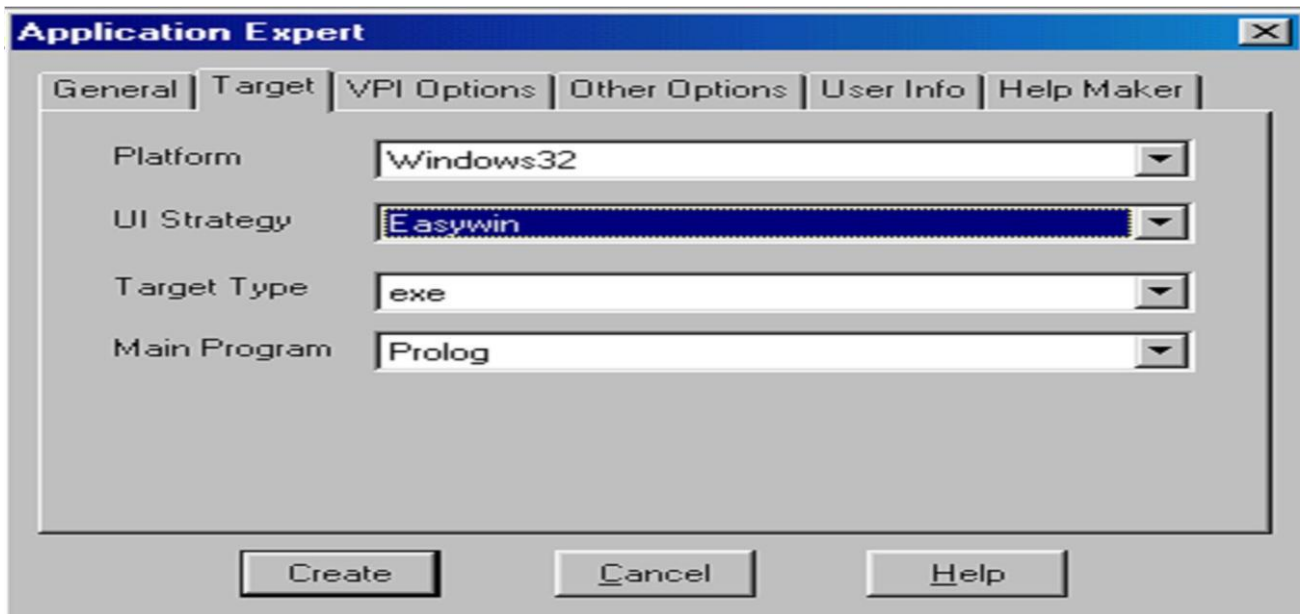


Figure 8 – Settings on the Target tab of the Application Expert dialog box

Click the Sgate button to create project files by default.

In the 19th century, the

- 4 Set the required compiler options for the created project.

To activate the Compiler Options dialog box, select the Options | project | Compiler Options command. Open the Warnings tab. Perform the following actions:

- set the Nondeterministic switch. This is necessary so that the Visual Prolog compiler assumes by default that all predicates defined by the user are non-deterministic (they can generate more than one solution);
- uncheck the Not Quoted Symbols, Strong Type Conversion Check and Check Type of Predicates checkboxes. This will suppress some possible compiler warnings;
- click OK to save the compiler option settings. As a result of these actions, the Computing Settings dialog box will look like the one shown in Figure 9.

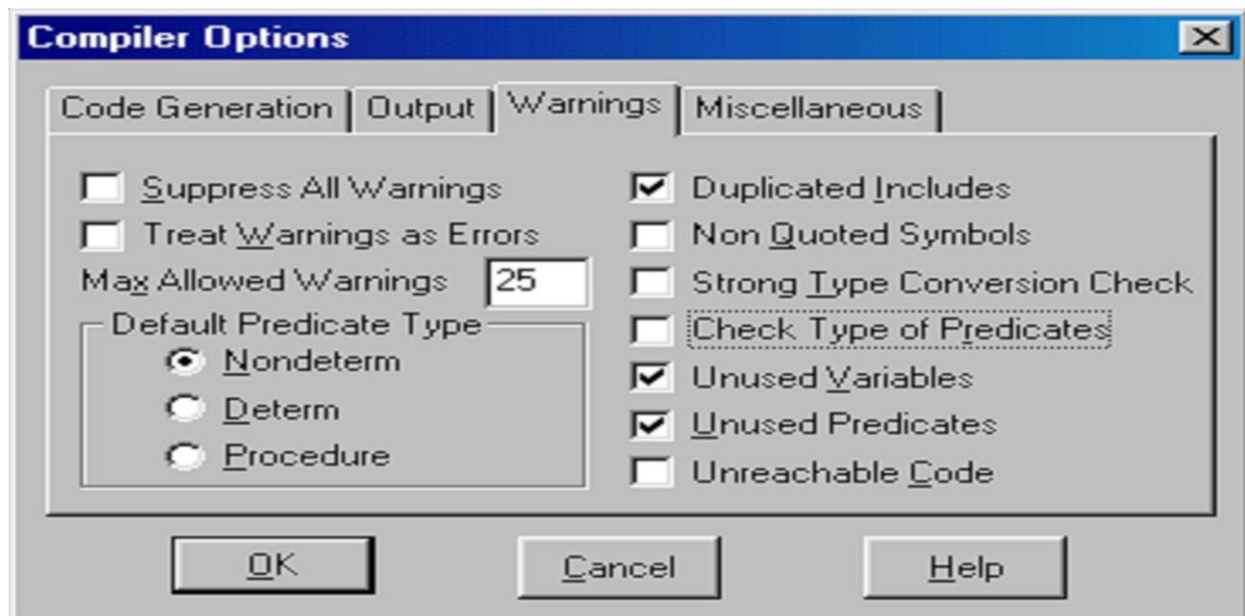


Figure 9 – Compiler Options Settings

To verify that the system is properly configured, the following steps should be performed:

- 1 In the project window, open the test.pro file (Figure 10)

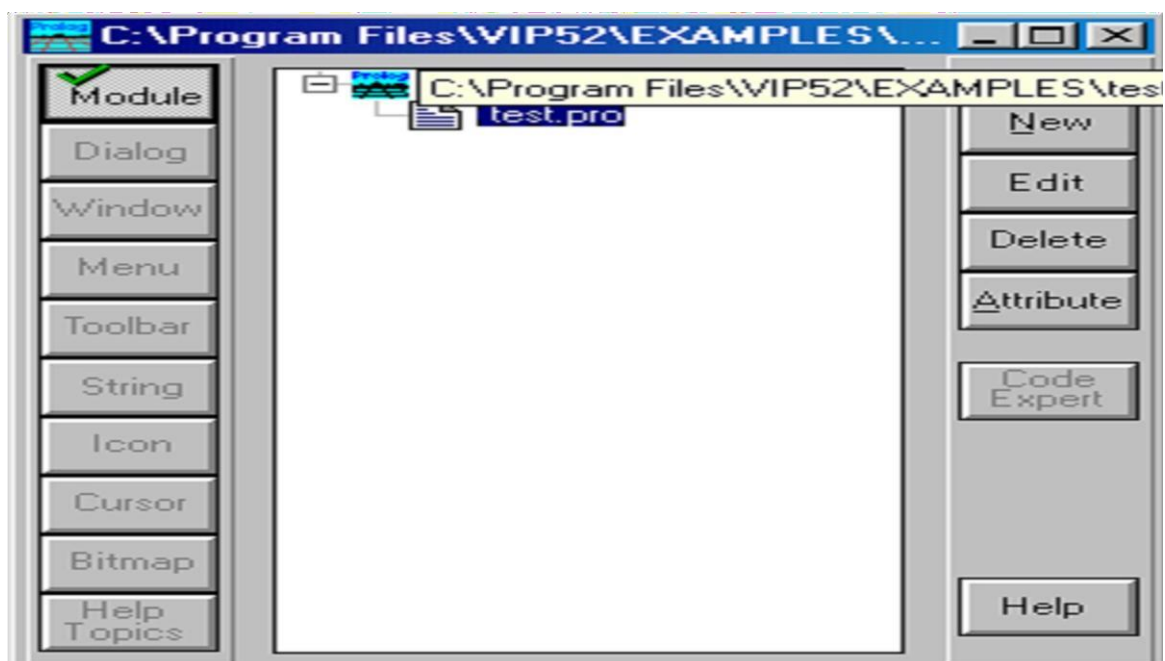


Figure 10 – Project window

- 2 In the GOAL section, type write ("Hello world"), nl .

3 Click a button (or a combination of keys) on the toolbar
<Ctrl>+<G>, or activate the Project | Test Goal). In Prolog terminology, this is called GOAL, and it is sufficient for the program to run. If the system is installed correctly, the monitor screen will look like Figure 11.

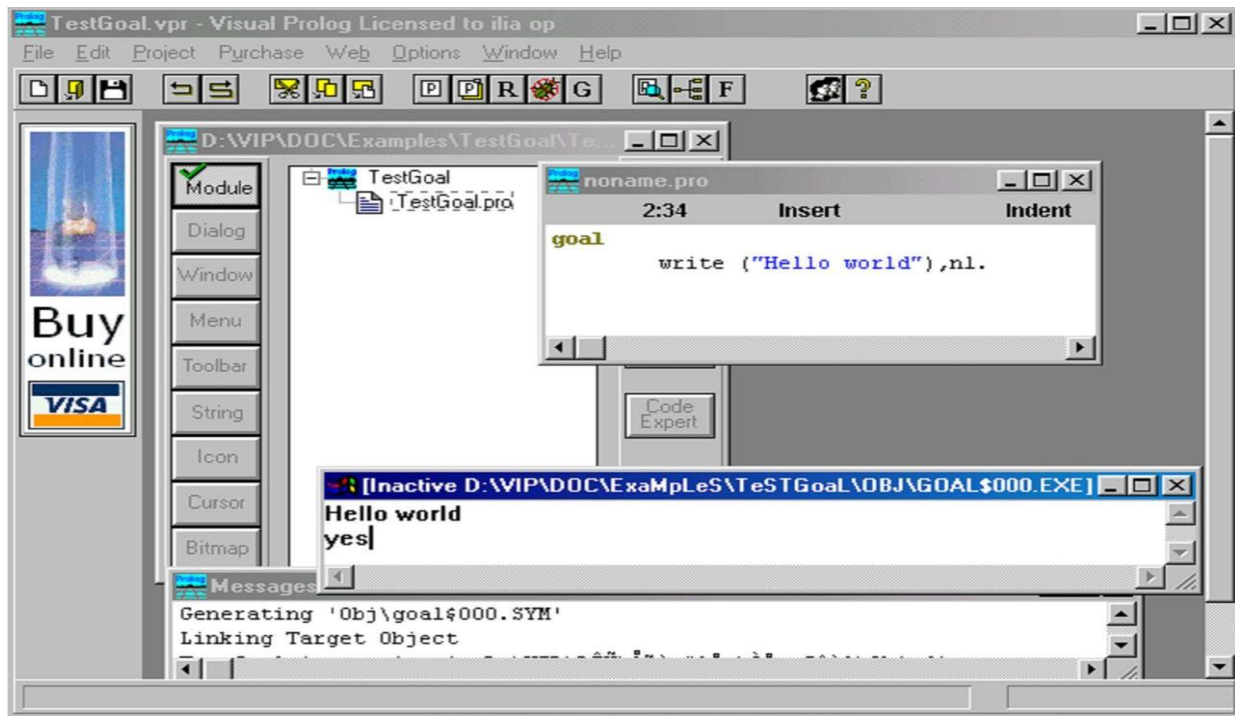


Figure 11 – Test program "Hello world"

The result of the program will be located at the top in a separate window, which must be closed before testing another GOAL.

The Visual Authoring Environment utility interprets GOAL as a special program that compiles, composes, is generated into an executable file, and Test Goal runs it. This utility internally extends the specified GOAL code so that the generated program finds all possible solutions and shows the values of all variables used. The Test Goal utility compiles this code using the compiler options specified for an open source project (the recommended compiler options for a TestGoal project were defined earlier).

The Test Goal utility compiles only the code that is defined in the active editor window (code in other open editors or project modules, if any, is ignored).

When linking an executable file, TestGoal uses the EASYWIN strategy. It is not possible to define any layout options for TestGoal, because any Make Settings specified for an open project are ignored. Therefore, TestGoal cannot use any global predicates defined in other modules.

The utility has a limit on the number of variables that can be used in GOAL. At the moment, there are 12 for a 32-bit visual development environment, but this number can be changed without additional notice.

If errors are made in the program and an attempt is made to compile it, the visual development environment will display the Errors (Warnings) window, which will contain a list of detected errors.

Double-clicking on one of these errors will take you to the location of the error in the source code. You can use the <F1> key to display the Visual Prolog online help system. When the help window opens, click the Search button, type the error number, and a help window will appear with more information about the error.

Build commands

1 Команда Project / Compile Module

This command (corresponding to the keyboard shortcut <Ctrl>+<F9>) attempts to compile the module that contains the file that is currently being edited. The execution of the command depends on the following file properties:

- if the file has a pro extension and is a module of the current project, then VDE tries to compile this file;
- If the file is not a module of the current project and its extension is pro, pre, inc, con or dom, then VDE tries to find the project module that includes this file and compile the first module found;

– in all other cases, VDE attempts to compile the module selected in the project window. VDE cannot compile a file that is not part of an open project. Instead, VDE will compile the module selected in the project window.

If no project is open in VDE, no files will be compiled. The Project | Compile Module menu command is blocked; the keyboard combination <Ctrl>+<F9> does not work. The only possible action is to run the Test Goal utility.

2 The Project/Build Team

If any resources have changed since the last time the project was built, code experts can update some sections in the source files before building.

This command (corresponding to the keyboard shortcut <Alt>+<F9>) builds the project by checking the timestamps of all the source files in the project, so if the source files (or the files that are included in them) are newer than the dependent OBJ files, the corresponding modules of the project will be recompiled.

The Build command also builds resource files and an online help file (if necessary). The project is then built to generate a target module (executable program or DLL).

3 Команда Project / Rebuild All

This command (corresponding to the keyboard shortcut <Ctrl>+<Alt>+<F9>) performs the same action as Project | Build, and all files will be regenerated or compiled and rebuilt, regardless of their timestamps.

4 Команда Project I Stop Building

This command (corresponding to the keyboard shortcut <Alt>+<F10>) is used to stop compiling/linking.

5 Project I Run Team

If necessary, this command (corresponding to the <F9> key) will execute the Project | Build action and then run the generated executable.

6 The Project I Link Only Team

This command (corresponding to the keyboard shortcut <Shift>+<F9>) is used to perform the link. In this case, Program Builder calls the linker and does not check whether any modules of the project need to be recompiled (or even compiled for the first time).

7 Команда Project / Test Goal

This command (corresponding to the keyboard shortcut <Ctrl>+<G>) is used to test simple Goals. The program is compiled and built in a special mode, and then the corresponding executable file is run.

The Test Goal utility searches for all solutions for the goal defined in the program. For each solution, Test Goal displays the values of all variables in the GOAL section and the number of solutions. This feature is a convenient way to check the local predicates in the module.

For example, the following goal: GOAL X = 2; X = 1, Y = X + 1 will lead to this result (Figure 12).



Figure 12 – Output of the target test mode

8 Команда Resource I Build Resource Only

When the project window is activated, the Resource command appears on the Project menu. With Selection of this Item (or нажатии Combinations keys <Alt>+<F8>) the selected files with the extensions rc and res and the required constant files are generated.

Example

Upload the program to the Visual Prolog visual development environment and run it with the Test Goal utility.

```
predicates
likes(symbol,symbol)
clauses
likes(ellen,tennis).
likes(john,football).
likes(tom,baseball).
likes(eric,swimming).
likes(mark,tennis) .
likes(bill,Activity):-likes (tom, Activity) .
goal
likes(bill, baseball).
```

The Test Goal utility will respond in the application window: yes

Try the following query in the GOAL section: likes(bill, tennis). Uti-lita Test Goal will answer: no.

The Project | Debug command starts the debugging process. You can also start the debugger by pressing CTRL+SHIFT+F9.

Using the View dialog, you can open additional information windows that display various states of the environment and variables in debug mode:

View → Call Stack (Открывает информационное окно стека вызова) View → Local Variables Use the following commands to perform the debugging steps:

Run → Trace Intro [F7]

Run → Step Over [F8]

Run → Run to Cursor [F4]

Задание

Prepare a program in the PROLOG language for the resulting version (see practical task No1, pages 9-10). Run the program in the Visual Prolog environment in debug mode.

Control questions and tasks

- 1 What is predicate logic?
- 2 Describe the Prolog language.
- 3 What are the features of recording the solution of a problem in the Prolog language?
- 4 Describe the inference mechanism implemented in the Prolog language.

Practical Lesson №4. Products in Artificial Intelligence Systems

The purpose of the lesson: to study the mechanism of hatching in production systems.

Brief theoretical information

The main feature of intelligent systems is that they are based on knowledge, or rather, on some representation of it. Knowledge is understood as information stored (with the help of a computer), formalized in accordance with certain rules, which a computer can use in logical inference according to certain algorithms. The most fundamental and important problem is the description of the semantic content of problems of the widest range, i.e. such a form of knowledge description should be used that would guarantee the correct processing of its content according to certain formal rules. This problem is called the problem of knowledge representation [5].

Production rules are the simplest way to represent knowledge. It is based on the representation of knowledge in the form of rules structured in accordance with the "IF-THEN" pattern. The part of the "IF" rule is called the premise, and the "THEN" part is called the conclusion or action. The rule is generally written as follows:

IF $A_1, A_2, \dots, A_n$, THEN B .

Such a notation means that "if all conditions from A_1 to A_n are true, then B is also true" or "when all conditions from A_1 to A_n are satisfied, then action B should be performed".

The variables x , y , and r show that the rule contains some universal, general knowledge abstracted from the specific values of the variables. The same variable used in the inference and in different premises can produce different specific values.

The knowledge presented in the intellectual system forms the knowledge base. The intellectual system also includes the mechanism of conclusions, which makes it possible to obtain new knowledge on the basis of the knowledge available in the knowledge base.

A formalized procedure that uses matching (in which it is established whether two forms of representation coincide with each other, including the substitution of possible values of variables), a search in the knowledge base, returning to the initial state in case of an unsuccessful attempt to solve, is a mechanism of conclusions.

The simplicity and clarity of knowledge representation with the help of products has led to its use in many systems, which are called production systems [6].

Methodical instructions for work

A production system consists of three main components. The first of them is a set of rules used as a knowledge base, sometimes it is also called a rule base. The second component is a fact base or working memory - memory for temporary storage, which stores premises related to specific problems of the subject area, and the results of conclusions obtained on their basis. The third component implements a logical inference mechanism that processes rules in accordance with the content of working memory; another name for this component is a logical inference machine.

There is another method of inference, called inverse inference. When using it, the system begins by stating what needs to be proved, for example, assuming that the situation is present, and follows only those rules that are relevant to proving the proposition.

If there is a set of applicable rules at each stage of logical inference, then this set is called a conflict set, and the selection of one of them is called conflict resolution. In order to increase the efficiency of a production system, it is necessary to solve the problem of controlling the sequence of rules or controlling output.

Example

Rule Base:

– if B and C and D, then

A; – if F and G,

then B;

– if H and D, then

E; – if E, then

A.

– Working memory G, H, D, F.

Let's consider how the rules "work". The system is built in such a way that a once-selected rule from the rule database will be executed only once. It seems to "burn out". The rule "F G B" burns out first $\rightarrow$, and the working[^] memory is supplemented with the fact B, then we process the rule[^] "H D $\rightarrow$ E", the fact E will be placed in the working memory "B[^] C[^] D[^] A $\rightarrow$ ", and, as a result, situation A is derived, and it is entered into the database and the last rule is "E $\rightarrow$ A".

After that, the system will stop, since there are more rules for execution It will not be.

This method of obtaining new facts is called direct inference or inference from data to the goal, because the search for new information here takes place in the direction of the arrow separating the left and right parts of the rules. The conclusion in this case is carried out in four stages.

If it is a question of inferring a single concrete fact, then direct inference may not be effective. In such a situation, from a cost point of view, it may be more appropriate to use the reverse inference procedure. With this method of inference, the system system begins with output A.

Step 1. The system receives an instruction about the need to output A. First, the presence of A in the fact base is checked and, after a negative answer, the search for rules ending with A, that is, the available A to the right of the arrow. The system finds two rules "B C D $\rightarrow$ A", "E $\rightarrow$ A" and a[^]c[^]c[^]e[^]p[^]t[^]s

a decision that in order to confirm A, it is necessary to establish the presence of either B and C and D or E.

Step 2. The system tries to establish B by first checking the basis of facts and then detecting the rules leading to B. The rule "F G $\rightarrow$ B" is detected. The system decides that in order to conclude that B is present, it needs to establish the presence of F and G.

Step 3. The system detects F and G in the fact base, confirming the possibility of inference B.

Step 4. The system tries to establish E by first checking the basis of facts and then detecting the rules leading to E. One such rule is detected: "H D $\rightarrow$ E". The system decides that in order to infer the presence of E, it needs to establish the presence of H and D.

Step 5. The system executes the rule "H D $\rightarrow$ E" and establishes fact E. Step 6. The system executes the rule "F G $\rightarrow$ B" and establishes fact B. Step 7. The system executes rule "E $\rightarrow$ A" and establishes fact A.

Step 8. The system executes the rule "B C D $\rightarrow$ A" and establishes the fact

A.

The chain of logical inference formed in this case does not differ in is derived from the procedure of direct inference. The difference between the two approaches is related to the method by which the search for facts and rules is carried out.

If there are a set of applicable rules at each stage of inference, then this set is called a conflict set, and the selection of one of them is called conflict resolution. In order to increase the efficiency of a production system, it is necessary to solve the problem of controlling the sequence of rules or controlling the output. Below is an example of a previously defined production system with five rules for inference of fact A (Figure 13).

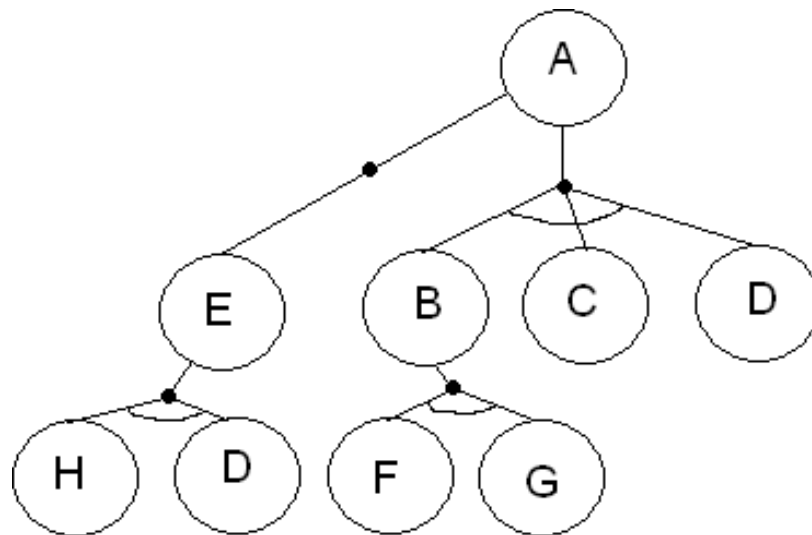


Figure 13 – Rules for inferring fact A

Fact A can be reached on the basis of the working memory of H, D and F, G and B, C, D, this is the conflict, that is, at this stage of logical inference there are many applicable rules, if you choose one of them, then the conflict will be resolved.

From the graph it can be seen that for a logically minimal derivation of fact A, a set of rules is required in the following order:

– if H D, then E $H \wedge D \rightarrow E$ –

if E, then A $E \rightarrow A$ **Task**

The database of rules and working memory in the production system has the content specified in the variants (see practical task No1, pp. 9-10). Graphically illustrate the mechanism of direct and reverse logical inference of fact A. Pay attention to changes in the contents of working memory in the process of inference. Arrange the rules of inference. Consider possible conflicts in direct and reverse inference.

Control questions and tasks

- 1 What is the main feature of intelligent systems?
- 2 What is the essence of the problem of knowledge representation?

3 What are production rules?

4 Describe the output mechanism in the product system.

Practical Lesson №5. Frame Knowledge Representation Models

Purpose of the lesson: to study the representation of static knowledge based on frames.

Brief theoretical information

The frame model is based on the concept of Marvin Minsky, a professor at the Massachusetts Institute of Technology, the founder of the Artificial Intelligence Laboratory, and the author of a number of fundamental works. The frame model is a systematized psychological model of human memory and consciousness.

A frame is a data structure for representing a conceptual object. Information related to a frame is contained in its constituent slots.

A slot can be a terminal (hierarchy sheet) or a lower-level frame.

Each frame consists of an arbitrary number of slots, with a few of them usually defined by the system itself to perform specific functions, and the rest defined by the user.

Frames form a hierarchy. Hierarchy in frame models generates a single multi-level structure that describes either an object, if the slots describe only the properties of the object, or a situation or process, if the individual slots are the names of procedures attached to the frame and called when it is actualized.

Formally, a frame is a type of data of the form:

$$F = \langle N, S_1, S_2, S_3 \rangle$$

N – Object name

S_1 – a set of slots containing facts that determine the declarative semantics of the frame;

S_2 – many slots that provide connections to other frames (causal, semantic, etc.);

S_3 – multiple slots that provide transformations that define the procedural semantics of the frame.

Frames are divided into:

- frame-instance – a specific implementation of a frame that describes the current state in the subject area;
- frame reference – a template for describing objects or permissible situations of the subject area;
- Frame-class—A top-level frame for representing a collection of sample frames.

The composition of frames and slots in each specific frame model may be different, but within the framework of one system, a single representation is advisable to eliminate unnecessary complication.

Objects of different types or objects corresponding to the concept of "multiplicity of worlds", which consists, for example, in the fact that a horse is wingless for one (real) world and at the same time winged (Pegasus in the mythical world) for another, can be described by different frames.

In general, the frame model allows the representation of all the properties of declarative and procedural knowledge. The depth of nesting of slots in the frame (the number of levels) depends on the subject area and the language that implements the model [7].

Methodical instructions for work

A frame has a certain internal structure, consisting of many elements called slots, which are also named. Slots are followed by spaces, in which data representing the current values of the slots are placed. Each slot, in turn, is represented by a certain data structure. A specific information related to the object described by this frame is substituted into the slot value.

The structure of the frame can be represented as follows:

FRAME NAME:

(1st slot name: 1st slot value), (2nd slot

name: 2nd slot value),

- - - -

(Nth slot name: Nth slot value).

The frame model is quite universal, since it allows you to display the entire variety of knowledge about the world:

- through frames-structures, to designate objects and concepts (loan, pledge, bill);
- through role frames (manager, cashier, client);
- through frames-scenarios (bankruptcy, meeting of shareholders, celebrating name days);
- through frame-situations (alarm, emergency, operating mode of the device) etc.

The value of a slot can be almost anything: numbers, formulas, natural language texts or programs, output rules, or references to other slots in a given frame or other frames. A slot value can be a set of lower-level slots, which allows you to implement the "matryoshka principle" in frame representations. Links between frames are defined by the values of a special slot called "Link".

In general, the data structure of a frame can contain a wider set of information.

Example

Let's consider an example of a frame system describing a certain audit (Figure 14).

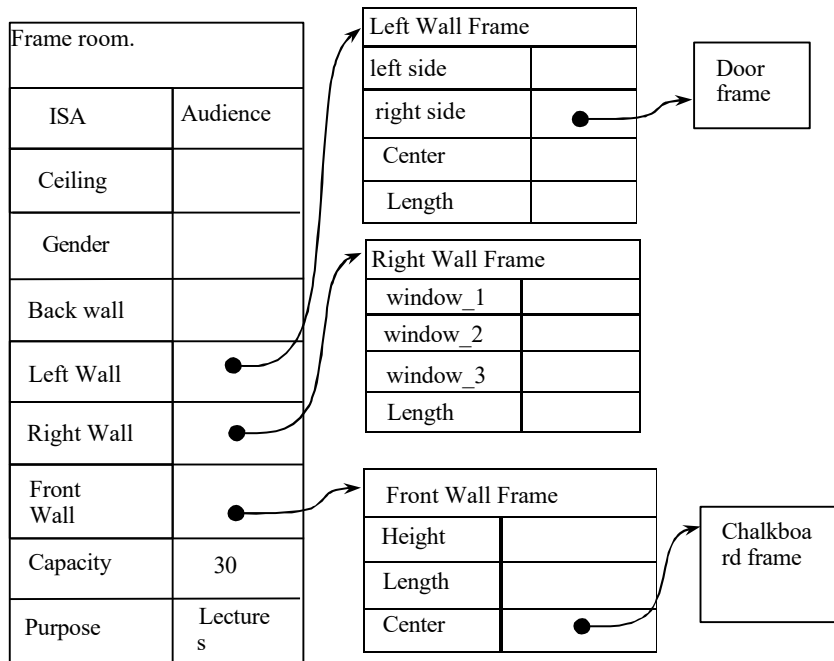


Figure 14 – Audience frame

Task

Build frames for the following concepts:

- 1 Option 1. The concept of "student".
- 2 Option 2. The concept of "professor".
- 3 Option 3. The concept of "wardrobe".
- 4 Option 4. The concept of "computer".
- 5 Option 5. The concept of "table".
- 6 Option 6. The concept of "journal".
- 7 Option 7. The concept of "book".
- 8 Option 8. The concept of "child".

Control questions and tasks

- 1 What is a frame?
- 2 Name the composite elements of the frame.
- 3 Name the types of frames.
- 4 What determines the depth of nesting of slots in a frame?

Practical Lesson No6. Neural Networks in Artificial Intelligence Systems. Approximation of Neural Network Functions

The purpose of the lesson: to study the process of building and training a neural network for approximating a tabulated given function.

Brief theoretical information

Neural networks are one of the areas of research in the field of artificial intelligence, based on attempts to reproduce the human nervous system. Namely, the ability of the nervous system to learn and correct errors, which should make it possible to simulate, albeit rather roughly, the work of the human brain.

Neural networks are built of simple elements – neurons. The mathematical model of a neuron consists of three elements – synapses, an adder and a nonlinear transducer. In particular, sign, threshold, sigmoid and other functions are used as nonlinear transducers.

Synapses communicate between neurons. They multiply the input signal by the weight factor of the synapse, i.e. by the number that characterizes the strength of the connection [8].

Neurons united in a certain structure form a neural network. Some tasks can be solved by a neural network if it contains several hundred neurons. To solve complex problems, the neural network must contain tens of millions of neurons.

There are a large number of models of neural networks. Among them: multilayer perceptrons, Hopfield, Kohonen, Kosko, Grossberg networks, co-gnitron, neocognitron, SMAK network and others. The characteristics of neurons have little effect on the properties of a neural network. The main factors that determine the properties of the network are its configuration or, as they say, the "topology" of the neural network, and the rules of learning.

From the point of view of topology, the following main types of neural network models are distinguished: multilayer, fully connected, with local connections.

Neural networks have many important properties, but the key one is the ability to learn. Learning a neural network primarily involves changing the "strength" of synaptic connections between neurons.

Today, neural networks are one of the priority areas of research in the field of artificial intelligence [9].

Methodical instructions for work

The selection of the network type is shown in Figure 15.

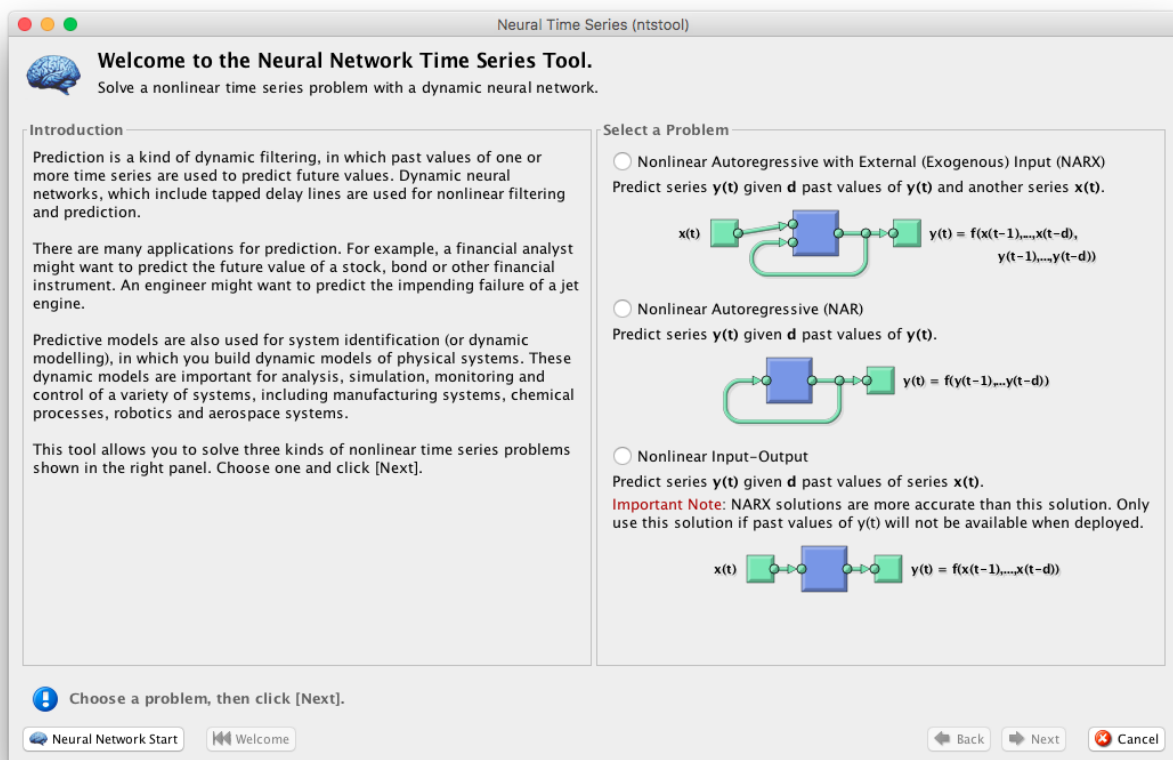


Figure 15 – ntstool welcome screen

On the right panel there is a panel for selecting the type of neural network:

- nonlinear autoregressive with an external input;
- nonlinear input-output (without feedbacks);
- nonlinear autoregressive (generator).

However, more network architectures can be implemented with code.

Example of creating neural networks using code: `inputDelays =`

`1:10;`

`feedbackDelays = 1:10;`

`hiddenLayerSize = 20;`

Neural networks without feedback

`netx = narxnet(inputDelays,feedbackDelays,hiddenLayerSize);`

`net = narnet(feedbackDelays,hiddenLayerSize);`

Neural networks with feedback `netxc =`

`closeloop(netx);`

`netc = closeloop(net);`

To view the received networks:

`view(net).`

The selection of data for training and its preparation are displayed in the selection window as a parameter (Figure 16).

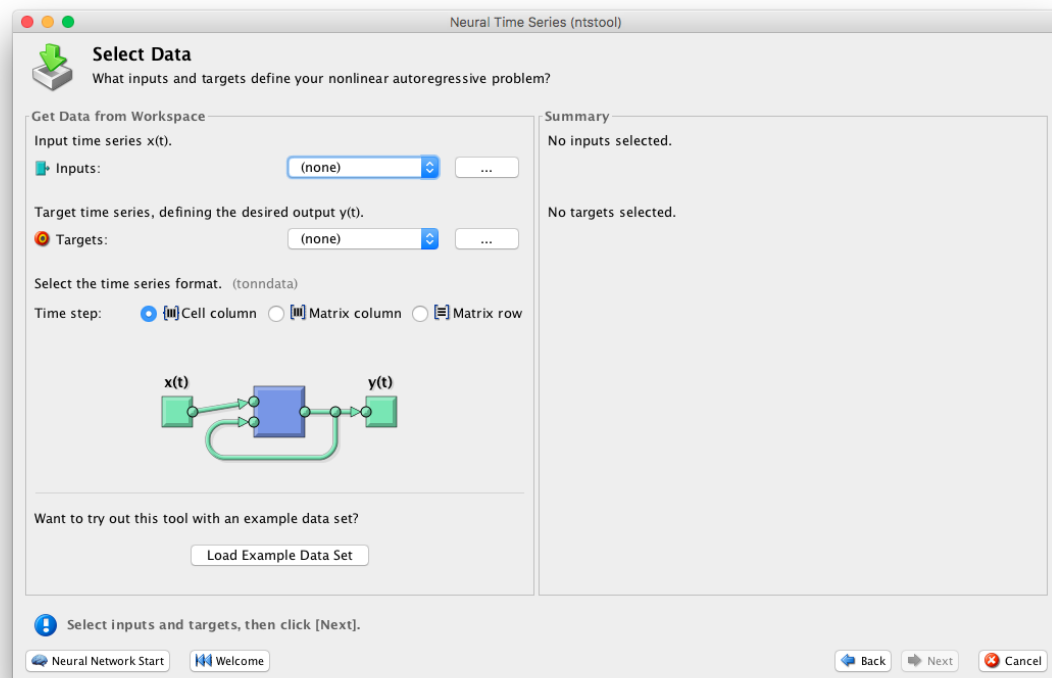


Figure 16 – Parameter Selection Window

In the *ntstool* utility, the selection of data for training is not a problem, so let's consider the choice of parameters in the code.

The *preparets* function prepares the time series for a portion of the network by changing the time to a minimum to "feel" the input data and internal variables. Using *preparets* allows you to leave the original time series unchanged by easily configuring them for a network with different delay sizes for both open and closed neural networks.

Command syntax:

`[Xs,Xi,Ai,Ts,EWs,shift] = preparets(net,Xnf,Tnf,Tf,EW).`

Аргументы:

Net – Neural network;

Xnf – Non-Returned Input; *Tnf* – Non-

Returned Output; *Tf* – Output;

EW – Error weight (Default = {1}); Return values:

Xs – Moved entrances;

Xi – Initial states of input delays; *Ai* – Initial states of hidden neurons; *Ts* – Displaced outputs;

EWs – Moved Error Weights;

Also, before using *preparets*, you may need to use the *tonndata* function, which converts the data to the standard type for neural networks.

The syntax of the *tonndata* command is:

`[y,wasMatrix] =`

`wavedata(x,columnSamples,cellTime)/` Аргументы:

x – Matrix or cell – an array of matrices;

columnSamples – True if the source data is represented as strings, false if as strings;

cellTime – True if the source data is represented as columns of matrix cells, false if as a matrix.

Return values:

y – Source data converted to standard view; *wasMatrix* – True if the source data is represented as matrices. The choice of data destination is shown in Figure 17.

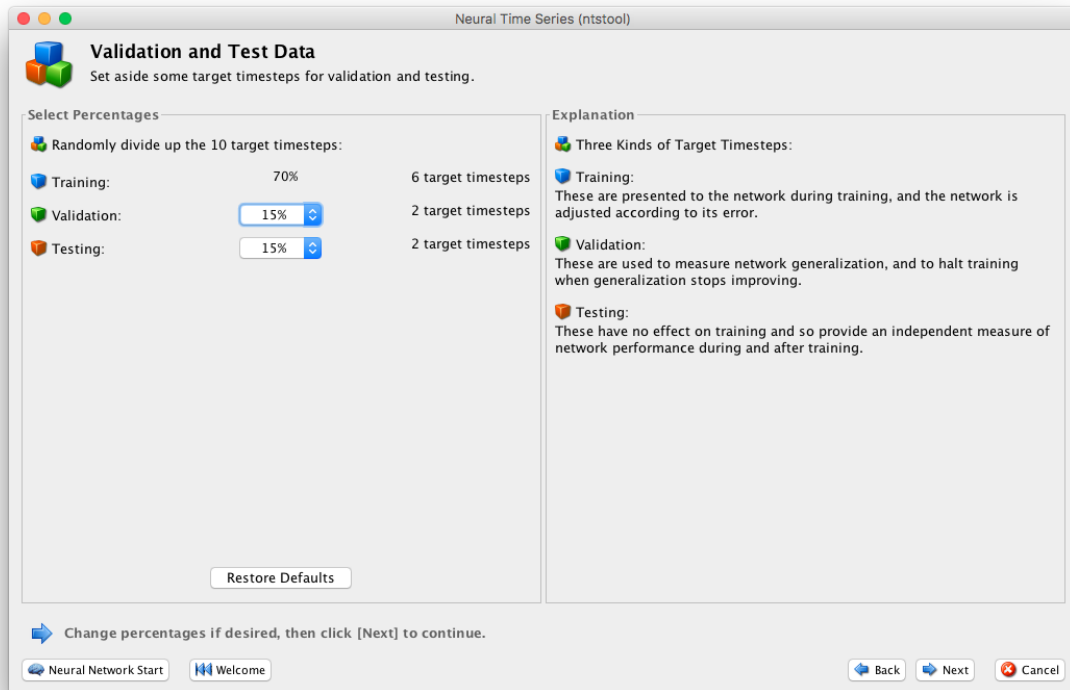


Figure 17 – Percentage selection window

Like the previous steps, working with *ntstool* is very simple and requires a simple selection of values. In the case of a neural network being designed using software code, the following commands must be applied:

```
net.divideParam.trainRatio = 85/100; %Тренировка;
```

```
net.divideParam.valRatio = 15/100; %Проверка;
```

```
net.divideParam.testRatio = 5/100; %Тестирование.
```

The last step before training the network in *ntstool* is to select the number of hidden neurons and the number of delay neurons (Figure 18).

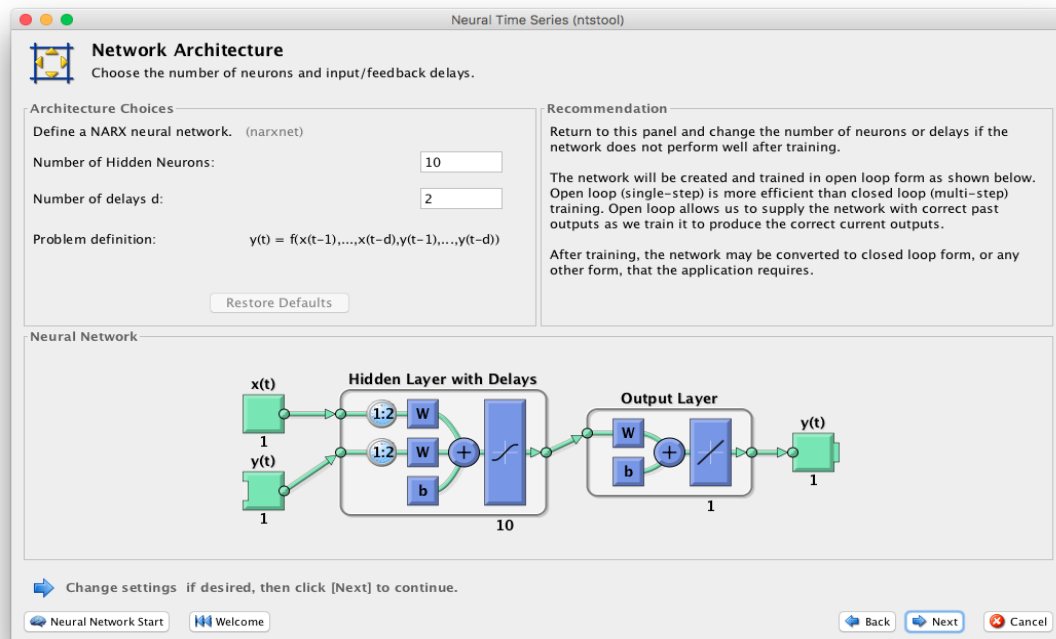


Figure 18 – Network Sizing Window

Note that the dimensions of the network are determined in the design code at the first step.

The training of a neural network is shown in Figure 19.

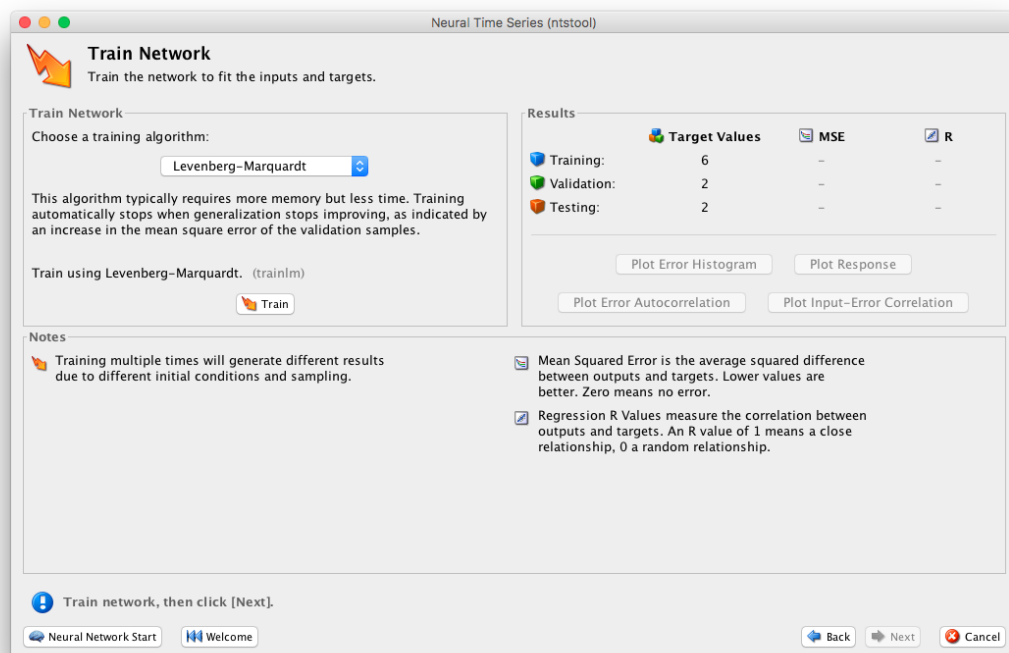


Figure 19 – Neural network training window

The code training operation looks like this:

```
[net,tr] = train(net,inputs,targets,inputStates,layerStates)/
```

In both cases, `nntraintool` is used to train the network, the window of which is shown in Figure 20.

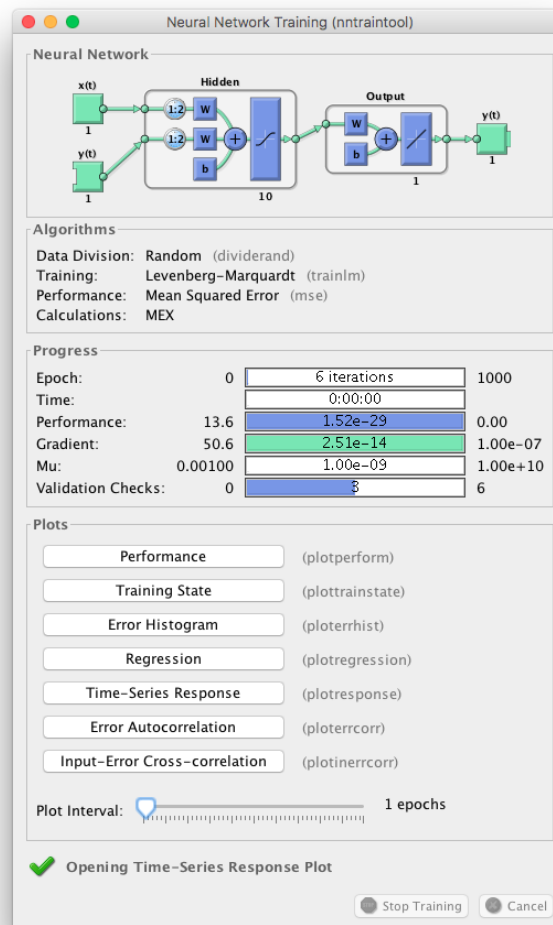


Figure 20 – nntraintool window

With the help of which it is possible to obtain various graphs that characterize the process and quality of network training. Graphs can also be called using the following commands:

```
outputs = net(inputs,inputStates,layerStates);
```

```
errors = gsubtract(targets,outputs);
```

```
performance = perform(net,targets,outputs)
```

For more information about the additional features of train, please use the official guide (there are many possibilities there).

To work with a trained network, you need to export the resulting neural network (Figure 21).

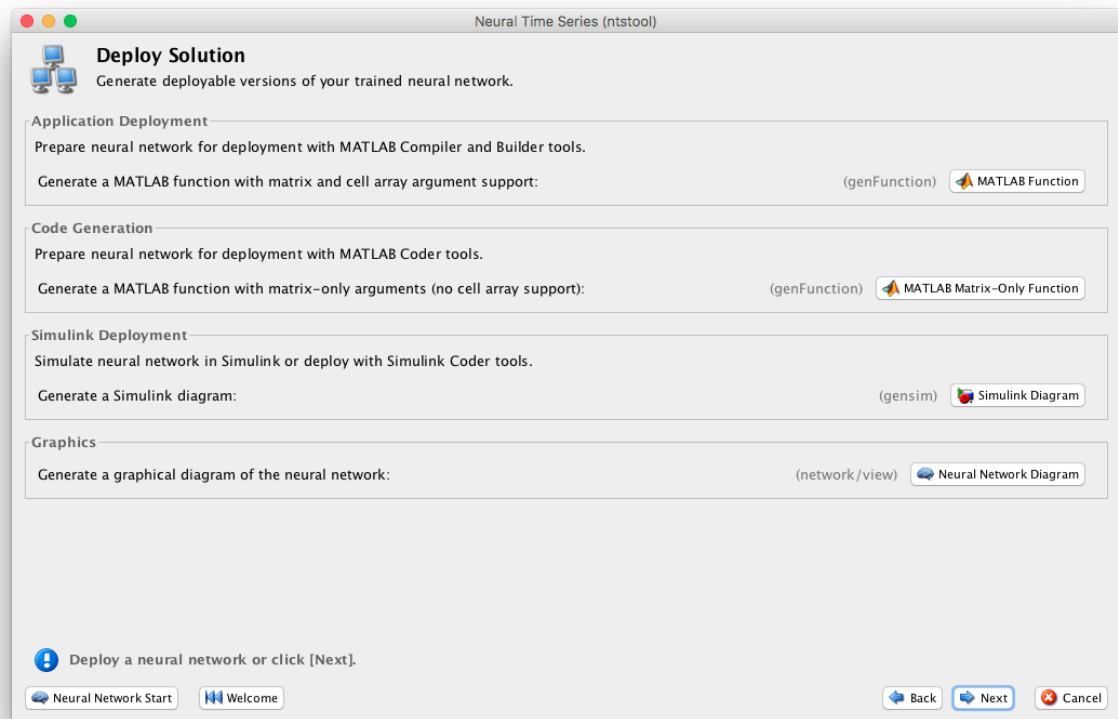


Figure 21 – Neural Network Export Window

The main question that arises at this step is whether it is possible to set a specific sampling time for a neural network. The answer is yes, you can, but not using ntstool, where there will always be a sampling step of 1 second.

To get a Simulink model after all operations on the neural network using the code, you need to call the function: `gensim(net,time)`, where *time* is the sampling time, and *net* is the neural network.

After the call, a model will be obtained in Simulink, with a specified sample rate.

Example

Here is an example of creating a neural network with the following data (Table 2).

Table 2 – Initial data

xi	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
f(xi)	2.86	2.21	2.96	3.27	3.58	3.76	3.93	3.67	3.90	3.64	4.09

The graph of the initial function is shown in Figure 22.

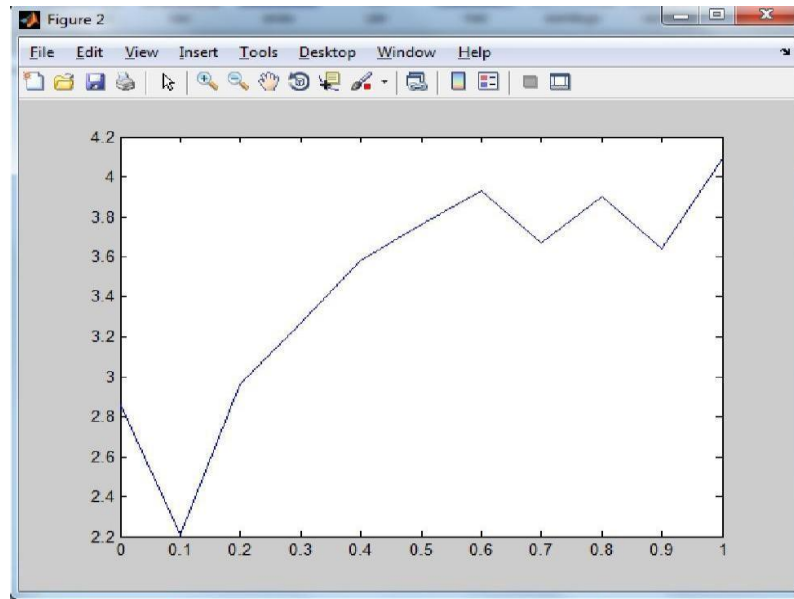


Figure 22 – Graph of the initial function

Creating and training a neural network:

```

x=[0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1];
f(x)=[2.86 2.21 2.96 3.27 3.58 3.76 3.93 3.67 3.90 3.64 4.09];
net=newff([0 3],[10,1],{'tansig','purelin'},'trainbfg');
net.trainParam.epochs=300;
net.trainParam.show=50;
net.trainParam.goal=1.37e-2;
[net,tr]=train(net,x,y);
an=sim(net,x);
plot(x,y,'+r',x,an,'-g');
hold on;
xx=[0.1850.86];

```

```
v=sim(net,xx);
```

```
plot(xx,v,'ob','MarkerSize',5,'LineWidth',2)
```

The process of network training is presented in Figure 23.

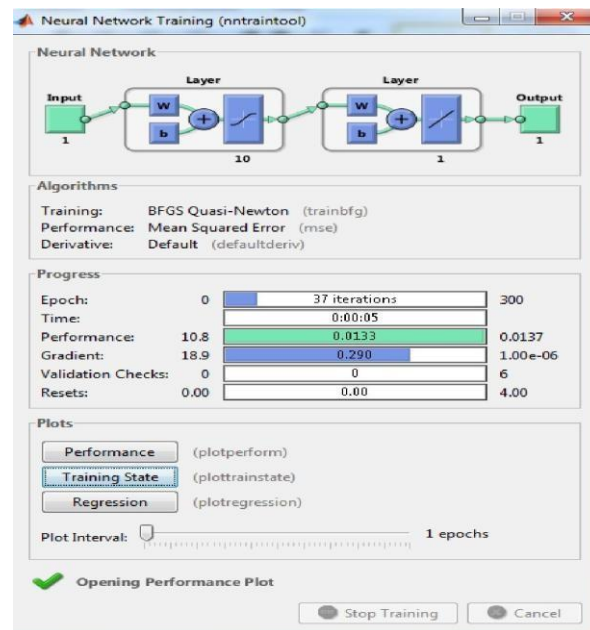


Figure 23 – Network training

In the process of training the network, a graph of the dependence of the accuracy of network training on the number of epochs (cycles) was obtained, and the calculation of the standard error of the network is 0.013305 for 37 cycles (Figure 24).

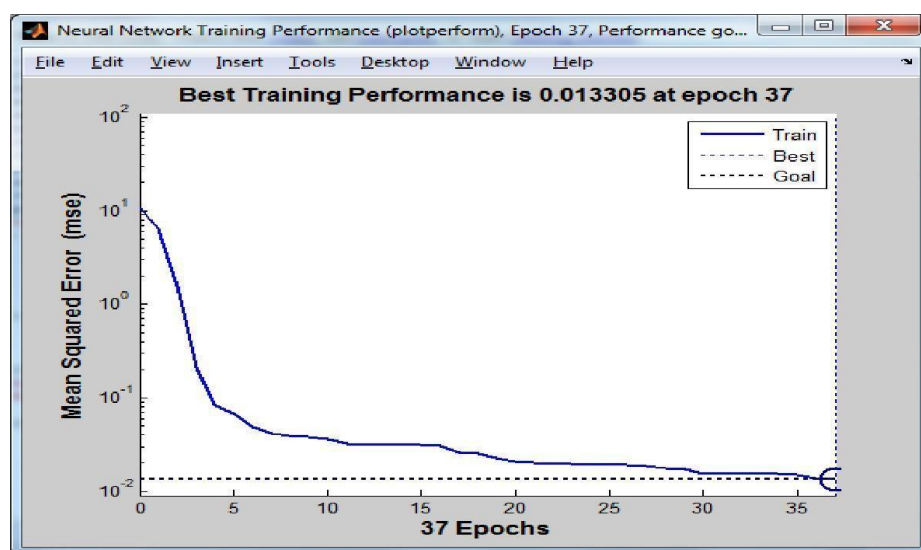


Figure 24 – Characteristics of the accuracy of learning depending on the number of epochs of learning

Next, you need to compare the graphs of the original function and the approximation (Figure 25).

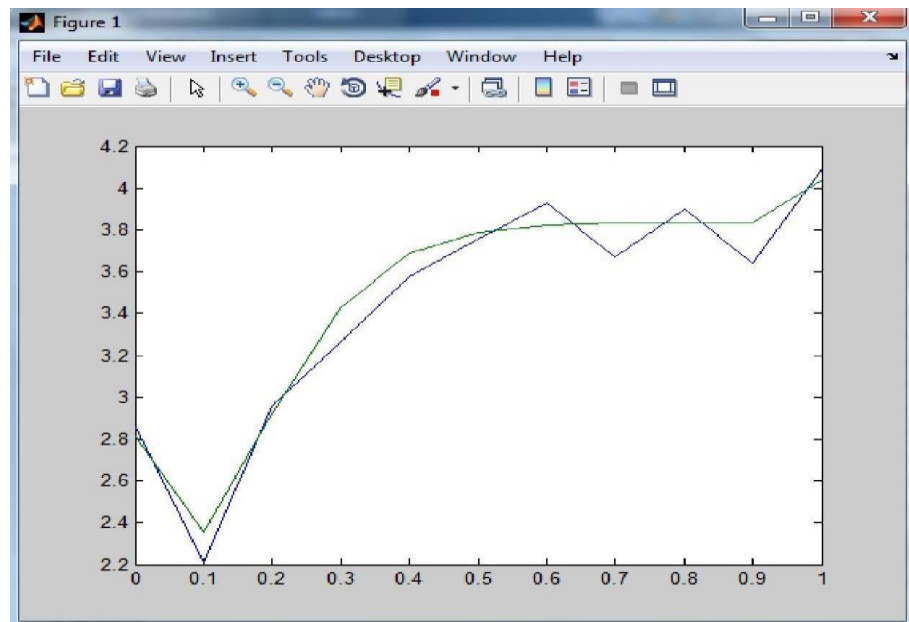


Figure 25 – Comparison of the plots of the original function and approximation

Approximate the incoming set of points using the least squares method. The result for the set is shown in Figure 26.

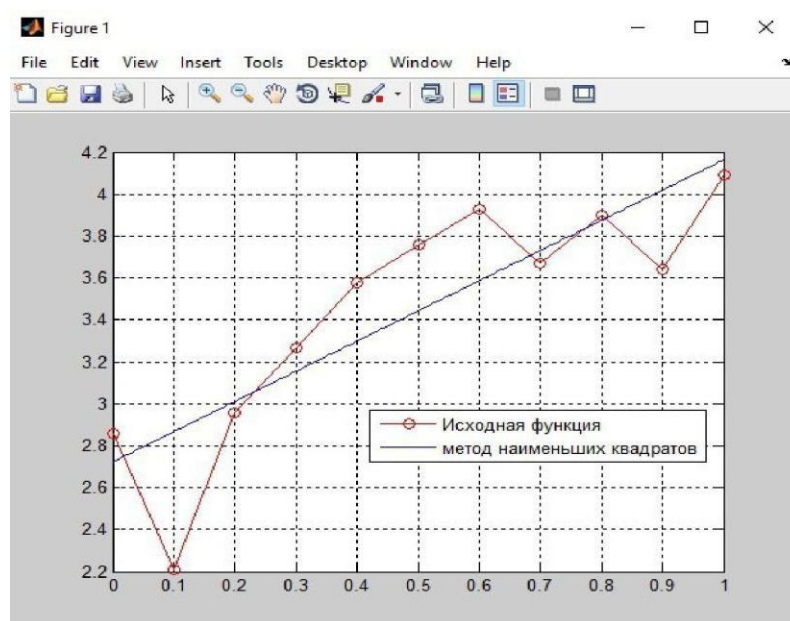


Figure 26 – Approximation of an incoming set of points using the naming square method

Based on the results of approximation of a given set of function values, it was found that the neural network approximates the initial value of the function much better than the method of least squares.

Task

In the MatLab environment, it is necessary to build and train a neural network for up-proximation of a tabulated function.

To do this, you need to create a table of experimental data: x_i – specified by the variant;

$f(x_i)$ is calculated using the following formula: $x_i = a + hi$, $i = 0.1, \dots, 10$, $h = (b - a)/10$ on the segment $[a, b]$.

x_i										
$f(x_i)$										

Variants of tasks:

- 1 Option 1: $x=[0 \ 0.1 \ 0.2 \ 0.3 \ 0.4 \ 0.5 \ 0.6 \ 0.7 \ 0.8 \ 0.9 \ 1]$, $a=2$; $b=12$.
- 2 Option 2: $x=[1 \ 1.1 \ 1.2 \ 1.3 \ 1.4 \ 1.5 \ 1.6 \ 1.7 \ 1.8 \ 1.9 \ 2]$, $a=7$; $b=27$.
- 3 Option 3: $x=[0 \ 0.2 \ 0.4 \ 0.6 \ 0.8 \ 1 \ 1.2 \ 1.4 \ 1.6 \ 1.8 \ 2]$, $a=2$; $b=12$.
- 4 Option 4: $x=[0 \ 0.1 \ 0.2 \ 0.3 \ 0.4 \ 0.5 \ 0.6 \ 0.7 \ 0.8 \ 0.9 \ 1]$, $a=4$; $b=14$.
- 5 Option 5: $x=[0 \ 0.5 \ 1 \ 1.5 \ 2 \ 2.5 \ 3 \ 3.5 \ 4 \ 4.5 \ 5]$, $a=6$; $b=21$.
- 6 Option 6: $x=[0 \ 0.1 \ 0.2 \ 0.3 \ 0.4 \ 0.5 \ 0.6 \ 0.7 \ 0.8 \ 0.9 \ 1]$, $a=8$; $b=18$.
- 7 Option 7: $x=[1 \ 1.1 \ 1.2 \ 1.3 \ 1.4 \ 1.5 \ 1.6 \ 1.7 \ 1.8 \ 1.9 \ 2]$, $a=9$; $b=29$.
- 8 Option 8: $x=[0 \ 0.1 \ 0.2 \ 0.3 \ 0.4 \ 0.5 \ 0.6 \ 0.7 \ 0.8 \ 0.9 \ 1]$, $a=1$; $b=16$.
- 9 Option 9: $x=[0 \ 0.2 \ 0.4 \ 0.6 \ 0.8 \ 1 \ 1.2 \ 1.4 \ 1.6 \ 1.8 \ 2]$, $a=6$; $b=21$.
- 10 Option 10: $x=[0 \ 0.5 \ 1 \ 1.5 \ 2 \ 2.5 \ 3 \ 3.5 \ 4 \ 4.5 \ 5]$, $a=9$; $b=29$.

Control questions and tasks

- 1 What are neural networks?
- 2 Name the constituent elements of neural networks.
- 3 What are the main factors that determine the properties of a neural network?
- 4 Describe a key property of neural networks.

References

- 1 Volkova V.N., Yuriev V.N., Shirokova S.V., Loginova A.V. Information Systems in Economics: Textbook for Academic Bachelor's Degree. Edited by V. N. Volkova, V.N. Yuriev. Moscow: Yurayt Publishing House, 2019. 402 p.
- 2 Abdikeev, N.M. Proektirovanie intellektual'nykh sistem v ekono-mike [Design of intelligent systems in economics] – Available at: www.dgybga.ru/.../proektirovanie-intellektualnih-sistem-v-ekonomike-uchebnikabdigeev...
- 3 Tarasyan, V.S. Fuzzy Logic Toolbox for Matlab Package. Textbook / V.S. Tarasyan. – Yekaterinburg : Ural State Transport University Publ., 2013. – 112 p.
- 4 Tsukanova, N.I. Theory and Practice of Logical Programming in the Visual Prolog 7 Language: Textbook for Higher Educational Institutions / N.I. Tsukanova, T.A. Dmitrieva. – Electron. dan. – Moscow: Hot Line-Telecom, 2013. – 232 p. – Access Mode: <http://e.lanbook.com/book/11847>.
- 5 Gromov, O.G. Ivanova, V.V. Alekseev et al. – Tambov: TSTU Publ., 2013. – 244 p.
- 6 Pavlov, S.N. Sistemy iskusstvennogo intellekta [Artificial intelligence systems]. Manual. In 2 parts / S.N. Pavlov. – Tomsk: El Content, 2011. –176 c
- 7 Smagin A.A., Lipatova S.V., Melnichenko A.S. Intellektual'nye informatsionnye sistemy: uchebnoye posobiye [Intelligent information systems: a textbook]. Ulyanovsk, Ulyanovsk State University Publ., 2010. 136 p.
- 8 Rutkovskaya, D. Neural Networks, Genetic Algorithms and Fuzzy Systems: Translated from Polish by I.D. Rudinskogo. EBS "Lan" [Elektronnyi resurs] / D. Rutkovskaya, M. Pilinskiy, L. Rutkovskiy. – Elektronny. dan. – M. : Goryachaya liniya-Telekom, 2013. – 384 p. – Rezhim dostupa: <http://e.lanbook.com/book/>.
- 9 Zamyatin N.V. Fuzzy Logic and Neural Networks: A Textbook / N.V. Zamyatin ; reviewers: I.A. Khodashinsky, S.N. Liventsov. – Tomsk: El Content, 2014. – 146 p.