МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания

по выполнению лабораторных работ
по дисциплине

«Системы искусственного интеллекта»

для направления подготовки 15.03.04 Автоматизация технологических процессов и производств
направленность (профиль) Информационно-управляющие системы

Невинномысск 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В результате изучения данной дисциплины студенты на основе приобретенных знаний, умений и навыков достигают освоения компетенций на определенном уровне согласно учебным планам и рабочим программам вышеперечисленных направлений:

- ✓ способность разрабатывать программное обеспечение, включая проектирование, отладку, проверку работоспособности и модификацию программного обеспечения;
- ✓ способность понимать принципы работы современных информационных технологий и использовать их для решения задач профессиональной деятельности.

Освоение студентами этих компетенций предполагает использование при их обучении специальных пакетов прикладных программ, в которых имеется широкий спектр методов математического анализа и моделирования. Одним из таких пакетов является математический пакет Scilab с открытой лицензией, в котором для программирования используется свой язык высокого уровня.

Пакет Scilab имеет обширный набор инструментальных средств как для выполнения математических расчетов и анализа, так и для моделирования различных типов систем. В Scilab моделируются

электрические и механические системы, а также системы управления и множество других систем. Кроме того, в качестве помощи для пользователей, данный пакет сопровождается обширной интерактивной документацией, содержащей подробные справки с примерами по всем применяемым модулям и функциям.

Состав пакета Scilab с открытой лицензией практически аналогичен составу коммерческого пакета Matlab, но для освоения пакета Scilab и использования интерактивной документации знание Matlab не обязательно. Свободно распространяемую последнюю версию пакета вместе с полной документацией в формате .pdf можно получить по официальному адресу URL: http://www.scilab.org/

В базовой комплектации Scilab – это система компьютерной математики, которая предназначена для выполнения различных инженерных и научных вычислений, например таких как:

- решение нелинейных уравнений и систем;
- решение задач линейной алгебры;
- решение задач оптимизации;
- дифференцирование и интегрирование;
- обработка экспериментальных данных;
- решение обыкновенных дифференциальных уравнений.

Кроме того, Scilab предоставляет широкие возможности по созданию и редактированию различных видов графиков как плоских, так и объемных. Несмотря на то, что Scilab содержит достаточное количество встроенных команд, операторов и функций, отличительная ее черта – это гибкость. Пользователь может создать любую новую команду или функцию, а затем использовать ее в своих программах.

К тому же имеется возможность расширять функционал пакета Scilab через загрузку и установку новых модулей Atoms. Для выполнения лабораторных работ по дисциплине «Системы искусственного интеллекта» требуются следующие дополнительные модули Atoms:

- ANN Toolbox (инструменты для работы с искусственными нейронными сетями);
- Scilab Neural Network Module (модуль нейронных сетей);
- Fuzzy Toolbox (инструменты нечеткой логики).

Пакет Scilab свободно распространяется вместе с исходными кодами, то есть его использование, копирование, изменение и дальнейшее распространение не предусматривает какие-либо ограничения. Этот пакет защищен специальной лицензией, основное отличие которой от стандартной GNU лицензии, по утверждению авторов, определяется стремлением избежать появления клонов.

Существуют версии Scilab как для платформ Linux, так и для Windows. Во многие платформы Linux пакет Scilab включен в стандартную поставку, например в Ubuntu, Red Hat и openSUSE. Исходные тексты, рабочая версия для Windows и документация доступны в локальной сети университета.

Пакет Scilab предусматривает интерфейс взаимодействия с прикладными программами. Например, имеется возможность использовать откомпилированные функции языков C и Fortran.

В данное учебное пособие, помимо дополнительной теории к основным лекциям, входит лабораторный цикл, содержащий 8 лабораторных работ по изучению как общих методов программирования в Scilab, так и функционала готовых модулей, использующих математические методы нечеткой логики, искусственных нейронных сетей и генетические алгоритмы.

Основное назначение пособия – обеспечение необходимой теорией лабораторного практикума дисциплины «Системы искусственного интеллекта» для студентов вышеперечисленных направлений, однако оно может быть полезным и для самостоятельной подготовки аспирантов и магистрантов, применяющих математические методы анализа и моделирования в своей работе.

Тема. Введение в интеллектуальные системы.

Лабораторная работа № 1

ОСНОВЫ МАТЕМАТИЧЕСКОГО ПАКЕТА SCILAB
Основные понятия, простые функции

В математическом пакете Scilab все числовые данные рассматриваются в виде матриц. Вектор представляется как частный случай матрицы (матрица-строка или матрица-столбец). Тип результата вычисления определяется автоматически по виду выражения.

Идентификаторы должны обеспечивать хорошую информативность переменных, поэтому высота их букв имеет существенное значение. С этой целью рекомендуется для имен простых переменных выбирать строчные буквы, а для структурированных переменных (векторы и массивы) – прописные буквы.

Для записи векторов используются квадратные скобки, в которых записываются элементы вектора, разделенные запятыми (или пробелами). Например, вектор можно записать так: V=[1,2,3].

Матрицы также записываются с помощью квадратных скобок, при этом считается, что матрица в качестве строк содержит компоненты векторов. Строки (вектора) разделяются между собой знаком точка с запятой (;). Пример записи матрицы: V=[1,2,3; 4,5,6; 7,8,9].

Если данные, которые требуется записать в матричной форме в виде нескольких строк, не умещаются построчно, то каждую строку можно отобразить несколькими строками, используя разделитель в виде многоточия (не менее трех точек).

Значения известных математических констант задаются с помощью знака процента (%). Например, число π задается системной константой с именем %pi.

Для правильного вызова и использования встроенных функций рекомендуется чаще обращаться к «Справочной системе». В панели оглавления слева нужно выделить нужный раздел (на рис. 1.1 выде-

лен раздел «Основные функции»), при этом в правой панели будет отображаться список доступных функций и их назначение.

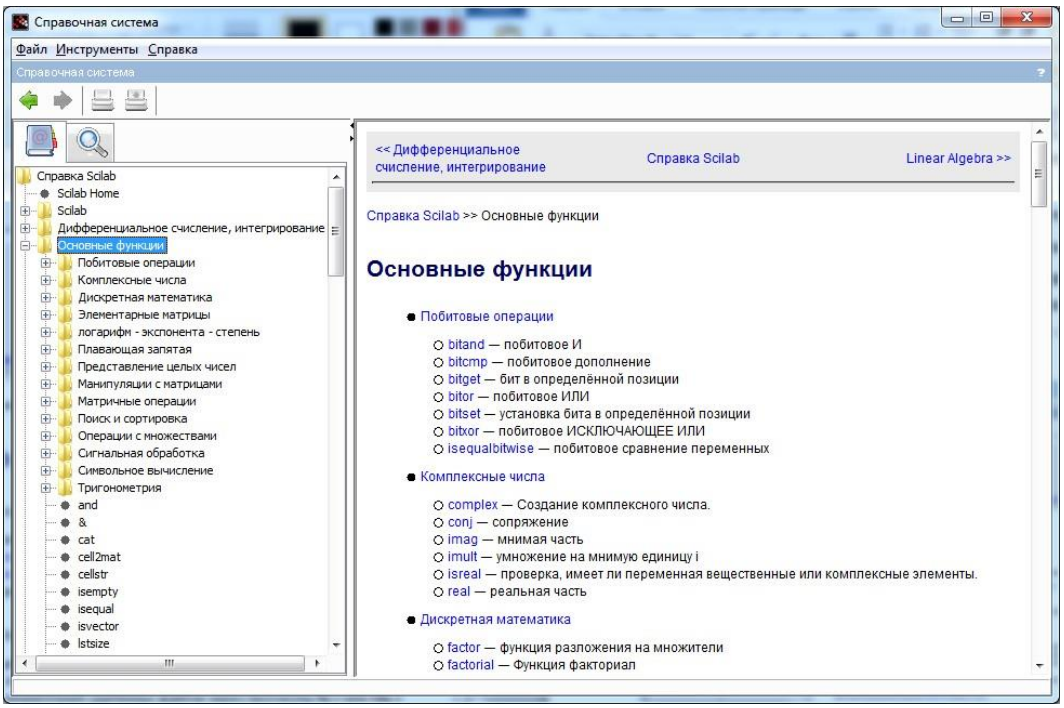


Рис. 1.1. Пример вывода содержания раздела

Для получения справки в правом окне нужно найти и щелкнуть по нужной функции. В правой панели отобразится ее описание с форматом обращения и примерами использования. На рис. 1.2 показано обращение к справке по функции случайных чисел **rand**.

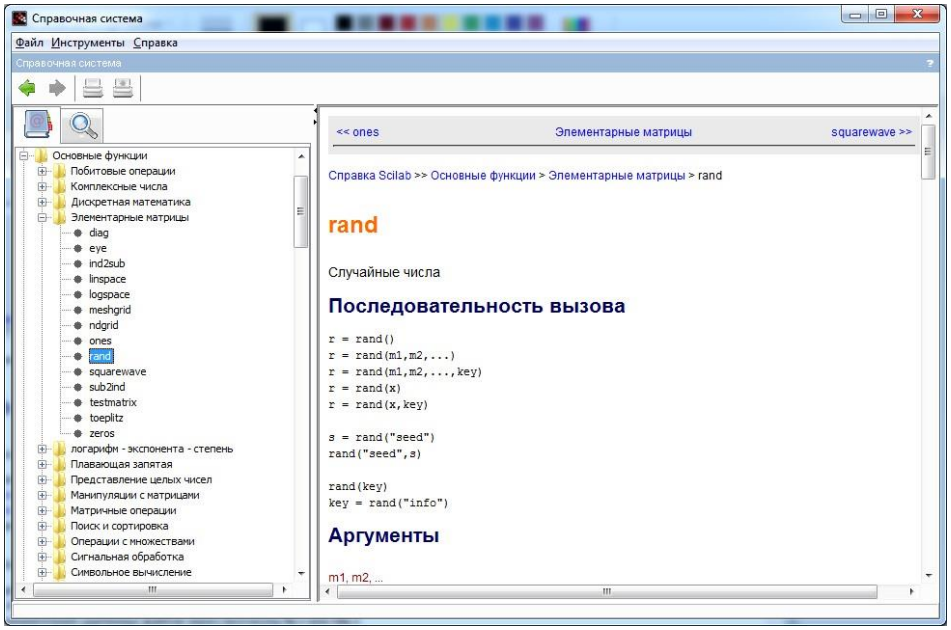


Рис. 1.2. Пример вывода справки по функции

Работа в математическом пакете Scilab может осуществляться в одном из двух возможных режимов:

- в командном окне, как с калькулятором (в этом случае каждое действие сразу же исполняется);
- в редакторе программ (в этом случае программа вводится как обычно, а исполняется по команде встроенного отладчика).

Когда выбрана работа в режиме калькулятора, то все выражения могут вводиться:

- в прямой форме, тогда после завершения ввода ответ будет выведен под встроенным системным именем **ans** (переменная с этим именем всегда хранит результат последнего вычисления);
- в форме оператора присвоения, когда переменной с выбранным именем присваивается значение выражения (ответ в этом случае выводится под именем этой переменной).

Если вычисляется значение переменной с выбранным именем по заданному выражению, то результат будет выводиться под именем этой переменной в следующей строке. Что касается векторов, то они выводятся в строке с пробелами, а матрицы выводятся построчно, причем каждая строка содержит вектор.

При работе в программном режиме результаты численных вычислений выводятся в командное окно строки, а построенные графики выводятся в отдельные окна. Командное окно вместе с окном редактора программ показаны на рис. 1.3.

Вывод результата можно заблокировать, если в конце строки ввода ввести знак точка с запятой (;). Значение переменной, которой результат присваивается, хранится в рабочей области.

При работе с массивами определены операторы почленного выполнения. В них перед символом операции вводится точка (**.**).

Символ присвоения – знак равенства (=). Равенство, как оператор отношения в условиях, вводится как двойное равенство (==).

На рис. 1.3 представлены совмещенные окна редактора программ и командное окно.

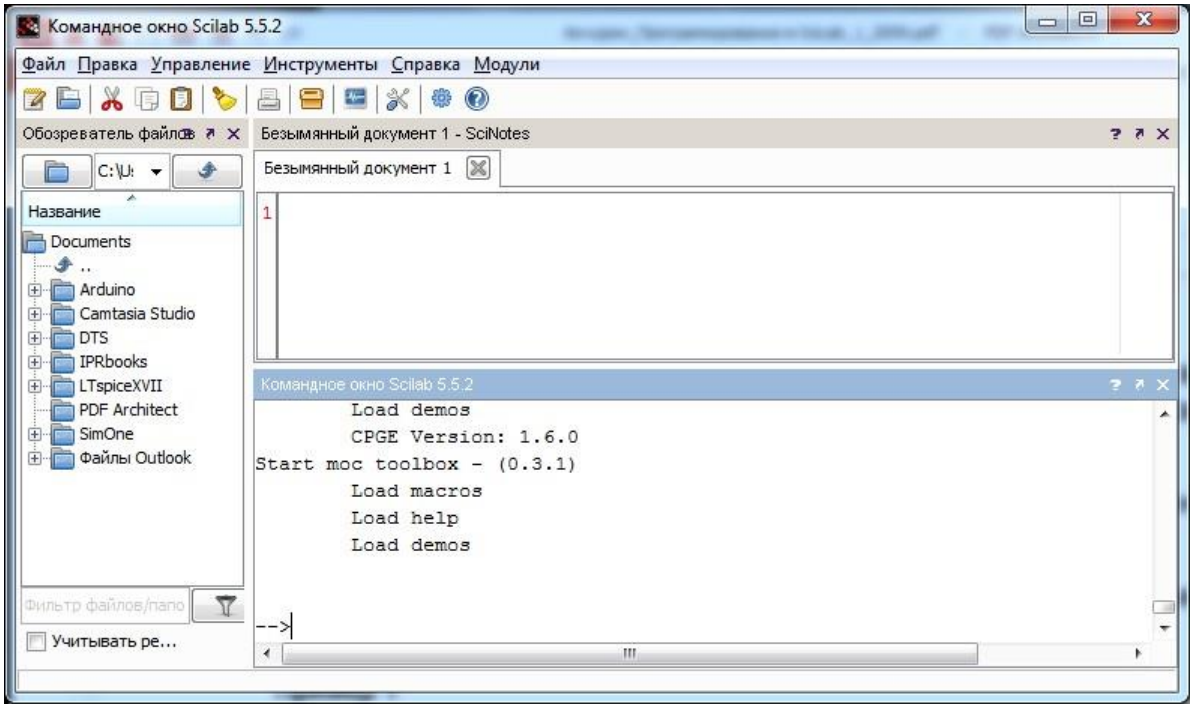


Рис. 1.3. Окно редактора программ и командное окно

Назначение кнопок панели инструментов в порядке слева направо:

– открывает редактор **SciNotes**

– открытие имеющегося файла

– вырезать фрагмент

– скопировать фрагмент

– вставить фрагмент

– очистка командного окна

– печать документа

– управление модулями **Atoms**

– моделирование в **Xcos**

– настройки Scilab

– демонстрационные примеры

– справочная система

Пункты главного меню

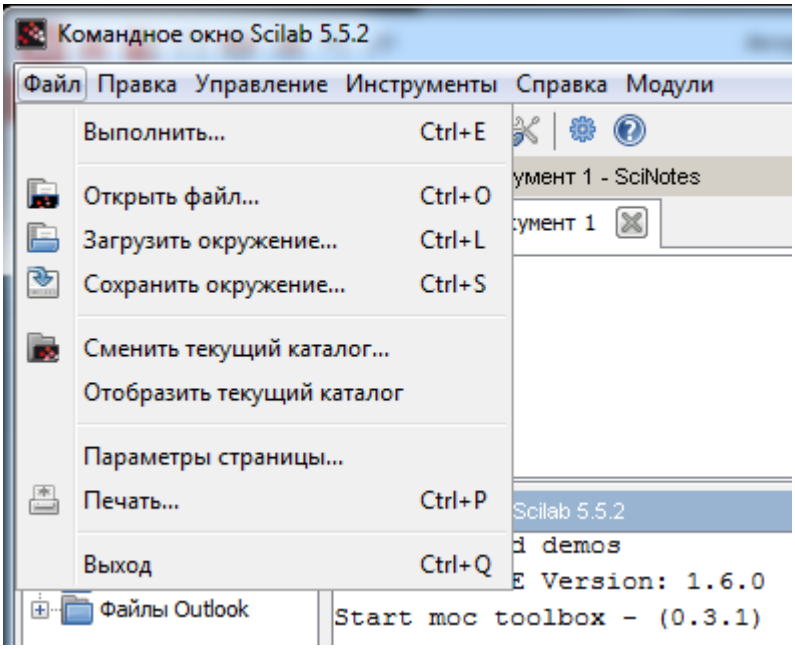


Рис. 1.4. Пункт «**Файл**»

В пункте «**Файл**» (рис. 1.4) имеются средства работы с файлами

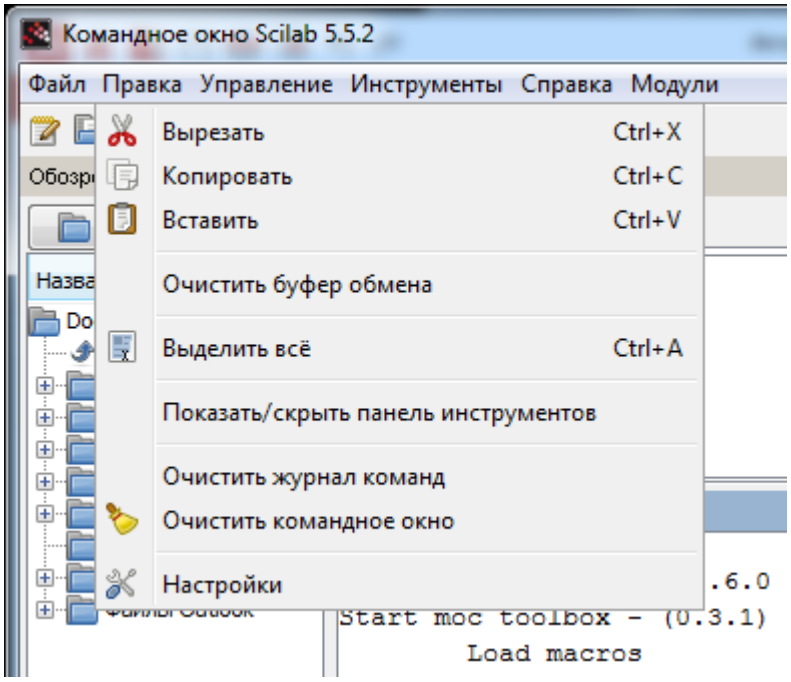


Рис. 1.5. Пункт «**Правка**»

В пункте «**Правка**» (рис. 1.5) определены стандартные средства редактирования программного кода.

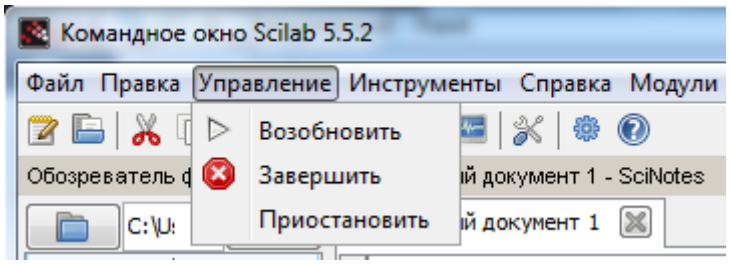


Рис. 1.6. Пункт «**Управление**»

В этом пункте (рис. 1.6) представлены основные средства управления программой.

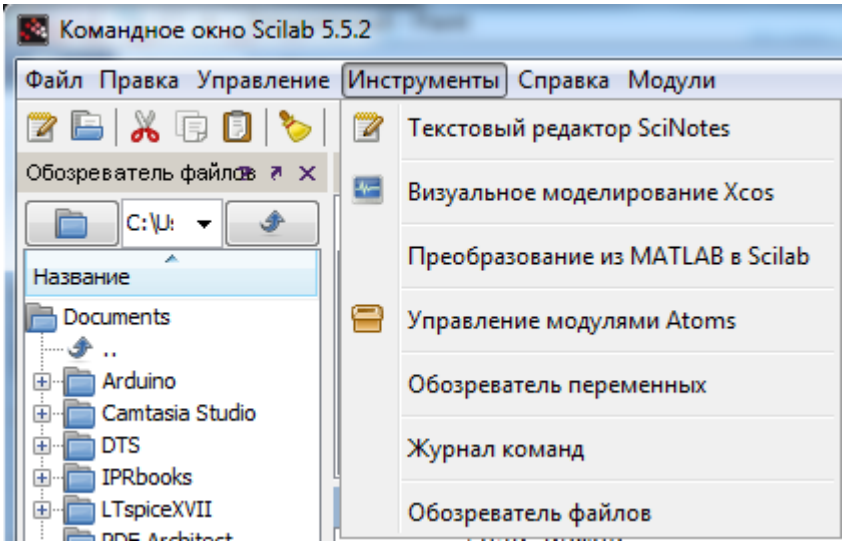


Рис. 1.7. Пункт «**Инструменты**»

Этот пункт (рис. 1.7) определяет инструментальные средства.

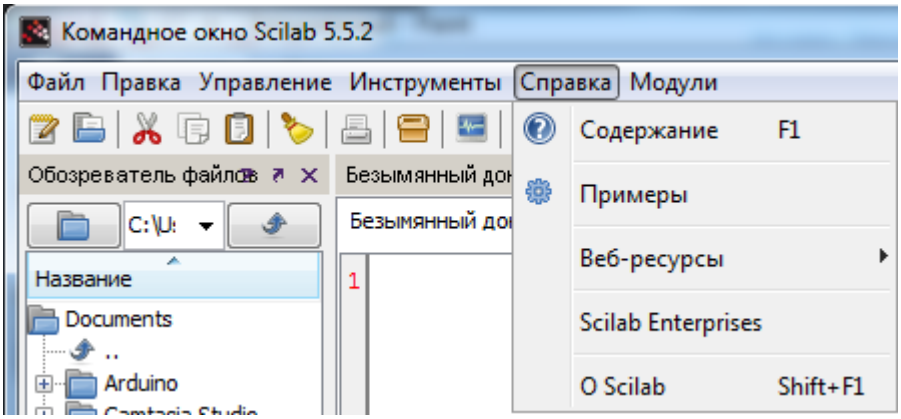


Рис. 1.8 Пункт «**Справка**»

В этом пункте (рис. 1.8) определена справочная информация, примеры и сведения о программе.

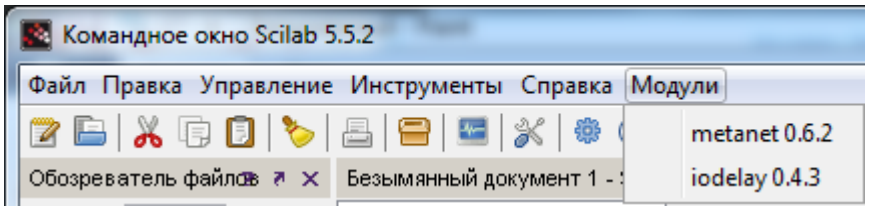


Рис. 1.9. Пункт «**Модули**»

Этот пункт (рис. 1.9) дает информацию о модулях.

Чтобы посмотреть демонстрационные примеры типовых программ Scilab через меню, нужно использовать команду «**Справка => Примеры**», либо нажать на кнопку  панели инструментов.

В этом случае откроются иерархически связанные окна выбора наглядных примеров работы программы Scilab.

На рис. 1.10 показан пример, в котором последовательно выбираются «**Графика**» => «**2D и 3D графики**» => «**hist3d**».

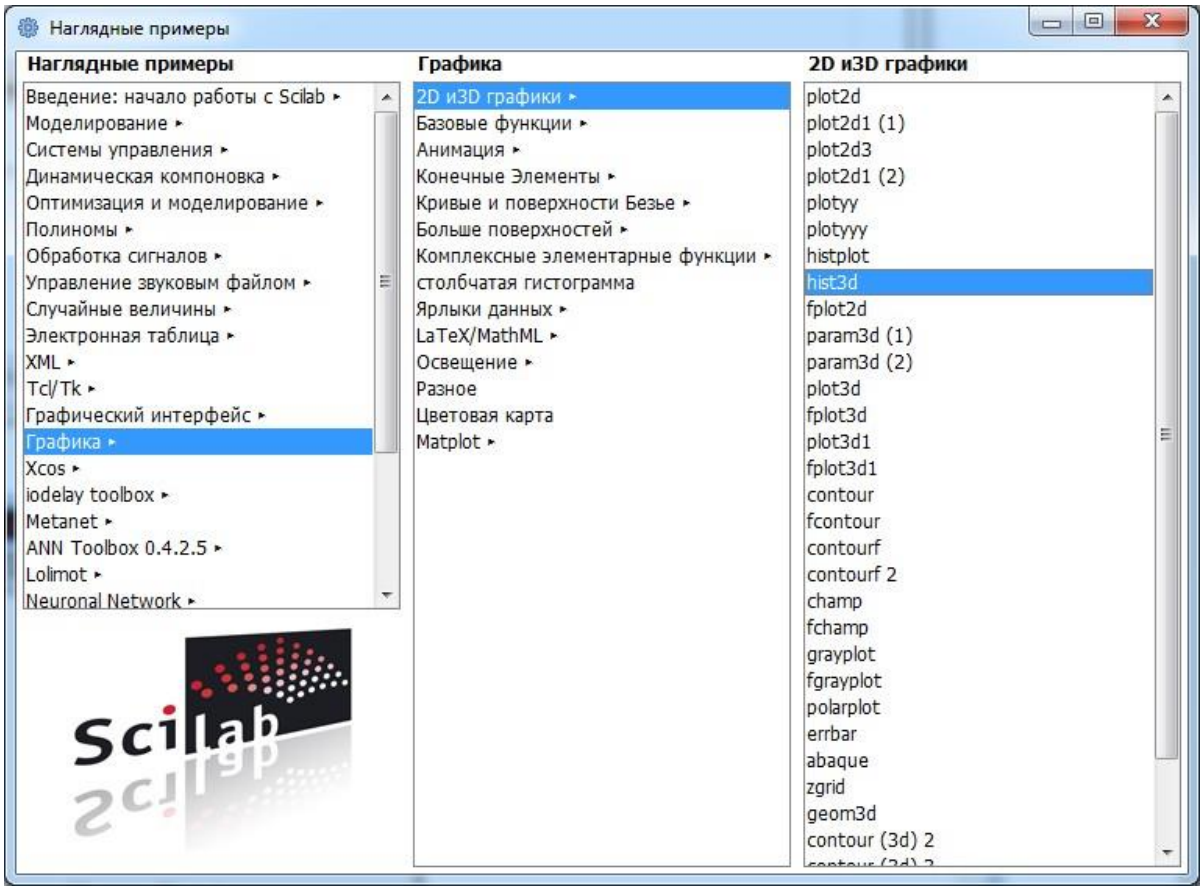


Рис. 1.10. Выбор примера построения графика

В результате этого выводится трехмерная гистограмма распределения, показанная на рис. 1.11.

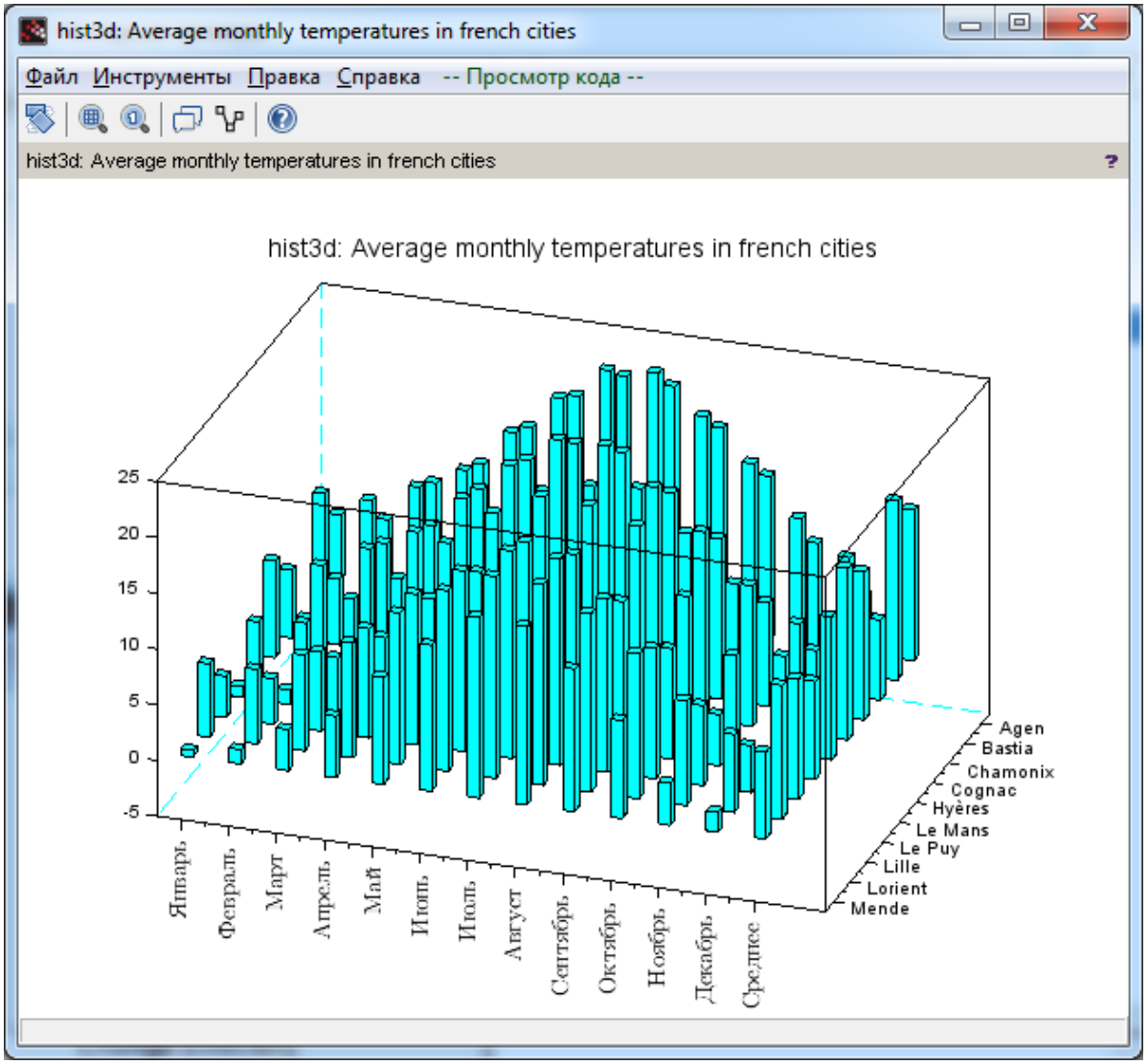


Рис. 1.11. Пример вывода трехмерной гистограммы распределения

Программирование сценариев в Scilab можно осуществлять как через командное окно, так и с помощью редактора Scipad. Второй вариант предпочтительнее, так как в редакторе содержится встроенный отладчик, облегчающий отладку сценариев.

В первой лабораторной работе даются навыки программирования вычисления двух функций. Результаты выводятся в виде двумерных графиков с использование графических функций высокого уровня.

Задания к работе

Задание 1.

- В заголовке программы ввести текст в виде комментария, в котором отобразить свою фамилию с инициалами, группу, а также номер варианта, назначенный преподавателем.
- Ввести исходные данные своего варианта.
- Задать изменение аргумента функций.
- Вычислить значения функции 1 и 2 для значений аргумента в заданном интервале.
- Вывести графики обеих функций на одном графике в декартовых координатах (для графиков использовать разные типы линий).

Задание 2. Повторить задание 1, но графики функций вывести в двух подокнах в столбиковом формате.

Задание 3. Повторить задание 1, но графики вывести в 4-х подокнах (по 2 подокна на каждую функцию) с разными стилями линий (использовать функции plot2d, plot2d2, plot2d3, plot2d4).

Варианты заданий

№	Функция 1	Функция 2	a	b	h
1	y = 2sin(x)	z= exp(x+4)/100 - 10	-2pi	2pi	pi/10
2	y = cos(x)	z = 0.1exp(5 - x) - 6	-2pi	2pi	pi/20
3	y = |sin(x)| + 1	z = (1+x)^2	-2pi	2pi	pi/10
4	y = (x^2 - 1)/10	z = 1+3sin(x)	-2pi	2pi	pi/20
5	y = (x^3 - 2)/100	z = 5cos(x)	-2pi	2pi	pi/10
6	y = x^2 - 100	z = 0.01exp(-1.5x)	-5	5	0.2
7	y = 3sin(x) - 2	z=0.025x^3	-6	6	0.3
8	y = 4cos(x) + 1	z = 0.05x^2	-4	+4	0.1
9	y = 0.2exp(1 - x)	z = 2 - sin(x)	-2pi	2pi	pi/10
10	y = cos(2x) - 1	z = (5+x)^2 - 20	-5	5	0.1
11	y = (3x^2 + 100)/10	z = cos(3x) + 2	-2pi	2pi	pi/10
12	y = |cos(4x)| - 1	z = 0.5exp(-3.5x)	-6	6	0.2

Методические указания

Все пояснения в виде текста вводятся в программу как комментарий. Синтаксически комментарий начинается с символов //, которые располагается в первой позиции строки. Нужно помнить, что комментарий – это просто текст, и в него не включают символы операций.

Чтобы сформировать двумерный график в координатах XY, необходимо выполнить следующие операции:

- Задать аргумент в формате **x=<начало>:<шаг>:<конец>**.
- Вычислить функцию, например, **y=f(x)**.

Далее осуществляют вывод графика, например, с помощью процедуры **plot(x,y,s)**. Эта процедура рисует график прямыми отрезками линий между вычисленными точками. Аргумент **s** – это строковая константа, задающая параметры линии (цвет, тип точки, тип линии). При ее отсутствии параметры линии выбираются по умолчанию. Определены следующие значения аргумента **s**:

Цвет линии		Тип точки		Тип линии	
y	желтый	.	точка	-	сплошная
m	фиолетовый	o	кружок	:	двойной пунктир
c	голубой	x	крест	-.	штрихпунктир
r	красный	+	плюс	--	штрих
g	зеленый	*	звездочка		
b	синий	s	квадрат		
w	белый	d	ромб		
k	черный	^	треугольник вверх		
		v	треугольник вниз		
		<	треугольник влево		
		>	треугольник вправо		

В случае необходимости отображения на одном графике несколько функций, например, **y1=f(x)** и **y2=f(x)**, их вначале вычисляют, а затем выводят процедурой **plot(x,y1,'s1',x,y2,'s2'…)**. В качестве параметров для каждой функции следуют строки символов в произвольной последовательности, которые определяют тип линии, тип точки и цвет линии.

Чтобы создать в одном графическом окне несколько подокон, для вывода графиков используется процедура **subplot(m,n,p)**, где параметр **m** – число подокон в окне по горизонтали (число строк), параметр **n** – число подокон по вертикали (число столбцов), а **p** – номер подокна, используемого для дальнейшего вывода графика. Нумерация начинается с единицы слева-направо и сверху-вниз.

Для формирования графика в столбиковой форме (каждому вычисленному отсчету соответствует достаточно широкая вертикальная полоска) нужно использовать процедуру **bar(x,y)**. При выводе такого графика в коде программы должны быть задействованы функции **subplot(m,n,p)** и **bar(x,y)**.

Если нужно сформировать графики в форме с разными типами линий (сплошная, ступенчатая, вертикальные полоски, со стрелками), то нужно использовать функции **plot2d, plot2d2, plot2d3, plot2d4**.

Пример 1

Задание

- Функция 1 — y = 2*sin(x)
- Функция 2 — z = 0.02*x^3
- Начальное значение аргумента — a = -8
- Конечное значение аргумента — b = 8
- Шаг изменения аргумента — h = 0.5

Листинг программы

```
// Диапазон и шаг
a=-8; b=8; h=0.5;
// Задание аргумента
X=a:h:b;
// Расчет функций
Y=2*sin(X); Z=0.02*X.^3;
// Вывод с типами по умолчанию в окно 1
scf(1); plot(X,Y,X,Z); xgrid()
// Вывод с выбираемыми типами в окно 2
scf(2); plot(X,Y,'-gx',X,Z,':*r'); xgrid()
```

После запуска программы выводятся графики в отдельных окнах, как показано на рис. 1.12 и 1.13.

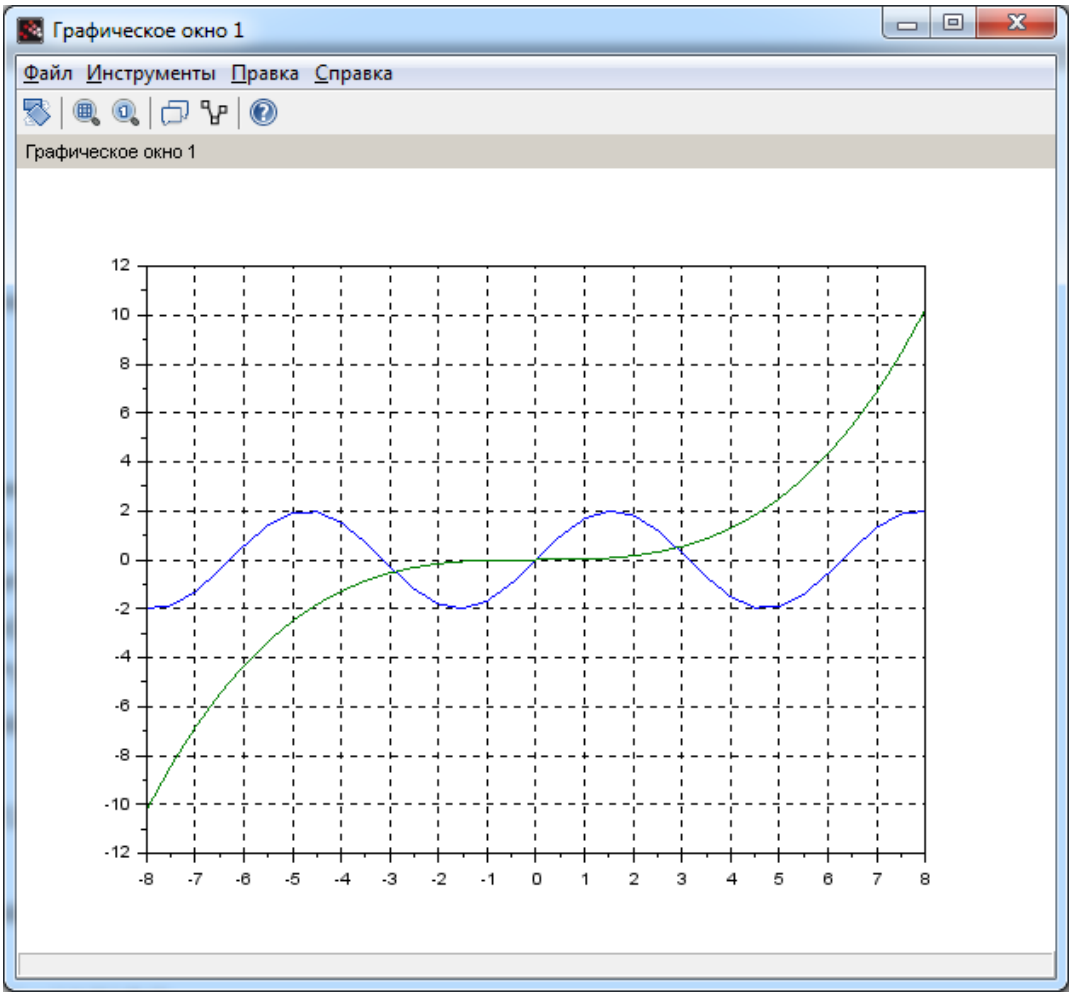


Рис. 1.12. Вывод графиков с одинаковым типом линий разного цвета

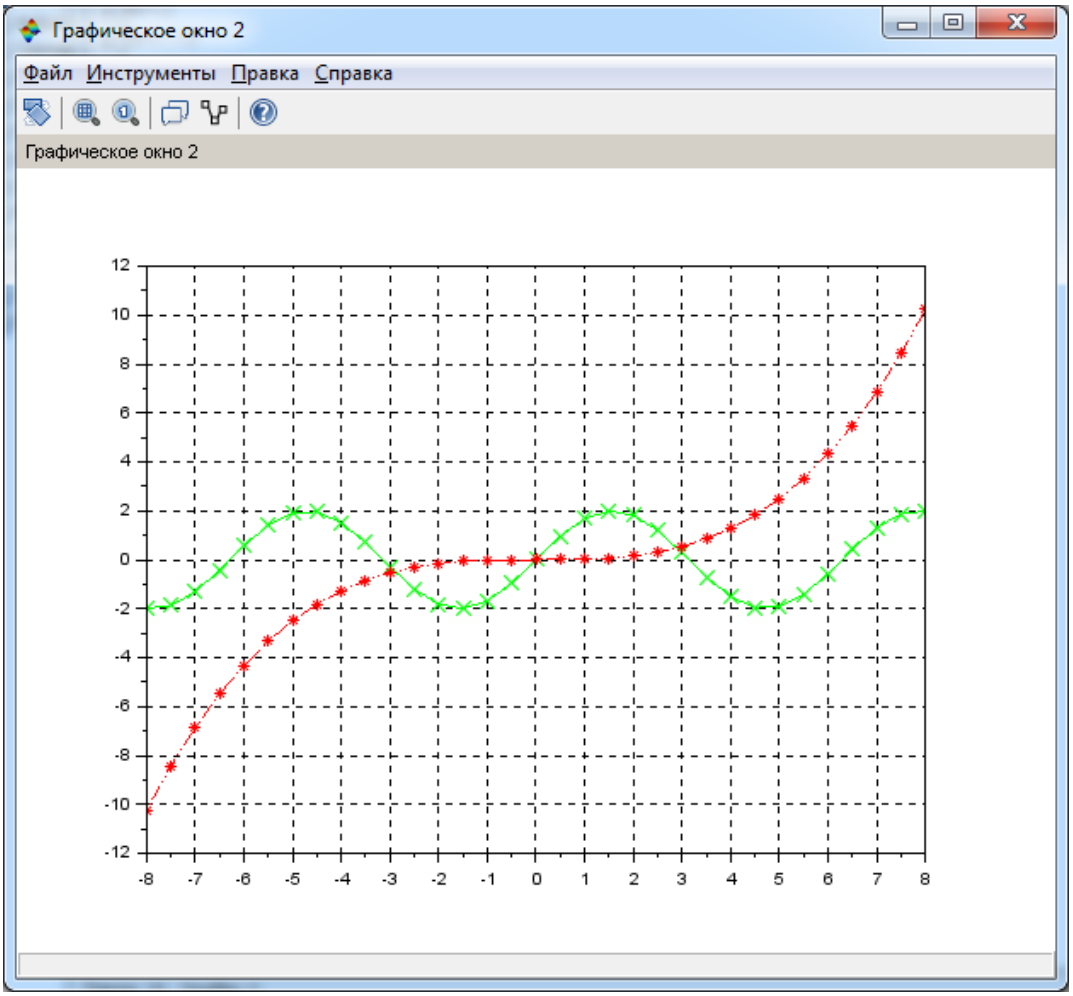


Рис. 1.13. Вывод графиков с разным типом линий и разного цвета

Пример 2

Листинг программы

```
// Подокна и функция bar
a=0;
b=20;
h=1;
// Задание аргумента
X=a:h:b;
// Расчет функций
Y=2*sin(X);
Z=0.02*X.^3;
// Вывод Y столбиками в подокно 1
subplot(2,1,1);
```

```
bar(X,Y);
// Включение координатной сетки
xgrid()
//Вывод Z столбиками в подокно 2
subplot(2,1,2);
bar(X,Z);
// Включение координатной сетки
xgrid()
```

После запуска программы выводятся графики в двух подокнах одного окна, как показано на рис. 1.14.

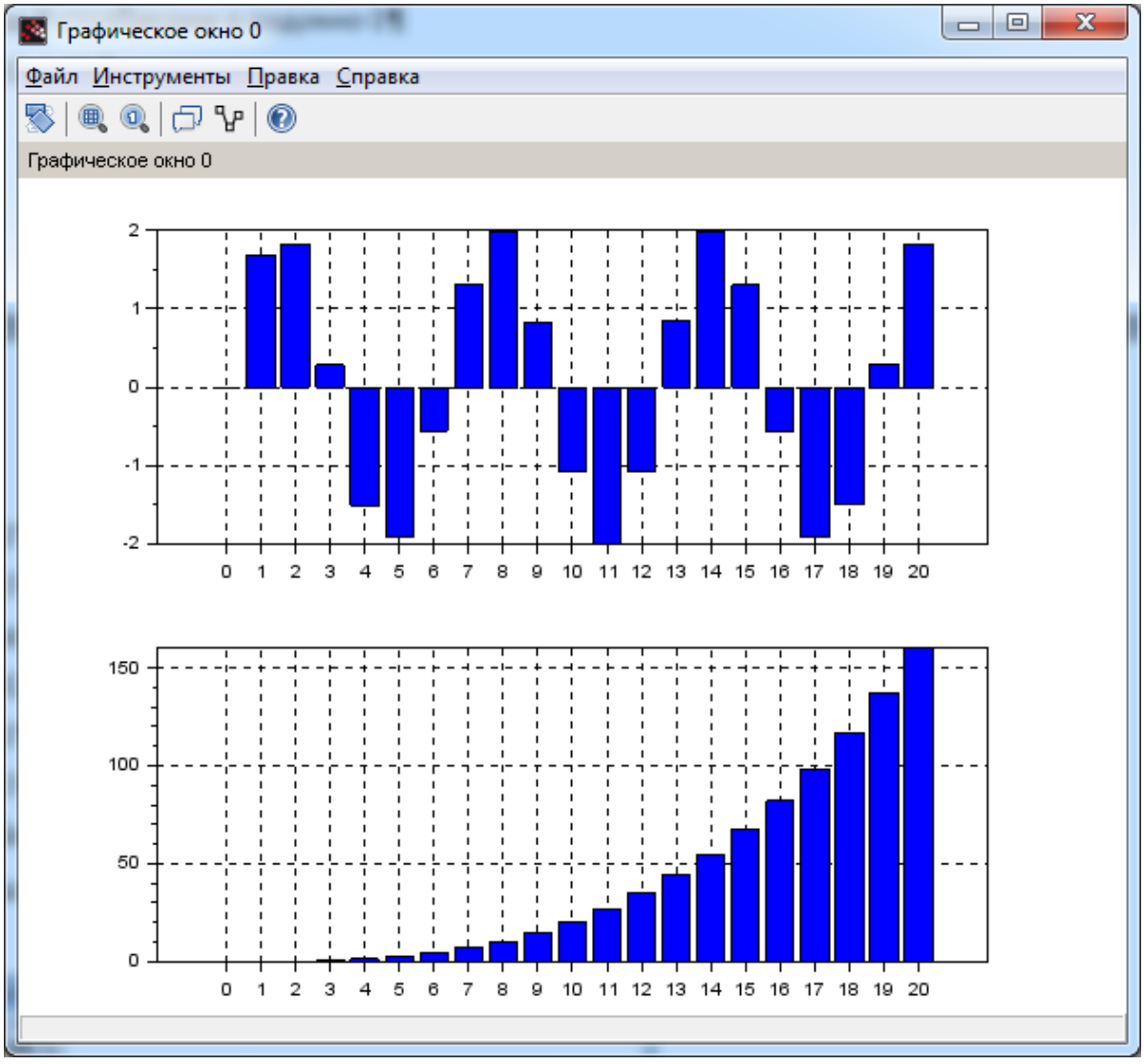


Рис. 1.14. Вывод графиков в двух подокнах одного окна

Пример 3

Листинг программы

```
// Подокна и функции со стилями линий
a=0;
b=20;
h=1;
// Задание аргумента
X=a:h:b;
// Расчет функций
Y=2*sin(X);
// Вывод Y в подокно 1
subplot(2,2,1);
plot2d(X,Y);
// Включить координатную сетку
xgrid()
// Вывод Y ступенькой в подокно 2
subplot(2,2,2);
plot2d2(X,Y);
//Включить координатную сетку
xgrid()
// Вывод Y вертикальными полосками в подокно 3
subplot(2,2,3);
plot2d3(X,Y);
// Включить координатную сетку
xgrid()
// Вывод Y со стрелками в подокно 4
subplot(2,2,4);
plot2d4(X,Y);
// Включить координатную сетку
xgrid()
```

После запуска программы выводятся графики в четырех подокнах одного окна, как показано на рис. 1.15.

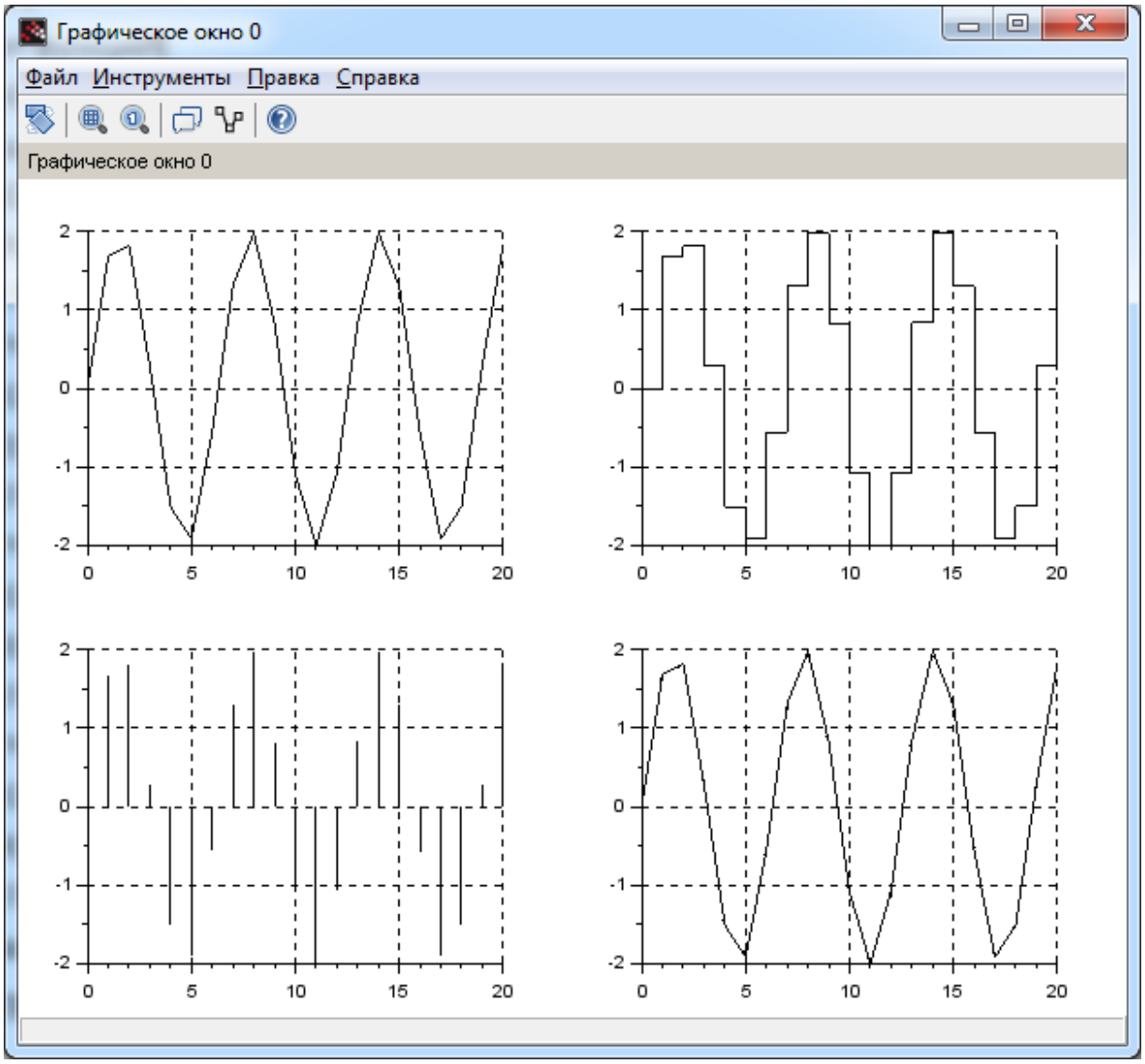


Рис. 1.15. Вывод графиков в двух подокнах одного окна

Контрольные вопросы

1. Структура окна редактора Scilab.
2. Правила ввода команд.
3. Правила ввода функций и операндов.
4. Правила ввода выражений.
5. Организация циклов.
6. Правила ввода комментариев.
7. Правила просмотра результатов операций.
8. Правила создания двумерных графиков.
9. Запуск и отладка программ.

Тема. Модель представления знаний

Лабораторная работа № 2

Пространственные кривые и поверхности

При программировании сценариев в данной лабораторной работе используется редактор Scipad, как и в первой работе. С помощью редактора формируется файл программы, в заголовке которого в виде комментария отображается фамилия с инициалами, группа, а также номер варианта, назначенный преподавателем.

Эта лабораторная работа состоит из двух частей, каждая из которых имеет свою таблицу вариантов. Соответственно, в первой части рассматриваются два задания, посвященные пространственным кривым, а во второй части рассматриваются два задания, посвященные построению поверхностей.

Часть первая. Вначале рассмотрим процесс программирования вычислений функций, необходимых для построения пространственных кривых. Предполагаем, что результаты вычислений должны выводиться в виде трехмерных графиков с использованием графических функций высокого уровня **param3d** и **param3d1**.

Формирование заданий. По первой части работы предусмотрены два задания, в каждом из которых вычисляется функции, описывающие пространственные кривые, и строятся объемные графики с использованием различных графических функций. В первом задании рисуется одна кривая, а во втором задании две кривые.

Для формирования пространственных кривых необходимо:

- задать число точек по координатам X и Y;
- вычислить элементы векторов координат X, Y;
- создать графическое окно и вывести график выбранного типа.

Задания к работе

Задание 1. Функции пространственных кривых (функция **param3d**).

- Ввести исходные данные варианта.
- Вычислить координаты пространственной кривой.

- Вывести эту кривую в виде трехмерного графика.

Задание 2. Функции пространственных кривых (функция **param3d1**).

- Ввести исходные данные варианта.
- Вычислить координаты двух пространственных кривых.
- Вывести обе кривые в виде трехмерного графика.

В таблице ниже задаются две функции для расчета пространственных координат X, Y. Для выполнения задания 1 нужно использовать только одну первую функцию.

Варианты заданий по первой части

№	X	Y	t
1	sin(t), sin(2t)	cos(t), cos(2t)	0:0.1:5*%pi
2	sin(1.5t), sin(2t)	cos(1.5t), cos(2t)	0:0.1:4*%pi
3	sin(2t), sin(3t)	cos(2t), cos(3t)	0:0.1:3*%pi
4	sin(2.5t), sin(4t)	cos(2.5t), cos(4t)	0:0.1:2*%pi
5	sin(t), sin(2t)	cos(t), cos(2t)	0:0.1:5.5*%pi
6	sin(1.5t), sin(2t)	cos(1.5t), cos(2t)	0:0.1:4.5*%pi
7	sin(2t), sin(3t)	cos(2t), cos(3t)	0:0.1:3.5*%pi
8	sin(2.5t), sin(4t)	cos(2.5t), cos(4t)	0:0.1:2.5*%pi
9	sin(t), sin(2t)	cos(t), cos(2t)	0:0.1:6.5*%pi
10	sin(1.5t), sin(2t)	cos(1.5t), cos(2t)	0:0.1:5.5*%pi
11	sin(2t), sin(3t)	cos(2t), cos(3t)	0:0.1:4.5*%pi
12	sin(2.5t), sin(4t)	cos(2.5t), cos(4t)	0:0.1:3.5*%pi

Пример 1

Кривая 1: x=sin(t), y=cos(t),

Используется функция **param3d**.

Листинг программы

```
// Функция param3d, пространственная кривая
t=0:0.1:4*%pi;
param3d(sin(t),cos(t),t/10,35,45,"X@Y@Z",[2,3])
e=gce() // обработчик 3D полилинии
e.foreground=color('red');
a=gca(); // обработчик осей
a.rotation_angles=[35 90];
```

Описание аргументов функции **param3d** имеется в справочной системе. Однако для более глубокого понимания их назначения рекомендуется с ними поэкспериментировать.

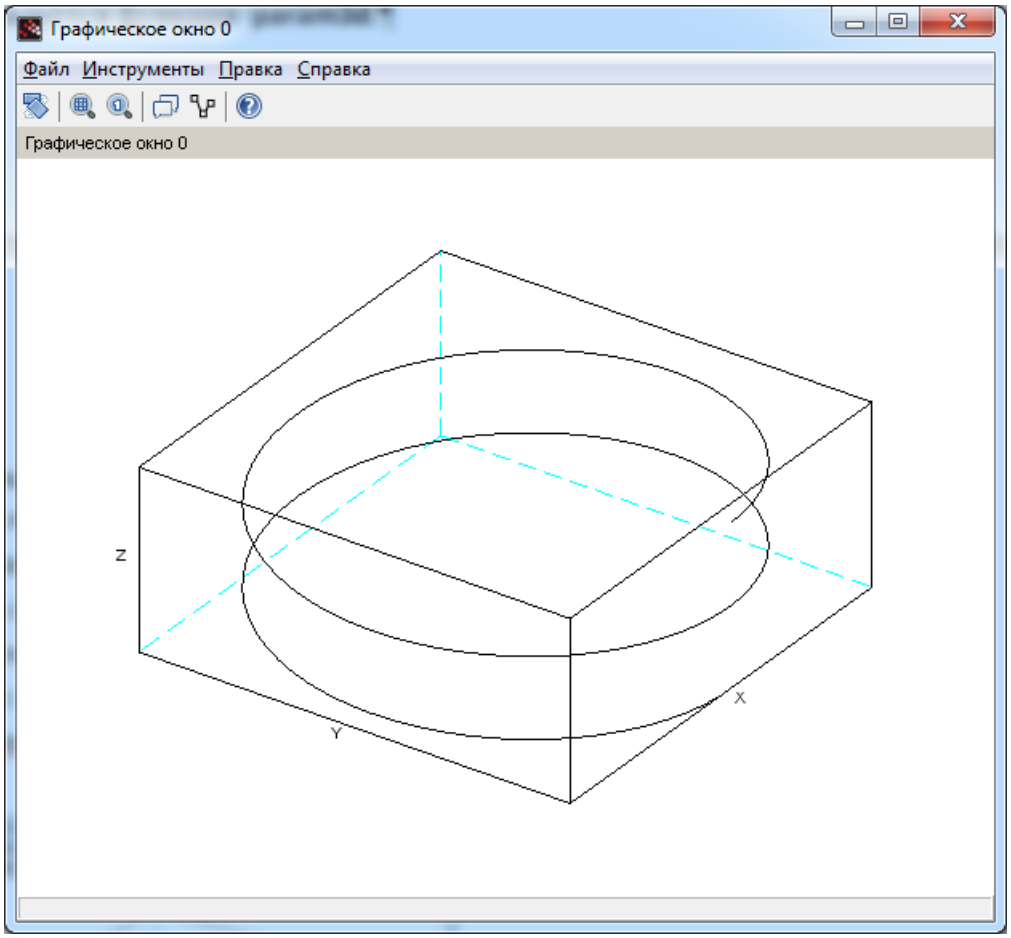


Рис. 1.16. Вывод кривой с параметрами по умолчанию (без последних 4-х строк)

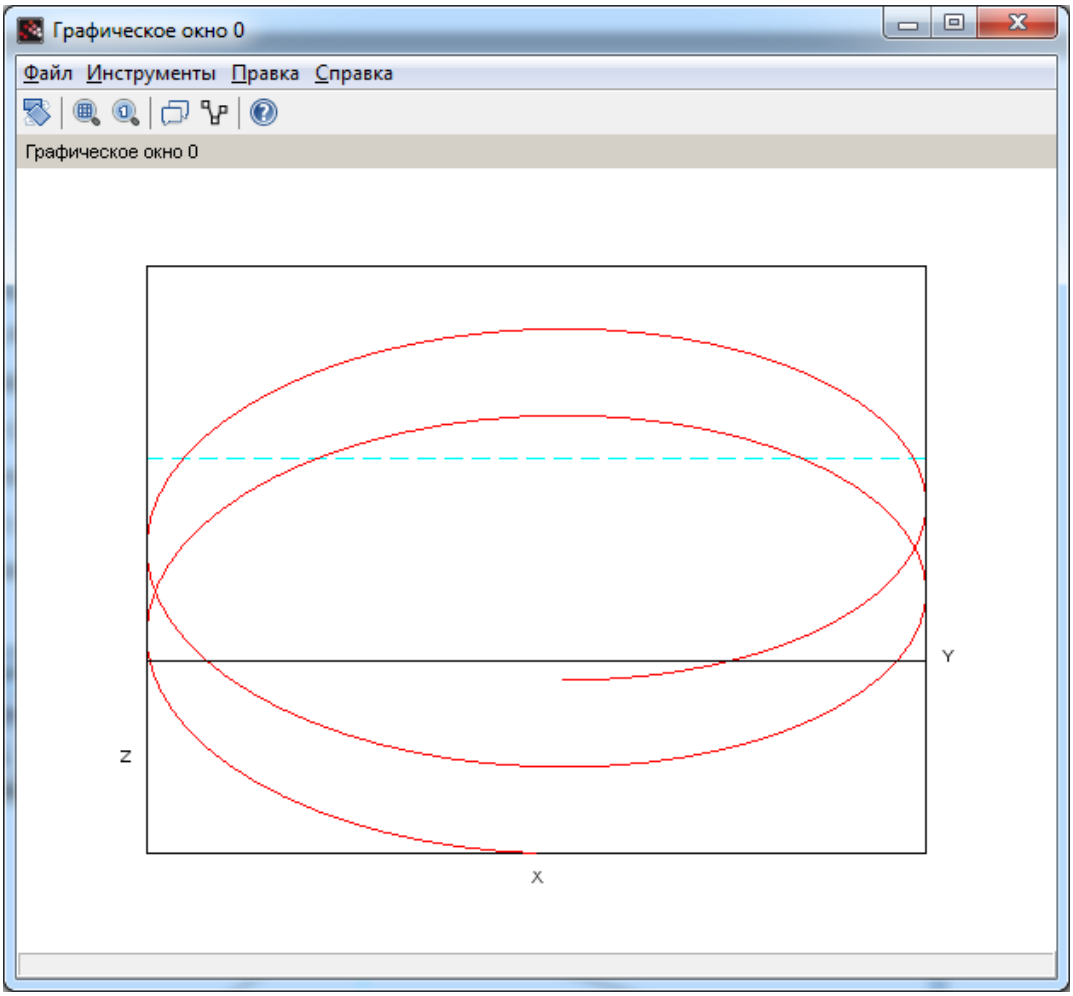


Рис. 1.17. Вывод пространственной кривой по полному листингу

Например, из анализа полного листинга и рис. 1.17 можно сделать некоторые полезные выводы о назначении двух угловых параметров **theta** и **alpha**.

Изменение параметра **alpha** до 90 градусов приводит к ориентации плоскости YZ перпендикулярно плоскости экрана. Значит **alpha** отвечает за поворот угла обзора в горизонтальной плоскости относительно оси Z. Аналогичный эксперимент с угловым параметром **theta** приводит к выводу, что этот параметр отвечает за поворот угла обзора в вертикальной плоскости относительно плоскости XY.

Пример 2

Кривая 1: sin(t), cos(t). Кривая 2: sin(2*t), cos(2*t).

Если используется функция **param3d1**, то это позволяет нарисовать сразу несколько пространственных кривых.

Листинг программы

```
// Функция param3d1, пространственные кривые
t=[0:0.1:5*%pi]';
param3d1([sin(t),sin(2*t)],[cos(t),cos(2*t)],..
list([t/10,sin(t)],[3,2]),35,45,"X@Y@Z",[2,3])
```

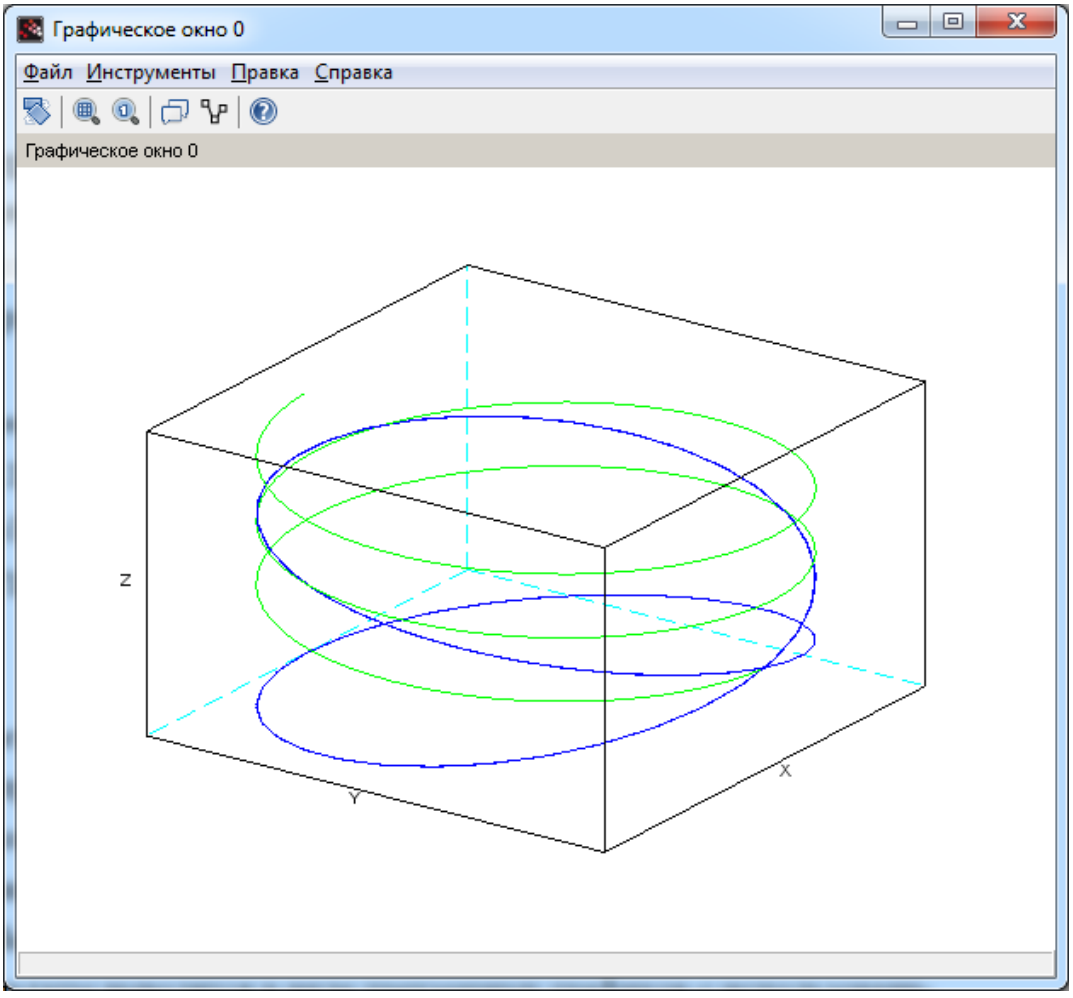


Рис. 1.18. Вывод двух пространственных кривых

Часть вторая. Во второй части работы программируются вычисления функций для поверхностей. При этом предполагается, что все результаты вычислений должны выводиться в виде трехмерных графиков с использованием графических функций высокого уровня, таких как **plot3d, mesh, surf** и **contour**.

Формирование заданий. Во второй части работы предусмотрены также два задания, в каждом из которых вычисляется двумерная функция, описывающая объемную фигуру, и строятся поверхностные и контурные графики с использованием перечисленных графических функций. В первом задании каждый график выводится в отдельное окно, а во втором задании в подокна одного общего окна.

Поверхностный и контурный графики. Для формирования поверхностного или контурного графика необходимо:

- задать число точек по координатам X и Y;
- создать вложенные циклы по X и Y с вычислением функции Z=f(X,Y);
- ввести номер графического окна и вывести в него график выбранного типа.

Следует использовать графики:

- трехмерный с аксонометрией, функция **plot3(X,Y,Z)**;
- трехмерный с функциональной окраской, функция **mesh(X,Y,Z)**;
- трехмерный с функциональной окраской и проекцией, функция **surf(X,Y,Z)**;
- контурный, функция **contour(X,Y,Z)**.

Задание 1. Трехмерная графика (функции **plot3d, mesh, surf, contour**).

- Ввести исходные данные.
- Вычислить функцию.
- Вывести функцию в виде трехмерных графиков разного типа.

Задание 2. Повторить задание 1 с отображением графиков в подокнах.

Варианты заданий по второй части

№	Функция	Пределы изменения	
		x	y
1	z=sin(x)cos(y)	от -6 до 6	от -6 до 6
2	z=sin(x/2)cos(y)	от -2pi до 2pi	от -2pi до 2pi
3	z=sin(2x)cos(y)	от -6 до 6	от -6 до 6
4	z = sin(x)cos(y/2)	от -2pi до 2pi	от -2pi до 2pi
5	z = sin(x/2)cos(2y)	от -6 до 6	от -6 до 6
6	z = sin(2x)cos(2y)	от -2pi до 2pi	от -2pi до 2pi
7	z = (1+sin(x)/x)(sin(y)/y)	от -6 до 6	от -6 до 6
8	z = (sin(x)/x)cos(y)	от -2pi до 2pi	от -2pi до 2pi
9	z = sin(2x)cos(y/3)	от -6 до 6	от -6 до 6
10	z = sin(x/2)cos(4y)	от -2pi до 2pi	от -2pi до 2pi
11	z = sin(3x)cos(2y)	от -6 до 6	от -6 до 6
12	z = (2+sin(x)/x)(sin(y)/y)	от -2pi до 2pi	от -2pi до 2pi

Пример 1

Функция $\frac{x}{x}\ \frac{y}{y}$

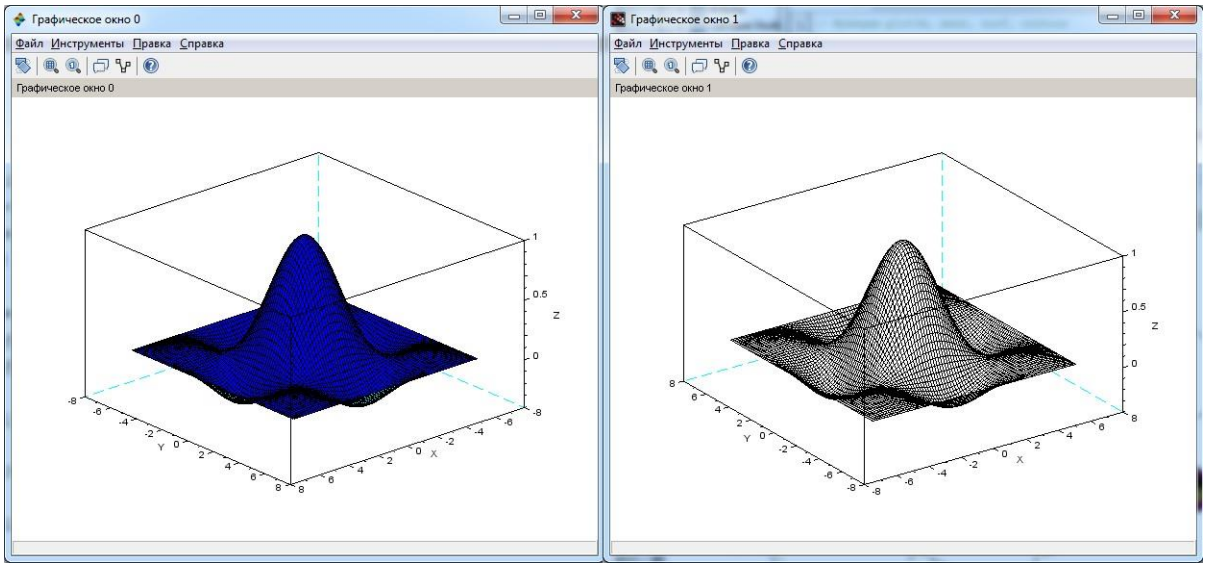
Диапазон аргументов -2pi...2pi

Листинг программы

```
// Функции plot3d, mesh, surf, contour
N=40;
h=%pi/20;
// Расчет матрицы
for n=1:2*N+1
  if n==N+1
    A(n)=1;
  else A(n)=sin(h*(n-N-1))/(h*(n-N-1));
  end;
```

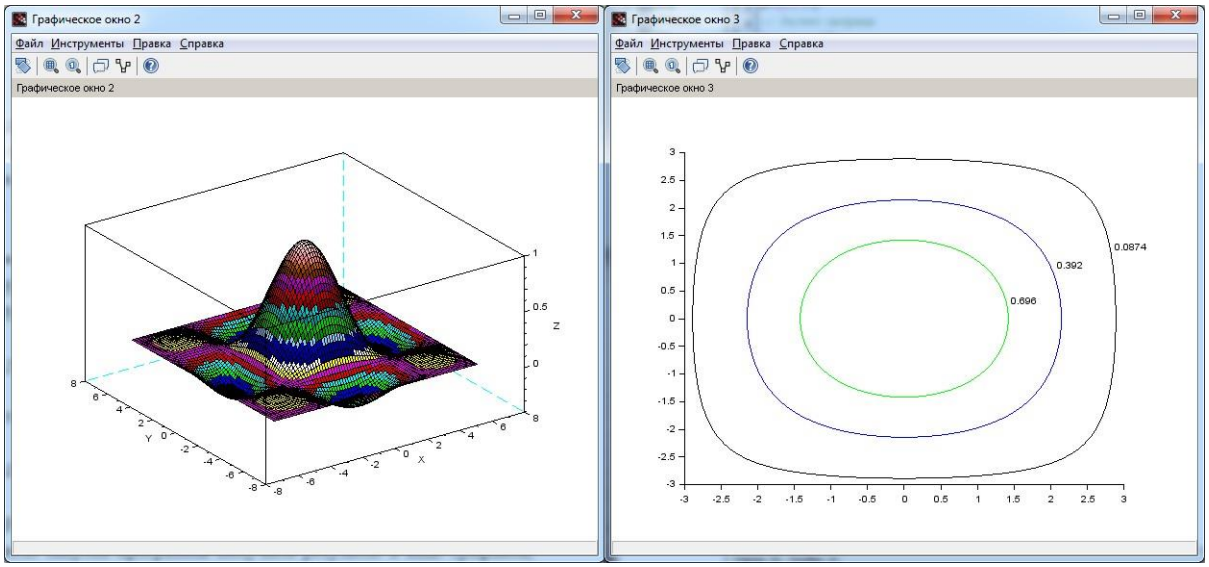
```
end;
for n=1:2*N+1
  for m=1:2*N+1
    Z(n,m)=A(n)*A(m);
  end;
end;
// Задание площадки
X=-N*h:h:N*h;
Y=-N*h:h:N*h;
scf();
plot3d(X,Y,Z*100);// Окно 1. 3d график с монотонной окраской
scf();
mesh(X,Y,Z);// Окно 2. 3d график, каркас
scf();
surf(X,Y,Z);// Окно 3. 3d график с функциональной окраской
scf();
contour(X,Y,Z,3);// Окно 4. 3d график, контуры
```

После запуска программы получаем результат (рис. 1.19) в виде графиков, которые представлены в отдельных окнах.



а) б)

Рис. 1.19. Вывод 3D графиков: а) с монотонной окраской; б) каркас



а) б)

Рис. 1.20. Вывод 3D графиков: а) с функциональной окраской; б) контуры

Пример 2

Листинг программы

```
// Функции plot3d, mesh, surf, contour. Используются подокна.
N=40;
h=%pi/20;
// Расчет матрицы
for n=1:2*N+1
   if n==N+1
     A(n)=1;
   else A(n)=sin(h*(n-N-1))/(h*(n-N-1));
   end;
 end;
for n=1:2*N+1
  for m=1:2*N+1
    Z(n,m)=A(n)*A(m);
  end;
end;
// Задание площадки
X=-N*h:h:N*h;
Y=-N*h:h:N*h;
```

```
subplot(2,2,1); // Подокно 1. 3d график с монотонной окраской
plot3d(X,Y,Z*100);
subplot(2,2,2); // Подокно 2. 3d график, каркас
mesh(X,Y,Z);
subplot(2,2,3); // Подокно 3. 3d график с функциональной окраской
surf(X,Y,Z);
subplot(2,2,4); // Подокно 4. 3d график, контуры
contour(X,Y,Z,3);
```

После запуска программы получаем результат (рис. 1.21), в котором графики отображены в отдельных подокнах одного общего окна.

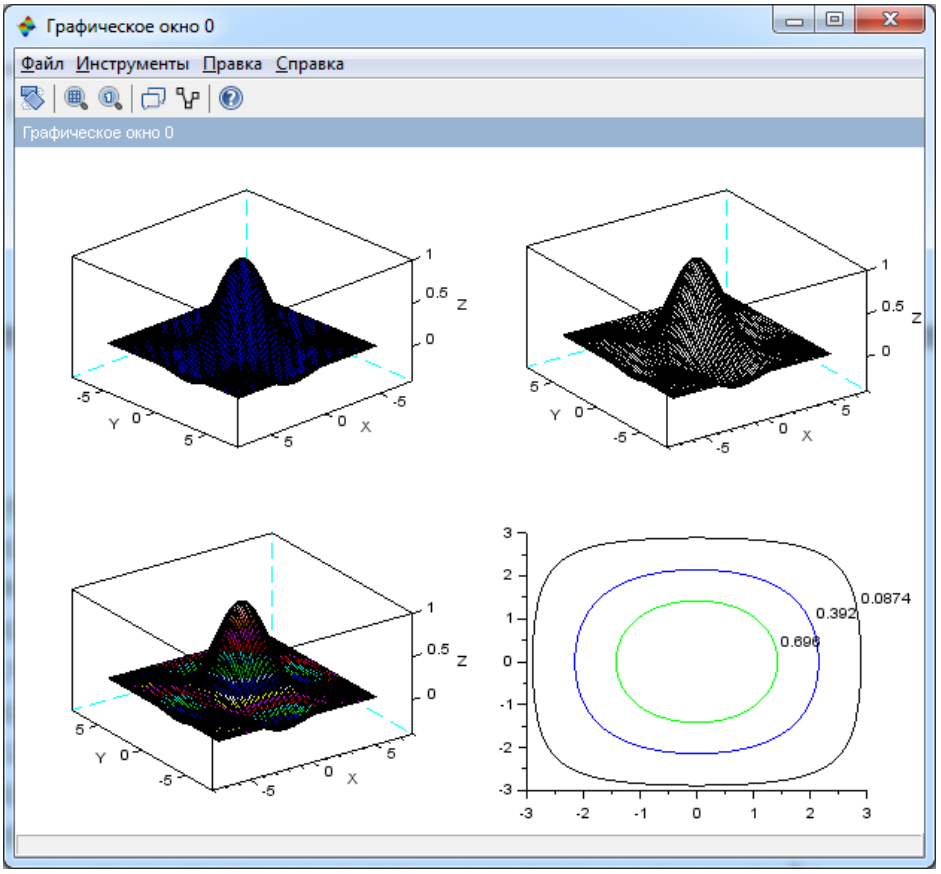


Рис. 1.21. Вывод 3D графиков в подокнах общего окна

Контрольные вопросы

1. Организация вложенных циклов.
2. Правила задания многомерных функций.
3. Связь двумерной функции с матрицей для вывода графиков.
4. Трехмерная графика в аксонометрии.

5. Функция param3d.
6. Функция param3d1.
7. Правила задания многомерных функций.
8. Связь двумерной функции с матрицей для вывода графиков.
9. Трехмерная графика в аксонометрии.
10. Трехмерная графика с функциональной раскраской.
11. Контурная графика.

Тема: Эволюционное (генетическое) программирование. Инструментальные средства разработки СИИ

Лабораторная работа № 3

Решение уравнений

Решение системы из N линейных уравнений в Scilab производится с использованием матричного деления. Результат решения системы линейных уравнений выводится в командное окно. Числа представляются в формате с плавающей запятой.

Задание 1. Решение системы линейных уравнений.

- Создать программу решения системы из N линейных уравнений в редакторе Scipad.
- Задать матрицу A постоянных коэффициентов системы линейных уравнений. Элементы матрицы A определяются умножением матрицы A примера 1 на номер варианта, заданный преподавателем. Постоянные коэффициенты размещаются по столбцам матрицы. Число строк в матрице равно N.
- Задать вектор правой части B размером N.
- Найти результат по формуле X=B/A.
- Проверить ответ по формуле B1=X*A. В результате проверки должно получиться B1=B.

Пример 1

Листинг программы

```
// Решение системы линейных уравнений
// Матрица коэффициентов
A=[1,4;2,3]
// Вектор правой части
B=[10,20]
// Решение
X=B/A
// Проверка
B1=X*A
```

```
-->// Решение системы линейных уравнений

-->// Матрица коэффициентов

-->A=[1,4;2,3]
 A  =

    1.    4.
    2.    3.

-->// Вектор правой части

-->B=[10,20]
 B  =

    10.    20.

-->// Решение

-->X=B/A
 X  =

    2.    4.

-->// Проверка

-->B1=X*A
 B1  =

    10.    20.
```

Рис. 1.22. Вывод результатов решения системы линейных уравнений

Решение нелинейного уравнения вида **f(x)=0** в Scilab можно реализовать с помощью функции **fsolve**. Метод решения основан на поиске корня в окрестности предполагаемого его значения **x0**. Для его определения проводится локализация решений по предварительно построенному графику **f(x)**. Результаты решения могут выводиться как в командное окно, так и в строку заголовка графика.

Задание 2. Решение нелинейного уравнения.

- Создать программу решения нелинейного уравнений в редакторе Scipad.
- В программе определить функцию **f1(x)**.
- Вывести **y1=f1(x)** в виде XY графика и по нему определить приближенно корни уравнения **y1(x)=0**. Если корни на графике не просматриваются, то необходимо подобрать пределы изменения аргумента и повторить операции.
- Для каждого корня найти точное значение, используя функцию **fsolve**. Перед расчетами задать приближенное значение корня **x0**.
- Сформировать строку с результатами и вывести ее в заголовок окна графика.

Задание 3. Решение системы из двух нелинейных уравнений.

- Создать программу решения нелинейных уравнений в редакторе Scipad.
- В программе определить функции **f1(x), f2(x), f3(x)=f2(x)-f1(x)**.
- Вывести y3=f1(x) в виде XY графика. По нему определить приближенно корни уравнения y3(x)=0. Если корни на графике не просматриваются, то изменить пределы изменения аргумента и повторить операции.
- Для каждого корня найти точное значение, используя функцию fsolve. Перед расчетами задать приближенное значение корня x0.
- Сформировать строку с результатами и вывести ее в заголовок окна графика.

Варианты для заданий 2 и 3

№	**f1(x)** - полином 3-й степени с коэффициентами **a**				**f2(x)**
	a3	**a2**	**a1**	**a0**	
1	0	-1	3	-2	0.1exp(x) - 80
2	0	4	-2	-5	20|cos(x)|
3	0	2	4	-1	10ln(x + 2.5)
4	0	9	-8	-70	40|sin(x)|
5	0	-4	4	50	30cos(x)
6	0.1	-5	3	40	2exp(|0.01*x|) - 50
7	0.2	-3	2	30	10sin(4x)
8	0.3	-2	1	50	exp(|x|)sin(4x)
9	0.4	6	-7	-70	90|sin(x)|
10	0.5	-2	4	50	50cos(x)
11	0.1	-7	8	40	10exp(|0.1*x|) - 90
12	0.2	-5	5	30	20sin(3x)

При решении нелинейного уравнения задания 2 оно формируется из таблицы вариантов как **f1(x)=0**. При решении системы из двух нелинейных уравнений задания 3 из таблицы вариантов формируется уравнение **f3(x) = f1(x) - f2(x) = 0**.

Локализация корней. Уравнение или система уравнений может иметь несколько корней, каждый из которых ищется отдельно. При этом для каждого корня необходимо задавать диапазон аргумента, в котором расположен именно этот корень.

Это осуществляется с помощью локализации корня. Для этого нужно вычислить значения функций в заданном интервале, а затем построить их графики. Начальное значение для решения одного уравнения – точка пересечения графиком функции оси X. График выво-

дится процедурой, в которой аргументы – переменная **x** и анализируемая функция. Рекомендуется включать координатную сетку:

```
plot(x,f1(x)); xgrid();
```

Начальное значение для решения системы из двух уравнений – это точка взаимного пересечения графиков этих функций. Графики выводятся процедурой, в которой для каждого графика следует группа параметров:

```
plot(x,f(x),x,f2(x)); xgrid();
```

Функция **fsolve**. Эта функция используется для нахождения корня нелинейного уравнения по его приближенному значению, и она имеет следующий формат записи: **<имя результата>=fsolve(x0, f1)**

Пример 1

Листинг программы

```
// Решение нелинейного уравнения
function y1=f1(x); // Функция f1
y1=x+3*(x-1)^2-2;
endfunction
x=0:0.1:2;
plot(x,f1(x)); xgrid; // Графики
x0=0.2;
x1=fsolve(x0,f1) // Корень 1
x0=1.4;
x2=fsolve(x0,f1)  // Корень 2
rezult='x1 = '+string(x1)+' x2 = '+string(x2);
title(rezult);
```

В этой программе осуществляется поиск двух корней нелинейного уравнения. Приближенные значения корней 0.2 и 1.4 определены по предварительно построенному графику нелинейной функции. Точные значения корней выводятся в окне графика функции.

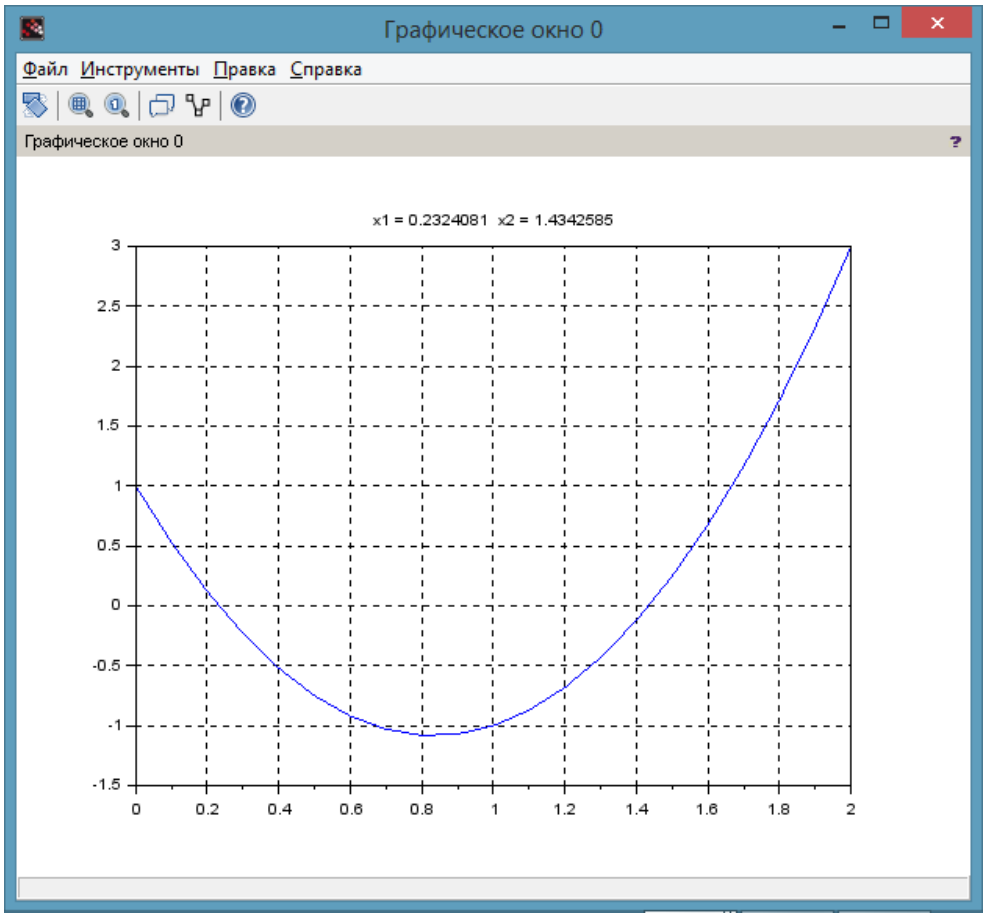


Рис. 1.23. Вывод результатов решения нелинейного уравнения

Пример 2

Листинг программы

```
// Решение системы нелинейных уравнений
function y1=f1(x);  // Функция f1
y1=x+3*(x-1)^2-2;
endfunction
function y2=f2(x); // Функция f2
y2=x-1;
endfunction
function y3=f3(x); // Функция f3
y3=f1(x)-f2(x);
endfunction
x=0:0.1:2;
plot(x,f1(x),x,f2(x)); xgrid; // Графики
x0=0.4;
x1=fsolve(x0,f3)  // Корень 1
```

```
x0=1.5;
x2=fsolve(x0,f3)  // Корень 2
rezult='x1 = '+string(x1)+' x2 = '+string(x2); title(rezult);
```

Программа осуществляет поиск двух корней системы из двух нелинейных уравнений. Приближенные значения корней 0.4 и 1.5 определены графически. Точный результат приведен в графическом окне.

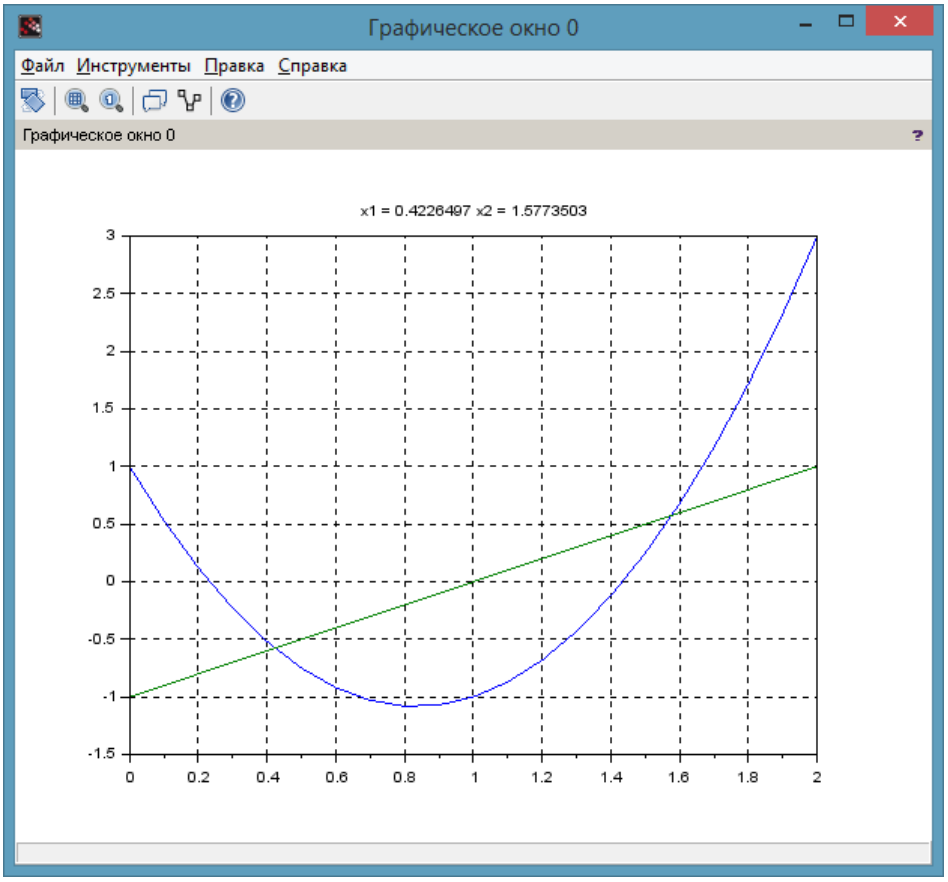


Рис. 1.24. Вывод результатов решения двух нелинейных уравнений

Контрольные вопросы

1. Методы решения системы линейных уравнений.
2. Ввод коэффициентов и вывод полученного решения.
3. Задание функции пользователя.
4. Локализация решений уравнения.
5. Решение нелинейного уравнения с помощью функции **fsolve**.
6. Локализация решений системы из двух уравнений.
7. Решение системы из двух уравнений.

Тема: Извлечение знаний. Концептуализация проблемной области. . Представление знаний с помощью логики предикатов.

Лабораторная работа № 4
НЕЧЕТКИЕ МНОЖЕСТВА

Функции принадлежности и базовые операции

Для работы с нечеткой логикой (НЛ) в математическом пакете Scilab предназначен специальный модуль Fuzzy Toolbox, в состав которого входит редактор SciFLT.

Данный модуль содержит в своем составе более десяти встроенных типов *функций принадлежности*, формируемых на основе кусочно-линейных функций, распределения Гаусса, сигмоидной кривой, квадратических и кубических полиномиальных кривых.

Линейные функции принадлежности являются наиболее простыми, что является их несомненным достоинством. Этих функций принадлежности в модуле Fuzzy Toolbox всего лишь две: треугольная и трапециевидная.

Треугольная функция принадлежности – **trimf** (от английских слов **tri**angle **m**embership **f**unction) в параметрическом виде представляет собой не что иное, как набор трех точек, соответствующих вершинам. С помощью этого набора трех точек можно образовывать различные виды треугольников.

Описание функции:

y = trimf(x, [a b c]).

Вектор **x** в данном выражении представляет базовое множество, на котором определяется функция принадлежности. Аргументы в квадратных скобках **a** и **c** определяют положение основания треугольника, а **b** – его вершину.

На рисунке 2.1 представлены примеры трех различных треугольных функций принадлежности, построенных на основе примера 1.

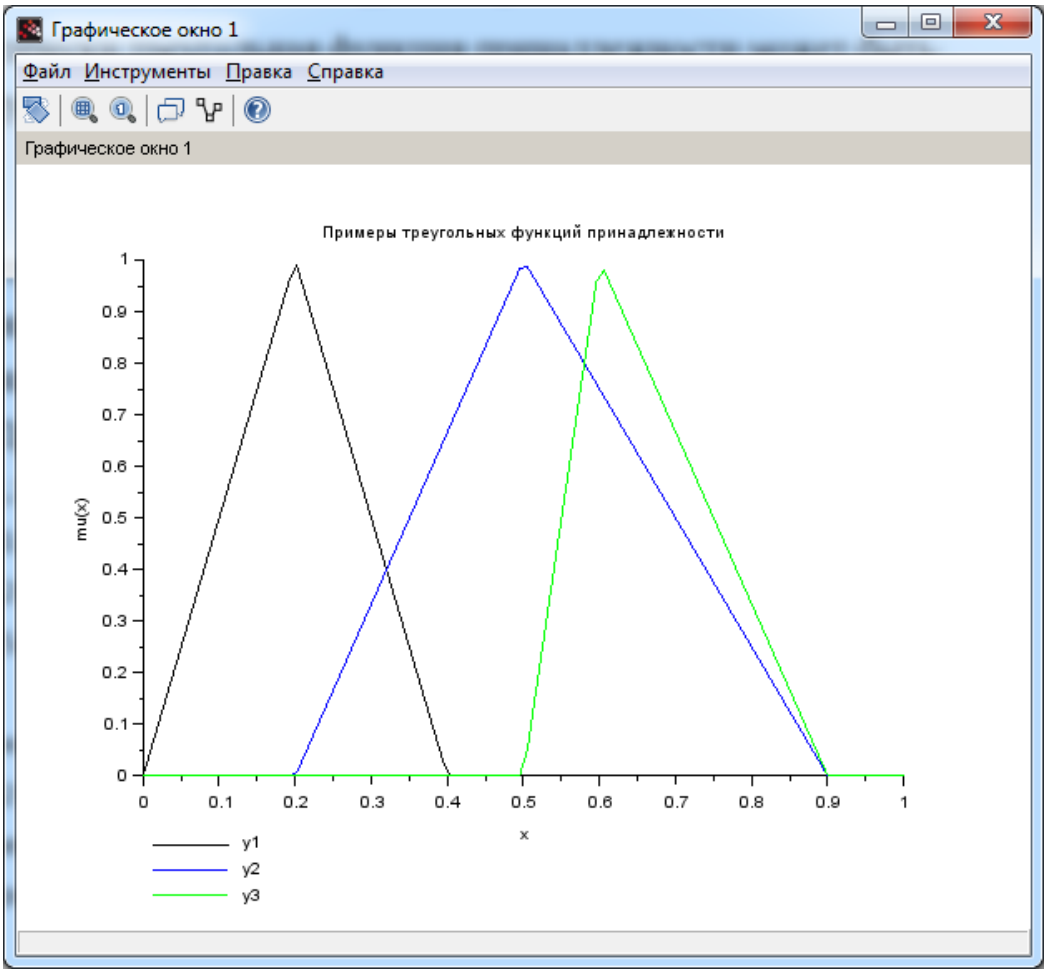


Рис. 2.1. Треугольные функции принадлежности

Аналитически треугольная функция принадлежности может быть задана следующим образом:

$$f(x,a,b,c) = \begin{cases} 0, & x < a \\ \dfrac{x-a}{b-a}, & a \le x \le b \\ \dfrac{c-x}{c-b} & b \le x \le c \\ 0, & x > c \end{cases}$$

Рассмотрим примеры использования различных функций принадлежности, входящих в состав модуля нечеткой логики.

Пример 1. Программа использования **trimf** (результат на рис.2.1)

```
x=linspace(0,1,100)';          //задаем множество x
y1=trimf(x,[0 0.2 0.4]);       //и три треугольных функции на нем
y2=trimf(x,[0.2 0.5 0.9]); y3=trimf(x,[0.5 0.6 0.9]);
```

```
scf(); clf(); plot2d(x,[y1 y2 y3],leg="y1@y2@y3");
xtitle("Примеры треугольных функций принадлежности","x","mu(x)");
```

Трапециевидная функция принадлежности – **trapmf** (от английских слов **trap**ezoid **m**embership **f**unction), отличается от треугольной функции лишь тем, что имеет верхнее основание.

Описание функции: y = trapmf(x, [a b c d]),

где аргументы **a** и **d** задают нижнее основание, **b** и **c** – верхнее основание трапеции (рис. 2.2).

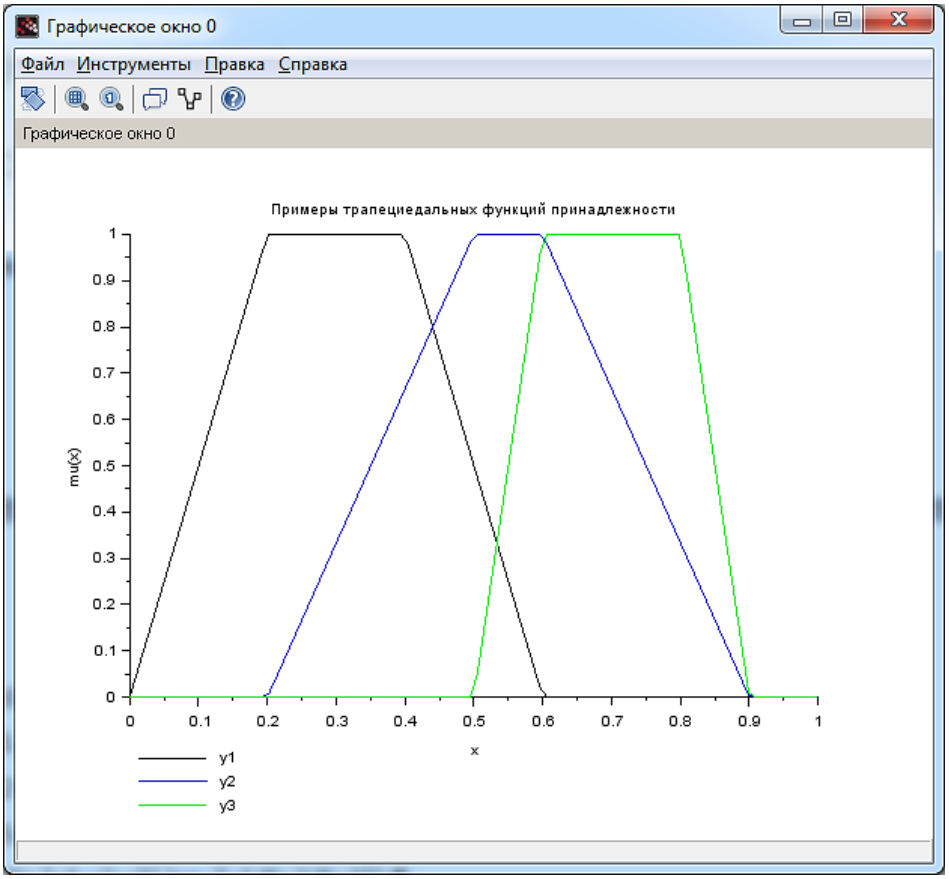


Рис. 2.2. Трапецеидальные функции принадлежности

Аналитически трапециевидную функцию записывают так:

$$f(x,a,b,c,d) = \begin{cases} 0, & x < a \\ \dfrac{x-a}{b-a}, & a \le x < b \\ 1, & b \le x \le c \\ \dfrac{d-x}{d-c}, & c < x \le d \\ 0, & x > d \end{cases}$$

Пример 2. Программа использования **trapmf** (результат на рис. 2.2)

```
x=linspace(0,1,100)';
y1=trapmf(x,[0 0.2 0.4 0.6]);
y2=trapmf(x,[0.2 0.5 0.6 0.9]);
y3=trapmf(x,[0.5 0.6 0.8 0.9 ]);
scf(); clf(); plot2d(x,[y1 y2 y3],leg="y1@y2@y3");
xtitle("Примеры трапецеидальных функций принадлежности","x","mu(x)");
```

На основе функции Гаусса в Scilab можно построить функции принадлежности двух видов: простую функцию Гаусса **gaussmf** и двухстороннюю функцию Гаусса **gauss2mf**, которая образована двумя функциями Гаусса с разными параметрами.

Описание простой функции Гаусса:

y = gaussmf(x, [σ , c]).

Простая функция Гаусса зависит только от 2-х параметров σ и **c**:

$$f(x, \sigma, c) = e^{\frac{-(x-c)^2}{2\sigma^2}}.$$

Двусторонняя функция принадлежности Гаусса:

y = gauss2mf(x,[σ1, c1, σ2, c2]).

Это выражение является комбинацией двух функций Гаусса с различными параметрами. Первая функция определяется параметрами σ**1** и **c1** и задает форму левой стороны, а вторая с параметрами σ**2** и **c2** задает форму правой стороны функции принадлежности.

Если **c1 < c2**, то функция **gauss2mf** достигает максимального значения, равного единице, в противном случае максимальное значение функции будет меньше единицы.

На рис. 2.3 представлены графики кривых **gaussmf** и **gauss2mf**, которые задаются в программе из примера 3.

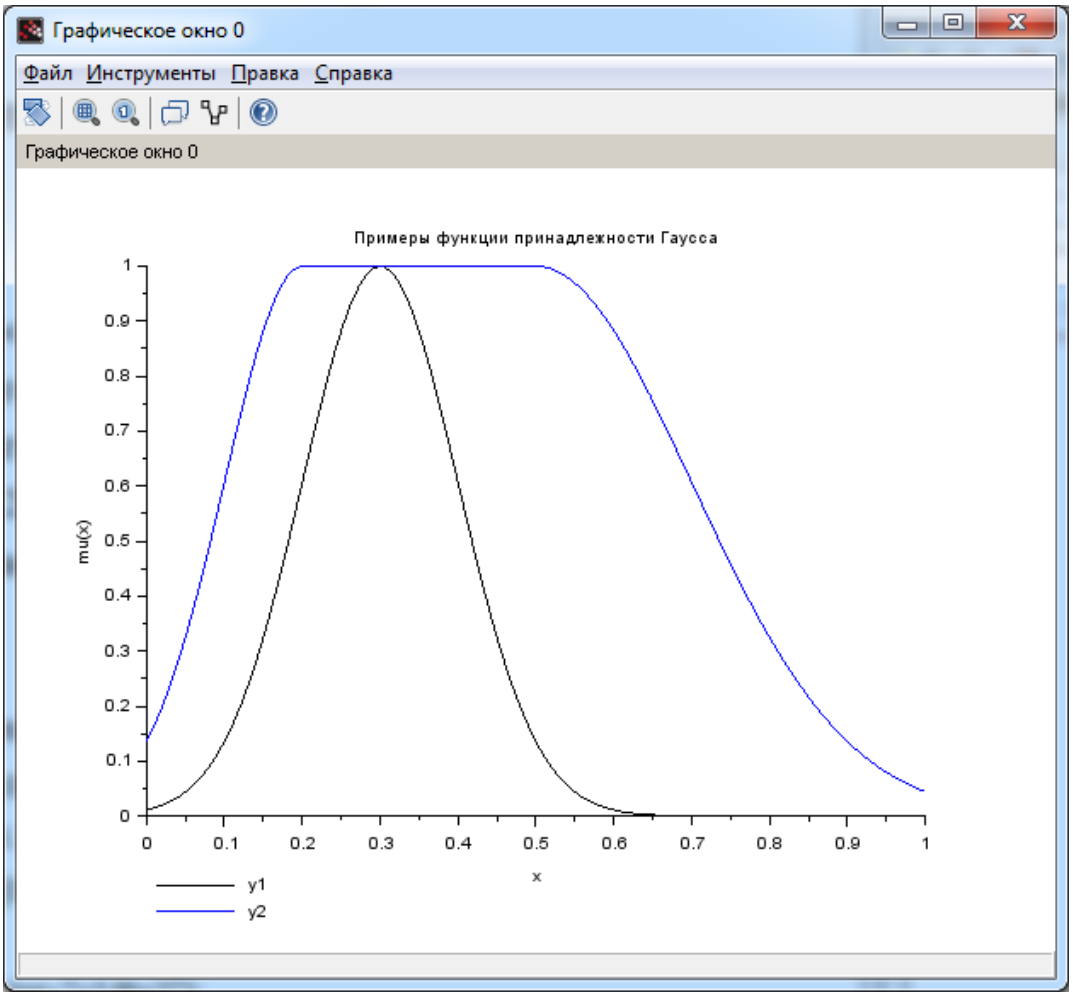


Рис. 2.3. Гауссовы кривые

Пример 3. Программа использования функции **gaussmf**

```
x=linspace(0,1,100)';
y1=gaussmf(x,[0.1 0.3]);
y2=gauss2mf(x,[0.1 0.2 0.2 0.5]);
scf(); clf(); plot2d(x,[y1 y2],leg="y1@y2");
xtitle("Примеры функции принадлежности Гаусса","x","mu(x)");
```

В первой строке определения базового множества **x** используется символ « ' », показывающий, что это множество транспонировано.

Следующая функция называется «обобщенный колокол» и она позволяет представлять нечеткие субъективные предпочтения. Ее особенность – наличие третьего параметра, который позволяет сделать плавный переход от нечеткого множества к четкому.

Описание функции:

y = gbellmf(x, [a b c]).

Аналитическое выражение для функции «обобщенный колокол»:

$$f(x, a, b, c) = \frac{1}{1 + \left|\frac{x - c}{a}\right|^{2b}},$$

где параметр ***c*** определяет расположение центра функции принадлежности, параметры ***a*** и ***b*** определяют форму кривой (рис. 2.4).

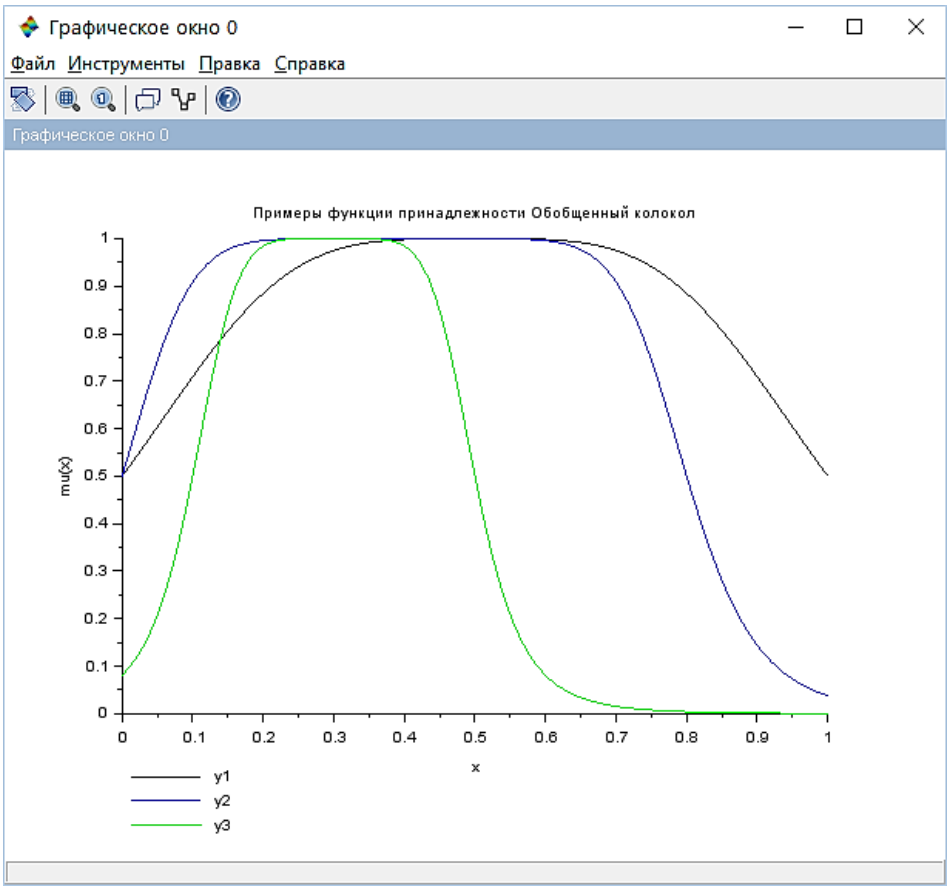


Рис. 2.4. Функции принадлежности «Обобщенный колокол»

Пример 4. Программа использования **gbellmf**

```
x=linspace(0,1,100)';
y1=gbellmf(x,[0.5 2 0.5]);
y2=gbellmf(x,[0.4 4 0.4]);
y3=gbellmf(x,[0.2 3 0.3]);
plot2d(x,[y1 y2 y3],leg="y1@y2@y3");
xtitle("Примеры функции принадлежности Обобщенный колокол","x","mu(x)");
```

Функции принадлежности Гаусса и «обобщенный колокол» отличаются гладкостью и простотой записи, а также являются наиболее используемыми при описании нечетких множеств.

Однако они не позволяют формировать асимметричные функции. Для этих целей используют сигмоидные функции, которые могут быть открыты как слева, так и справа.

Симметричные и закрытые функции синтезируют с помощью двух дополнительных сигмоид. Основная сигмоидная функция обозначается **sigmf**, а дополнительные – **dsigmf** и **psigmf**.

Описание основной сигмоидной функции:

y = sigmf(*x*, [*a c*]).

Аналитическая запись сигмоидной функции **sigmf** имеет вид:

$$f(x, a, c) = \frac{1}{1 + e^{-a(x-c)}}.$$

В зависимости от знака параметра ***a*** функция будет открыта справа или слева (см. рис. 2.5), что позволит применять ее при описании нечетких понятий «очень большой», «крайне малый» и т. д.

Описание дополнительной сигмоидной функции:

y = dsigmf(*x*, [a_1 c_1 a_2 c_2]).

ФП **dsigmf** зависит от четырех параметров $\boldsymbol{a_1}$, $\boldsymbol{c_1}$, $\boldsymbol{a_2}$, $\boldsymbol{c_2}$ и определяется разностью двух сигмоидных функций: $f_1(x, a_1, c_1) - f_2(x, a_2, c_2)$.

Описание дополнительной сигмоидной функции:

y = psigmf(*x*,[a_1 c_1 a_2 c_2]).

ФП **psigmf** также, как и предыдущая функция, зависит от четырех параметров $\boldsymbol{a_1}$, $\boldsymbol{c_1}$, $\boldsymbol{a_2}$, $\boldsymbol{c_2}$ и определяется как произведение двух сигмоидных функций: $f_1(x, a_1, c_1) \cdot f_2(x, a_2, c_2)$.

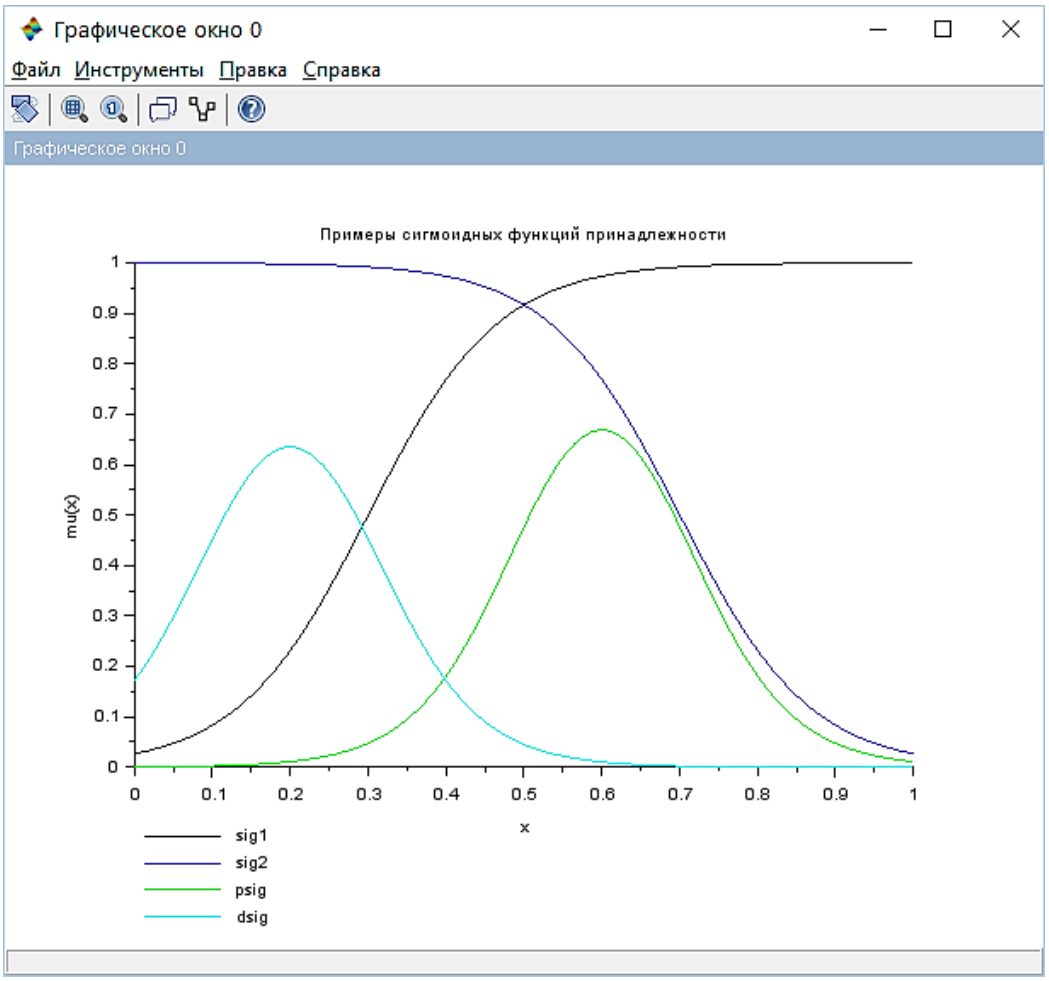


Рис. 2.5. Сигмоидные функции принадлежности

Пример 5. Программа использования сигмоидных функций

```
x=linspace(0,1,100)';
sig1=sigmf(x,[12 0.3]);
sig2=sigmf(x,[ -12 0.7]);
psig=psigmf(x,[15 0.5 -15 0.7]);
dsig=dsigmf(x,[15 0.1 15 0.3]);
plot2d(x,[sig1 sig2 psig dsig],leg="sig1@sig2@psig@dsig");
xtitle("Примеры сигмоидных функций принадлежности","x","mu(x)");
```

Можно формировать ФП на основе полиномиальных кривых. Это Z-функции (**zmf**), PI-функции (**pimf**) и S-функции (**smf**). Функция **zmf** – асимметричная кривая, открытая слева, функция **smf** – ее зеркальное отображение. Функция **pimf** равна нулю справа и слева и равна единице в середине некоторого отрезка (см. рис. 2.6).

Описание функции:

y = zmf(*x*, [*a b*]).

Параметры ***a*** и ***b*** определяют экстремальные значения кривой.

Описание функции:

y = pimf(*x*, [*a b c d*]).

Параметры ***a*** и ***d*** задают переход функции в нулевое значение, параметры ***b*** и ***c*** – в единичное.

Описание функции:

y = smf(*x*, [*a b*]).

Параметры ***a*** и ***b*** определяют экстремальные значения кривой.

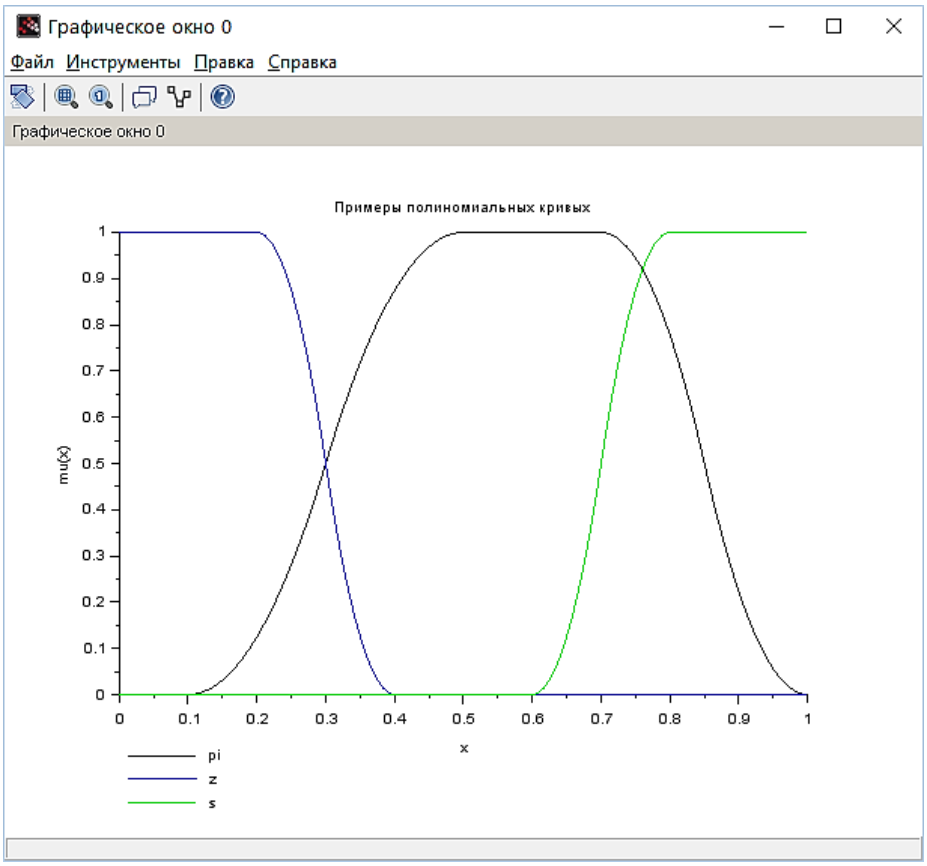


Рис. 2.6. Полиномиальные кривые

Пример 6. Программа использования полиноминальных кривых

```
x=linspace(0,1,100)';
pi=pimf(x,[0.1 0.5 0.7 1.0]);
z=zmf(x,[0.2 0.4]);
s=smf(x,[0.6 0.8]);
```

```
plot2d(x,[pi z s],leg="pi@z@s");
xtitle("Примеры полиномиальных кривых","x","mu(x)");
```

Кроме рассмотренных выше функций, позволяющих представлять нечеткие множества, в SciFLT можно формировать собственные функции принадлежности, а также модифицировать встроенные.

Среди базовых операций в модуле SciFLT выделяют три логические операции: конъюнкцию, дизъюнкцию и логическое отрицание. При этом существует возможность определять конъюнктивные и дизъюнктивные операторы различными методами.

Конъюнкция, или Т-норма, представляет собой нахождение логического И. Она представлена следующими операторами:

$$tnorm([x1,x2],class,w)=\begin{cases}\dfrac{x1x2}{\max(x1,x2,w} & \text{if } class \text{ is } dubois\\ 1-min\left[1,((1-x1)^{w}+(1-x2)^{w})^{1/w}\right] & \text{if } class \text{ is } yager\\ x1 \text{ if } x2=1 & \\ \{x2 \text{ if } x1=1 & \text{if } class \text{ is } dprod\\ 0\ otherwise & \\ \dfrac{x1x2}{2-(x1+x2-x1x2)} & \text{if } class \text{ is } eprod\\ x1x2 & \text{if } class \text{ is } aprod\\ \min(x1,x2) & \text{if } class \text{ is } min\end{cases}$$

Определение функции:

```
y = tnorm(x, class [,class_par]),
```

где **x** – матрица, размерностью [m,n]; **y** – матрица, размерностью [m,1]; **class** – строка, которая задает вид Т-нормы (принимает значения: «dubois» для Т-норм Дюбуа-Прада, «yager» для Т-норм Ягера, «dprod» для вероятностного И, «eprod» для произведения Эйнштейна, «aprod» для алгебраического произведения и «min» для операции минимума); **class_par** – скалярная величина для Т-норм «dubois» и «yager».

Дизъюнкция или S-конорма представляет собой логическое ИЛИ и может быть найдена с помощью следующих операторов:

$$snorm([x1,x2],class,w)=\begin{cases}\dfrac{x1+x2-x1x2-\min(x1,x2,1-w)}{\max(1-x1,1-x2,w} & \text{if } class \text{ is } dubo\text{i}\\ \min[1,(x1^{w}+x2^{w})^{1/w}] & \text{if } class \text{ is } yager\\ x1 \text{ if } x2=0 & \\ x2 \text{ if } x1=0 & \text{if } class \text{ is } dsum\\ 1 \ otherwise & \\ \dfrac{x1+x2}{1+x1x2} & \text{if } class \text{ is } esum\\ x1+x2-x1x2 & \text{if } class \text{ is } asum\\ \max(x1,x2) & \text{if } class \text{ is } max\end{cases}$$

Определение функции:

y = snorm(x, class [,class_par]),

где **x** – матрица, размерностью [m,n]; **y** – матрица, размерностью [m,1]; **class** – строка, задает вид S-конормы (принимает значения: «dubois» для S-конормы Дюбуа-Прада, «yager» для S-конормы Ягера, «dsum» для вероятностного ИЛИ, «esum» для суммы Эйнштейна, «asum» для алгебраической суммы и «max» для операции нахождения максимума; **class_par** – скалярная величина для S-конорм «dubois» и «yager».

Пример 7. Программа использования операций min и max

```
x=[0:0.1:10]';
y1=gaussmf(x,[1.2 3]);
y2=gaussmf(x,[1 7]);
yy1=tnorm([y1 y2],'min');
yy2=snorm([y1 y2],'max');
yy3=tnorm([y1 y2],'dprod');
yy4=snorm([y1 y2],'dsum');
subplot(3,1,1);
plot2d(x,[y1 y2],leg='mf1@mf2',rect=[0 -0.1 10 1.1]);
xtitle('Member Function Evaluation','x','mu(x)'); subplot(3,1,2);
plot2d(x,[yy1 yy3],leg='min@dprod',rect=[0 -0.1 10 1.1]);
```

```
xtitle('AND OPERATION','x','and(mf1,mf2)'); subplot(3,1,3);
plot2d(x,[yy2 yy4],leg='max@dsum',rect=[0 -0.1 10 1.1]);
xtitle('OR OPERATION','x','or(mf1,mf2)');
```

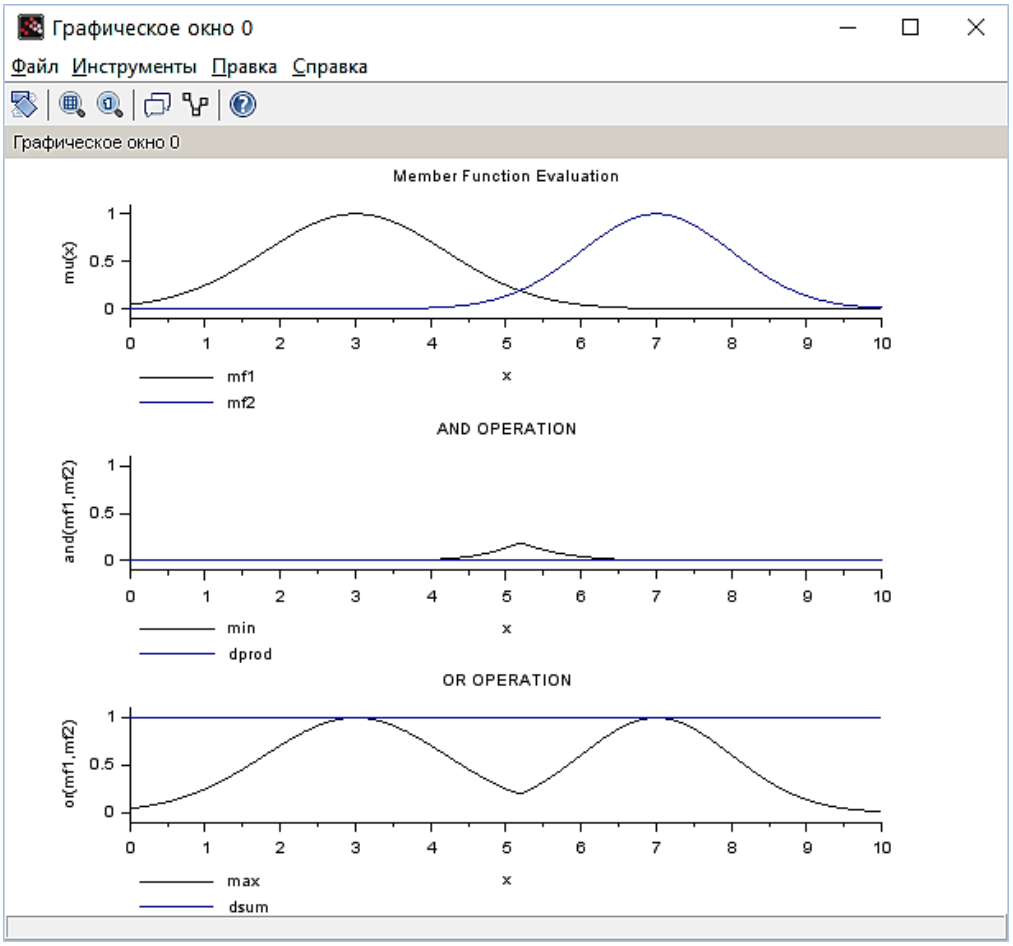


Рис. 2.7. Результаты работы программы примера 7

Минимаксная интерпретация наиболее часто используется при построении нечетких систем. Однако на практике также распространена и альтернативная вероятностная интерпретация конъюнктивных и дизъюнктивных операторов.

Операция «дополнение» есть не что иное, как математическое представление вербального выражения “НЕ А”, где А – нечеткое множество, описывающее некоторое размытое суждение.

Описание операции дополнения:

y=complement(x, class [,class_par]),

где **x**, **y** – матрицы, размерностью [m,n];

class – строка вида оператора (значения: «one» для обычного дополнения, «sugeno» для дополнения по Сугено и «yager» для Ягера);
class_par – скаляр, используемый в S-конормах «sugeno» и «yager».

Аналитически функцию дополнения записывают так:

$$complement(x, class, w) = \begin{cases} 1 - x & \text{if } class \text{ is } on \\ \dfrac{1-x}{1+wx} & \text{if } class \text{ is } sugeno \\ (1 - x^{w})^{1/w} & \text{if } class \text{is } yager \end{cases}$$

Пример 8. Программа использования операции дополнения

```
x=[0:0.1:10]';
y1=gaussmf(x,[1.2 3]);
y2=complement(y1,”one”);
plot2d(x,[y1 y2],leg='mf1@Not_mf1',rect=[0 -0.1 10 1.1]);
xtitle('Member Function and Inverse','x','mu(x)');
```

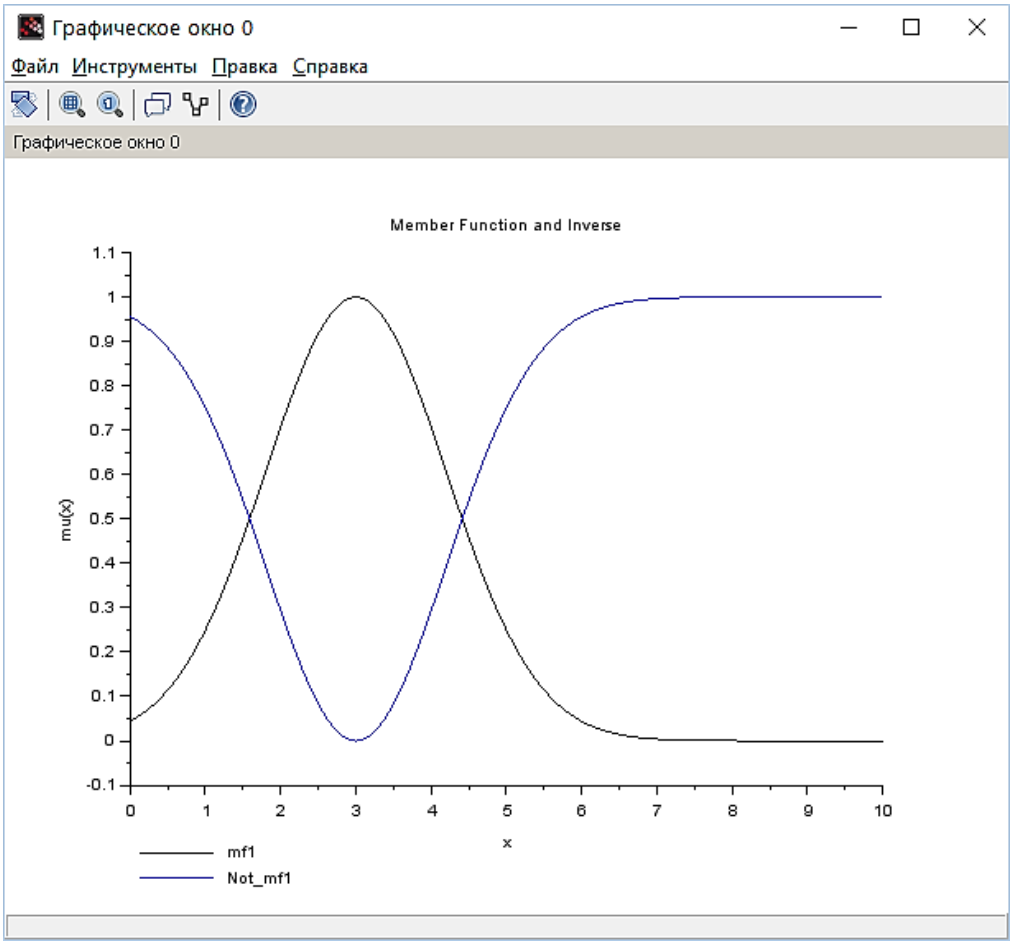


Рис. 2.8. Функция дополнения

Контрольные вопросы

1. Нечеткое множество и его отличие от обычного множества.
2. Функция принадлежности и ее свойства.
3. Конъюнктивные и дизъюнктивные операторы.

Тема: Индуктивный метод приобретения знаний. Использование графических средств ввода-вывода.

Лабораторная работа № 5

Моделирование нечетких систем

Цель работы: изучить метод построения нечеткой системы в модуле SciFLT среды Scilab.

В SciFLT реализованы все основные функции, необходимые для создания и модифицирования систем нечеткого вывода.

Перед построением системы нечеткого вывода необходимо вначале в командной строке основного окна Scilab набрать команду ***editfls***. С помощью этой команды вызывается графический редактор систем нечеткого вывода, как показано на рисунке 2.9.

Рис. 2.9. Графический редактор систем нечеткого вывода (СНВ)

Редактор CHB. Построение нечетких систем

В меню нужно выбрать **File → New fls → тип** системы (Мамдани или Сугено). Выбираем систему нечеткого вывода по **Мамдани**, при этом в левой панели появится дерево создаваемой нечеткой системы.

Сначала выбираем в нем пункт **Description** и заполняем появившуюся справа форму. В ней необходимо указать имя системы, комментарии (не обязательно), виды T-нормы и S-конормы, тип операторов дополнения, импликации, агрегации и метод дефаззификации.

Для системы по Мамдани выберите операторы (см. рис. 2.10).

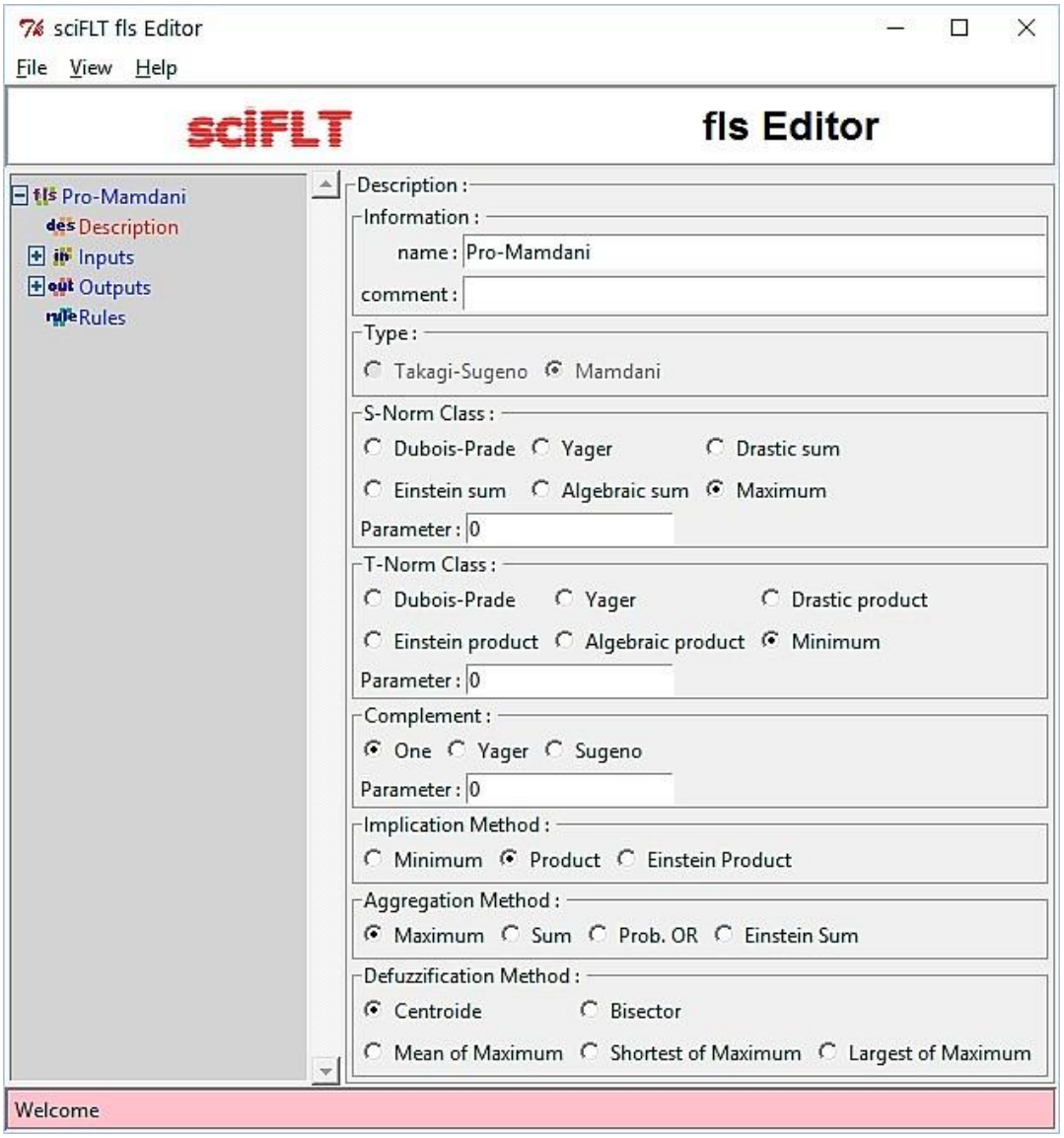


Рис. 2.10. Описание выбранных операторов

Затем определяем состав входных и выходных переменных. Для этого выбираем пункты **Inputs** и **Outputs** соответственно. В окне справа появится число переменных, их список, а также кнопки **Add**, **Delete** и **Exit**. Добавляем новые переменные кнопкой **Add**.

Сохранение проектируемой системы в рабочее пространство среды Scilab осуществляют командой **File → Export → to workspace**. При этом данные сохраняются до окончания сеанса работы с Scilab. Для сохранения данных на диск после окончания сеанса работы применяется соответствующий пункт того же меню – **to fls file**….

Редактор переменных

Для редактирования переменных (переименования, задания диапазона значений, связывания функциями принадлежности) необходимо выбрать в списке входных или выходных переменных нужную.

Тогда в правом окне появится редактор переменной:

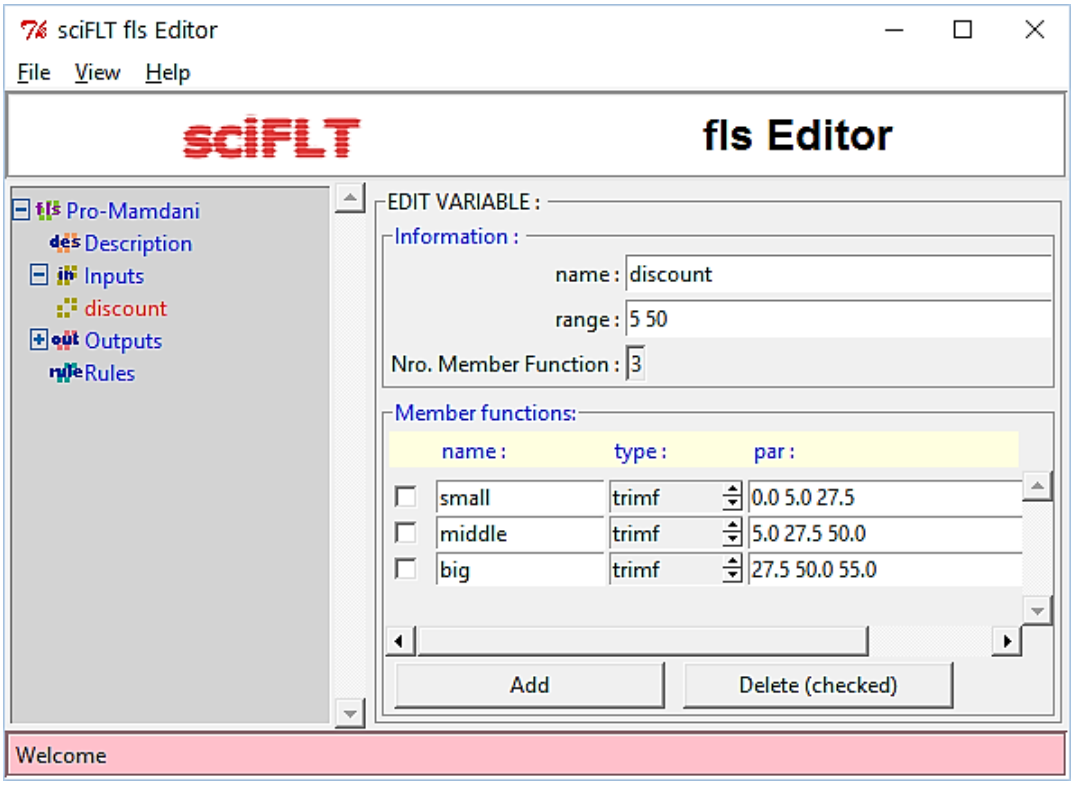


Рис. 2.11. Редактор переменных (входная переменная discount)

1) в редакторе нужно задать имя;
2) диапазон допустимых значений;
3) с помощью кнопки **Add** внизу окна добавить функции принадлежности (удаляют их нажатием кнопки **Delete**).

Редактор переменных – инструмент, позволяющий отображать и редактировать любые ФП, ассоциированные (связанные) со всеми входными и выходными переменными разрабатываемой СНВ.

Редактируют ФП текущей переменной изменяя ее характеристики (имя, тип и числовые параметры). То есть при построении СНВ нужно с помощью редактора ФП определить соответствующие функции для каждой из входных и выходных переменных.

Редактор правил вывода

После указания количества входных и выходных переменных, определения их наименования и выбора соответствующих ФП, в СНВ включают правила вывода с помощью пункта **Rules**.

Основываясь на описаниях входных и выходных переменных, определенных в редакторе ФП, редактор правил вывода формирует структуру правила автоматически. От пользователя требуется лишь связать значения входных и выходных переменных, выбирая из списка заданных ранее ФП, и определить логические связки между ними. Также допускается использование логического отрицания (НЕ) и изменение весов правил в диапазоне от 0 до 1.

Правила вывода отображаются в окне в следующей форме:

*if (input_*1 *is [not] mf_*1j_1*) < and, or >... (input_i is [not] mf_ij*$_i$*) ... <and, or > (input_n is [not]mf_nj*$_n$*) then (output_*1 *is [not]mf_n+*1j_{n+1}*) < and, or > ... (output_k is [not]mf_k+nj*$_{k+n}$*) <and,or>...(output_m is [not]mf_m+nj*$_{m+n}$*)(w),*

где ***i*** – номер входной переменной; j_i – номер ФП *i*-й переменной; ***k*** – номер выходной переменной; ***n*** – количество входных переменных; ***m*** – количество выходных переменных; ***w*** – вес правила. Круглые скобки заключают обязательные параметры, квадратные – необязательные, а угловые – альтернативные (один на выбор).

Пример 1. *Создание СНВ*

В качестве примера, иллюстрирующего метод построения СНВ, рассмотрим следующую ситуацию. Необходимо оценить степень инвестиционной привлекательности конкретного бизнес-проекта на основании данных о ставке дисконтирования и периоде окупаемости.

Шаг 1. Вызываем редактор для создания СНВ, набирая в командной строке ***editfls***. В появившемся окне редактора создаем новую систему по Мамдани. Заполняем описание как показано на рис. 2.10. Добавляем 2 входные переменные и 1 выходную переменную.

В результате получаем следующую структуру СНВ: два входа, механизм нечеткого вывода по Мамдани, один выход. Объявляем первую переменную как *discount*, а вторую – *period*, которые, соответственно, будут представлять ставку дисконтирования и период окупаемости бизнес-проекта. Наименование выходной переменной, на основании которой принимается решение о степени инвестиционной привлекательности бизнес-проекта, задается как *rate*. Сохраним создаваемую модель под именем *Invest*. На рис. 2.13 представлено состояние окна редактора СНВ после сохранения.

Шаг 2. Каждой входной и выходной переменной поставим в соответствие набор ФП. Для *discount* определяем диапазон базовой переменной от 5 до 50 (единица измерения – проценты). Такой же диапазон выбираем для ее отображения. Добавим три ФП, тип которых – *trimf*, и присвоим им наименования – *small*, *middle*, *big*, соответственно, небольшой, средней и большой ставке дисконтирования (рис. 2.2).

Переменной *period* диапазон базовой переменной определен равным [3, 36] (единица измерения – месяцы), поставлены в соответствие три ФП типа *gaussmf* с наименованиями – *short*, *normal*, *long*. Таким образом, переменная срока окупаемости бизнес-проекта будет принимать следующие значения: короткий, обычный и длительный срок окупаемости (рис. 2.12).

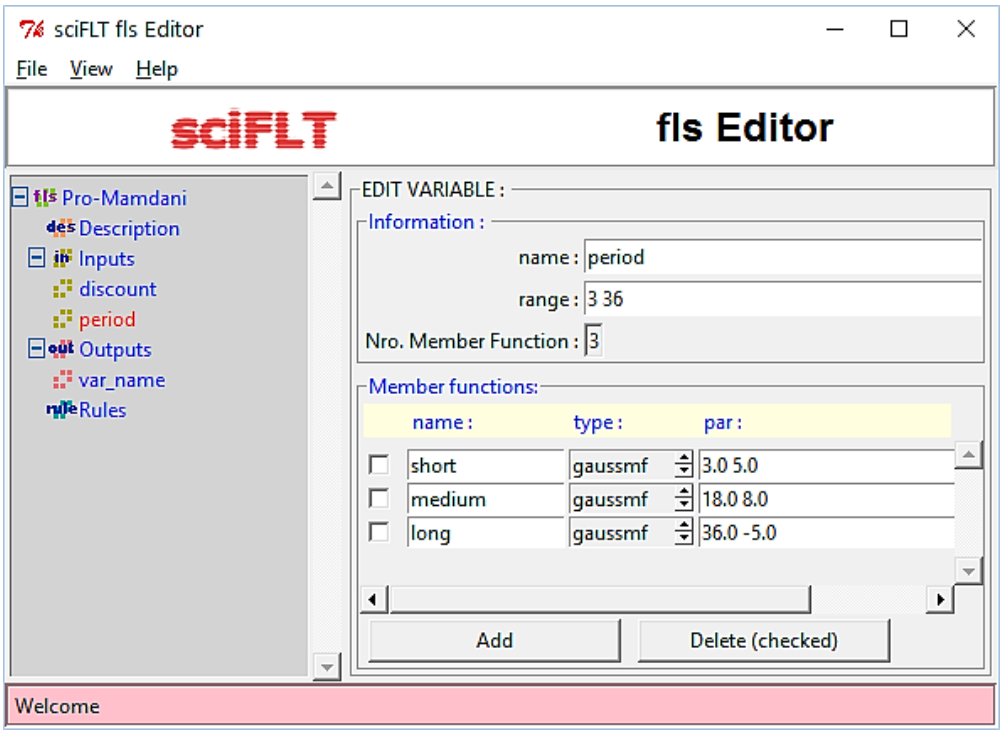


Рис. 2.12. Входная переменная Period

Наконец, для переменной *rate* определяем: базовая переменная изменяет значение в диапазоне [0, 1], семантика описывается тремя ФП типа *trimf* с наименованиями: *bad*, *normal*, *good* (рис. 2.13).

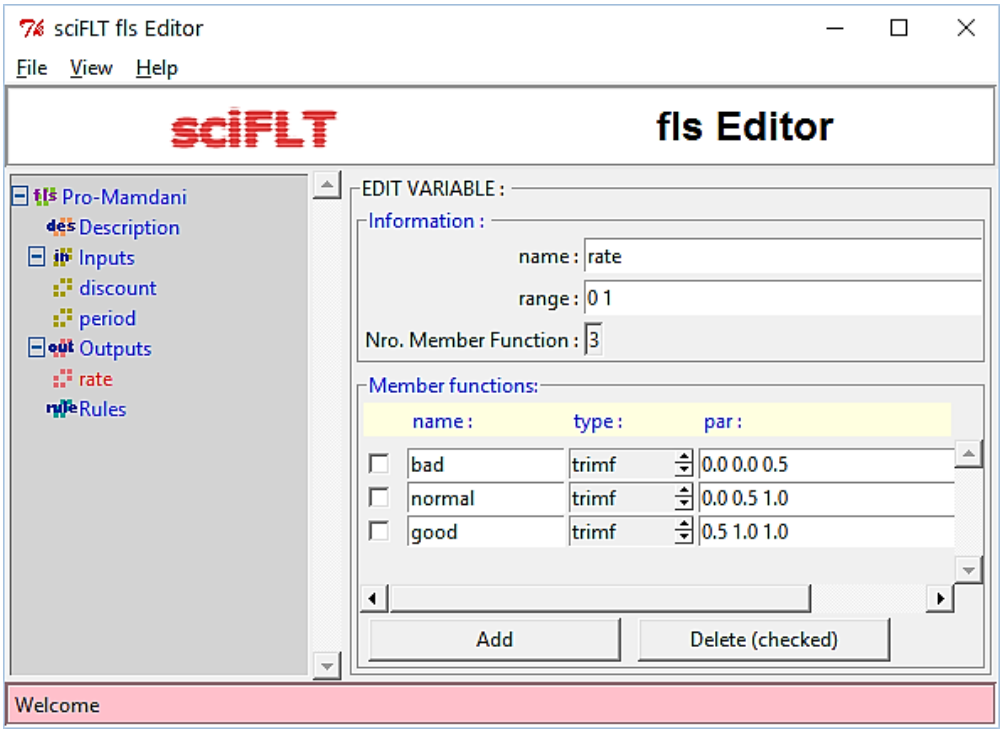


Рис. 2.13. Выходная переменная Rate

В графическом виде переменные представлены на рис. 2.14 и 2.15 (меню **View → Plot Current Var**).

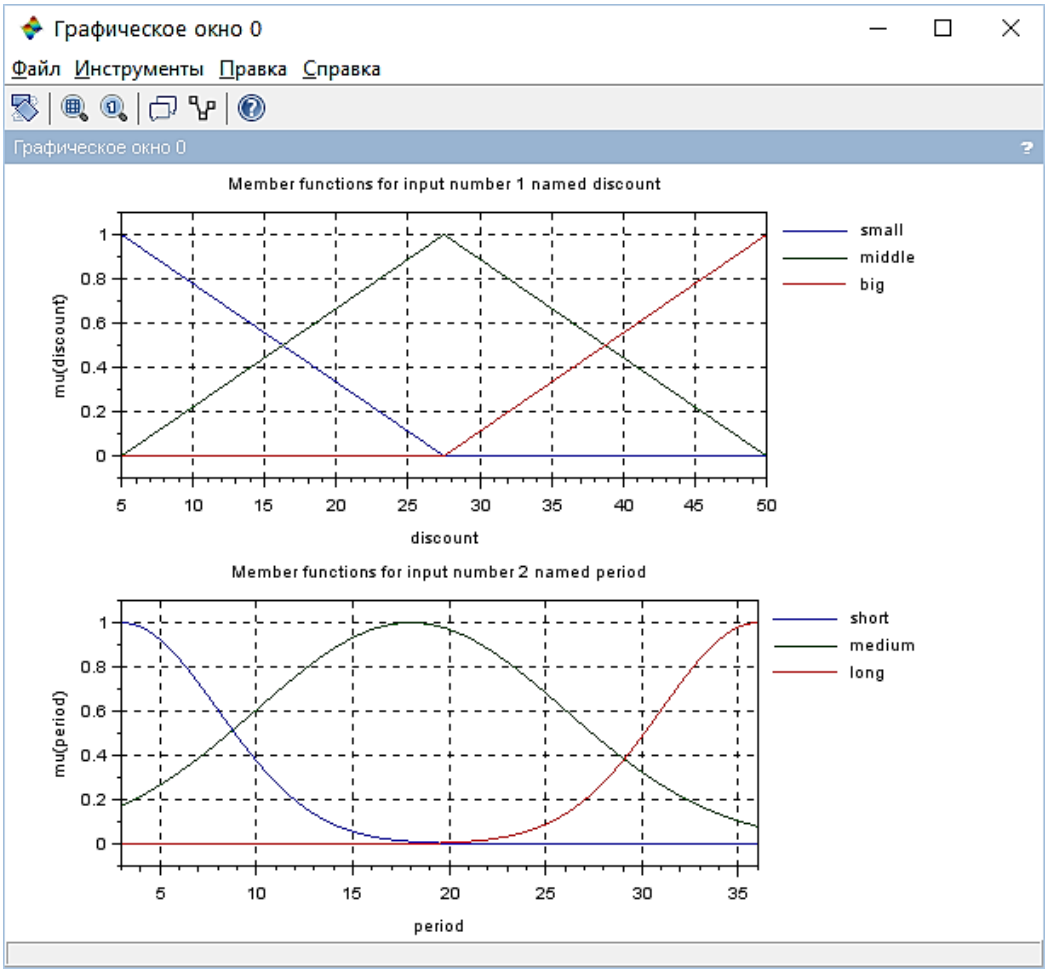


Рис. 2.14. Графическое представление входных переменных

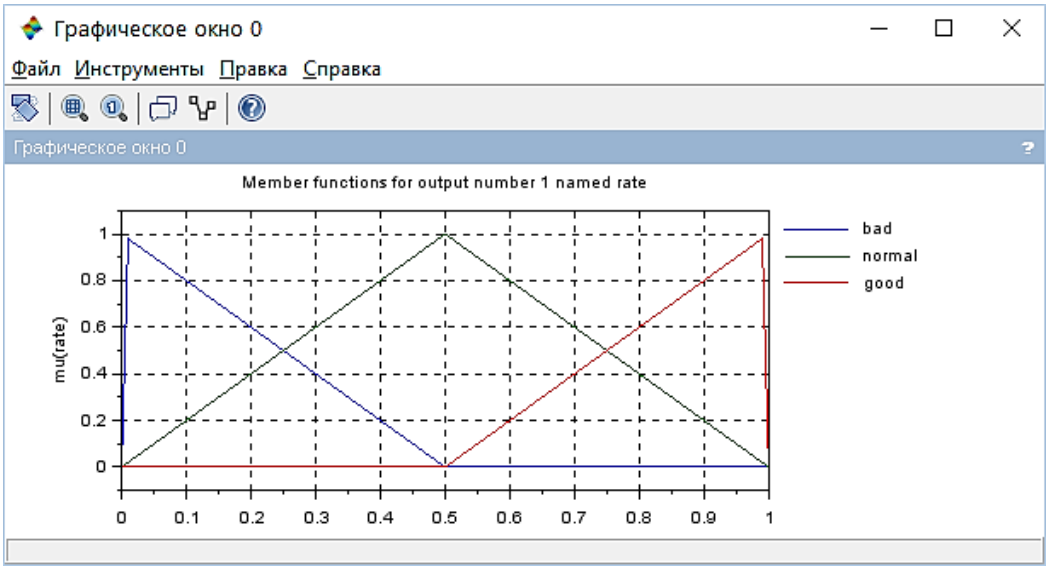


Рис. 2.15. Графическое представление выходной переменной

Шаг 3. Заключительным этапом построения СНВ является определение набора правил, которые задают связь входных переменных с выходными. Для этого в редакторе правил вывода определим:

ЕСЛИ *discount* = *small* И *period* = *short* ТО *rate* = *good*

ЕСЛИ *discount* = НЕ *small* И *period* = *long* ТО *rate* = *bad*

ЕСЛИ *discount* = *middle* И *period* = *normal* ТО *rate* = *normal*

ЕСЛИ *discount* = *big* И *period* = *short* ТО *rate* = *normal*

Состояние окна редактора правил вывода показано на рис. 2.16.

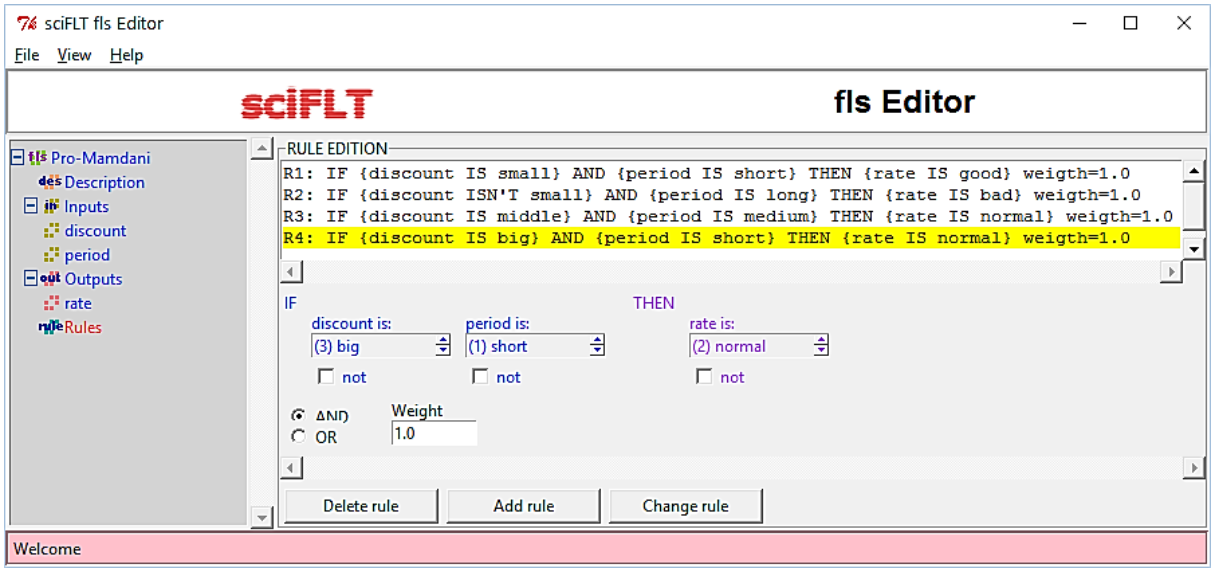


Рис. 2.16. Редактор правил вывода

Средство просмотра поверхности вывода

Для этого существует функция **plotsurf**, которая вызывается из командной строки окна Scilab и имеет следующие параметры:

plotsurf (fls [, ivar, ovar, vivar [,npart [,mod]]]).

Описание параметров функции:

fls – имя fls-структуры;

ivar – вектор входных переменных (задаются порядковые номера входных переменных, которые необходимо построить);

ovar – скаляр, номер выходной переменной;

vivar – вектор значений входных переменных;

npart – вектор, число разделов для каждой входной переменной;

mod – скаляр, вид отображения поверхности на экране (1 – grayscale, 2 – jetcolormap, 3 – hotcolormap, 4 – internal color map).

Пример 2.

```
fls=loadfls("E:\Tmp\Invest.fls");
plotsurf(fls,[1 2],1,[0 0]);
```

Поверхность вывода, которая соответствует правилам вывода из примера 1, показана на рисунке 2.17.

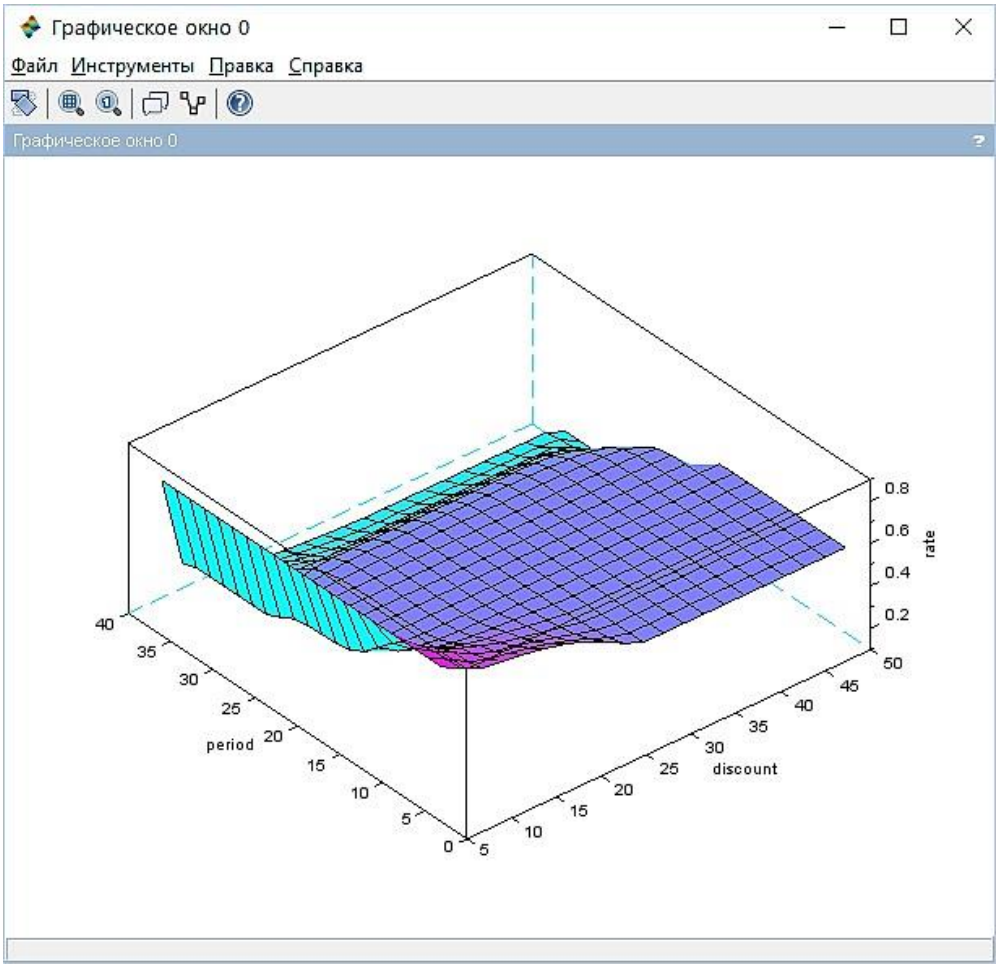


Рис. 2.17. Поверхность нечеткого вывода

Построение нечетких систем типа Суджено

Чтобы построить СНВ типа Суджено, нужно в меню **File** выбрать **New fls → Takagi-Sugeno**. Количество переменных определяется так же, как и при построении СНВ типа Мамдани.

Для СНВ типа Суджено изменения касаются только схемы определения ФП для выходных переменных. В Scilab можно разрабатывать два вида нечетких моделей. Первая модель – это нечеткая модель Суджено нулевого порядка с нечетким правилом вывода:

$$\text{if } x \text{ is } A \text{ and } y \text{ is } B \text{ then } z = k,$$

где A и B – нечеткие множества антецедента; k – четко заданная константа консеквента. Для построения такой модели при добавлении ФП выбирают тип – константа (constant) и задают ее численное значение. Вторая модель – нечеткая модель Суджено первого порядка. Для нее нечеткое правило вывода имеет следующий вид:

$$\text{if } x \text{ is } A \text{ and } y \text{ is } B \text{ then } z = p \cdot x + q \cdot y + r,$$

где p, q и r – константы.

В данном случае тип ФП – линейная зависимость (linear). Для определения параметров ФП необходимо ввести вектор, элементы которого соответствуют численным значениям констант консеквента.

Работа с редактором правил вывода, а также со средствами просмотра правил и поверхности вывода выполняется аналогично случаю построения СНВ по Мамдани.

Основное отличие нечеткого вывода по Суджено с использованием нечеткой модели нулевого порядка и правил вывода, определенных выше, заключается в том, что выходная переменная имеет три значения: bad, normal, good, которые задаются соответственно тремя константами: 0, 0.5, 1.

Контрольные вопросы

1. Структура типовой системы нечеткого вывода.

2. Отличие метода нечеткого вывода по Суджено от метода нечеткого вывода по Мамдани.
3. Формирование антецеденты и консеквенты для нечеткого правила вывода в Scilab.

Тема: Морфологический, синтаксический, семантический анализ запросов и синтез выходных сообщений

Лабораторная работа № 6

Алгоритм нечеткой кластеризации

Цель работы: изучить алгоритм нечеткой кластеризации, получить навыки решения задач методами нечеткой логики.

FCM алгоритм кластеризации

Алгоритм нечеткой кластеризации называют FCM-алгоритмом (**F**uzzy **C**lassifier **M**eans, **F**uzzy **C**-**M**eans). Его целью является автоматическая классификация множества объектов, задаваемых векторами признаков в пространстве признаков. Цель достигается так: вначале этот алгоритм определяет кластеры, а затем классифицирует объекты. Кластеры представляются нечеткими множествами, причем границы между кластерами также являются нечеткими.

FCM-алгоритм кластеризации предполагает, что объекты принадлежат всем кластерам с определенной ФП. Степень принадлежности определяется расстоянием от объекта до соответствующих кластерных центров. Данный алгоритм с помощью итераций вычисляет центры кластеров и новые степени принадлежности объектов.

Для множества K входных векторов x_k и N выделяемых кластеров c_j предполагается, что любой x_k принадлежит любому c_j с принадлежностью $\mu_{jk} \in [0,1]$, где j – номер кластера, а k – входного вектора. Принимаются во внимание следующие условия нормирования для μ_{jk}:

$$\sum_{j=1}^{N} \mu_{jk} = 1, \qquad \forall k = 1, \dots, K,$$

$$0 < \sum_{k=1}^{K} \mu_{\mathbf{j}k} \le K, \qquad \forall \mathbf{j} = 1, \dots, N.$$

Цель алгоритма заключается в минимизации суммы всех взвешенных расстояний $\|x_k - c_\mathrm{j}\|$:

$$\sum_{\mathrm{j}=1}^{N} \sum_{k=1}^{K} (\mu_{\mathrm{j}k})^q \|x_k - c_\mathrm{j}\| \to min,$$

где $\boldsymbol{q}$ – фиксированный параметр, задаваемый перед итерациями.

Чтобы достигнуть указанной цели работы, необходимо решить следующую систему уравнений:

$$\partial / \partial \mu_{\mathrm{j}k} \left(\sum_{\mathrm{j}=1}^{N} \sum_{k=1}^{K} (\mu_{\mathrm{j}k})^q \|x_k - c_\mathrm{j}\| \right) = 0,$$

$$\partial / \partial c_\mathrm{j} \left(\sum_{\mathrm{j}=1}^{N} \sum_{k=1}^{K} (\mu_{\mathrm{j}k})^q \|x_k - c_\mathrm{j}\| \right) = 0.$$

Совместно с условиями нормирования μ_{jk} данная система дифференциальных уравнений имеет следующее решение:

$$c_\mathrm{j} = \frac{\sum_{k=1}^{K} (\mu_{\mathrm{j}k})^q \cdot x_k}{\sum_{k=1}^{K} (\mu_{\mathrm{j}k})^q}$$

(взвешенный центр гравитации) и

$$\mu_{\mathrm{j}k} = \frac{1 / \|x_k - c_\mathbf{j}\|^{1/(q-1)}}{\sum_{\mathrm{j}=1}^{N} \left(1 / \|x_k - c_\mathbf{j}\|^{1/(q-1)}\right)}.$$

Алгоритм нечеткой кластеризации выполняется по шагам.

Шаг 1: *Инициализация*

Выбираются следующие параметры:

- необходимое количество кластеров *N*, 2 ≤ N ≤ K;

- мера расстояний, как Евклидово расстояние;
- фиксированный параметр q (обычно ~ 1.5);
- начальная матрица принадлежности $U^{(0)} = (\mu_{jk})^{(0)}$ объектов x_k с учетом заданных начальных центров кластеров c_j.

Шаг 2: *Регулирование позиций $c_j^{(t)}$ центров кластеров*

На t-м шаге итерации при известной матрице $\mu_{jk}^{(t)}$ вычисляется $c_j^{(t)}$ в соответствии с решением системы дифференциальных уравнений, которое приведено выше.

Шаг 3: *Корректировка значений принадлежности μ_{jk}*

Учитывая известные $c_j^{(t)}$, вычисляются $\mu_{jk}^{(t)}$, если $x_k \neq c_j$, в противном случае значение принадлежности определяется так:

$$\mu_k^{(t+1)} = \{ \begin{matrix} 1, l = \text{j}, \\ 0, \end{matrix}$$

Шаг 4: *Остановка алгоритма*

Алгоритм нечеткой кластеризации останавливается при выполнении следующего условия:

$$\|U^{(t+1)} - U^{(t)}\| \leq \text{s},$$

где $\cdot$ – матричная норма (например, Евклидова норма), а ε – заранее задаваемый уровень точности.

Решение задач кластеризации в Scilab

В Scilab отсутствуют готовые средства для решения задач кластеризации. Поэтому для данной лабораторной работы необходимо разработать приложение, реализующее алгоритм кластеризации, который был описан выше. Данный алгоритм должен обеспечивать вычисление функции нахождения центров кластеров.

Описание функции: [*center, U, obj_fcn*] = *fcm*(*data, cluster_n*).

Аргументами данной функции являются:

data – множество данных, подлежащих кластеризации, каждая строка описывает точку в многомерном пространстве характеристик;

cluster_n – количество кластеров (более одного).

Левая часть функции (в квадратных скобках) содержит параметры вывода, которые эта функция возвращает:

center – матрица центров кластеров, каждая строка которой содержит координаты центра отдельного кластера;

U – результирующая матрица ФП;

obj_fcn – значение целевой функции на каждой итерации.

Для работы данной функции необходимо обеспечить загрузку данных, которые подлежат кластеризации. Эту операцию можно осуществить либо из заранее подготовленного файла, либо обеспечить ручной ввод данных с помощью клавиатуры.

Также необходимо предусмотреть возможность вызова функции кластеризации при обеспечении ею дополнительного набора параметров: fcm(data, cluster_n, options):

- options(1) – показатель степени матрицы U (по умолчанию 2.0);
- options(2) – максимальное число итераций (по умолчанию 100);
- options(3) – шаг изменения целевой функции (по умолчанию 1e-5);
- options(4) – отображение на каждом шаге (по умолчанию 1).

Задание. В таблице вариантов, которая приведена на следующей странице, для каждого из вариантов представлены сверху вниз координаты 20-ти точек (первая координата по оси X, вторая по оси Y). Необходимо выполнить их кластеризацию. Число кластеров нужно подобрать так, чтобы были явно видны разделяемые области (рекомендованное число кластеров 3...5). Каждая точка отдельного кластера и его центр должны быть представлены своим цветом.

Таблица вариантов

1	2	3	4	5	6	7	8
0.1964506	62.25464	2.3218025	18.993749	59.145097	89.41365	5.1847671	77.075744
50.752213	98.225833	72.654473	25.839815	68.067427	34.903639	41.492418	24.352242
40.76043	75.429888	15.340586	9.8787374	7.3929611	11.053652	72.212356	21.261149
84.080461	54.547881	23.552638	6.1990272	94.336947	20.233778	7.7462539	10.992342
50.172657	72.86016	8.7973828	4.0349683	12.863307	13.04691	58.558784	69.814808
91.287808	2.5259695	71.059537	74.001472	20.190808	85.73953	37.079446	41.509065
44.357295	40.251685	68.887276	61.626601	19.693034	63.780164	21.161167	50.298188
59.83784	9.8313199	65.953195	65.835834	89.286902	40.711227	19.032685	75.116068
77.418426	26.086253	18.151161	25.145971	46.17919	66.919379	56.079538	99.401472
79.220083	36.363423	39.04966	38.433501	62.512917	20.426016	94.247916	18.287624
55.046049	17.466178	15.869047	43.964602	70.597066	83.104313	68.177248	30.219174
40.850437	92.341395	62.40715	65.407369	70.181696	1.221633	27.34241	37.854864
72.174381	76.051409	63.760356	58.781064	40.879997	48.844617	20.717754	71.531986
47.685359	56.402041	42.704886	60.208319	6.3622138	95.498771	19.379388	95.241537
63.930579	37.970652	10.716815	4.5350203	6.5739339	5.8743121	67.978376	47.039186
99.638653	87.762262	23.822966	20.294443	53.310041	82.584649	58.836574	18.709417
15.747883	82.174258	94.629474	78.442738	3.3158187	29.807416	93.317538	25.571879
53.506937	67.870581	45.766853	26.375362	31.578356	7.7575968	55.091229	44.350661
21.290646	8.2200981	89.644787	43.832764	37.858232	58.460923	80.40547	72.340782
55.914506	25.527314	44.384705	86.64859	46.195234	75.287136	10.744897	87.619101
43.04966	74.444567	80.895682	37.921421	62.873698	5.172298	74.039251	3.7332086
2.2805485	22.695036	68.317985	76.687161	28.785153	59.586251	56.103317	42.934664
57.614598	68.369308	3.4019315	60.066213	32.920487	38.337053	76.61155	31.572331
71.491304	93.650726	23.805456	78.567356	47.19233	49.002203	78.306589	36.824773
93.21636	50.530174	94.920116	73.871156	33.537696	52.727951	14.388315	14.587743
12.326993	25.248146	21.827886	55.442603	55.530697	6.8894547	16.471925	67.683793
28.655522	68.188398	61.546878	99.291496	11.960808	88.430778	31.774142	52.619794
1.2479957	28.363682	83.135434	97.574285	76.139997	71.912938	50.265956	40.036257
57.694048	14.094857	77.340126	37.096223	47.909885	6.9425958	69.204961	0.2910803
39.386961	67.591096	42.44191	30.322382	28.169693	11.522096	70.065794	30.681815
68.885837	45.126776	72.62126	95.195201	23.800978	48.626807	88.70612	79.026939
97.023218	75.430292	70.999773	71.278581	32.942055	76.715826	69.797695	95.779504
85.157643	13.702143	47.45746	11.923701	23.06728	8.8052981	67.989912	66.892712
33.933045	66.082405	94.386921	50.091632	21.362966	70.085613	36.159398	29.29616
87.725318	38.900542	14.596486	32.900535	40.54998	18.791388	26.739977	82.238994
11.314025	70.018205	7.1410105	48.089468	30.953712	20.178856	7.7368706	1.798455
52.641283	91.680057	67.337386	33.03696	67.629716	40.628213	14.941003	87.107014
52.973941	21.229	65.369247	63.044754	97.069163	40.96657	32.018391	31.810243
92.917561	26.978331	19.968961	21.171908	54.417966	17.695645	20.260546	57.244733
97.654303	31.998894	60.141252	44.860231	2.0474797	33.129312	44.988587	57.386581

Контрольные вопросы

1. Понятия кластера и его особенности.
2. Назначение алгоритма нечеткой кластеризации.
3. Условия нормирования функции принадлежности.
4. Понятие суммы взвешенных расстояний.
5. Содержание шагов алгоритма нечеткой кластеризации.
6. Функция нахождения центров кластеров и ее параметры.

Тема: Реализация и тестирование СИИ
Лабораторная работа № 7
НЕЙРОННЫЕ СЕТИ

Обучение классификации нейронной сети

Цель работы: изучить пример классификации объектов с помощью нейронной сети, а также получить навыки решения практических задач классификации.

На сайте модулей ATOMS пакета Scilab имеется руководство с описанием примера использования модуля Neural Network Module. Этот модуль разработал китайский программист Тан Чин Лух на основе англоязычной книги известного американского специалиста по нейронным сетям Мартина Т. Хагана Neural Network Design [9].

В данной работе для классификации используется готовый набор данных, который требуется загрузить по приводимой ссылке: csv.

Импорт данных

Этот набор данных представляет собой 100 образцов, которые подразделяются на два вида, как 0 или 1 (хранится в третьей колонке), по двум параметрам (хранятся в первом и втором столбцах файла data_classification.csv). Непосредственно данные импортируются в Scilab с помощью следующей команды:

```
t=csvRead("data_classification.csv");
```

Представление данных

Функция реализована в модуль нейронной сети для упрощения построения 2-х групп точек данных, которые показаны на рис. 3.1. Источник данных (P) и цель (T) разделены, а данные транспонируются в требуемый формат модуля.

```
P = t(:,1:2)';
T = t(:,3)';
plot_2group(P,T);
```

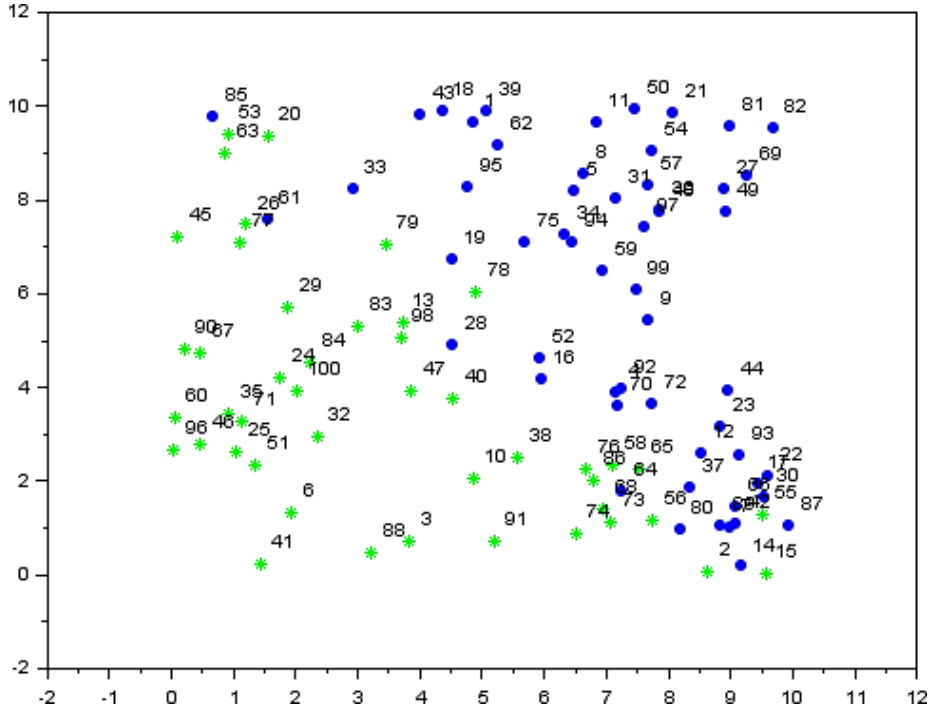


Рис. 3.1. Исходные данные для классификации

Алгоритм обратного распространения
с градиентной функцией обучения

Полный синтаксис алгоритма обратного распространения с градиентной функцией такой:

W = ann_FFBP_gd(P,T,N,af,lr,itermax,mse_min,gd_min),

где P – источник входных данных для обучения; T – подготовленная цель; N – число нейронов в слоях, включая входной и выходной слои; af – функция активации от скрытого слоя к выходному слою; lr – скорость обучения; itermax – максимальное время для обучения; mse_min – минимальная ошибка (достигаемая цель) и gd_min – минимальный градиент для остановки обучения.

С левой стороны W представляет вес выхода и смещение.

Для простоты иллюстрации будем использовать только первые три обязательных поля, а другие поля оставим пустыми, чтобы использовались значения по умолчанию. Код программы:

```
clear W; tic();
W = ann_FFBP_gd(P,T,[2 3 1]);
toc(); y = ann_FFBP_run(P,W);
sum(T == round(y))
```

Консольный вывод Scilab:

```
-->clear W;
-->tic();
-->W = ann_FFBP_gd(P,T,[2 3 1]);
-->toc();
ans =
        31.655
-->y = ann_FFBP_run(P,W);
-->sum(T == round(y))
ans =
        74
```

Результат работы программы представлен на рис. 3.2 и рис. 3.3.

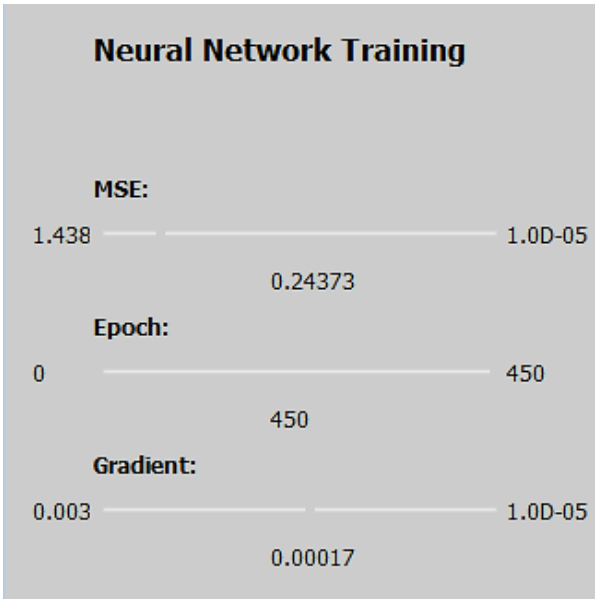


Рис. 3.2. Параметры запуска градиентной функции обучения

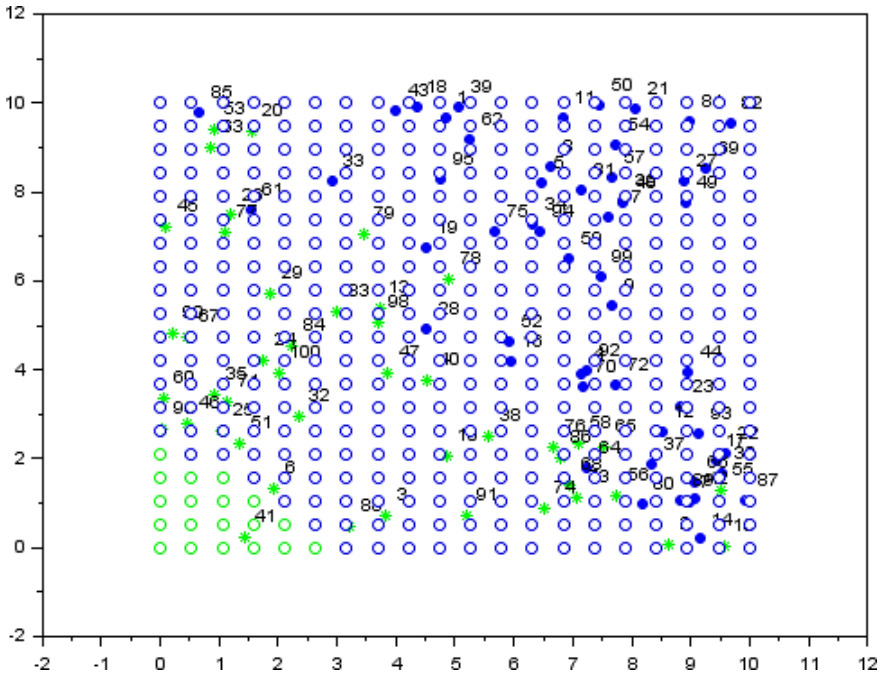


Рис. 3.3. Данные классификации с градиентной функцией обучения

Алгоритм обратного распространения
с градиентной функцией скорости и момента обучения

Для того чтобы улучшить сходимость градиентного обучения, в книге «Neural Network Design» [9] предлагается несколько методов. Для данного примера используется сочетание функций адаптации скорости обучения и момента обучения.

Код программы выглядит следующим образом:

```
clear W; tic();
W = ann_FFBP_gdx(P,T,[2 3 1]); toc()
y = ann_FFBP_run(P,W); sum(T == round(y))
```

Это обучение сходится быстрее, если сравнивать с предыдущим примером обучения.

Консольный вывод Scilab:

```
-->clear W;
-->tic();
-->W = ann_FFBP_gdx(P,T,[2 3 1]);
-->toc()
```

```
ans  =
          38.829
-->y = ann_FFBP_run(P,W);
-->sum(T == round(y))
ans  =
          89.
```

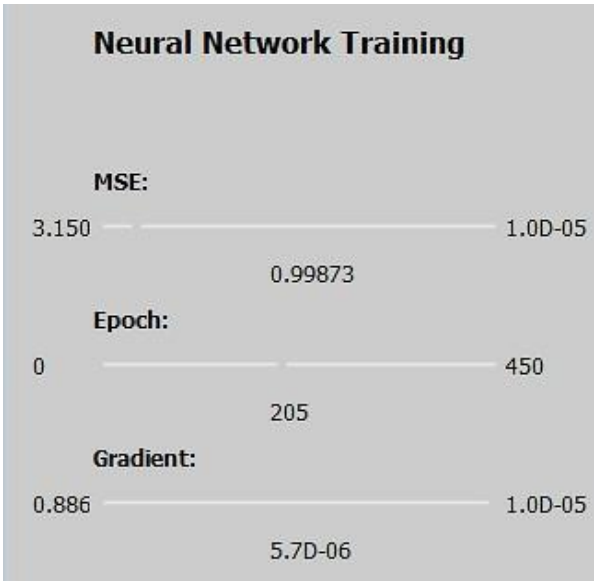


Рис. 3.4. Параметры запуска градиентной функции скорости и момента обучения

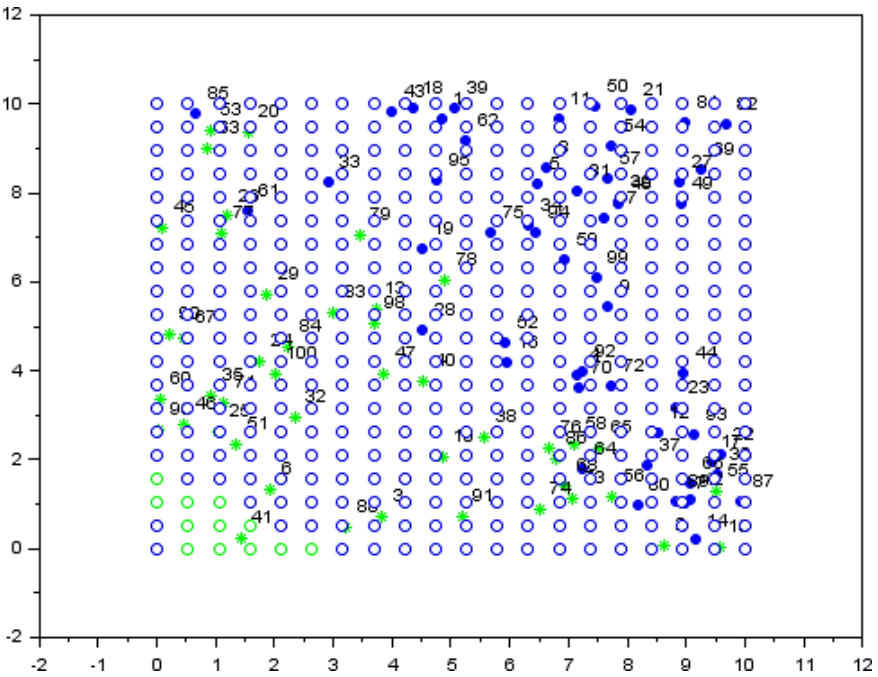


Рис. 3.5. Выходные данные градиентной функцией скорости и момента обучения

Алгоритм обратного распространения
с функцией обучения Левенберга-Маркуарда

Еще более быстрым алгоритмом сходимости является алгоритм Левенберга-Маркуарда, который сходится после нескольких десятков итераций. Его код выглядит следующим образом:

```
clear W; tic();
W = ann_FFBP_lm(P,T,[2 3 1]); toc()
y = ann_FFBP_run(P,W); sum(T == round(y))
```

Консольный вывод Scilab:

```
-->clear W;
-->tic();
-->W = ann_FFBP_lm(P,T,[2 3 1]);
-->toc()
ans =
        4.565
-->y = ann_FFBP_run(P,W);
-->sum(T == round(y))
ans =
        90
```

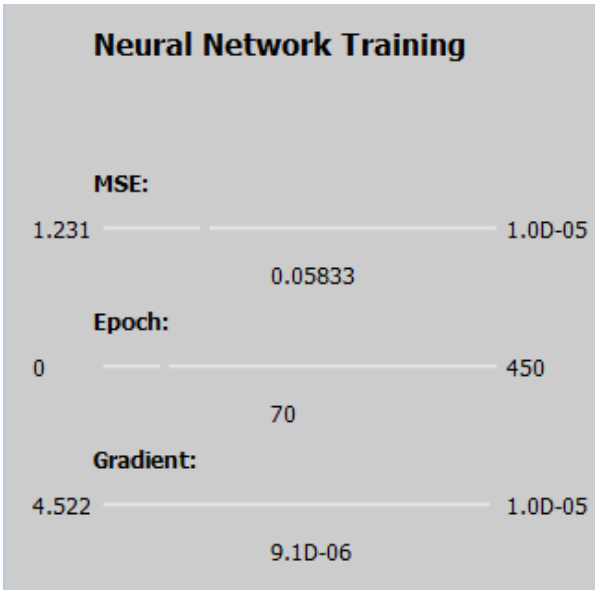


Рис. 3.6. Параметры запуска для функции обучения Левенберга-Маркуарда

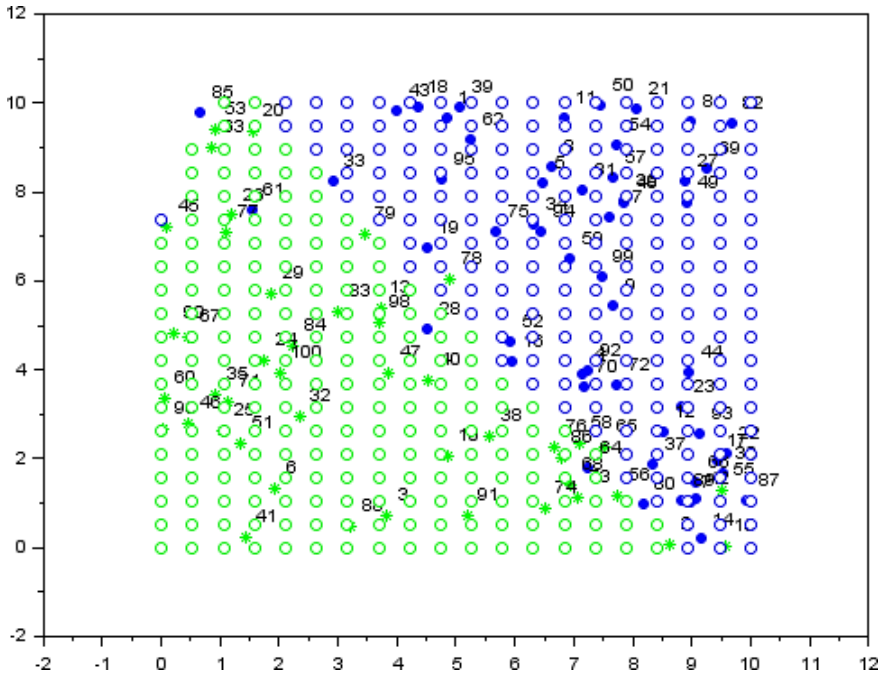


Рис. 3.7. Выходные данные функции обучения Левенберга-Маркуарда

Визуализация результатов

Границу нейронной сети сложно описать математически, особенно при большом числе нейронов скрытого слоя или при увеличении числа слоев, поэтому визуализация этой границы затруднена.

Используем для визуализации набор данных, который охватывает некоторый диапазон, имеющий области или границы сети.

```
clf(0); scf(0);
// Построение набора данных
plot_2group(P,T);
// Создание сетки входного диапазона
nx = 20; ny = 20;
xx = linspace(0,10,nx); yy = linspace(0,10,ny);
[X,Y] = ndgrid(xx,yy);
// Использование NN для классификации данных сетки
P2 = [X(:)';Y(:)'];
y2 = ann_FFBP_run(P2,W);
```

```
// Извлечение данных в соответствии с категориями
G0 = P2(:,find(round(y2)==0))
G1 = P2(:,find(round(y2)==1))
// Построение границ областей для групп
plot(G0(1,:),G0(2,:),'go');
plot(G1(1,:),G1(2,:),'bo');
```

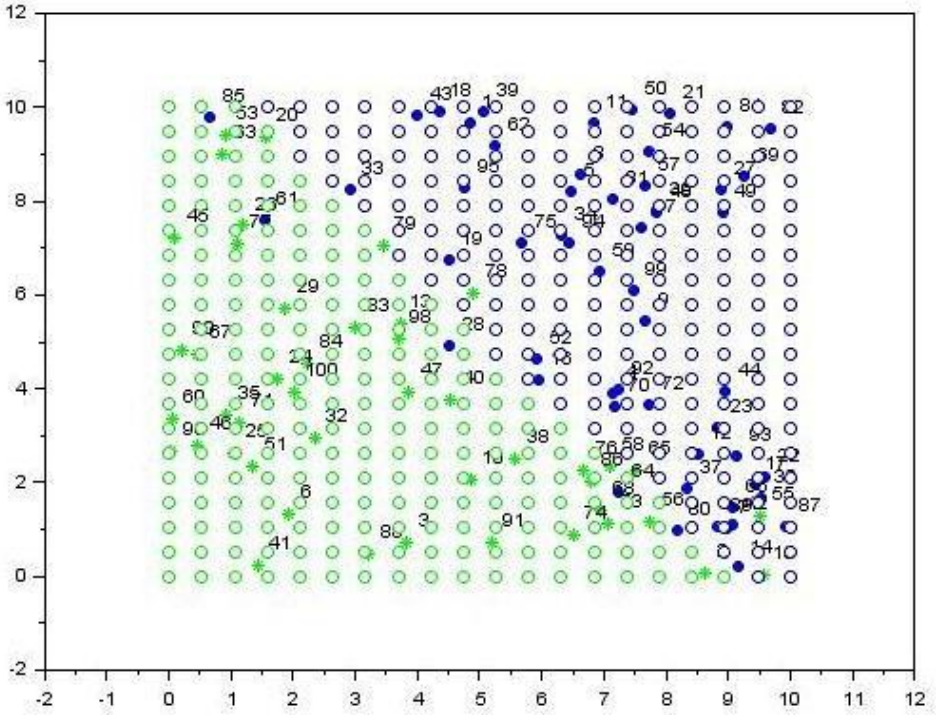


Рис. 3.8. Графическое окно с построенной границей областей групп

Функции модуля нейронных сетей поставляются с кратким описанием, к которому можно получить доступ через помощь Scilab. Подстановка разных параметров дает разные результаты. Сложные наборы данных требуют увеличения числа нейронов или слоев.

Контрольные вопросы

1. Понятия источника данных и цели.
2. Параметры функции алгоритма обратного распространения.
3. Условия визуализации результатов классификации.

Тема: Реализация и тестирование СИИ
Лабораторная работа № 8
ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ

Исследование генетического алгоритма

Цель работы: изучить параметры и особенности генетического алгоритма и получить навыки его использования.

В Scilab имеется готовая функция для решения задач оптимизации с использованием генетического алгоритма. В данной лабораторной работе на основе примера исследуется влияние исходных параметров этой функции на формирование оптимальной популяции.

Последовательность вызова функции

```
[pop_opt,fobj_pop_opt,pop_init,fobj_pop_init] =
optim_moga(ga_f,pop_size,nb_generation,p_mut,p_cross,Log,param)
```

Аргументы

`ga_f` – оптимизируемая функция, которая имеет следующий тип:

```
y = f(x)
```

или

```
y = list(f,p1,p2,...).
```

```
// вставка в начало списка l(0) - по-прежнему не существует
    l(0)  = "foo"
    l(1)                  // равно "foo"
    l($+1) = "hello"      // вставка в конец списка
    l(2) = "toto"         // перезапись значения типа double
    l(3) = rand(1,2)    // перезапись вектора строковых значений
    l(3) = null()       // удаление третьего элемента
// объявление нового списка
    lbis = list("gewurtz", "caipirina" ,"debug")
    lter = lstcat(l,lbis) // слияние двух списков
    size(lter) - size(lbis) - size(l) // должен быть ноль
```

`pop_size`	размер популяции (значение по умолчанию: 100).
`nb_generation`	число поколений или итераций (по умолчанию: 10).
`p_mut`	вероятность мутации (по умолчанию: 0,1).
`p_cross`	вероятность кроссовера (по умолчанию: 0,7).
`Log`	если %T, то функция вывода будет вызываться в конце каждой итерации, смотрите «`output_func`» под переменной `param` ниже.
`param`	список параметров:

- «`codage_func`»: функция кодирования и декодирования индивидуумов (по умолчанию: `coding_ga_identity`);
- «`init_func`»: функция инициализации популяции (по умолчанию: `init_ga_default`);
- «`dimension`», «`minbounds`» и «`maxbounds`»: параметры функции инициализации, используемые для определения начальной совокупности;
- «`crossover_func`»: функция скрещивания между двумя индивидами (по умолчанию: `crossover_ga_default`);
- «`mutation_func`»: функция мутации одного индивида (по умолчанию: `mutation_ga_default`);
- «`selection_func`»: функция выбора индивидов в конце поколения (по умолчанию: `selection_ga_elitist`);

- «nb_couples»: количество пар, которые будут выбраны для выполнения кроссовера и мутации (по умолчанию: 100);
- «pressure»: значение эффективности наихудшего индивидуума (значение по умолчанию: 0,05);
- «output_func»: функция вывода, вызываемая после генерации, если Log %T (по умолчанию output_moga_default).

pop_opt — популяция оптимальных индивидуумов.

fobj_pop_opt — набор значений многоцелевой функции, связанных с pop_opt (необязательно).

pop_init — начальная популяция индивидуумов (необязательно).

fobj_pop_init — множество мультизначений целевой функции, связанные с pop_init (по желанию).

Пример

```
function f=deb_1(x) // оптимизируемая функция
  f1_x1 = x(1);
  g_x2 = 1+9*sum((x(2:$)-x(1)).^2)/(length(x)-1);
  h = 1-sqrt(f1_x1/g_x2);
  f(1,1) = f1_x1;   // 1-й элемент индивидуума
  f(1,2) = g_x2*h;  // 2-й элемент индивидуума
endfunction
PopSize   = 100;    // размер популяции
Proba_cross = 0.5;  // вероятность кроссовера
Proba_mut = 0.3;    // вероятность мутации
NbGen = 4;          // число итераций (поколений)
NbCouples = 110;    // число пар для кроссовера и мутации
Log = %T;           // вывод значений после каждой итерации
nb_disp = 10;  // число отображений из оптимальной совокупности
pressure = 0.1;     // эффективность наихудшего индивидуума
ga_params = init_param(); // инициализация начальной популяции
ga_params = add_param(ga_params,'dimension',2);
ga_params = add_param(ga_params,'minbound',zeros(2,1));
ga_params = add_param(ga_params,'maxbound',ones(2,1));
[pop_opt, fobj_pop_opt, pop_init, fobj_pop_init] =..
```

```
optim_moga(deb_1, PopSize, NbGen, Proba_mut, Proba_cross, Log,
           ga_params)
```

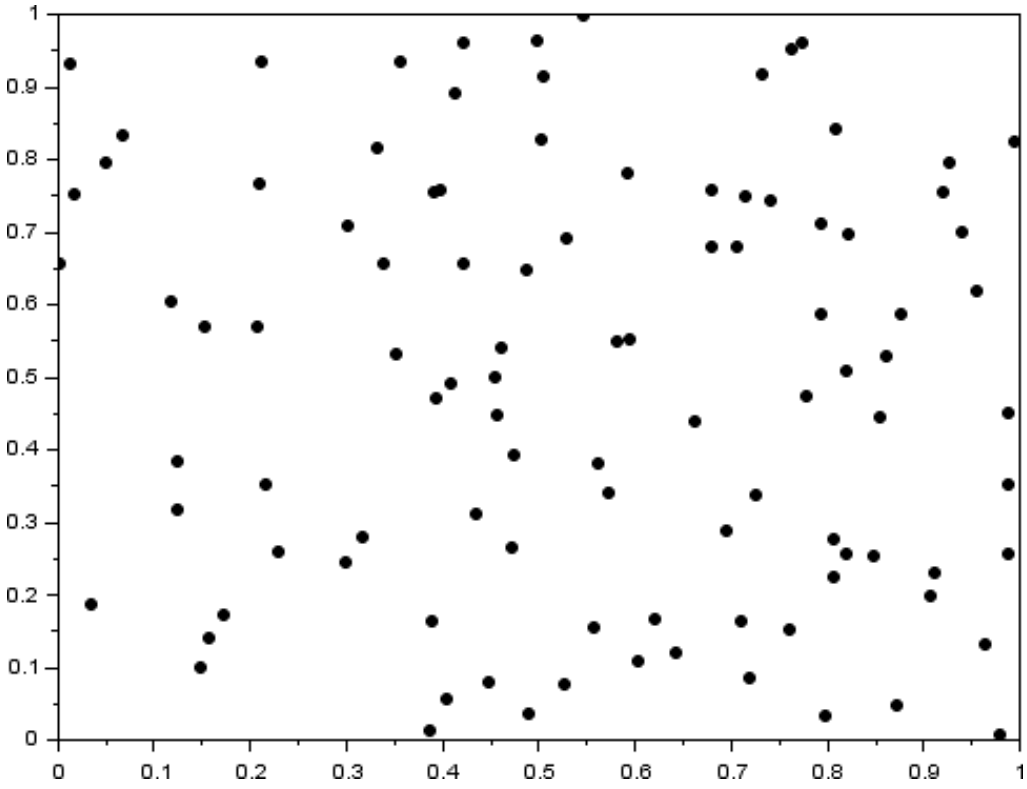


Рис. 4.1. Графическое изображение начальной популяции

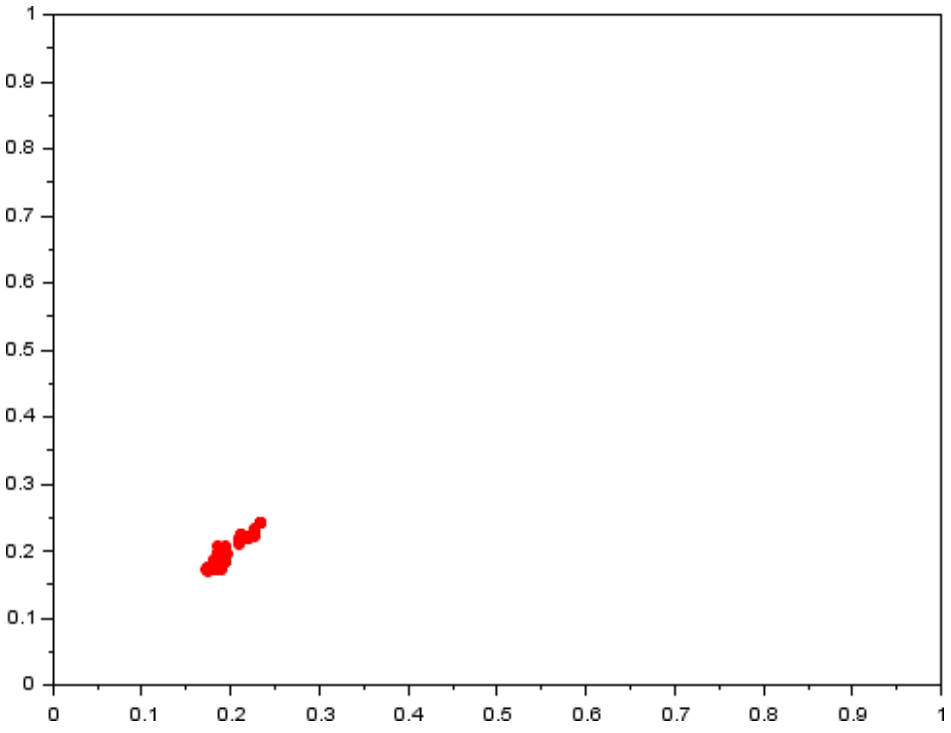


Рис. 4.2. Графическое изображение популяции оптимальных индивидуумов

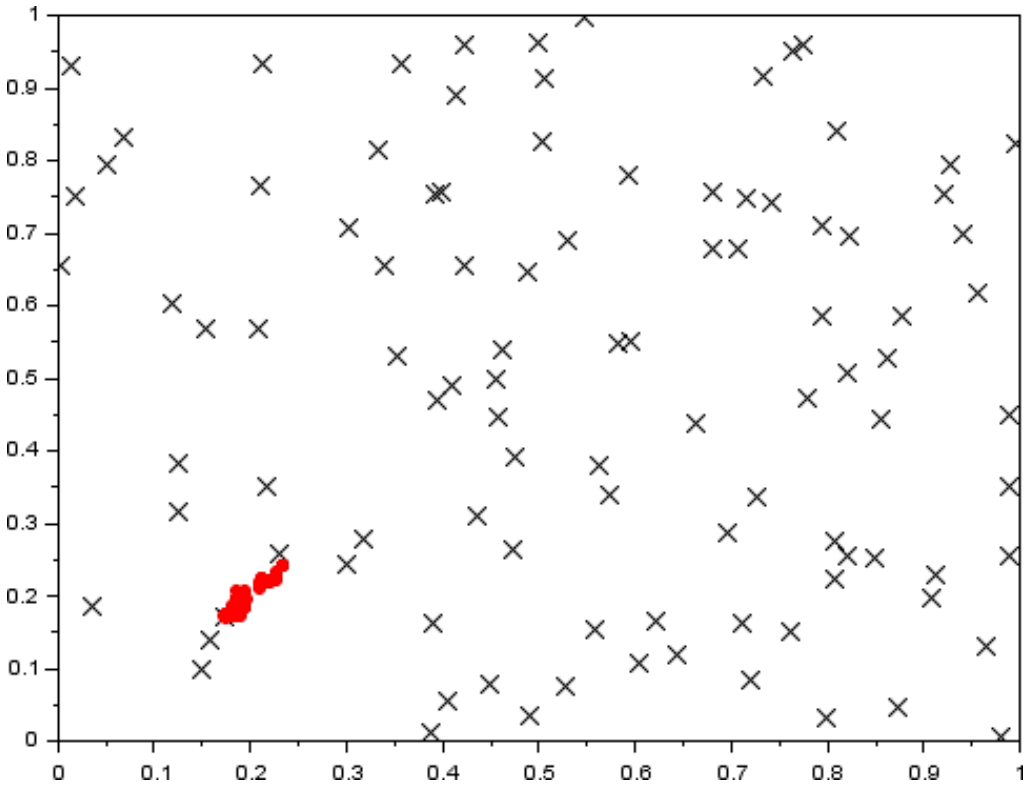


Рис. 4.3. Популяция оптимальных индивидуумов на фоне начальной популяции

Задание

Для варианта, заданного преподавателем, используя приведенный пример в качестве образца, исследовать влияние исходных параметров на значения оптимальной популяции.

Варианты заданий

№вар	1	2	3	4	5	6	7	8	9	10	11	12
P_cros	0.7	0.6	0.5	0.7	0.6	0.5	0.7	0.6	0.5	0.7	0.6	0.5
P_mut	0.05	0.15	0.25	0.35	0.05	0.15	0.25	0.35	0.05	0.15	0.25	0.35
N_gen	4	5	6	7	8	9	8	7	6	5	4	9
press	0.1	0.05	0.1	0.05	0.1	0.05	0.1	0.05	0.1	0.05	0.1	0.05

Содержание отчета

1. Вариант задания.
2. Листинг программы.
3. График начальной популяции.

4. График популяции оптимальных индивидуумов.
5. Совместный график начальной и оптимальной популяций.
6. Выводы по работе.

Контрольные вопросы

1. Понятия индивидуума и популяции.
2. Характеристика операции отбора.
3. Скрещивание и мутация и их параметры.
4. Популяция оптимальных индивидуумов.

ЗАКЛЮЧЕНИЕ

Учебное пособие включает в себя дополнительную теорию к основным лекциям, а также задания лабораторного практикума, содержащего восемь лабораторных работ.

Тематика лабораторных работ включает как общие методы программирования в Scilab, так и функционал готовых модулей, использующих математические методы нечеткой логики, искусственных нейронных сетей и генетические алгоритмы.

Лабораторный практикум проводится на персональных ЭВМ с применением базового математического пакета Scilab версии 5.5.2, а также дополнительных модулей:

- ANN Toolbox (инструменты для работы с искусственными нейронными сетями);
- Scilab Neural Network Module (модуль нейронных сетей);
- Fuzzy Toolbox (инструменты нечеткой логики).

Пособие дает возможность студентам на практических примерах усвоить положения и принципы систем искусственного интеллекта, а также способы анализа задач в этой области.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Алексеев, Е. Scilab. Решение инженерных и математических задач / Е. Алексеев, Москва: ALT Linux; БИНОМ. Лаборатория знаний. 2008, 260 с.
2. Андриевский, Б. Элементы математического моделирования в программных средах MATLAB 5 и Scilab / Б. Андриевский, А. Фрадков, Санкт-Петербург: Наука, 2001. 286 с.
3. Chandler G. Stephen R. Introduction to Scilab. 2002, 27 p.
4. Domanie de Voluceau. Introduction to Scilab. Scilab Groupe. 125 p. URL: http://www./Intro.pdf.
5. Domanie de Voluceau. Scilab Reference Manual. Scilab Groupe. 700p. URL: http://www./manual.pdf.
6. Gomez C. Communication Toolbox. 12 p. URL: http://www./comm.pdf.
7. Domanie de Voluceau. Guide for Developers. Scilab Groupe. 29 p. URL: http://www./Internals.pdf.
8. Domanie de Voluceau. InterSci. A Scilab interfacing tool. Scilab Groupe. 14 p. URL: http://www./Intersci.pdf.
9. Hagan M.T., Demuth H.B., Beale M.H., De Jesus O. Neural Network Design, 2-nd Edition, eBook. URL: https://hagan.okstate.edu/nndesign.pdf.
10. Nikoukbah K. LMI Tool: a Package for LMI Optimization in Scilab. 16 p. URL: http://www./Imi.pdf.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
НЕВИННОМЫССКИЙ ТЕХНОЛОГИЧЕСКИЙ ИНСТИТУТ
(ФИЛИАЛ)

Методические указания к самостоятельной работе
для студентов направления
15.03.04 «Автоматизация технологических процессов и производств»
по дисциплине

«СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА»

Невинномысск, 2025

Методические указания разработаны в соответствии с требованиями ФГОС ВО в части содержания и уровня подготовки выпускников направления подготовки 15.03.04 «Автоматизация технологических процессов и производств».

Методические указания содержат рекомендации по организации самостоятельной работы студента при изучении дисциплины «Системы искусственного интеллекта».

Составитель доцент кафедры ИСЭА Д.В. Болдырев

Ответственный редактор доцент кафедры ИСЭА А.А. Евдокимов

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Дисциплина «Системы искусственного интеллекта» ставит своей целью формирование следующих компетенций будущего бакалавра по направлению подготовки 15.03.04 — Автоматизация технологических процессов и производств.

ОПК-2. Применять основные методы, способы и средства получения, хранения, переработки информации	ИД-З$_{ОПК-2}$ Решает типовые задачи профессиональной деятельности, связанные с получением, хранением и переработкой информации.	Демонстрирует понимание парадигмы искусственного интеллекта, представления знаний в интеллектуальных системах управления; применяет новые методы решения задач автоматизации технологических процессов и производств; проводит сравнительный анализ и обосновывает выбор модели и средств представления знаний при решении задач автоматизации
ОПК-11. Способен проводить научные эксперименты с использованием современного исследовательского оборудования и приборов, оценивать результаты исследований.	ИД-З$_{ОПК-11}$ Проводит математическую и статистическую обработку опытных данных о характеристиках средств и систем автоматизации.	Использует методы поиска решений, применяемые в системах искусственного интеллекта

Главными задачами дисциплины являются: усвоение студентами основных положений теории искусственного интеллекта, способов представления знаний и их инженерии, правил анализа и синтеза экспертных систем.

В результате освоения дисциплины студент должен:

- знать основные положения теории искусственного интеллекта; способы организации интеллектуальных систем; способы построения баз данных, баз знаний и экспертных систем; модели и методы формализации и представления знаний в интеллектуальных системах;
- уметь использовать интеллектуальные информационные системы, инструментальные средства управления базами данных и знаний;
- владеть навыками решения задач с помощью систем искусственно интеллекта и экспертных систем.

Методические указания предназначены для выполнения самостоятельной работы по дисциплине «Интеллектуализация систем управления» с учетом требований ФГОС ВО для направления подготовки 15.03.04 — Автоматизация технологических процессов и производств. Они способствуют лучшему усвоению студентами теоретических положений и обеспечивает приобретение практических навыков по исследованию элементов и систем автоматического регулирования и управления.

1. ОБЩАЯ ХАРАКТЕРИСТИКА САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ ПРИ ИЗУЧЕНИИ ДИСЦИПЛИНЫ «СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА»

Самостоятельная работа студентов (далее — СРС) является неотъемлемой составляющей образовательного процесса в Университете и является обязательной для каждого студента. Основная цель СРС — освоение в полном объеме образовательной программы и последовательное формирование компетенций эффективной самостоятельной профессиональной (практической и научно-теоретической) деятельности. Самостоятельная работа конкретна по своей предметной направленности и сопровождается непрерывным контролем и оценкой ее результатов.

Количество часов, отводимое на самостоятельную работу, определяется учебным планом направления подготовки 15.03.04.

Содержательно самостоятельная работа студентов определяется ФГОС ВО направления подготовки 15.03.04, программой и учебно-методическим комплексом дисциплины «Системы искусственного интеллекта».

Методика организации самостоятельной работы студентов зависит от структуры, характера и особенностей дисциплины «Системы искусственного интеллекта», объема часов на ее изучение, вида заданий для СРС, индивидуальных возможностей студентов и условий учебной деятельности.

Формы самостоятельной работы студентов определяются содержанием дисциплины «Системы искусственного интеллекта», степенью подготовленности студентов. Они могут быть тесно связаны с теоретическим курсом и иметь учебный или учебно-исследовательский характер. Форму самостоятельной работы студентов определяют кафедра ИСЭА при разработке программы дисциплины «Системы искусственного интеллекта».

Самостоятельная работа может осуществляться индивидуально или группами студентов в зависимости от цели, объема, конкретной тематики самостоятельной работы, уровня сложности, уровня умений студентов.

СРС, не предусмотренная образовательной программой, учебным планом и учебно-методическими материалами, раскрывающими и конкретизирующими их содержание, осуществляется студентами инициативно, с целью реализации собственных учебных и научных интересов.

В учебном процессе выделяют аудиторную и внеаудиторную самостоятельную работу.

Аудиторная самостоятельная работа по дисциплине «Системы искусственного интеллекта» выполняется на учебных занятиях (лекциях, практических, лабораторных занятиях и консультациях) под руководством преподавателя и по его заданию.

Внеаудиторная самостоятельная работа студентов выполняется во внеаудиторное время по заданию и при методическом руководстве и контроле преподавателя, но без его непосредственного участия. СРС включает в себя:

- подготовку к аудиторным занятиям (лекционным и практическим) и выполнение соответствующих заданий;
- работу над отдельными темами учебных дисциплин (модулей) в соответствии с учебно-тематическими планами;
- выполнение контрольных работ;
- подготовку ко всем видам промежуточных и итоговых контрольных испытаний.

1.1. Подготовка к лабораторным занятиям

Для того чтобы лабораторные занятия приносили максимальную пользу, необходимо помнить, что упражнение и решение задач проводятся по рассмотренному на лекциях материалу и связаны, как правило, с детальным разбором отдельных вопросов лекционного курса. Следует подчеркнуть, что только после усвоения лекционного материала с определенной точки зрения (а именно с той, с которой он излагается на лекциях) он будет закрепляться студентом на лабораторных занятиях как в результате обсуждения и анализа лекционного материала, так и с помощью решения проблемных ситуаций, задач. При этих условиях студент не только хорошо усвоит материал, но и научится применять его на практике, а также получит дополнительный стимул (и это очень важно) для активной проработки лекции.

При самостоятельном решении задач нужно обосновывать каждый этап решения, исходя из теоретических положений курса. Если студент видит несколько путей решения проблемы (задачи), то нужно сравнить их и выбрать самый рациональный. Полезно до начала вычислений составить краткий план решения проблемы (задачи). Решение проблемных задач или примеров

следует излагать подробно, вычисления располагать в строгом порядке, отделяя вспомогательные вычисления от основных. Решения при необходимости нужно сопровождать комментариями, схемами, чертежами и рисунками.

Следует помнить, что решение каждой учебной задачи должно доводиться до окончательного логического ответа, которого требует условие, и по возможности с выводом. Полученный ответ следует проверить способами, вытекающими из существа данной задачи. Полезно также (если возможно) решать несколькими способами и сравнить полученные результаты. Решение задач данного типа нужно продолжать до приобретения твердых навыков в их решении.

1.2. Самостоятельное изучение материала тем

Конспект — наиболее совершенная и наиболее сложная форма записи. Слово «конспект» происходит от латинского «conspectus», что означает «обзор, изложение». В правильно составленном конспекте обычно выделено самое основное в изучаемом тексте, сосредоточено внимание на наиболее существенном, в кратких и четких формулировках обобщены важные теоретические положения.

Конспект представляет собой относительно подробное, последовательное изложение содержания прочитанного. На первых порах целесообразно в записях ближе держаться тексту, прибегая зачастую к прямому цитированию автора. В дальнейшем, по мере выработки навыков конспектирования, записи будут носить более свободный и сжатый характер.

Конспект книги обычно ведется в тетради. В самом начале конспекта указывается фамилия автора, полное название произведения, издательство, год и место издания. При цитировании обязательная ссылка на страницу книги. Если цитата взята из собрания сочинений, то необходимо указать соответствующий том. Следует помнить, что четкая ссылка на источник —

непременное правило конспектирования. Если конспектируется статья, то указывается, где и когда она была напечатана.

Конспект подразделяется на части в соответствии с заранее продуманным планом. Пункты плана записываются в тексте или на полях конспекта. Писать его рекомендуется четко и разборчиво, так как небрежная запись с течением времени становиться малопонятной для ее автора. Существует правило: конспект, составленный для себя, должен быть по возможности написан так, чтобы его легко прочитал и кто-либо другой.

Формы конспекта могут быть разными и зависят от его целевого назначения (изучение материала в целом или под определенным углом зрения, подготовка к докладу, выступлению на занятии и т.д.), а также от характера произведения (монография, статья, документ и т.п.). Если речь идет просто об изложении содержания работы, текст конспекта может быть сплошным, с выделением особо важных положений подчеркиванием или различными значками.

В случае, когда не ограничиваются переложением содержания, а фиксируют в конспекте и свои собственные суждения по данному вопросу или дополняют конспект соответствующими материалами их других источников, следует отводить место для такого рода записей. Рекомендуется разделить страницы тетради пополам по вертикали и в левой части вести конспект произведения, а в правой свои дополнительные записи, совмещая их по содержанию.

Конспектирование в большей мере, чем другие виды записей, помогает вырабатывать навыки правильного изложения в письменной форме важные теоретических и практических вопросов, умение четко их формулировать и ясно излагать своими словами.

Таким образом, составление конспекта требует вдумчивой работы, затраты времени и труда. Зато во время конспектирования приобретаются знания, создается фонд записей.

Конспект может быть текстуальным или тематическим. В текстуальном конспекте сохраняется логика и структура изучаемого произведения, а запись ведется в соответствии с расположением материала в книге. За основу тематического конспекта берется не план произведения, а содержание какой-либо темы или проблемы.

Текстуальный конспект желательно начинать после того, как вся книга прочитана и продумана, но это, к сожалению, не всегда возможно. В первую очередь необходимо составить план произведения письменно или мысленно, поскольку в соответствии с этим планом строится дальнейшая работа. Конспект включает в себя тезисы, которые составляют его основу. Но, в отличие от тезисов, конспект содержит краткую запись не только выводов, но и доказательств, вплоть до фактического материала. Иначе говоря, конспект — это расширенные тезисы, дополненные рассуждениями и доказательствами, мыслями и соображениями составителя записи.

Как правило, конспект включает в себя и выписки, но в него могут войти отдельные места, цитируемые дословно, а также факты, примеры, цифры, таблицы и схемы, взятые из книги. Следует помнить, что работа над конспектом только тогда будет творческой, когда она не ограничена текстом изучаемого произведения. Нужно дополнять конспект данными из других источников.

В конспекте необходимо выделять отдельные места текста в зависимости от их значимости. Можно пользоваться различными способами: подчеркиваниями, вопросительными и восклицательными знаками, репликами, краткими оценками, писать на полях своих конспектов слова: «важно», «очень важно», «верно», «характерно».

В конспект могут помещаться диаграммы, схемы, таблицы, которые придадут ему наглядность.

Составлению тематического конспекта предшествует тщательное изучение всей литературы, подобранной для раскрытия данной темы. Бывает, что какая-либо тема рассматривается в нескольких главах или в разных местах

книги. А в конспекте весь материал, относящийся к теме, будет сосредоточен в одном месте. В плане конспекта рекомендуется делать пометки, к каким источникам (вплоть до страницы) придется обратиться для раскрытия вопросов. Тематический конспект составляется обычно для того, чтобы глубже изучить определенный вопрос, подготовиться к докладу, лекции или выступлению на семинарском занятии. Такой конспект по содержанию приближается к реферату, докладу по избранной теме, особенно если включает и собственный вклад в изучение проблемы.

2. СРЕДСТВА ОЦЕНИВАНИЯ УРОВНЯ СФОРМИРОВАННОСТИ КОМПЕТЕНЦИЙ ОБУЧАЮЩИХСЯ ПРИ ИЗУЧЕНИИ ДИСЦИПЛИНЫ «СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА»

Вопросы для собеседования

1. История искусственного интеллекта.
2. Основные направления исследований в области искусственного интеллекта.
3. Модели представления знаний. Вывод, основанный на знаниях.
4. Основные понятия теории нечетких знаний.
5. Этапы развития интеллектуальных систем.
6. Перспективы развития интеллектуальных систем.
7. Приведите примеры на каждую модель представления знаний.
8. Опишите шкалы оценивания нечетких знаний.
9. Основные понятия и структура экспертных систем.
10. Классификации систем, основанных на знаниях.
11. Технология проектирования и разработки интеллектуальных систем.
12. Чем экспертные системы отличаются от базы данных?
13. Назовите признаки, по которым классифицируются системы искусственного интеллекта.

14. Назовите основные особенности в разработке интеллектуальных систем.
15. Этапы разработки экспертных систем.
16. Разработка прототипа экспертной системы.
17. Коллектив разработчиков интеллектуальных систем.
18. Какие этапы следуют после разработки промышленного варианта?
19. Как осуществляется взаимодействие всех разработчиков экспертной системы?
20. Назовите характерные психологические черты каждого из разработчиков экспертной системы.
21. Поле знаний. Пирамида знаний.
22. Стратегии получения знаний.
23. Теоретические аспекты извлечения знаний.
24. Теоретические аспекты структурирования знаний.
25. Назовите отличия данных от знаний.
26. Чем отличаются понятия «извлечение» знаний от «приобретения»?
27. Назовите методы извлечения знаний.
28. Назовите отличия систем с естественно-языковым интерфейсом от информационных систем.
29. Перечислите основные характеристики систем с естественно-языковым интерфейсом.
30. Понятие и характеристика самообучающихся систем.
31. Классификация самообучающихся систем
32. Проектирование адаптивных обучающих систем.
33. Какие системы можно отнести к самообучающимся?
34. Назовите признаки, по которым классифицируются самообучающиеся системы.
35. Основные схемы адаптивных систем
36. Идентификация моделей

37. Цели, принципы и парадигмы технологий разработки программного обеспечения
38. Модели жизненного цикла интеллектуальных систем.
39. Языки представления знаний и проектирования искусственного интеллекта.
40. Инструментальные пакеты для искусственного интеллекта.
41. WorkBench-системы.
42. Какие технологии не используются при разработке систем искусственного интеллекта и почему?
43. Приведите примеры систем, соответствующих основным моделям жизненного цикла.
44. Онтологии и онтологические системы.
45. Программные агенты. Мультиагентные системы.
46. Проектирование и реализация агентов.
47. Информационный поиск в среде Интернет.
48. Как осуществляется интеллектуальный поиск в Интернет?

Критерии оценивания компетенций

Оценка «**зачтено**» выставляется студенту, если он твердо знает материал, грамотно и по существу излагает его, правильно применяет теоретические положения при решении практических вопросов и задач, владеет необходимыми навыками и приемами их выполнения. Допускаются некоторые неточности, недостаточно правильные формулировки в изложении программного материала, затруднения при выполнении практических работ.

Оценка «**не зачтено**» выставляется студенту, если он не знает значительной части программного материала, допускает существенные ошибки, неуверенно, с большими затруднениями выполняет практические задания.

3. ОТЧЕТНОСТЬ ПО ДИСЦИПЛИНЕ

Успеваемость студентов по дисциплине оценивается в ходе текущего контроля успеваемости и промежуточной аттестации.

Промежуточная аттестация

Процедура зачета как отдельное контрольное мероприятие не проводится, оценивание знаний обучающегося происходит по результатам текущего контроля.

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

Перечень основной литературы:

1. Исаев, С. В. Интеллектуальные системы : учебное пособие / С. В. Исаев, О. С. Исаева. — Красноярск : Сибирский федеральный университет, 2017. — 120 с. — ISBN 978-5-7638-3781-0. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: https://www.iprbookshop.ru/84365.html (дата обращения: 16.01.2025). — Режим доступа: для авторизир. пользователей.

2. Представление знаний в информационных системах : учебное пособие / Ю. Ю. Громов, О. Г. Иванова, М. Ю. Серегин [и др.]. — Тамбов : Тамбовский государственный технический университет, ЭБС АСВ, 2012. — 169 с. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: https://www.iprbookshop.ru/64163.html (дата обращения: 16.01.2025). — Режим доступа: для авторизир. пользователей.

Перечень дополнительной литературы:

1. Загорулько, Ю. А. Инженерия знаний : учебное пособие / Ю. А. Загорулько, Г. Б. Загорулько. — Новосибирск : Новосибирский государственный университет, 2016. — 93 с. — ISBN 978-5-4437-0452-4. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: https://www.iprbookshop.ru/93454.html (дата обращения: 24.07.2024). — Режим доступа: для авторизир. пользователей.

2. Цильковский, И. А. Методы анализа знаний и данных : конспект лекций / И. А. Цильковский, В. М. Волкова. — Новосибирск : Новосибирский государственный технический университет, 2010. — 68 с. — ISBN 978-57782-1377-7. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: https://www.iprbookshop.ru/45385.html (дата обращения: 16.01.2025). — Режим доступа: для авторизир. пользователей.