

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам по дисциплине

"Интеллектуальные системы и технологии"

Содержание

1. Основы Scilab	3
2. Простые вычисления в Scilab	15
3. Пространственные кривые в Scilab	22
4. Графика поверхностей в Scilab	25
5. Решение системы линейных уравнений в Scilab	30
6. Решение нелинейных уравнений в Scilab	32
7. Работа с полиномами	37
8. Операции с нечеткими множествами	39
9. Моделирование нечетких систем	51
10. Алгоритм нечеткой кластеризации	60
11. Обучение классификации нейронной сети	64
12. Исследование генетического алгоритма	68

Введение

Лабораторный цикл содержит 7 работ по изучению программирования с использованием математической системы Scilab

Цикл может быть использован в учебных курсах для аспирантов, а также в дисциплине "Методы искусственного интеллекта" для направления 09.03.02

Рекомендуемая литература:

1. Алексеев Е. Scilab. Решение инженерных и математических задач. М.: ALT Linux; БИНОМ. Лаборатория знаний. 2008, 260с.
2. Андриевский Б., Фрадков А. "Элементы математического моделирования в программных средах MATLAB 5 и Scilab" СПб.: Наука, 2001. 286с.
3. Chandler G. Stephen R. Introduction to Scilab. 2002, 27с.
4. Domanie de Voluceau. Introduction to Scilab. Scilab Groupe. 125с.
<http://www./Intro.pdf>.
5. Domanie de Voluceau. Scilab Reference Manual. Scilab Groupe. 700с.
<http://www./manual.pdf>.
6. Gomez C. Communication Toolbox. 12с. <http://www./comm.pdf>.
7. Domanie de Voluceau. Guide for Developers. Scilab Groupe. 29с.
<http://www./Internals.pdf>.
8. Domanie de Voluceau. InterSci. A scilab interfacing tool. Scilab Groupe. 14с.
<http://www./Intersci.pdf>.
9. Nikoukbah K. LMI Tool: a Package for LMI Optimisation in Scilab. 16с.
<http://www./lmi.pdf>.
10. Gomez C., Goursat M. Metanet User's Guide and Tutorial. 19с.
<http://www./metanet.pdf>.
11. Nikoukbah K. A Dynamic System Builder and Simulator. User's Guide. 15с.
<http://www./scicos.pdf>.
12. Domanie de Voluceau. Signal Processing with Scilab. Scilab Groupe. 205с.
<http://www./signal.pdf>.

Содержание отчета по каждой работе:

- Название работы, задание в соответствии с вариантом.
- Программа.
- Результаты выполнения программы на ПК.
- Выводы.

1. Основы Scilab

Предметная область

Знакомство с СКМ Scilab Освоение интерфейса, просмотр демосов, работа в режиме калькулятора.

Контрольные вопросы

1. Структура окна системы Scilab.
2. Команды пункта "File" системного меню.
3. Команды пункта "Edit" системного меню.
4. Команды пункта "Preference" системного меню.
5. Команды пункта "Control" системного меню.
6. Команды пункта "Editor" системного меню.
7. Команды пункта "Applications" системного меню.
8. Команды пункта "?" системного меню.
9. Правила ввода команд.
10. Правила ввода функций и операндов.
11. Правила ввода выражений.
12. Правила ввода комментариев.
13. Правила просмотра результатов операций.

Задания к работе

Задание 1. Изучить интерфейс Scilab.

Задание 2. Ознакомиться с демонстрационными примерами Scilab.

Задание 3. Выполнить в режиме калькулятора следующие действия:

- Ввод исходных операндов.
- Выполнить над операндами 1 и 2 операцию 1.
- Выполнить над результатом и операндом 1 операцию 2.
- Выполнить над результатом и операндом 2 операцию 3.
- Возвести операнд 1 почленно в степень 3.

Варианты заданий

№	Операнд 1	Операнд 2	Операции		
			1	2	3
1.	V=[12 34 61 45]	v = 34	*	./	+
2.	V=[80 67 34 11]	v = 43	/	.*	-
3.	V=[19 77 45 11]	v = -5	+	.\	/
4.	V=[11 98 67 45]	v = 7	-	.*	/
5.	V=[67 34 67 45]	v = -12	+	.\	*
6.	V=[18 36 45 45]	v = 10	/	./	-
7.	V=[55 43 18 45]	v = 44	/	.*	/
8.	V=[32 28 55 45]	v = 87	*	-	/

Методические указания

В Scilab все данные рассматриваются, как матрицы. Тип результата определяется автоматически по виду выражения.

В идентификаторах высота буквы имеет значение. Рекомендуется для имен простых переменных выбирать строчные буквы, а для структурированных (векторы и массивы) - прописные.

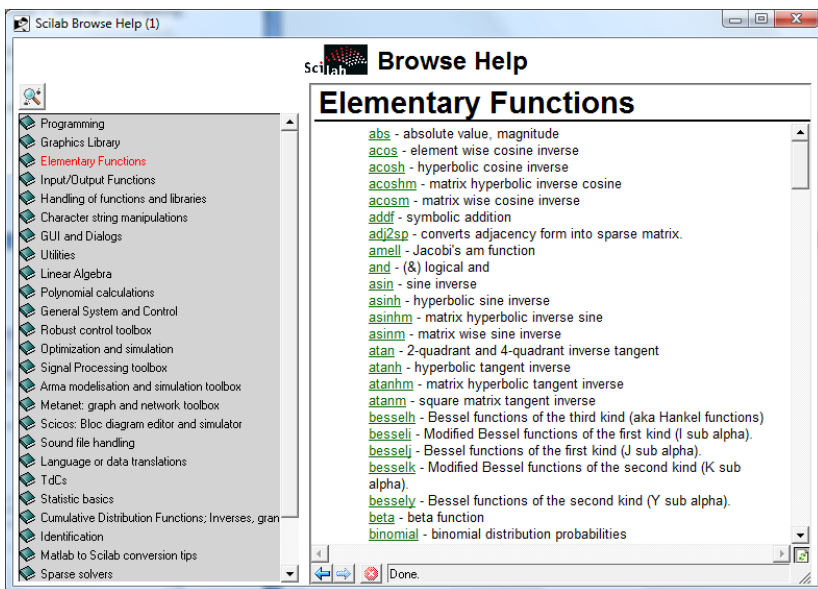
Векторы вводятся в квадратных скобках, компоненты вектора разделяются запятыми (или пробелами). Например, V=[1,2,3].

Матрицы вводятся в квадратных скобках, внутри размещаются векторы строк, разделенные знаком точка с запятой (;). Например, V=[1,2,3;4,5,6;7,8,9].

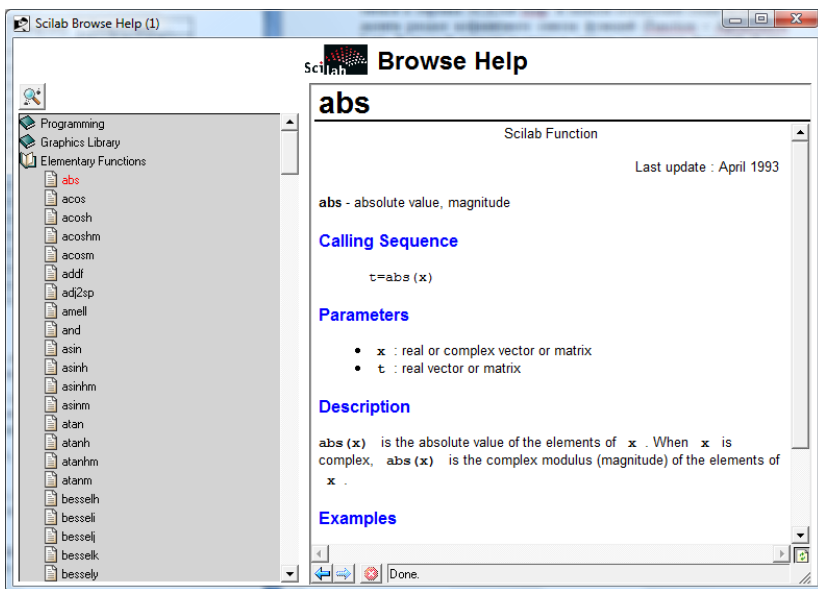
Если данные не умецаются в строке, строку можно отобразить в нескольких строках, используя разделитель в виде многоточия (не менее трех точек).

Значение π задается системной константой с именем %pi.

Для правильного вызова встроенных функций рекомендуется обратиться к справке Scilab Help. В панели оглавления слева нужно выделить раздел, в примере список элементарных функций (Elementary Functions). В правой панели отображается список доступных функций.



При открытии раздела отображается список доступных функций. В нем нужно найти и выделить нужную функцию. В правой панели отображается описание выбранной функции с форматом обращения и примерами использования. Ниже в качестве примера показано обращение к справке по функции взятия модуля $\text{abs}(X)$. Тот же результат достигается при щелчке по функции в правом поле.



В Scilab возможны два режима работы:

- В командном окне, как с калькулятором. В этом случае каждое действие сразу же выполняется.
- В редакторе программ. В этом случае программа вводится, как обычно, а выполняется по команде встроенного отладчика.

При работе в режиме калькулятора 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, результат выводится под именем этой переменной в следующей строке. Векторы выводятся в строке с пробелами, матрицы - построчно, каждая содержит вектор строки.

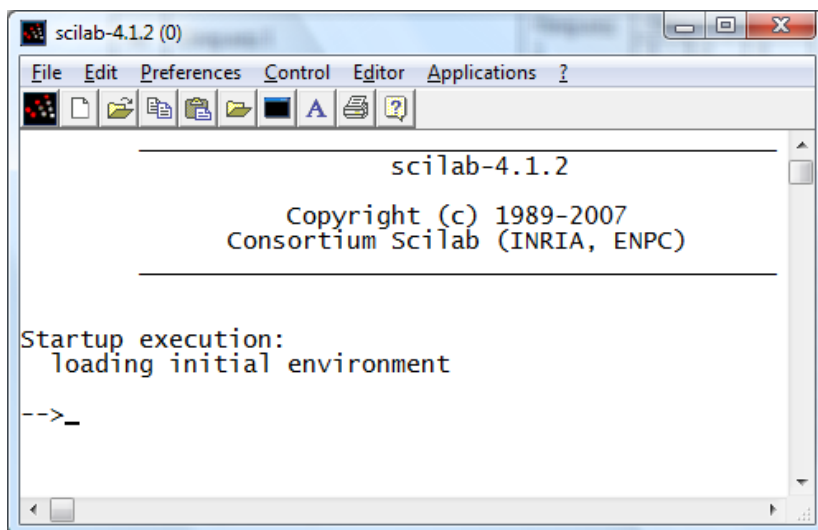
При работе с программой неграфические результаты выводятся в окно командной строки.

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 (==).

Пример 1

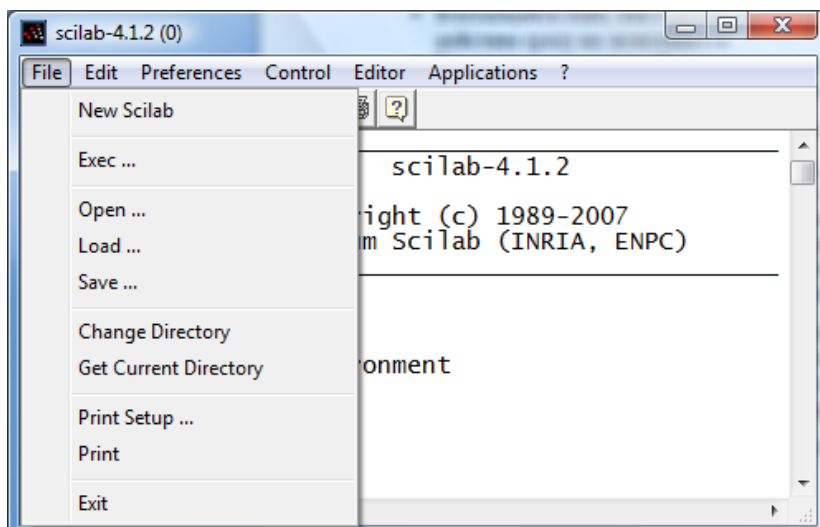


Пункты панели инструментов в порядке слева направо:

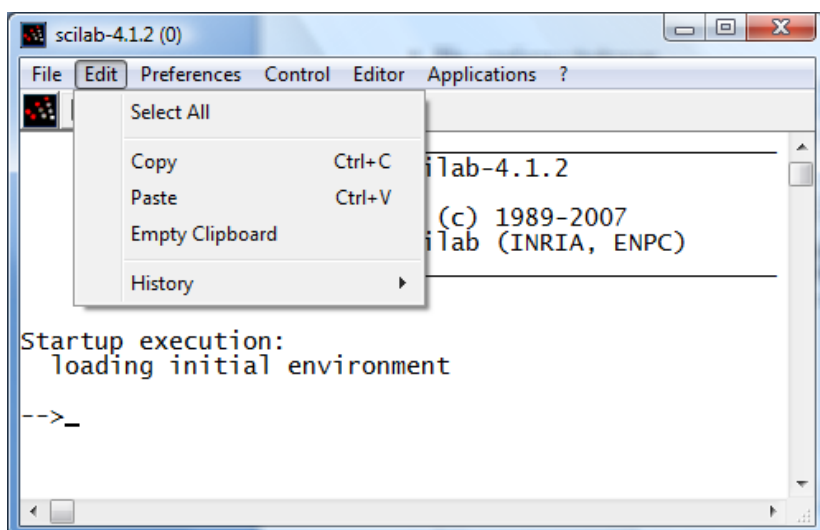
- New Scilab - новая Scilab.
- Open Scilab - открыть редактор Scilab.
- Open File - открыть файл.
- Copy - копировать.
- Paste - вставить.
- Change directory - изменить каталог.
- Scilab Output – Консоль Scilab в отдельном окне.
- Print – печать.
- Scilab Help – справка.

Пункты главного меню:

File – работа с файлами. Определены средства работы с файлами.

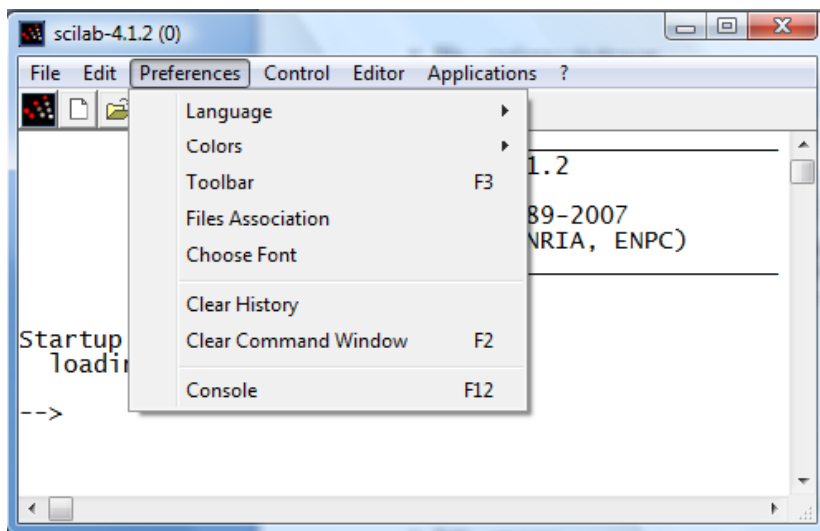


Edit – правка. Определены стандартные средства редактирования.

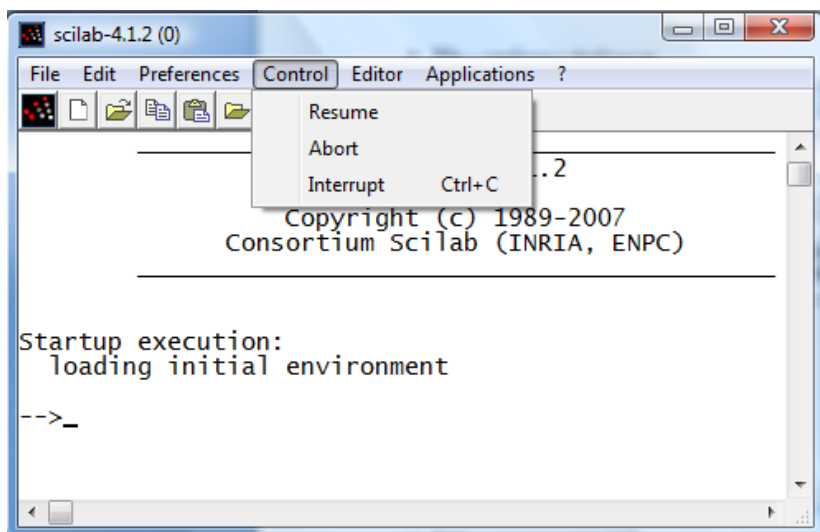


Preferences – предпочтения. Содержит команды: Language (выбрать язык, английский или французский), Colors (цвета текста и фона), Toolbar (отображать ли панели инструментов), Files Association (ассоциированные файлы по расширениям), Choose font (выбрать шрифт), Clear History (очистить историю)

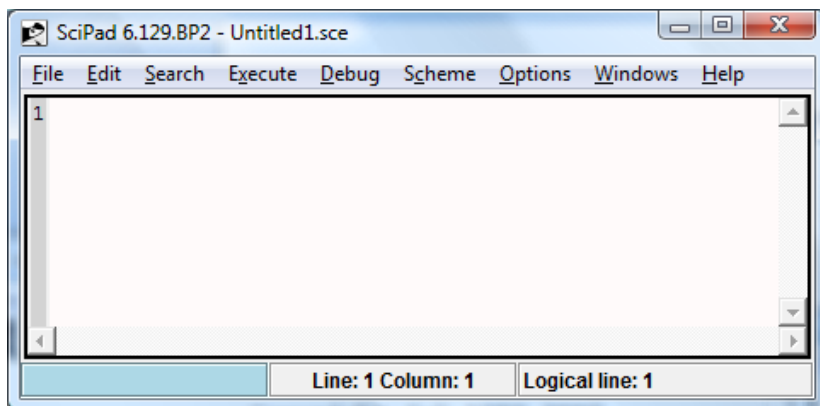
команд), Clear Command Window (очистить командное окно), Console (консоль).



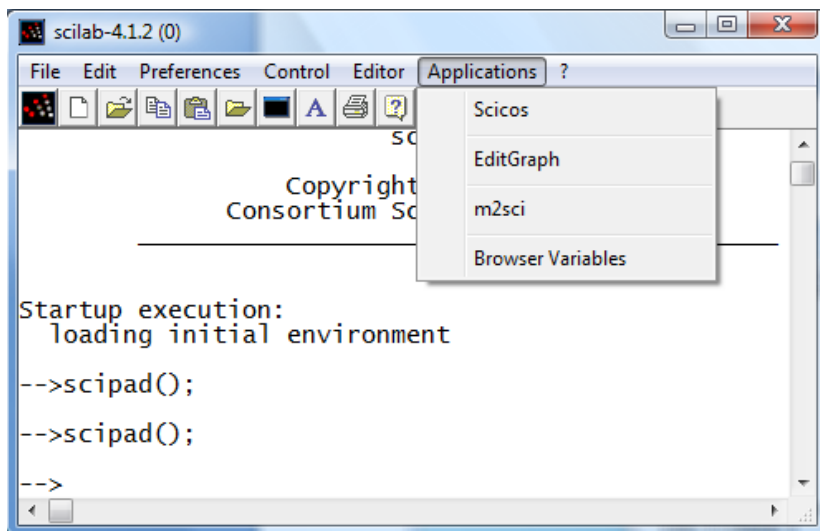
Control – управление. Содержит команды: Resume (повторить), Abort (отменить), Interrupt (прервать).



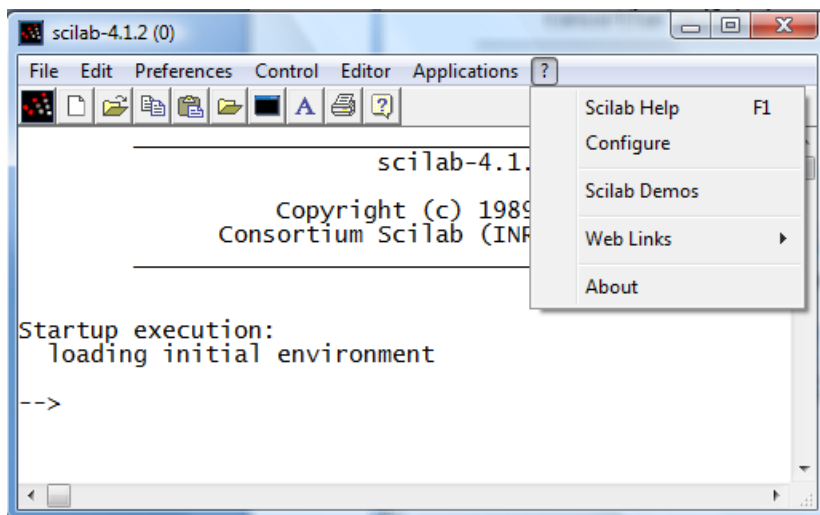
Editor – редактор. Открывается окно создания исполняемого файла с расширением .sce.



Applications – приложения. Выбираемые средства: Scicos (программа имитационного моделирования), EditGraph (графический редактор), m2sci (преобразователь m-файла MATLAB в sce-файл Scilab), Browser Variables (браузер переменных, новый или старый).

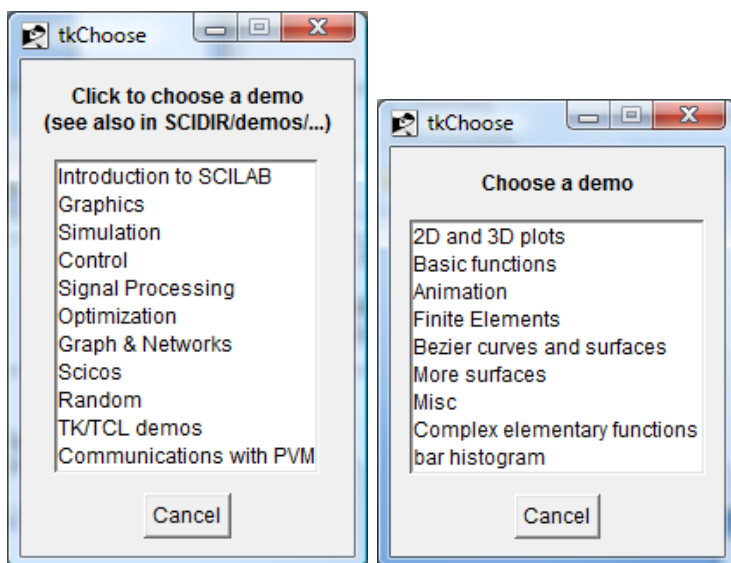


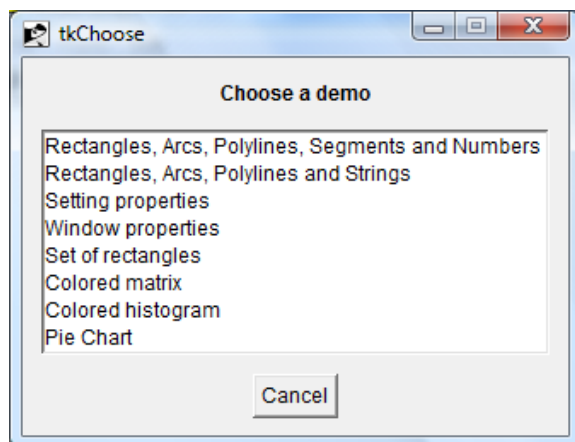
? – справка. Доступные команды: Scilab Help (Справка Scilab), Configure (Конфигурация), Scilab Demos (демонстрационные программы), Web Links (ссылки Интернета), About (о программе).



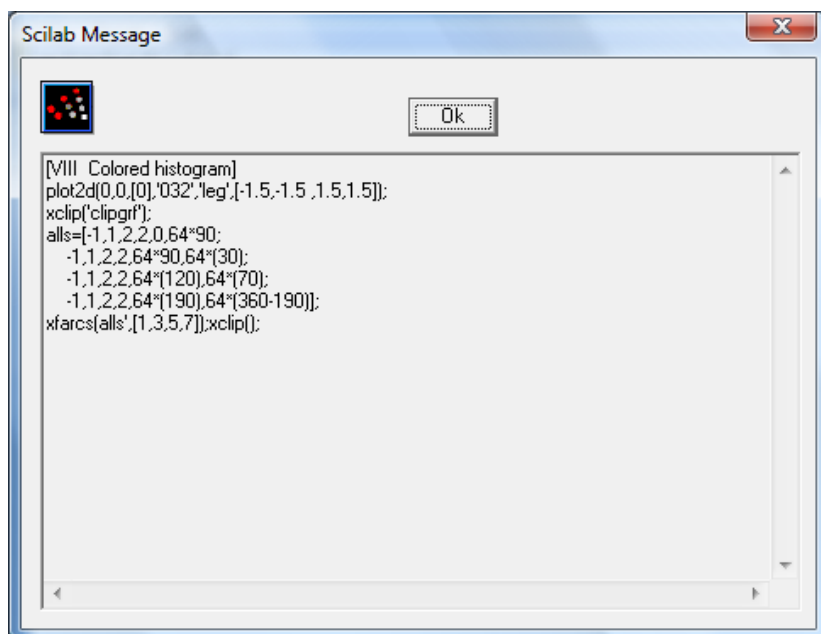
Пример 2

Для обращения к демонстрационным программам Scilab нужно использовать команду `?=>Scilab Demos`. Открываются иерархически связанные окна выбора программы. В примере последовательно выбираются Graphics (графика), Basic functions (основные функции), Pie Chart (Нарезанный торт).

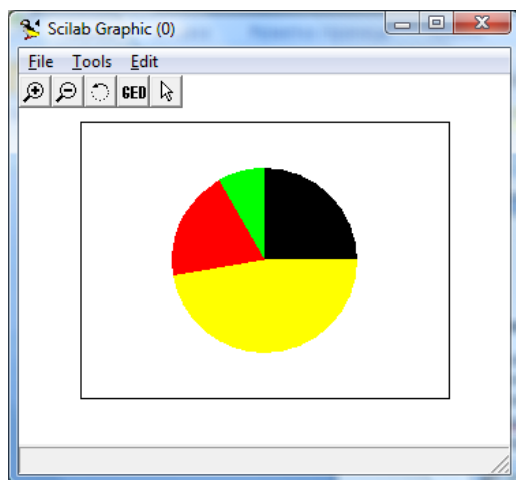




После выбора отображается окно сообщений Scilab с кодом исполняемой программы.



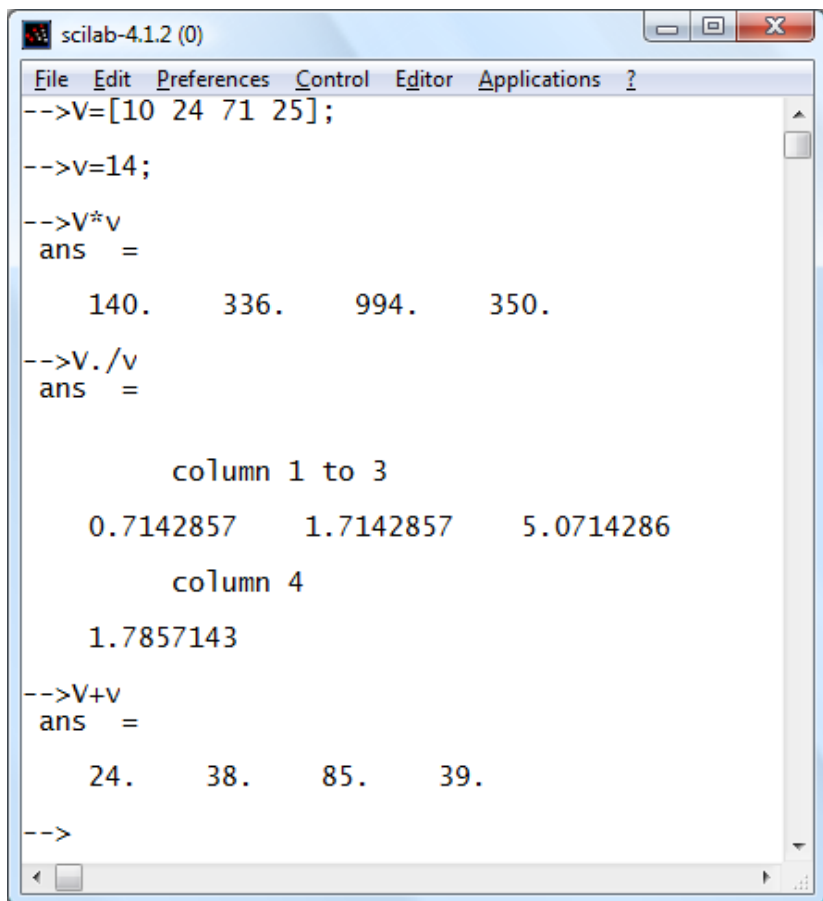
Нажатие кнопки Ok приводит к отображению графика, рисуемого по этой программе:



Пример 3

Операнд 1	Операнд 2	Операторы		
		1	2	3
$V=[10 \ 24 \ 71 \ 25]$	$v = 14$	*	./	+

Командное окно Scilab после выполнения команд:



The image shows a screenshot of the Scilab-4.1.2 (0) window. The window has a menu bar with 'File', 'Edit', 'Preferences', 'Control', 'Editor', 'Applications', and a help icon. The main area contains the following code and output:

```
-->V=[10 24 71 25];  
  
-->v=14;  
  
-->V*v  
ans =  
  
    140.    336.    994.    350.  
  
-->V./v  
ans =  
  
      column 1 to 3  
    0.7142857    1.7142857    5.0714286  
      column 4  
    1.7857143  
  
-->V+v  
ans =  
  
    24.    38.    85.    39.  
  
-->
```

2. Простые вычисления в Scilab

Предметная область

Для программирования сценариев в Scilab можно использовать командное окно или редактор Scipad. Редактор предпочтительнее, так как он содержит встроенный отладчик.

В работе программируются вычисления двух функций. Результаты выводятся в виде двумерных графиков с использованием графических функций высокого уровня.

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.

Задания к работе

Задание 1.

- Ввести в коде программы текст в виде комментария, как заглавие программы.
- Ввести исходные данные.
- Задать изменение аргумента.
- Вычислить значения функций 1 и 2 для аргумента в заданном интервале.
- Вывести графики функций одновременно на одном графике в декартовых координатах. Для разных графиков использовать разный тип линий.

Задание 2 . Повторить задание1, но графики функций вывести в двух подокнах на одном графике. Графики в столбиковом формате.

Задание 3 . Повторить задание1, но графики функций вывести в четырех подокнах с разными стилями линий на одном графике. Использовать функции plot2d, plot2d2, plot2d3, plot2d4.

Варианты заданий

№	Функция 1	Функция 2	a	b	h
1.	$y = \sin(x)$	$z = \exp(x+3)/5000 - 1$	-2π	2π	$\pi/20$
2.	$y = \cos(x)$	$z = 0.2\exp(3 - x) - 0.6$	-2π	2π	$\pi/20$
3.	$y = \lg(x) + 0.1$	$z = (1+x)^6$	-2π	2π	$\pi/20$
4.	$y = (x^2 - 1)/15$	$z = 1 + \sin(x)$	-2π	2π	$\pi/20$
5.	$y = (x^3 - 2)/15$	$z = 5\cos(x)$	-2π	2π	$\pi/20$
6.	$y = x^2 - 10$	$z = 0.025\exp(-1.2x)$	-5	5	1
7.	$y = 3\sin(x)$	$z = 0.015x^3$	-5	5	1
8.	$y = 4\sin(x)$	$z = 0.05x^2$	1	10	1

Методические указания

Текстовые пояснения в программу вводятся, как комментарий. Он начинается с символов //, которые располагается в первой позиции строки. Комментарий - это текст! В него не надо включать символы операций.

Для формирования XY графика необходимо:

- Задать аргумент в формате $x = \langle \text{начало} \rangle : \langle \text{шаг} \rangle : \langle \text{конец} \rangle$.
- Вычислить функцию, например, $y = f(x)$.

Вывести график процедурой `plot(x,y,s)`. Процедура рисует график прямыми линиями между вычисленными точками. Здесь `s` - строковая константа, задающая параметры линии (цвет, тип точки, тип линии), ее можно пропускать (тогда параметры выбираются по умолчанию). Определены следующие значения `s`:

Цвет линии		Тип точки		Тип линии	
y	желтый	.	точка	-	сплошная
m	фиолетовый	o	кружок	:	двойной пунктир
c	голубой	x	крест	-.	штрих пунктир
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		

Если на одном графике нужно отобразить несколько функций, например, (например, $y_1 = f(x)$ и $y_2 = f(x)$) то они вначале вычисляются, а затем выводятся процедурой `plot(x,y1,'s1',x,y2,'s2'...)`. В качестве параметров для каждой функции следуют строки символов

делителей символы типа линии, типа точки, цвета линии в произвольной последовательности.

Для создания в графическом окне нескольких подокон для вывода графиков используется процедура `subplot(m,n,p)`, где `m` - число подокон в окне по горизонтали, `n` - число подокон по вертикали, `p` - номер используемого подокна (нумерация с 1).

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 `subplot(m,n,p)` и `bar(x,y)`.

Для формирования графика в форме с разными типами линий (сплошная, ступенчатая, вертикальные полосы, со стрелками) нужно использовать функции `plot2d`, `plot2d2`, `plot2d3`, `plot2d4`.

Пример 1

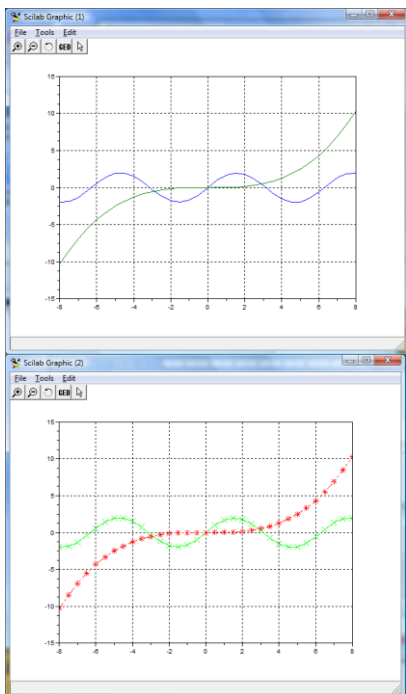
Задание

- | | |
|--------------------------------|-----------------------|
| • Функция 1 | $y = 2 \cdot \sin(x)$ |
| • Функция 2 | $z = 0.02 \cdot x^3$ |
| • 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); //Создать окно 1
plot(X,Y,X,Z);
xgrid() //Включить координатную сетку
// Вывод с выбираемыми типами в окно 2
scf(2); //Создать окно 2
plot(X,Y,'-gx',X,Z,'*r');
xgrid() //Включить координатную сетку
```

Запуск программы командой Execute => Load Into Scilab. Результат – графики в отдельных окнах.



Пример 2

Листинг программы

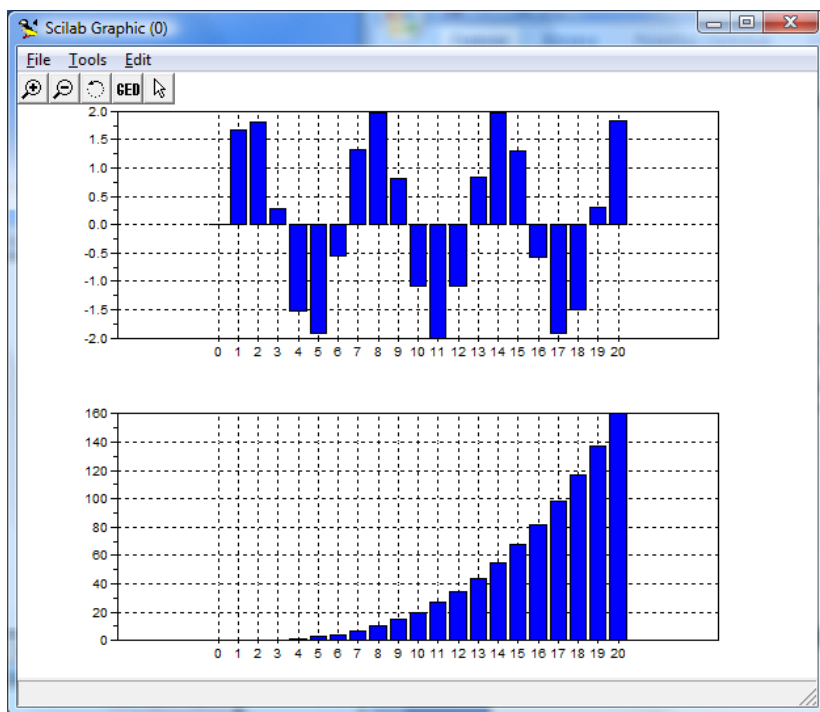
```
// Подокна и функция bar
a=0;
b=20;
h=1;
// Задание аргумента
X=a:h:b;
// Расчет функций
Y=2*sin(X);
Z=0.02*X.^3;
// Вывод Y столбиками в подокно 1
subplot(2,1,1);
```

```

bar(X,Y);
xgrid() //Включить координатную сетку
// Вывод Z столбиками в подокно 2
subplot(2,1,2);
bar(X,Z);
xgrid() //Включить координатную сетку

```

Запуск программы командой Execute => Load Into Scilab. Результат – графики в отдельных подокнах общего окна.



Пример 3

Листинг программы

```

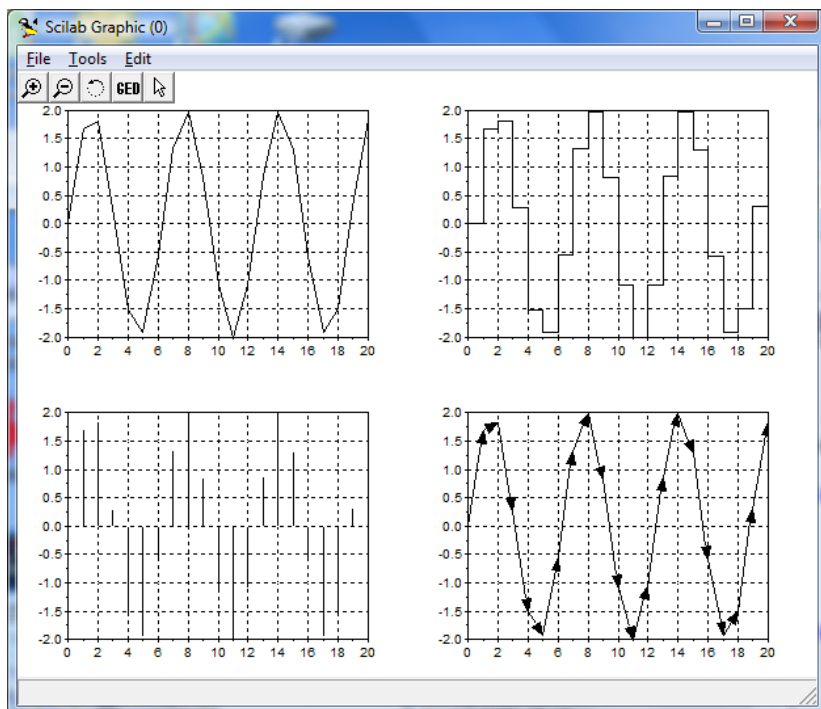
// Подокна и функции со стилями линий
a=0;
b=20;
h=1;

```

```

// Задание аргумента
X=a:h:b;
// Расчет функций
Y=2*sin(X);
//Вывод Y в подокно 1
subplot(2,2,1);
plot2d(X,Y);
xgrid()           //Включить координатную сетку
// Вывод Y ступенькой в подокно 2
subplot(2,2,2);
plot2d2(X,Y);
xgrid()           //Включить координатную сетку
// Вывод Y вертикальными полосками в подокно 3
subplot(2,2,3);
plot2d3(X,Y);
xgrid()           //Включить координатную сетку
// Вывод Y со стрелками в подокно 4
subplot(2,2,4);
plot2d4(X,Y);
xgrid()           //Включить координатную сетку
Запуск программы командой Execute => Load Into Scilab. Результат – графики
в отдельных подокнах общего окна.

```



3. Пространственные кривые в Scilab

Предметная область

Для программирования сценариев в Scilab можно использовать командное окно или редактор Scipad. Редактор предпочтительнее, так как он содержит встроенный отладчик.

В работе программируются вычисления функций для пространственных кривых. Результаты выводятся в виде трехмерных графиков с использованием графических функций высокого уровня `param3d` и `param3d1`.

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`.

Задания к работе

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

- Ввести исходные данные.
- Вычислить координаты пространственной кривой.
- Вывести кривую в виде трехмерного графика.

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

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

Варианты заданий. Заданы две функции расчета координат X, Y. В задании 1 использовать одну.

№	X	Y	t
1	$\sin(t), \sin(2t)$	$\cos(t), \cos(2t)$	$0:01:2^*\pi$
2	$\sin(1,5t), \sin(2t)$	$\cos(1,5t), \cos(2t)$	$0:01:3^*\pi$
3	$\sin(2t), \sin(3t)$	$\cos(2t), \cos(3t)$	$0:01:4^*\pi$
4	$\sin(2,5t), \sin(4t)$	$\cos(2,5t), \cos(4t)$	$0:01:5^*\pi$
5	$\sin(t), \sin(2t)$	$\cos(t), \cos(2t)$	$0:01:2,5^*\pi$
6	$\sin(1,5t), \sin(2t)$	$\cos(1,5t), \cos(2t)$	$0:01:3,5^*\pi$
7	$\sin(2t), \sin(3t)$	$\cos(2t), \cos(3t)$	$0:01:4,5^*\pi$
8	$\sin(2,5t), \sin(4t)$	$\cos(2,5t), \cos(4t)$	$0:01:5,5^*\pi$

Методические указания

Формирование задач. В работе предусмотрены 2 задачи, в каждой из которых вычисляются функции, описывающие пространственные кривые, и строятся поверхностные графики с использованием различных графических функций. В первой задаче рисуется одна кривая, во второй две.

Для формирования пространственного графика необходимо:

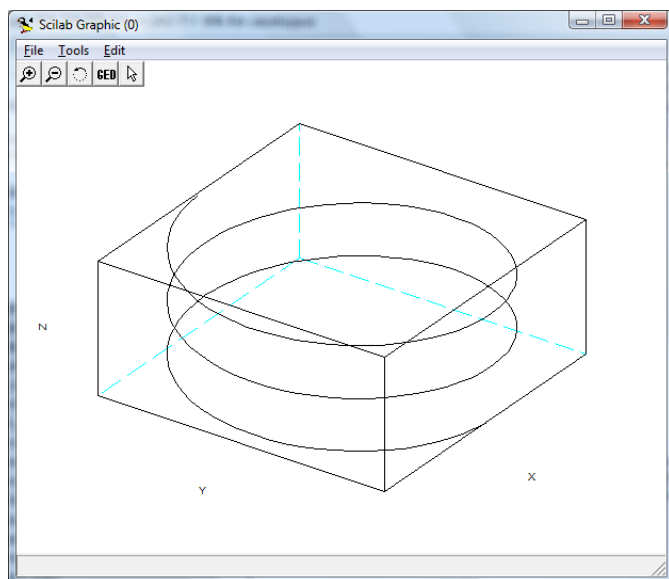
- задать число точек по координатам X и Y ,
- вычислить векторы координат X , Y .
- создать графическое окно и вывести туда график выбранного типа.

Пример 1

Кривая 1: $x=\sin(t)$, $y=\cos(t)$,
Используется функция `param3d`.

Листинг программы

```
// Функция param3d, пространственная кривая  
t=0:0.1:5*pi;  
param3d(sin(t),cos(t),t/10,35,45,"X@Y@Z",[2,3])
```



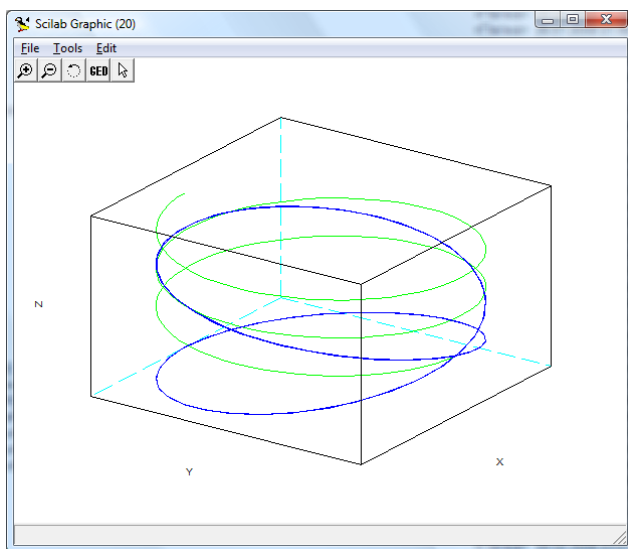
Пример 2

Кривая 1: $\sin(t)$, $\cos(t)$. Кривая 2: $\sin(2t)$, $\cos(2t)$.

Функция `param3d1` рисует 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])
```



4. Графика поверхностей в Scilab

Предметная область

Для программирования сценариев в Scilab можно использовать командное окно или редактор Scipad. Редактор предпочтительнее, так как он содержит встроенный отладчик.

В работе программируются вычисления функций для поверхностей. Результаты выводятся в виде трехмерных графиков с использованием графических функций высокого уровня plot3d, mesh, surf, contour.

Контрольные вопросы

1. Правила задания многомерных функций.
2. Связь двумерной функции с матрицей для вывода графиков.
3. Трехмерная графика в аксонометрии.
4. Трехмерная графика с функциональной раскраской.
5. Контурная графика.

Задания к работе

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

№	Функция	Пределы изменения	
		x	y
1.	$z = \sin(x)\cos(y)$	от -2π до 2π	от -2π до 2π
2.	$z = \sin(x/2)\cos(y)$	от -2π до 2π	от -2π до 2π
3.	$z = \sin(2x)\cos(y)$	от -2π до 2π	от -2π до 2π
4.	$z = \sin(x)\cos(y/2)$	от -2π до 2π	от -2π до 2π
5.	$z = \sin(x/2)\cos(2y)$	от -2π до 2π	от -2π до 2π
6.	$z = \sin(2x)\cos(2y)$	от -2π до 2π	от -2π до 2π
7.	$z = (1 + \sin(x)/x)(\sin(y)/y)$	от -2π до 2π	от -2π до 2π
8.	$z = (\sin(x)/x)\cos(y)$	от -2π до 2π	от -2π до 2π

Методические указания

Формирование задач. В работе предусмотрены 2 задачи, в каждой из которых вычисляется двумерная функция, описывающая объемную фигуру, и строятся поверхностные и контурные графики с использованием различных графических функций. В первой задаче каждый график выводится в свое окно, во второй в подокна 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, вычислить функцию $Z=f(X,Y)$,
- ввести номер графического окна, вывести туда 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)`,

Пример 1

Задание

$$\text{Функция } z = \frac{\sin(x)}{x} \cdot \frac{\sin(y)}{y}.$$

Пределы изменения аргументов $-2\pi \dots 2\pi$

Листинг программы

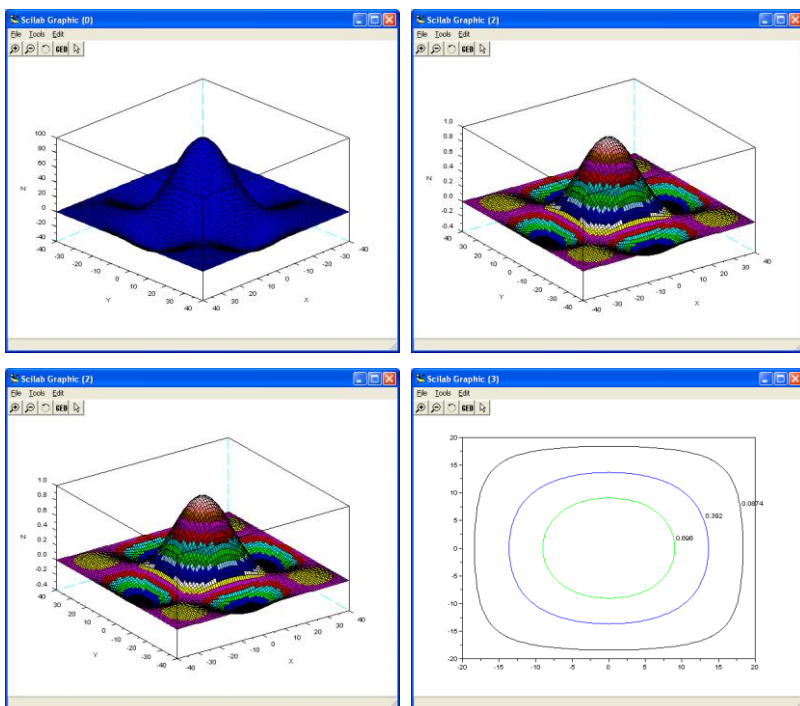
```
// Функции plot3d, mesh, surf, contour
N=40;
h=%pi/20;
// Расчет матрицы
for n=1:2*N+1
    if n==N+1 A(n)=1; else A(n)=sin(h*(n-N-1))/(h*(n-N-1)); end;
end;
for n=1:2*N+1
    for m=1:2*N+1
        Z(n,m)=A(n)*A(m);
    end;
end;
// Задание площадки
X=-N:1:N;
```

```

Y=-N:1:N;
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

```

Запуск программы командой Execute => Load Into Scilab. Результат – графики в отдельных окнах.



Пример 2

Листинг программы

```

// Функции plot3d, mesh, surf, contour. Используются подокна.
N=40;
h=%pi/20;

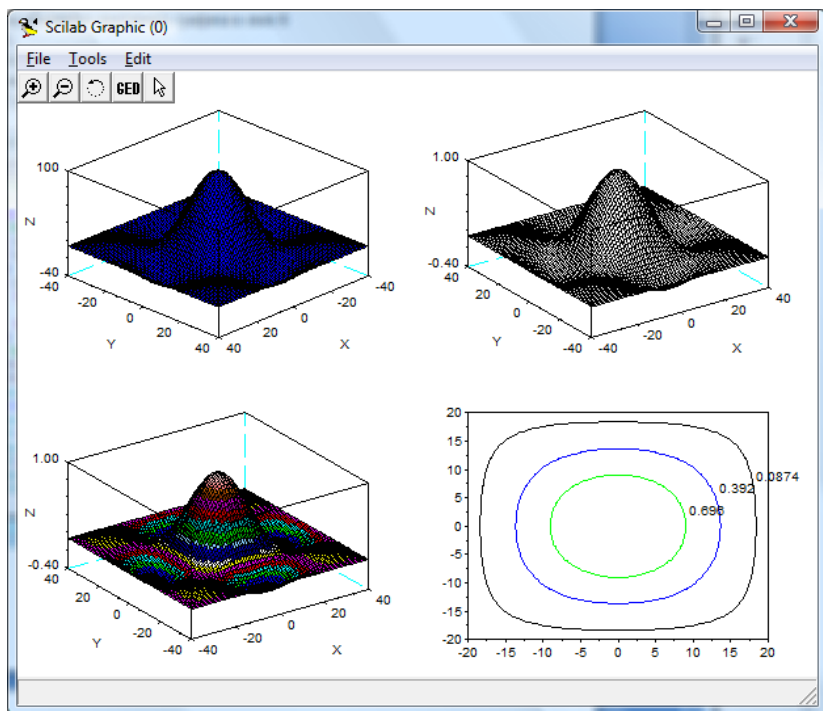
```

```

// 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:1:N;
Y=-N:1:N;
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);

```

Запуск программы командой Execute => Load Into Scilab. Результат – графики в отдельных подокнах общего окна.



5. Решение системы линейных уравнений в Scilab

Предметная область

Решение системы линейных уравнений в Scilab осуществляется с использованием матричного деления. Результаты выводятся в командное окно.

Контрольные вопросы

1. Решение системы линейных уравнений с использованием матричного деления.
2. Ввод системы уравнений.
3. Вывод полученного решения
4. Проверка решения..

Задание к работе

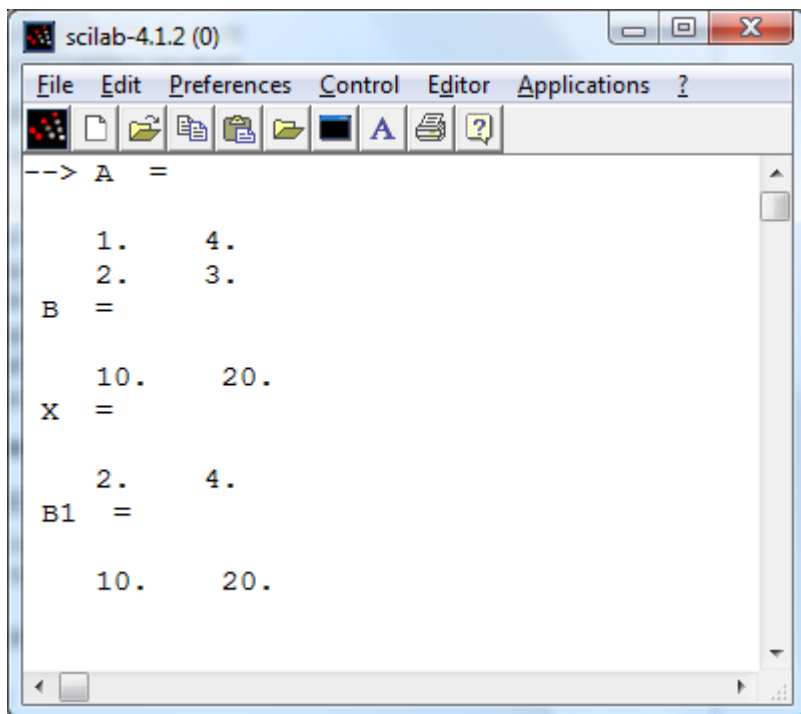
Задача 1. Решение системы линейных уравнений.

- Создать программу решения системы из N линейных уравнений в редакторе scipad.
- Задать матрицу A коэффициентов. Коэффициенты уравнений 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 $B1=X*A$. 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
```



6. Решение нелинейных уравнений в Scilab

Предметная область

Решение нелинейного уравнений $f(x)=0$ в Scilab осуществляется с использованием функции `fsolve`. Решение ищется в окрестности предполагаемого значения x_0 . Для его определения проводится локализация решений по предварительно построенному графику $f(x)$. Результаты выводятся в командное окно или в строку заголовка графика.

Контрольные вопросы

1. Задание функции пользователя.
2. Локализация решений уравнения.
3. Решение нелинейного уравнения с использованием функции `fsolve`.
4. Вывод полученных решений уравнения.
5. Локализация решений системы из двух уравнений.
6. Решение системы из двух уравнений.
7. Вывод полученных решений системы уравнений.

Задание к работе

Задача 1. Решение нелинейного уравнения.

- Создать программу решения нелинейного уравнений в редакторе `scilab`.
- В программе определить функцию $f_1(x)$.
- Вывести $y_1=f_1(x)$ в виде XY графика. По нему определить приближенно корни уравнения $y_1(x)=0$. Если корни на графике не просматриваются, то изменить пределы изменения аргумента и повторить операцию.
- Для каждого корня найти точное значение, используя функцию `fsolve`. Перед расчетами задать приближенное значение корня x_0 .
- Сформировать строку с результатами и вывести ее в заголовок окна графика.

Задача 2. Решение системы из двух нелинейных уравнений.

- Создать программу решения нелинейных уравнений в редакторе `scilab`.
- В программе определить функции $f_1(x)$, $f_2(x)$, $f_3(x)=f_2(x)-f_1(x)$.
- Вывести $u_3=f_3(x)$ в виде XY графика. По нему определить приближенно корни уравнения $u_3(x)=0$. Если корни на графике не просматриваются, то изменить пределы изменения аргумента и повторить операцию.

- Для каждого корня найти точное значение, используя функцию `fsolve`. Перед расчетами задать приближенное значение корня x_0 .
- Сформировать строку с результатами и вывести ее в заголовок окна графика.

Варианты заданий

№	f1(x)- полином 3-ей степени с коэффициентами a				f2(x)
	a3	a2	a1	a0	
1	0	-1	4	-1	$0.2\exp(x)-20$
2	0	2	-2	-15	$40 \cos(x) $
3	0	1	4	-1	$10\ln(x+5.5)$
4	0	9	-8	-70	$100 \sin(x) $
5	0	-4	4	50	$70\cos(x)$
6	.1	-5	4	40	$60\exp(0.1*x)-100$
7	.2	-3	2	30	$20\sin(2x)$
8	.3	-6	1	50	$\exp(x)\sin(2x)$

Методические указания

При решении нелинейного уравнения оно формируется из функций задания, как $f_1(x)=0$. При решении системы из двух нелинейных уравнений из функций задания формируется уравнение $f_3(x) = f_1(x) - f_2(x) = 0$.

Локализация корней. Уравнение или система уравнений может иметь несколько корней, каждый из которых ищется отдельно. При этом для каждого корня надо задать диапазон аргумента, в котором он находится (только один!).

Это делается путем локализации корня. Для этого надо просчитать значения функций в заданном интервале и построить их графики. Начальное значение для решения одного уравнения - точка пересечения графиком функции оси X. График выводится процедурой, в которой аргументы - переменная x и анализируемая функция. С помощью `grid on` график делается с координатной сеткой:

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

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

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

Функция `fsolve`. Используется для нахождения корня нелинейного уравнения. Формат этой функции:

<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

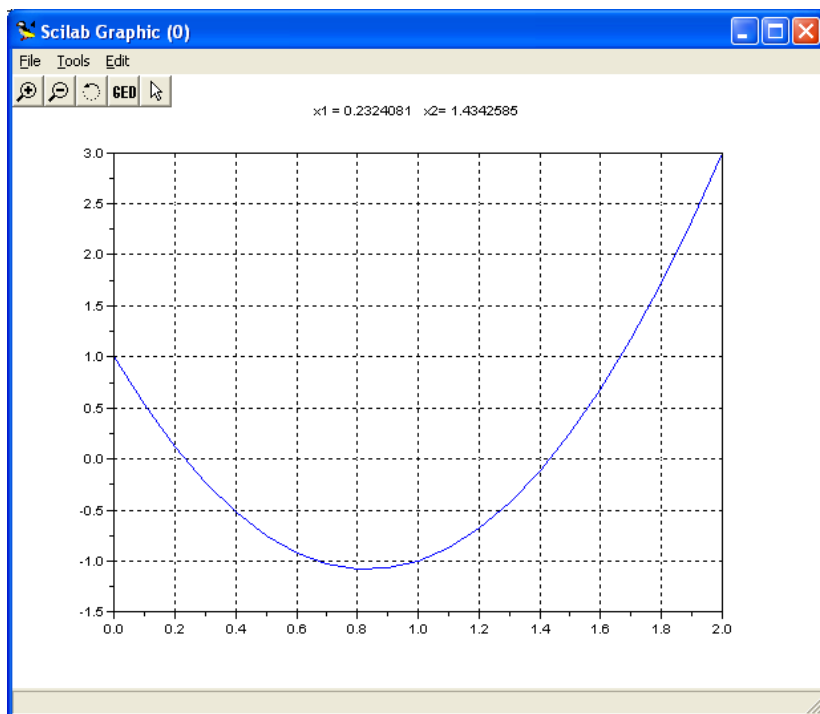
x0=1.4;

x2=fsolve(x0,f1) // Корень 2

rezult='x1 = '+string(x1)+' x2= '+string(x2);

title(rezult);

В программе ищутся 2 корня. Их приближенные значения 0.2 и 1.4 определены при пробном прогоне программы. Окончательный результат в окне графики.



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

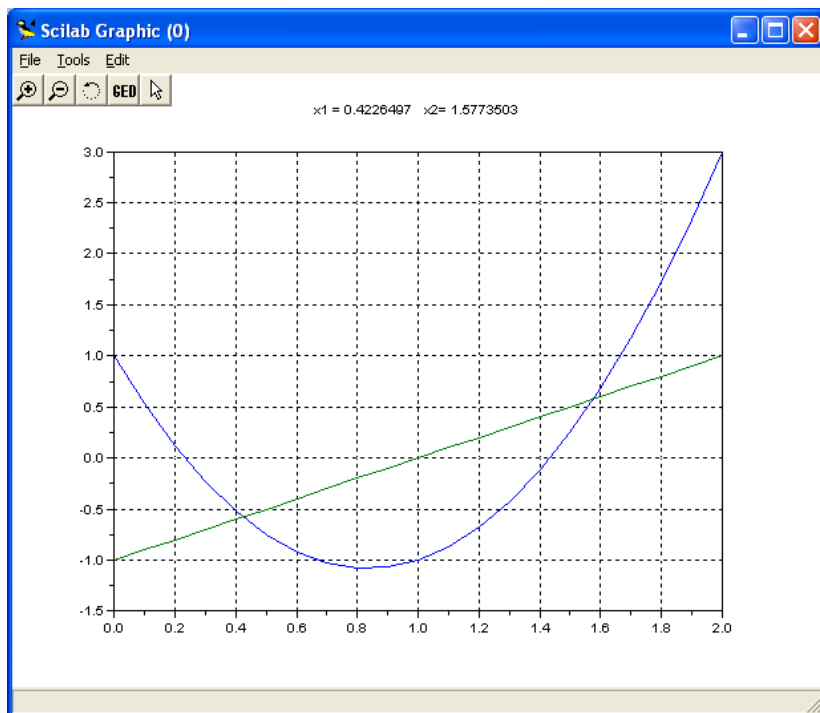
x0=1.5;

```

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

```

В программе ищутся 2 корня. Их приближенные значения 0.4 и 1.5 определены при пробном прогоне программы. Окончательный результат в окне графики.



7. Работа с полиномами

Предметная область

В Scilab определены удобные средства работы с полиномами:

- Функция `poly(V,'x','c')`. Признак режима 'c' задает интерпретацию вектора V , как набора коэффициентов полинома. Функция формирует полином от переменной 'x', коэффициенты которого определены вектором V . В векторе V позиции слева направо соответствуют степеням полинома, начиная с 0.
- Функция `poly(V,'x','r')`. Признак режима 'r' задает интерпретацию вектора V , как набора корней полинома. Корни полинома это корни уравнения $f(x)=0$, где $f(x)$ – полином.. Функция формирует полином от переменной 'x', корни которого определены вектором V .
- Функция `roots(p)`, возвращает корни полинома p .

Контрольные вопросы

1. Задание коэффициентов полинома.
2. Формирование полинома по коэффициентам степеней.
3. Корни полинома.
4. Задание корней полинома.
5. Вывод полученных результатов.
6. Вычисление корней полинома.

Задание к работе

Задача 1. формирование полинома по коэффициентам степеней и корням полинома.

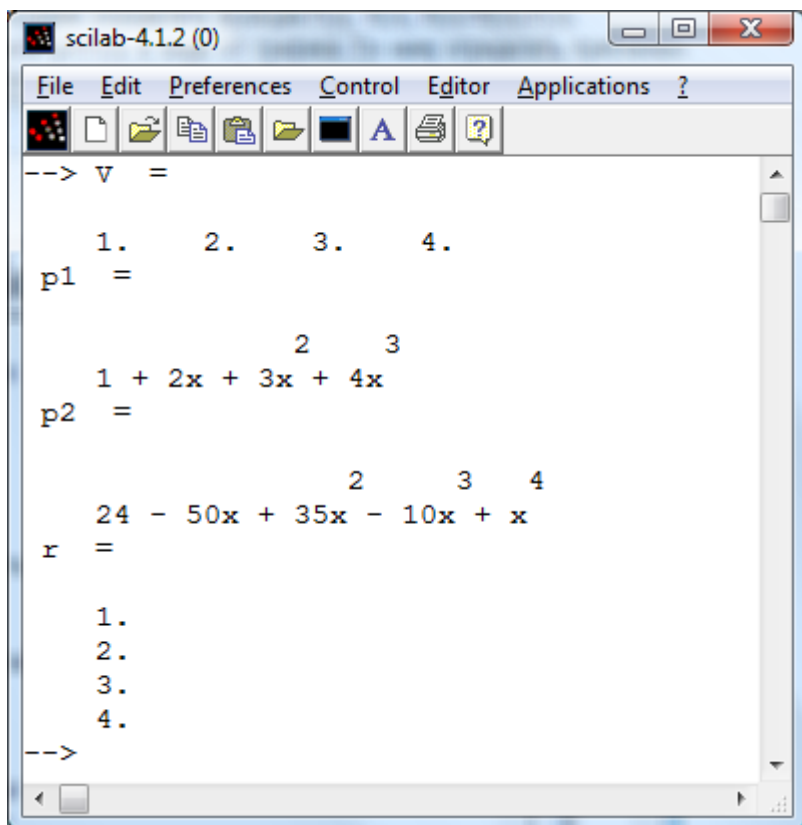
- Создать в редакторе scipad программу работы с полиномами.
- Задать вектор V .
- Добавить формирование полинома $p1$ по коэффициентам степеней.
- Добавить формирование полинома $p2$ по его корням.
- Для проверки $p2$ вычислить корни, используя функцию `roots(p2)`.

Пример 1

Листинг программы

```
// Работа с полиномами
V=[1,2,3,4]
// формирование по коэффициентам степеней
p1=poly(V,'x','c')
// Формирование по корням полинома
```

```
p2=poly(V,'x','r')
// Вычисление корней полинома
r=roots(p2)
```



Лабораторная работа № 8

Операции с нечеткими множествами

Цель работы: изучить методы построения нечетких множеств с использованием различных типов функций принадлежности. Ознакомиться с наиболее распространенными логическими операциями над нечеткими множествами.

Используемые в системе SciLAB типы функций принадлежности

Инструментарий нечеткой логики (ИНЛ) в составе пакета SciFLT содержит 11 встроенных типов функций принадлежности (ФП), формируемых на основе кусочно-линейных функций, распределения Гаусса, сигмоидной кривой, квадратических и кубических полиномиальных кривых.

К наиболее простым ФП (и в этом их главное достоинство) можно отнести треугольную и трапециевидную. Наименование треугольной ФП – ***trimf*** (***triangle membership function***). В параметрическом виде она представляет собой ни что иное, как набор трех точек, образующих треугольник.

Описание функции:

$$y = \text{trimf}(x, [a \ b \ c]).$$

Вектор x обозначает базовое множество, на котором определяется ФП. Величины a и c задают основание треугольника, b – его вершину.

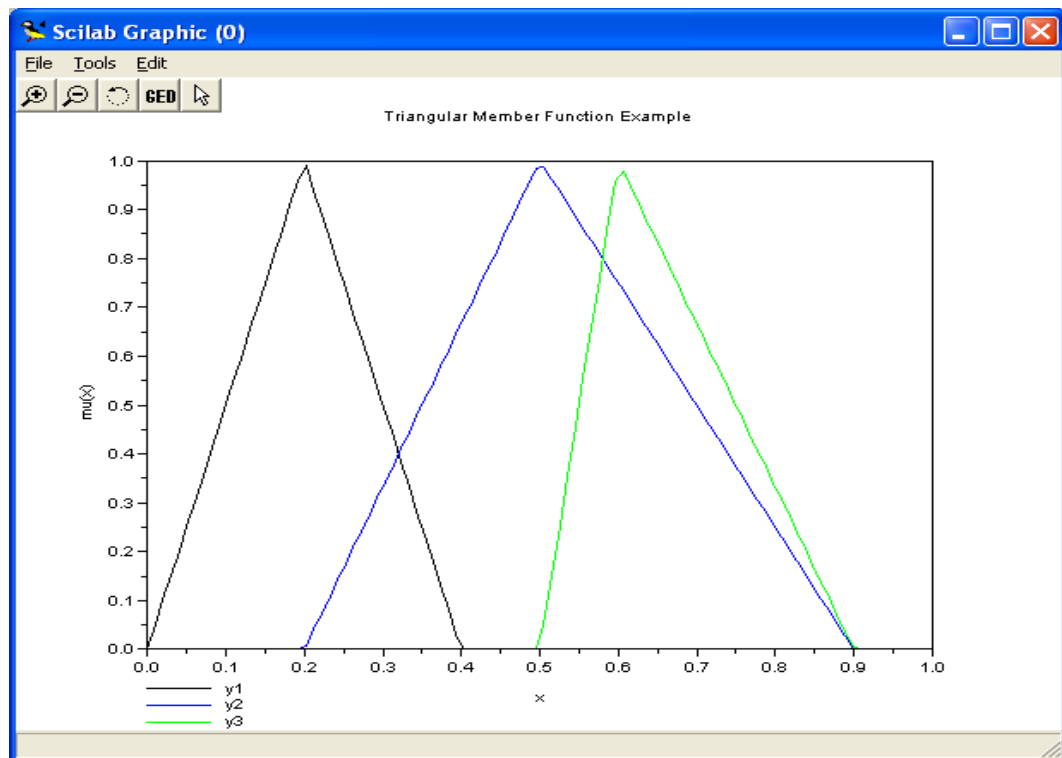


Рис. 1.1. Треугольные функции

В аналитическом виде треугольная ФП может быть задана следующим образом (рис. 1.1):

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

Далее рассмотрим примеры использования различных ФП в системе *SciFLT*. Примеры представляют собой фрагменты программ и комментариев на языке пакета *SCILAB*.

Пример 1.1. Программа использования ФП *trimf* (результат на рис.1.1)

```
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]);
xbasc();
//строим функции принадлежности
plot2d(x,[y1 y2 y3],leg="y1@y2@y3");
//подписываем рисунок
xtitle("Triangular Member Function Example","x","mu(x)");
```

Трапецевидная ФП, *trapmf* (*trapezoid membership function*), отличается от предыдущей функции лишь тем, что имеет верхнее основание.

Описание функции:

$$y = \text{trapmf}(x, [a \ b \ c \ d]),$$

где параметры a и d задают нижнее основание, b и c – верхнее основание трапеции (рис. 1.2).

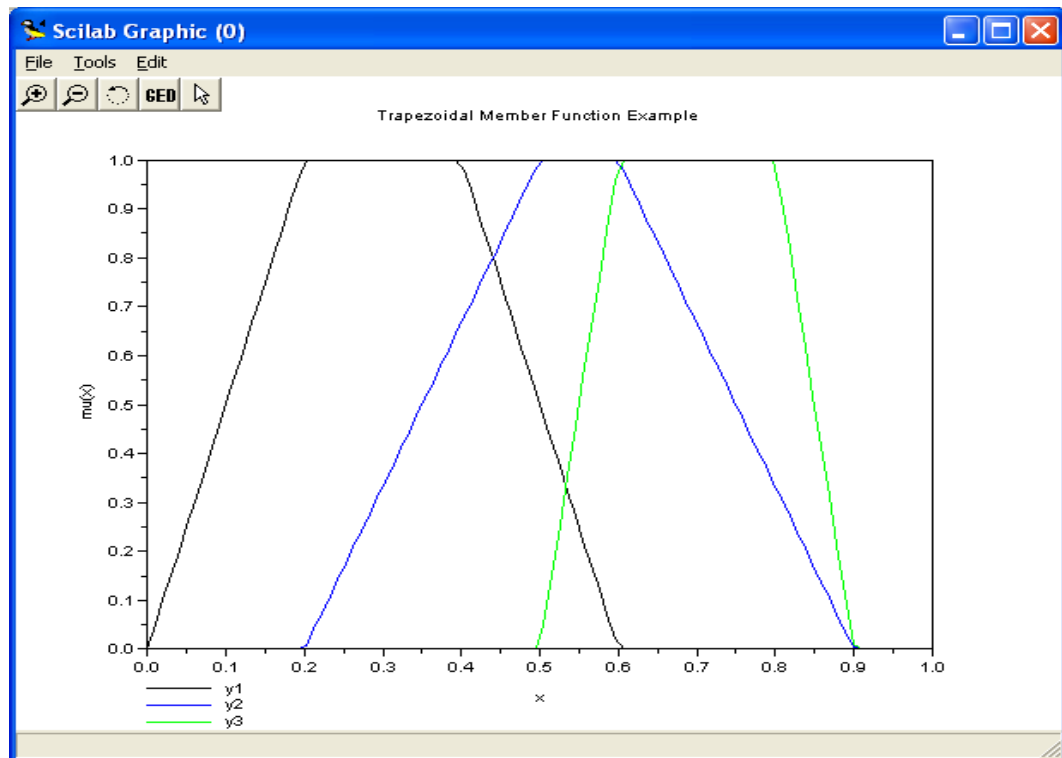


Рис.1.2. Трапециевидные функции принадлежности

Аналитическая запись трапециевидной функции имеет вид:

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

Пример 1.2. Программа использования ФП `trapmf` (результат на рис. 1.2)

```
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]);
xbasec(); plot2d(x,[y1 y2 y3],leg="y1@y2@y3");
xtitle("Trapezoidal Member Function Example","x","mu(x)");
```

На основе функции распределения Гаусса в *SCIFLT* можно построить ФП двух видов: простую ФП Гаусса и двухстороннюю ФП, образованную с помощью различных функций распределения Гаусса. Первая из них обозначается – *gaussmf*, а вторая – *gauss2mf*.

Описание функции:

$$y = \text{gaussmf}(x, [\sigma, c]).$$

Симметричная функция Гаусса зависит от двух параметров σ и c :

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

Описание двусторонней функции принадлежности:

$$y = \text{gauss2mf}(x, [\sigma_1, c_1, \sigma_2, c_2]).$$

Последнее выражение является комбинацией двух различных функций распределения Гаусса. Первая определяется параметрами σ_1 и c_1 и задает форму левой стороны, а вторая (параметры σ_2, c_2) – правой стороны ФП.

Если $c_1 < c_2$, то в этом случае функция **gauss2mf** достигает своего максимального значения на уровне 1. Иначе – максимальное значение функции меньше 1.

На рис. 1.3. представлены графики кривых **gaussmf** и **gauss2mf**, заданных в программе из примера 1.3.

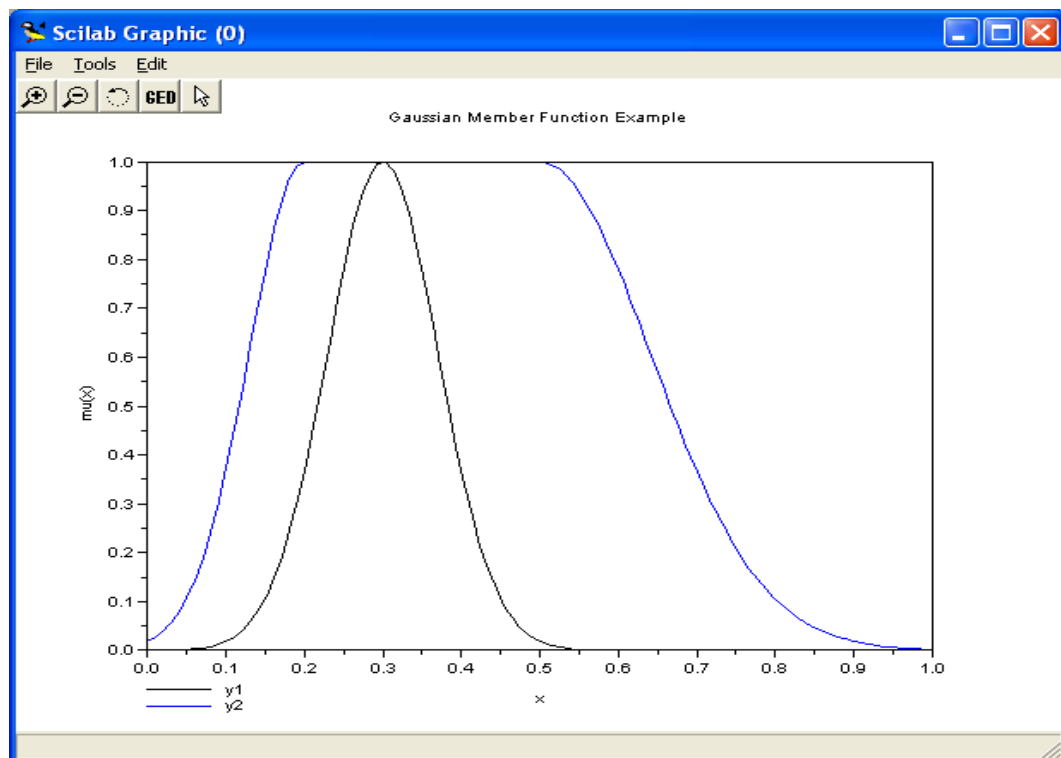


Рис. 1.3. Гауссовы кривые

Пример 1.3. Программа использования ФП **gaussmf**

```
x=linspace(0,1,100)';  
y1=gaussmf(x,[0.3 0.1]);  
y2=gauss2mf(x,[0.2 0.1 0.5 0.2]);  
xbasc();  
plot2d(x,[y1 y2],leg="y1@y2");  
xtitle("Gaussian Member Function Example","x","mu(x)");
```

Символ “ ’ ” в строке определения базового множества x показывает транспонированность базового множества.

Следующей функцией, которая позволяет представлять нечеткие субъективные предпочтения, является ФП «обобщенный колокол». Она обозначается ***gbellmf*** (*bell shape membership function*). Ее отличие от рассмотренных ранее ФП заключается в добавлении третьего параметра, что позволяет осуществлять плавный переход от нечеткого множества к четкому.

Описание функции:

$$y = \text{gbellmf}(x, [a \ b \ c]).$$

Функция «обобщенный колокол» имеет следующий аналитический вид:

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

где c определяет расположение центра ФП, a и b оказывают влияние на форму кривой (рис. 1.4).

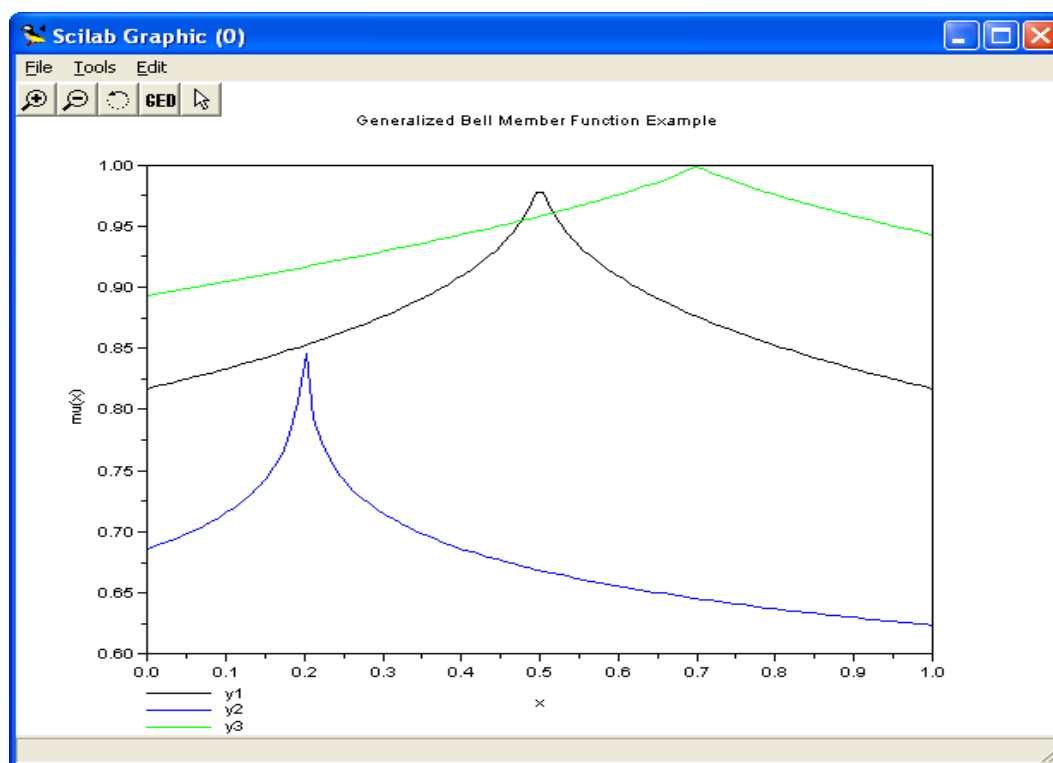


Рис. 1.4. Колоколообразные функции принадлежности

Пример 1.4. Программа использования ***gbellmf***

```
x=linspace(0,1,100)';  
y1=gbellmf(x,[0.5 10 0.5]);  
y2=gbellmf(x,[0.2 10 0.2]);  
y3=gbellmf(x,[0.7 10 0.8]);  
xbasc();
```

```
plot2d(x,[y1 y2 y3],leg="y1@y2@y3");
xtitle("Generalized Bell Member Function Example","x","mu(x)");
```

ФП на основе функции распределения Гаусса и ФП «обобщенный колокол» отличаются гладкостью и простотой записи и являются наиболее используемыми при описании нечетких множеств.

Но несмотря на то, что гауссовы и колоколообразные ФП обладают свойством гладкости, они не позволяют формировать асимметричные ФП. Для этих целей в *SciFLT* существует набор сигмоидных функций, которые могут быть открыты либо слева, либо справа в зависимости от типа функции. Симметричные и закрытые функции синтезируют с использованием двух дополнительных сигмоид. Основная сигмоидная ФП обозначается ***sigmf***, а дополнительные – ***dsigmf*** и ***psigmf***.

Описание основной сигмоидной функции:

$$y = \text{sigmf}(x, [a \ c]).$$

В аналитической форме сигмоидная функция ***sigmf*** записывается следующим образом:

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

В зависимости от знака параметра a рассматриваемая ФП будет открыта справа или слева (рис. 1.5), что позволит применять ее при описании таких нечетких понятий, как «очень большой», «крайне отрицательно» и т.д.

Описание дополнительной сигмоидной функции:

$$y = \text{dsigmf}(x, [a_1 \ c_1 \ a_2 \ c_2]).$$

ФП ***dsigmf*** зависит от четырех параметров a_1 , c_1 , a_2 и c_2 и определяется как разность двух сигмоидных функций: $f_1(x, a_1, c_1) - f_2(x, a_2, c_2)$.

Описание дополнительной сигмоидной функции:

$$y = \text{psigmf}(x, [a_1 \ c_1 \ a_2 \ c_2]).$$

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

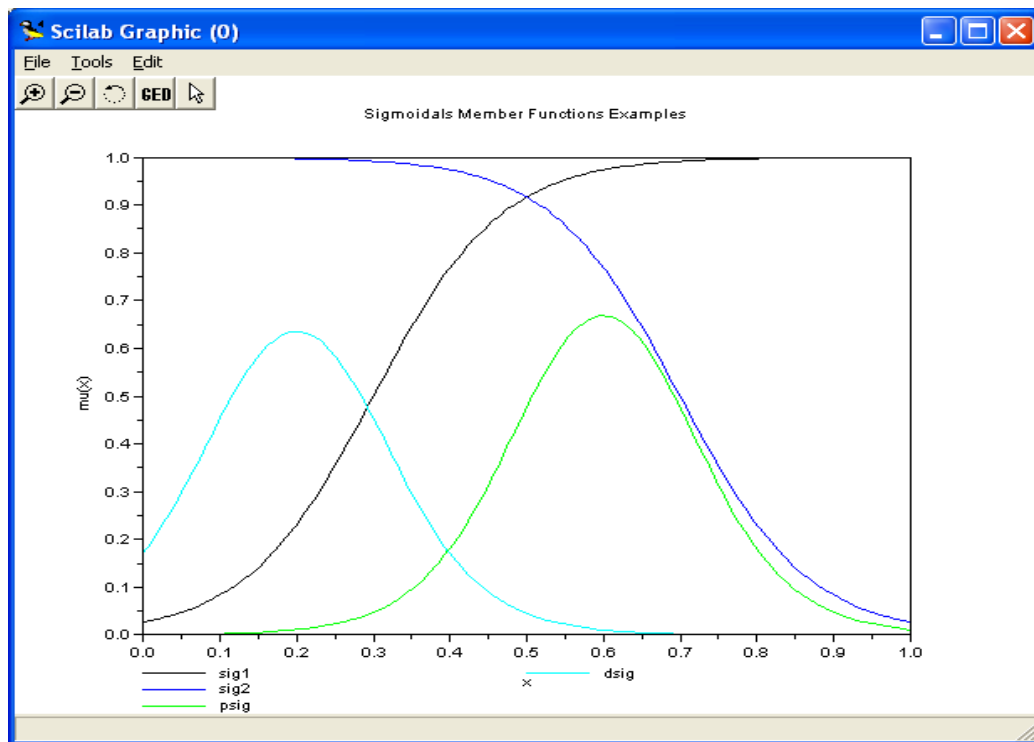


Рис. 1.5. Сигмоидные функции принадлежности

Пример 1.5. 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]);
xbasc();
plot2d(x,[sig1 sig2 psig dsig],leg="sig1@sig2@psig@dsig");
xtitle("Sigmoidals Member Functions Examples","x","mu(x)");
```

Инструментарий нечеткой логики *SciFLT* предоставляет возможность формирования ФП на основе полиномиальных кривых. Соответствующие функции называются *Z*-функции (*zmf*), *PI*-функции (*pimf*) и *S*-функции (*smf*). Функция *zmf* представляет собой асимметричную полиномиальную кривую, открытую слева, функция *smf* – зеркальное отображение функции *zmf*. Функция *pimf* равна нулю в правом и левом пределах и принимает значение, равное единице в середине некоторого отрезка (рис. 1.6).

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.

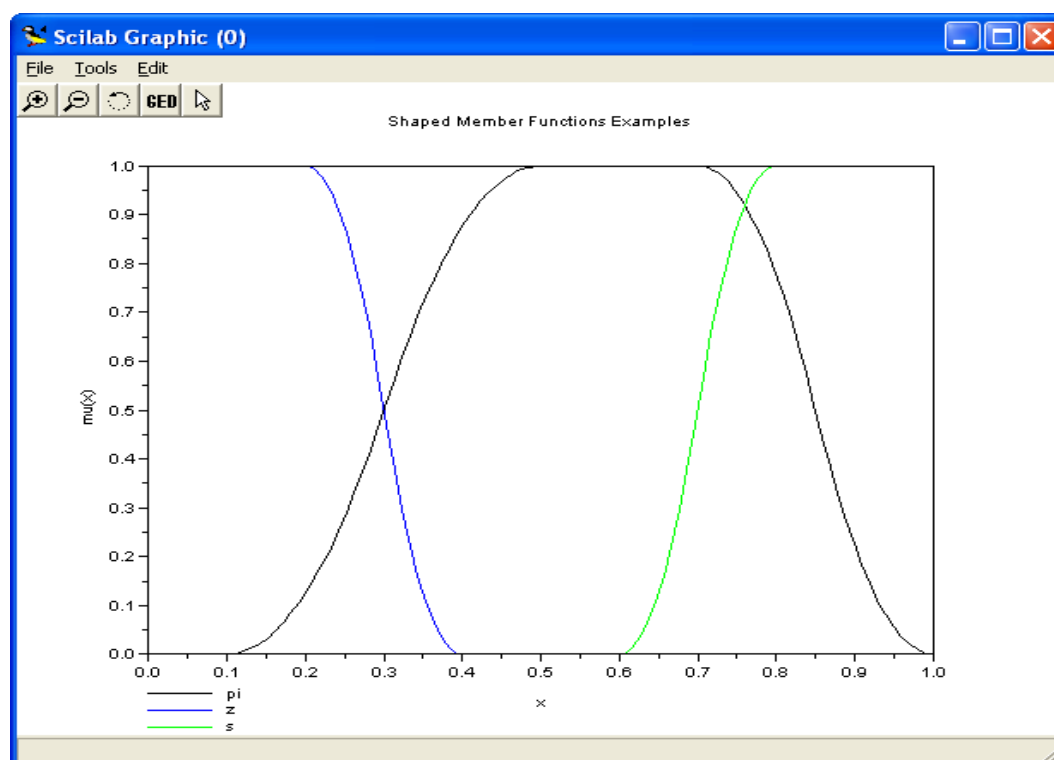


Рис. 1.6. Полиномиальные кривые

Пример 1.6. Программа использования полиномиальных 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]);  
xbasc();  
plot2d(x,[pi z s],leg="pi@z@s");  
xtitle("Shaped Member Functions Examples","x","mu(x)");
```

Помимо рассмотренных выше функций, позволяющих представлять нечеткие множества, в *SciFLT* можно формировать собственные ФП или модифицировать встроенные.

Операции с нечеткими множествами в модуле *SciFLT*

Выделяют три основные логические операции с нечеткими множествами: конъюнкцию, дизъюнкцию и логическое отрицание.

В модуле *SciFLT* существует возможность определять конъюнктивные и дизъюнктивные операторы различными методами.

Конъюнкция или Т-норма представляет собой нахождение логического И и представлено в программе следующими операторами:

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

Определение функции:

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

x – матрица, размерностью $[m, n]$.

y – матрица, размерностью $[m, 1]$.

class – строка, задает вид Т-нормы. Может принимать следующие значения: «**dubois**» для Т-норма Дюбуа-Прада, «**yager**» для Т-норм Ягера, «**dprod**» для вероятностного И, «**eprod**» для произведения Энштейна, «**aprod**» для алгебраического произведения и «**min**» для операции нахождения минимума.

class_par – скалярная величина, которая используется в Т-нормах «**dubois**» и «**yager**».

Дизъюнкция или S-конормы представляет собой логическое ИЛИ и может быть найдена посредством следующих операторов:

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

Определение функции:

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

x – матрица, размерностью [m,n].

y – матрица, размерностью [m,1].

class – строка, задает вид S-конормы. Может принимать следующие значения: «**dubois**» для S-конормы Дюбуа-Прада, «**yager**» для S-конормы Ягера, «**dsum**» для вероятностного ИЛИ, «**esum**» для суммы Энштейна, «**asum**» для алгебраической суммы и «**max**» для операции нахождения максимума.

class_par – скалярная величина, которая используется в S-конормах «**dubois**» и «**yager**»

Пример 1.7. Программа использования операций min и max

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

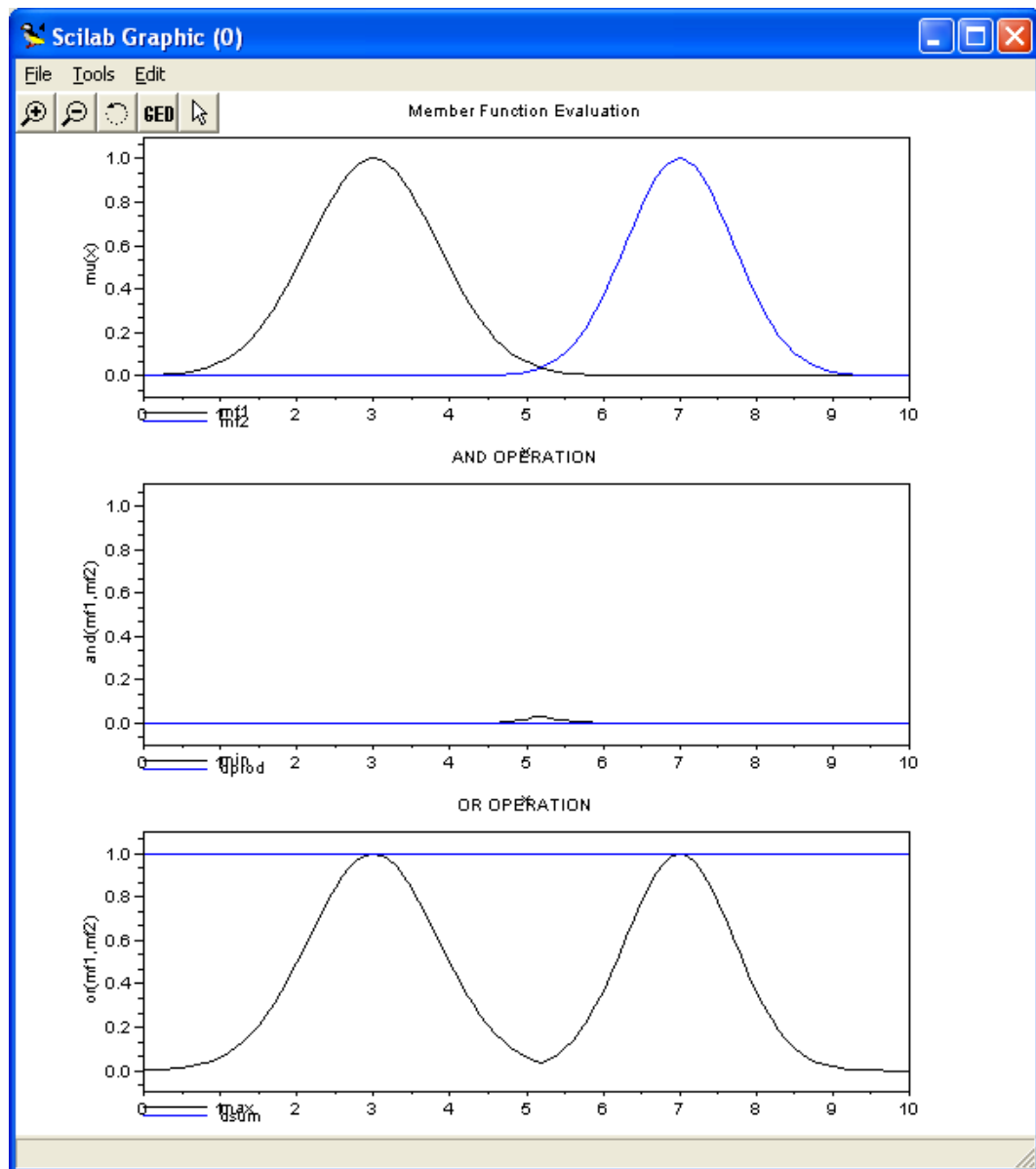



Рис. 1.7. Результаты работы программы из примера 1.7.

Минимаксная интерпретация является наиболее распространенной при построении нечетких систем. Тем не менее, на практике довольно часто используется альтернативная вероятностная интерпретация конъюнктивных и дизъюнктивных операторов.

Дополнение нечеткого множества есть не что иное, как математическое представление вербального выражения “НЕ A ”, где A – нечеткое множество, описывающее некоторое размытое суждение.

Описание функции дополнения:

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

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

class – строка, задает вид оператора дополнения. Может принимать следующие значения: «**one**» для обычного дополнения, «**sugeno**» для дополнения по формуле Сугено и «**yager**» для формулы Ягера.

class_par – скалярная величина, которая используется в S-конормах «sugeno» и «yager».

Математическая запись функций дополнения следующая:

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

Пример 1.8. Программа использования операции дополнения

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

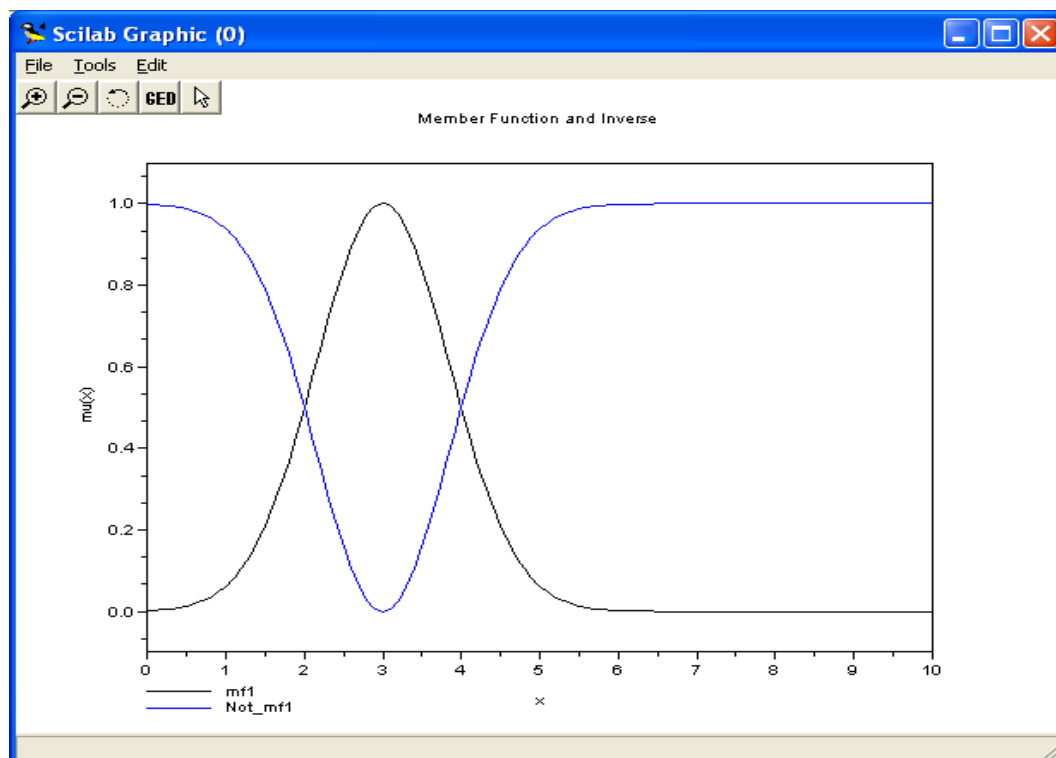


Рис.1.8. Функция дополнения

Контрольные вопросы к лабораторной работе № 1

1. Что такое нечеткое множество и каково его основное отличие от обычного (четкого) множества?
2. Что такое функция принадлежности?
3. Какие конъюнктивные и дизъюнктивные операторы Вы знаете?

Лабораторная работа № 9

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.

Для построения системы нечеткого вывода в командной строке основного окна SCILAB необходимо набрать команду *editfls*. Эта команда вызывает графический редактор систем нечеткого вывода.

Редактор СНВ. Построение нечетких систем

В меню необходимо выбрать **File->New FLS->тип создаваемой системы (Мамдани или Сугено)**.

Сначала создадим систему нечеткого вывода по Мамдани.

В левой панели появится ***Tree-control*** – дерево создаваемой нечеткой системы.

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

Для системы по Мамдани обычно выбирают следующие типы операторов (рис. 2.1):

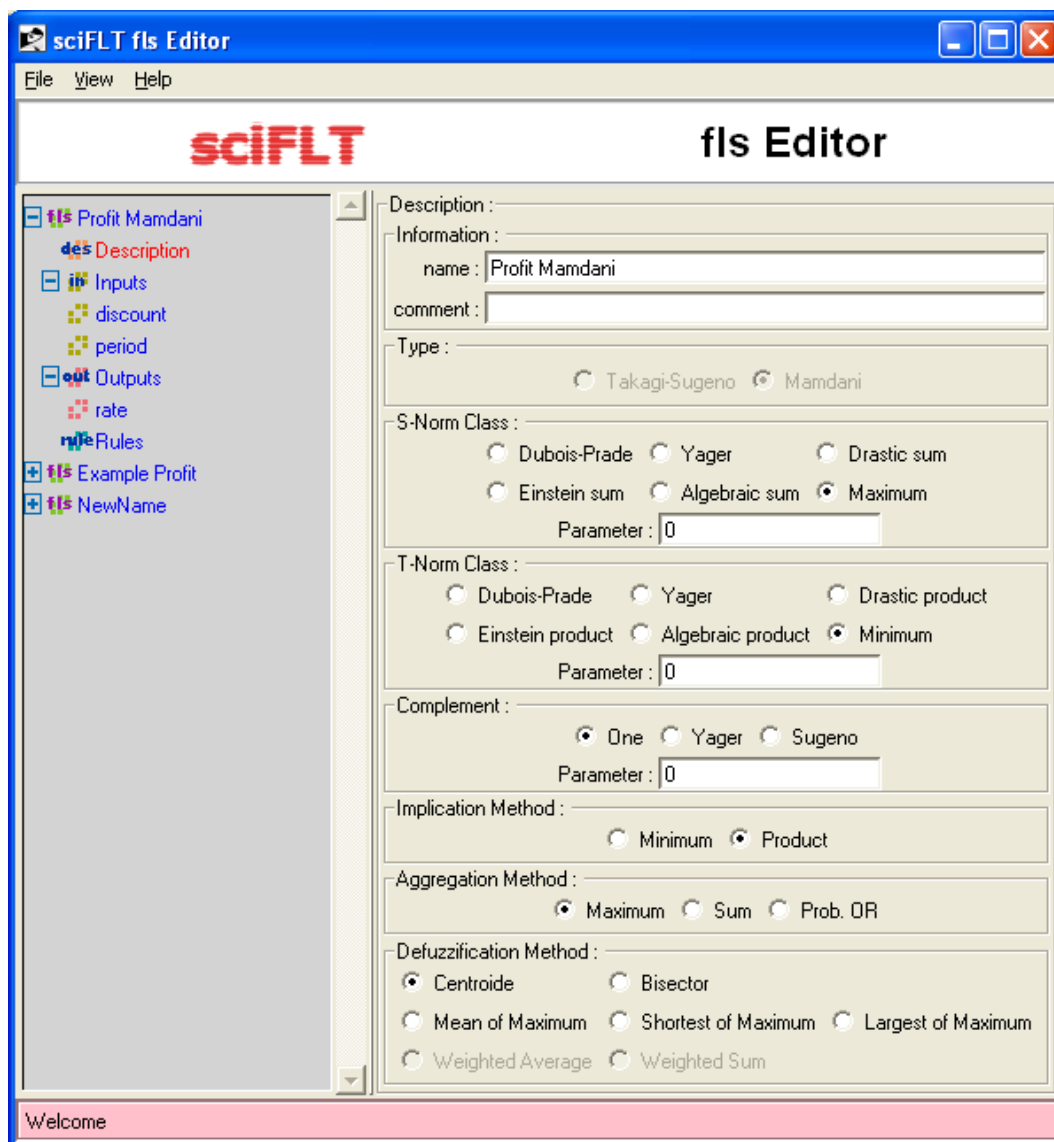


Рис. 2.1. Описание системы

Далее определяем состав входных и выходных переменных. Для этого необходимо выбрать в дереве 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* производится с помощью пункта меню **File** → **Export** → **To workspace**. В этом случае 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:

EDIT VARIABLE :

Information :

name : discount

range : 5 50

Nro. Member Function : 3

Member functions:

	name :	type :	par :
<input type="checkbox"/>	small	trimf	0 5 27.5
<input type="checkbox"/>	middle	trimf	5.0 27.5 50
<input type="checkbox"/>	big	trimf	27.5 50 55

Add Delete (checked)

Рис. 2.2. Редактор переменных (входная переменная из примера 2.1 Discount)

- 1) в нем необходимо задать имя;
- 2) диапазон допустимых значений;
- 3) с помощью кнопки Add внизу окна добавить функции принадлежности. Их можно также удалять, пометив щелчком мыши и нажав кнопку Delete (checked).

Редактор ФП является инструментом, который позволяет отображать и редактировать любые ФП, ассоциированные (связанные) со всеми входными и выходными переменными разрабатываемой СНВ.

Редактируют ФП текущей переменной изменяя характеристики ФП (наименование, тип и числовые параметры). Таким образом, при построении СНВ необходимо посредством редактора ФП определить соответствующие функции для каждой из входных и выходных переменных.

Редактор правил вывода

После того, как указано количество входных и выходных переменных, определены их наименования и построены соответствующие ФП, в СНВ необходимо включить правила вывода. Для этого в дереве контроля выбирается пункт ***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:

$$\begin{aligned} & \text{If}(\text{input_1 is [not]mf_1j}_1) < \text{and, or} > \dots (\text{input_i is [not]mf_ij}_i) \dots < \text{and, or} > \\ & (\text{input_n is [not]mf_nj}_n) \text{ then } (\text{output_1 is [not]mf_n+1j}_{n+1}) < \text{and, or} > \dots \\ & (\text{output_k is [not]mf_k+nj}_{k+n}) < \text{and, or} > \dots (\text{output_m is [not]mf_m+nj}_{m+n})(w), \end{aligned}$$

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

Пример 2.1. 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 первый. 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. 2.1. 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 *discont*, 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. 2.1 представлено текущее состояние окна редактора СНВ.

Шаг второй. Каждой входной и выходной переменной поставим в соответствие набор ФП. Для *discont* 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. 2.3).

EDIT VARIABLE :

Information :

name : period

range : 3 36

Nro. Member Function : 3

Member functions:

name :	type :	par :
<input type="checkbox"/> short	gaussmf	3.0 5
<input type="checkbox"/> medium	gaussmf	18 8
<input type="checkbox"/> long	gaussmf	36.0 -5

Add Delete (checked)

Рис.2.3. 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. 2.4).

EDIT VARIABLE :

Information :

name : rate

range : 0 1

Nro. Member Function : 3

Member functions:

name :	type :	par :
<input type="checkbox"/> bad	trimf	0.0 0.0 0.5
<input type="checkbox"/> normal	trimf	0.0 0.5 1.0
<input type="checkbox"/> good	trimf	0.5 1.0 1.0

Add Delete (checked)

Рис.2.4. Выходная переменная Rate

В графическом виде переменные представлены на рис. 2.5 и 2.6 (меню **View->Plot variables**)

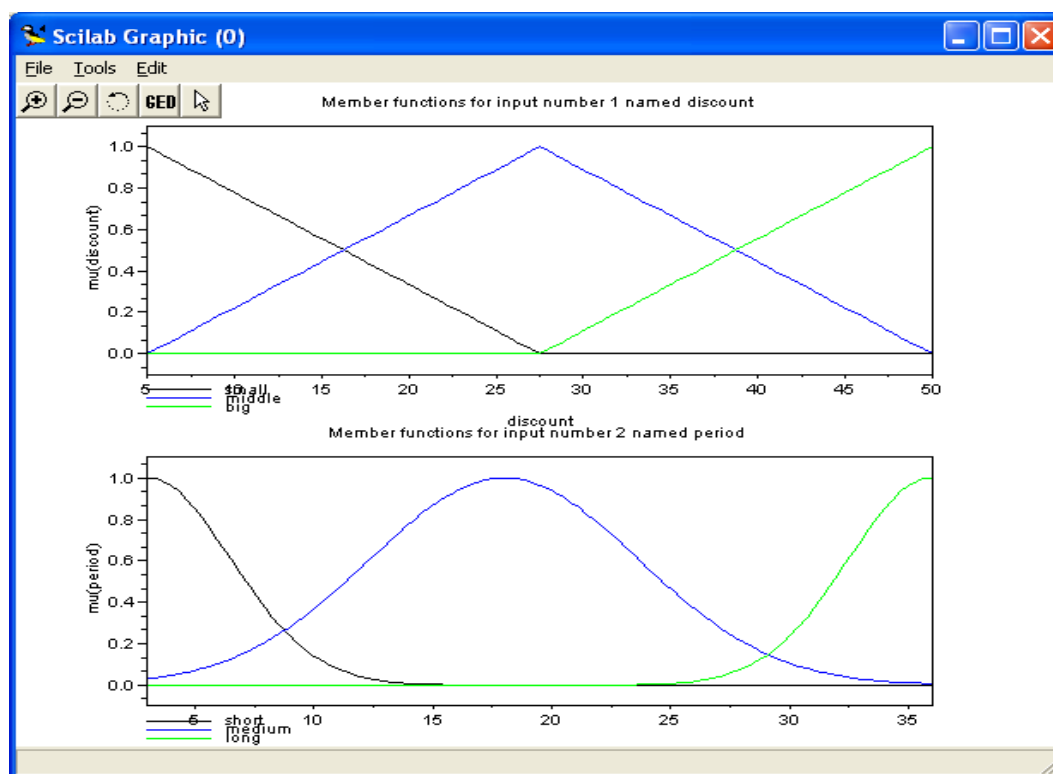


Рис. 2.5. Графическое представление входных переменных

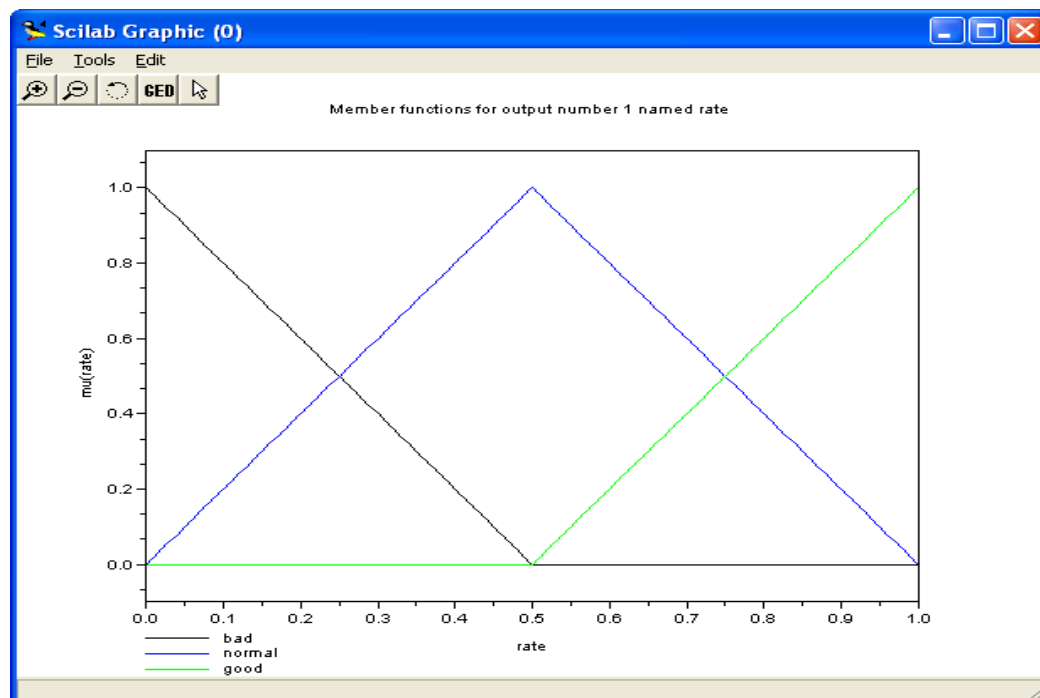


Рис. 2.6. Графическое представление выходной переменной

Шаг третий. 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*

Текущее состояние окна редактора правил вывода показано на рис. 2.7.

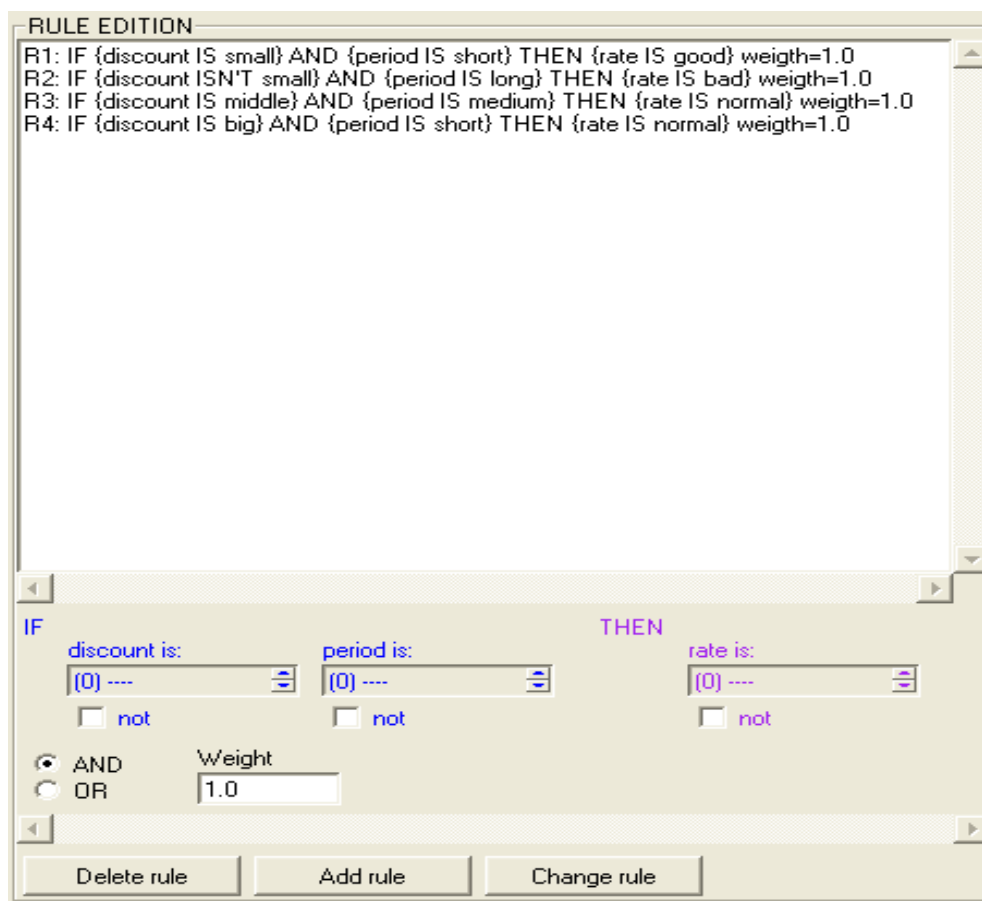


Рис.2.7. Редактор правил вывода

Средство просмотра поверхности вывода

Для этого существует функция **plotsurf**.

Она вызывается из командной строки окна SciLAB и имеет следующие параметры вызова:

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

Параметры 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, number of partitions domain for each input variable.
mod – скаляр, вид отображения поверхности на экране. 1 – grayscale, 2 – jetcolormap, 3 – hotcolormap, 4 – internal color map.

Пример 2.2.

```
fls=loadfls(flt_path()+"demos/MyEX.fls");
xbasc();
plotsurf(fls,[1 2],1,[0 0]);
```

Поверхность вывода, соответствующая правилам вывода примера 2.1 показана на рис. 2.8.

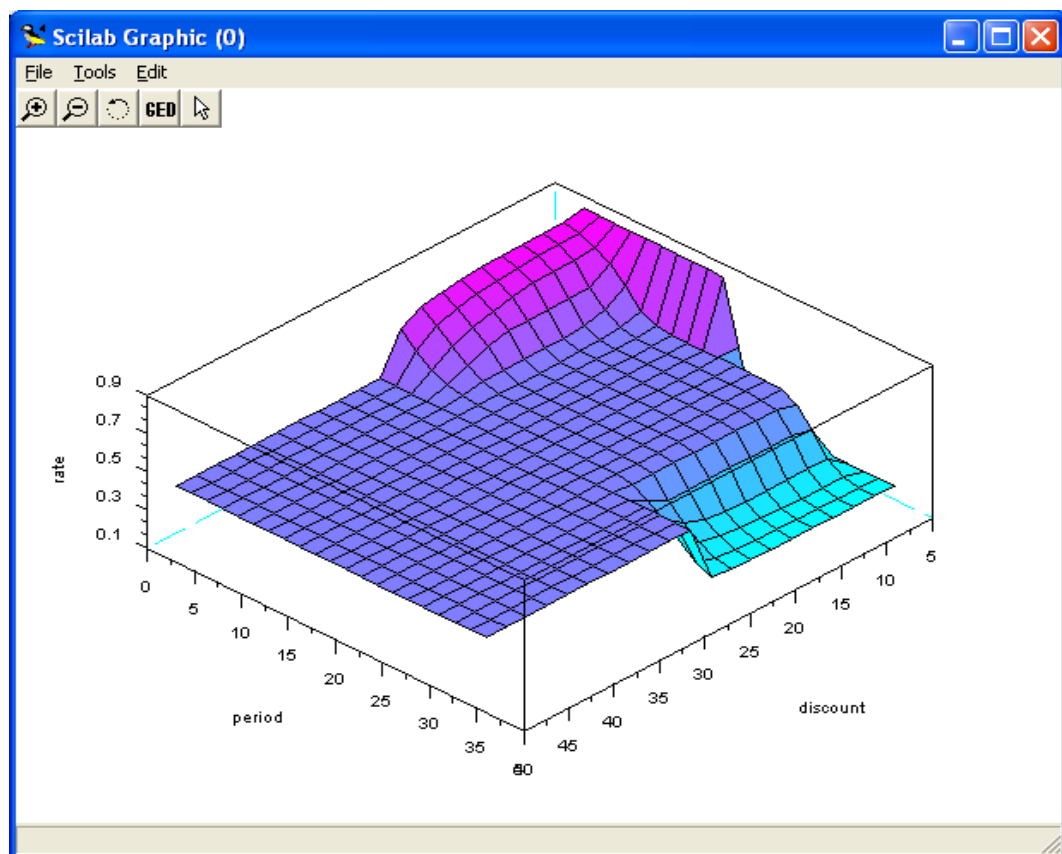


Рис. 2.8. Поверхность нечеткого вывода

Построение нечетких систем типа Суджено

Редактор СНВ. Для построения СНВ типа Суджено необходимо в меню **File** выбрать пункт **New Sugeno FIS**. Количество входных и выходных переменных определяется так же, как и при построении СНВ типа Мамдани.

Редактор ФП. Для СНВ типа Суджено изменения касаются только схемы определения ФП для выходных переменных. ИНЛ в среде *SCILAB* позволяет

разрабатывать два вида нечетких моделей. Первая модель – это нечеткая модель Суджено нулевого порядка. Нечеткое правило вывода имеет следующий вид:

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

где A и B – нечеткие множества антецедента, k – четко заданная константа консеквента. Для построения такой модели при добавлении ФП необходимо выбрать тип – константа (*constant*) и задать в качестве параметра ФП численное значение соответствующей константы. Вторая модель – нечеткая модель Суджено первого порядка. Для нее нечеткое правило вывода записывается следующим образом:

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

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 по лабораторной работе № 2

1. Какова структура типовой системы нечеткого вывода?
2. В чем отличие метода нечеткого вывода по Суджено от метода нечеткого вывода по Мамдани?
3. Каким образом формируются антецеденты и консеквенты нечетких правил в *SCILAB*?

Лабораторная работа № 10

Алгоритм нечеткой кластеризации

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

FCM алгоритм кластеризации

Алгоритм нечеткой кластеризации называют FCM-алгоритмом (*Fuzzy Classifier Means, Fuzzy C-Means*). Целью FCM-алгоритма кластеризации является автоматическая классификация множества объектов, которые задаются векторами признаков в пространстве признаков. Другими словами, такой алгоритм определяет кластеры и, соответственно, классифицирует объекты. Кластеры представляются нечеткими множествами и, кроме того, 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 итеративно вычисляет центры кластеров и новые степени принадлежности объектов.

Для заданного 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} :

$$\sum_{j=1}^N \mu_{jk} = 1, \forall k = 1, \dots, K,$$
$$0 < \sum_{k=1}^K \mu_{jk} \leq K, \forall j = 1, \dots, N.$$

Цель алгоритма заключается в минимизации суммы всех взвешенных расстояний $\|x_k - c_j\|$:

$$\sum_{j=1}^N \sum_{k=1}^K (\mu_{jk})^q \|x_k - c_j\| \rightarrow \min,$$

где q – фиксированный параметр, задаваемый перед итерациями.

Для достижения вышеуказанной цели необходимо решить следующую систему уравнений:

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

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

$$c_j = \frac{\sum_{k=1}^K (\mu_{jk})^q \cdot x_k}{\sum_{k=1}^K (\mu_{jk})^q}$$

(взвешенный центр гравитации) и

$$\mu_{jk} = \frac{1 / \|x_k - c_j\|^{1/(q-1)}}{\sum_{j=1}^N (1 / \|x_k - c_j\|^{1/(q-1)})}.$$

Алгоритм нечеткой кластеризации выполняется по шагам.

Шаг 1: Инициализация

Выбираются следующие параметры:

- необходимое количество кластеров N , $2 \leq N \leq K$;
- мера расстояний, как Евклидово расстояние;
- фиксированный параметр q (обычно ~ 1.5);

- начальная (на нулевой итерации) 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 итерационном шаге при известной матрице $\mu_{jk}^{(t)}$ вычисляется $c_j^{(t)}$ в соответствии с вышеприведенным решением системы дифференциальных уравнений.

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:

$$\mu_{lk}^{(t+1)} = \begin{cases} 1, & l = j, \\ 0, & \text{иначе} \end{cases}$$

Шаг 4: Остановка алгоритма

Алгоритм нечеткой кластеризации останавливается при выполнении следующего условия:

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

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

Решение задач кластеризации в SCILAB

В SCILAB отсутствуют средства для решения задач кластеризации.

Поэтому в рамках лабораторной работы необходимо разработать приложение, реализующее алгоритм кластеризации, описанный выше.

Для этого необходимо реализовать функцию нахождения центров кластеров.

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

Аргументами данной функции являются:

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

2) *cluster_n* – количество кластеров (более одного).

Функцией возвращаются следующие параметры:

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

2) *U* – результирующая матрица ФП;

3) *obj_fcn* – значение целевой функции на каждой итерации.

Кроме того, необходимо обеспечить загрузку данных, подлежащих кластеризации, из файла или обеспечить их ручной ввод с клавиатуры.

И предусмотреть возможность вызова функции кластеризации с дополнительным набором параметров: $fcm(data, cluster_n, options)$. Дополнительные аргументы используются для управления процессом кластеризации:

1) *options*(1) – показатель степени для матрицы *U* (по умолчанию нужно установить - 2.0);

2) *options*(2) – максимальное количество итераций (по умолчанию - 100);

3) *options*(3) – предельное изменение значений целевой функции (по умолчанию сделать 1e-5);

4) *options*(4) – отображение информации на каждом шаге (по умолчанию - 1).

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	18.869047	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

Лабораторная работа № 11

Обучение классификации нейронной сети

Здесь приводится описание примера использования модуля ATOMS [Neural Network Module](#).

Этот модуль нейронной сети разработан на основе книги Мартина Т. Хагана "[Neural Network Design](#)". Разработчиком этого модуля нейронной сети является Тан Чин Лух.

Мы будем использовать данные, которые можно загрузить по ссылке: [csv](#)

Импорт данных

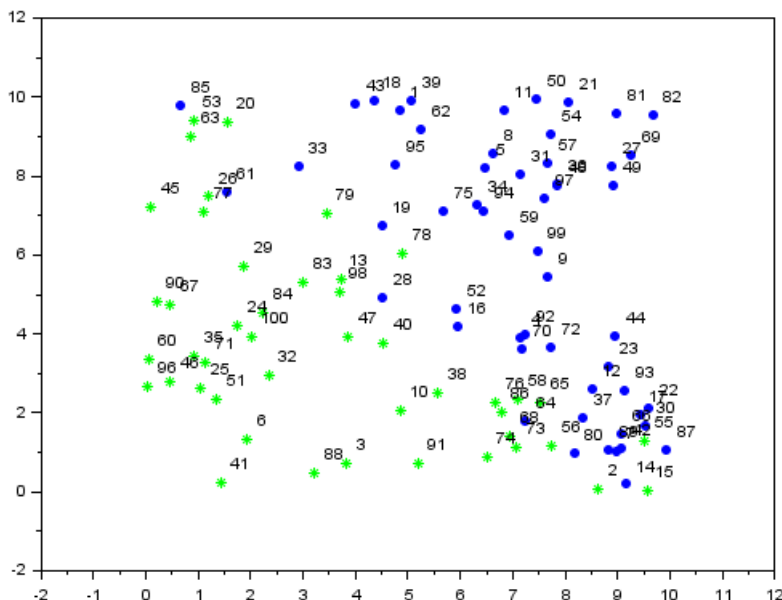
Этот набор данных представляет собой 100 образцов, которые подразделяются на два вида, как 0 или 1 (хранится в третьей колонке), по двум параметрам (хранятся в первом и втором столбцах файла `data_classification.csv`). Непосредственно данные импортируются в Scilab с помощью следующей команды:

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

Представление данных

Функция реализована в нейро-сетевой модуль для упрощения построения 2-х групп точек данных. Во-первых, это разделяет источник данных (P) и цель (T), а во-вторых, данные транспонируются в соответствии с требуемым форматом модуля.

```
P = t(:,1:2)';  
T = t(:,3)';  
plot\_2group(P,T);
```



ANN алгоритм обратного распространения с градиентной функцией обучения

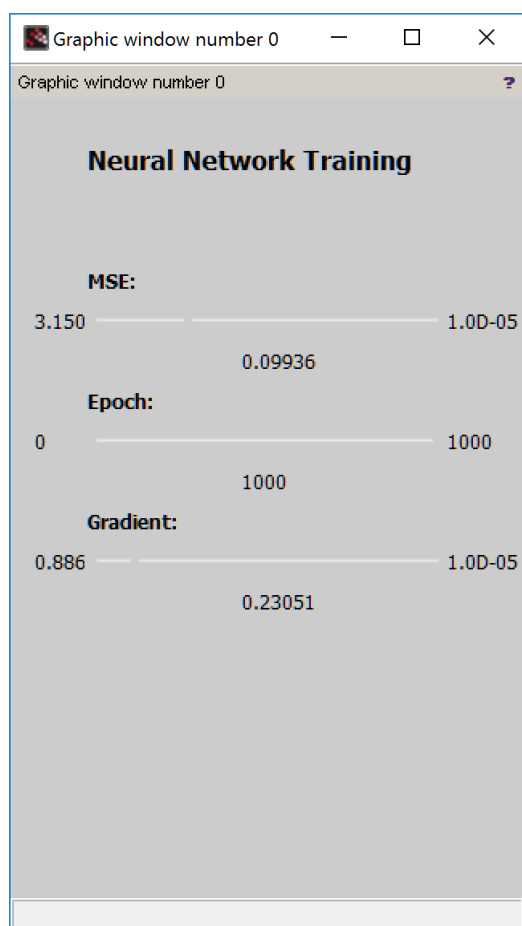
Полный синтаксис алгоритма обратного распространения с градиентной функцией такой:

$W = \text{ann_FFBP_gd}(P, T, N, af, lr, \text{itermax}, \text{mse_min}, \text{gd_min})$

где **P** - входное обучение (источник данных), **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 мы используем только первые 3 обязательных поля, а другие поля мы оставили пустыми, чтобы использовались значения по умолчанию.



```
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.
```

ANN алгоритм обратного распространения с градиентной функцией скорости и момента обучения

Для того чтобы улучшить сходимость GD обучения в книге "Neural Network Design" предлагаются несколько методов. В этом примере мы будем использовать сочетание функции адаптивной скорости обучения и функции 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.

```

ANN алгоритм обратного распространения с функцией обучения Левенберга-Маркуарда

Еще более быстрым алгоритмом сходимости является алгоритм Левенберга-Маркуарда.

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))

```

Результат показывает, что он сходится после нескольких десятков итераций.

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.

```

Визуализация результатов

Поскольку границу нейронной сети сложно описать математически, особенно при большом числе нейронов скрытого слоя (увеличивается число слоев), визуализация этой границы затруднена.

Мы будем использовать 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
```

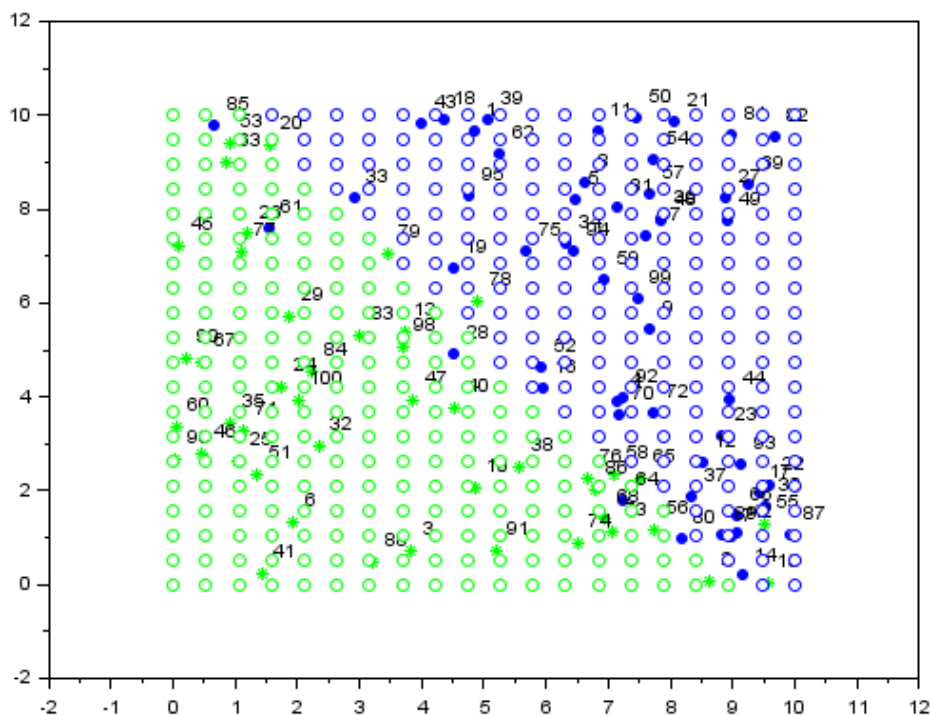
```
G0 = P2(:,find(round(y2)==0));
```

```
G1 = P2(:,find(round(y2)==1));
```

```
// Построение границ областей для групп
```

```
plot(G0(1,:),G0(2:),'go');
```

```
plot(G1(1,:),G1(2:),'bo');
```



Функции для модуля нейронных сетей поставляются с кратким описанием, к которому можно получить доступ командой помощи Scilab. Подстановка разных параметров приводит к разным результатам. Сложные наборы данных требуют увеличения количества нейронов / слоев.

Лабораторная работа № 12

Исследование генетического алгоритма

Последовательность вызова

```
[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

$y = \text{list}(f, p1, p2, \dots)$

Создаёт список **list** с элементами, которые являются произвольными объектами Scilab'a (**matrix**, **list**,...). Тип объекта **list** равен 15.

Пример

```
l = list(1, ["a" "b"])           // объявление основного списка с элементом типа  
                                // double и вектором двух строковых значений  
size(l)                         // размер равен 2  
                                // l(0) - не существует!  
l(1)                            // доступ к значению типа double  
l(2)                            // доступ к вектору строковых значений  
size(l(2))                     // размер равен 1,2  
  
l(0) = "foo"                    // вставка в начало списка  
                                // l(0) - по-прежнему не существует  
l(1)                            // равно "foo"  
  
l($+1) = "hello"                // вставка в конец списка  
l(2) = "toto"                   // перезапись моего значения типа double  
l(3) = rand(1,2)                 // перезапись моего вектора строковых значений  
  
l(3) = null()                   // удаление третьего элемента  
  
lbis = list("gewurtz", "caipirina", "debug") // объявление нового списка  
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 (значение по умолчанию: 10).

p_mut вероятность мутации (значение по умолчанию: 0,1).

p_cross вероятность кроссовера (значение 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, которая будет выполнять кодирование и декодирование индивидуумов (по умолчанию: [coding_ga_identity](#)).
- «**init_func**»: функция, которая будет выполнять инициализацию популяции (по умолчанию: [init_ga_default](#)).
- '**dimension**', '**minbounds**' и '**maxbounds**': параметры, используемые функцией инициализации для определения начальной совокупности.
- «**crossover_func**»: функция, которая будет выполнять кроссовер между двумя индивидами (по умолчанию: [crossover_ga_default](#)).

- «**mutation_func**»: функция, которая будет выполнять мутацию одного индивида (по умолчанию: [mutation_ga_default](#)).
- «**selection_func**»: функция, которая будет выполнять выбор индивидов в конце поколения (по умолчанию: [selection_ga_elitist](#)).
- '**nb_couples**': количество пар, которые будут выбраны для выполнения кроссовера и мутации (значение по умолчанию: 100).
- '**pressure**': значение эффективности наихудшего индивида (значение по умолчанию: 0,05).
- «**output_func**»: функция обратного вызова, вызываемая после каждой генерации, если **Log %T** (по умолчанию **output_moga_default**).

pop_opt	популяция оптимальных индивидов
fobj_pop_opt	набор значений многоцелевой функции, связанных с pop_opt (необязательно).
pop_init	начальная популяция индивидов (необязательно).
fobj_pop_init	множество мульти-значений целевой функции, связанные с pop_init (по желанию)

Пример

```
function f=deb_1(x)                                     // оптимизируемая функция
    fl_x1 = x(1);
    g_x2 = 1 + 9 * sum((x(2:$)-x(1)).^2) / (length(x) - 1);
    h      = 1 - sqrt(fl_x1 / g_x2);

    f(1,1) = fl_x1;                                     // 1-й элемент индивида
    f(1,2) = g_x2 * h;                                 // 2-й элемент индивида
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 индивида

ga_params = init_param();                               // инициализация начальной популяции
ga_params = add_param(ga_params, 'dimension', 2);       // число элементов индивида
ga_params = add_param(ga_params, 'minbound', zeros(2,1)); // минимум значений
ga_params = add_param(ga_params, 'maxbound', ones(2,1)); // максимум значений

[pop_opt, fobj_pop_opt, pop_init, fobj_pop_init] =..

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

Задание

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