Государственное образовательное учреждение
высшего профессионального образования
Московский технический университет связи и информатики
Кафедра информатики

Семенова Т.И.

Математический пакет Scilab

Лабораторный практикум

Для студентов по направлению подготовки

11.03.02– «Инфокоммуникационные технологии и системы связи»

Москва, 2017 г.

Оглавление

Введение……4

Раздел 1. Основы работы с математическим пакетом Scilab……5

Тема 1.1. Элементы рабочей среды Scilab……..……5

1.1.1. Элементы рабочей среды Scilab и простейшие вычисления……5

1.1.2. Основные объекты системы Scilab……. 15

1.1.3. Лабораторная работа ……25

1.1.4. Контрольные вопросы …..……..27

Тема 1.2. Вектора, матрицы и построение графиков в системе Scilab……..…… 28

1.2.1. Вектора и матрицы…… 28

1.2.2. Построение графиков и визуализация вычислений в системе Scilab……..……37

1.2.3. Лабораторная работа ……..……..……..48

1.2.4. Контрольные вопросы ……....51

Тема 1.3. Средства алгоритмизации и программирования в Scilab……..……52

1.3.1. Основные понятия и средства программированиям в Scilab……..……52

1.3.2. Описание и работа с sci-сценариями……..……55

1.3.3. Описание и работа с sci-функциями……..……..58

1.3.4. Основные операторы sci-языка и программирование в Scilab…..……62

1.3.5. Примеры решения задач средствами Scilab……..74

1.3.6. Лабораторная работа ……79

1.3.7. Контрольные вопросы ……83

Раздел 2. Технология решения вычислительных задач средствами Scilab……84

Тема 2.1. Решение нелинейных уравнений……...84

2.1.1. Численное решение нелинейных уравнений……84

2.1.2. Лабораторная работа …….……...90

2.1.3. Контрольные вопросы ……...91

Тема 2.2. Технология аппроксимации и интерполяции функций в среде пакета Scilab……...92

2.2.1.Аппроксимация и интерполяция функций……92

2.2.2. Лабораторная работа ……..….99

2.2.3. Контрольные вопросы ……..……101
Тема 2.3. Технология интегрирования в среде Scilab……102
2.3.1. Вычисление неопределенных и определенных интегралов……..……102
2.3.2. Лабораторная работа ……..…….105
2.3.3. Контрольные вопросы…….……107

Тема 2.4. Технология решения обыкновенных дифференциальных уравнений……..……..108
2.4.1. Численное решение обыкновенных дифференциальных уравнений…….……..108
2.4.2. Лабораторная работа ……....113
2.4.3. Контрольные вопросы …….. 115

Тема 2.5. Технология решения задач одномерной оптимизации……..116
2.5.1. Решение задач одномерной оптимизации функций……116
2.5.2. Лабораторная работа …….……120
2.5.3. Контрольные вопросы ……121

Тема 2.6. Технология решения задач многомерной оптимизации ……..……122
2.6.1. Решение задач многомерной оптимизации функций…..122
2.6.2. Лабораторная работа ……..……..126
2.6.3. Контрольные вопросы …….……...128

Введение

Возрастающий объем задач требует автоматизации их решения. В этой связи становится весьма актуальным применение для их решения функциональных программных средств. К их числу относятся математические пакеты Matlab, MathCad, Mathematica, Scilab и др. Все они содержат необходимый набор методов решения математических задач, а также средства для визуализации и отображения полученных результатов. Наиболее известным и популярным среди вышеперечисленных программных средств является математический пакет Matlab. Он позволяет производить технические вычисления различной сложности, содержит одноименный язык программирования, предоставляет большое количество функций анализа данных, связанных практически со всеми областями математики и большинством операционных систем. Однако данный пакет является коммерческим. Разумеется, этот факт затрудняет широкое использование пакета Matlab. Но существуют свободно распространяемые альтернативы данного пакета, например, программная система Scilab. Версии Scilab существуют для различных операционных систем: для ОС Linux, ОС семейства Windows (в том числе и для MS WindowsVista) и даже для MacOS. Последнюю версию пакета (Scilab 6.0) можно скачать на официальном сайте программы www.scilab.org.

Scilab – это система компьютерной математики, являющаяся самым полным аналогом пакета Matlab, предназначенная для выполнения научных и инженерных вычислений. Scilab во многом уступает пакету Matlab, но возможности этого пакета вполне достаточны для его использования в учебном процессе при изучении методов вычислительной математики. Так, например, в системе Scilab реализованы следующие методы решения вычислительных задач: задачи линейной алгебры; методы решения нелинейных уравнений и систем уравнений; обработка экспериментальных данных; задачи оптимизации; интегрирование и дифференцирование; решение обыкновенные дифференциальных уравнений и их систем. Кроме того Scilab позволяет работать с большим числом специальных функций (Бесселя, Неймана и т.д.), имеет средства для построения и работы с графиками, а для выполнения численных расчётов могут использоваться библиотеки Lapack, LINPACK, Atlas и другие.

Для решения нестандартных задач в Scilab имеется встроенный объектно-ориентированный язык программирования, sci-язык, с помощью которого пользователь может создавать свое визуальное приложение в виде отдельной программы. Кроме того, в состав Scilab входит утилита, осуществляющая конвертирование документов из Matlab в Scilab, что немаловажно при разработке программ в системе Scilab, использующих готовые модули пакета Matlab.

Рассмотрим некоторые возможности пакета Scilab, используемые при решении задач численными методами.

Раздел 1. Основы работы с математическим пакетом Scilab

Тема 1.1. Элементы рабочей среды Scilab

1.1.1. Элементы рабочей среды Scilab
1.1.2. Основные объекты системы Scilab
1.1.3. Лабораторная работа по теме
1.1.4. Контрольные вопросы по теме

1.1.1. Элементы рабочей среды Scilab

Графический интерфейс пользователя **Scilab** версий **6.0** во многом напоминает интерфейс **Matlab**. После запуска **Scilab** на экране дисплея появляется **Рабочая среда** (интерфейс пользователя) системы **Scilab** в стандартной конфигурации (рис. 1.1.1-1). При этом система **Scilab** готова к проведению вычислений в командном окне.

Рабочая среда системы **Scilab** – это обычное окно приложений MS Windows, поэтому его можно перемещать, изменять в размерах, открывать на весь экран. В окне стандартной конфигурации могут быть размещены следующие компоненты (на рис. 1.1.1-1 отображены соответствующими выносками, пронумерованными от 1 до 6):

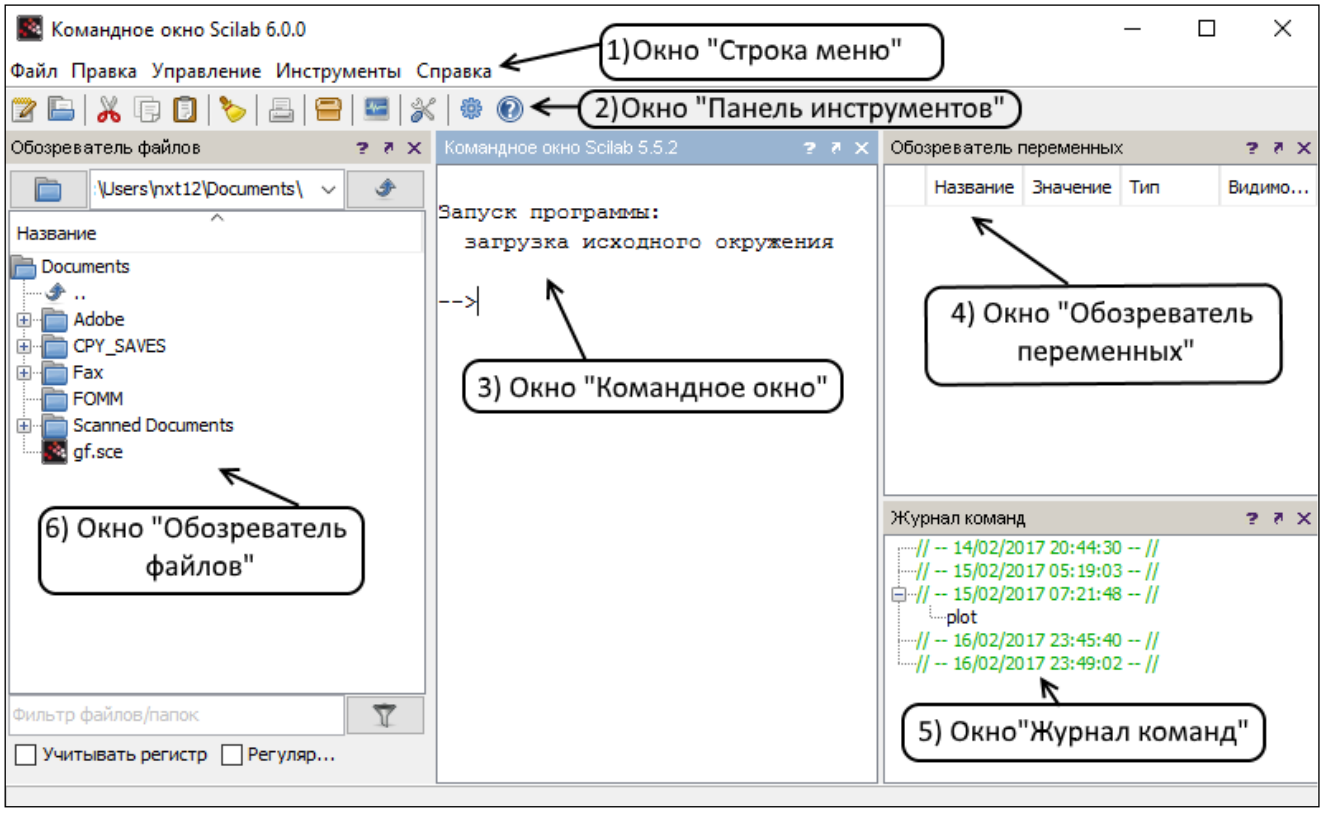

Рис. 1.1.1-1. Стандартная конфигурация **Рабочей среды Scilab**

1) **Строка меню**, является элементом управления, обеспечивающим доступ ко всем функциональным возможностям программы: **Файл, Правка, Управление, Инструменты, Справка**. Более подробно все категории строки меню будут рассмотрены ниже.

2) **Панель инструментов** позволяет выполнить команды: **Открыть SciNotes, Открыть файл, Вырезать, Скопировать, Вставить, Очистить командное окно, Печать, Управление модулями, Xcos, Настройки Scilab, Примеры и Справочная.** Все команды отображаются на панели в форме кнопок с соответствующим изображением, подсказывающим их назначение.

3) **Командное Окно** отображает вводимые команды, результаты их выполнения, а также сообщения об ошибках.

4) **Окно Обозреватель переменных** отображает содержимое рабочего пространства объектов **Scilab**, и, позволяет выполнять определенные действия с объектами этого пространства (скалярными переменными, векторами, матрицами, функциями и др.).

5) **Окно Журнал команд** осуществляет просмотр и повторный вызов ранее введенных команд.

6) **Окно Обозреватель файлов** предназначен для просмотра путей доступа файловой системы. В нем перед работой в Scilab с конкретным файлом (чтение или запись файла на внешний носитель), расположенным в определенной папке, необходимо указать путь доступа к файлу. В низу окна отображается фильтр файлов/папок

В стандартной конфигурации четыре основных окна, вписанные в **Рабочую среду** Scilab, они могут передвигаться вместе с основным окном и вместе с ним изменять свои размеры.

Рассмотрим основные компоненты **Рабочей среды** в стандартной конфигурации более подробно.

Строка меню расположенная в верхней части **Рабочей среды** Scilab**,** и состоит из пяти категорий: **Файл, Правка, Управление, Инструменты, Справка.** Каждому из этих команд соответствует своё меню команд.

При активизации категории **Файл** в ниспадающем меню отображаются инструментов, необходимых для текущей работы с файлами:

- **Выполнить;**
- **Открыть файл;**
- **Загрузить окружение …,**
- **Сохранить окружение …,**

- **Сменить текущий каталог;**
- **Отобразить текущий каталог;**
- **Параметры страницы** - инструмент, для изменения типа и формата страницы;
- **Печать …;**
- **Выход.**

Категория **Правка** содержит меню инструментов, которые позволяют осуществлять различные действия над выделенной частью окна или всем его содержимым:

- **Вырезать –** инструмент, для удаления выделенной области;
- **Копировать –** инструмент, для создания копии выделенной области;
- **Вставить –** инструмент, для вставки скопированной области;
- **Очистить буфер обмена –** инструмент, для очистки области памяти;
- **Выделить все;**
- **Показать/ скрыть панель инструментов;**
- **Очистить журнал команд;**
- **Очистить командное окно;**
- **Настройки –** команда для настройки элементов пакета Scilab.

Рассмотрим, например, как произвести с использованием команд окна **Настройки** выбор шрифта и цвета.

Для того чтобы изменить шрифт, нужно в окне **Настройка** (рис. 1.1.1-2) в меню **Общее** выбрать **Шрифт** - откроется вкладка (рис. 1.1.1-3). Шрифт можно настроить для ввода информации в командную строку (**Console)** или в строку текстового редактора **Scinotes**.

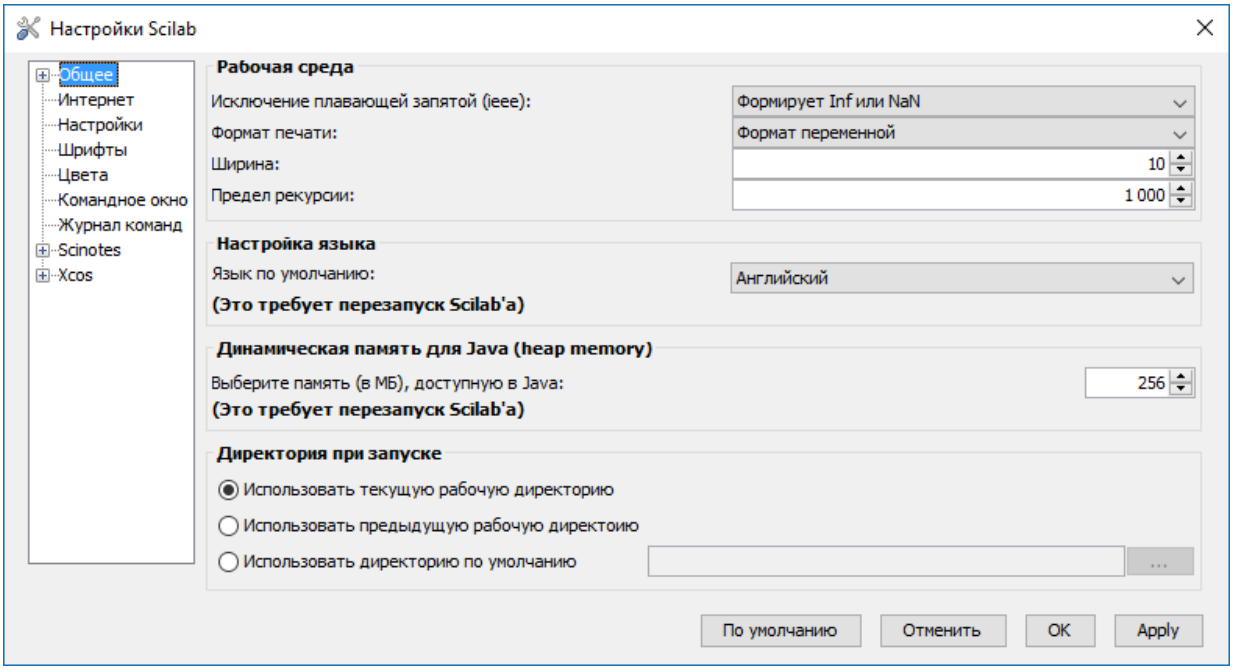

Рис. 1.1.1-2. Окно Настройки

Например, при изменении шрифта в **Командном окне** следует в списке **Свой шрифт** выбрать **Console**, в правой части окна нажать на клавишу с многоточием и в открывшемся окне произвести выбор нужного шрифта (рис. 1.1.1-3). Далее можно выбрать тип, размер и стиль шрифта. Завершается настройка нажимаем клавиши **ОК**, а в меню **Настройки** нажатием кнопок **Применить** (или **Apple**) и **ОК**.

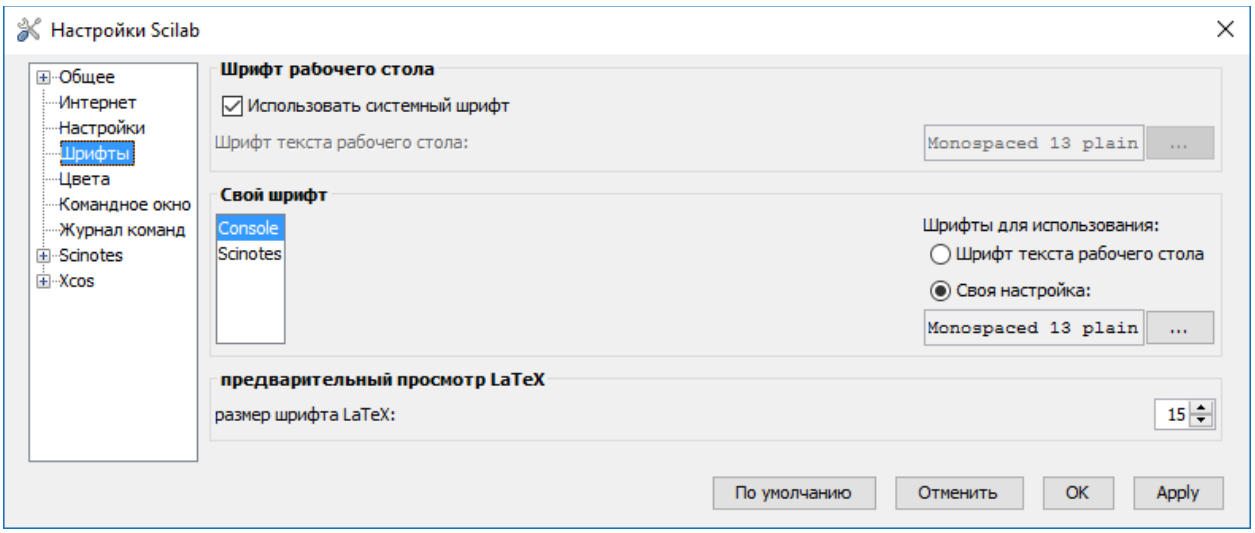

Рис. 1.1.1-3. Настройка шрифта **Console**

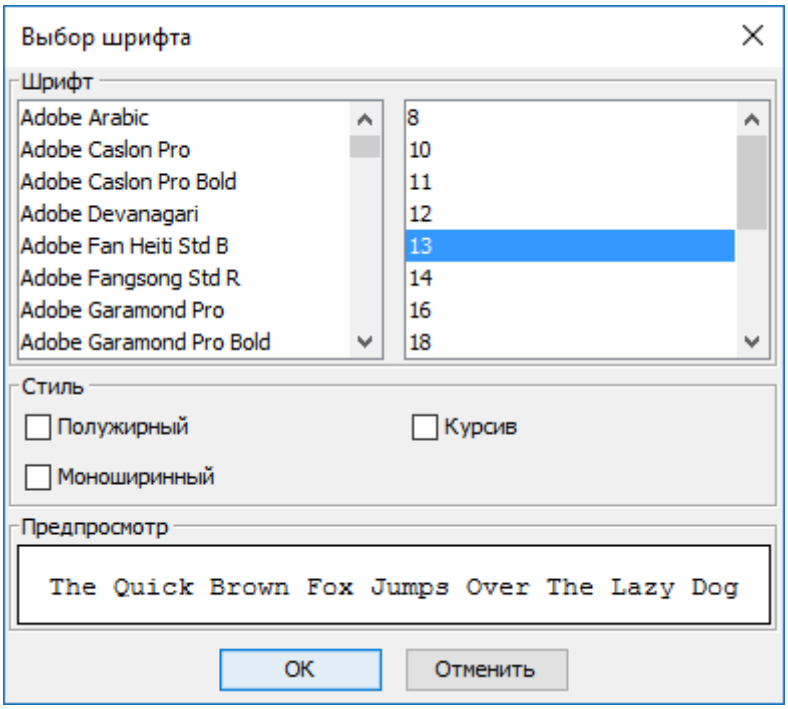

Рис. 1.1.1-4. Выбор выбрать тип, размер и стиль шрифта

Для того чтобы изменить цвет, в списке **Свой шрифт** следует выбрать **Цвета** - откроется вкладка (рис. 1.1.1-5).

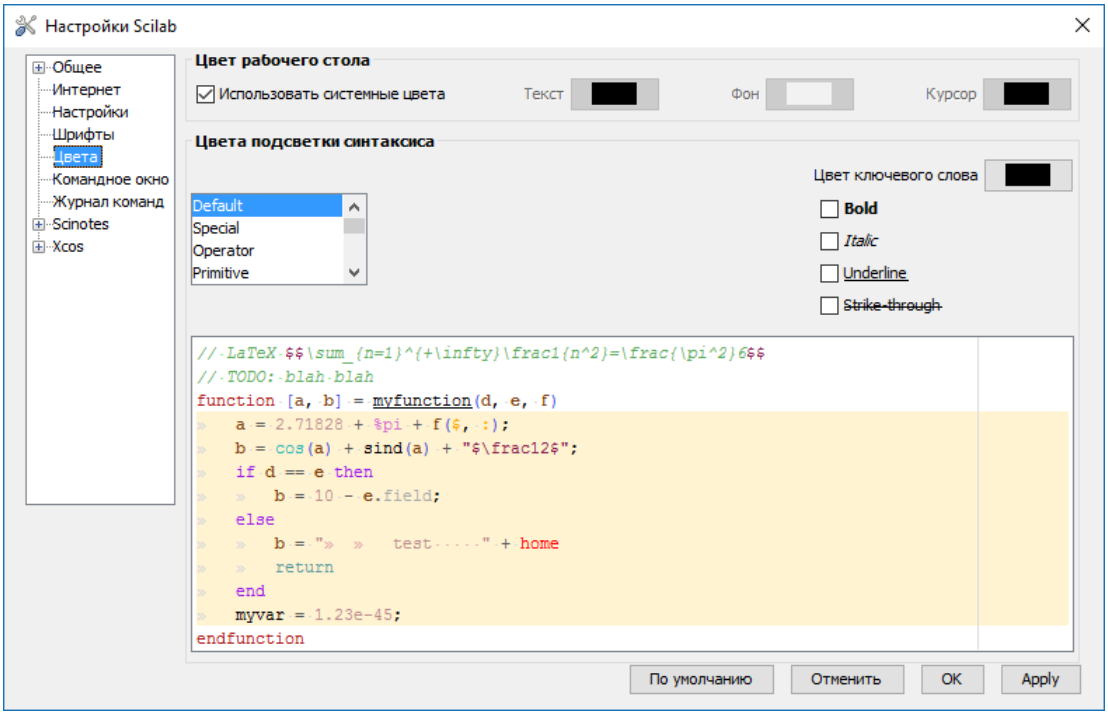

Рис. 1.1.1-5. Вкладка Цвета

Цвет можно настроить для команд, текста, фона или курсора. Например, при изменении цвет фона, нужно в поле команды **Использовать системные цвета** снять галочку, после чего нажать на клавишу **Фон**. В открывшейся цветовой палитре (рис. 1.1.1-6) следует выбрать цвет фона и нажать **ОК**. Затем в меню **Настройки** выбрать **Применить** (или **Apple**) и **ОК**.

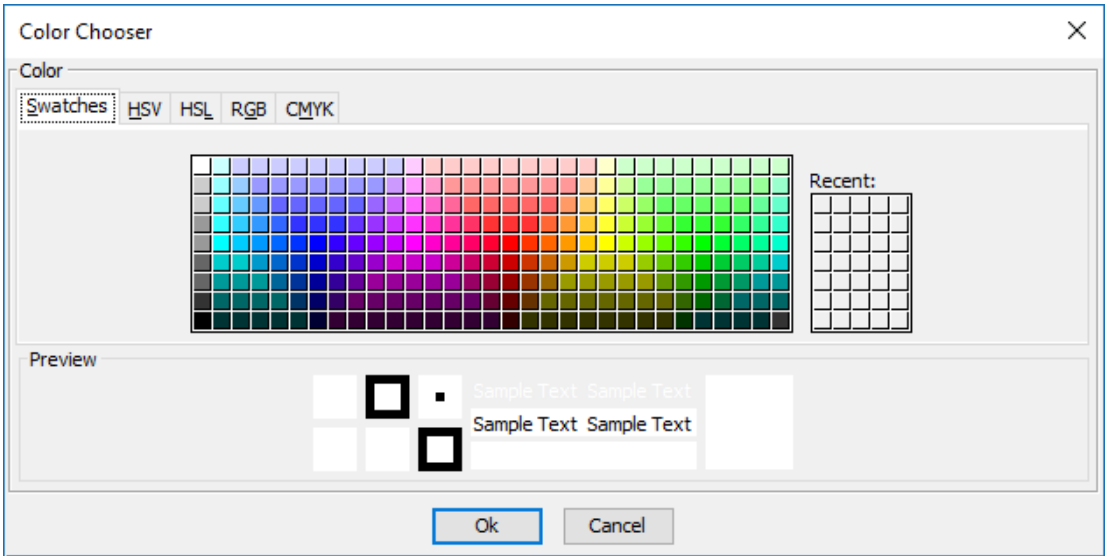

Рис. 1.1.1-6. Настройка цвета фона

Категория **Управление** содержит инструменты для управления программой в Scilab. Они разбиты на следующие категории:

- **Возобновить;**
- **Возобновить;**

- **Завершить;**
- **Приостановить.**

При активизации категории **Инструменты,** отображается меню команд, в котором присутствуют инструменты, необходимые для текущей работы в Scilab. Эти инструменты разбиты на следующие категории:

- **Текстовый редактор SciNotes –** открывает редактор SciNotes;
- **Визуальное моделирование Xcos;**
- **Преобразование из Matlab в Scilab;**
- **Управление модулями Atoms;**
- **Обозреватель переменных;**
- **Журнал команд;**
- **Обозреватель файлов.**

Окно редактора **SciNotes** можно просто открыть, нажав на первую кнопку в панели инструментов  или вводом в командном окне команды:

```
-->SciNotes
```

При этом открывается новое окно редактора с именем «Безымянный1» (рис. 1.1.1-7). По завершении работы с окном редактора его можно сохранить в файле с расширением .sce (или .sci). Сохранение файла выполняется использованием команды **Сохранить как** или соответствующей иконки в панели инструментов  . Для сохранения изменений в файле, созданного с использованием редактора используется команда **Сохранить**.

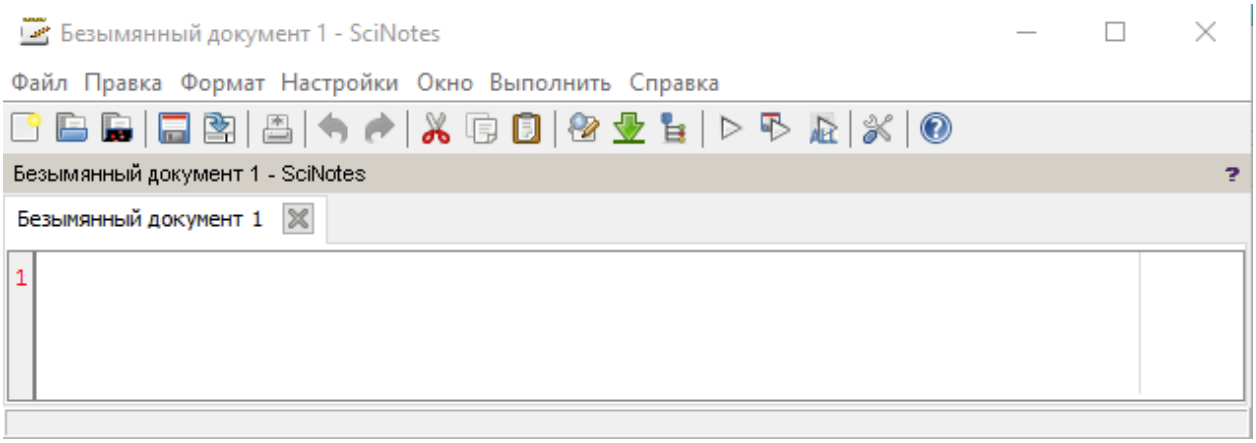

Рис. 1.1.1-7. Открытие окна Редактора **SciNotes**

Категория **Справка,** служит для ознакомления пользователя с возможностями пакета Scilab и получения текущей справки. Она содержит инструменты, разбитые на следующие инструменты:

- **Содержание –** открывается окно с содержанием справочной системы Scilab;

- **Примеры** – открываются наглядные примеры использования возможностей Scilab;
- **Веб-ресурсы**;
- **ScilabEnterprises**;
- **О Scilab.**

Наиболее простым способом получить справку по возможностям пакета Scilab является функция **help**. Окно справки Scilab показано на Рис. 1.1.1-7. Для его отображения наберите **help** нажмите клавишу <Enter>:

```
-->help
```

Если вас интересует информация о конкретной функции (например, **optim**), вы можете пролистать содержание справки, найти раздел, посвященный оптимизации, и выбрать **optim**, после чего будет отображено окно со справкой по данной функции. Однако более удобным способом получить информацию о конкретной функции, если вы знаете ее название, является использование команды help с указанием имени интересующей функции:

```
-->help optim
```

В этом случае Scilab автоматически отобразит окно справки, содержащее информацию о выбранной функции. Если функции с указанным именем не существует, будет отображено сообщение об ошибке.

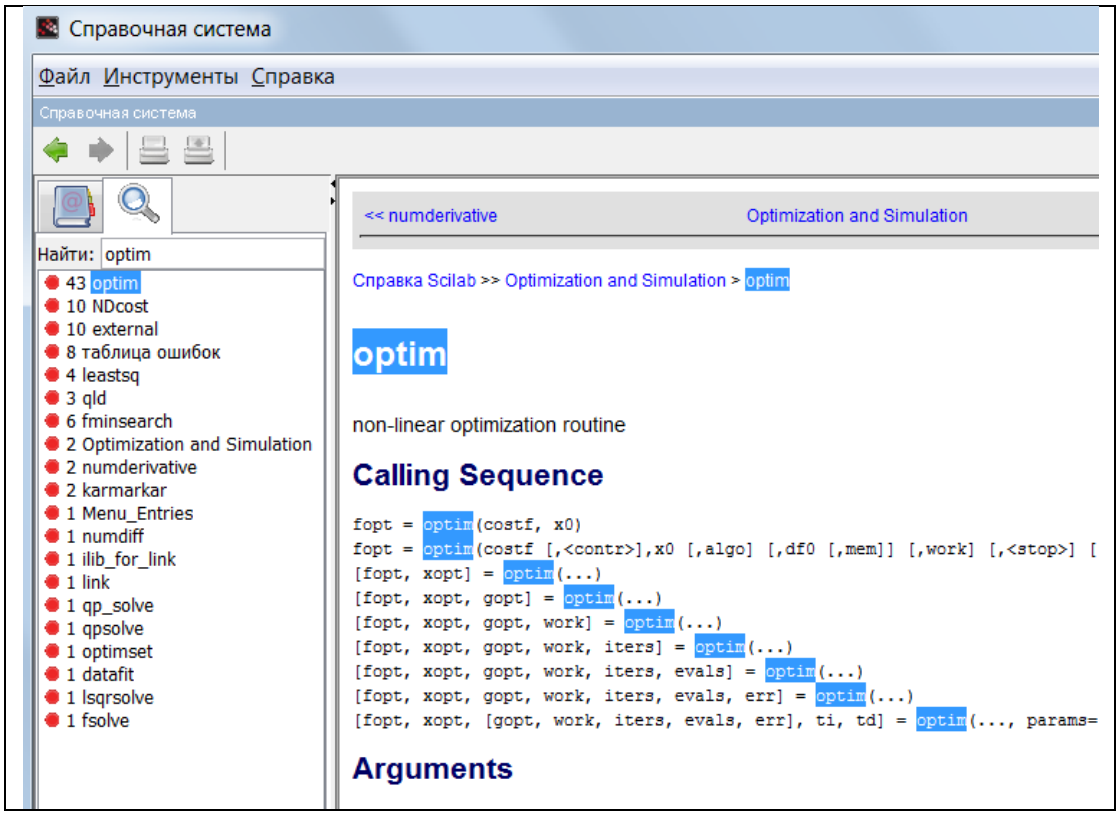

Рис. 1.1.1-7. Окно справки Scilab

Как уже было сказано выше, в стандартной конфигурации Scilab имеются четыре основных окна: **Командное окно**, окно **Обозреватель переменных**, **Обозреватель файлов** и **Журнал команд**.

Командное окно используется для ввода команд с соответствующими данными и вывода результатов их выполнения. Работа происходит в диалоговом режиме: пользователь вводит команду и передает ее ядру Scilab, ядро обрабатывает полученную команду и возвращает результат. Все команды вводятся в командную строку после появления ***приглашения*** **-->**. Заканчивается ввод каждой командной строки нажатием клавиши<Enter>.

Вышеописанный сеанс работы с Scilab в Командном окне принято называть **сессией**.

Рассмотрим несколько простых примеров, выполненных в командном окне (Рис. 1.1.1-8).

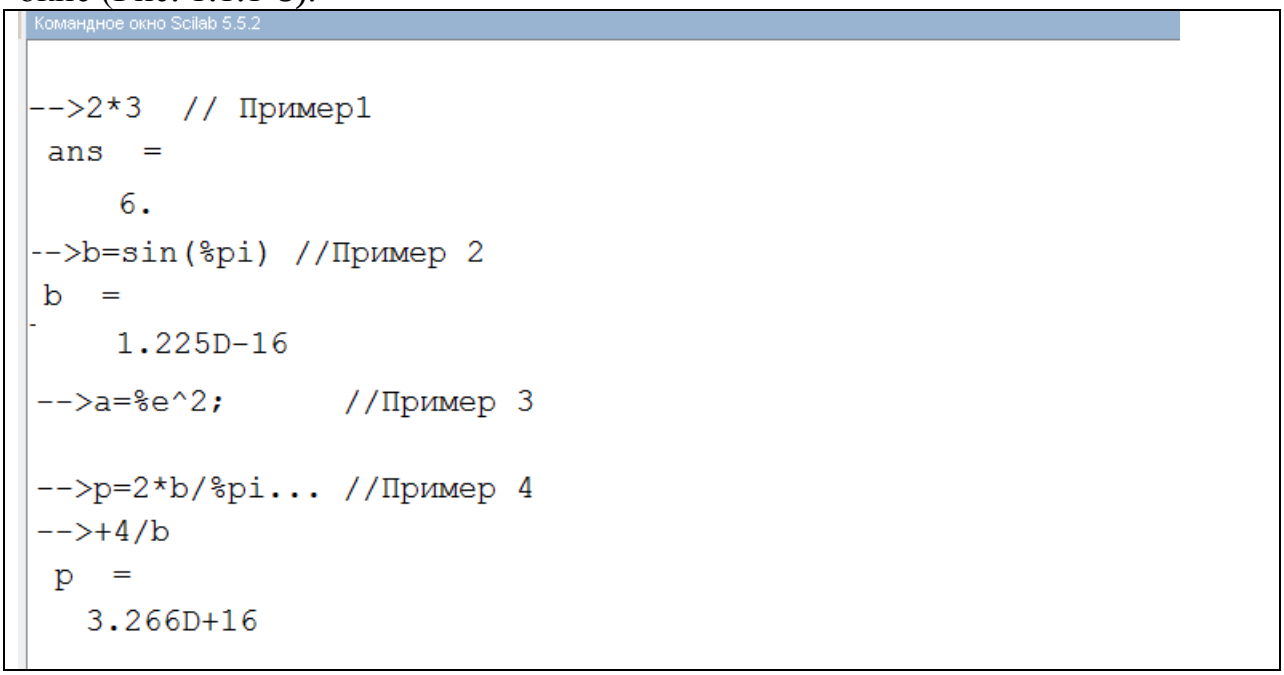

Рис. 1.1.1-8. Примеры простейших вычислений в **Командном окне**

В Примере 1 вычисляется результат выражения **2+3**. Scilab по умолчанию создаёт переменную с именем **ans**, в которую записывает значение результата текущей операции, и выводит его в следующей строке.

В Примере 2 создаётся переменная **b**, вычисляется значение выражения **sin(pi),** и результат присваивается переменной **b**.

В Примере 3 выражение заканчивает точка с запятой, которая «гасит» вывод результата, но он по-прежнему сохраняется, в этом случае в переменной с именем **r**,

Особенность Примера 4 состоит в том, что часть выражения переносится на следующую командную строку. В качестве символа переноса в командной строке Scilab используются три точки.

В двух примерах были использованы системные переменные **pi** и **e**. Чтобы указать, что эти переменные являются системными (т.е. они имеют значения по умолчанию), перед их именами вводится символ %.

Теперь все атрибуты определенные переменные появились в окне **Обозреватель переменных** и ее можно использовать для дальнейших вычислений.

Окно **Обозреватель переменных** предназначено для быстрого просмотра атрибутов переменных, расположенных в рабочей области, а также их записи в файл и чтения из файла (рис. 1.1.1-9). В этом окне можно увидеть имя переменной, значение, тип и видимость.

Команды контекстного меню окна **Обозревателя переменных** позволяют выполнять следующие действия: изменить значение переменной; удалить выделенную переменную; экспортировать в …; построить графическое изображение матрицы. При выполнении команды построения графика предлагаются различные типы графиков.

Рис. 1.1.1-9. Окно **Обозреватель переменных**

Все выполненные операции над данными отражаются не только в **Командном окне**, но и дублируются в окне **Журнал команд** (рис.1.1.1-10).

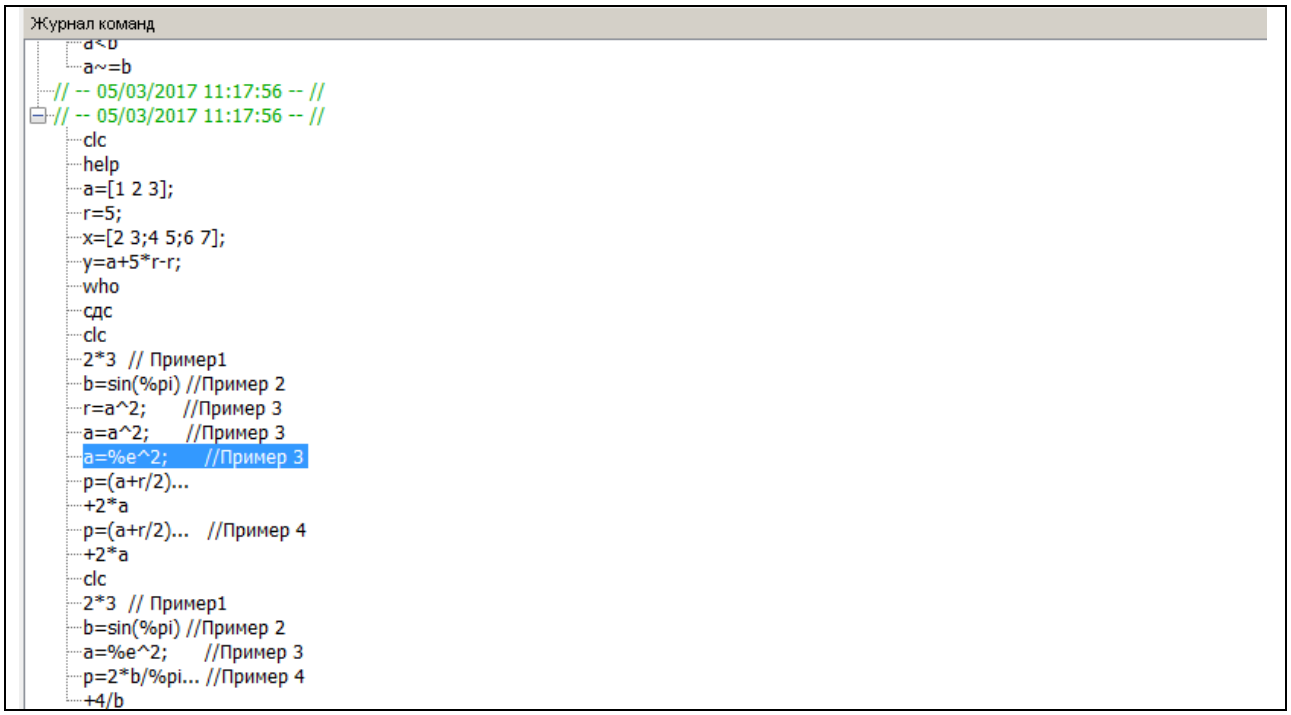

Рис. 1.1.1-10. Окно **Журнал команд**

Окно **Журнал команд** целесообразно использовать при вводе команд, однотипных вычислений или просто повторений группы командных строк. Чтобы ввести в текущую строку содержимое ранее введенной командной строки, достаточно нажатием клавиш <↑> или <↓> подобрать нужную строку. Также перенести конкретную строку в командное окно можно двойным щелчком мыши по этой строке в окне **Журнал команд**.

Окно **Обозреватель файлов** (рис. 1.1.1-11) предназначено для просмотра путей доступа файловой системы. Для установки текущего каталога можно воспользоваться кнопкой  . На кране возникает окно **Выберете папку** (рис. 1.1.1-12), в котором папка выбирается традиционным образом. Выбор папки завершается щелчком по кнопке **Открыть**, после чего окно **Обозреватель файлов** обновляется.

Открытие файла Scilab из окна **Обозреватель файлов** из окна производится двойным щелчком по его имени.

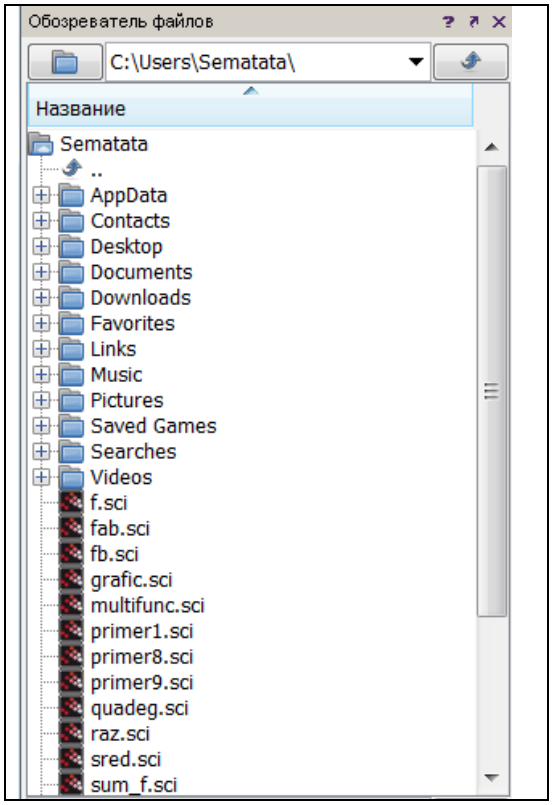

Рис. 1.1.1-11. Окно **Обозреватель файлов**

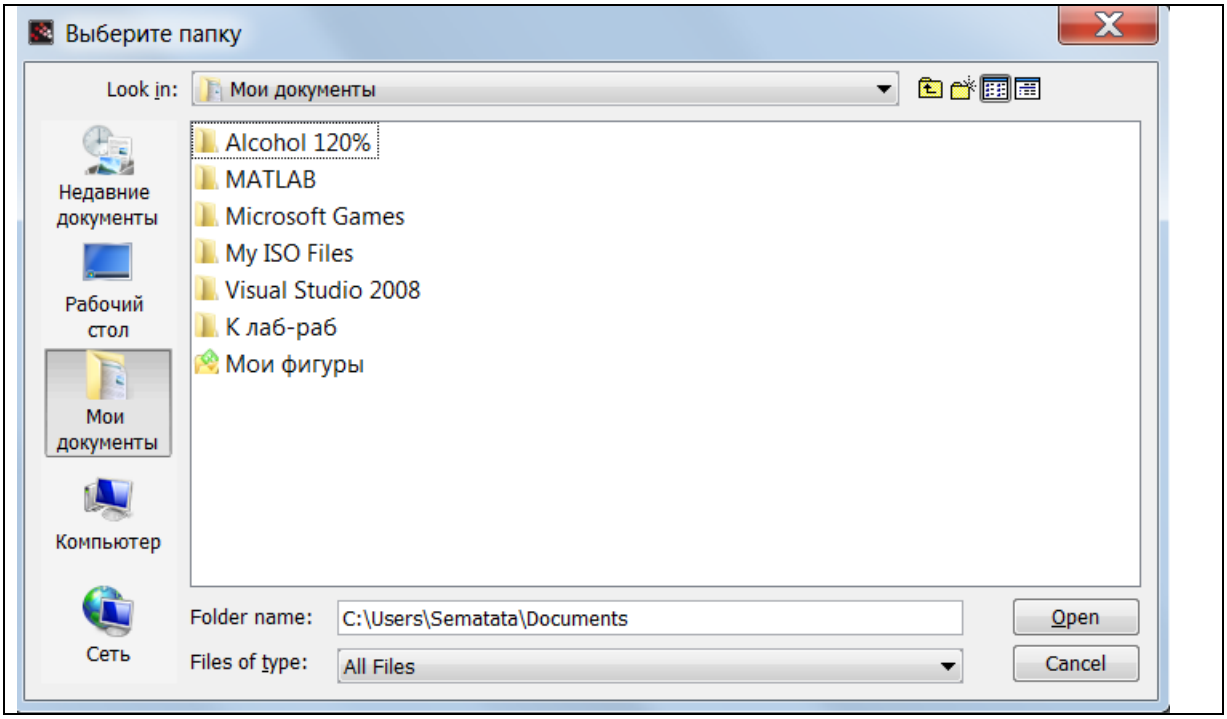

Рис. 1.1.1-12. Выбор текущего каталога

1.1.2. Основные объекты системы Scilab

Математические выражения в Scilab строятся на основе чисел, числовых констант, переменных, стандартных и нестандартных функций, соединенных знаками арифметических операций и круглых скобок. Кроме того, в математических выражениях могут использоваться различные спецзнаки. Вид результата зависит от установленного формата.

Число – объект языка Scilab**,** представляющий числовые данные. Числа могут быть представлены в целом, дробном, с фиксированной и плавающей точкой, а также в экспоненциальном виде. Например

0, 2, -4, 4.67, 0.0005, 567.9e-7, 0.89e12.

Причем числа могут быть как **действительными**, так и **комплексными.** Комплексные числа содержат действительные и мнимые части. В Scilab мнимая часть имеет множитель **i,** означающий корень квадратный из**-1.** Например,

3i, 3+5i, -i8, 0.05*%e- 0.006i.

Числовая константа – это предварительно определенное число (числовое значение). Числа (например, **1, -5, 3.97**) являются безымянными числовыми константами.

Системные константы (табл. 1.1.2-1) – это такие константы, значения которых задаются системой при загрузке, но при необходимости их можно переопределить. Имена системных констант начинаются с символа %.

Таблица1.1.2-1

Константа	Назначение
%i	Мнимая единица.
%pi	Число **π=3.1415926…**
%e	Число**e=2.141592653589793**
%eps	Условный ноль **%eps=2.220E-16**
%inf	Значение машинной бесконечности.
ans	Переменная,хранящая результат последней операции.
%NaN	Указание на нечисловой характер данных **(Not-a-Number)**

Переменные – это объекты имеющие имена. Они способны хранить некоторые, разные по значению, данные. В зависимости от этих данных, переменные могут быть числовыми или символьными, векторными или матричными. Имена переменных (идентификаторы) задаются по следующим правилам:

- имя переменной может состоять из символов латинского алфавита, знака подчёркивания и цифр, но начинаться обязательно с буквы;
- в имени переменной различаются прописные и строчные буквы;
- в имя переменной не входит пробел;
- имя переменной не должно совпадать с именами других переменных, функций, то есть должно быть уникальны;
- желательно использовать содержательные имена для обозначения переменных.

Обратите внимание, что переменные и их типы заранее не декларируются (не объявляются). Они объявляются выражением, значение которого, после вычисления, присваивается переменной.

В оперативной памяти компьютера переменные занимают определенное количество байтов. Как известно эта область называется **Обозреватель переменных.**

Операция присваивания используется для задания переменным определенных значений и обозначается знаком равенства =:

Имя переменной = *Выражение*

Имя переменной = *Выражение;*

Арифметические операции в системе Scilab можно проводить вычисления, как с вещественными, так и с комплексными числами. Полный список арифметические операции приведен в табл. 1.1.2-2.

Следует обратить внимание, что в математических выражениях с операциями +, -, * ,^, /, \ участвуют как вектора так и матрицы.

Первые восемь операций действуют в соответствии с правилами линейной алгебры, т.е. с правилами операций над векторами и матрицами.

Последние четыре операции, так называемые операции с точкой, осуществляют поэлементные операции над массивами.

Таблица 1.1.2-2

Названиефункции	Операция	Синтаксис
Плюс	+	**M1+M2**
Унарный плюс	+	**+M**
Минус	-	**M1-M2**
Унарный минус	-	**-M**
Матричное умножение	*	**M1*M2**
Поэлементное умножение массивов	.*	**A1.*A2**
Возведение матрицы в степень	^	**M^x**
Поэлементное возведение массива в степень	.^	**A.^x**
Обратное (справа налево) деление матриц	\	**M1\M2**
Деление матриц слева направо	/	**M1/M2**
Поэлементное деление массивов справа налево	.\	**A1.\A2**
Поэлементное деление массивов слева направо	./	**A1./A2**

Подробнее операции над векторами и матрицами рассмотрены в **Теме 1.2**.

Приоритет исполнения математических операций в Scilab аналогичен приоритету в языках программирования высокого уровня: возведения в степень, умножения и деления, сложения и вычитания. Для изменения приоритета в математических выражениях используются круглые скобки.

Функции – это имеющие уникальные имена объекты, выполняющие определенные преобразования своих аргументов (параметров) и при этом возвращающие результаты этих преобразований. **Возврат результата**–

отличительная черта функций. Функции могут быть ***встроенными*** (внутренними) ***функциями*** системы Scilab***, функциями пользователя***, которые могут дополнять встроенные функции и описанные конкретным пользователем для своих нужд, ***внешними функциями***, или ***sci-функциями***.

Scilab обладает большим набором элементарных ***встроенных (библиотечных) функций,*** в том числе математических**.** Встроенные функции хранятся в откомпилированном виде в **ядре** системы Scilab. Некоторые из них приведены в табл. 1.1.2-3.

Таблица 1.1.2-3

Тригонометрические функции (аргумент задается в радианах)	
sin(), cos(), tan(), cotg()	Синус, косинус, тангенс и котангенс
sec(), csc()	Секанс, косеканс
Обратные тригонометрические функции (результат в радианах)	
asin(),acos(),atan(), atan2(), acot()	Арксинус, арккосинус, арктангенс и арккотангенс
asec(), acsc()	Арксеканс, арккосеканс
Алгебраические и арифметические функции	
abs(), sign()	Модуль и знак числа
exp()	Экспоненциальная функция
log(), log2(), log10()	Логарифм натуральный, по основанию 2 и 10
sqrt()	Квадратный корень
fix()	Целая часть числа
floor()	Округление до ближайшего целого значения, которое не превышает аргумент
sign()	Знак числа
factorial()	Вычисление факториала числа
Гиперболические функции	
sinh(), cosh(), tanh(), coth()	Гиперболические синус, косинус, тангенс и котангенс
sech(), csch()	Гиперболические секанс и косеканс
asinh(), acosh(), atanh(), acoth()	Гиперболические арксинус, арккосинус, арктангенс и арккотангенс;
Функции для работы с комплексными числами	
conj()	Комплексно-сопряжённое число
imag()	Мнимая часть числа
real()	Вещественная часть числа
gcd(), lcm()	Наибольший и наименьший общий делитель
angle()	P = angle(Z) возвращает фазу угла в радианах. Для комплексного Z, модуль R и фаза угла P Связаны следующим образом: R = abs(Z) theta = angle(Z) Z = R.*exp(i*theta)
isreal()	Возвращает логическую 1, если число действительное и 0 – если комплексное.

Система Scilab предоставляет пользователю средства для создания и использования своих собственных функций – так называемых ***функций пользователя***. О создании и использовании функций в виде **m-файлов** речь пойдет в главе 1.1.3.

Рассмотрим несколько примеров рис. 1.1.2-1.

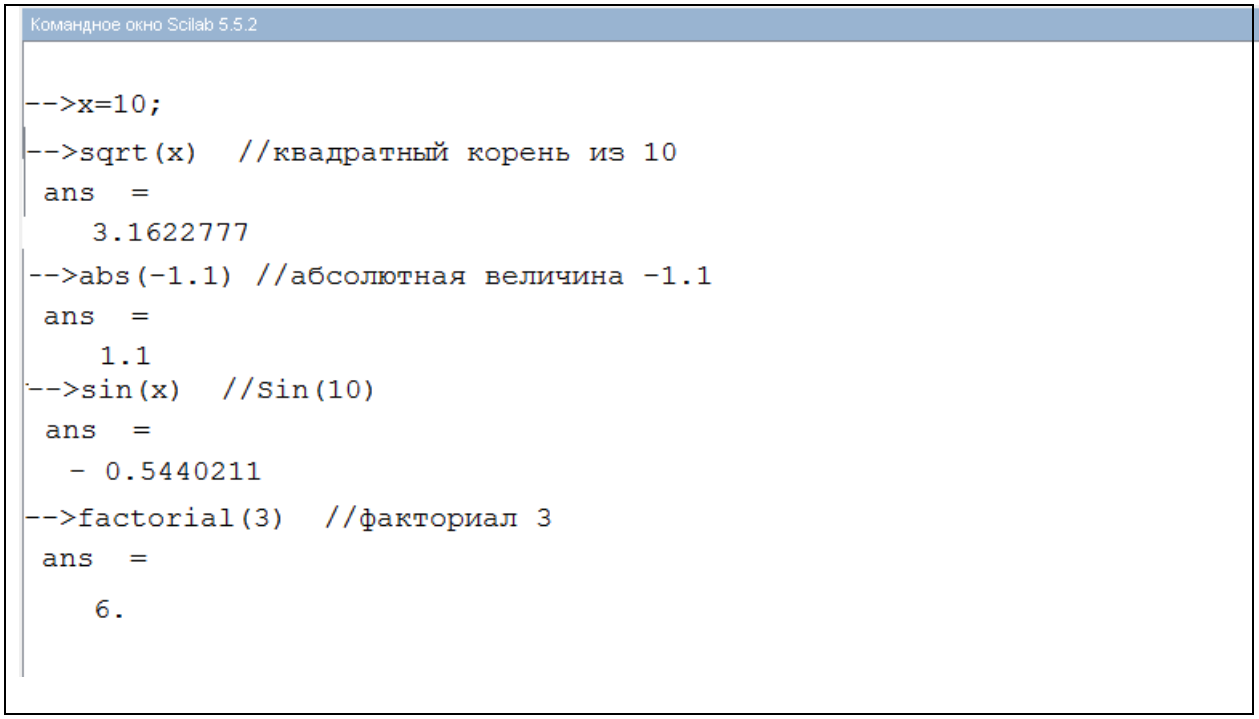

Рис. 1.1.2-1. Примеры использования встроенных функций Scilab

Кроме математических выражений в Scilab используются *логические выражения.*

Логические выражения в Scilab строятся на основе математических выражений, операций отношения и соответствующим им функций, логических операций и функций, а также круглых скобок. Результатом логического выражения является значение равное **T,** если выражение Истинно (True), или значение **F** в противном случае – выражение Ложно (False). Если операнды – действительные числа, то использование операций аналогично.

Операции отношения и соответствующие им функции служат для сравнения двух величин, векторов и матриц. Список операций отношений приведен в табл. 1.1.2-4.

Таблица 1.1.2-4

Функция	Операция	Описание	Пример
eg()	==	Равно	x==y; eg(x,y);
ne()	~=	Не равно	x~=y; ne(x,y);
lt()	<	Меньше чем	x<y; lt(x, y);
gt()	>	Больше чем	x>y; gt(x,y);
le()	<=	Меньше или равно	x<=y; le|(x,y);
ge()	>=	Больше илиравно	x>=y; ge(x,y);

Операции отношений выполняют поэлементное сравнение векторов или матриц одинакового размера и возвращает значение равное 1, если элементы идентичны, и значение 0 в противном случае.

Следует отметить, что операции <, <=, >, >= при комплексных операндах используются для сравнения только действительные части операндов – мнимые отбрасываются. В то же время операции == и ~= ведут сравнения с учетом как действительной, так и мнимой частью операндов (рис. 1.1.2-2).

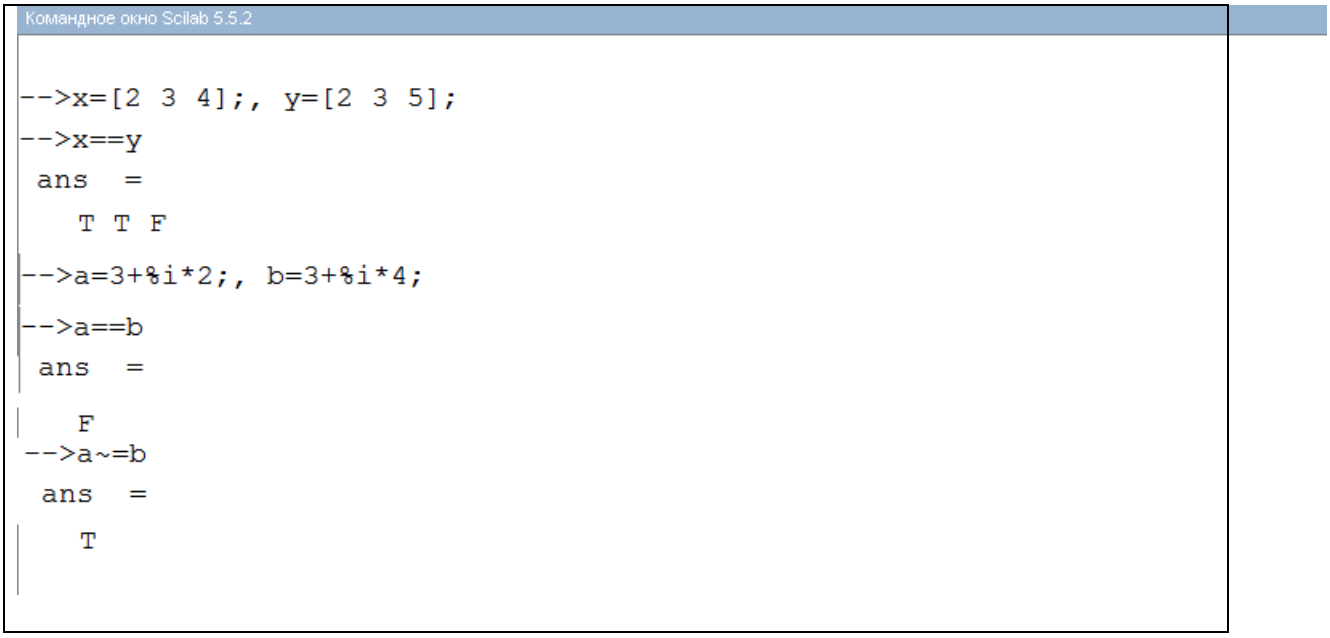

Рис. 1.1.2-2. Примеры операций отношения над векторами и комплексными числами

Операции отношения обычно применяются в логических выражениях, которые являются условиями операторов **if, for, while, switch**и служат для изменения последовательности выполнения операторов программы Scilab. Использование этих операторов подробно рассмотрено в **Теме 1.3**.

При вычислении выражений операции отношений имеют более низкий приоритет чем арифметические, но более высокий чем логические операции.

Логические операции и соответствующие им функции служат для поэлементных логических операций над элементами одинаковых по размеру массивов. Список логических операций отношений приведен в табл. 1.1.2-5.

Таблица 1.1.2-5

Функция	Операция	Описание	Пример
and()	&	Логическое умножение (И)	>>x&y ans = TFF
or()	!	Логическое сложение (ИЛИ)	>>or(x,y) ans = TTT
not()	~	Логическое НЕ	>> ~x ans = FTT

Логические функции дополняют логические операции и представлены в табл. 1.1.2-6 где в примерах используются вектора x=[1 0 0] и y=[111].

Таблица 1.1.2-6

Функция	Описание	Пример
xor ()	Исключающее ИЛИ	>>xor(x,y) ans = FTT
all()	Верно, если все элементы вектора не равны 0	>>all(x) ans = F
any()	Верно, если все элементы вектора равны 0	>>any(x) ans = T
find()	Нахождение ненулевых элементов в векторах	>>find(x) ans = T

Специальные операции реализуются с помощью специальных символов (табл. 1.1.2-7). Они предназначены для создания самых разнообразных объектов входного языка системы Scilab и придания им различных форм.

Таблица 1.1.2-7

Обозначение	Название
:	**Двоеточие** – формирование под векторами и подматриц.
( )	**Круглые скобки** используются для задания порядка выполнения операций в выражениях, указания последовательности аргументов функции и указания индексов элемента вектора или матрицы.
[ ]	**Квадратные скобки** –формирование векторов и матриц.

{ }	**Фигурные скобки** –формирование массивов ячеек**.**
.	**Десятичная точка** используется для отделения дробной части чисел от целой.
..	**Две точки подряд** указывают на родительский каталог – переход по дереву каталогов на один уровень вверх.
...	**Три точки подряд** указывают на продолжение строки.
,	**Запятая** – разделитель элементов.
;	**Точка с запятой** используется внутри круглых скобок для разделения строк матриц, а также в конце операторов для запрета вывода на экран результата вычислений.
%	**Знак процента** – начало комментария.
=	**Символ равно** используется для присваивания (операция присваивания) значений в выражениях.
'*текст*'	**Одиночные кавычки(апострофы),внутри которых заключен текст,** интерпретируемый как вектор символов с компонентами, являющимися символами. Кавычка внутри строки задается двумя кавычками
'	**Кавычка (апостроф)**–транспонирование матриц (A'). Для комплексных матриц транспонирование дополняется комплексным сопряжением, т.е. строки транспонированной матрицы соответствуют столбцам исходной матрицы.
. '	**Точка с кавычкой (точкам с апострофом)**–транспонирование массива (A.') Для комплексных массивов операция сопряжения не выполняется.
[,]	**Квадратные скобки с перечислением внутри их элементов через запятую** –горизонтальная конкатенация– объединение матриц A и B[A,B].
[;]	**Квадратные скобки с перечислением внутри их элементов через точкой с запятой** – вертикальная конкатенация– объединение матриц A и B[A;B].

Функции пользователя **–** это функции, создаваемые пользователем системы, которые значительно облегчают работу за счет частого использования в математических выражениях, получении таблиц значений функций и построении графиков.

Как правило, функции создаются для обработки множества значений аргументов, поэтому при описании их математических выражений используются операции с точкой. Функцию можно создать внутри командного окна. Начинается она со слова **function,** а заканчивается словом **endfunction.** Первая строка имеет следующий формат:

function [y1,…,yn]=Имя(x1,…xk),

где ***yi*** – выходные аргументы и ***xi***–входные аргументы.

Функция может иметь несколько аргументов, тогда они перечисляются через запятую. Имя функции формируется аналогично имени переменной и должно быть уникально.

Создание функции с использованием команды **deff** имеет следующий формат:

deff('[w]=fun(x1,…,xk)',['onepamop1';…' onepamopk'])

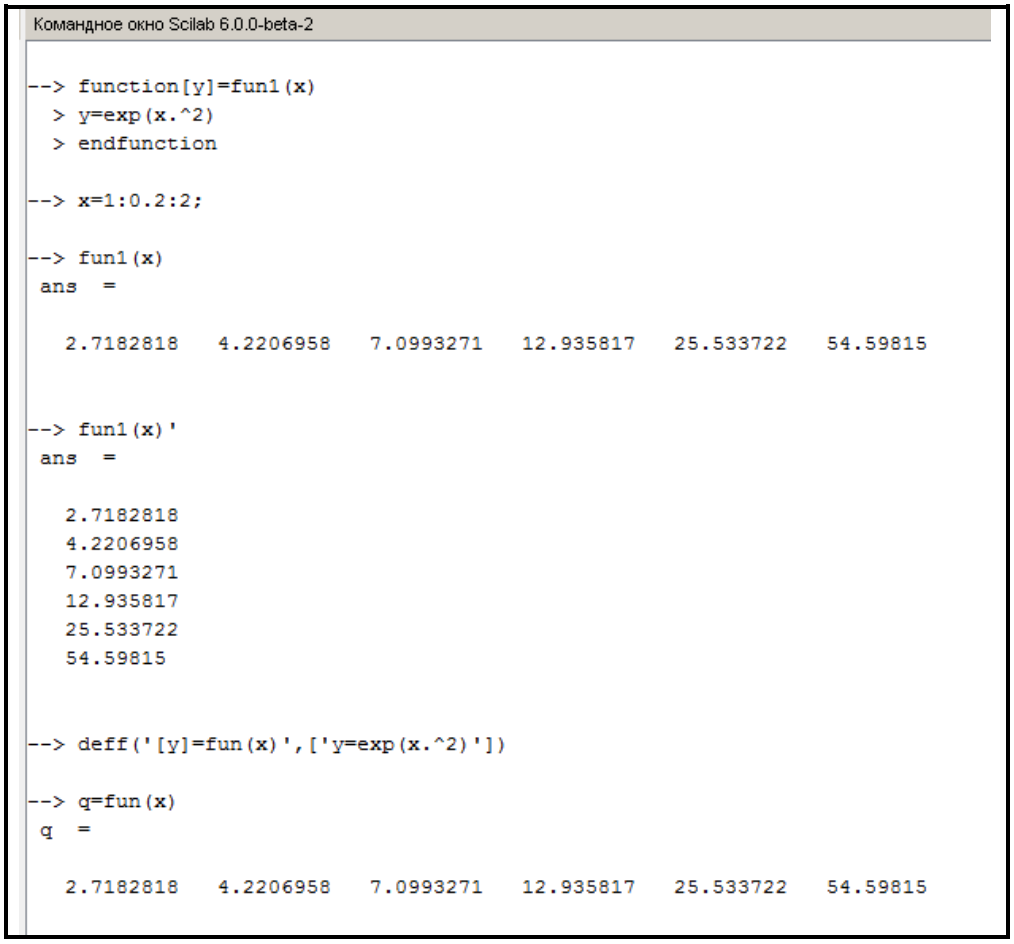

Рис.1.1.2-3. Примеры описания и использования функции

В примере (рис. 1.1.2-3) описания функция $y(x)=e^{x^2}$ и получения ее значений для множества значений аргумента **x**с использованием описанных выше средств. Традиционное обращение к функции выводит значения функции в строку, а добавление символа апостроф (') – в столбец.

Наряду с рассмотренными выше функциями пользователя в Scilab имеется возможность создания функций (или некоторой последовательности вычислений) в виде **sci-файлов**, которые можно сохранять и использовать в других сеансах работы. Подробнее материал о создании и работе с **sci**-файлами, а также о средствах программирования в среде Scilab, изложен в **Теме 3.1**.

Символьная константа – это последовательность символов, заключенных в одиночные апострофы. Например, **'Информатика'**

Комментарии в Scilab определяются с помощью символов//. Например,

% **Это комментарий**

1.1.3. Лабораторная работа по теме «Элементы рабочей среда Scilab и простейшие вычисления»

1. Вопросы, подлежащие изучению

1) Элементы основного окна Scilab.
2) Окно панели **Командного окна**.
3) Установка свойств среды системы Scilab.
4) Основные объекты системы Scilab.
5) Правила записи и вычисления арифметических выражений.
6) Работа с функциями пользователя, заданными в окне ***Командное окно***.
7) Назначение окон ***Обозреватель переменных*** и ***Журнал команд*.**

2. Общее задание

1) ***Изучите материал Темы 1.1 (п.п. 1.1.1 – 1.1.2).***
2) ***Выполните команду clear all*** *для очистки* ***Рабочей среды***.
3) ***Выберите вариант*** *задания формул из* ***табл. 1.1.3-1***.
4) ***Задайте*** *переменным* ***x*** *и* ***y*** *допустимые числовые значения.*
5) ***Проанализируйте информацию****, возникшую в окне* **Обозреватель переменных.**
6) ***Введите формулу*** *для вычислений арифметического выражения и получите результат.*
7) ***Измените значения*** *исходных данных.*
8) ***Измените формат*** *вывода результата, выполнив команду* ***format long****, и* ***произведите перерасчет*** *значения выражения.*
9) ***Верните формат*** *вывода данных* ***short.***
10) ***Представьте*** *арифметическое выражение в виде правой части функции* ***f(x).***
11) ***Опишите функцию f(x)*** *с помощью оператора* ***deff*** *числовое значение при новом значении переменной* ***x.***
12) ***Опишите функцию f1(x)*** *с помощью оператора* ***function.***
13) ***Измените*** *значение переменной* ***y***, ***вычислите*** *значения выражения* ***b*** *и функции* ***f1(x)***.
14) ***Объясните****, почему изменение значения* ***y*** *привело к изменению значении* ***b,*** *но не повлияло на значение функции.*
15) ***Задайте диапазон изменения*** *аргумента функции с шагом, позволяющим получить таблицу значений функции* ***f(x)*** *и* ***f1(x)*** *для заданных значений аргумента (порядка 8-10 точек),* ***выведете значения*** *функции* ***f(x)*** *в выбранном интервале вначале в строку, а затем в столбец.*

16) ***Выполните команду who*** *и* ***проанализируйте выведенную*** *информацию о данных***.**
17) ***Установите путь*** *к папке, находящейся на вашей флешь-карте, для сохранения содержимого* ***Командного окна***.
18) ***Создайте файл*** *в текущей папке*.
19) ***Выделите в окне Командного окна*** *с использованием команд* ***diary('имя_файла')*** *и* ***diary(0)****область сохранения*.
20) ***Сохраните текст*** *рабочего окна на внешнем носителе*
21) ***Предоставьте результаты*** *работы преподавателю, ответьте на поставленные вопросы.*
22) ***Выполните команду clear all*** *для очистки* ***Рабочей среды***.
23) ***Оформите отчет*** *по выполненной работе***.**

3. Варианты индивидуальных заданий

Таблица 1.1.3-1

№	Формулы для вычислений	№	Формулы для вычислений
1.	$t = \cos\frac{\pi}{7} * \frac{\sin^2(x-8y)}{2{,}7(x-\pi)}$	**16.**	$b = \frac{\lg x - \sin^2 xy}{0{,}8 \cdot \ln(1-x)^2}$
2.	$d = \frac{(1-e^{xy})^2}{0{,}7\lg\lvert 1-x^2\rvert}$	**17.**	$d = 10^4 \cdot \frac{e^{-\frac{x}{2y}} + \sqrt{\lvert \sin y^3 \rvert}}{2{,}5\cos^2 x}$
3.	$h = \frac{xy + \sin x}{\lvert 1-y\rvert * \ln x}$	**18.**	$f = \frac{\frac{\pi}{3} + \ln x^3}{3y - x} + x \cdot \sin y^2$
4.	$c = \frac{(yx^2-1)^2}{2} \cdot (\cos^2 y - \sin x^2)$	**19.**	$h = \frac{208 \cdot \lg x + x^2}{\lvert x - y^2\rvert - e^{-y}}$
5.	$b = \sqrt[3]{\frac{x+y}{0{,}2x}} \cdot \sin(tg^2 x)$	**20.**	$a = 10^5 \cdot \lg\lvert 0.8x\rvert \cdot e^{\frac{-x^2}{2xy}}$
6.	$d = \frac{xe^{xy} + 8\sin^2 x}{x(x-y)(3x+y)}$	**21.**	$b = \frac{x^y}{1 - \frac{1}{e^{-x+\sin y}}}$
7.	$z = \frac{\pi}{2} - \sqrt{2x} - \frac{x + y^2}{0{,}75tg\lvert x+y\rvert}$	**22.**	$c = x \cdot \lg\lvert x-6\rvert - \frac{\sin x^2}{yx^3}$
8.	$d = \frac{xy^2 - \sqrt{\lvert x^2 - 2{,}5 \cdot 10^{-3} y\rvert}}{2\sin xy} + 0.5$	**23.**	$a = \frac{14 \cdot \sin x + y^2}{0.92 \cdot \cos^3 x}$
9.	$f = 5{,}2^3 \cdot \frac{\lg(x+y)}{x - \frac{1}{0{,}45\sin(x-8y)}} + 0.5$	**24.**	$a = \frac{x^2\text{-}xy}{0.7\sin\ln\lvert x\rvert}$
10.	$a = 0{,}8 \cdot 10^{-5}(xe^{-x(y-1{,}2)} - yx)^3$	**25.**	$c = \frac{2.71x^2 - \cos y}{tg(x^2) \cdot e^{-y}}$

11.	$d = \frac{\sqrt{\lvert x \rvert} + e^{-y}}{5{,}8 \cdot \cos y^3}$	**26.**	$d = \frac{1 - tgxy^2}{\sqrt[3]{x}} + 4\sqrt{x^2 - 0{,}1}$
12.	$f = -\frac{2x^2 - \sin x^2}{2 - e^{-y}}$	**27.**	$f = 0.5 + \frac{1}{2} \cdot \cos\frac{1 - \sin xy^2}{1 + \sin^2 xy}$
13.	$h = \frac{\sin^3 x + e^{-\sin y}}{0{,}6x^2y^2}$	**28.**	$g = x \cdot e^{-y} + \frac{(x+y)^2}{2 \cdot \cos^3 x}$
14.	$a = 10 \cdot \frac{\ln y^2 - \sqrt[4]{\lvert x - y \rvert}}{1 - \cos^3 y}$	**29.**	$z = \frac{x - y}{\sqrt{x + y}} + \frac{xy^2}{\sin x^2 \cdot \cos^2 y}$
15.	$c = \frac{1}{2\pi} - x\sqrt{2{,}5 \cdot 10^3 y} \cdot \lvert \cos x^3 \rvert$	**30.**	$b = \left\lvert \pi - \frac{x}{3} \right\rvert \cdot e^{\frac{1 - \sin e^{-y}}{2x}}$

4. Содержание отчета

1) В начале сессии введите в формате комментариев:
- название лабораторной работы;
- ФИО студента, номер группы;
- № варианта;
- индивидуальное задание.

2) Протокол вычислений (сессии) в ***Командном окне*** в соответствии с общим заданием, снабженный подробными комментариями.

1.1.4. Контрольные вопросы по теме

1) Назначение ***Командного окна***.
2) Назначение окна ***Журнал команд***.
3) Окно отображения информации о данных.
4) Назначение команд ***who***.
5) Каким образом установить формат для вывода числовых значений на экран?
6) Каким образом перенести командную строку из окна ***Журнал команд*** в окно ***Командное окно***?
7) Каким образом установить формат для вывода числовых значений на экран?
8) Форматы числовых данных в системе Scilab.
9) Что такое системные константы?
10) Что такое символьные константы?
11) Команда объявления символьных переменных.
12) Какой символ используется для определения комментария?
13) Какая из операций **.*** или ***** является операций поэлементного умножения?

14) Какой символ предназначен для запрета вывода результата выполнения команды на экран?
15) Создание функций с помощью команды ***deff***.
16) Создание функции с помощью ***function***.

Тема 1.2. Вектора, матрицы и построение графиков в системе Scilab

1.2.1. Вектора и матрицы
1.2.2. Построение графиков и визуализация вычислений в системе Scilab
1.2.3. Лабораторная работа по теме
1.2.4. Контрольные вопросы по теме

1.2.1. Вектора и матрицы

Scilab построена как система, ориентированная на работу с матрицами, то есть все численные вычисления производятся в матричной форме. Даже обычные числа и переменные в Scilab рассматриваются как матрицы размера **1 x 1**. К особенностям работы с массивами в Scilab относится то, что одномерный массив может быть **вектором-строкой** или **вектором-столбцом (**рис. 1.2.1-1).

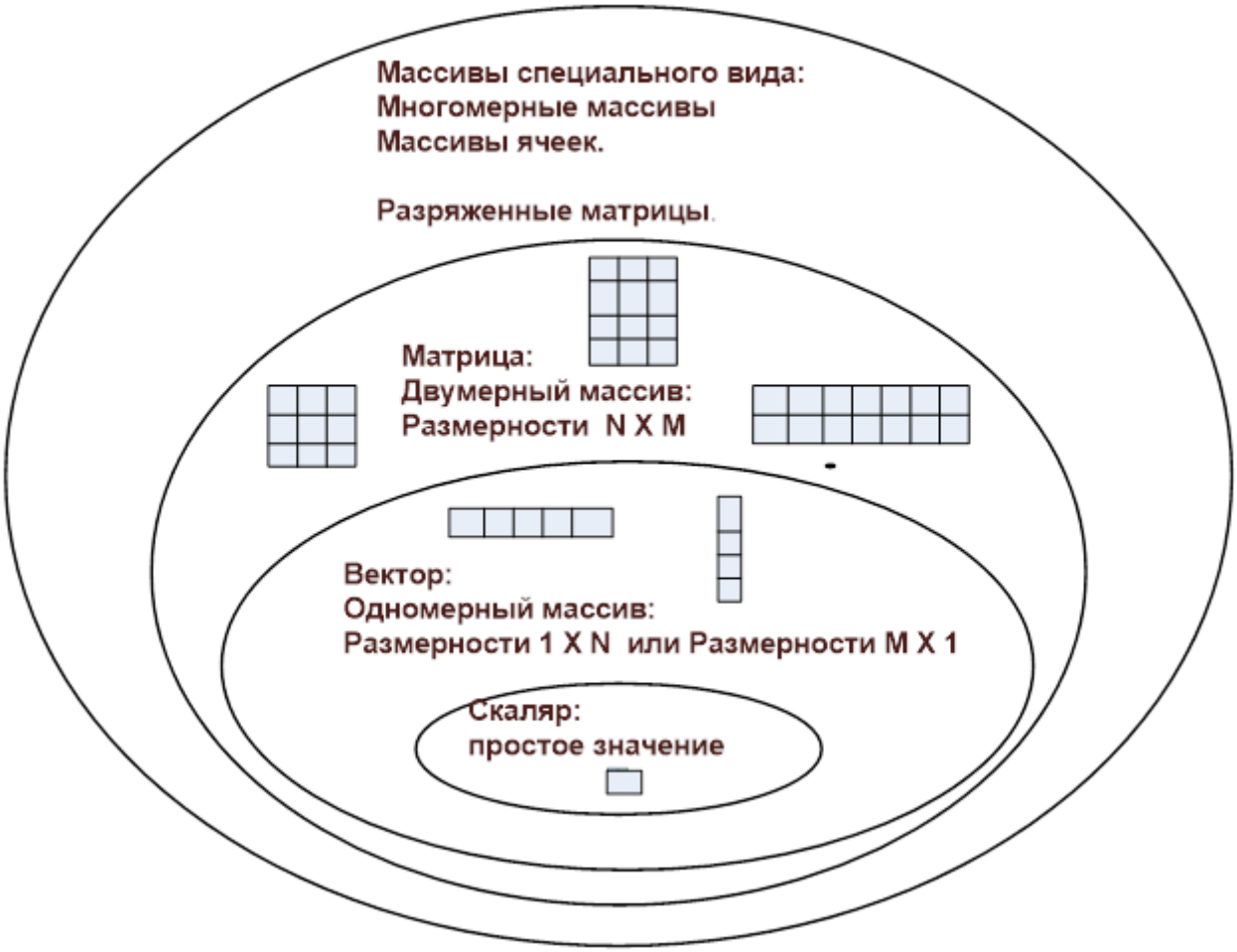

Рис. 1.2.1-1.Представление данных в Scilab: переменные, вектора (одномерные массивы) и матрицы (двумерные массивы)

Для определения вектора используются операция квадратные скобки (операция объединения), а элементы вектора (рис.1.2.1-2) отделяются друг от друга:

- точкой с запятой, если нужно получить вектор-столбец;
- пробелом или запятой, если нужно разместить элементы в векторе-строке.

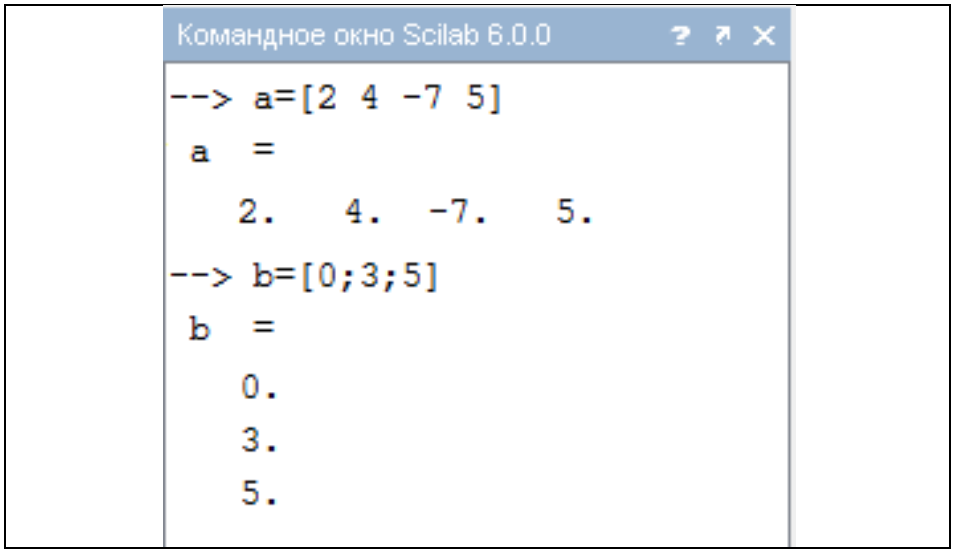

Рис. 1.2.1-2. Создание вектора-строки **a** и вектор-столбца **b**

Для определения длины вектора используется функция **length(a)**, где **a** – имя вектора, а для операции транспонирования используется символ апостроф (') (рис.1.2.1-3).

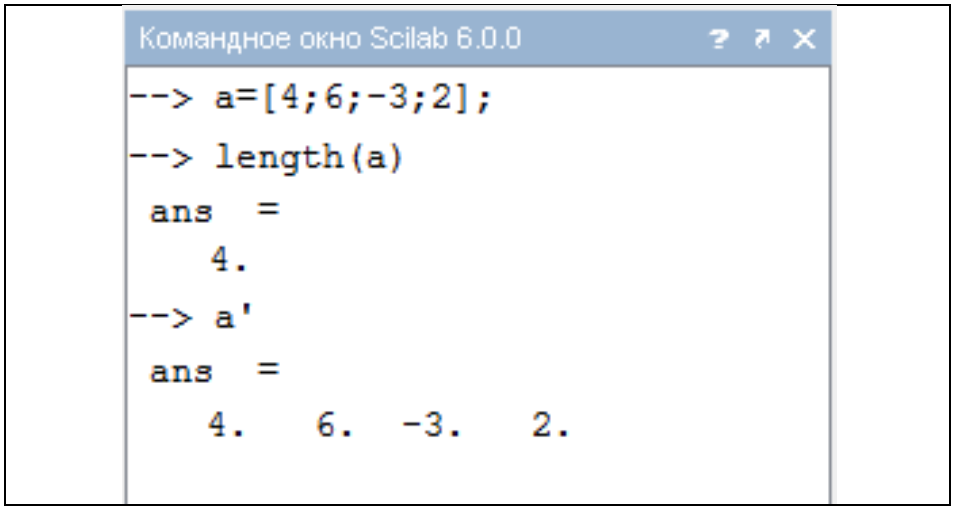

Рис. 1.2.1-3. Определение длины вектора и его транспонирование

В следующем примере (рис. 1.2.1-4.) при описании вектора **x** символ двоеточие, поставленный между двумя числами, указывает, что его элементы

последовательно принимают значения, начиная от первого числа (0) до последнего числа (5) с шагом **1** (по молчанию шаг равен 1). При описании вектораиспользован шаг **0.5** и выведены значения элементов вектора.

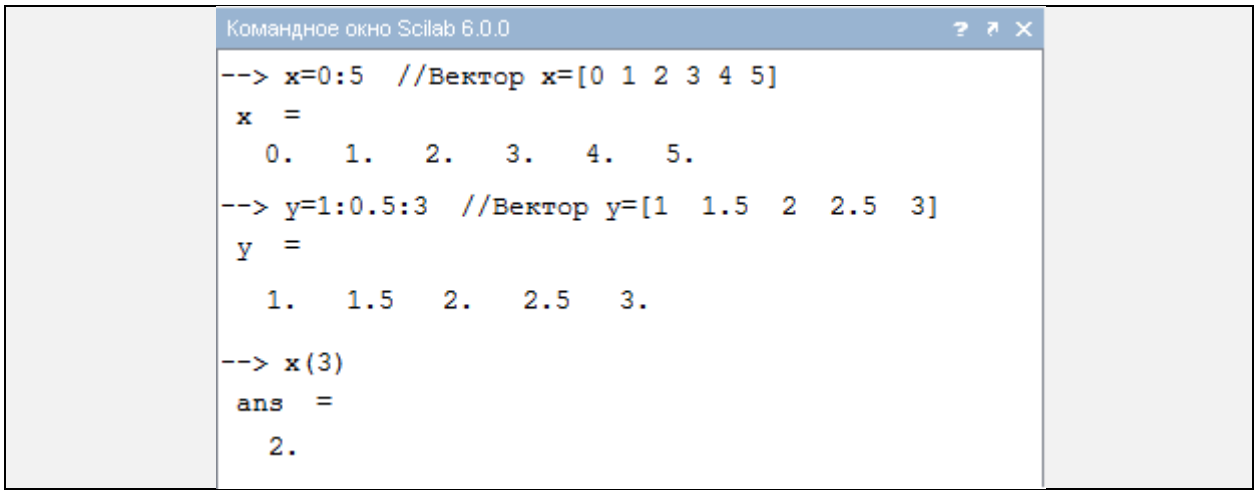

Рис. 1.2.1-4. Способы описания векторов с постоянным шагом

Для описания матрицы необходимо ввести ее имя и знак присваивания, а затем в квадратных скобках значения ее элементов. При этом значения элементов строк вводятся через пробел, а строки матрицы разделяет символ точка с запятой (**;**) (рис.1.2.1-5):

A=[v1;v2;v3], где **v1, v2, v3** -векторы одинаковой размерности.

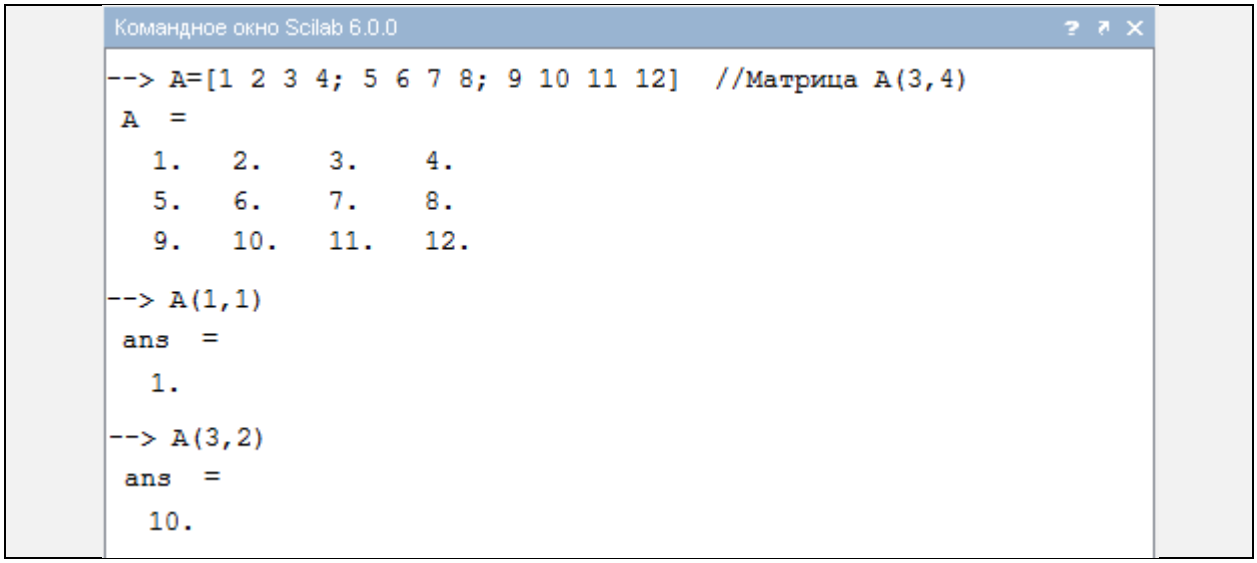

Рис. 1.2.1-5. Создание матрицы **A(3,4)** и доступ к ее элементам

Следует помнить, что нумерация элементов матрицы (в строках и столбцах) начинается с 1.

Scilab обладает большим набором встроенных функций для обработки векторных данных. Полный список имеющихся функций выводится в справке (Справка Scilab>>Основные функции> Элементарные матрицы). Некоторые из них приведены в таблице 1.2.1-1.

Таблица 1.2.1-1

Функции операции над векторами	
prod(V)	Произведение элементов вектора
sum(V)	Сумма элементов вектора
min(V)	Минимальный элемент вектора
max(V)	Максимальный элемент вектора
mean(V)	Среднее значение элементов вектора
gsort(V)	Сортировка элементов вектора по убыванию
Функции определения матриц и операций над ними	
det(A)	Вычисляет определитель квадратной матрицы
rand (n,m)	Возвращает матрицу, элементы которой распределены по равномерному закону
rand(n,m,"normal")	Возвращает матрицу, элементы которой распределены по нормальному закону
size(A)	Определяет число строк и столбцов матрицы**A**, результат – вектор **[n;m]**
sum(A [,k])	Формирует вектор-строку (k – есть) или вектор-столбец (k- нет), каждый элемент которого – сумма элементов строки или столбца
min(A)	Формирует вектор-столбец из минимальных элементов строк
max(A)	Формирует вектор-столбец из максимальных элементов строк
gsort(A)	Упорядочивает элементы столбцов по убыванию
norm(A [,p])	Возвращает норму матрицы (по умолчанию вычисляется 2-я норма)
inv(A)	Возвращает матрицу, обратную A

Примеры использования некоторых функций над векторами приведены на рис. 1.2.1-6.

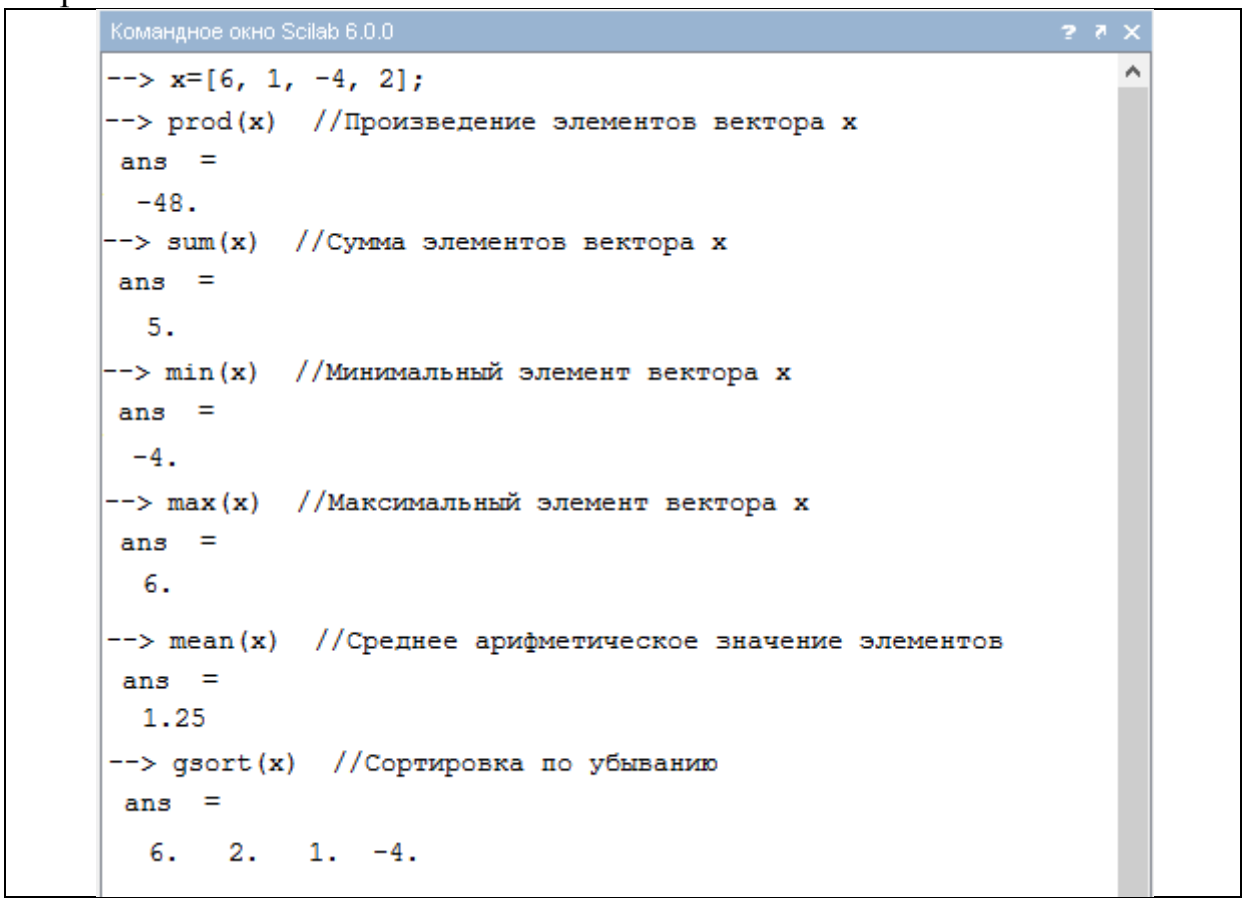

Рис. 1.2.1-6. Примеры функции над векторами

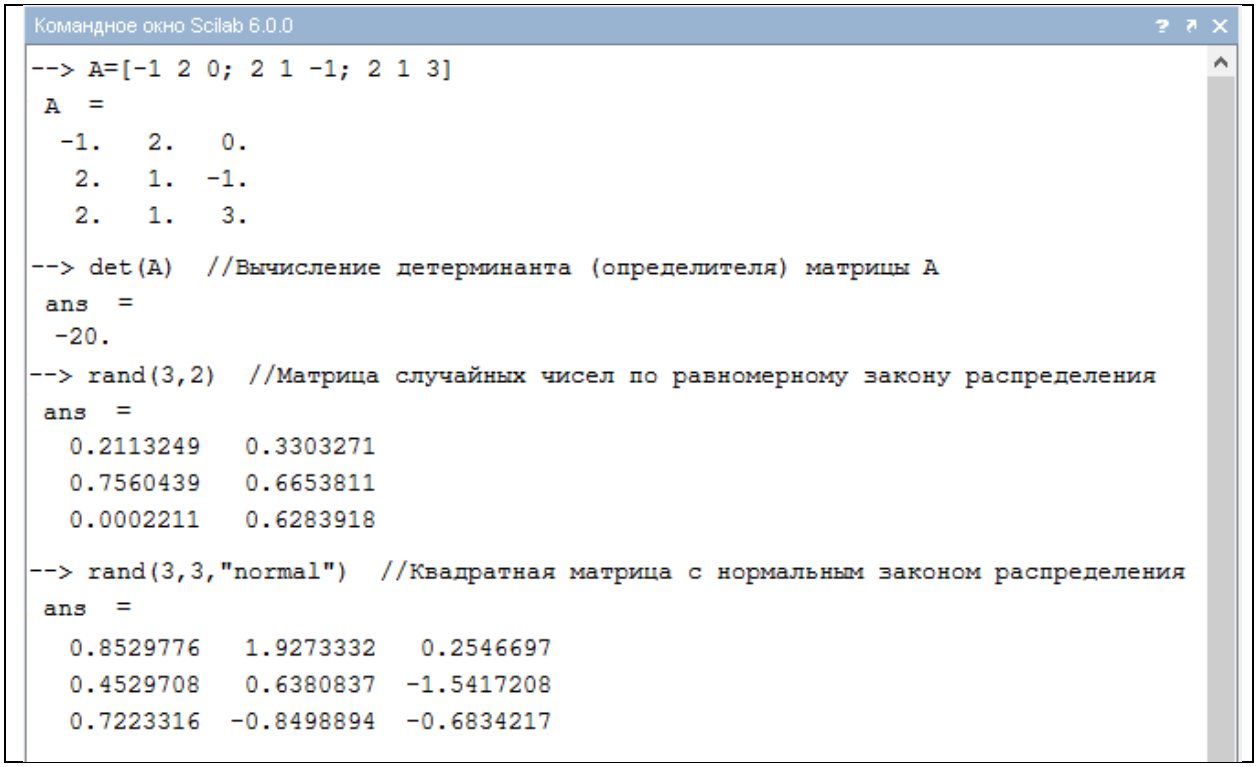

Рис. 1.2.1-7. Примеры определения матриц

На рис.1.2.1-7 приведены примеры вычисления определителя квадратной матрицы и заполнения матриц случайными числами, сгенерированных по равномерному и нормальному законам.

В следующем примере (рис. 1.2.1-8) показано использование функции Scilab **sum()**.

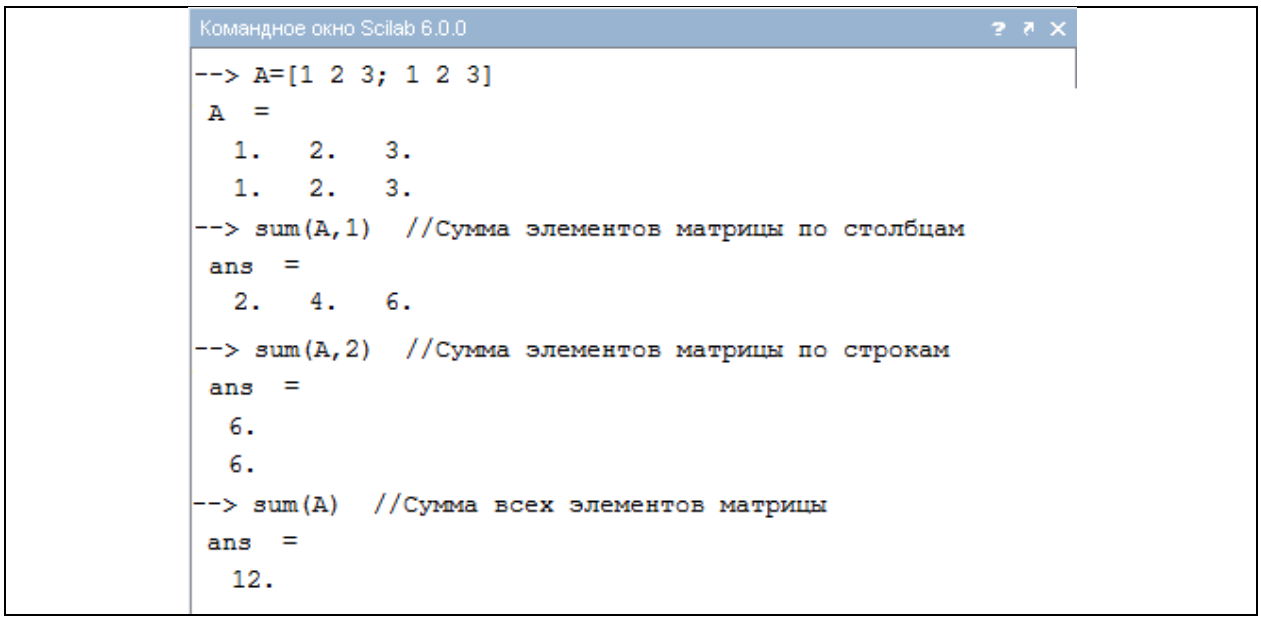

Рис.1.2.1-8. Варианты использования функции Scilab **sum()**

Пример, приведенный на рис.1.2.1-9 демонстрирует работу функций, определяющих минимальное и максимальное значения матриц.

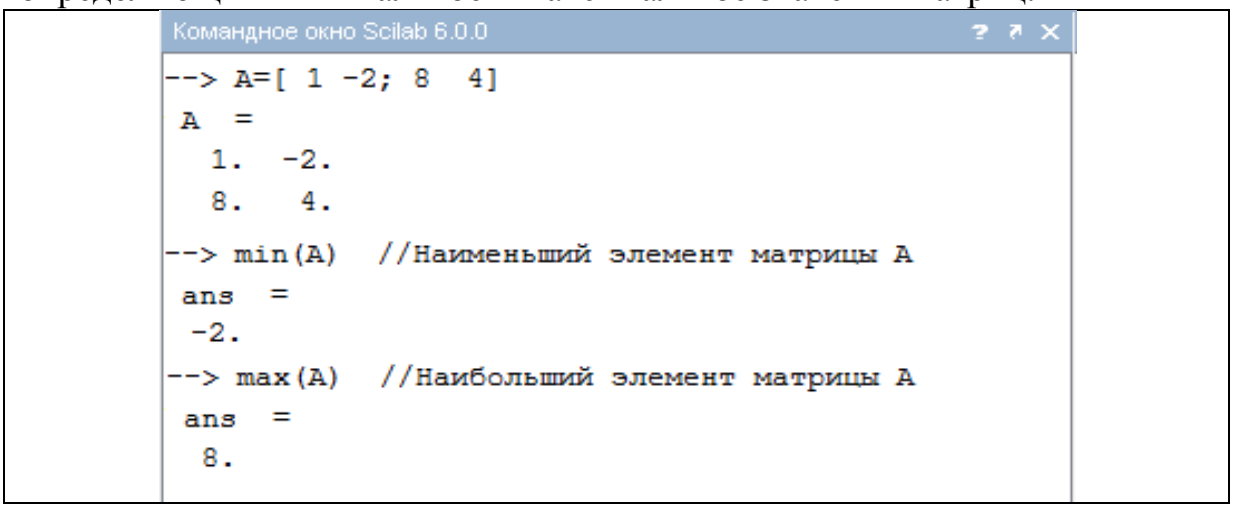

Рис.1.2.1-9. Определение минимальных и максимальных значений

На рис.1.2.1-10 показаны применения функций, позволяющих определять средние значения элементов в столбцах (или в строках) и проводить упорядочение (сортировку) элементов в строках (или столбцах).

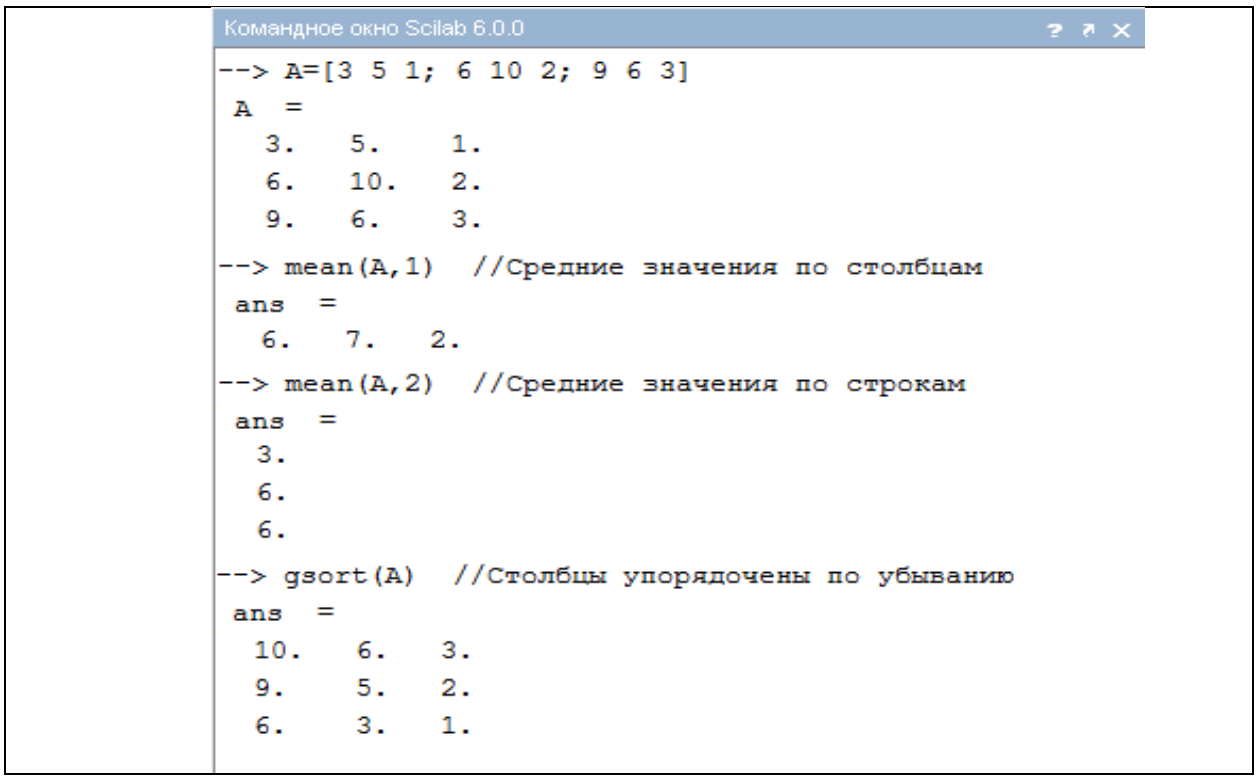

Рис.1.2.1-10. Определение средних значений и упорядочение элементов

Известно, что если детерминант матрицы отличен от нуля, то это невырожденная матрица. Для такой матрицы может быть вычислена обратная матрица (A^{-1}), которая при умножении на исходную матрицу A, дает единичную (по диагонали расположены единицы, а прочие элементы равны нулю). Для получения обратной матрицы используется функция **inv().** Умножение матриц в Scilab производится только с использованием их имен. Описанные действия приведены нарис.1.2.1-11.

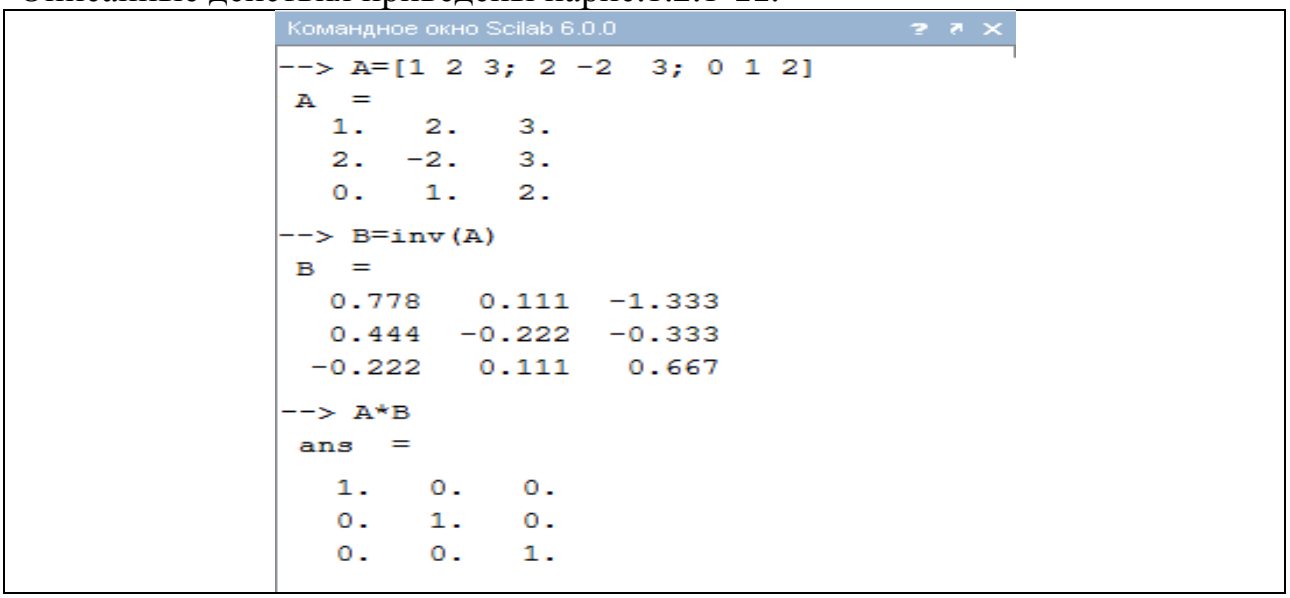

Рис. 1.2.1-11. Получение обратной и единичной матриц

Рассмотрим еще один пример, в котором матрица умножается на скаляр, матрица делится на скаляр, и матрица умножается на вектор (рис.1.2.1-12).

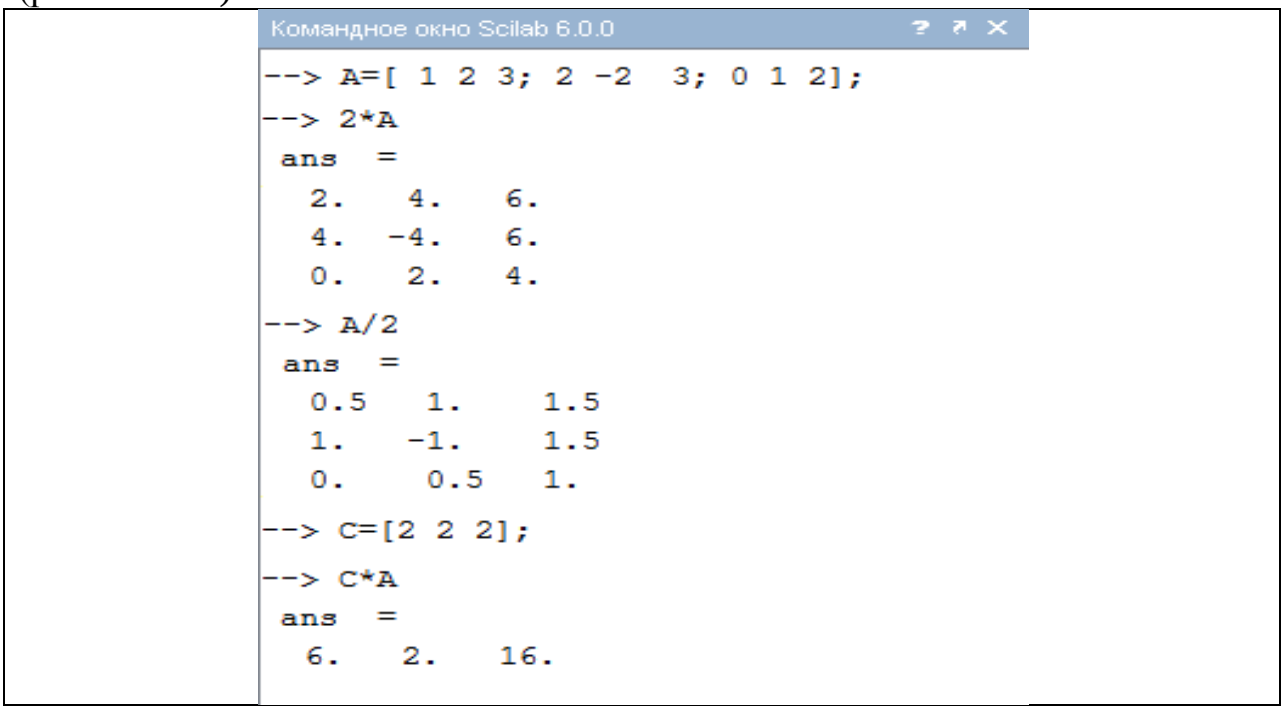

Рис. 1.2.1-12. Действия над матрицами

Вектора и матрицы кроме традиционного их применения для хранения и обработки данных необходимы и для построения графиков функций. При этом вектора используются для построения плоских графиков (графиков функций от одной переменной), а матрицы – для построения трехмерных изображений (графиков функций от двух переменных).

Обратите внимание, что после ввода или формирования элементов векторов и матриц могут возникнуть ошибки. Для контроля и исправления отдельных элементов векторов и матриц можно воспользоваться окном редактора данных (рис. 1.2.1-13).

Окно редактора переменных состоит из панели инструментов и области просмотра значений переменных. В окне редактора переменных (рис. 1.2.1-13) можно отображать несколько переменных и редактировать значения используемых переменных. Переключение между переменными реализуется с помощью вкладок.

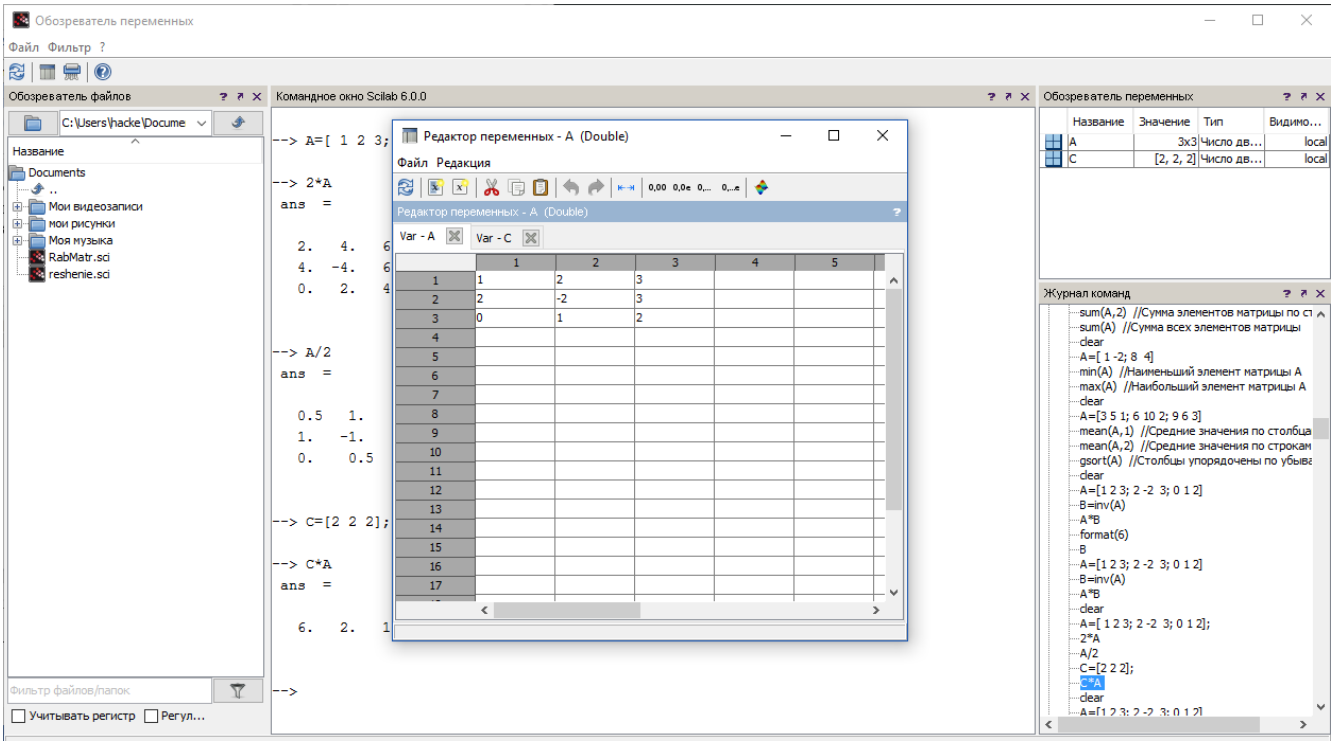

Рис. 1.2.1-13. Окно **Рабочей среды** с активной инструментальной панелью **Редактором переменных**

1.2.2. Построение графиков и визуализация вычислений в системе Scilab

Самый простой способ для построения графика функции одной переменной **y=f(x)**, это предварительное формирование двух векторов одинаковой длины: вектор значений аргументов **x** и вектор соответствующих им значений функции **y**, а затем выполнение команды **plot(x, y)**. Выполнение команды **plot(x, y)**открывает графическое окно и отображает в нем график функции **y(x)**.

Рассмотрим пример построения графика функции $y = e^{-x^2}$ на отрезке [-1.5;1.5] (рис.1.2.2-1).

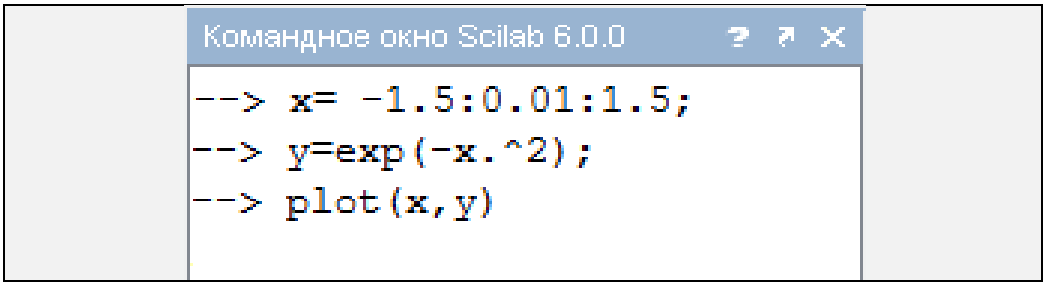

Рис. 1.2.2-1. Команды построения графика функции $y = e^{-x^2}$

В результате выполнения команды **plot(x,y)** появляется графическое окно с номером (рис. 1.2.2-2).

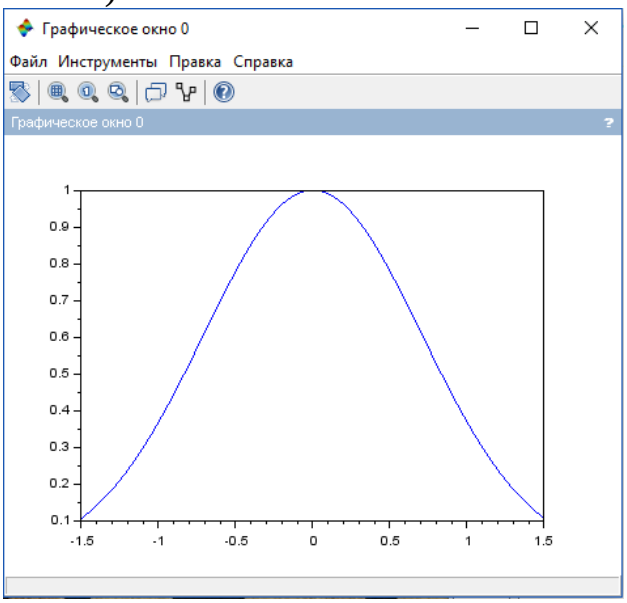

Рис. 1.2.2-2. Графическое окно с изображением графика функции $y = e^{-x^2}$

Переход между окнами (от командного окна к графическому окну и обратно) осуществляется с помощью комбинации клавиш<Alt+Tab> или с помощью мыши.

В общем случае, число аргументов у функции **plot()** не ограничивается двумя. Эта функция имеет следующий формат:

plot(x1,y1,x2,y2,...).

Таким образом, в одном графическом окне можно построить не один, а несколько графиков. Рассмотрим два способа построения в одном графическом окне 3-х графиков (рис.1.2.2-3):

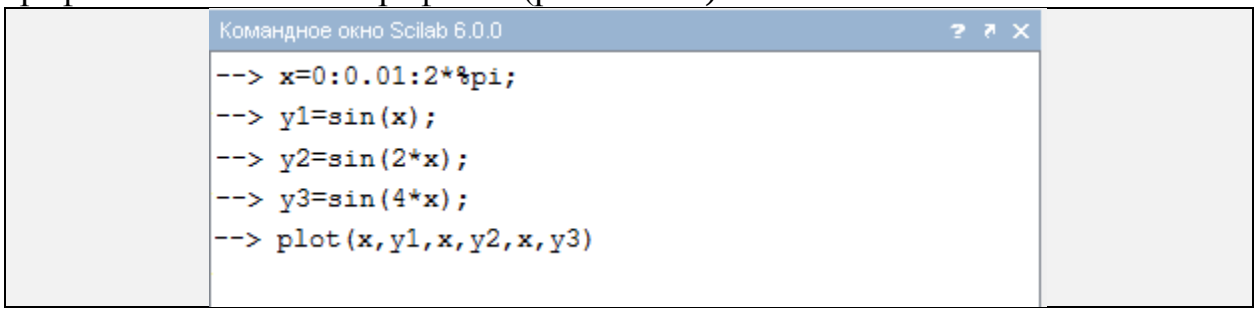

Рис. 1.2.2-3. Использование векторов значений функций для построения графиков 3-х функций

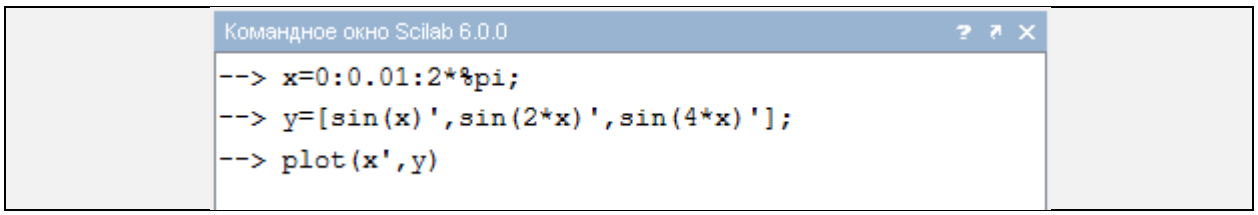

Рис. 1.2.2-4. Использование матрицы значений функций для построения графиков 3-х функций

Приведенные в первом и втором примерах наборы команд позволяют получить один и тот же результат (рис. 1.2.2-5.) Разница в том, что в первом примере формируется три вектора значений функций (**y1, y2, y3**), а во втором – матрица **y**, содержащая значения функций в виде столбцов.

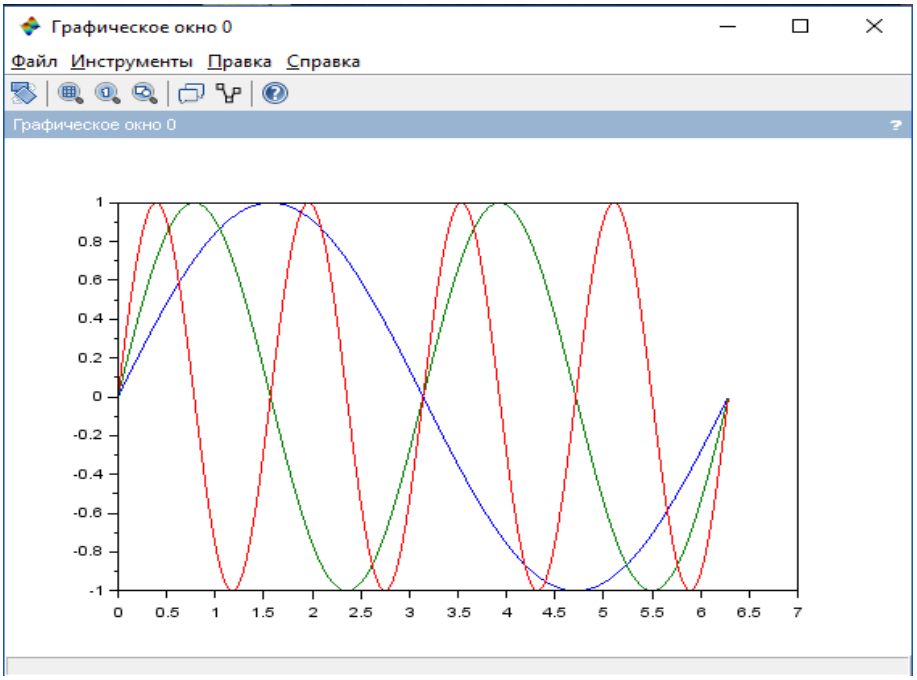

Рис1.2.2-5. Графики функций**y1=sin(x); y2=sin(2*x);y3=sin(4*x)**

График, выведенный в графическое окно Scilab, может быть снабжен ***заголовком***, ***именами осей***, дополнительным ***текстом, сеткой*** и другой поясняющей информацией. Аргументами команд, управляющими пояснениями, являются текстовые строки. Например, команда **xtitle('title', 'xstr', 'ystr')** добавит к графику заголовок и подписи осей. Команда **xgrid()** позволяет отобразить координатную сетку.

В случаях, когда в одной координатной плоскости изображаются графики нескольких функций, как в нашем примере, возникает необходимость в выводе обозначений («легенде»). Имена созданных кривых можно вывести с помощью команды **legend**:

legend(leg1, leg2, ..., legn, [pos]),

где: **leg1** — имя первого графика;
leg2 — имя второго графика;
legn — имя n-го графика;
pos—числовой параметр, определяющий место расположения обозначений в графическом окне (табл. 1.2.2-1).

Таблица 1.2.2-1

Значение аргумента	Размещение легенды
-1	В правом верхнем углу над областью графика
0	Место выбирается автоматически, чтобы не перекрывать область кривых
1	В правом верхнем углу (и по умолчанию)
2	В левом верхнем углу области графика
3	В левом нижнем углу области графика
4	В правом нижнем углу области графика

При выводе в одно графическое окно нескольких графиков удобнее каждый график выводить с использованием своей команды **plot(),** однако в этом случае происходит создание нового графического окна. Для того, чтобы этого избежать используется команда **mtlb_hold('on')**. Эта команда позволяет расположить все в дальнейшем выводимые графики в одном окне.

При выводе графика можно сменить принятый по умолчанию Scilab цвет и тип точек, с помощью которых рисуется данный график (табл.1.2.2-2). Символы, указывающие на цвет и тип точки, заключаются в апострофы и указываются в команде **plot()** после имени функции.

Таблица 1.2.2-2

Символ	Цвет	Символ	Маркер
y	Желтый		точка
m	фиолетовый	**.**	:
c	голубой	**x**	x-метка
r	красный	**+**	плюс
g	зеленый	*****	звездочка
b	синий	**s**	квадрат
w	белый	**d**	алмаз
k	черный	**v**	Треугольник(вниз)
€	треугольник (вверх)	**-**	сплошная
<	треугольник (влево)		точечная
>	треугольник (вправо)	**-.**	Штрих - пунктирная
p	шестиугольник	**-**	пунктирная
h	восьмиугольник		

Рассмотрим пример (рис. 1.2.2-6), в котором используются перечисленные выше опции, инструкции и функции. Результат выполнения инструкций, комментирующих графики, приведен на рис.1.2. 2-7.

```
Командное окно Scilab 6.0.0
--> x=0:0.01:2*%pi;
--> y1=sin(x);
--> y2=sin(2*x);
--> y3=sin(4*x);
--> plot(x,y1,'-k') //Сплошная кривая черного цвета
--> xgrid()  // Включаем сетку
--> mtlb_hold('on')
--> plot(x,y2,'b--') // Штриховая кривая синего цвета
--> plot(x,y3,'r.-') // Штрих-пунктирная кривая красного цвета
--> xtitle('Построение графиков трех функций','X','Y')
--> legend('y1(x)','y2(x)','y3(x)',-1)
```

Рис. 1.2.2-6. Использование инструкций при построении графиков

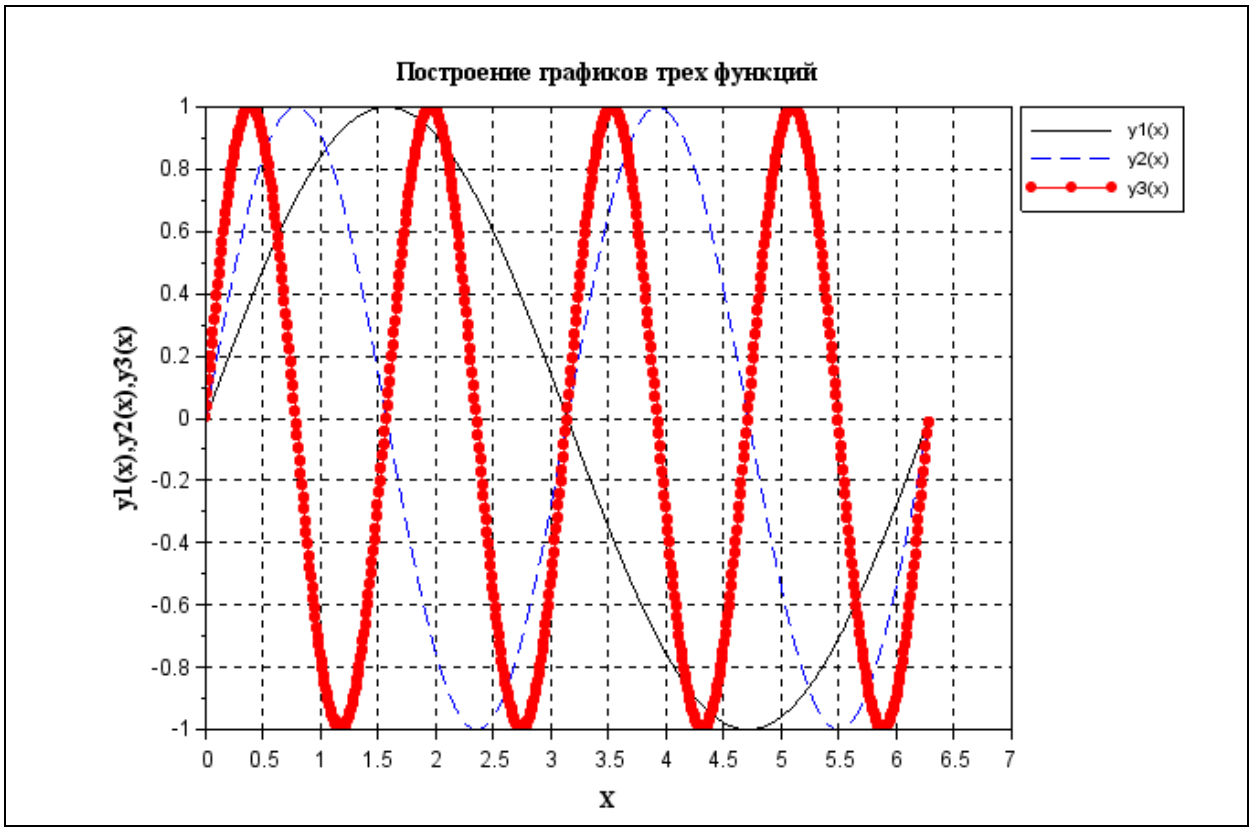

Рис. 1.2.2-7. Результат выполнения инструкций, комментирующих графики

Большую часть вышеописанных действий можно реализовать также с помощью команд инструментальной панели и кнопок графических окон (рис.1.2.1-8).

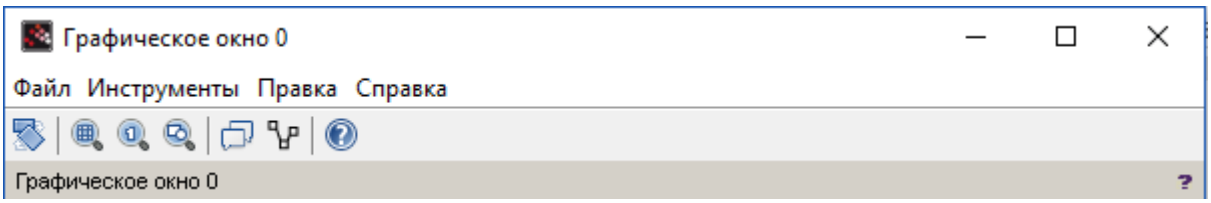

Рис.1.2.2-8. Компоненты меню графического окна

Команды основного меню графического окна представляют большой набор средств, предназначенных для отображения и оформления графиков, позволяющих в интерактивном режиме и без использования команд придать графику желаемый вид, а кнопки панели дублируют наиболее часто используемые пункты меню, ускоряя тем самым доступ к ним.

Трехмерные поверхности описываются функцией двух переменных **z(x,y).** Для построения трехмерных графиков необходимо сформировать два двумерных массива, например, **X** и **Y** с использованием функции **meshgrid()** (рис.1.2.2-9).

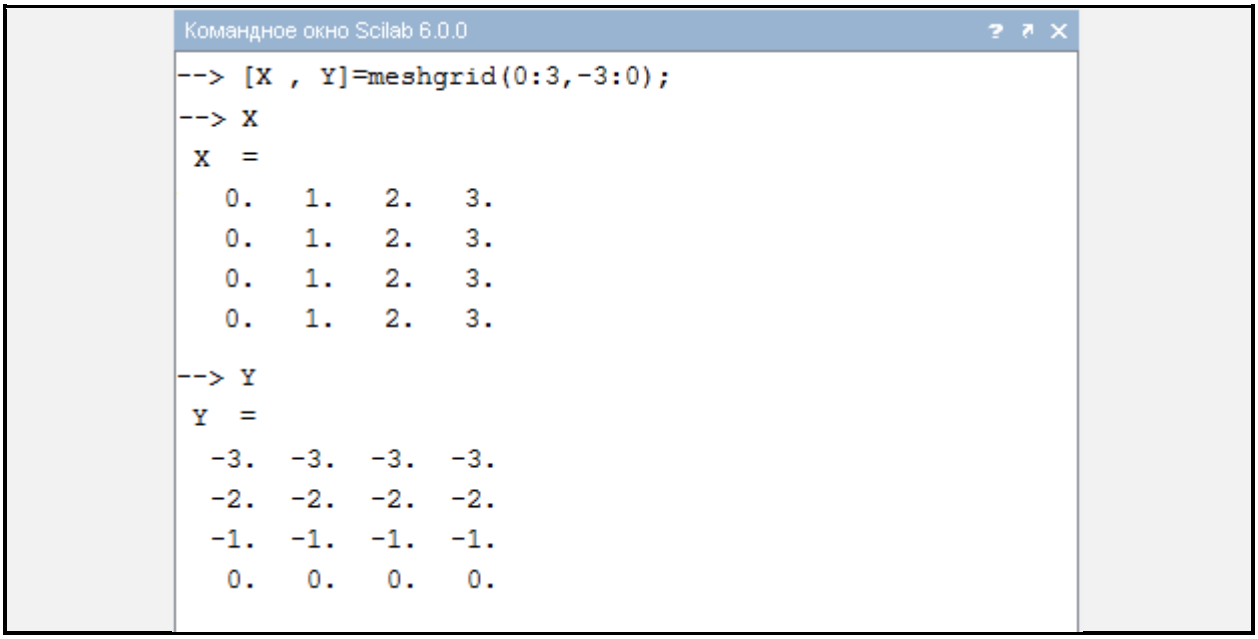

Рис.1.2.2-9. Формирование двумерных массивов функцией **meshgrid()**

Сформированные в виде двумерных массивов данные используются функциями:

- **mesh(X,Y,Z)** – построение сетчатого графика;
- **contour(X,Y,Z)** – построение графика контурных линий;
- **surf(X,Y,Z)** – построения графика сплошной поверхности;
- **plot3d(X,Y,Z) –** построение точек, соединенных отрезками прямых и др.

Рассмотрим примеры использования перечисленных выше функций, для чего сформируем матрицу **z(x,y)** с использованием функции **f(x,y)** (Рис. 1.2.2-10).

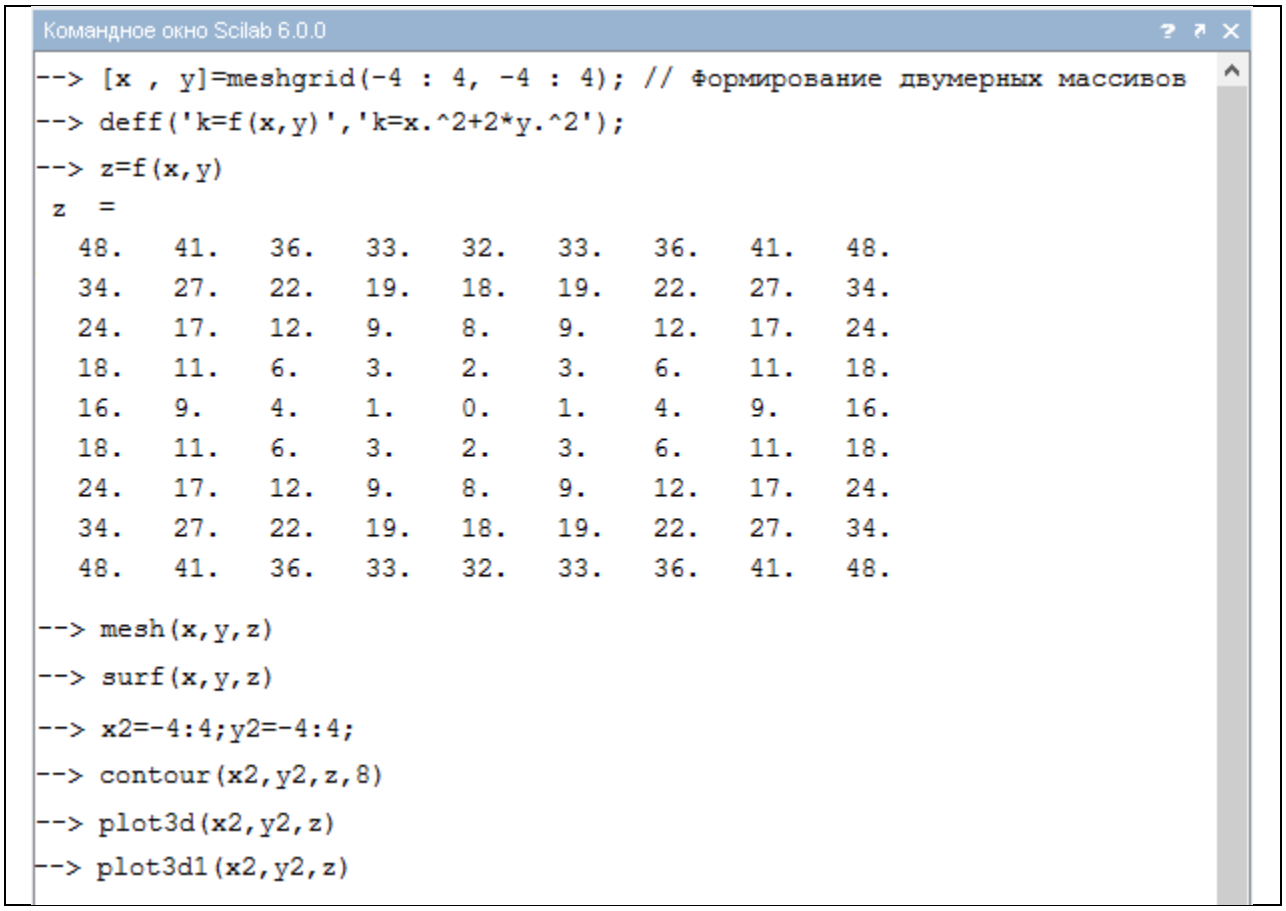

Рис. 1.2.2-10. Построение различных видов графиков функций двух переменных

Результатом выполнения команды **mesh(x,y,z)** является построение графика поверхности в виде сетки (рис.1.2.2-11).

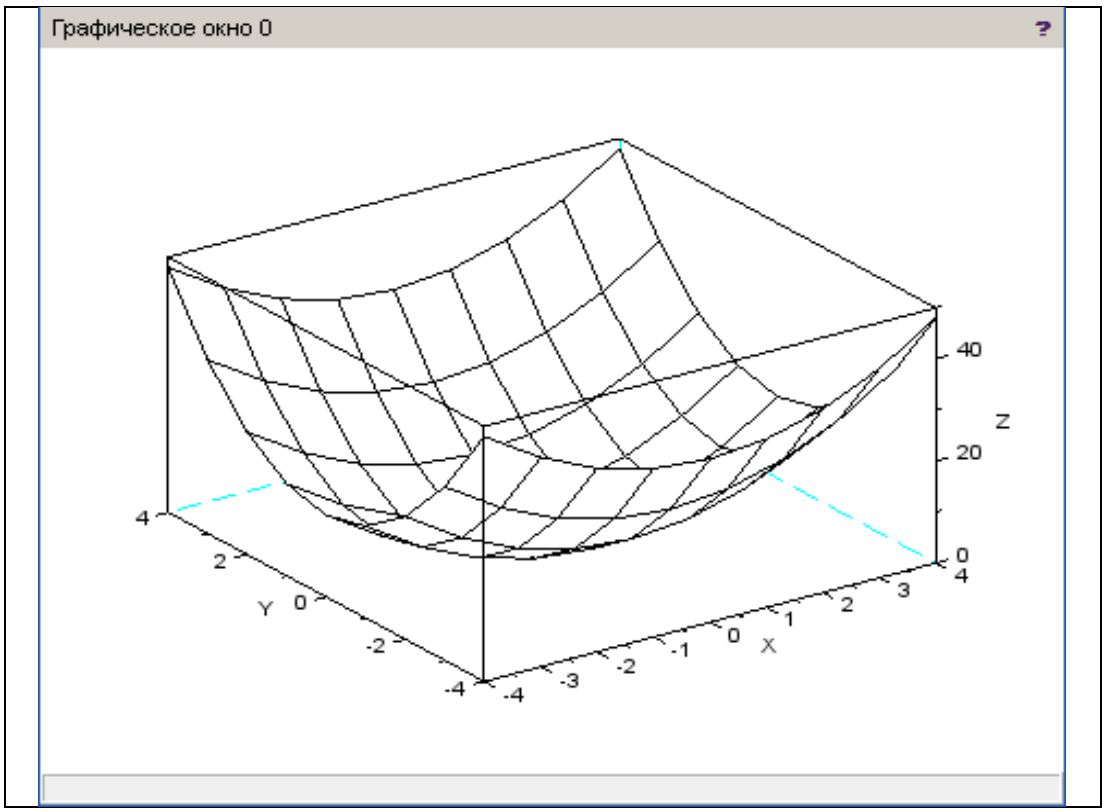

Рис. 1.2.2-11. Результат выполнения команды **mesh(x,y,z)**

В результате выполнения команды **plot3d(x2,y2,z)** происходит построение графика поверхности, где точка соединены отрезками прямой (рис. 1.2.2-12).

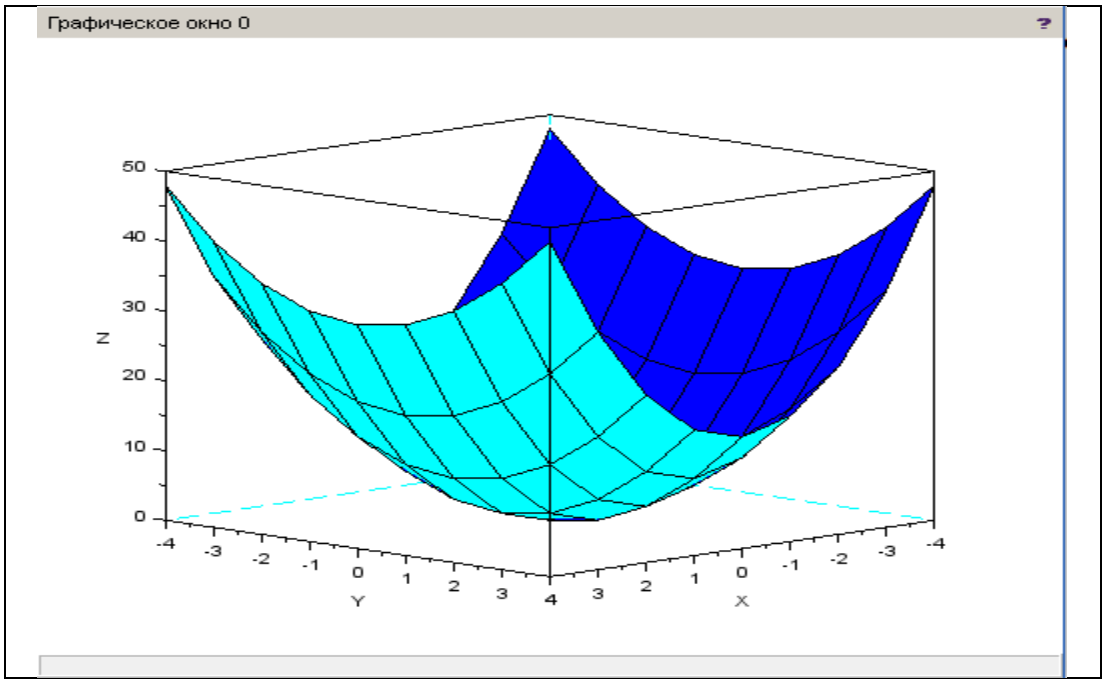

Рис. 1.2.2-12. Результат выполнения команды **plot3d(x2,y2,z)**

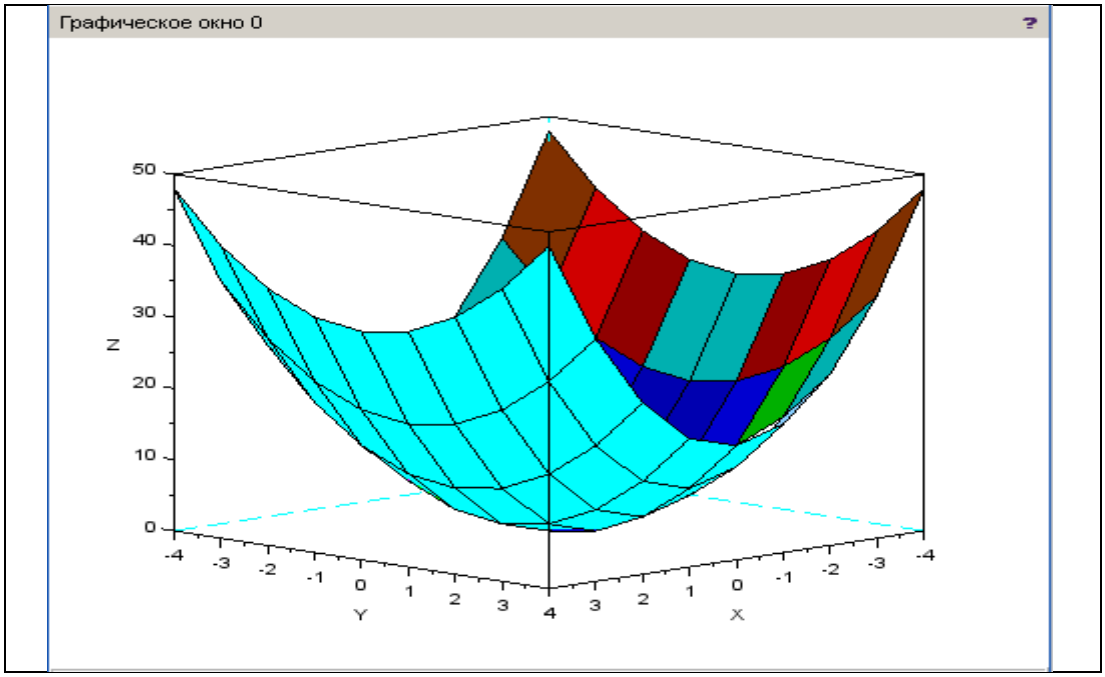

Рис. 1.2.2-13. Результат выполнения команды **plot3d1(x2,y2,z)**

Команда **surf(x,y,z)** выполняет построение графика сплошной поверхности (рис. 1.2.2-14).

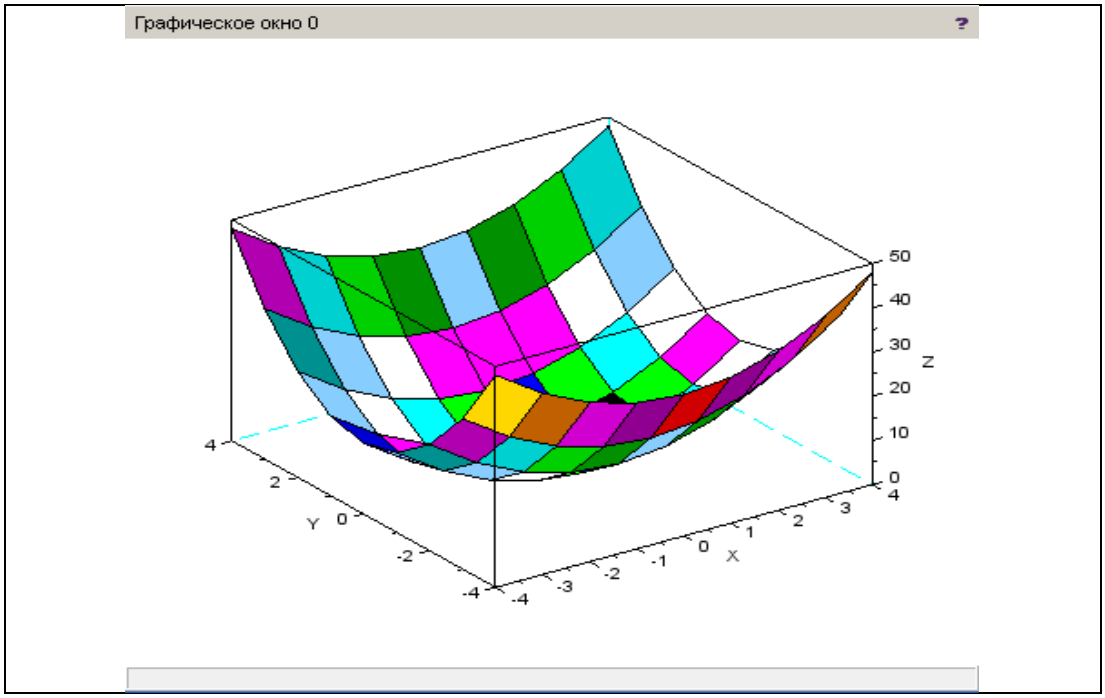

Рис. 1.2.2-14. Результат выполнения команды **surf(x,y,z)**

Команда **contour(x,y,z)** позволяет получить график контурных линий, а добавление команд

>>x2=-4:4; y2=-4:4;

>>contour(x2,y2,z);

позволяет нанести на контурные линии значения функции в отдельных точках (рис. 1.2.2-15).

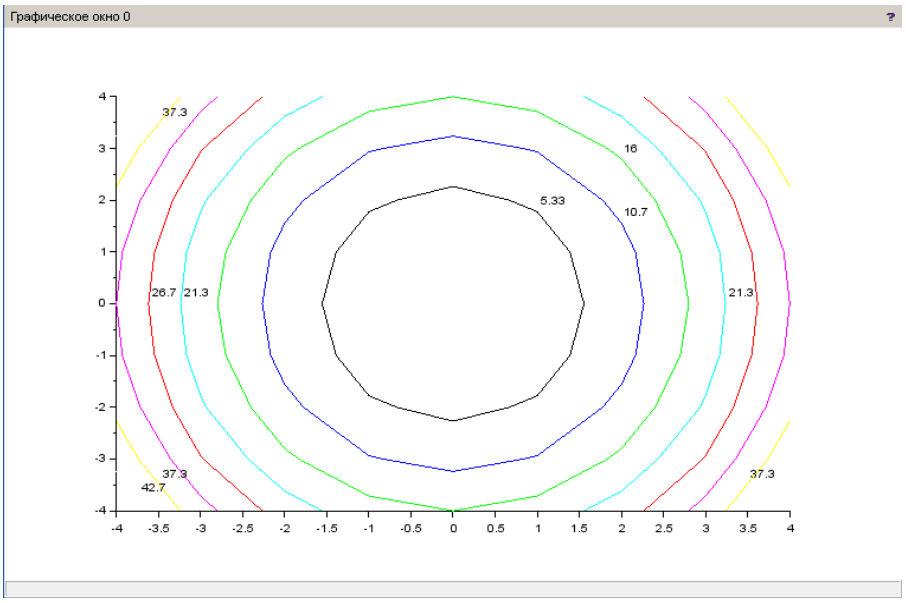

Рис. 1.2.2-15. Результат выполнения команды **contour(x2,y2,z)**

1.2.3. Лабораторная работа по теме «Вектора, матрицы и построение графиков в системе Scilab»

1. Вопросы, подлежащие изучению

1) Работа с векторами и матрицами.
2) Построение графиков функций одной переменной**.**
3) Средства инструментальной панели графических окон.
4) Построение трехмерных изображений с использованием функций Scilab **mesh( ), plot3( ), surf( )** и **controur( ).**

2. Общеезадание

1) ***Изучите*** *материал* ***Темы 1.2 (п.п. 1.2.1 – 1.2.2)****.*
2) ***Выберете вариант*** *индивидуального задания из* ***табл. 1.3.3.-1****.*
3) ***Выполните команду clear all*** *для очистки Рабочей области.*
4) ***Опишите функцию f1(x)*** *и получите ее символьное выражение.*
5) ***Задайте диапазон*** *изменения аргумента функции* ***f1(x)*** *и вычислить ее значения.*
6) ***Задайте диапазон*** *изменения аргумента функции* ***f1(x)*** *для построения графика.*
7) ***Выполните команду plot( )****для получения графика* ***f1(x).***
8) ***Опишите функцию f2(x).***
9) ***Разместите графики*** *функций* ***f1(x)*** *и* ***f2(x)*** *в одном окне, для чего после построения графика первой функции выполнить команду* **mtlb_hold('on').**
10) ***Дополните графики*** *необходимыми пояснениями: заголовок, имена осей, координатная сетка и легенда.*
11) ***Задайте диапазоны*** *изменения значений* ***x*** *и* ***y*** *для функций* ***f3(x,y)*** *и получить таблицы их значений.*
12) ***Опишите функцию f3(x, y).***
13) ***Получите таблицу*** *значений функции* ***f3(x,y).***
14) ***Получите график функции f3(x, y)*** *с использованием команд* ***mesh( ), plot3( ), surf( )*** *и* ***controur( ).***
15) ***Сохраните текст*** *рабочего окна на внешнем носителе*
16) ***Предоставьте результаты*** *работы преподавателю, ответьте на поставленные вопросы.*
17) ***Выполните команду clear all*** *для очистки Рабочей среды.*
18) ***Оформите отчет*** *по выполненной работе****.***

1. Варианты индивидуальных заданий

Таблица 1.3.3-1

1	$f1(x) = Sin(1-0.2x^2) - x$ $f2(x) = e^{-x} - 2$ $f3(x,y) = x^2 + y^2 - 8\sin(x-y)\cdot\cos(x+y)$
2	$f(x) = x - Sin(1/x)$ $f2(x) = e^{x} + \ln(x) - x$ $f3(x,y) = x^2 - 2y^2 - \sin^2(x+y) - 0.5e^{x\cdot\sin(y)}$
3	$f1(x) = 1 - x + Sinx - \ln(1+x)$ $f2(x) = (1-x)^{1/2} - Cos(1-x)$ $f3(x,y) = x^3 - 2y^2 - \sin(x+y)\cdot\cos(x-y) - x + 2y$
4	$f(x) = Sinx^2 + Cosx^2 - 10x$ $f2(x) = x^2 - \ln(1+x) - 3$ $f3(x,y) = x^3 - 2y^2 - \sin(x+y)\cdot\cos(x-y) - x + 2y$
5	$f(x) = Cos(x/2)\cdot\ln(x-1)$ $f2(x) = Cos(x/5)\cdot(1+x)^{1/2} - x$ $f3(x,y) = x^3 - 2y^3 - 8\sin(x+y)\cdot\cos(x-y) - 5x$
6	$f1(x) = 3x - e^{-x}$ $f2(x) = 4(1+x^{1/2})\ln(x) - 10$ $f3(x,y) = 3x^3 - y^2 - \cos(x-y) - x + 2x - 9xy$
7	$f1(x) = Sinx - 3^{1/2}Cosx + 4x - 4$ $f2(x) = x - 1/(3 + Sin(3.6x))$ $f3(x,y) = x^2 - 2y^2 - \sin^2(x+y) - 0.5e^{x\cdot\sin(y)}$
8	$f1(x) = 0{,}25x^3 + Cos(x/4)$ $f2(x) = 2 - x - \ln(x)$ $f3(x,y) = 3x^3 - y^2 - \cos(x-y) - x + 2y - 9xy - e^{x\cdot\cos(y)}$
9	$f1(x) = x^2 + 4Sin(x)$ $f2(x) = tg(0{,}36x + 0{,}4) - x^2$ $f3(x,y) = 3x^3 - y^2 - \cos(x-y) - x + 2y - 9xy - e^{x\cdot\cos(y)}$
10	$f1(x) = 1 + \lg(x) - 0{,}5$ $f2(x) = 2\lg(x) - x/2$ $f3(x,y) = 3x^3 + y^2 - \cos(x-y) - x + 2y - 9xy$
11	$f1(x) = X - \sin(x) - 0{,}25$ $f2(x) = \lg(0{,}4x + 0{,}4) - x^2$ $f3(x,y) = x^3 + y^2 - x\cdot\sin^2(x+y) + 2y - 9xy$

12	$f1(x) = \sqrt{x} - Cos(0{,}387x)$ $f2(x) = \lg(x) - 7/(2x+6)$ $f3(x,y) = x^3 + y^3 \sin^2(x+y) + 2y - 9xy$
13	$f1(x) = tg(0{,}5x + 0{,}2) - x^2$ $f2(x) = 3x - Cos(x) - 1$ $f3(x,y) = x + y^3 \sin^2(x+y) + 2y - 9x$
14	$f1(x) = x - \lg x - 0{,}5$ $f2(x) = 1{,}8x^2 - Sin(10x)$ $f3(x,y) = x + y^3 \sin^2(x+y) + 2y - 9x^2$
15	$f1(x) = ctg(1{,}05x) - x^2$ $f2(x) = x \cdot \lg x - 1.2$ $f3(x,y) = 5x^2 + y^3 + 2y - 9x - 0.5 \cdot e^y$
16	$f1(x) = \sqrt{\lg(x+2)}$ $f2(x) = Sin0.5x + 2 - (x/2)^2$ $f3(x,y) = 5x^2 - y^2 + 2y - 9xe^{\sin(x)}$
17	$f1(x) = 0{,}5x + \lg(x-1) - 2$ $f2(x) = Sin(0{,}5 + x) - 2x + 0{,}5$ $f3(x,y) = 5x^2 - 7y^2 \cdot \cos(x+y) + 2y - 9xe^{\sin(x)}$
18	$f1(x) = \ln(x/6) + \sqrt{x}$ $f2(x) = \log_2(x) - 1/(x+2)$ $f3(x,y) = -15x^2 - 7y^2 \cdot \cos(x+y) + 2x - 9x + 6$
19	$f1(x) = \lg(2+x) + x^2 - 3$ $f2(x) = \ln(1+2x) - 2 + x$ $f3(x,y) = 15x^2 + 7y^2 \cdot \cos(x+y) + 2y - 9x\sqrt{\lvert y\rvert} + 6$
20	$f1(x) = e^{-x} - 2 + x^2$ $f2(x) = 2e^x - x^2 + 2$ $f3(x,y) = -3x^2 + y \cdot \cos(x+y) + 2y^4 - 9x \cdot \sqrt{\lvert 3x^2 - y\rvert}$
21	$f1(x) = 2 \cdot x^2 - e^{x/2}$ $f2(x) = 2 \cdot arqtg(x) - 3x + 2$ $f3(x,y) = -5x^2 + y^3 + 2x^3y - x^3 \cdot \sqrt[3]{\lvert 3x - y\rvert}$
22	$f1(x) = Sin(x - 0{,}5) - x - 0{,}8$ $f2(x) = (x-3)^2 \lg(x-2) + 2$ $f3(x,y) = -5x^2 - y^3 + 2x^3y - x^3 \cdot \sin(3x - y)$
23	$f1(x) = x - 3 + Cosx + x^2$ $f2(x) = (x-1)\lg(x+11) - 1$ $f3(x,y) = -5x^2 - y^3 + 2x^3y - x^3 \cdot \sin(3x + y)$

24	$f1(x) = e^{2x}Cos(2x) + x$ $f2(x) = x^2Cos(2x) + 1$ $f3(x, y) = -5x^2 + y^3 + 2x \cdot y - 3 \cdot \sin(3x - y^3) + 8y \cdot \cos(x)$
25	$f1(x) = (2 - x)2^x - 1$ $f2(x) = (x - 2)^2 - 1 - 2^x$ $f3(x, y) = x^2 + y^3 + 2x \cdot y - 3 \cdot \sin(3x - y^3) + 5y \cdot \cos(x) \cdot e^x$
26	$f1(x) = e^x + x + 1$ $f2(x) = 0{,}5^x - 3 - (x + 2)^2$ $f3(x, y) = -5x^2 + 8y^2 + 2x \cdot y - 10 \cdot \sin(3x - y^3) + 5y \cdot x$
27	$f1(x) = (x - 2)^2 \lg(x + 5) - 1$ $f2(x) = (x - 1)^2 \lg(x - 3) - 1$ $f3(x, y) = -5x^2 + 8y^2 + 2x^2 \cdot y^2 - 10 \cdot \sin(x - y^3) + 5x^3 \cdot \cos(x + y)^3$
28	$f1(x) = 2x^2 - 2^x - 20$ $f2(x) = x \log_3(x + 1) - 1$ $f3(x, y) = -5x^3 - y^3 + 2x^2 \cdot y - 7(x \cdot y)^2 \sin(3x)$
29	$f1(x) = 0{,}5^x - 3 + (x + 1)^2$ $f2(x) = 2arctg(x) - x + 3$ $f3(x, y) = e^x(5x^2 - 8y^2) + 2x^2 \cdot y^2 - 10 \cdot \sin(x) \cdot \cos(10y)$
30	$f1(x) = 5^x - 6x - 3$ $f2(x) = 2Cos(x) + x^2 - 3x + 2$ $f3(x, y) = 5x^2 - y^3 + 2x^3 \cdot y^2 - x \cdot \sin(3x - y) + y^2 \cdot \cos(x)$

3. Содержание отчета

1) В форме комментариев:
   - Название лабораторной работы
   - ФИО студента, номер группы
   - № варианта
   - Индивидуальное задание
2) Протокол вычислений (сессии) в ***Командном окне,*** снабженный соответствующими комментариями.

1.2.4. Контрольные вопросы по теме

1) Как создать вектор-строку?
2) Как создать вектор -столбец?
3) Как транспонировать вектора?
4) Какая функция служит для определения длины вектора?

5) Каким образом создать вектор с постоянным шагом?
6) Требуется ли при работе с векторами и матрицами предварительное объявление их размера?
7) Какой символ используются для разделения элементов матрицы в строке, а какой для разделения ее строк?
8) Какие команды предназначены для заполнения матрицы случайными числами, распределенными по равномерному или нормальному закону распределения?
9) Формат команд выбора минимального и максимального значения элемента матрицы.
10) Назначение команды ***plot*().**
11) Каким образом построить в одном графическом окне несколько графиков?
12) Какой пояснительной информацией может быть снабжен график, построенный в графическом окне?
13) Для чего используется функция ***legend()?***
14) Каково назначение функции ***mtlb_hold()?***
15) Каково назначение функции ***meshgrid()*** при построении трехмерных изображений?
16) Какие типы графиков позволяют строить встроенные функции: ***plot()***, ***contour(), surf()*** *и* ***plot3()***?

Тема 1.3. Средствами Scilab для создания и описания sci-файлов

1.3.1. Основные понятия и средства программированиям в Scilab
1.3.2. Описание и работа с sci-сценариями
1.3.3. Описание и работа с sci-функциями
1.3.4. Основные операторы sci-языка и программирование в Scilab
1.3.5. Примеры решения задач средствами Scilab
1.3.6. Лабораторная работа по теме
1.3.7. Контрольные вопросы по теме

1.3.1. Основные понятия и средства программированиям в MatLab

Использование системы Scilab только в режиме непосредственного расчета (в командном режиме) явно недостаточно для решения серьезных задач, поскольку, во-первых, зачастую требует выполнение сложных алгоритмических процессов, а, во-вторых, необходим механизм хранения в библиотеках тексты команд и операторов системы Scilab. То есть необходима средства, какие есть в языках программирования высокого уровня.

Такие средства в Scilab существует. Они состоят из так называемых **sci**-файлов и средств, их создания и отладки – **Редактора программного кода. Sci**-файлы представляют собой текстовые файлы, которые могут хранится в файлах («библиотеках») Scilab с расширением **sci.**

Если вспомнить технологию процедурного программирования, то **sci**-файлы фактически являются процедурами системы Scilab. Эти **sci**-файлы могут состоят из следующих **элементов (**средств языка программирования Scilab**)**:

- данные различного типа;
- константы и переменные, в том числе системных;
- операции;
- системные команды и функции;
- функции пользователя;
- оператор присваивания и управляющие операторы;
- системные операторы и функции;
- средства работы с файлами данных;
- средства расширения языка.

Программирование инженерных задач в среде Scilab очень напоминает программирование на универсальных языках программирования. Однако поскольку в Scilab можно использовать не только выражения над структурированными данным (например, массивами) но и системные

команды и функции, то система программирования Scilab является еще и мощной программной системой.

При создании новых **sci**-файлов Редактор можно открыть путем активизации инструмента ***SciNotes*** (рис. 1.1.1-1), а для редактирования существующих инструментом **Открыть** или двойным щелчком мышки по имени существующего **sci**-файла. (1.3.1-1).

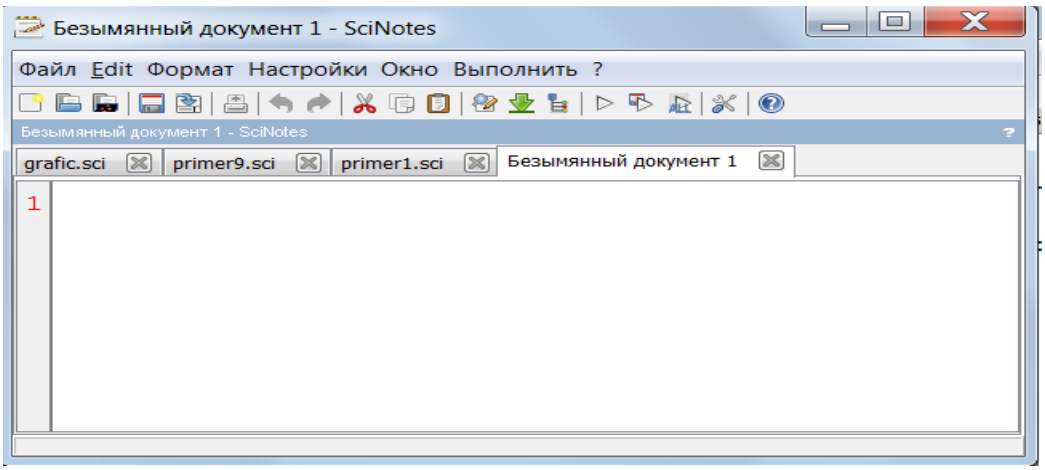

Рис. 1.3.1-1. Рабочая среда Редактора для создания sci-файла

При активной вкладке ***SciNotes*** инструменты инструментальной панели позволяют открывать, сохранять, редактировать, запускать и осуществлять отладку **sci**-файлов.

Эти инструменты разбиты на следующие категории:

- ***Файл*** – категория, включающая инструменты, которые позволяют создавать новые наборы команд и программы, и сохранять их в файлах; открывать существующие наборы команд и программ и загружать их из файлов; создавать различные объекты Scilab, осуществлять поиск файлов различных типов т.д.
- ***Edit***–категория, включающая инструменты, которые позволяю тработать с текстом **sci**-файла в окне редактора: вставлять в текст **sci**-файлов различные элементы; превратить в комментарий текущую строку или вернуть ее к исходному виду; уменьшить или увеличить отступы текущей строки или нескольких выделенных строк на заданное число позиций влево или вправо; выполнить интеллектуальный отступ.
- ***Формат*** – категория, включающая инструменты, которые позволяют осуществить форматирование документа: сделать или удалить отступ, удалить пробелы в конце строки, сменить регистр, добавить или удалить комментарий, заменить одинарные кавычки на двойные.
- ***Настройка*** – категория, включающая инструменты по текущей кодировки файла, настройке шрифтов, цвета и др.

- ***Окно*** **-** категория, включающая инструменты копирования (или отделения) вкладки в новое окно, включение или отключение разделителя на две колонки и др.
- ***Выполнить*** **–** категория, включающая инструменты, которые позволяют выполнить sci-файл и сохранить в текущий каталог, а также выполнять выделенную часть файла.

В SciLab существует два типа **sci**-файлов: ***файл-сценарии*** и ***sci-функции***.

Файл-сценарий представляет собой последовательность команд и функции Scilab (**без входных и выходных параметров**), которые оперируют данными из **Рабочей области**, причем результаты выполнения ***сценария*** доступны **Рабочей области** и могут быть использованы для дальнейших вычислений.

Sci-функции – это функции Scilab, аналогичные функциям языков программирования высокого уровня, таких как C++,C# и VB, которые могут иметь как входные, так и выходные параметры, а также локальные переменные.

1.3.2. Описание и работа с файлами-сценариями

Сценарий является простейшей реализацией **sci**-файлов. Он может содержать последовательность команд, операторов, функций и комментарии.

Файлы-сценарии имеют свои особенности:
- не имеют входных и выходных параметров;
- работают с данными из рабочей области;
- в процессе выполнения не компилируются;
- строки автоматически нумеруются;
- представляют собой зафиксированную в виде файла последовательность команд, операторов и функций полностью аналогичную той, что используется во время сессии в **Командном окне**.

Откроем окно ***SciNotes*** для создания **файла-сценария** и введем в нем несколько команд, например, необходимых для построения графика (рис. 1.3.2-1.).

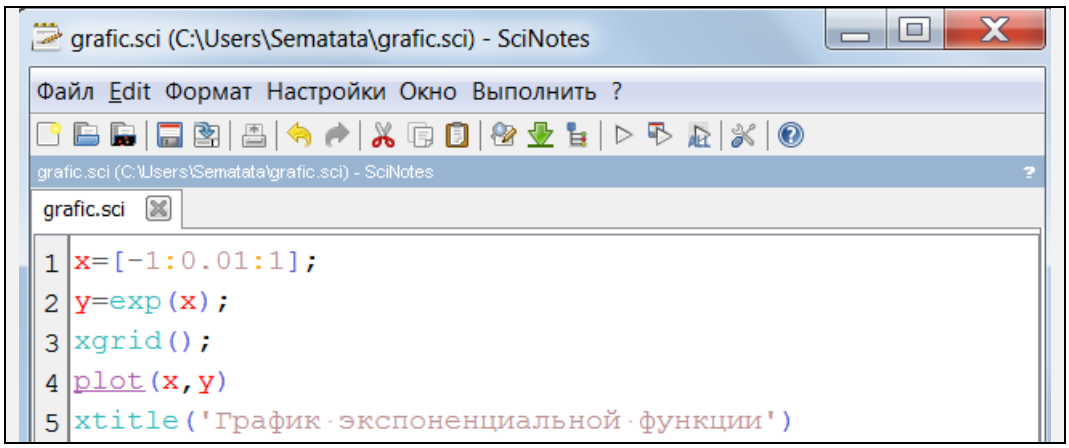

Рис. 1.3.2-1.**Файл-сценарий**, содержащий команды для построения графика

Для сохранения созданного файла следует щелкнуть по кнопку инструмента **Сохранить** в соответствующей строке окна ввести имя **sci**-файла, а затем щелкнуть по кнопке **Save.** Имя файла появится в окне Текущего каталога с расширением**.sci.**

Запуск файла (рис.1.3.2-3), сохраненного, например, с именем grafic**.**sci, можно произвести из строки Командного окна, скопировав его содержимое в текущую командную строку и нажать<Enter> **(**то есть выполнить, как команду **Scilab**). Зададим предварительно в командном окне переменным **x** и **z** числовые значения, а следом, подключив соответствующий файл-сценарий выполним его. Запуск файла позволил в следующей строке командного окна получить результат выполнения (рис.1.3.2-4).

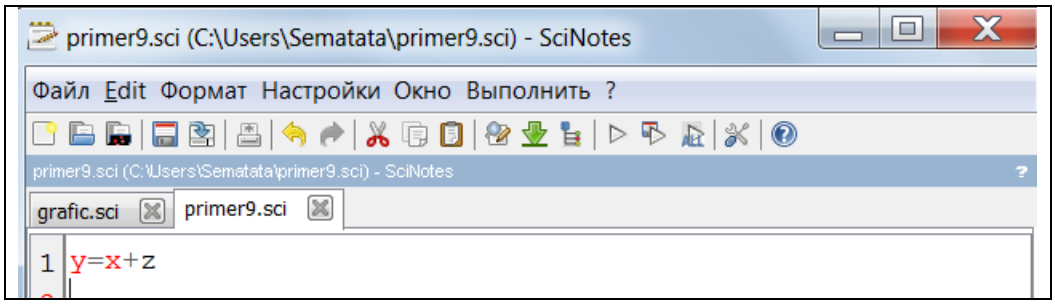

Рис.1.3.2-3. **Sci**-сценарий **primer9.m**

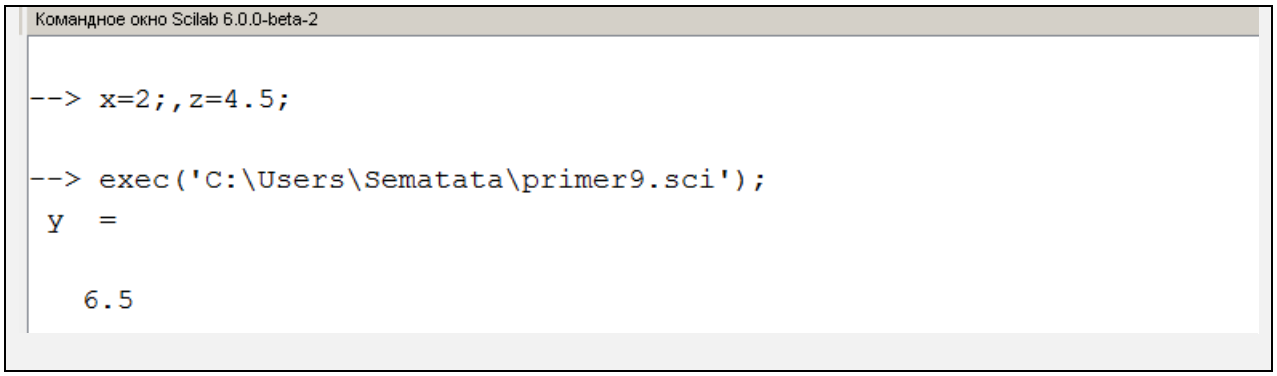

Рис. 1.3.2-4. Выполнение **sci**-сценария с именем **primer9**

Рассмотрим еще один пример работы (**primer1)**, который предназначен для построения графика (рис. 1.3.2-5).

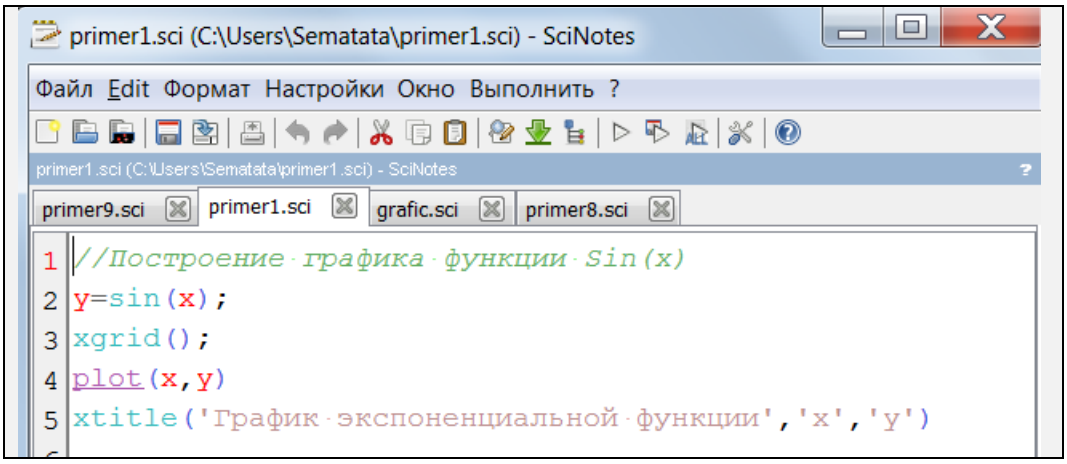

Рис.1.3.2-5. **Sci**-сценарий с именем **primer1**

Чтобы запустить этот файл на выполнение следует предварительно задать последовательность значений переменной **x**, которая используются в теле файла (помня, что **sci**-сценарий работает с данными из рабочей области) и набрать в командной строке имя (рис.1.3.2-6).

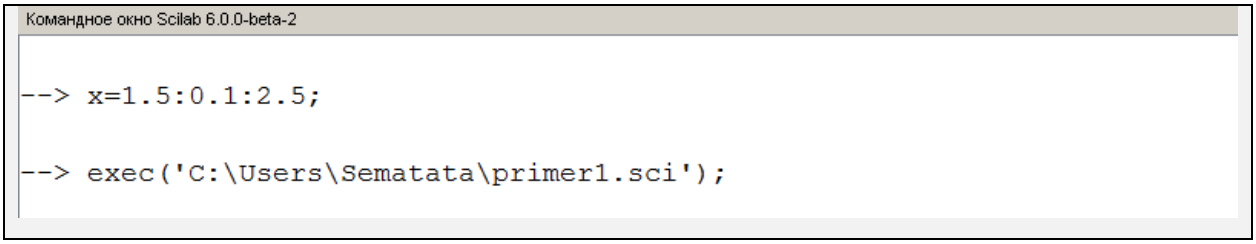

Рис.1.3.2-6. Выполнение **script**-файла с именем **primer1**

Этот пример еще раз подтверждает, что все переменные, используемые в **sci**-сценарии, являются **глобальными**, т. е. они действуют одинаково в командах сессии и внутри программного блока, которым является **sci**-сценарий. Результат выполнения команд сценария с именем **primer1** приведен на рис. 1.3.2-7.

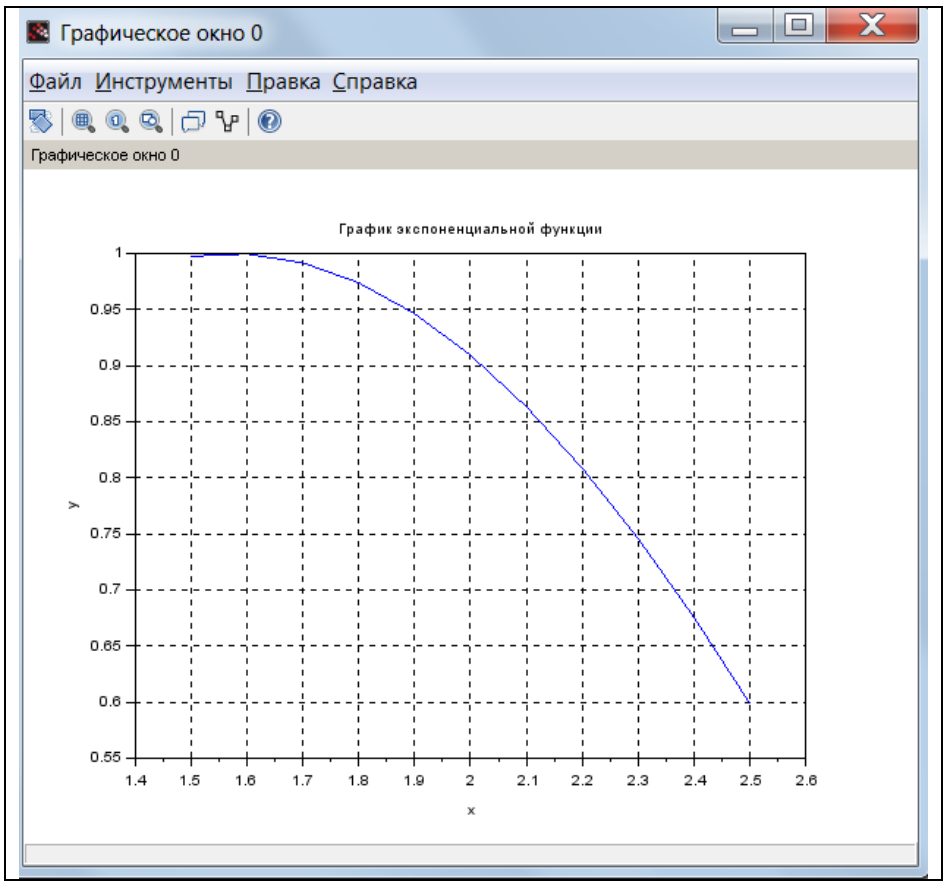

Рис. 1.3.2-7. Результат работы **sci**-сценария **primer1**

1.3.3. Описание и работа с sci-функциями

Sci-функции, также как и **sci**-сценарии, содержат команды, операторы и функции, но являются более сложным типом **sci**-файлов по сравнению со сценарием и имеют свои особенности:

- начинаются с заголовка описания **sci**-функции**;**
- могут иметь входные и выходные параметры;
- все переменные, описанные в теле **sci**-функции, являются **локальными**, т.е. действуют только в пределах тела функции;
- являются самостоятельными программными единицам, которые общаются с другими модулями посредством имени с входными и выходными параметрами.

Sci-функция является типичным объектом языка программирования высокого уровня.

При создании новой **sci**-функции открывается окно редактора со следующим шаблоном (рис. 1.3.3-1).

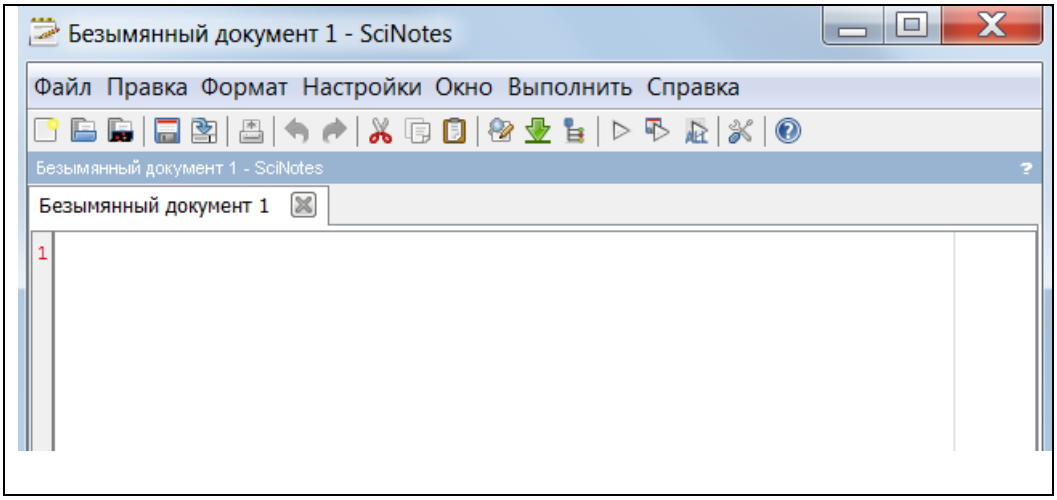

Рис.1.3.3-1. Структура новой **sci-функции**

Общая структура **sci**-функции с **n** входными и **m** выходными параметрами имеет вид:

function[*var1, …, var**m**,…***]** *=f_name (список входных параметров)*
// Основной комментарий
// Тело m-функции
var1 = выражение
…
*var**m** = выражение*
end

Начинаются **sci**-функции с заголовка **function**, затем в квадратных скобках через запятую указываются имена выходных параметров, далее ***f_name*** – имя функции, а затем в круглых скобках - список входных параметров функции. Имена функций должны быть уникальными.

Sci-функция возвращает свое значение (или значения) и может быть вызвана из выражений, расположенных в Рабочей области или в других программных модулях:

*f_name***(***список_параметров***)**

По умолчанию все переменные, описанные в теле **sci**-функции, являются **локальными**, т.е. определены только в пределах функции, в которых они описаны. Между собой **sci**-функции общаются посредством своего имени и входных и выходных параметров. Конструкция

vari=выражение

приведенная в общей структуре **sci**-функции, используется, если требуется, чтобы функция возвращала результаты вычислений.

Поскольку **sci**-функции может иметь не один, а несколько выходных параметров, то она во многом напоминает процедуру. Поэтому ее нельзя использовать непосредственно в математических выражениях. Если функция, имеющая несколько выходных параметров, используется как функция, имеющая единственный выходной параметр, то для возврата значения будет использоваться первый из них. Это зачастую ведет к ошибкам в математических вычислениях. Поэтому обращение к **sci**-функции с несколькими выходными параметрами должно иметь вид:

[*var1,var2,...* **]** = *f_name(Список_параметров)*

Создадим простейшую **sci**-функцию с двумя входными и одним выходным аргументами (рис. 1.3.3- 2).

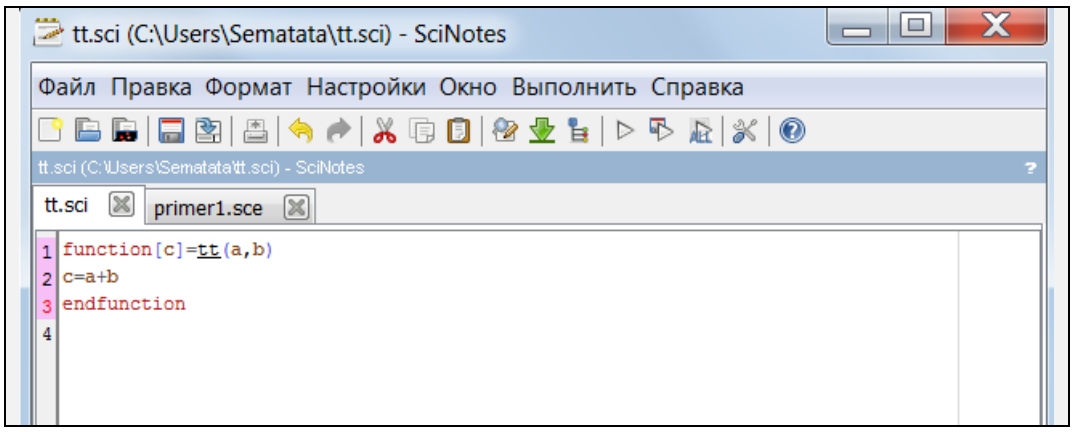

Рис. 1.3.3- 2. Описание **sci**-функции с одним выходным параметром

Сохраним функцию в файле, отметив, что **Scilab** предлагает в качестве имени **sci**-функции название самой функции, т.е. **tt.sci**. Для вызова функции в Командном окне следует выполнить функцию **exec** (указать «Пусть к файлу») (рис.1.3.3-3). В Командном окне появится код функции **tt.sci**. Теперь можно выполнять обращение к функции, показанное также на рис.1.3.3-3.

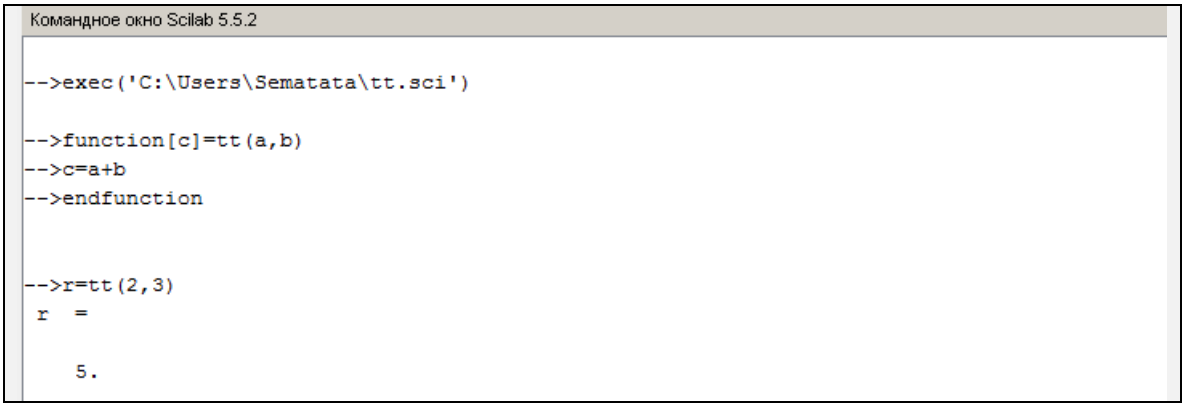

Рис. 1.3.3-3. Обращение к sci-функции **tt(a,b)**

Следует отметить, что если в конце первой командной строки не поставлена точка с запятой (;), то в окно выводится текст функции, точка с запятой в конце строки подавляет вывод текста функции.

При вызове **sci**-функции **tt(a, b)** входные аргументы **a** и **b** получили соответственно значения 2 и 3, сумма **a** и **b** записана в выходной параметр **c**, значение выходного аргумента **c** присвоено переменной r,а результат вывелся в следующую строку командного окна.

Следующий пример показывает создание **sci**-функции **с несколькими выходными параметрами**. Список выходных параметры в заголовке **sci**-функции заключен в квадратные скобки, а сами параметры отделены запятыми. В качестве примера создадим и выполним **sci**-функцию **quadeq(a,b,c)**,которая по заданным коэффициентам квадратного уравнения находит его корни (рис.1.3.3-4 , 1.3.3-5).

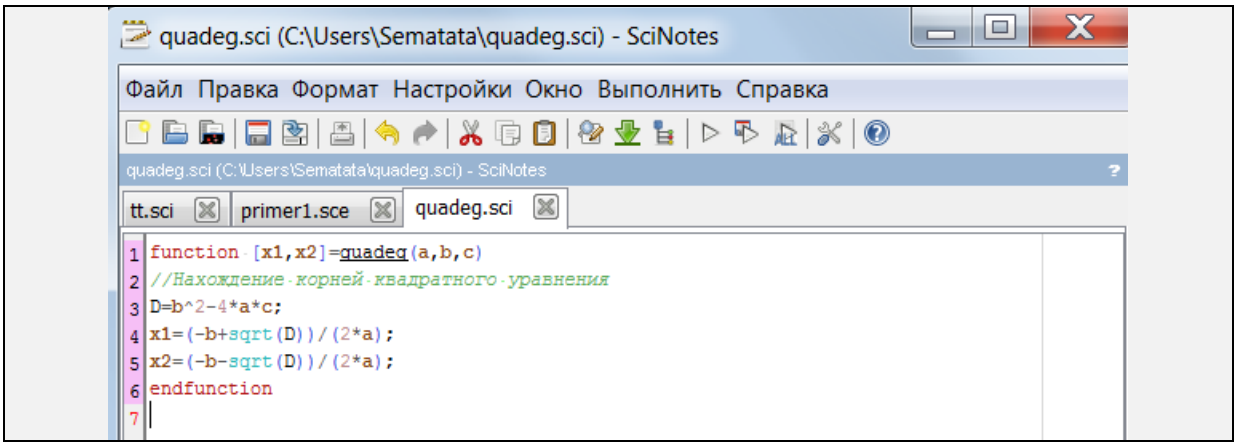

Рис. 1.3.3-4.Описание функция **quadeq(a, b, c)**
с двумя выходными параметрами

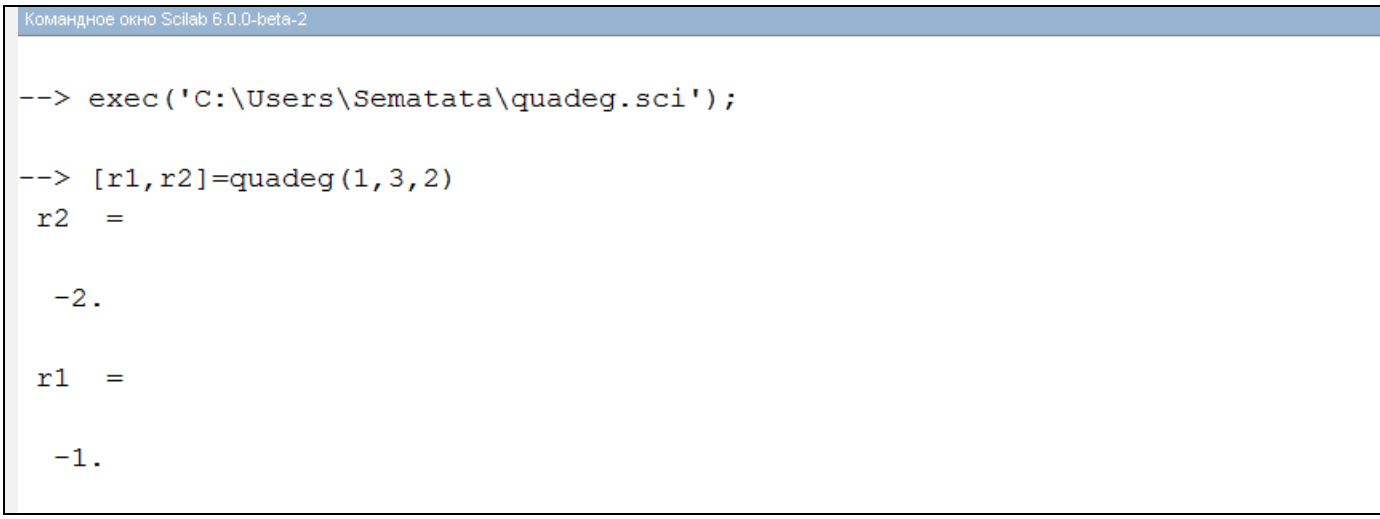

Рис. 1.3.3-5. Выполнение функции **quadeq(a, b, c)**
с двумя выходными параметрами

Иногда и при создании **sci**-функций желательно применение **глобальных переменных** (например, если параметров слишком много). В таких случаях используемые глобальные переменные надо объявить командой:

global *varl, var2,...*

Для того чтобы несколько функций могли совместно использовать глобальные переменные, они должны быть объявлены как **global** в каждом из модулей.

1.3.4. Алгоритмические операторы Scilab

Помимо программ с **линейной структурой**, инструкции которых исполняются строго по порядку, существует множество алгоритмов, структура которых **нелинейная**. При этом последовательность элементов алгоритмов могут выполняться в зависимости от определенных условий, иногда с конечным числом повторений – регулярных циклов, иногда в виде циклов, завершаемых при выполнении заданного условия. Практически любая серьезная программа имеет нелинейную структуру. Для создания таких программ необходимы специальные управляющие структуры. Они имеются в любом языке программирования высокого уровня, и в частности в **Scilab**. Рассмотрим операторы **sci**-файлов подробнее.

Оператор присваивания. Основным оператором системы программирования **Scilab** является **оператор присваивания**, имеющий следующую структуру:

ИмяПеременной = выражение

Оператор предназначен для идентификации переменных и обозначается символом =, слева от которого находится имя переменной, а справа арифметическое или строковое выражение (правила записи арифметических и строковых выражений были рассмотрены в п.1.1.2). Приведем несколько примеров операторов присваивания (рис.1.3.4-1).

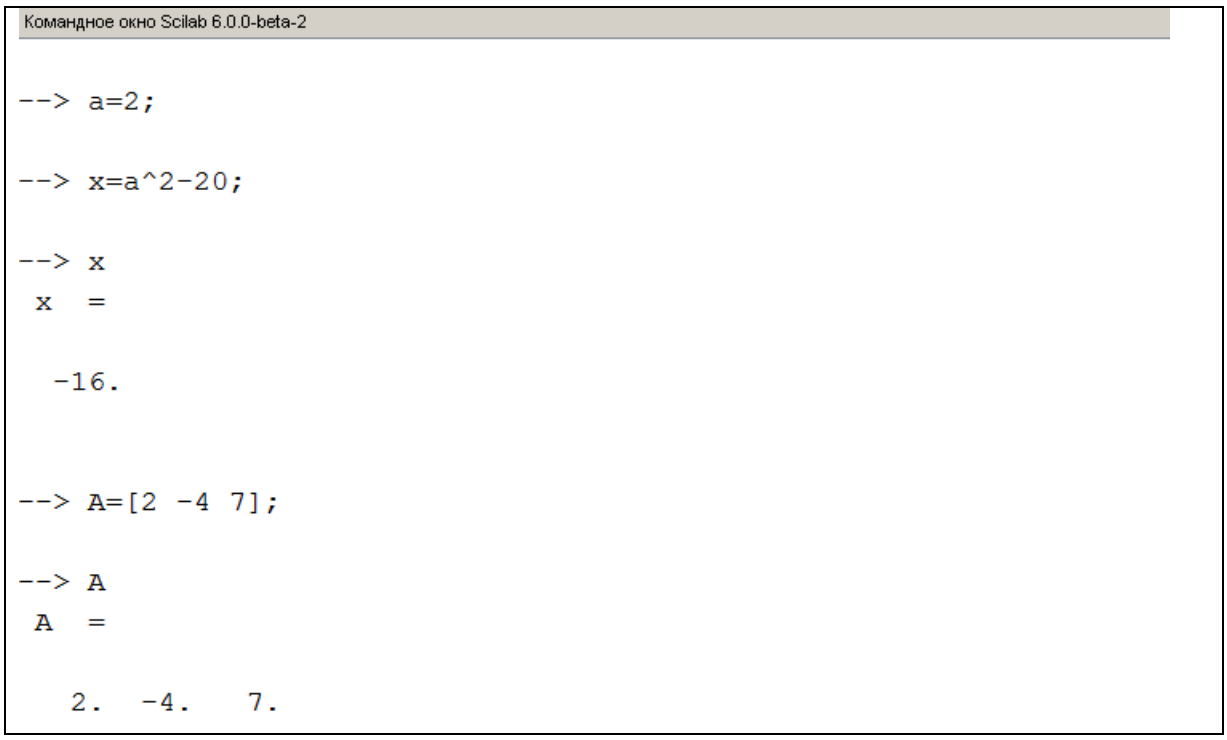

Рис. 1.3.4-1. Примеры операторов присваивания

Все переменные, используемые в правой части оператора присваивания должны быть предварительно определены. Если командная строка заканчивается символом точка с запятой (**;**), то результат выполнения оператора не выводится, иначе он выводится в следующей строке командного окна. Это замечание распространяется и на выполнение операторов присваивания, расположенных в **sci**-файлах.

Операторы ввода данных. Для организации простейшего ввода в Scilab можно воспользоваться функциями

x=input('title');
или
x=x_dialog('title', 'stroka');

Функция **input** выводит в командной строке Scilab подсказку **title**, и ждет пока пользователь введет значение, которое в качестве результата возвращается в переменную **x**. Функция **x_dialog** выводит на экран диалоговое окно с именем **title**. После чего пользователь может щелкнуть OK и тогда **stroka** вернется в качестве результата в переменную **x**, либо ввести новое значение вместо **stroka**, которое и вернется в качестве результата в переменную **x**. На рис.1.3.4-2 представлено диалоговое окно, которое формируется строкой **x=x_dialog('Input X','5')**.

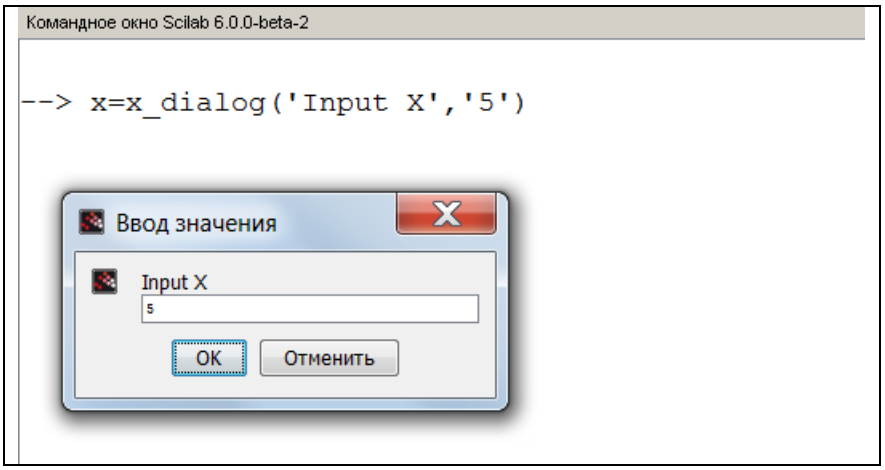

Рис. 1.3.4-2. Ввод данных с клавиатуры

Функция **input** преобразовывает введенное значение к числовому типу данных, а функция **x_dialog** возвращает строковое значение. Поэтому при использовании функции **x_dialog** для ввода числовых значений, возвращаемую ею строку следует преобразовать в число с помощью функции **evstr**. Поэтому можно предложить следующую форму использования функции **x_dialog**для ввода числовых значений

x=evstr(x_dialog('title', 'stroka'));

Для вывода в текстовом режиме можно использовать функцию disp следующей структуры

disp(b)

Здесь **b** -имя переменной или заключенный в кавычки текст .

Условный операторif…end.Условный оператор **if**в*общем виде* записывается следующим образом:

if *ЛогическоеВыражение1****then***
Инструкции1
elseif *Условие2*
ЛогическоеВыражение2
else
ЛогическоеВыражение3
end

Правил записи логических выражений описано в Теме1.1.

Эта конструкция допускает несколько частных вариантов. Простейшее – **усеченное разветвление**, имеет следующий вид:

if *ЛогическоеВыражение*
Инструкции
end

Напомним, что если *ЛогическоеВыражение* ***T*** (то есть «Истина»), то выполняются *Инструкции*, составляющие тело структуры ***if...end***. При этом оператор **end** указывает на конец перечня инструкций. Инструкции в списке разделяются запятая или точка с запятой. Если *ЛогическоеВыражение* ***F*** («Ложь»), то *Инструкции* не выполняются.

Ниже приведен пример использования простейшего усеченного разветвления, реализованного с использования оператора ***if*** (рис.1.3.4-3).

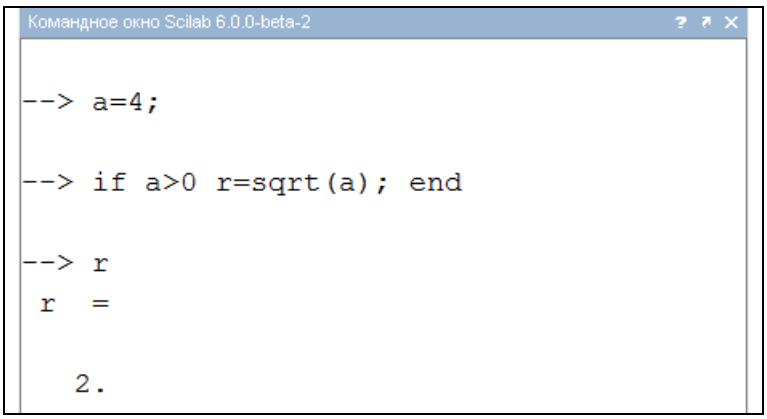

Рис. 1.3.4-3. Пример усеченного разветвления

Вторая частная конструкция напоминает **стандартное разветвление**:
if *ЛогическоеВыражение* ***then***
Инструкции1
else
Инструкции2
end

Здесь выполняются Инструкции1, если ЛогическоеВыражение T («Истина), а в противном случае выполняются Инструкции2.

В примере, приведенном на рис.1.3.4-4 рассматривается стандартное разветвление, реализованное с использованием оператора ***if***.

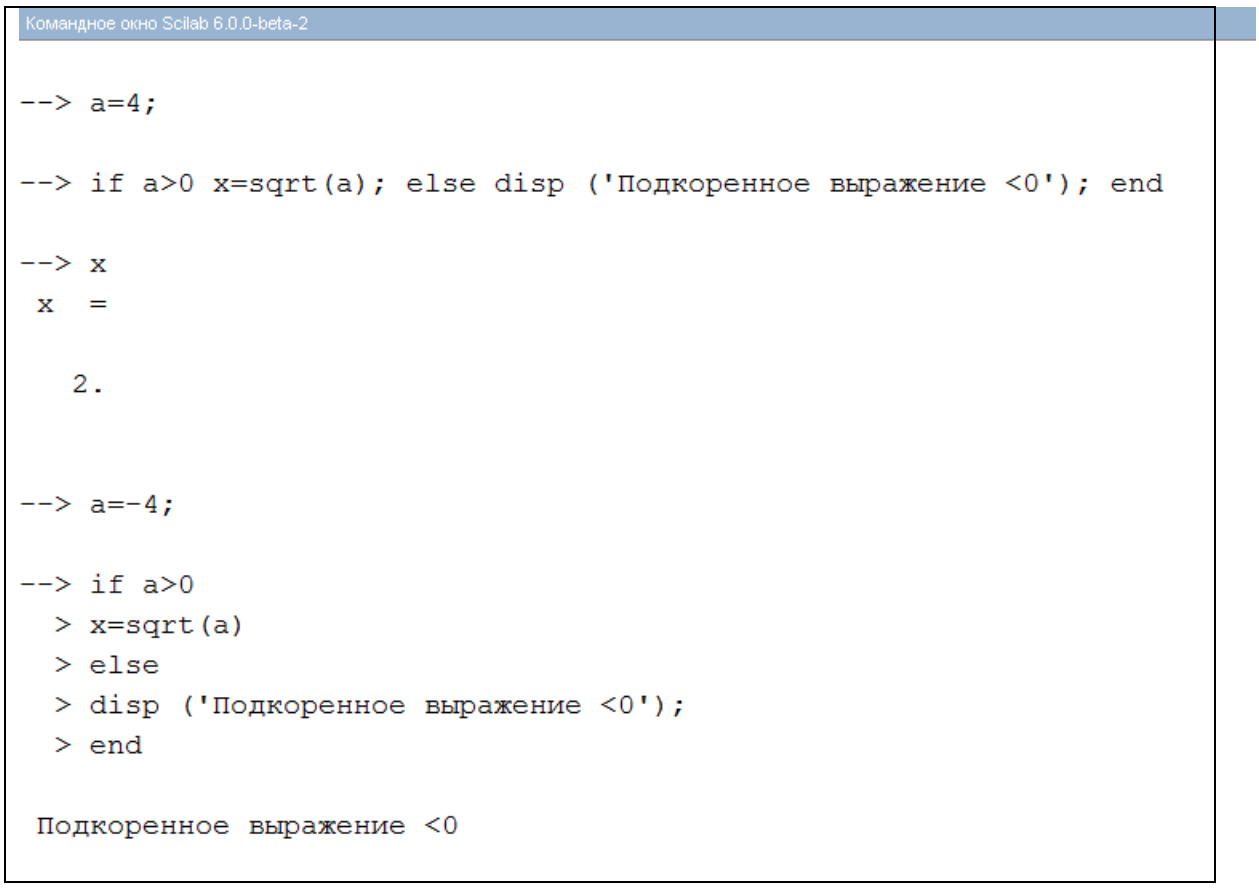

Рис. 1.3.4-4. Пример стандартного разветвления

Из приведенного примера видно, что оператор ***if*** может быть в одну строку, так и несколько строк.

На следующем примере рассмотрим более сложное - **вложенное разветвление**:

$$t = \begin{cases} \max\{x, y\}, \text{если } xy < 0; \\ \max\{x^2, \sin(y), \cos(x)\}; \text{если } xy > 2; \\ x / y; \quad \text{в противном случае.} \end{cases}$$

При его реализации, для того чтобы полностью отразить структуру сложного разветвления, используем **sci**-функцию (рис.1.3.4-6). Подберем данные для проверки основного разветвления и обратимся к функции **raz()** с различными исходными данными (рис.1.3.4-7).

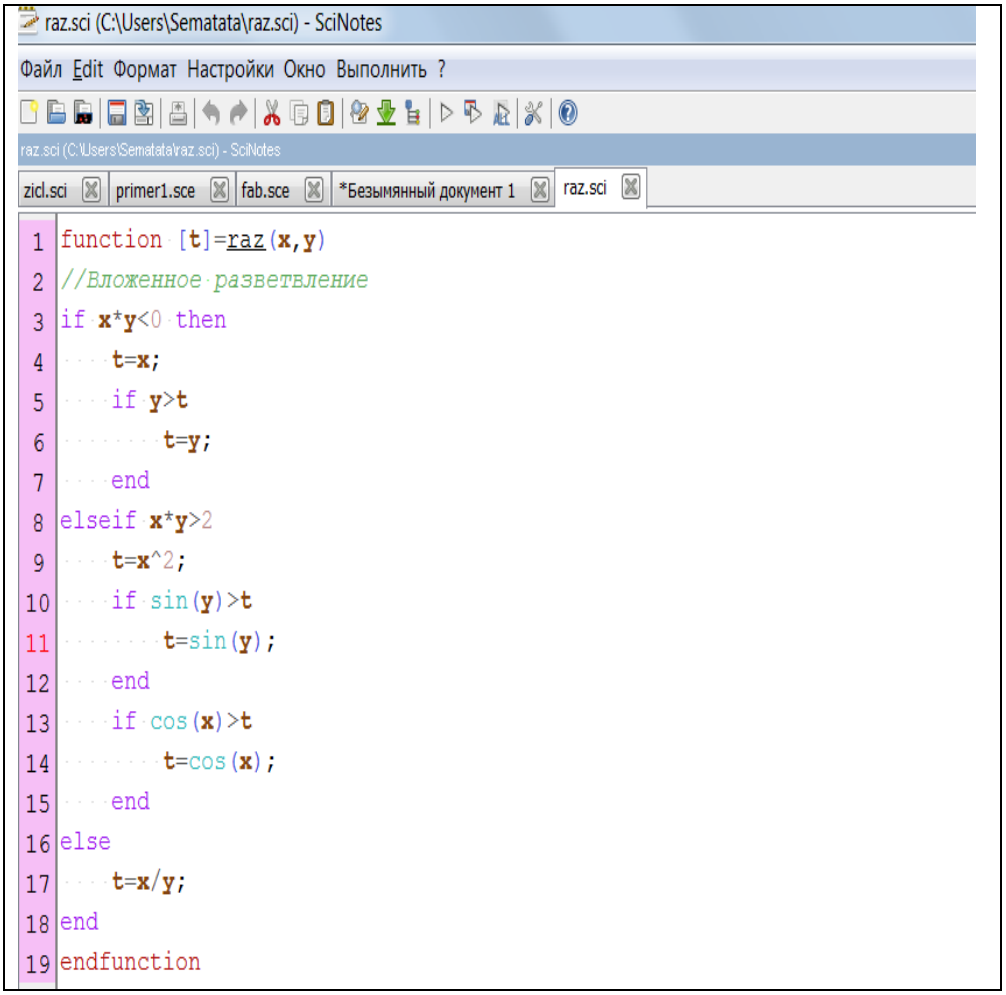

Рис. 1.3.4-6. Функция, реализующая вложенное разветвление

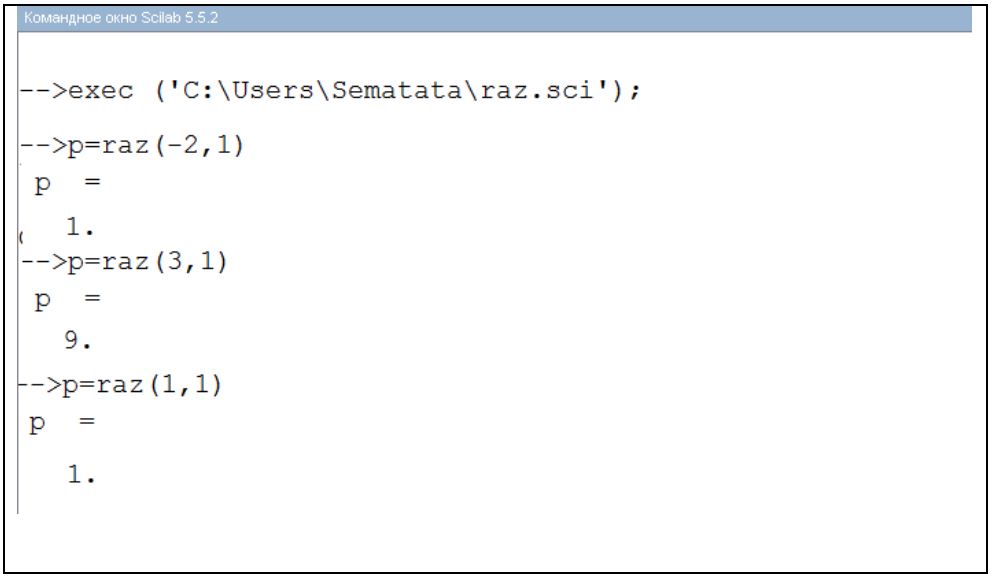

Рис. 1.3.4-7. Обращение к функции **raz()** с различными исходными данными

Оператор множественного выбора –switch. Для осуществления множественного выбора используется следующая конструкция **switch:**

switch *параметр*
case *Зачение_1 then операторы1*
case *Значение_2 then операторы2*
…
else *операторы*
end

Если выражение после заголовка ***switch*** имеет значение одного из выражений *Значение...,* то выполняется блок операторов ***case***, в противном случае — список инструкций после оператора ***else***. При выполнении блока ***case*** исполняются те списки инструкций, для которых *Значение* совпадает с *параметром*. Обратите внимание на то, что *Значение* может быть числом, константой, переменной, вектором ячеек или даже строчной переменной. Поясним использования оператора перебора ***switch*** следующим примером:

$$t=\begin{cases} y=x, если \quad n=1; \\ y=x(10-x), если \quad n=2; \\ y=xSin(nx), если \quad n=3,4,5; \\ y=1/(1+x^2), в противном случае. \end{cases}$$

Sci-функция, реализующая множественное разветвление, приведена на рис.1.3.4-8, а обращение к ней при исходных данных, позволяющих проверить каждую ветвь разветвления, показано на рис.1.3.4-9.

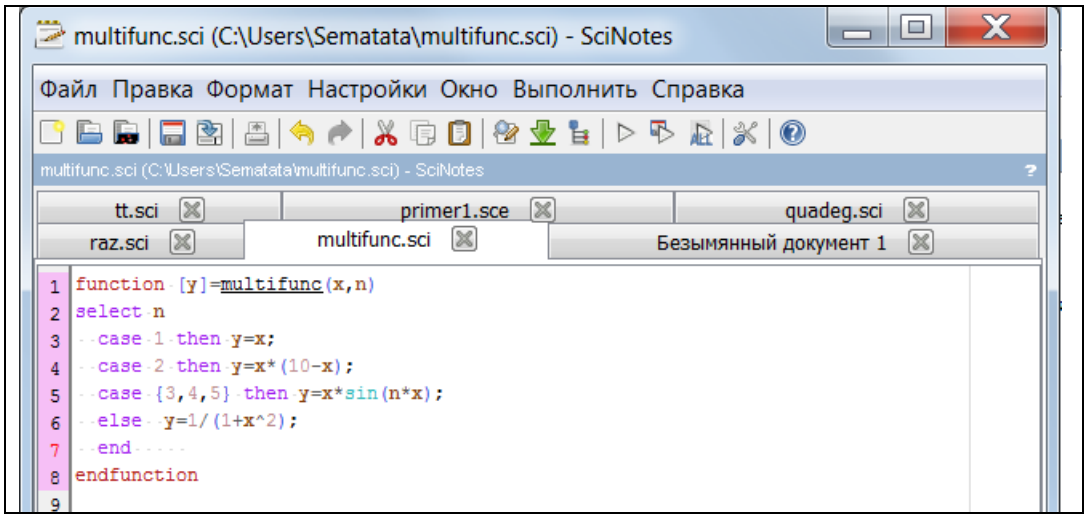

Рис. 1.3.4-8. Функция, реализующая множественное разветвление

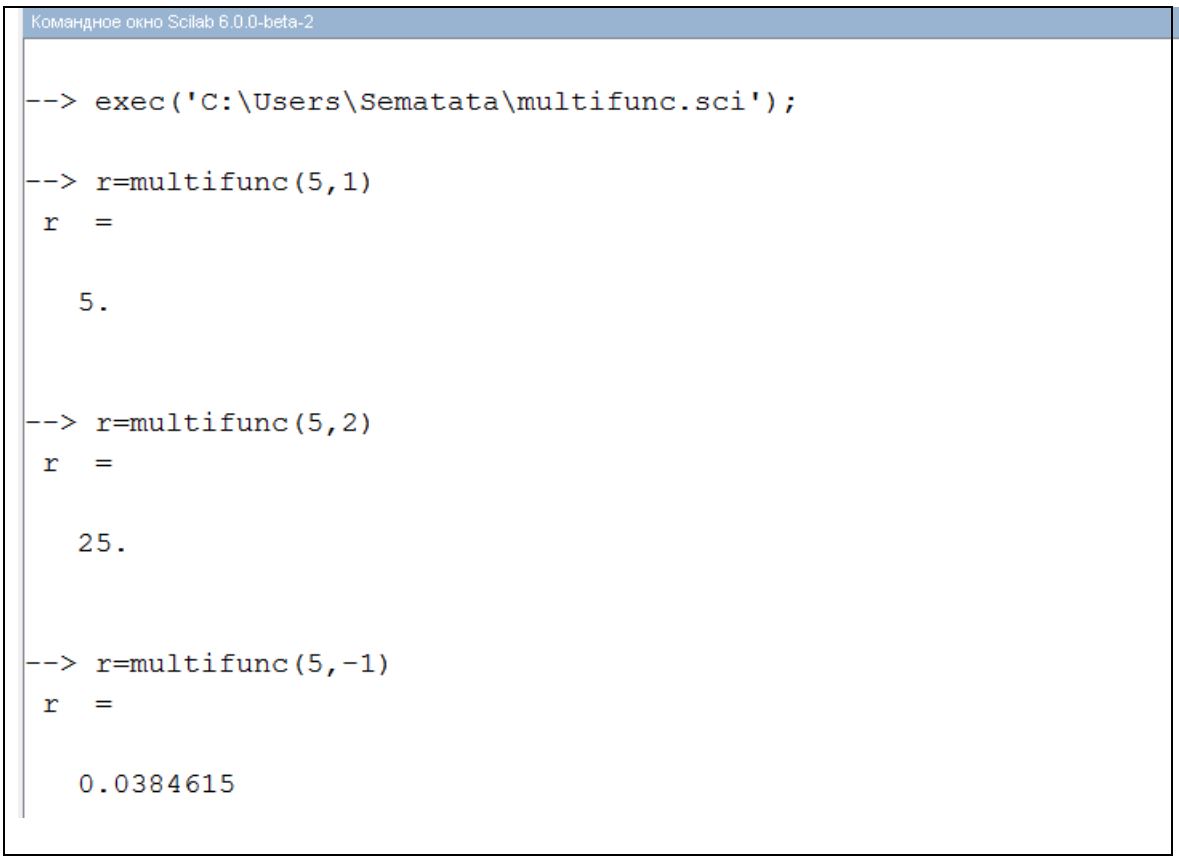

Рис. 1.3.4-9. Обращения к функции **multifunc()**

Оператор регулярного цикла - for...end. Оператор цикла типа ***for...end*** обычно используются для организации вычислений с заданным числом повторений циклов. Конструкция такого цикла имеет следующий вид:

for **var=s:d:e**
Инструкция1
....
ИнструкцияN
end

где **s** - начальное значение переменной цикла **var**, **d** - приращение этой переменной и **e** - конечное значение управляющей переменной, при превышении которого цикл завершается. Возможна и запись в виде **s:e** (в этом случае **d=l**). Список выполняемых в цикле инструкций завершается оператором **end**.

В качестве примера использования оператора ***for...end*** вычислим сумму элементов массива **x**, значения которого определены в командном окне с использованием **sci**-функции **summa()** (рис.1.3.4-10), параметром которой служит вектор **x**. Количество элементов массива **x** определяется функцией **length**. Кроме обращения к функции в командном окне предусмотрена проверка результата вычислений с использованием встроенной функции **sum(x)** (рис.1.3.4-11).

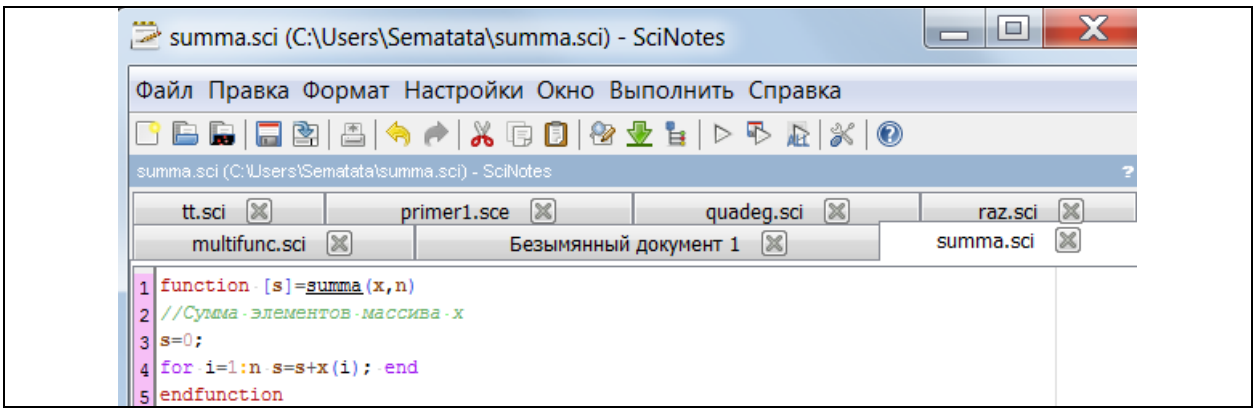

Рис. 1.3.4-10. Функция, вычисляющая сумму элементов массива

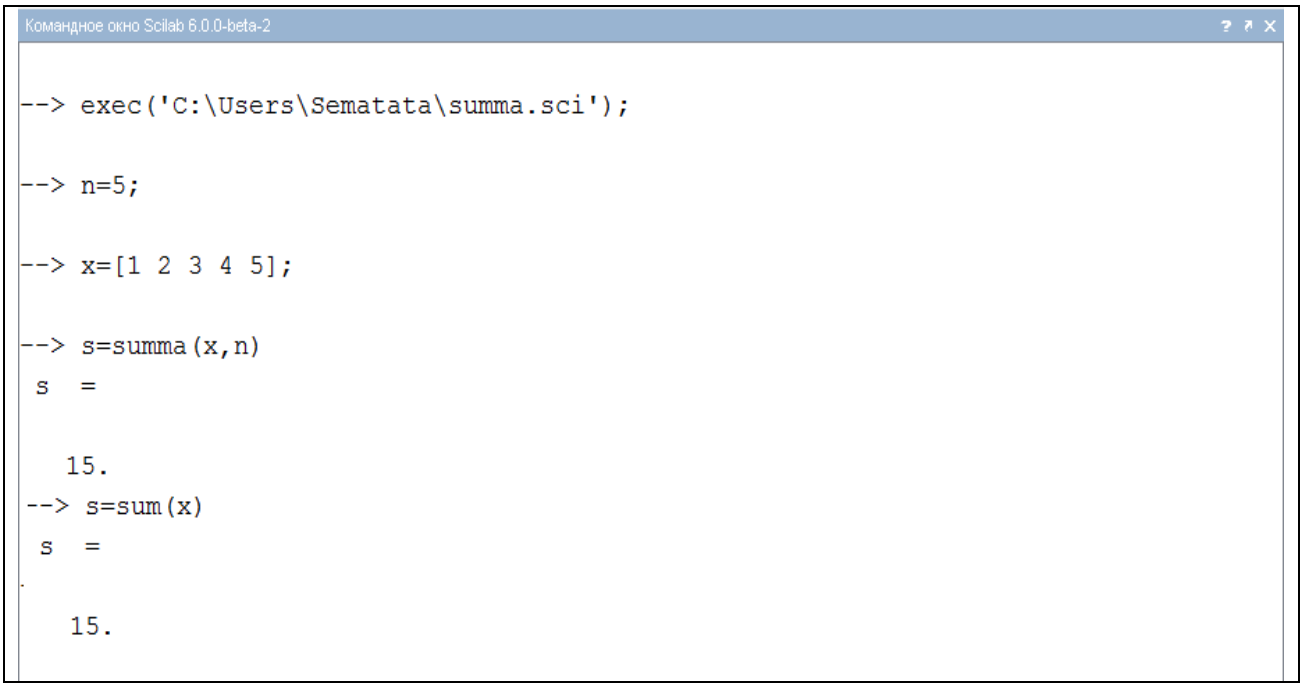

Рис. 1.3.4-11. Обращение к функции **summa()** и встроенной функции **sum()**

В цикле может быть использован оператор ***continue***, который передает управление в следующую итерацию цикла, пропуская операторы, которые записаны за ним, причем во вложенном цикле он передает управление на следующую итерацию основного цикла. Оператор ***break*** может использоваться для досрочного прерывания выполнения цикла (например, при отладке участка программы). Как только он встречается в программе, цикл прерывается.

Кроме простых регулярных циклов в Scilab имеется возможность организации **вложенные циклы**. Рассмотрим пример формирования двумерного массива **a(3,3)** (рис. 1.3.4-12 -13).

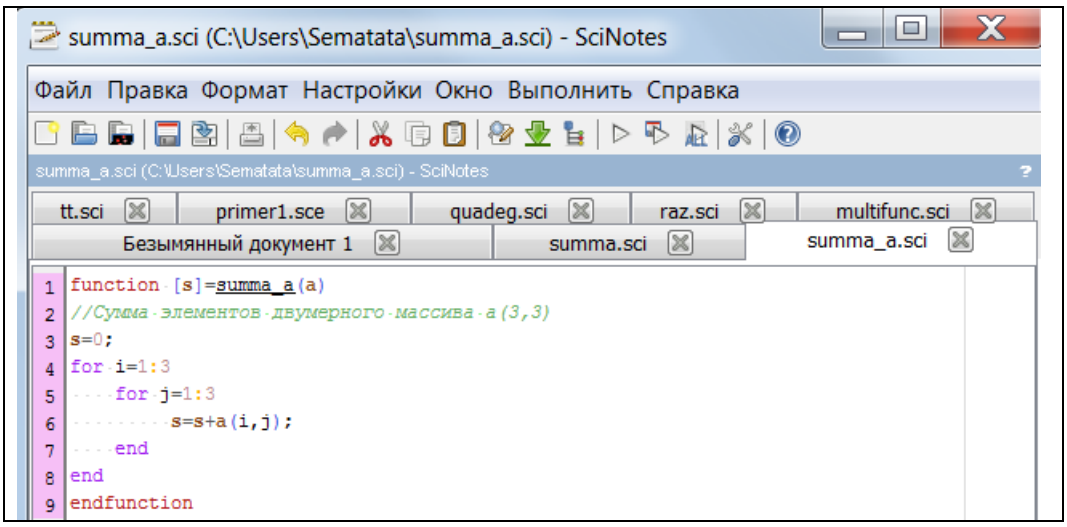

Рис. 1.3.4-12. **Sci**-файл, иллюстрирующий вложенные циклы

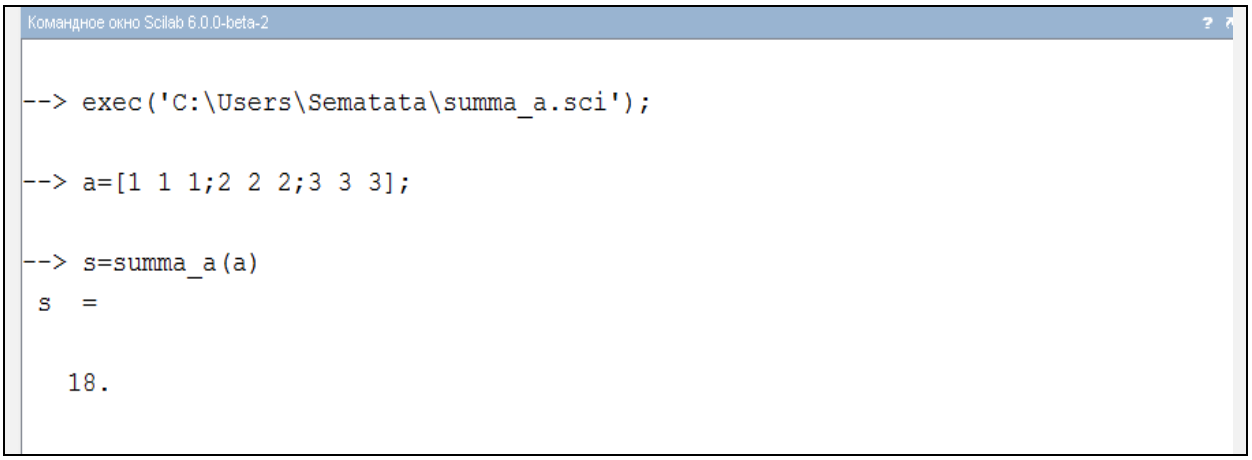

Рис. 1.3.4-13. Обращение к **sci**-файлу с именем **summa-a.sci**

Оператор итеративного цикла– while…end. Общий вид структуры ***while…end*** выглядит следующим образом:

while *ЛогическоеВыражение*
Инструкции
End

Отличительной особенностью этой структуры является то, что инструкции, расположенные в теле структуры повторения, выполняются только в том случае, если некоторое *ЛогическоеВыражение* «истинно». Как только условие становится «ложным», происходит выход из структуры повторения, и управление передается на инструкцию, расположенную после ключевого слова **end**. Приведем простой пример (рис 1.3.4-14).

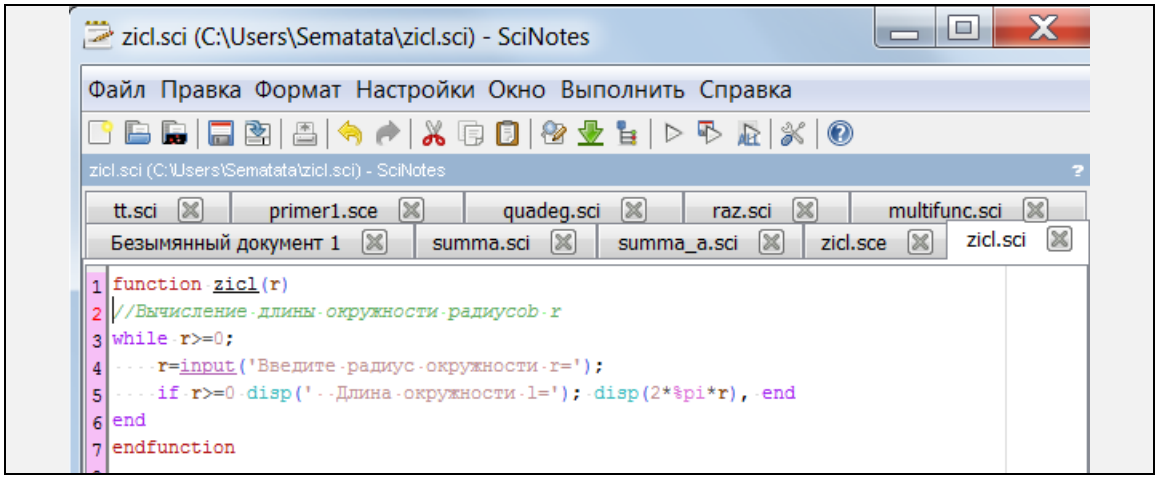

Рис. 1.3.4-14. Диалоговая программа, использующая оператор ***while…end***

Эта программа, сохраненная в **sci**-файле с именем **zicl.sci**, служит для многократного вычисления длины окружности по вводимому пользователем значению радиуса **r**, где диалог реализован с помощью команды **input.** Строки, связанные с вводом переменной **r** и вычислением длины окружности включены в управляющую структуру **while...end**. Это необходимо для циклического повторения вычислений при вводе различных значений **r**. Пока **r>=0**. цикл повторяется. Но стоит задать **r<0**, вычисление длины окружности перестает выполняться, а цикл завершается. Поскольку во 2-й строке программы величина **r** определена равной 0, цикл повторяется хотя бы один раз.

Работа с программой в командном окне представлена на рис. 1.3.4-15.

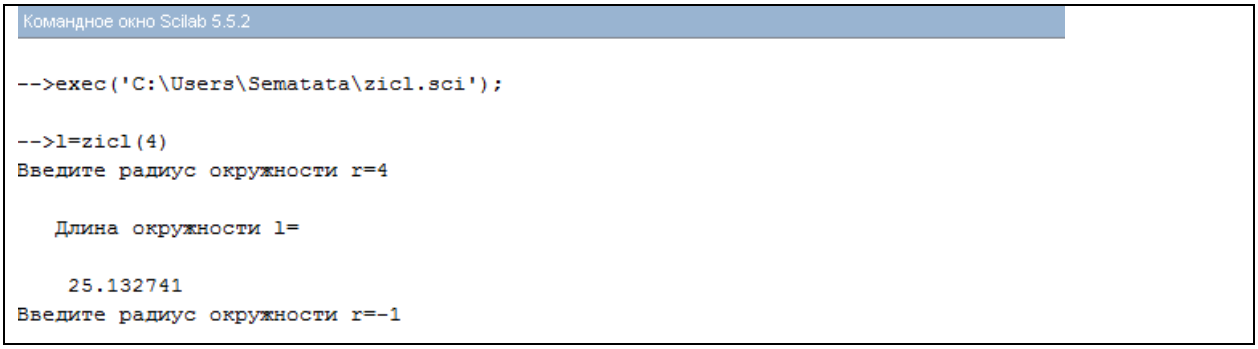

Рис. 1.3.4-15. Обращения к программе вычисления длины окружности

В управляющих структурах, в частности в циклах ***for*** и ***while***, часто используются операторы, влияющие на их выполнение. Так, оператор ***break*** может использоваться для досрочного прерывания выполнения цикла. Как только он встречается в программе, цикл прерывается.

Рассмотрим пример досрочного прерывания цикла при выполнении условия i=5 (рис.1.3.4-16).

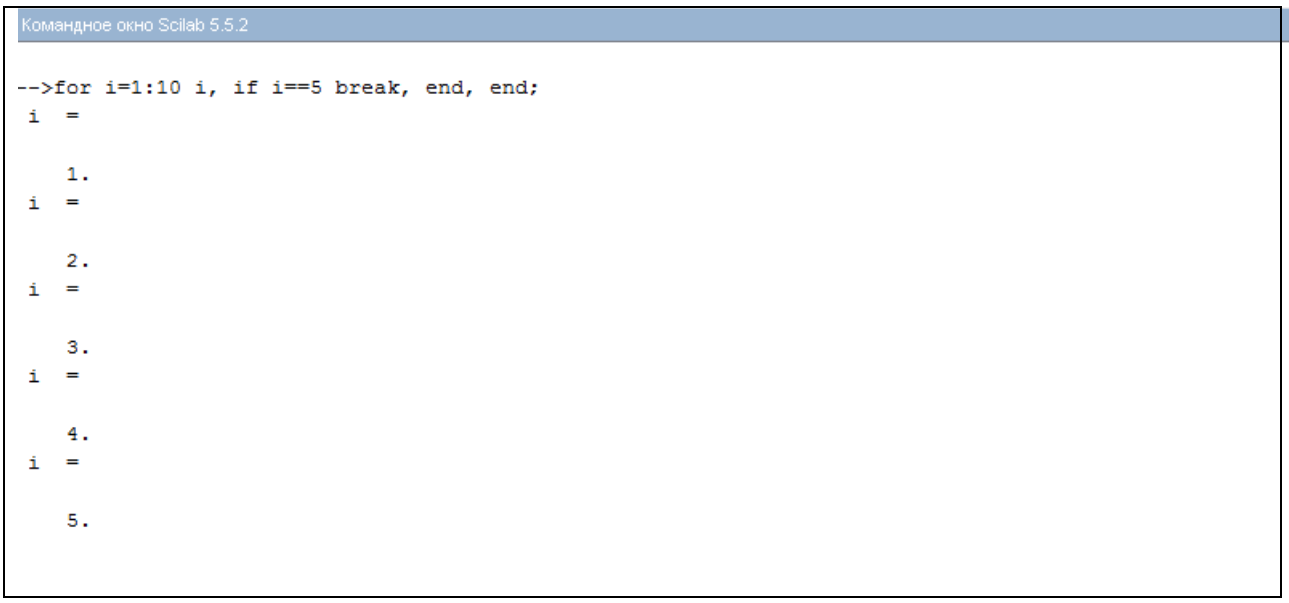

Рис. 1.3.4-16. Прерывание программы с применением оператора ***break***

Оператор ***continue*** передает управление в следующую итерацию цикла, пропуская операторы, которые записаны за ним, причем во вложенном цикле он передает управление на следующую итерацию основного цикла. Ниже приведен пример вычисления суммы и произведения положительных элементов двумерного массива b(3,3) (рис. 1.3.4-17).

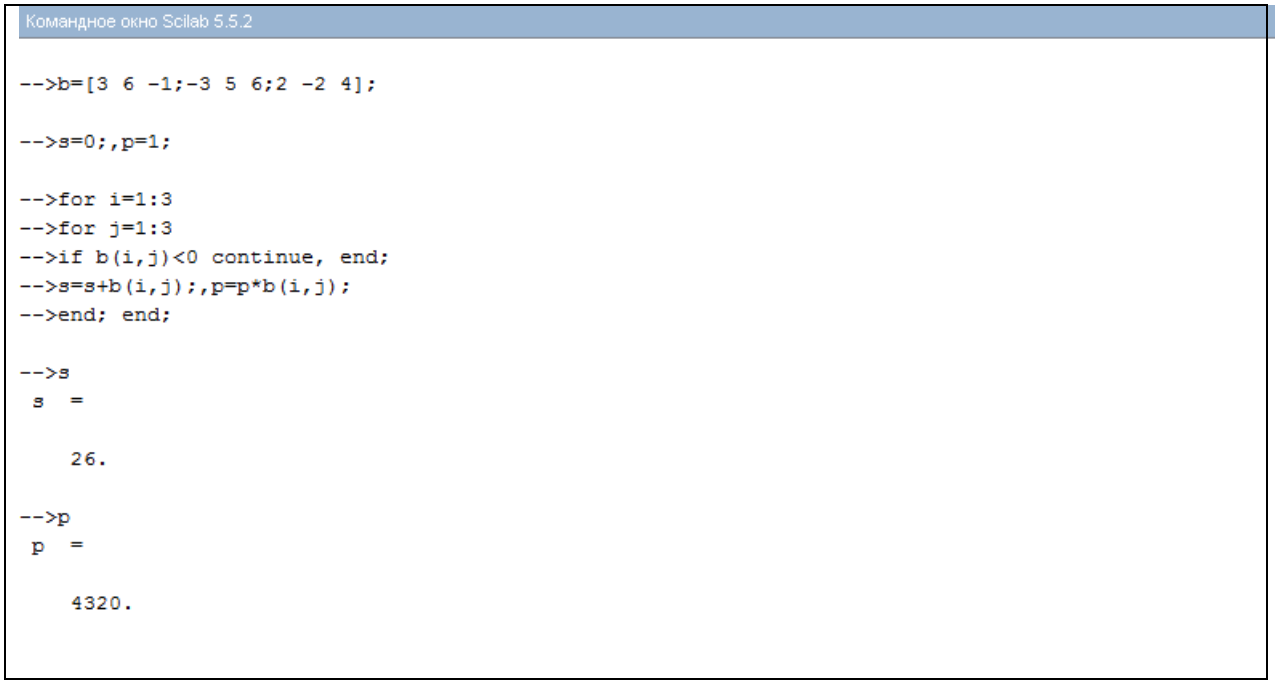

Рис. 1.3.4-17. Прерывание программы с применением оператора ***continue***

1.3.5. Примеры решения задач с использованием sci-файлов

Пример. 1.3.5-1. **Даны n чисел** $b_1, b_2, ...b_n$ **. Требуется вычислить их сумму:** $f(b_1)+f(b_2)+...+f(b_n)$, *где*

$$f(x)=\begin{cases} x^2, & \text{если } x>3; \\ x, & \text{если } 1<x\le 3; \\ x/3 & \text{в остальных случаях.} \end{cases}$$

Для решения поставленной задачи разработана функция **fb(x)**, реализующая алгоритм вычисления текущего значения функции. Функция имеет один входной параметр – текущее значение элемента массива **b**, и один выходной параметр – **y** (рис.1.3.5-1). Обращение к функции происходит в цикле, организованном для вычисления суммы (рис. 1.3.5-2).

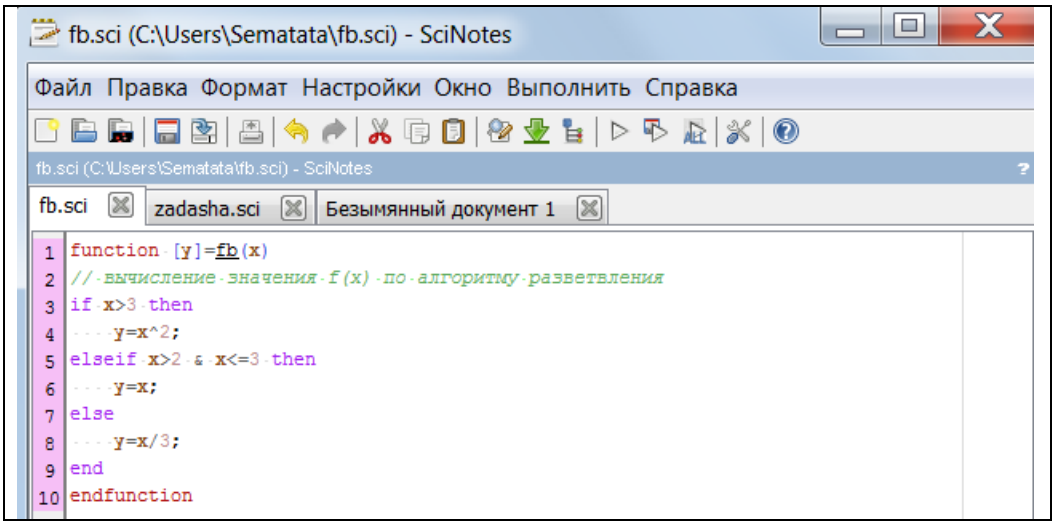

Рис. 1.3.5-1. Функция, реализующая алгоритм Примера 1. 3.5-1

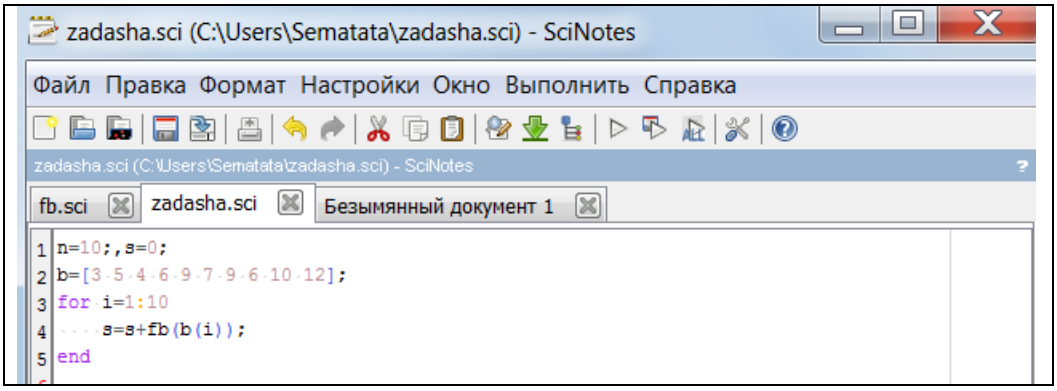

Рис. 1.3.5-2. Программа, реализующая вычисление суммы чисел

Для вычисления суммы значений функции создан файл с именем **zadasha.sci**, в котором сначала заданы количество чисел (**n=10**) и вектор их

значений (**b**), а затем организован регулярный цикл для обращения в функции **fb()** и вычислению суммы.

Вычисления производятся запуском **fb.sci** и функции **zadash.sci** в Командном окне. Результат его выполнения выведен на рис.1.3.5-5.

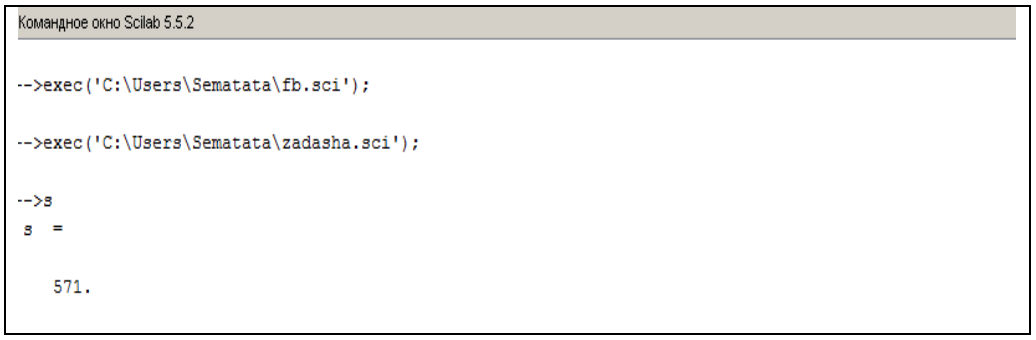

Рис. 1.3.5-3. Сборка и выполнение задачи

Пример. 1.3.5-2.Сформировать из произвольных чисел двумерный массив a(3,4). Вычислить и вывести одномерный массив b, каждый элемент которого есть среднее арифметическое элементов соответствующей строки массива a.

На рис. 1.3.5-4 приведен **script**-файл с именем **zadasha2**, где введена матрица, **a**, состоящая из 3-х строк и 4-х столбцов. Организован цикл по количеству формируемых элементов массива **b** путем обращения к функции **sred_ar()**. В функцию передается массив **a**, номер строки **i** и количество элементов в строке **m.** Вывод элементов массива **b** предусмотрен в столбец.

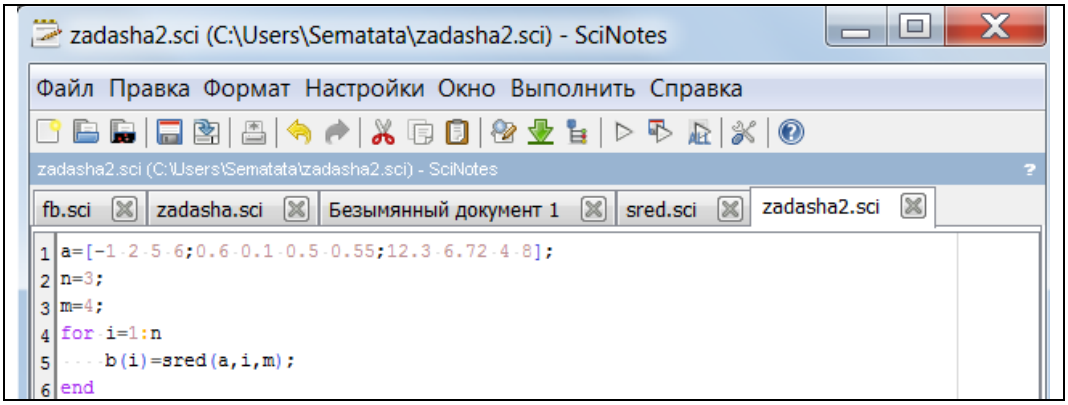

Рис. 1.3.5-4. Программа формирования массива **b**

Функция **sred_ar()** (рис. 1.3.5-5) предназначена для формирования **i**-го элемента массива **b**, равного среднему арифметическому элементов строки массива **a.**

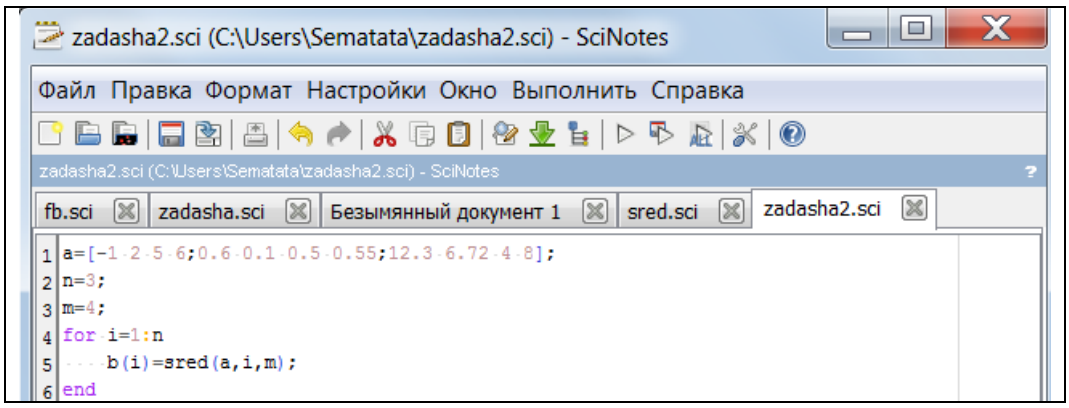

Рис. 1.3.5-5. Функция **sred_ar()**, вычисляющая среднее арифметическое элементов строки массива **a**

В результате запуска **sci**-файле с именем **zadasha2** в окно **Командного окна** выводится столбец элементов массива **b**

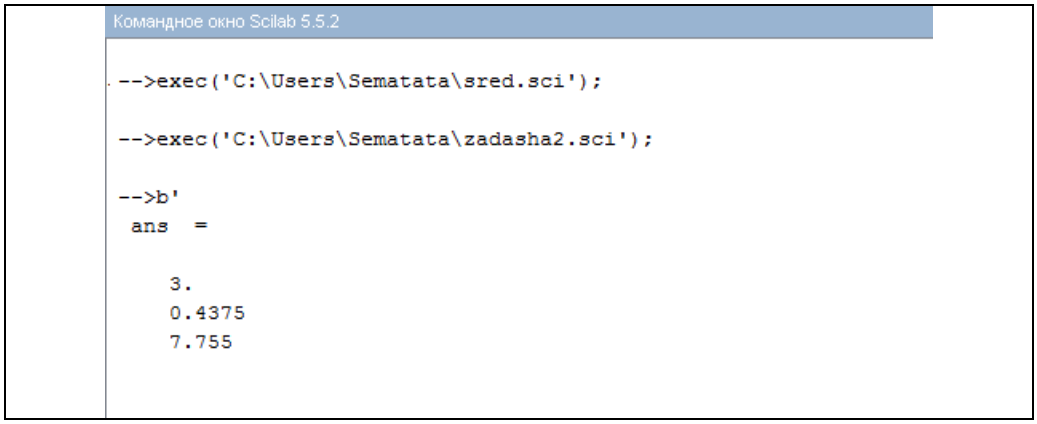

Рис. 1.3.5-6. . Сборка и выполнение задачи

Пример. 1.3.5-3. Задать действительные числа a,b, натуральное n (a<b) и вычислить выражение $s=(F_1+F_2+...+F_n)h$**, где** $h=\frac{b-a}{n}$**, если** $F_i=-\frac{a+(i-1/2)h}{1+(i-1/2)h^2}, \quad i=1,2,...,n.$

Решение задачи требует разработки 2-х функций: **fab(a, h,i)**, предназначенной для вычисления i-го слагаемого (рис.1.3.5-7) и **sumf(a,h,n)**, предназначенной для вычисления заданного выражения (рис. 1.3.5-8).

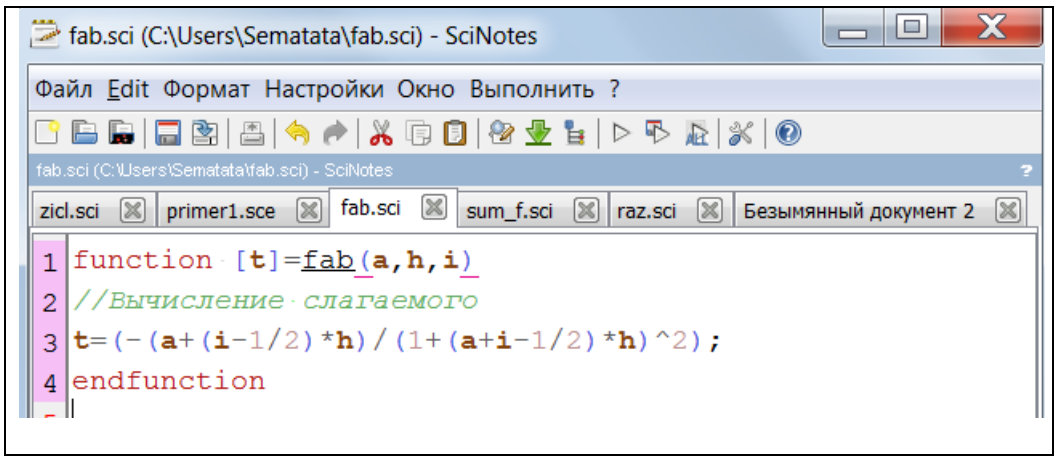

Рис. 1.3.5-7. Функция **fab(),** вычисляющая значение i-го слагаемого

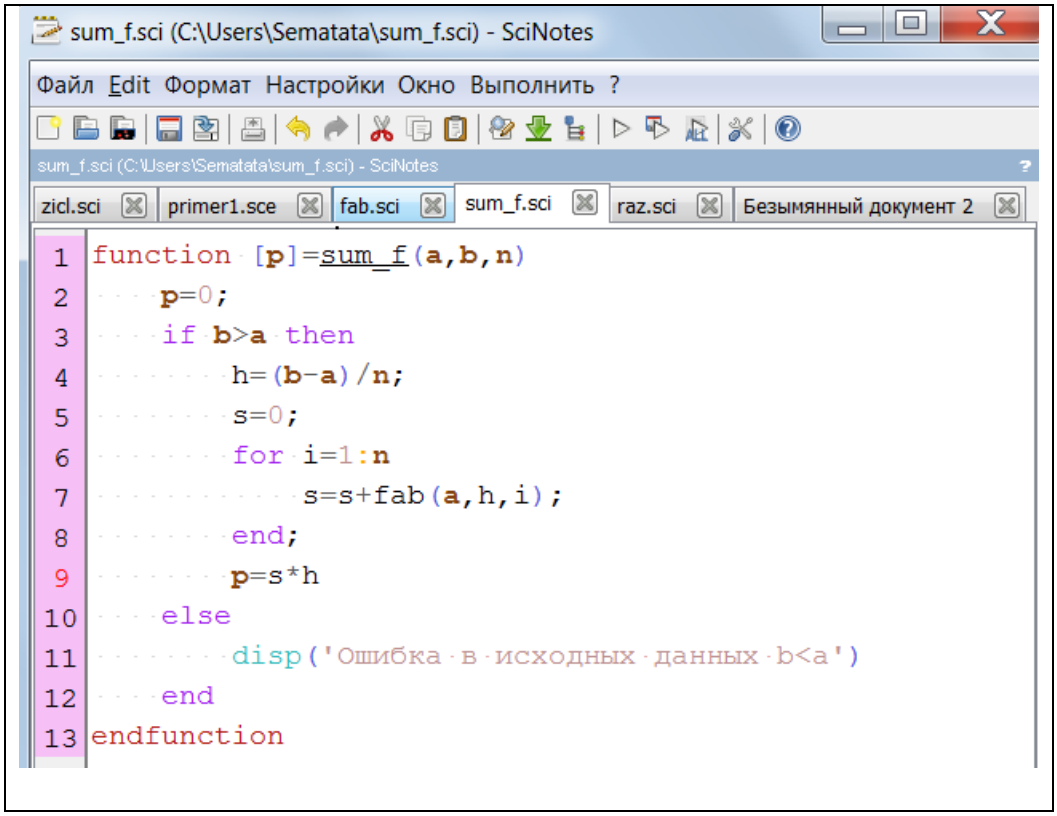

Рис. 1.3.5-8. Функция **sum_f()**, вычисляющая заданное выражение

Запуск на выполнение осуществляется из командного окна к функции **sum_f()**. Предварительно переменным **a**, **b** и **n** присваиваются числовые значения. Проверка правильности ввода исходных данных предусмотрена в функции **sum_f()**. Вычисления выполняются и результат выводится на экран только в случае если **b>a**, иначе в командной строке появляется сообщение «Ошибка в исходных данных b<a» (рис.1.3.5-9).

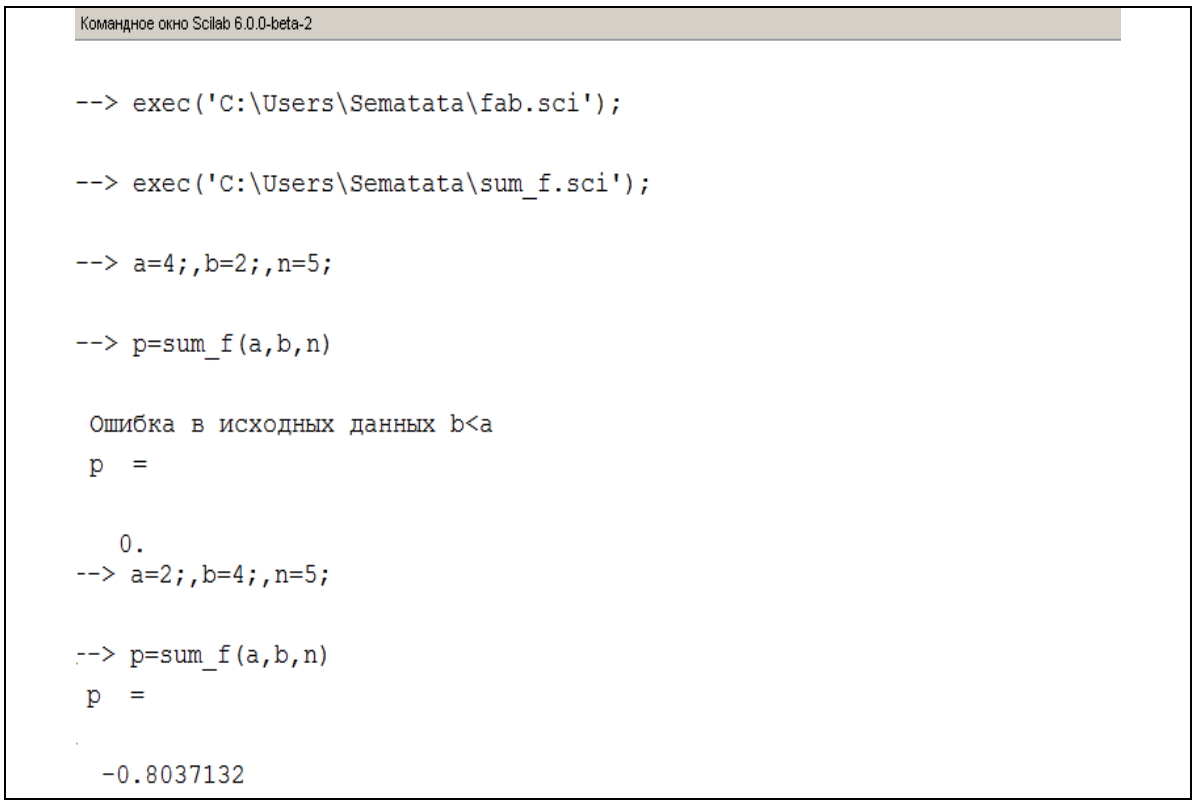

Рис. 1.3.5-9. Запуск функции **sumf()** на выполнение в ***Командном окне***

1.3.6. Лабораторная работа по теме «Средства алгоритмизации и программирования в Scilab»

1. Вопросы, подлежащие изучению

1) Виды **sci-**файлов.
2) Создание и сохранение новых, и открытие ранее созданных sci-файлов.
3) Особенности файлов-сценариев и **sci-**функций.
4) Запуск на выполнение файла-сценария из текстового редактора.
5) Запуск на выполнение файла-сценария из Командного окна
6) Обращения к файлам и **sci-**функциям.
7) Средства языка программирования в системе Scilab.
8) Основные операторы **sci**-языка их назначение и форматы**.**

2. Общее задание

1) ***Изучите материал Темы 1.3 (п.п. 1.3.1 – 1.3.5).***
2) ***Выберете индивидуальное задание*** *из* ***табл. 1.3.6-1****.*
3) ***Разработайте sci****-функции для реализации стандартных алгоритмов: вычисления конечных сумм, разветвлений, поиска минимума и максимума в последовательности данных и т.п.*
4) ***Введите*** *и* ***сохранитеm****-функции на внешнем носителе.*
5) ***Создайте*** *новый файл-сценарий, в который введите код программы, описывающий логику решения поставленной задачи.*
6) ***Сохраните****созданный файл в текущем каталоге.*
7) ***Произведите отладку*** *файла-сценария, запуская его на выполнение из текстового редактора командой* ***Выполнить.***
8) ***Подготовьте*** *и* ***введите*** *исходные данные для решения поставленной задачи;*
9) ***Выполните*** *файл-сценарий в* ***Командном окне.***
10) ***Сохраните текст*** *рабочего окна на внешнем носителе.*
11) ***Предоставьте результаты*** *работы преподавателю,* ***ответьте*** *на поставленные вопросы.*
12) ***Выполните*** *команду* ***clear all*** *для очистки* ***Рабочей среды****.*
13) ***Оформите отчет*** *по выполненной работе****.***

3. Варианты индивидуальных заданий

Таблица. 1.3.6-1.

№	Задание
1	Ввести натуральное число n*и вектор действительных чисел* $y_1, y_2 \ldots y_n$ Найти: $\max(\lvert z_1\rvert,\ldots\lvert z_n\rvert)$, где $z_i = \begin{cases} y_i, если \quad \lvert y_i\rvert \le 2; \\ 0.5, если \quad 2 < \lvert y_i\rvert < 4; \\ y_i + Sin(y_i), в\ противных\ случаях. \end{cases}$
2	Вычислить $\sum_{i=1}^{10}(a_i - b_i)^2$, где $a_i = \begin{cases} i, если \quad i - нечетное; \\ i/2, если \quad i - четноею \end{cases}$ $b_i = \begin{cases} i^2, если \quad i - нечетное; \\ i^3/2, если \quad i - четноею \end{cases}$
3	Задать массив $a_1, a_2 \ldots a_{2n}$, состоящий из четного количества элементов. Каждая пара чисел $a_i + a_{i+1}$, где i+1кратно двум, задает координаты вершины ломаной. Построить ломаную, соединив при этом последнюю вершину с первой.
4	Ввести натуральное число nи вектор действительных чисел $b_1, b_2 \ldots b_n$. Вычислить произведение $f(b_1)\cdot f(b_2)\ldots f(b_n)$, где $f(x_i) = \begin{cases} x_i^{\,2}, если \quad x_i\ кратно \quad 2; \\ 2x_i, если \quad x_i\ кратно \quad 7; \\ y_i + Sin(y_i), в\ противных\ случаях. \end{cases}$
5	Ввести натуральное число n и действительное число x. Вычислить $\sum_{i=0}^{n}\frac{1}{n!(n+i)!}\cdot\left(\frac{x}{2}\right)^{2i+n}$
6	Ввести натуральное число n. Найти наибольшее среди значений $k\cdot e^{Sin^2(k+1)}$,гдеk=1, 2,…,n, а также сумму всех полученных значений.
7	Ввести натуральное число n. Среди значений $a_1, a_2 \ldots a_n$, где $a_i = \frac{i-1}{i+1} + Sin\left(\frac{i-1}{i+1}\right)^2$ (i=1,2,…n), найти все положительные и вычислить их сумму.
8	Ввести натуральное число nи вектор действительных чисел $b_1, b_2 \ldots b_n$. Определить положительных или отрицательных чисел в векторе больше и определить наибольшее из отрицательных и наименьшее их положительных чисел.
9	Ввести матрицу B(5,7) и сформировать из первых наибольших элементов строк вектор C(5). Вывести его элементы с строку и столбец.

10	Сформировать вектор по правилу: $a_{k+1} = 2a_k - a_{k-1}$, где k=2,3,…7, если $a_1 = Cos^2(1), a_2 = -Sin^2(1)$. Найти сумму квадратов тех чисел, которые не превосходят 2.
11	Ввести натуральное число nи вектор действительных чисел $b_1, b_2 ... b_n$. Найти количество двух соседних положительных чисел и двух соседних чисел разного знака.
12	Ввести квадратную матрицу A(4,4). Сформировать их максимальных элементов ее столбцов вектор X, вывести его элементы на экран в прямой и обратной последовательности.
13	Ввести вектор целых чисел $b_1, b_2 ... b_{10}$. Преобразовать его таким образом, чтобы сначала располагались нули, затем все остальные элементы. Определить сумму и количество элементов, значения которых кратно 5.
14	Ввести вектор вещественных чисел $z_1, z_2 ... z_{18}$. Создать из него массив x, каждый элемент которого максимальный из 3-х элементов, идущих подряд в массивеz.
15	Сформировать матрицу A(4,4) по правилу: $A(i, j) = \begin{cases} i + 5j, если\ i \le 2; \\ 7i + 2(j-1), если\ i > 2. \end{cases}$ Найти и вывести значения и индексы двух одинаковых элементов. Если таковых не окажется, вывести сообщение.
16	Сформировать матрицу D(3,2) по правилу: $D(i, j) = \frac{i^2 - j^2}{2.}$ Создать вектор из отрицательных элементов полученной матрицы.
17	Задать натуральное число **n**. Посчитать какая из матриц размером **n**на **n** содержит больше положительных элементов, если их элементы формируются по правилу: $a(i, j) = Sin(i + j/2); \quad b(i, j) = Cos(i^2 + n); \quad c(i.j) = Sin\left(\frac{i^2 - j^2}{n}\right).$ Вывести на экран сформированные матрицы.
18	Ввести квадратную матрицу вещественных чисел A(4,4). Найти сумму наибольших значений элементов ее строк. Сформировать новую матрицу B(4,4) путем умножения каждого матрицы A на найденную сумму и делением его на определитель исходной матрицы.
19	Ввести матрицу вещественных чисел A(4,7) и получить из нее вектор C(4), элементы которого это: • наибольший из элементов в 1-й строке; • наименьший из элементов во 2-й строке;

	• среднее арифметическое элементов 3-й строки; • сумма элементов 4-й строки.
20	Ввести натуральное число nи матрицу вещественных чисел C(n,n). Найти среднее арифметическое наибольшего и наименьшего значений ее элементов и заменив этим значением диагональные элементы вывести матрицу C на экран.
21	Ввести натуральные числа k1, k2 и действительную матрицу размера 8x4. Поменять в матрице местами элементы k1 и k2 строк.
22	Ввести натуральное число nи матрицу вещественных чисел C(n,9). Найти среднее арифметического каждого из столбцов, имеющих четные номера.
23	Ввести вектора действительных чисел x(5), y(6),z(7). Вычислить величину tпо следующему алгоритму: $t = \begin{cases} (\max(x_1, x_2, ...x) + \min(y_1, y_2, ...y_6))/2, & \text{если} \quad \lvert\min(z_1, z_2, ...z_7)\rvert < 4; \\ 1 + (\max(z_1, z_2, ..z_7))^2 & \text{в противном случае.} \end{cases}$
24	Ввести вектора действительных чисел x(5). Получить для x=1, 3, 4 значения $p(x+1) - p(x)$, где $p(y) = a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1$
25	Ввести вектора действительных чисел x(10).Получить из него другой массив p(10), элементы которого упорядочены по возрастанию.
26	Ввести матрицу вещественных чисел A(3,4). Заменить элементы строки матрицы с максимальной суммой значений элементов – единицами, с минимальной - 2, а остальные элементы матрицы положить равными нулю.
27	Сформировать матрицу A(4,4) по правилу $A(i, j) = 5(i + j) - j$. Удалить из него столбцы, содержащие элементы, меньшие 10.
28	Сформировать матрицу B(9,3) по правилу $B(i, j) = Sin(i + j/2)$. Определить наименьший элемент в каждой строке матрицы и записать его в соответствующий элемент вектора C. Вывести полученный вектор C.
29	Ввести матрицу вещественных чисел A(3,4), все элементы которой различны. В каждой строке следует выбрать наибольшее и наименьшее значение, а сумму индексов столбцов, в которых они расположены записать в соответствующий элемент вектора C(3).
30	Ввести матрицу вещественных чисел A(4,4). Получить последовательности элементов главной и побочной диагонали, создать из этих элементов вектора B(4) и C(4) и вывести их на экран.

4. Содержание отчета

1) В форме комментариев:
   - Название лабораторной работы
   - ФИО студента, номер группы
   - № варианта
   - Индивидуальное задание
2) Протокол вычислений (сессии) в **Командном окне**, снабженный необходимыми комментариями.

1.3.7. Контрольные вопросы по теме

1) Что такое файл-сценарий и каковы его особенности?
2) Каким образом файл-сценарий запускается на выполнение?
3) Что такое **sci-**функци**я**?
4) В чем отличие файла-сценария от **sci-**функции?
5) Может ли **sci-функция** иметь несколько выходных параметров?
6) Обращение к **sci-функции**.
7) Формат оператора **input().**
8) Как с использованием оператора **if…end** реализовать стандартное, усеченное и вложенное разветвление?
9) Формат оператора множественного разветвления **switch**.
10) Формат оператора регулярного цикла **for…end**, особенности задания значений переменной цикла.
11) Назначение операторов **continue** и **brek**.
12) Оператор итеративного цикла **while…end** и его структура.

Раздел 2. Технология решения вычислительных задач средствами Scilab

Тема 2.1. Решение нелинейных уравнений

2.1.1. Численное решение нелинейных уравнений
2.1.2. Лабораторная работа
2.1.3. Контрольные вопросы по теме

2.1.1. Численное решение нелинейных уравнений

В общем виде уравнение можно представить, как **f(x)=0**. В зависимости от вида функции **f(x)**различают алгебраические и трансцендентные уравнения.

Алгебраическими уравнениями называются уравнения, в которых значение функции f(x) представляет собой полином **n**-й степени (**n≥2**). Всякое неалгебраическое уравнение называется **трансцендентным.** Функция **f(x)**в таких уравнениях содержит хотя бы одну из следующих функций: показательную, логарифмическую, тригонометрическую или обратную тригонометрическую.

Решением уравнения **f(x)=0** называется совокупность корней $\bar{x}$, при которых уравнение обращается в тождество $\mathbf{f(\bar{x}) \equiv 0}$. Однако, точные значения корней могут быть найдены аналитически только для некоторых типов уравнений. Еще меньше возможностей при получении точного решения трансцендентных уравнений. Следует отметить, что задача нахождения точных значений корней не всегда корректна. Так, если коэффициенты уравнения являются приближенными числами, точность вычисленных значений корней заведомо не может превышать точности исходных данных. Эти обстоятельства заставляют рассматривать возможность отыскания корней уравнения с ограниченной точностью (приближенных корней).

Задача нахождения корня уравнения с заданной точностью ε (ε **>0**)считается решенной, если найдено приближенное значение $\bar{x}$, которое отличается от точного значения корня ξ не более чем на значение ε: $|\xi - \bar{x}| \le \varepsilon$.

Для отделения корней нелинейных уравнений применяются **графический** и **аналитический** методы. Для уточнения корней с заданной степенью точности существуют множество численных методов. Самыми распространенными из них являются: **метод Ньютона, метод хорд, метод итераций** и **метод половинного деления.**

Процесс нахождения приближенного корня нелинейного уравнения состоит из двух этапов:

1) **отделение корня уравнения** (локализация корня на отрезке);
2) **уточнение корня с заданной точностью.**

Пример 2.1.1-1.Отделить и уточнить корень уравнения $2^x - 4x = 0$**.**

Согласно теореме о существовании и единственности корня на отрезке, найдем отрезок, на концах которого функция $f(x) = 2^x - 4x$ имеет разные знаки, а первая производная непрерывна и знакопостоянна (рис. 2.1.1-1).

```
Командное окно Scilab 6.0.0-beta-2

--> x=0:0.2:1;

--> deff('[y]=f(x)','y=2^x-4*x');

--> deff('[y1]=f1(x)','y1=2^x*log(2)-4');

--> f(x)
 ans  =

   1.    0.3486984  -0.2804921  -0.8842834  -1.4588989  -2.


--> f1(x)
 ans  =

  -3.3068528  -3.203783  -3.0853868  -2.9493853  -2.7931607  -2.6137056
```

Рис. 2.1.1-1. Отделение корня нелинейного уравнения

Условия существования и единственности корня на отрезке **[0;1]**выполняются. Команды построения графика функции **f(x)** на отрезке **[0;1]**и график функции приведены на рис. 2.1.1-2.

```
--> z=f(x);

--> plot(x,z,x,0*z)

--> xtitle('График функции f(x)','x','f(x)');
```

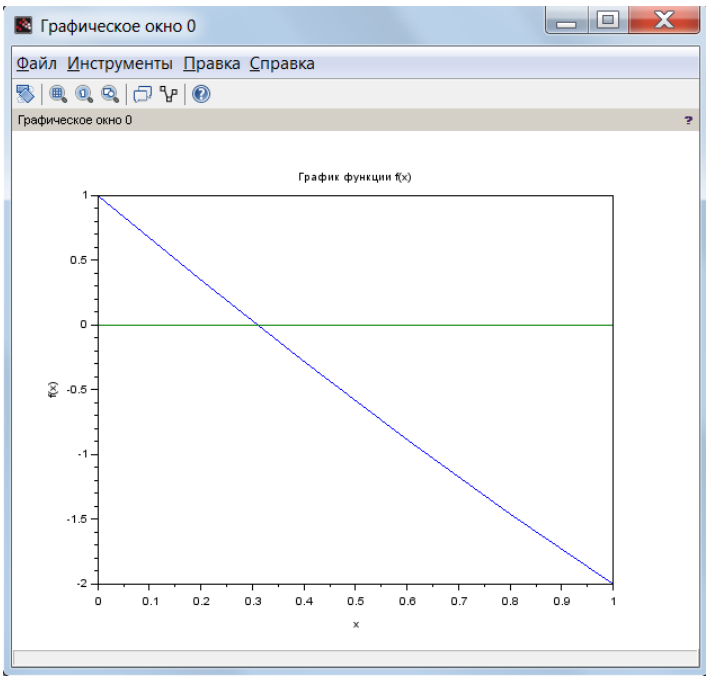

Рис. 2.1.1-2. Построение графика функции**f(x)**

Для решения нелинейных уравнений в среде Scilab используются встроенные функции: f**solve()** и **roots()**.

Функция **fsolve()**используется для нахождения вещественных корней уравнений вида **f(x)=0**. В простейшем варианте обращения кроме указателя на функцию, корень которой необходимо найти, задается **x0**, с которой начинается поиск:

x = fsolve(x0,f),

где **f-** функция, описывающая левую часть уравнения **y(x)=0;**

x0- начальное приближение корня.

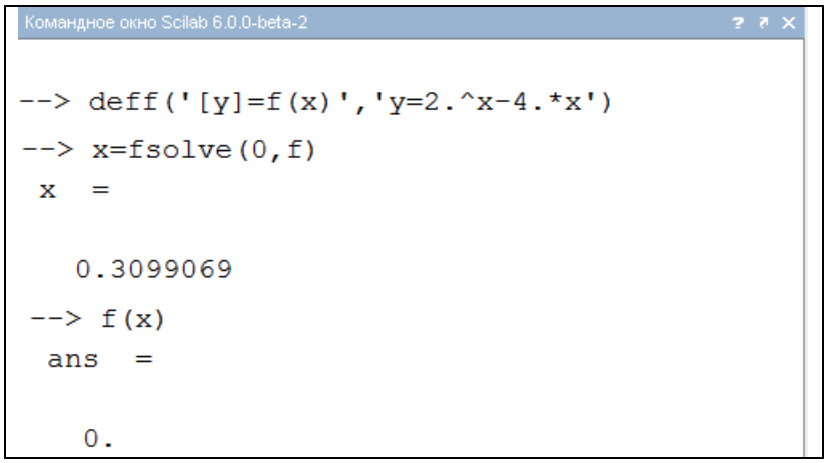

Рис. 2.1.1-3. Уточнение корня уравнения с использованием функции **fsolve()**

Вместо явного задания выражения для функции **f(x)** можно создать соответствующую функцию, запомнив ее в виде **sci**-файла (рис. 2.1.1-4).

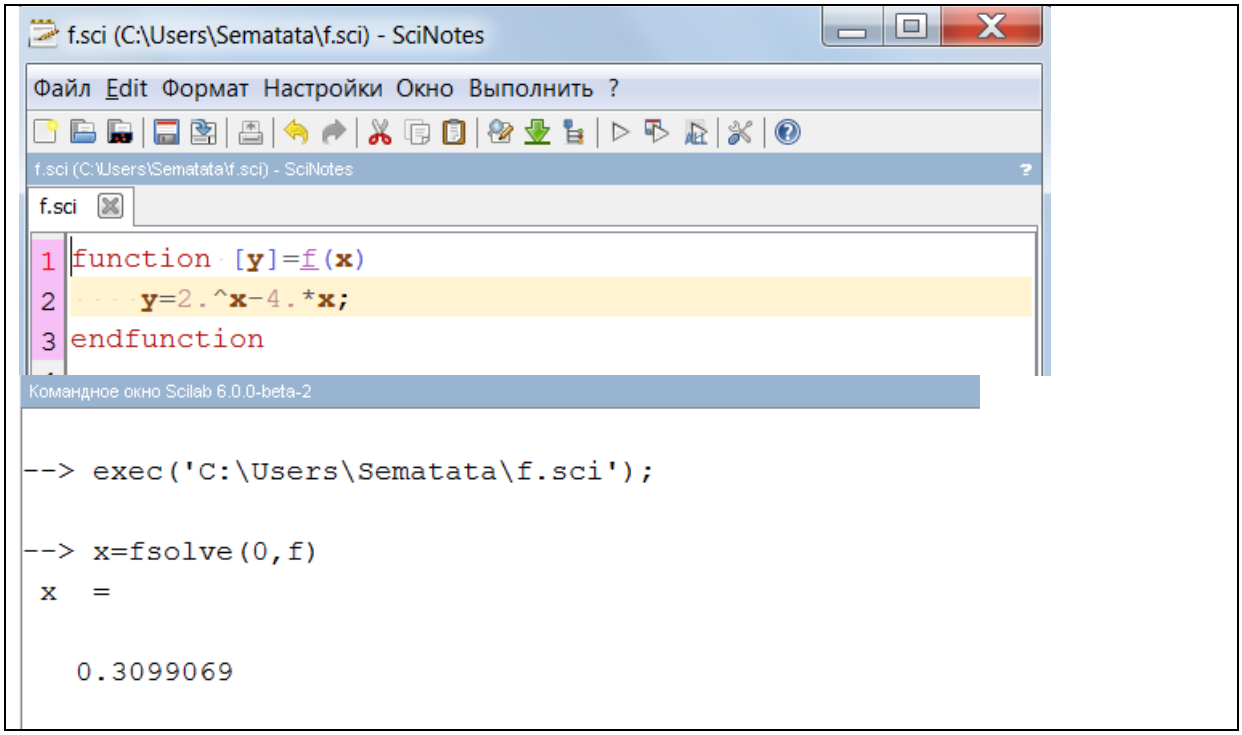

Рис. 2.1.1-4.Описание левой части уравненияв виде sci-файла

Рассмотрим технологию определения корня с помощью функции **solve()** на следующем примере, в котором уравнение имеет несколько корней (рис.2.1.1-5).

Пример 2.1.1-2. Решить уравнение $e^x/5-2(x-1)^2=0$ **.**

Отделим корни уравнения $e^x/5-2(x-1)^2=0$ ($x_1 \in [0;1], x_2 \in [1;2], x_3 \in [5;6]$) и вычислим все корни, последовательно вызывая функцию **fsolve()** с различными начальными приближениями. Ниже показана возможность задания начальных приближений к корням в виде вектора, и тогда функцию можно вызвать один раз (рис. 2.1.1-5).

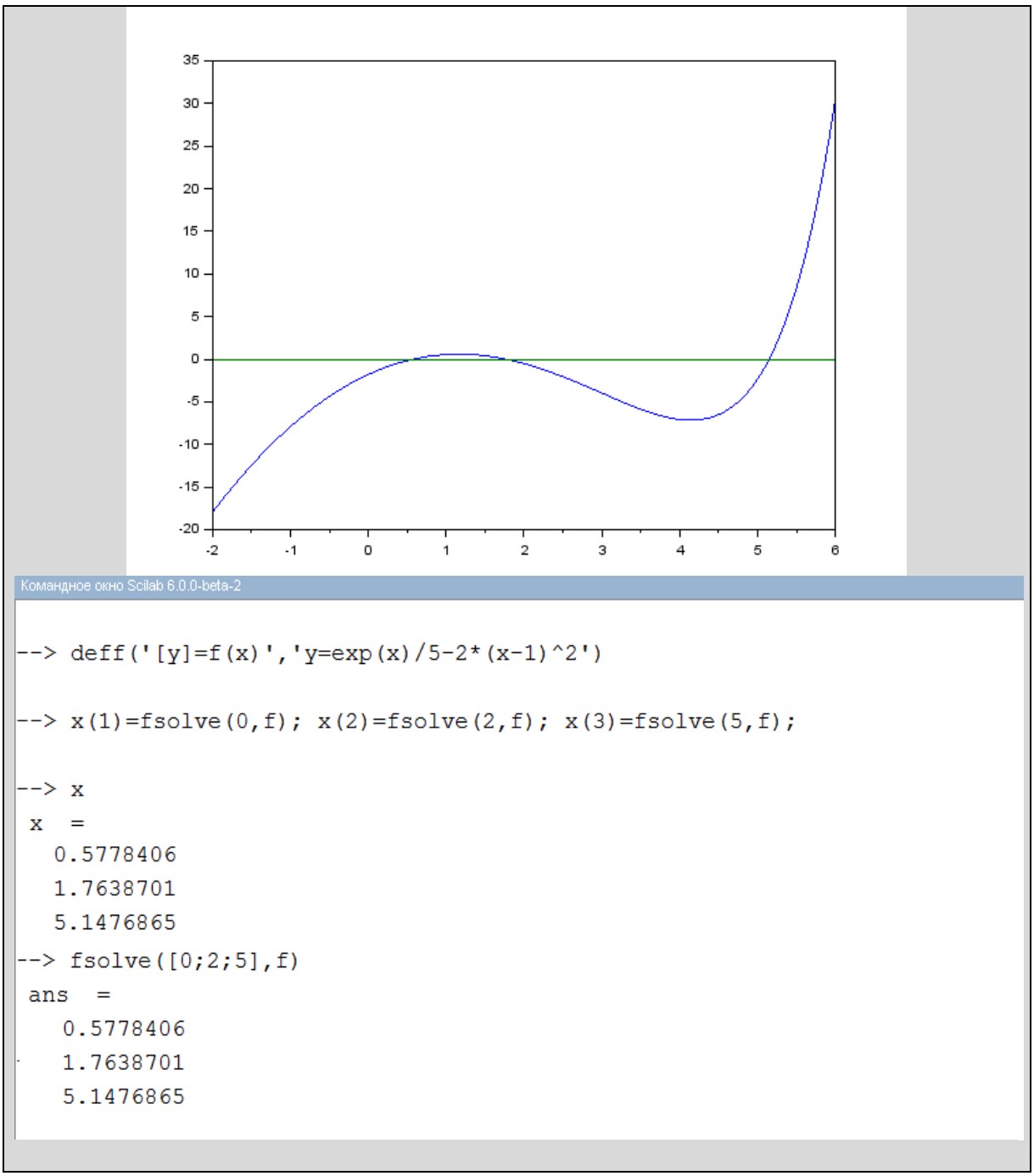

Рис. 2.1.1-5. Уточнение нескольких корней уравнения с использованием функции **solve()**

Функция Scilab **roots()** используется для вычисления корней полинома вида $P(x) = a_1 x^n + a_2 x^{n-1} + ... + a_n x + a_{n+1}$. Перед ее использованием создается список коэффициентов (даже нулевых), затем выполняется функция

poly(a, ‘x’, [‘fl’]),

где **a**- это число или матрица чисел, **x** - символьная переменная, **fl** – необязательная символьная переменная, определяющая способ задания полинома. Символьная переменная fl может принимать только два значения "roots" или "coeff” (соответственно ‘r’или ‘c’). Если fl=c, то будет

сформирован полином с коэффициентами, хранящимися в параметре **a**. Если же fl=r, то значения параметра **a** воспринимаются функцией как корни, для которых необходимо рассчитать коэффициенты соответствующего полинома. По умолчанию fl=r.

Теперь можно использовать функцию **roots(p)**, у которой в качестве аргумента указывается имя полинома, созданного функцией **poly** (рис. 2.1.1-6).

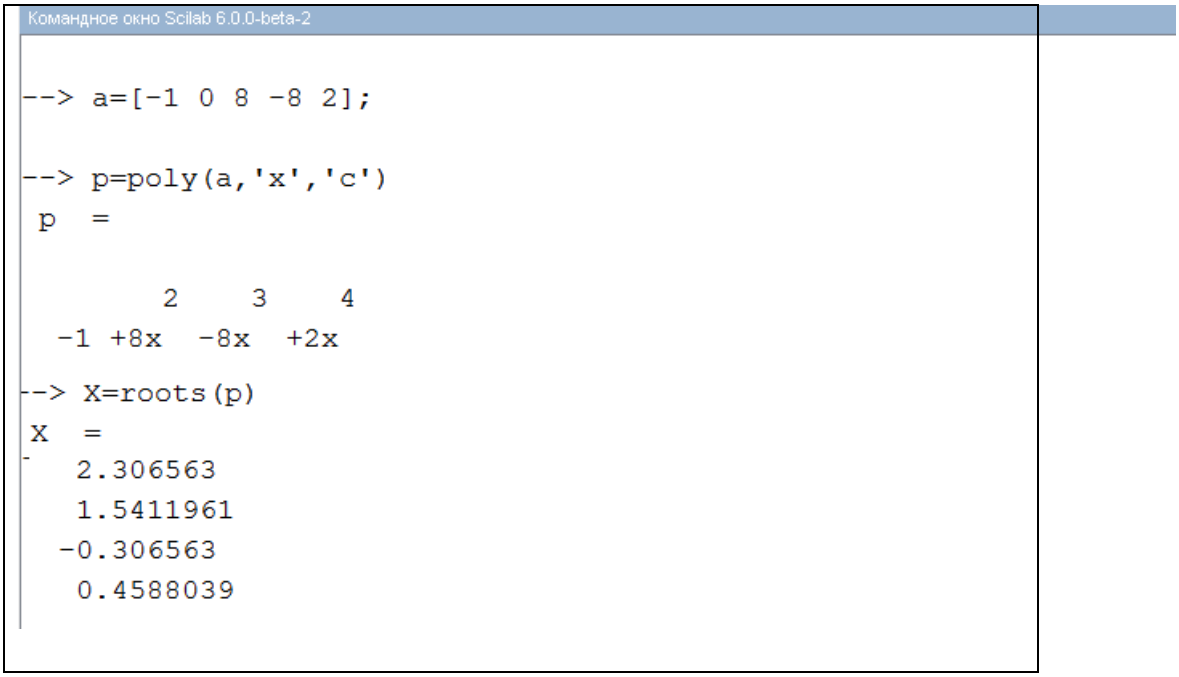

Рис. 2.1.1-6. Уточнение корня уравнения с использованием функции **roots()**

Вычисление корней полинома $P_6(x) = x^6 - 10x^5 + 16x^4 - x^2 + 10x - 16$ с использованием команды **roots(a)** показало, что у полинома есть четыре действительных корня.

2.1.2. Лабораторная работа по теме «Технология решения нелинейных уравнений средствами пакета Scilab»

1. Вопросы, подлежащие изучению

1) Задание функций и их производных с использованием средств пакета **Scilab**.
2) Получение таблиц значений функций в заданных границах изменения аргумента.
3) Построение графиков функций средствами **Scilab**.
4) Этапы решение нелинейных уравнений: отделение и уточнение корня.
5) Решение нелинейных уравнений с использованием встроенных функций пакета **Scilab**: **fsolve(), poly()** и **roots()**.

2. Общее задание

1) ***Изучите материал Темы 2.1 (п. 2.1.1).***
2) ***Выберите индивидуальный вариант*** *задания из* ***табл. 2.1.2-1***.
3) ***Отделите корень*** *нелинейного уравнения* ***f(x)=0*** *с использованием средств пакета Scilab, для чего:*
   - ***Построить графики*** *функции* ***f(x)*** *и ее первой производной;*
   - *на выбранном отрезке пересечения графика с осью* ***OX получить таблицы*** *значений функции* ***f(x)*** *и ее первой производной*
   - ***проверить условие*** *существования единственного корня на выбранном отрезке.*
4) ***Решите нелинейное уравнение*** *с использованием функций* ***fsolve() .***
5) ***Предъявите*** *результаты выполнение задания на ПК преподавателю.*

3. Варианты индивидуальных заданий

Таблица 2.1.2-1

№	Уравнение	№	Уравнение
1	$x^2 - 6\ln(x+1) + 2{,}8 = 0$	16	$2x^2 - \sin x - 1{,}5 = 0$
2	$x^2 - 4\sin x + 0{,}8 = 0$	17	$2\cos(x - x^3) - x = 0$
3	$x^2 - 3\sin x - 2{,}4$	18	$x^2 - 3{,}4\ln(1+x) - 1 = 0$
4	$x^2 + \sin x - 0{,}5\sqrt[3]{(1+2x)} = 0$	19	$e^x - (x+1)^2 - 2 = 0$
5	$3x^2 - 2\cos(x+1) - 4{,}8 = 0$	20	$x^2 - \sin x = 0$
6	$x^2 - 1{,}5\cos(x+3) - 1{,}8 = 0$	21	$e^x - 4x - 9{,}5 = 0$

7	$\ln(x+2)+\cos(2+x)-0{,}35=0$	22	$x^2-3\sin x-2x=0$
8	$x^3-4{,}8\sin(1+2x)-1=0$	23	$3-2\sin x-\sqrt{x^2+1=0}$
9	$x^2-2\sin(x+1)-2=0$	24	$-(2+3\sin x-\sqrt{x^2+2}=0$
10	$e^{x-3}-\ln(x+3)+1=0$	25	$e^{-0{,}4x}+3x^2-5x-3=0$
11	$-\sin x-\sqrt{x}+1=0$	26	$x^2+\cos(x-4)+\sqrt{x+1}-3=0$
12	$\cos(x-0{,}2x^2)-x+6=0$	27	$x^2-\ln x-3=0$
13	$e^x-4x=0$	28	$-x+e^{0{,}5x}=0$
14	$0{,}1x^2+x\ln x-x+0{,}5=0$	29	$-\sin x-\sqrt{1+x}+2=0$
15	$2e^{-x}+e^x-4=0$	30	$4x^3+2x^4+9{,}5e^{-2x}-35=0$

4. Содержание отчета

1) В форме комментариев:
   - Название лабораторной работы
   - ФИО студента, номер группы
   - № варианта
   - Индивидуальное задание
3) Протокол вычислений (сессии) в окне **Командного окна**, снабженный необходимыми комментариями.

2.1.3. Контрольные вопросы по теме

1) Что называется, нелинейным уравнением?
2) Этапы решения нелинейного уравнения.
3) Графическое отделение корней нелинейного уравнения средствами Scilab.
4) Аналитическое отделение корней нелинейного уравнения средствами Scilab.
5) Способы задания функции нелинейного уравнения.
6) Назначение и формат функции **poly().**
7) Назначение и формат функции f**solve().**
8) Назначение и формат функции **roots().**

Тема 2.2. Технология аппроксимации интерполяции функций в среде пакета MatLab

2.2.1. Аппроксимация и интерполяция функций
2.2.2. Лабораторная работа
2.2.3. Контрольные вопросы по теме

2.2.1. Аппроксимация и интерполяция функций

Пусть имеется набор узловых точек $\mathbf{x_k}$ (где **k=1,2,…,n**) и значения функции $\mathbf{y(x_k)}$в этих точках, а также некоторая функция $\mathbf{f(x,a_1, a_2, …,a_m)}$, которая кроме аргумента **x** зависит еще и от параметров $\mathbf{a_s}$(где **s=1, 2, …m**). Задача аппроксимации состоит в том, чтобы подобрать такие значения параметров $\mathbf{a_s}$, что функция $\mathbf{f(x, a_1, a_2, …a_m)}$наилучшим образом описывала бы исходную функцию. Как правило, **m<<n**, поэтому добиться, чтобы функция $\mathbf{f(x, a_1, a_2, …a_m)}$ давала точные результаты даже в узловых точках не удастся. Нужен критерий, который оценивает точность аппроксимации таблично заданной функции. Например, в методе наименьших квадратов в качестве такого критерия используется **среднеквадратическое отклонение**

$$\sqrt{\frac{\sum_{i=0}^{m}(y_i - f(x_i))^2}{m+1}} < \varepsilon$$

Частным случаем задачи аппроксимации является задача **интерполяции** функции. В этом случае также имеем набор узловых точек $\mathbf{x_k}$ (где **k=1,2,…,n**) и значения функции $\mathbf{y_k}$ в этих точках. Однако, в соответствии с критерием интерполяции, требуется построить такую функцию **f(x),** которая в узловых точках $\mathbf{x_1, x_2,…,x_n}$принимала бы значения $\mathbf{y_1, y_2, …,y_n}$, то есть $\mathbf{f(x_k) = y_k}$для всех **k**от **1** до**n**.Чаще всего функцию **f(x)** ищут в виде полинома, степень которого **n-1**. Поэтому задача сводится к определению коэффициентов интерполяционного полинома на основании значений функции в базовых точках.

На практике для решения задачи интерполяции (вычисления значений функции в точках, несовпадающих с узлами интерполяции) используются **интерполяционные формулы Ньютона** и **формула Лагранжа.**

Для выполнения полиномиальной аппроксимации в Scilab используется функция

[a,S]=datafit(f,z,a0),

где **f** - функция, с помощью которой можно вычислить разницу между заданными и расчетными значениями (y-f(x,a_0, a_1, …a_k);

z - матрица исходных данных (первая строка - массив x, вторая строка –y);

a0 - вектор начальных коэффициентов (нулевой вектор из k элементов);

a – вектор коэффициентов, используемых в аппроксимирующей функции) (a0, a1, …ak);

S - сумма квадратов отклонений измеренных значений от расчетных данных.

Функция **datafit(f,z,a0)**возвращает вектор коэффициентов полинома степени **k**, который с наименьшей среднеквадратичной погрешностью аппроксимирует функцию, заданную таблично. Результатом является вектор строка длиной **k+1**, содержащий коэффициенты полинома в порядке увеличения степеней. Как правило, степень полинома много меньше количества узлов(k<<**n**).

На рис.2.2.1-1 приведен пример линейной аппроксимации функции, заданной таблицей. Функция R(a,z) рассчитывает разности между исходной и аппроксимирующей функцией($y-(a_1+a_2x)=y-a_1-a_2x$).

```
Командное окно Scilab 6.0.0-beta-2
--> x=[1.32 1.4 1.5 1.62 1.7 1.8 1.9 2 2.11 2.2 2.32 2.24 2.51];
--> y=[3.3 3.5 3.85 4.25 4.5 4.85 5.4 6 6.6 7.3 9.2 10.2 13.5];
--> z=[x;y]; //матрица исходных данных
--> a=[0;0]; //вектор начальных приближений
--> function [zr]=R(a,z)
  > zr=z(2)-a(1)-a(2)*z(1)
  > endfunction
--> //линейная аппроксимация
--> [a,err]=datafit(R,z,a)
 err  =
    17.371602
 a   =
   -7.6955977
    7.412379
--> t=1.3:0.1:2.52;
--> y1=a(1)+a(2)*t;
--> plot(x,y,'o');
--> mtlb_hold('on');
--> plot(t,y1,'b-.');
```

Рис. 2.2.1-1. Пример линейной аппроксимации функции

Добавим необходимые команды для кубической аппроксимации (рис.2.2.1-2). Здесь разности между исходной и аппроксимирующей функцией вычисляются функцией R3(c,z)

$$y-(c_1+c_2x+c_3x^2+c_4x^3)=y-c_1-c_2x-c_3x^2-c_4x^3.$$

```
Командное окно Scilab 6.0.0-beta-2
--> function [zr]=R3(c,z)
  > zr=z(2)-c(1)-c(2)*z(1)-c(3)*z(1)^2-c(4)*z(1)^3
  > endfunction
--> c=[0;0;0;0];
--> [c,err]=datafit(R3,z,c)
 err  =
    3.7248294
 c  =
   -24.698932
    50.938895
   -31.731649
    6.9729286
--> y3=c(1)+c(2)*t+c(3)*t^2+c(4)*t^3;
--> plot(t,y3,'r-')
```

На рис.2.2.1-3 приведены графики исходной и аппроксимирующих функций.

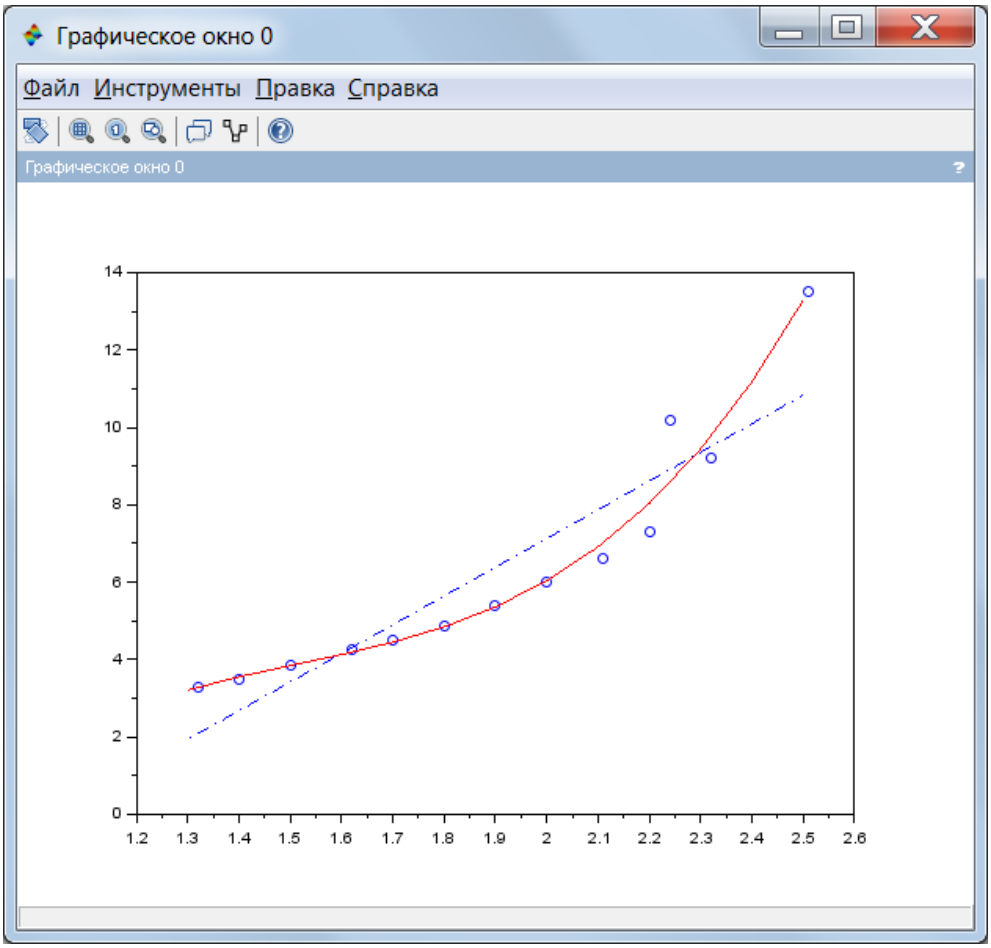

Рис. 2.2.1-3. Графики исходной и аппроксимирующих функций

Чтобы использовать функцию **datafit()** для решения задачи интерполяции необходимо, чтобы степень этого полинома была на единицу меньше количества узловых точек.

Пример 2.2.1-1. Используя в качестве узлов интерполяции x=1,2,4,5, построить полином, интерполирующий функцию $y(x) = Sin(x)$.

В начале, получим таблицу значений функции и проведем ее интерполяцию ее с использованием функции **datafit()** (рис. 2.2.1-5). Графики интерполяционных узлов и интерполяционного полинома приведены на рис. 2.2.1-6.

```
Командное окно Scilab 6.0.0-beta-2
--> x=[1 2 4 5];
--> y=sin(x)
 y  =
  0.841471   0.9092974  -0.7568025  -0.9589243
--> z=[0.841471 0.9092974 -0.7568025 -0.9589243];
--> d=[x;z];
--> c=[0;0;0;0];
--> function [zr]=R3(c,d)
  > zr=d(2)-c(1)-c(2)*d(1)-c(3)*d(1)^2-c(4)*d(1)^3
  > endfunction
--> [c,err]=datafit(R3,d,c)
 err  =
   6.223D-10
 c  =
  -0.8480361
   2.755657
  -1.1937864
   0.1276438
--> function [r]=ip3(t)
  > r=-0.8480361+2.755657*t-1.1937864*t^2+0.1276438*t^3;
  > endfunction
--> ip3(x)
 ans  =
    0.8414783   0.9092827  -0.7567873  -0.9589361
```

Рис. 2.2.1-5. Интерполяция таблично заданной функции с использованием функции Scilab **datafit()**

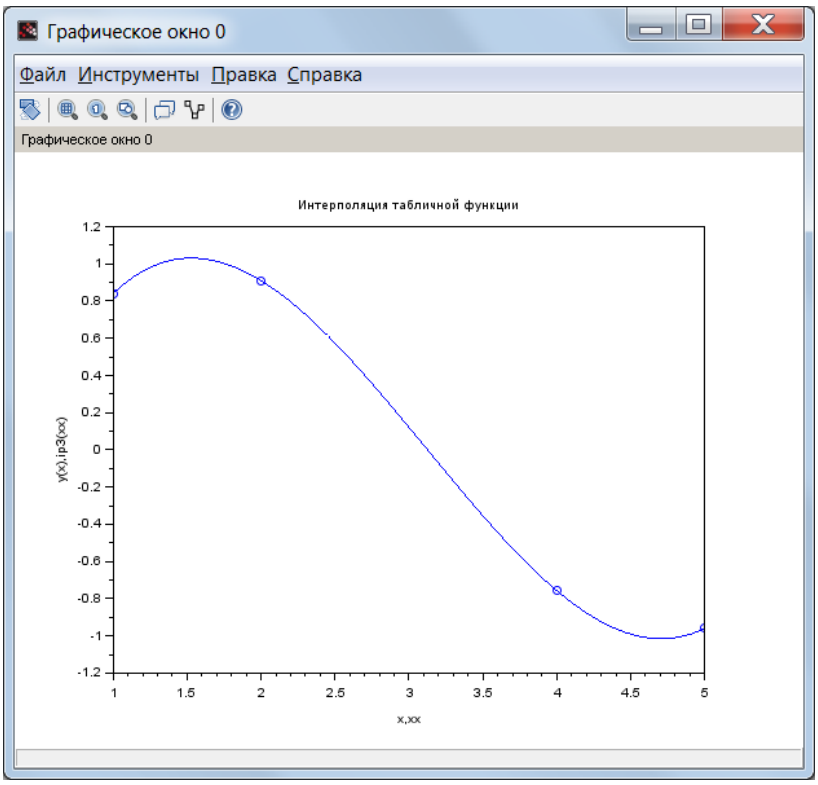

Рис. 2.2.1-6. Графики интерполируемой и интерполирующей функций

При большом количестве базовых точек интерполяции полиномом может оказаться малопродуктивной, поэтому нередко используют интерполяцию ***сплайнами***. Идея ***сплайн-интерполяции*** состоит в разбиении диапазона интерполирования на отрезки, в пределах которых используются разные функции одного вида (чаще всего алгебраические многочлены). Эта функция и ее несколько производных на всем диапазоне интерполяции непрерывны. В результате имеем кусочно-гладкую интерполяционную зависимость.

Один из наиболее распространенных вариантов интерполяции интерполяция кубическими сплайнами . Кроме того существуют

квадратичные и линейные сплайны.

В Scilab для построения линейной интерполяции служит функция

y=interpln(z,x),

где **z** - матрица исходных данных;

x - вектор абсцисс;

y - вектор значений линейного сплайна в точка **x.**

Рассмотрим пример использования функции **interpln().**

Пример 2.2.1-2.Используя значения интерполирующей функции, заданной таблично, и выполнить сплайн-интерполяцию с использованием линейных сплайнов и получить значения функции в точке x=0.58.

x	-1	0	1	2
y(x)	4	2	0	1

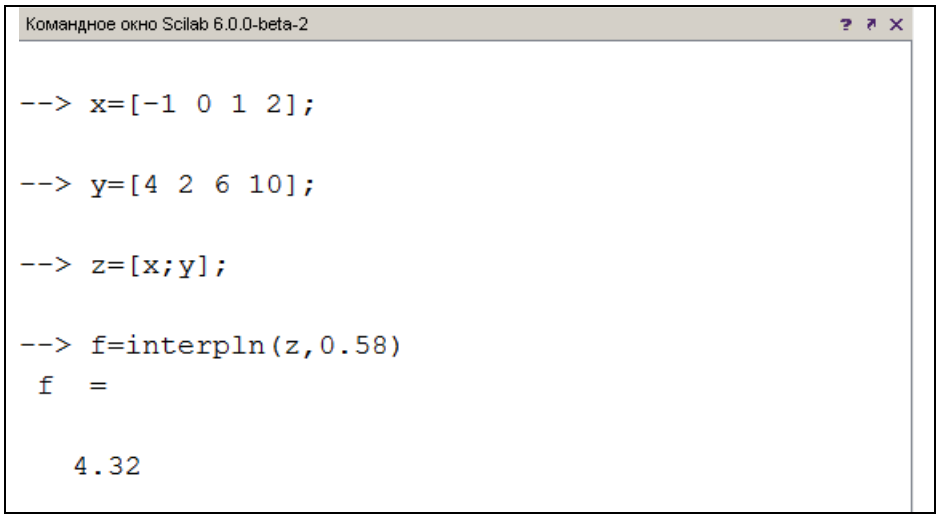

Рис. 2.2.1-7. Вычисление функции в точке **x=0.58**

Построение кубического сплайна в Scilab состоит из двух этапов: вначале вычисляются коэффициенты сплайна с помощью функции **d=splin(x,y)**, а затем рассчитывается значения интерполяционного полинома в точке **y=interp(t,x,y,d).**

Функция **d=splin(x,y)** имеет следующие параметры:

x - строго возрастающий вектор, состоящий минимум из двух компонент;

y- вектор того же формата, что и **x**;

d - результат работы функции, т.е. коэффициенты кубического сплайна.

Для функции **y=interp(t,x,y,k)** параметры **x**, **y** и **d** имеют те же значения, параметр **t** - это вектор абсцисс, а **y** - вектор ординат, являющихся значениями кубического сплайна в точках x.

Пример 2.2.1-3.Найти приближенное значение функции при заданном значении аргумента с помощью интерполяции кубическими сплайнами в точках x1 = 0,702, x2 = 0,512, x3 = 0,608.

x	0.43	0.48	0.56	0.62	0.7	0.75
y(x)	1.635	1.732	1.876	2.033	2.228	2.359

На рис.2.2.1-8 и рис.2.2.1-9 приведены команды, необходимые для проведения интерполяции таблично заданной функции и построение графиков интерполируемой функции и интерполирующих ее **с использованием кубических сплайнов**.

```
Командное окно Scilab 6.0.0-beta-2

--> x=[0.43 0.48 0.55 0.62 0.7 0.75];
--> y=[1.635 1.732 1.876 2.033 2.228 2.235];
--> koeff=splin(x,y);
--> X=[0.702 0.512 0.608];
--> //значения функции в заданных точках
--> Y=interp(X,x,y,koeff)
 Y  =
   2.2309267   1.7972605   2.0028862

--> t=0.43:0.01:0.75;
--> p=interp(t,x,y,koeff);
--> plot(x,y,'o');
--> mtlb_hold('on');
--> plot(t,p,'-');
```

Рис. 2.2.2-8. Команды построения графиков интерполяционных функций

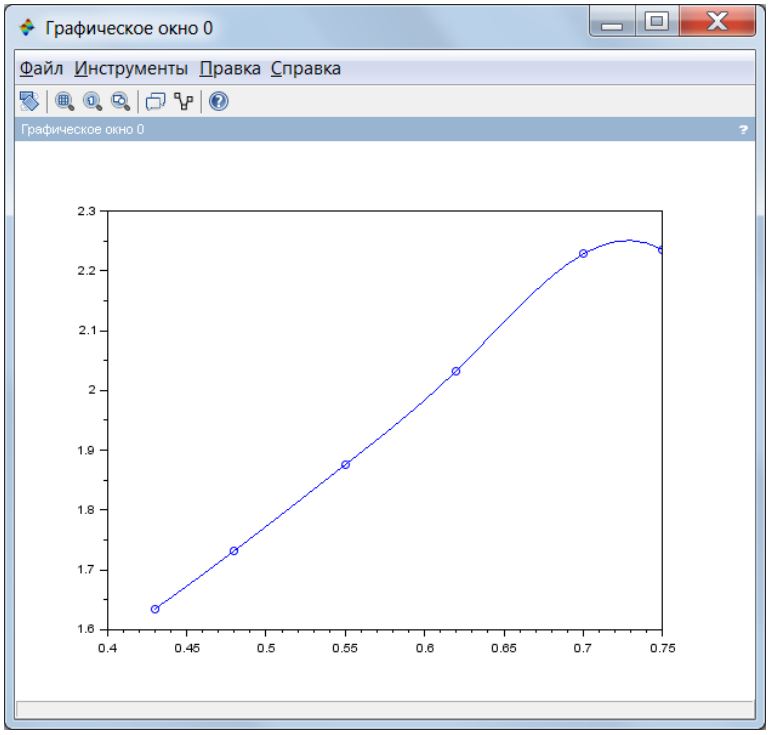

Рис. 2.2.2-9. График функции, полученной с использованием кубических сплайнов

2.2.2. Лабораторная работа по теме «Технология аппроксимации и интерполяции функций»

1. Вопросы, подлежащие изучению

1) Задание векторов и матриц в пакете Scilab.
2) Технология аппроксимации функции, заданной таблично, с использованием функций **datafit**().
3) Технология линейной, кубической и сплайн-интерполяции таблично заданной функции с использованием функций **interpln().**
4) Получение интерполяционных многочленов в явном виде.
5) Построение графиков аппроксимирующих и интерполирующих функций.

2. Общее задание

1) ***Изучите материал Темы 2.2. (п. 2.2.1).***
2) ***Выберите индивидуальное задание****: номера узлов и номер аппроксимируемой функции из* ***табл. 2.2.2-1****; узлы аппроксимации и значения функции в узлах из* ***табл. 2.2.2-2****.*
3) ***Задайте*** *в виде векторов значения узлов и значения функции в выбранных узлах;*
4) ***Вычислите коэффициенты*** *аппроксимирующих функции для линейной, квадратичной и кубической аппроксимации с использованием функции* **datafit***( )и* ***получите*** *три аппроксимирующие функции в явном виде.*
5) ***Получите*** *с использованием этих функций значение аппроксимирующей функции в произвольной точке, принадлежащей отрезку, но не совпадающей с узлами аппроксимации, и* ***сравните полученные*** *результаты.*
6) ***Постройте графики*** *табличной и трех аппроксимирующих функций в одном шаблоне, снабдив их легендой.*
7) ***Проведите*** *линейную и кубическую интерполяцию функции с использованием функции* ***interpln()****, заданной таблично. Получив значения интерполирующей функций в точке, не совпадающей с узлами интерполяции и* ***сравните*** *полученные результаты.*
8) ***Постройте*** *графики табличной и двух интерполирующих функций в одном шаблоне, снабдив их легендой.*
9) ***Предоставьте*** *результаты работы преподавателю,* ***ответить*** *на поставленные вопросы.*

10) ***Выполните*** *команду* ***clear all***.

11) ***Оформите отчет*** *по выполненной работе****.***

3. Варианты заданий

Таблица 2.2.2-1

Вариант №	**Номера узлов x_i**	**Номер функции**
1	1 3 5 7 9 10 13	$f1(x)$
2	1 2 4 5 7 10 12	$f1(x)$
3	1 3 6 7 10 11 13	$f1(x)$
4	1 2 4 7 9 11 13	$f1(x)$
5	3 6 7 9 10 11 12	$f1(x)$
6	2 3 6 8 9 10 13	$f1(x)$
7	1 4 5 7 9 11 12	$f1(x)$
8	1 2 4 7 9 12 13	$f1(x)$
9	2 3 5 7 8 11 12	$f1(x)$
10	1 3 6 7 9 10 13	$f1(x)$
11	1 3 7 8 10 11 13	$f2(x)$
12	1 2 5 6 7 10 12	$f2(x)$
13	1 4 5 8 10 12 13	$f2(x)$
14	1 3 5 7 9 10 13	$f2(x)$
15	1 3 6 7 8 10 13	$f2(x)$
16	1 4 5 7 9 11 12	$f2(x)$
17	2 4 5 6 8 12 13	$f2(x)$
18	1 4 5 7 9 11 12	$f2(x)$
19	1 4 5 8 10 11 12	$f2(x)$
20	2 4 5 6 8 12 13	$f2(x)$
21	1 4 5 810 12 13	$f3(x)$
22	2 3 6 8 9 10 13	$f3(x)$
23	1 3 5 8 10 12 13	$f3(x)$
24	1 4 5 7 9 11 12	$f3(x)$
25	2 4 5 6 8 12 13	$f3(x)$
26	3 4 5 7 8 9 12	$f3(x)$
27	3 5 8 10 11 12 13	$f3(x)$
28	2 4 7 9 10 11 13	$f3(x)$
29	2 4 5 7 8 10 12	$f3(x)$
30	1 4 5 7 9 11 13	$f3(x)$

Таблица 2.2.2-2

i	$\mathbf{x_i}$	$f1(x_i)$	$f2(x_i)$	$f3(x_i)$
1	-5	1.38	2.44	1.676
2	-4.5	1.221	2.359	2.025
3	-4	1.511	1.751	1.736
4	-3.5	1.501	2.13	1.203
5	-3	1	1.455	1.511
6	-2.5	0.728	1.482	1.362
7	-2	0.976	1.437	0.75
8	-1.5	1.065	0.803	0.976
9	-1	0.599	1.175	0.957
10	-0.5	0.192	0.49	0.272
11	0	0.3	0.375	0.3
12	0.5	0.319	$-6.51*10^{-3}$	0.165
13	1	-0.405	-1.965	-1.185

4. Содержание отчета

1) В форме комментариев:
   - Название лабораторной работы
   - ФИО студента, номер группы
   - № варианта
   - Индивидуальное задание
2) Протокол вычислений (сессии) в Командном окне, снабженный необходимыми комментариями.

2.2.3. Контрольные вопросы по теме

1) Что такое аппроксимация функции, и в каких случаях она используется?
2) В чем отличие аппроксимации от интерполяции?
3) Какой метод аппроксимации реализован в функции **datafit()?**
4) Что служит результатом выполнения функции **datafit()?**
5) Назначение и формат функции **interpln()?**
6) Каким параметром определяется тип интерполяции в функции **interpln()?**

Тема 2.3. Технология интегрирования в среде Scilab

2.3.1.Вычисление неопределенных и определенных интегралов
2.3.2. Лабораторная работа
2.3.3. Контрольные вопросы по теме

2.3.1. Вычисление неопределенных и определенных интегралов

При вычислении определенных интегралов первообразную функцию **F(x)**не всегда удается выразить аналитически, а кроме того иногда подынтегральная функция **f(x)** задана в виде таблицы (**x_i и y_i, где i= 1, 2, …, n**).Это приводит к необходимости использования численных методов интегрирования.

Существует ряд методов численного интегрирования. Во всех этих методах вычисление осуществляется по приближенным формулам, называемым **квадратурами**.

Суть получения формул численного интегрирования состоит в том, что на элементарных отрезках интегрирования подынтегральную функцию заменяют простейшим интерполяционным полиномом, который легко может быть проинтегрирован в аналитическом виде. Так, например, для получения формул **прямоугольников, трапеций** и **Симпсона** используют полиномы соответственно нулевой, первой и второй степени.

Формулы прямоугольников:

$$\int_a^b f(x)dx = \left\{ \begin{array}{l} h\sum_{i=0}^{n-1} y_i \\ h\sum_{i=1}^{n} y_k \end{array} \right\},$$

где:**h**–шаг интегрирования;
y_i – значение подынтегральной функции от аргумента **x_i, k=0, 1, …,n**;
n–число разбиений интервала интегрирования **a**, **b**.

Формула трапеций:

$$\int_a^b f(x)dx = \frac{h}{2}\left(y_0 + 2\cdot\sum_{i=1}^{n-1} y_i + y_n \right),$$

где:**h**– шаг интегрирования;
y_0 – значение подынтегральной функции при **x = a**;
y_n – значение подынтегральной функции при **x = b**;

Формула Симпсона:

$$\int_a^b f(x)dx = \frac{h}{3}\left(y_0 + 4y_1 + 2y_2 + \ldots 2y_{n-2} + 4y_{n-1} + y_n\right),$$

Для вычисления интеграла $\int_a^b y(x)\partial x$ по **формуле трапеции** в Scilab используется функция **inttrap([x],y)**. Эта функция вычисляет площадь фигуры под графиком функции y(x), которая описана набором точек (x, y). Простейший пример использования функции **inttrap()** приведен на рис.2.3.1-1.

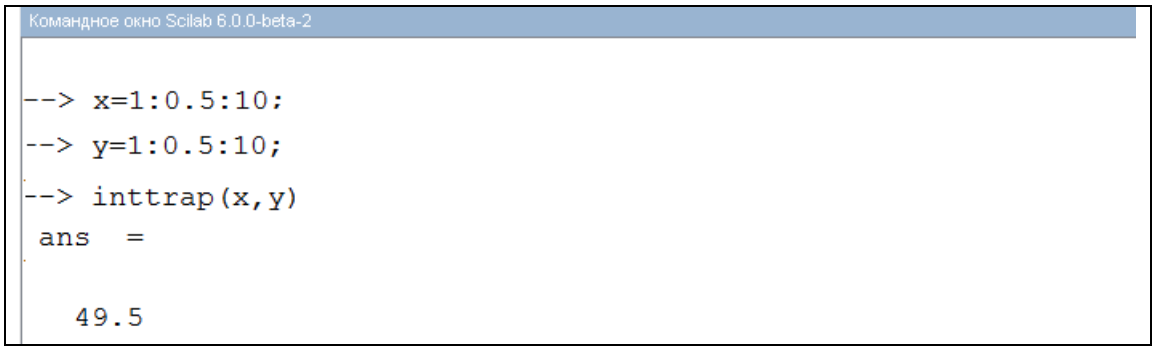

Рис.2.3.1-1. Вычисление интеграла по формуле трапеций

Параметр **x** является необязательным. Для формата функции **inttrap(y)** элементы вектора x могут принимать значения номеров вектора**y**.

Рассмотрим несколько примеров вычисления значений определенных интегралов методом трапеций (рис. 2.3.1-2). В последнем примере использован формат **inttrap(y)**, то есть значения **x** заменены номерами элементов вектора **y**. В результате получено другое значение интеграла. Это показывает, что получена площадь совершенно другой фигуры.

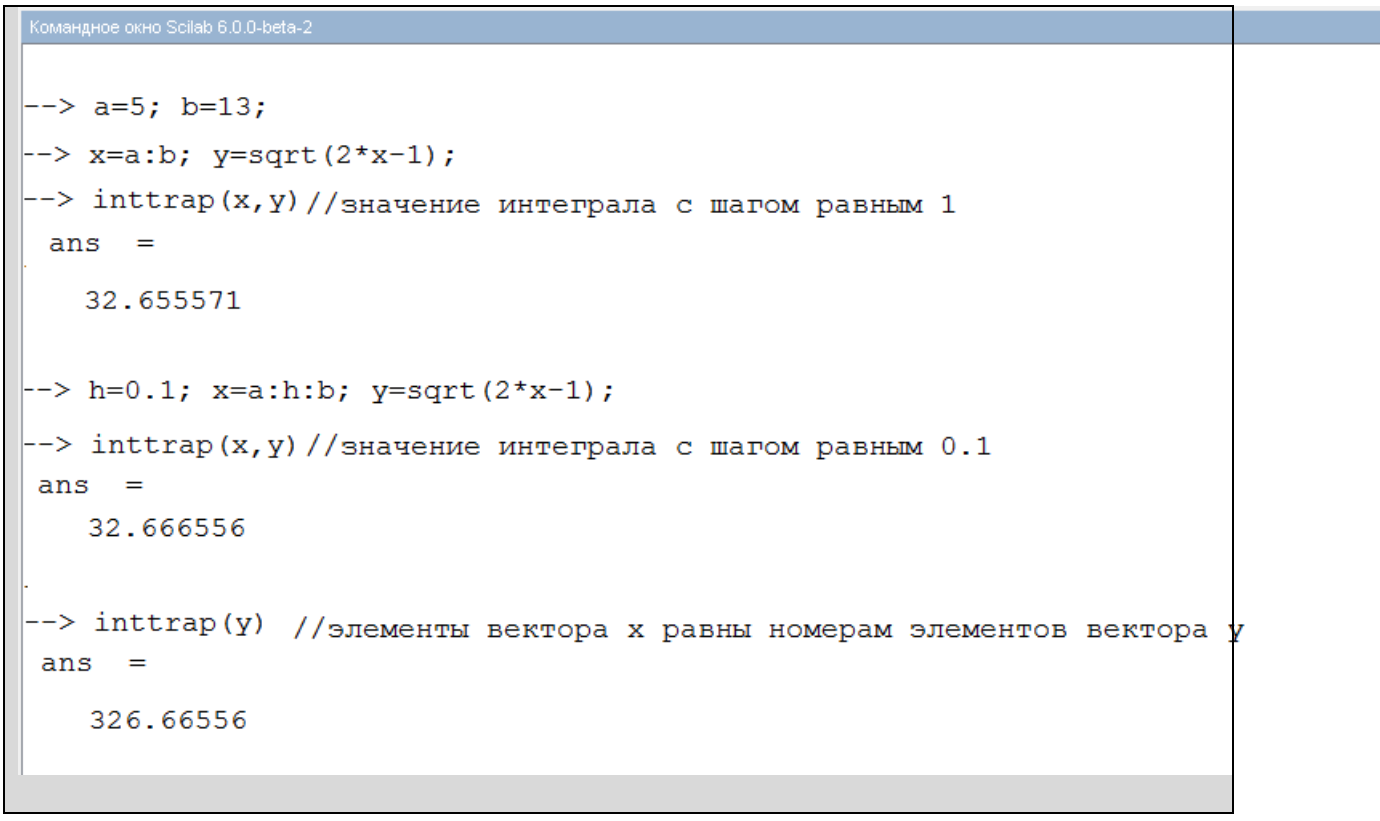

Рис.2.3.1-2. Примеры вычисления определенных интегралов с использованием функции **inttrap()**

Для вычисления интеграла по **формуле Симпсона** в Scilab применяется функция **integrate**() имеющей формат

integrate(fun, x, a, b, [,er1 [,er2]])

где **fun**- функция, задающая подынтегральное выражение в символьном виде;
x- переменная интегрирования, так же задается в виде символа;

a, b- пределы интегрирования, действительные числа;
er1 и **er2**– переменные, задающие абсолютную или относительную точность вычислений (действительные числа).

При обращении к этой функции шаг интегрирования не задается, а используется параметр – требуемая точность вычисления интеграла (рис.2.3.1-3). Если погрешность вычисления интеграла отсутствует, то вычисление проводится с погрешностью, установленной по умолчанию.

.

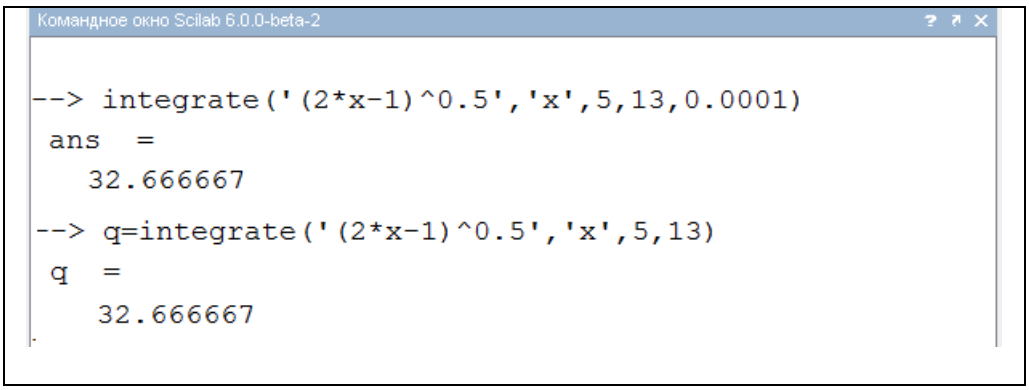

Рис.2.3.1-3. Вычисление определенных интегралов
с использованием функции **integrate()**

2.3.2. Лабораторная работа по теме «Технология интегрирования в среде scilab»

1. Вопросы, подлежащие изучению

1) Вычисление значения определенного интеграла с использованием функций.
2) Вычисление значения определенного интеграла с использованием функций Scilab **inttrap()** и **integrate**().
3) Формулы численного интегрирования: средних прямоугольников, трапеций и Симпсона.

2. Общее задание

1) ***Изучите*** *материал* ***Темы 2.3 (п. 2.3.1.).***
2) ***Выберите*** *из* ***табл. 2.3.2-1*** *вариант индивидуального задания.*
3) ***Вычислите*** *определенный интеграл с использованием функции*Scilab*, реализующей формулу трапеций, -* ***inttrap(x, y),*** *предварительно получив таблицу значений подынтегральной функции с шагом* h*, и задав значения аргумента и функции в виде векторов.*
4) ***Вычислите*** *значение определенного интеграла с использованием функции* Scilab*, реализующей формулу Симпсона, –* **integrate *('f',a,b),*** *где f– имя функции, взятое в одинарные кавычки.*
5) ***Предоставьте*** *результаты работы преподавателю,* ***ответьте*** *на поставленные вопросы.*
6) ***Выполните*** *команду* ***clear all.***
7) ***Оформите отчет*** *по выполненной работе****.***

1. Варианты индивидуальных заданий

Таблица 2.3.2-1

№	Интеграл 1	Интеграл 2
1	$\int\left(Cos(x)+e^{x}\right)dx$	$\int_{0}^{4}\left(8e^{-x}Sin(-2x)\right)dx$
2	$\int\frac{x}{1+x^{4}}\partial x$	$\int_{1}^{3}\left(Cos(x/5)(1+x)^{1/2}-x\right)\partial x$
3	$\int\frac{1}{x(1+x^{2})}\partial x$	$\int_{3.5}^{5}\left(5\ln(x)-x/2\right)\partial x$
4	$\int\frac{1}{\sqrt{(2+x)^{2}}}\partial x$	$\int_{0.5}^{1,5}\left(x^{2}+\ln(x)-2\right)\partial x$
5	$\int(-8Sin(x)+e^{-x})\partial x$	$\int_{1.5}^{2.5}\left(1+Sin(4x)/\ln(x)\right)\partial x$

6	$\int (Cos(x^2+3x))\partial x$	$\int_2^{3.5}\left(Sin(x^2)+Cos^2(x)-10x\right)\partial x$
7	$\int \frac{x^2}{\sqrt{3+2x}}\partial x$	$\int_{0.5}^{3}\left(3Cos(x^2)/\ln(x+3)\right)\partial x$
8	$\int (Sin(x^{-2})+x)\partial x$	$\int_1^{2.5}\left(x+Sin(x)/e^x\right)\partial x$
9	$\int (x^2+Cos^2(x+3)-x)\partial x$	$\int_2^{3.5}\left(Cos^2(x)-Sin(x)+x\right)\partial x$
10	$\int \frac{x}{1-x^2}\partial x$	$\int_{-1}^{0}\frac{x}{\sqrt{1-x}}\partial x$
11	$\int \frac{1}{x\sqrt{2+x^2}}\partial x$	$\int_0^{1.25}\frac{x^2}{\sqrt{1+x^2}}\partial x$
12	$\int (Sin(x)Cox^2(x)\partial x$	$\int_0^{1}\frac{x^2}{\sqrt{1+x}}\partial x$
13	$\int \frac{1}{Sin(x)}\partial x$	$\int_0^{1.2} xCos(x)\partial x$
14	$\int \frac{1}{Cos(x)}\partial x$	$\int_1^{2.5}\left(x+Sin(x)/x^2\right)\partial x$
15	$\int \frac{1}{Sin^2(x)}\partial x$	$\int_{0.6}^{1.8}\frac{x}{\sqrt{0.8+x^2}}\partial x$
16	$\int \frac{x}{x^2(x-1)}\partial x$	$\int_{0.5}^{1.5}\frac{1}{\sqrt{x^2+2}}\partial x$
17	$\int (x+Cox^2(x)\partial x$	$\int_1^{2.5}\left(Cos(x)/e^x\right)\partial x$
18	$\int \frac{x}{Cos(x)}\partial x$	$\int_{0.4}^{1.2}\frac{Cox(x^2)}{x+1}\partial x$
19	$\int \frac{1}{Sin(x)Cos(x)}\partial x$	$\int_{0.4}^{3.2} x^2\ln(x)\partial x$
20	$\int (xCox^2(x)\partial x$	$\int_{0.2}^{0.8}\sqrt{x+1}Cos(x)\partial x$
21	$\int (2x+Cox^2(2x)\partial x$	$\int_{0.7}^{2.1}\frac{Sin(x)}{x+1}\partial x$
22	$\int Cos(x)+\frac{3}{Sin(x)}\partial x$	$\int_1^{2.5}(x+1)Cos(x^2)\partial x$
23	$\int \frac{Sin(x)}{Cos^4(x)}\partial x$	$\int_{0.4}^{1.2}\sqrt{x}Cos(x^2)\partial x$
24	$\int \frac{x^2}{e^x}dx$	$\int_{0.7}^{2.1}\frac{Sin(2x)}{x}\partial x$
25	$\int \frac{x^2}{\sqrt{1+2x}}dx$	$\int_{0.4}^{1.2}\frac{\cos(x)}{x+2}dx$

26	$\int (2x+1)\cdot e^{3x}dx$	$\int_0^{\pi} \frac{1}{3+2\cos(x)}dx$
27	$\int \frac{1}{1+\sin^2(x)}dx$	$\int_{-1}^{1} \frac{1}{(1+x^2)^2}dx$
28	$\int \frac{1}{\sqrt{4x^2+2x+1}}dx$	$\int_1^5 \frac{\sqrt{x-1}}{x}dx$
29	$\int \frac{\sqrt{4-x^2}}{x^2}dx$	$\int_1^3 \frac{1}{x\sqrt{x^2+1}}dx$
30	$\int \cos^2(x)\sin^3(x)dx$	$\int_1^2 \frac{1-e^{2x}}{x\cdot e^{2x}}dx$

2. Содержание отчета

1) В форме комментариев:
   - Название лабораторной работы
   - ФИО студента, номер группы
   - № варианта
   - Индивидуальное задание
2) Протокол вычислений (сессии) в **Командном окне**, снабженный необходимыми комментариями**.**

2.3.3. Контрольные вопросы по теме

3) Какой функцией в **Scilab** определяется символьное значение определенного интеграла?
4) Назначение функции **inttrap(x,y)?**
5) Что возвращает функция **inttrap(x,y),** если **y(x)** – матрица?
6) Можно ли использовать функцию **inttrap(x,y),** если узлы по оси **x** - не равноотстоящие?
7) Какая функция Scilab позволяет вычислить определенный интеграл с заданной точностью?
8) Способы задания подынтегральной функции при вычислении определенного интеграла с использованием функции **integrate().**
9) Как задать точность вычисления определенного интеграла?

Тема 2.4. Технология решения обыкновенных дифференциальных уравнений

2.4.1. Численное решение обыкновенных дифференциальных уравнений
2.4.2. Лабораторная работа
2.4.3. Контрольные вопросы по теме

2.4.1. Численное решение решения обыкновенных дифференциальных уравнений

Представим обыкновенное дифференциальное уравнение (ОДУ) первого порядка в виде, разрешенном относительно производной $y' = f(x, y)$,и пусть $y(x_0) = y$ – начальные условия его решения.

Тогда решением ОДУ является функция $y = \varphi(x)$, которая будучи подставленной в исходное уравнение, обратит его в тождество, и одновременно будут выполняться начальные условия. Эта задача в математике называется ***задачей Коши***.

Задача Коши при решении ОДУ n-го порядка $y^{(n)} = f(x, y, y', ...y^{(n-1)})$ формулируется аналогично, при этом начальными условиями должны быть: $y(x_0) = y_0, y'(x_0) = y_0', ...\ \ y^{(n-1)}(x_0) = y_0^{(n-1)}$. При решении Оду n-го порядка уравнение путем выполнения ряда обозначений $(y' = y_1, y_1' = y_2, ...\ \ y_{n-2}' = y_{n-1})$ представляется в виде системы дифференциальных уравнений:

$$\begin{cases} y' = y_1; \\ y_1' = y_2; \\ ..... \\ y_{n-1}' = y_n; \\ y^{(n)} = f(x, y, y_1, ..., y_{n-1}). \end{cases}$$

Результатом решения **ОДУ** численными методами является ***таблица значений*** **y = φ(x)** на некотором множестве значений аргумента **x**. Поэтому при постановке задачи численного решения **ОДУ** наряду с начальными условиями $\mathbf{x_0}$, $\mathbf{y_0}$необходимо задать область решения – отрезок **[a;b]** и шаг изменения аргумента **h** (шаг интегрирования).

Для получения численного решения ОДУ используются **методы Рунге-Кутты.** Методы различаются порядком. Чем выше порядок метода, тем точнее решение, полученное при равном шаге интегрирования.

Для решения дифференциальных уравнений и систем в Sciab предусмотрена функция:

[y,w,iw]=ode([type],y0,t0,t [,rtol [,atol]],f [,jac] [,w,iw]),

для которой, ***обязательными*** входными параметрами являются:

y0 - вектор начальных условий;

t0 - начальная точка интервала интегрирования;

t - координаты узлов сетки, в которых происходит поиск решения;

f - внешняя функция, определяющая правую часть уравнения или системы уравнений;

y - вектор решений.

Таким образом, для того чтобы решить обыкновенное дифференциальное уравнение необходимо вызвать функцию **y=ode(y0,t0,t,f).**

Рассмотрим ***необязательные*** параметры функции **ode**:

type– параметр, с помощью которого можно выбрать метод решения или тип решаемой задачи, указав одну из строк: **adams** - применяют при решении дифференциальных уравнений или систем методом прогноза - коррекции Адамса;

stiff - указывают при решении жестких задач;

rk - используют при решении дифференциальных уравнений или систем методом Рунге-Кутты четвертого порядка;

rkf - указывают при выборе пятиэтапного метода Рунге-Кутты четвертого порядка;

fix - тот же метод Рунге-Кутты, но с фиксированным шагом;

rtol, atol - относительная и абсолютная погрешности вычислений, вектор, размерность которого совпадает с размерностью вектора y, по умолчанию **rtol=0.00001**, **atol=0.0000001** (при использовании параметров **rkf** и **fix** - **rtol=0.001, atol=0.0001**);

jac- матрица, представляющая собой якобиан правой части жесткой системы дифференциальных уравнений, задают матрицу в виде внешней функции вида **J=jak(t,y)**;

w, iw - векторы, предназначенные для сохранения информации о параметрах интегрирования, которые применяют для того, чтобы последующие вычисления выполнялись с теми же параметрами.

Рассмотрим использование функции **ode()** на примере решения следующей задачи.

Пример 2.4.1-1. Найти решение ОДУ $y' = -\sin(x \cdot y)$ **на отрезке [0;5] сшагом 0.5 при начальных условиях** $y(0) = 1.5;$

```
Командное окно Scilab 6.0.0-beta-2

--> function yd=f(t,x), yd=-sin(x*t), endfunction;
--> x0=1.5;t0=0;t=0:0.5:5;
--> y=ode(x0,t0,t,f);
--> plot(t,y,'-o')
 -> xgrid();
--> yy=[t;y];
--> yy'
 ans  =

    0.    1.5
    0.5   1.3302706
    1.    0.9566472
    1.5   0.5574285
    2.    0.2477507
    2.5   0.0822207
    3.    0.0208664
    3.5   0.0041103
    4.    0.0006303
    4.5   0.0000753
    5.    0.000007
```

Рис. 2.4.1-1. Решение ОДУ с использованием функции **ode()**

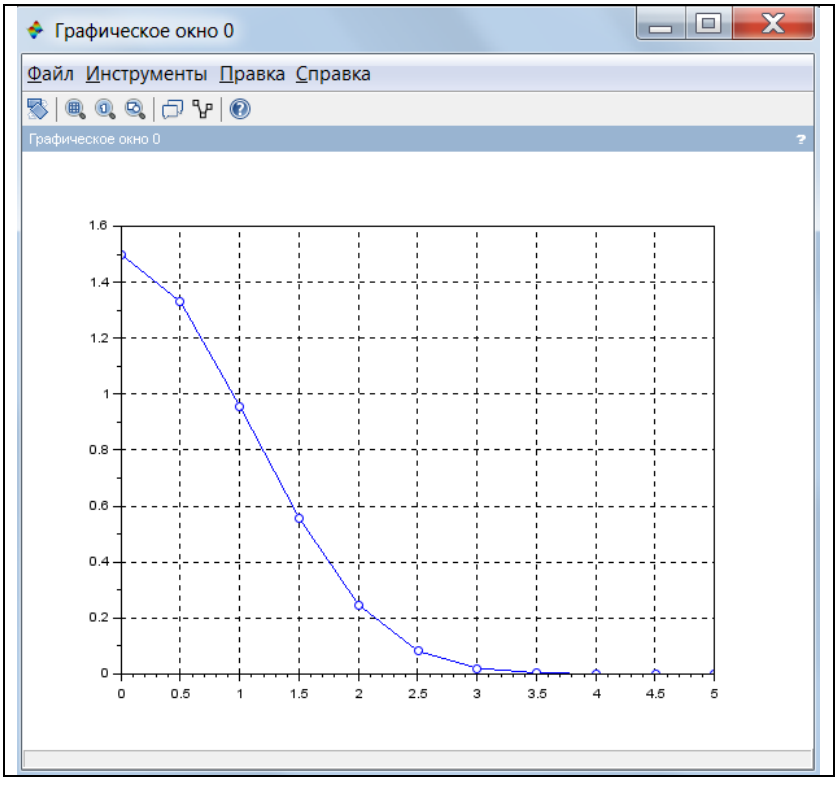

Рис. 2.4.1-2. Графическое решение ОДУ

Пример 2.4.1-2.Решить систему ОДУ

$$\begin{cases} x' = \cos(xy); \\ y' = \sin(x) + ty; \end{cases}$$
$$x(0) = 0; y(0) = 0.$$

Решение системы ОДУ в Scilab начинается с функции, описывающей систему (рис.2.4.1-3).

Для решения дифференциального уравнения (системы уравнений) используем команды, где решение системы ОДУ записывается в переменную **y**. При этом элемент **y(1)** содержит значение функции**y(x),** а **y(2)** значение производной **y'(x)**. Команда **plot(x, y)** строит график функции решения ОДУ и ее производной (рис. 2.4.1-6).

Командное окно Scilab 5.5.2

```
-->function dy=syst(t,y)
-->dy=zeros(2,1);
-->dy(1)=cos(y(1)*y(2));
-->dy(2)=sin(y(1)+y(2)*t);
-->endfunction
-->//Решение системы ДУ
-->x0=[0;0];t0=0;t=0:1:10;y=ode(x0,t0,t,syst);
-->//Формирование графического решения
-->plot(t,y)
-->xgrid();
-->z=[t;y];
-->z'
 ans  =

   0.      0.           0.
   1.      0.9802401    0.533358
   2.      1.4096497    0.9693978
   3.      1.7429464    0.6024417
   4.      2.4027415    0.2586293
   5.      3.3312751    0.0057610
   6.      4.2504071  - 0.1650347
   7.      4.826261   - 0.2358589
   8.      5.1581587  - 0.2515654
   9.      5.3963437  - 0.2509281
   10.     5.5981318  - 0.2461951
```

Рис.2.4.1-4. Решение системы ОДУ

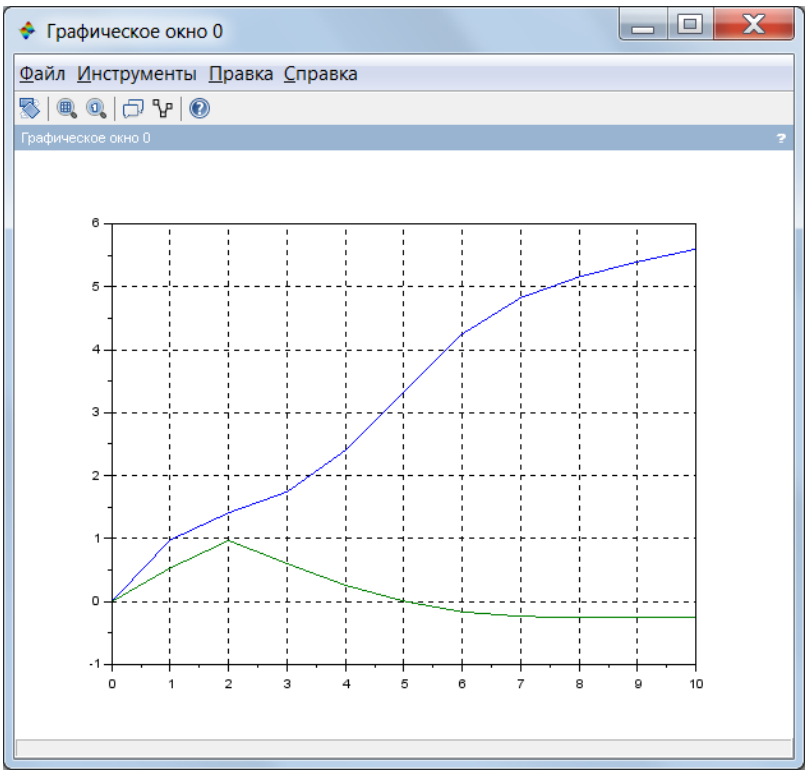

Рис.2.4.1-5. Графики функций решения ОДУ и производной

2.4.2. Лабораторная работа по теме «Технология решения обыкновенных дифференциальных уравнений средствами Scilab»

1. Вопросы, подлежащие изучению

1) Вычисление числовых значений производных в точках.
2) Функция **Scilab**, предназначенная для решения ОДУ **ode()**.
3) Вывод результатов решения ОДУ в виде таблицы.
4) Получение графического решения ОДУ.

2. Общее задание

1) ***Изучите*** *материал* ***Темы 2.4 (п. 2.4.1)****.*
2) ***Выберите*** *индивидуальный вариант задания из* ***табл. 2.4.2-1***.
3) ***Найдите*** *символьное выражение производной от функции f(x) и вычислить значение производной от функции f(x) в произвольной точке* ***c;***
4) ***Найдите*** *решение ОДУ на отрезке [a;b] с шагом h с использованием функции* **ode() .**
5) ***Создайте*** *матрицу решений, записав в первый столбец аргумент, во второй и третий - решение, полученное с использованием функций* **ode().**
6) ***Выведите*** *полученную таблицу по столбцам.*
7) ***Постройте*** *график полученных решений ОДУ.*
8) ***Сохраните*** *текст рабочего окна на внешнем носителе.*
9) ***Предоставьте*** *результаты работы преподавателю,* ***ответьте*** *на поставленные вопросы.*
10) ***Выполните*** *команду* **clearall.**
11) ***Оформите отчет*** *по выполненной работе***.**

2. Варианты индивидуальных заданий

Таблица 2.4.2-1

№	f(x)	ОДУ	Начальные условия	b	h
1	$8e^{-x}Sin(-2x)$	$y' = x + 2y$	$y(0) = 1$	0,5	0,5
2	$Sin(x+1)e^{2/x}$	$y' = 2\sqrt{xy}$	$y(1) = 1$	1.2	0.2
3	$3x - Cos(x) - 1$	$y' = x + e^{y}$	$y(0) = 0$	0.4	0.2
4	$tgx - \frac{7}{2x+6}$	$y' = \sqrt{y}x$	$y(0) = 1$	1	0,5
5	$(x-3)Cos(x) - 1$	$y' = \sqrt{y} + 2x$	$y(0) = 1$	0.4	0,2
6	$Sin(x+1) - 0{,}5x$	$y' = x^2 - 2y$	$y(0) = 0$	1	0.25
7	$x^2Cos(2x) + 1$	$y' = e^{y} - 2x$	$y(0) = 1$	0.2	0,2
8	$(x-1)^2 \cdot \ln(x+11) - 1$	$y' = ye^{x/2}$	$y(0) = 1$	0.5	0,25
9	$Cos(x + 0{,}5) - x^2$	$y' = e^{x} + x$	$y(0) = 1$	0,25	0.25
10	$\frac{x}{\sin(x)} + 2x^2$	$y' = y^2 - x$	$y(0) = 1$	1	0,5
11	$\frac{e^{x}}{2} - (x-1)^2$	$y' = e^{y} - 2x$	$y(0) = 1$	0,3	0,15
12	$e^{-6x} + 3x^2 - 18$	$y' = y + x^2y$	$y(0) = 1$	0,5	0,25
13	$x - \ln(7x - 4)$	$y' = 1.5x^2 / 2(y+1)$	$y(0) = 0$	0,5	0.5
14	$Sin(Sin(x)) - 0{,}5x$	$y' = (y-1)/x$	$y(0) = 1$	1	0,5
15	$x^2Sin(\ln(x))$	$y' = 2xy - 3$	$y(1) = 1$	2	0,5
16	$4e^{-1/x} + x^3 - Cos(x)$	$y' = (x+y)^2$	$y(1) = 1$	2	0,5
17	$e^{-2x} + \frac{3}{x} + x^3$	$y' = xe^{-y}$	$y(1) = 0$	2	0,5
18	$2^{x} + \ln(2x) - 5{,}6x^4$	$y' = (x-y)^2$	$y(1) = 1$	2	0,5
19	$x^2 + Sin(x/2) + 1$	$y' = \cos(y) + x^2$	$y(1) = 1$	2	0,5
20	$2e^{-2x}Sin(2x)$	$y' = 2xy + x^2$	$y(1) = 1$	2	0,25
21	$x - \ln(7x - 4)$	$xy' = x^2 - y$	$y(1) = 1$	2	0.25
22	$8e^{-x}Sin(-2x)$	$y' = y^2 - x$	$y(0) = 1$	1	0,5
23	$Sin(x+1)e^{2/x}$	$y' = x^2 - y$	$y(1) = 1$	1,5	0,5
24	$3x - Cos(x) - 1$	$y' = xy^2 + 1$	$y(1) = 1$	1.4	0.2
25	$tgx - 7/(2x+6)$	$xy' + y = x$	$y(1) = 1$	2	0,5
26	$Sin(x+1)e^{2/x}$	$xy' + y = x^2$	$y(1) = 1$	2	0,5
27	$x - \ln(7x - 4)$	$xy' - y = x^2$	$y(1) = 1$	2	0,5
28	$Cos(x + 0{,}5) - x^2$	$y' = y/(1+x)$	$y(0) = 1$	1,4	0,2
29	$(x-3)Cos(x) - 1$	$y' - x^2y = x$	$y(1) = 1$	1,4	0,2
30	$Sin(x+1)e^{2/x}$	$xy' - y = x$	$y(1) = 1$	2	0.5

3. Содержание отчета

1) В форме комментариев:
   - Название лабораторной работы
   - ФИО студента, номер группы
   - № варианта
   - Индивидуальное задание
2) Протокол вычислений (сессии) в Командном окне, снабженный необходимыми комментариями.

2.4.3. Контрольные вопросы по теме

1) Какие начальные условия должны быть заданы в соответствии с задачей Коши при решении ОДУ средствами системы **Scilab**?
2) Какие численные методы реализованы в функции **ode()?**
3) Какие параметры являются в функции **ode()** обязательными?
4) В какой форме должны быть записаны функции ОДУ при использовании **ode()?**
5) Какими средствами решается система ОДУ?
6) Что представляет собой решение ОДУ 1-го и 2-го порядка при использовании функций **ode()?**

Тема 2.5. Технология решения задач одномерной оптимизации

2.5.1. Решение задач одномерной оптимизации функций
2.5.2. Лабораторная работа
2.5.3. Контрольные вопросы по теме

2.5.1. Решение задач одномерной оптимизации

При решении задачи поиска экстремума (максимума или минимума) функции **y=f(x)** одной переменной выделяют задачи поиска ***локального*** и ***глобального*** экстремума. При этом задача нахождения максимума целевой функции сводится к задаче нахождения минимума путем замены функции **f(x)**на-**f(x)**, поэтому в дальнейшем будем говорить только о поиске минимума функции, то есть такого**x*∈[a, b],** при котором **f(x*) = minf(x)**.

Интервал, на котором локализован единственный минимум, называется **отрезком неопределенности.**

Задача одномерной оптимизации имеет единственное решение в том случае, если функция **f(x)** на отрезке **[a;b]** имеет только один экстремум. Тогда говорят, что функция унимодальная на отрезке **[a;b]**.

Известно, что **необходимым** условием существования экстремума дифференцируемой функции **f(x)** является выполнение равенства **f′(x) = 0**.

Достаточным условием унимодальности функции **f(x)** на отрезке **[a;b]** является следующее: если функция **f(x)** дважды дифференцируема на отрезке **[a,b]** и **f′′(x*)>0** в любой точке этого отрезка, то функция **f(x)** - унимодальна на отрезке **[a,b]**.

Для решения задачи одномерной оптимизации с заданной степенью точности используются методы: **дихотомии**, **золотого сечения**, **средней точки** и многие другие. При этом суть методов одномерного поиска заключается в том, что на каждой итерации интервал неопределенности уменьшается и стягивается к точке минимума. Уменьшение отрезка происходит до тех пор, пока на некоторой **n**-й итерации отрезок неопределенности $[b_n;a_n]$ не станет соизмеримым с заданной погрешностью **ε**, то есть будет выполняться условие $|b_n-a_n| <\varepsilon$. Тогда за точку минимума можно принять любую точку, принадлежащую этому отрезку, в частности, его середину.

В Scilab поиск локального минимума любой функции (одномерной или многомерной) осуществляет функция **optim()**. В случае решения задачи одномерной оптимизации функция имеет следующий формат:

[f,xopt]=optim(costf,x0),

где **входными параметрами** функции **optim()** в случае решения задачи одномерной оптимизации являются:

x0- начальное приближение к точке минимума;

constf- имя функции, для которой ищется минимум.

Функция **optim()** возвращает значения следующих параметров:

f- минимальное значение функции;

xopt - точку, в которой функция достигает этого значения.

Главной особенностью функции **optim()** является структура вспомогательной функции **costf():**

```
unction [f,g,ind]=costf(x,ind)
f=gg(x);
g=numdiff(gg,x);
endfunction
```

где **f**- функция, минимум которой ищется, а выходной параметр **g –** ее производная.

Если возвращаемое сформированной функцией **costf()** значение **ind** равно 2, 3 или 4, то функция **costf()** обеспечивает поиск минимума, т.е. в качестве результата функции **optim** возвращается **f** и **xopt**. Если **ind=1**, то в функции **optim**() ничего не считается, условие **ind<0** означает, что минимум функции **f(x)** не может быть оценен, а **ind=0** прерывает оптимизацию. **ind** является внутренним параметром для связи между **optim()** и **costf()**. При использовании функции **optim** необходимо помнить, что параметр **ind** должен быть определен в функции **costf()**.

Таким образом, при использовании функции **optim()** необходимо сформировать функцию **costf()**, которая должна возвращать значение минимизируемой функции **f** и ее производную.

Чтобы с использованием функции Scilab **optim()** вычислить локальный ***максимум*** необходимо взять целевую функцию с противоположным знаком.

Пример2.5.1-1.Найти локальный минимум функции $f(x) = x^4 + 3x^3 - 13x^2 - 26x + 26$.

Решение задачи начнем с построения графика (рис.2.5.1-1 и 2.5.1-2). Определим отрезок, содержащий точку минимума, исходя из вида графика функции. Если это возможно, то на этом отрезке следует провести полное исследование функции на унимодальность: от функции $f(x)$ получить и первую и вторую производные и показать, что $f'(x)$ должна быть неубывающей, а $f''(x) > 0$.

```
Командное окно Scilab 6.0.0-beta-2

--> x=-5:0.1:1;
--> y=x.^4+3*x.^3-13*x.^2-6*x+26;
--> plot(x,y);
--> xtitle('График функции f(x)=x^4+3*x^3-13*x^2-6*x+26','X','Y');
--> xgrid();
```

Рис. 2.5.1-1. Построение графика функции f(x)

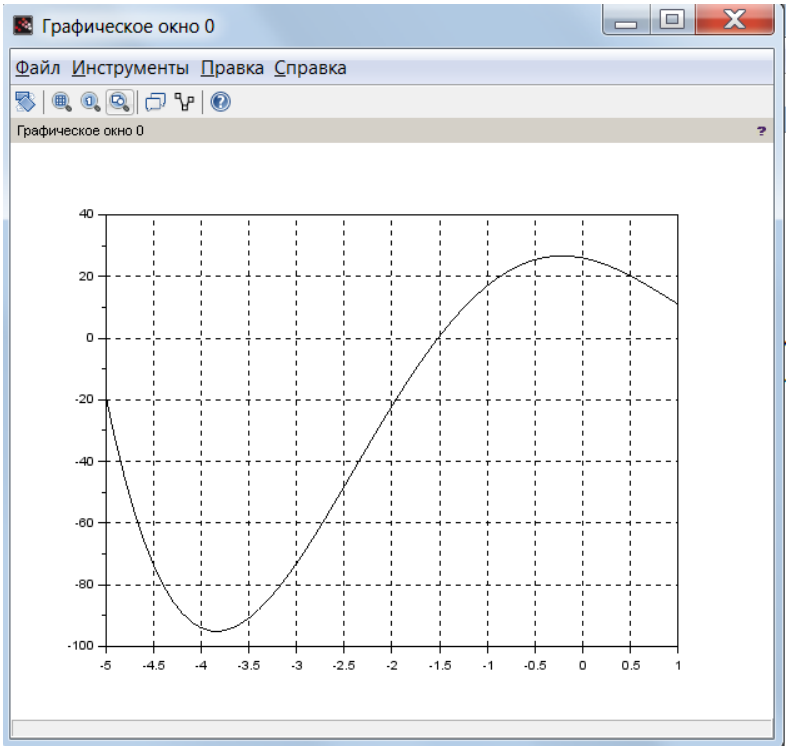

Рис. 2.5.9-2. График функции f(x)

Из графика следует, что на отрезке**[-4;-2]** имеется локальный минимум. Исследуем функцию и поведение производных.

Получение значений функции и производных на выбранном отрезке приведено на рис. 2.5.1-3.

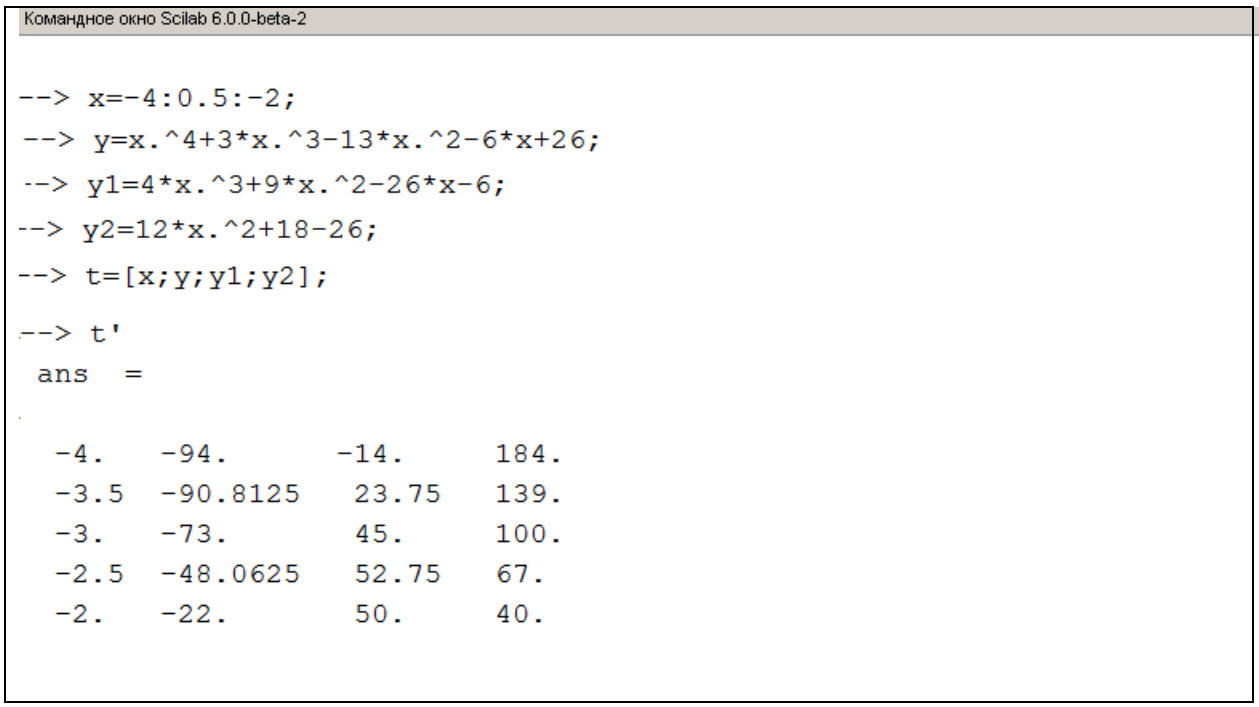

Рис.2.4.1-3. Исследование функции f(x) на отрезке [-4;-2]

Из полученной таблицы значений функции и производных видно, что на отрезке [-4;-2] существует единственный минимум. Найдем координаты этого минимума с использованием функции Scilab **optim()** (рис. 2.5.1-4).

Если для целевой функции затруднительно находить производные, то для того чтобы найти границы унимодальности функции можно построить ее график и выбрать отрезок, содержащий единственный минимум.

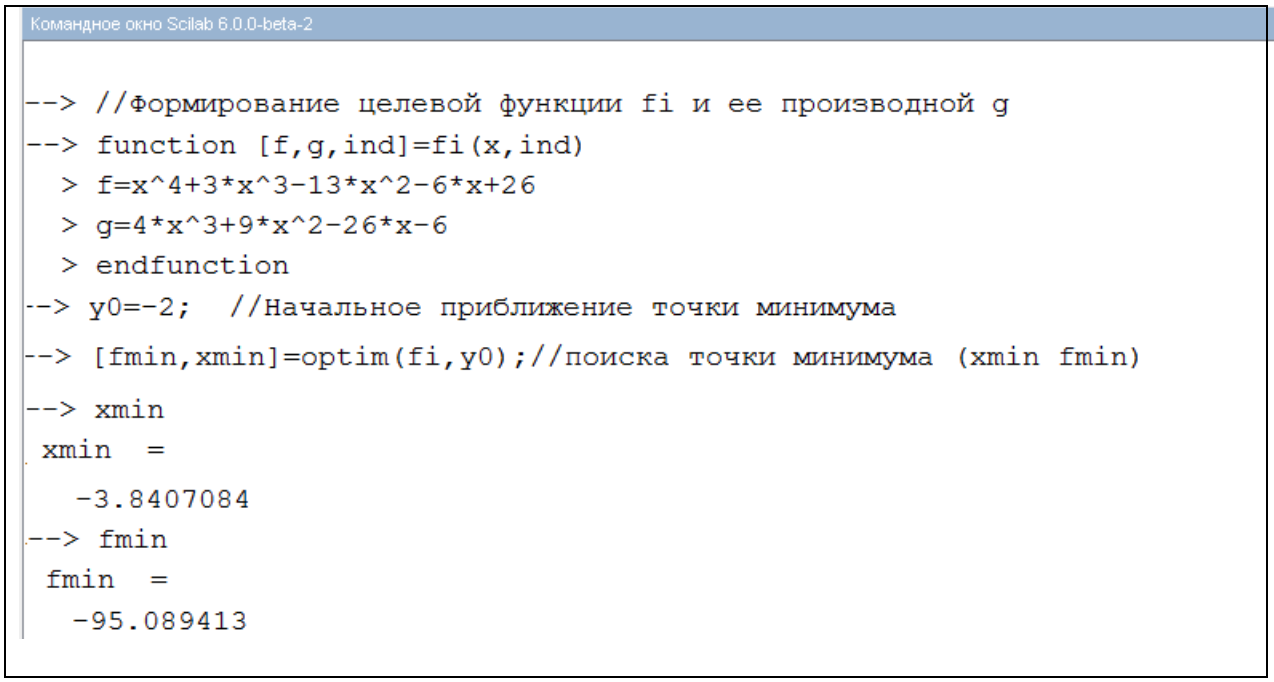

Рис.2.4.1-4. Нахождение координат точки минимума функции f(x)

2.5.2. Лабораторная работа по теме «Технология решения задач одномернойоптимизации»

1. Вопросы, подлежащие изучению

1) Условие унимодальности функции на отрезке.
2) Получение таблиц значений функции и ее производных с использованием средств пакета Scilab.
3) Технология использования встроенной функции пакета Scilab – **optim().**

2. Общее задание

1) ***Изучите*** *материал* ***Темы 2.5 (п. 2.5.1)****.*
2) ***Выберите*** *индивидуальное задание из* ***табл. 2.5.2-1****.*
3) ***Постройте*** *график функции* ***f(x),*** *выбрать отрезок, содержащий минимум;*
4) ***Проверьте*** *на выбранном отрезке условие унимодальности функции, получив таблицу значений первой или второй производной.*
5) ***Создать*** *функцию для формирования значений целевой функции и ее производной.*
6) ***Найдите*** *координаты точки минимума* ***f(x)*** *с использованием встроенной функции Scilab* ***optim().***
7) ***Сохраните*** *текст рабочего окна на внешнем носителе.*
8) ***Предоставьте*** *результаты работы преподавателю,* ***ответьте*** *на поставленные вопросы.*
9) ***Выполните*** *команду* ***clear all.***
10) ***Оформите отчет*** *по выполненной работе****.***

3. Варианты индивидуальных заданий

Таблица 2.5.2-1

№	f(x)	№	f(x)
1	$x^2 - 6\ln(x+1) + 2{,}8$	16	$2x^2 - \sin x - 1{,}5$
2	$x^2 - 4\sin x + 0{,}8$	17	$2\cos(x - x^3) - x$
3	$x^2 - 3\sin x - 2{,}4$	18	$x^2 - 3{,}4\ln(1+x) - 1$
4	$x^2 + \sin x - 0{,}5\sqrt[3]{(1+2x)}$	19	$\mathrm{e}^{\mathrm{x}} - (\mathrm{x}+1)^2 - 2$
5	$3x^2 - 2\cos(x+1) - 4{,}8$	20	$x^2 - \sin x$

6	$x^2 - 1{,}5\cos(x+3) - 1{,}8$	21	$\mathrm{e}^{\mathrm{x}} - 4\mathrm{x} - 9{,}5$
7	$\ln(x+2) + \cos(2+x) - 0{,}35$	22	$x^2 - 3\sin x - 2x$
8	$x^3 - 4{,}8\sin(1+2x) - 1$	23	$3 - 2\sin x - \sqrt{x^2+1}$
9	$x^2 - 2\sin(x+1) - 2$	24	$-(2 + 3\sin x - \sqrt{x^2+2}$
10	$e^{x-3} - \ln(x+3) + 1$	25	$\mathrm{e}^{-0{,}4\mathrm{x}} + 3\mathrm{x}^2 - 5\mathrm{x} - 3$
11	$-\sin x - \sqrt{x} + 1$	26	$x^2 + \cos(x-4) + \sqrt{x+1} - 3$
12	$\cos(x - 0{,}2x^2) - x + 6$	27	$x^2 - \ln x - 3$
13	$\mathrm{e}^{\mathrm{x}} - 4\mathrm{x}$	28	$-x + e^{0{,}5x}$
14	$0{,}1x^2 + x\ln x - x + 0{,}5$	29	$-\sin x - \sqrt{1+x} + 2$
15	$2\mathrm{e}^{-\mathrm{x}} + \mathrm{e}^{\mathrm{x}} - 4$	30	$4\mathrm{x}^3 + 2\mathrm{x}^4 + 9{,}5\mathrm{e}^{-2\mathrm{x}} - 35$

1. Содержание отчета

1) В форме комментариев:
   - Название лабораторной работы
   - ФИО студента, номер группы
   - № варианта
   - Индивидуальное задание
2) Протокол вычислений (сессии) в **Командном окне**.

2.5.3. Контрольные вопросы по теме

1) Понятия локального и глобального минимума функции?
2) Можно ли средствами **Scilab** вычислить глобальный минимум заданной функции?
3) Какие исследования функции необходимо произвести перед применением перед поиском локального минимума?
4) Функция **optim()** и назначение ее входных и выходных параметров.
5) Для чего при использовании функции **optim()** необходимо сформирование вспомогательной функции?
6) Можно ли с использованием функции вычислить локальный максимум?

Тема 2.6. Технология решения задач многомерной оптимизации

2.6.1. Решение задач многомерной оптимизации функций
2.6.2. Лабораторная работа
2.6.3. Контрольные вопросы по теме

2.6.1. Решение задач многомерной оптимизации

Задача, требующая нахождения оптимального значения функции m переменных $\mathbf{Q(X)=Q(x_1, x_2, \ldots, x_m)}$, называется задачей **многомерной оптимизации**. Так же, как и для случая одномерной оптимизации, задача нахождения максимума функции сводится к задаче нахождения минимума путем замены целевой функции **Q**на **-Q**.

В постановке задачи безусловной оптимизации для $\mathbf{Q(X)=Q(x_1, x_2, \ldots, x_m)}$ требуется найти хотя бы одну точку минимума **X*** и вычислить **Q*=f(X*)**. Точка $\mathbf{X^* \in R^m}$ называется точкой глобального минимума функции **Q**на множестве **X**, если для всех $\mathbf{X \in R^m}$ выполняется неравенство $\mathbf{Q(X^*) \le Q(X)}$. В этом случае значение **Q(X*)** называется минимальным значением функции **Q**на $\mathbf{R^m}$. Точка $\mathbf{X^* \in R^m}$ называется точкой локального минимума функции **Q**, если существует такая δ - окрестность $\mathbf{U_\delta}$этой точки ($\delta>0$), что для всех $\mathbf{X \in X_\delta = X \cap U_\delta}$ выполняется неравенство $\mathbf{Q(X^*) \le Q(X)}$.

Для всякой непрерывно дифференцируемой функции Q достаточным условием того, функция имеет точку минимума, является положительная определенность матрицы вторых частных производных (**матрицы Гессе**):

$$G(\bar{x}) = f''(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_m} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_m} \\ \cdots & \cdots & \cdots \\ \frac{\partial^2 f}{\partial x_m \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_m \partial x_m} \end{bmatrix}$$

Известно, что, для того чтобы матрица была положительно определена, необходимо, чтобы все угловые миноры были положительны. Так для функции двух переменных Q(x, y) матрица Гессе имеет вид:

$$G(x,y) = \begin{vmatrix} \frac{\partial^2 Q}{\partial x^2} & \frac{\partial^2 Q}{\partial x \partial y} \\ \frac{\partial^2 Q}{\partial y \partial x} & \frac{\partial^2 Q}{\partial y^2} \end{vmatrix},$$

а достаточное условие существования минимума является выполнение неравенств:

$$\Delta_1 = \frac{\partial^2 Q}{\partial x^2} > 0,$$

$$\Delta_2 = \frac{\partial^2 Q}{\partial x^2} \cdot \frac{\partial^2 Q}{\partial y^2} - \left(\frac{\partial^2 Q}{\partial x \partial y} \right)^2 > 0.$$

Аналитический метод поиска минимума применяется только для ограниченного круга задач. В основном это связано с необходимостью решения системы нелинейных уравнений, которая, как правило, решается численными методами. Гораздо проще решать задачу многомерной оптимизации численными методами, например, такими как **метод градиентного спуска с дроблением шага** и **методы наискорейшего спуска с аналитическим или численным выбором шага**.

Вычисление минимума функции нескольких переменных

$$z = f(x_1, x_2, \ldots x_n)$$

в **Scilab** осуществляет той же функцией **optim()**, что и для функции от одной переменной, но в назначении параметров имеются некоторые отличия. Общий формат функции **optim()** следующий:

[f,xopt]=optim(costf,x0),

где входными параметрами в случае в случае решения задачи многомерной оптимизации являются:

x0- вектор-столбец начальных приближений длиной **n** (**n** – количество параметров оптимизации);

constf– имя функции, для которой ищется минимум.

Функция возвращает значения параметров:

f - минимальное значение функции и **xopt** - значения параметров оптимизации, при которых функция достигает минимального значения.

Функция **optim()** требует вспомогательную функцию **costf()**, имеющую следующую структуру**:**

```
function [f,g,ind]=costf(x,ind)
f=gg(x);
g=numderivative(gg,x);
endfunction
```

где **f** - функция, минимум которой ищется;

g - градиент функции **f** (вектор частных производных **f** по **x**).

ind- является внутренним параметром для связи между **optim** и **costf**. Используя функцию **optim()** необходимо помнить, что параметр **ind** должен быть определен в функции **costf**. Подробности об этом параметре описаны в 2.5.1.

Таким образом, решение задачи оптимизации в пакете Scilab требует дополнить функцию optim() функцией costf, которая возвращает значение минимизируемой функцию f и ее частных производных.

Чтобы с использованием optim() вычислить локальный максимум необходимо взять целевую функцию с противоположным знаком.

Рассмотрим работу функции **optim()** на примере определения минимума двумерной функции$f(x_1, x_2) = 100(x_2–x_1^2)^2+(1-x_1)^2$.

Построим график (рис. 2.6.1-1) с использованием функции **ezsurf()**, аргументами которой служат: выражение функции, заключенное в одинарные кавычки, вектор изменения первой производной и вектор изменения второй переменной.

```
--> [x1 x2]=meshgrid(-1:0.1:1,-1:0.1:1);
--> z=100*(x2-x1.^2).^2+(1-x1).^2;
--> surf(x1,x2,z)
```

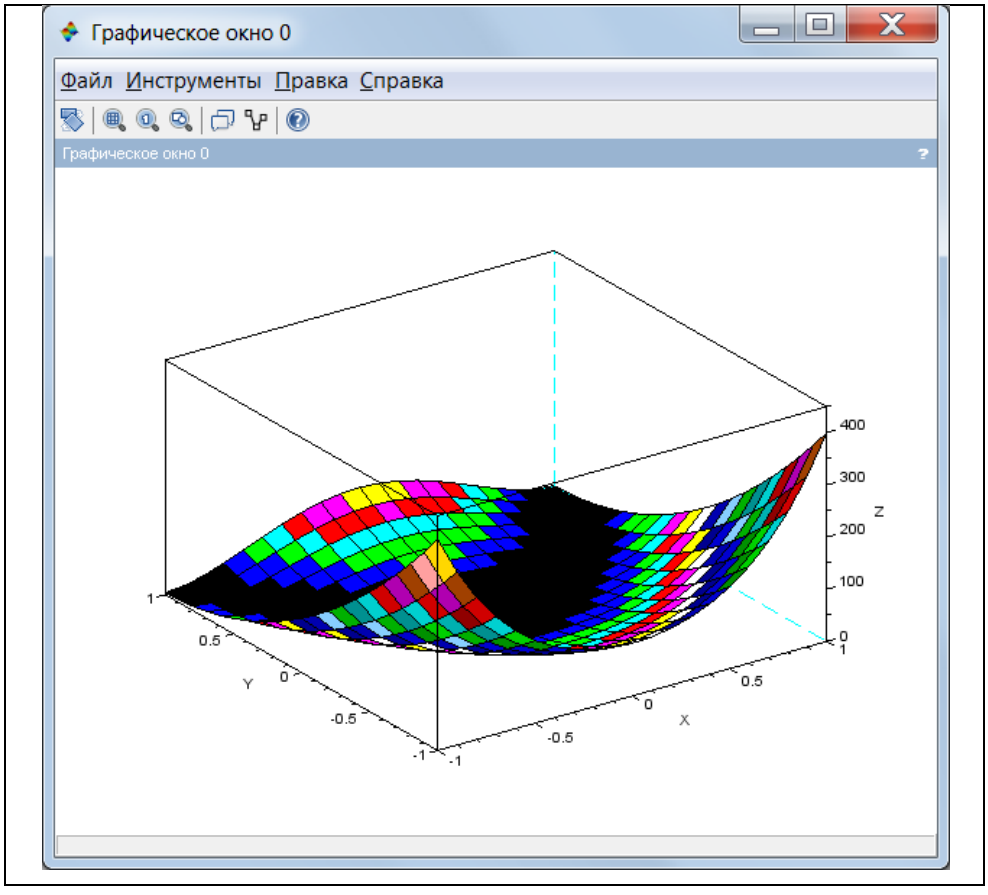

Рис. 2.6.1-1. Результат выполнения функции **surf()**

Определим координаты точки минимума и значение функции в этой точке с использованием функции Scilab **optim()**(рис.2.6.1-2).

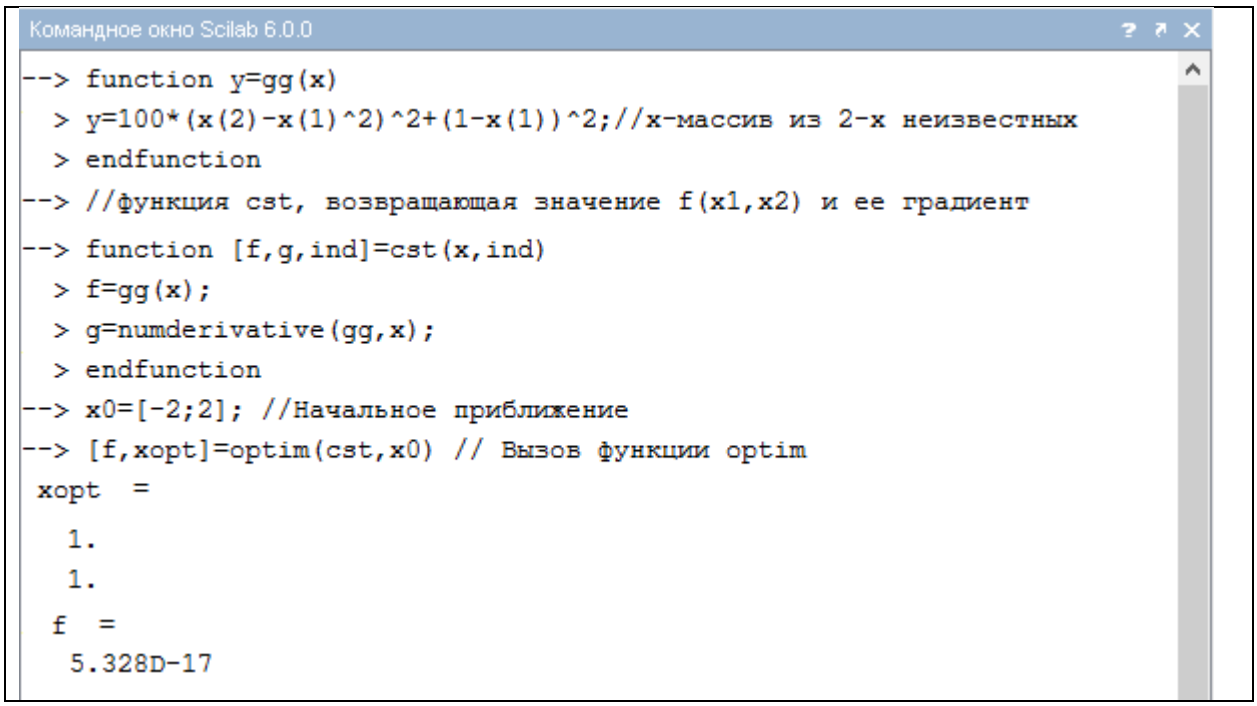

Рис. 2.6.1-3.Использование функции **optim()** для нахождения минимума многомерной функции

2.6.2. Лабораторная работа по теме «Технология решения задач многомерной оптимизации»

1. Вопросы, подлежащие изучению

1) Построение графиков функции от двух переменных **F(x_1,x_2)**средствами пакета Scilab.
2) Нахождение координат точки минимум функции **F(x_1,x_2)** с использованием встроенных функций пакета **Scilab - optim().**

2. Общее задание

1) ***Изучите*** *материал****Темы 2.6 (п. 2.6.1)****.*
2) ***Выберите*** *индивидуальное задание из* ***табл. 2.6.2-1****.*
3) ***Постройте*** *график функции* ***F(x_1,x_2);***
4) ***Сформируйте*** *для использования функции* ***optim()*** *вспомогательную функцию, необходимую для вычисления функции и частных производных.*
5) ***Найдите*** *координаты точки минимум функции* ***F(x_1,x_2)****с использованием встроенной функций* ***optim();***
6) ***Получите*** *значение функции, значений ее параметров, обеспечивающих оптимальное значение и частных производных в точке минимума.*
7) ***Сохраните*** *текст рабочего окна на внешнем носителе.*
8) ***Предоставьте*** *результаты работы преподавателю,* ***ответить*** *на поставленные вопросы.*
9) ***Выполните*** *команду* ***clearall.***
10) ***Оформить отчет*** *по выполненной работе****.***

1. Варианты индивидуальных заданий

Таблицы 2.6.2-1

№	Функции для вычисления минимума
1	$F(x_1, x_2) = 2x_1^2 + 5x_2^2 - 3x_1x_2 - 2x_1 - 22$
2	$F(x_1, x_2) = 2x_1^2 + x_2^2 + 2x_1x_2 + x_1 + x_2$
3	$F(x_1, x_2) = 3x_1^2 + 5x_2^2 - 3x_1x_2 + x_1 + 13$
4	$F(x_1, x_2) = x_1^2 + 2x_2^2 - x_1x_2 - x_2$
5	$F(x_1, x_2) = 5x_1^2 + 3x_2^2 - 4x_1x_2 + 2x_1 - 18$

6	$F(x_1, x_2) = 2x_1^2 + 3x_2^2 + x_1 + 3x_2 + 10$
7	$F(x_1, x_2) = 5x_1^2 + x_2^2 - 2x_1x_2 + 2x_2 + 21$
8	$F(x_1, x_2) = 2x_1^2 + 3x_2^2 - 2x_1x_2 + x_1 + 3$
9	$F(x_1, x_2) = x_1^2 + 5x_2^2 + x_2^2 - x_1 + 2x_2 + 10$
10	$F(x_1, x_2) = 4x_1^2 + 3x_2^2 - 3x_2 + 8$
11	$F(x_1, x_2) = 3x_1^2 + 6x_2^2 - 2x_1 + 5$
12	$F(x_1, x_2) = x_1^2 + 2x_2^2 - x_1x_2 + x_2 + 16$
13	$F(x_1, x_2) = x_1^2 + 2x_2^2 - x_1x_2 + x_2 + 16$
14	$F(x_1, x_2) = x_1^2 + 4x_2^2 + 2x_1x_2 + 11$
15	$F(x_1, x_2) = 2x_1^2 + x_2^2 - x_1x_2 + 2x_1$
16	$F(x_1, x_2) = x_1^2 + 2x_2^2 + 2x_1 + x_2 + 4$
17	$F(x_1, x_2) = 3x_1^2 + x_2^2 + x_1x_2 + 2x_1 + 3x_2 + 15$
18	$F(x_1, x_2) = 2x_1^2 + x_2^2 + 2x_1x_2 + x_1 - 12$
19	$F(x_1, x_2) = 2x_1^2 + x_2^2 + x_1x_2 + 5x_1 + 18$
20	$F(x_1, x_2) = 3x_1^2 + 2x_2^2 + x_1 + 2x_2 + 9$
21	$F(x_1, x_2) = 2x_1^2 + x_2^2 - x_1x_2 + x_1 + 2x_2$
22	$F(x_1, x_2) = 2x_1^2 + 3x_2^2 - x_1 + x_2 + 26$
23	$F(x_1, x_2) = x_1^2 + 5x_2^2 - x_1x_2 - 3x_2 + 13$
24	$F(x_1, x_2) = 4x_1^2 + 3x_2^2 - 2x_1 + 11$
25	$F(x_1, x_2) = 3x_1^2 + 6x_2^2 + 2x_1x_2$
26	$F(x_1, x_2) = x_1^2 + 2x_2^2 - x_1x_2 + 2x_1 - x_2 + 4$
27	$F(x_1, x_2) = x_1^2 + 3x_2^2 + 2x_1 + 3x_2 + 12$
28	$F(x_1, x_2) = x_1^2 + 4x_2^2 + x_1x_2 + 2x_1 - 12$
29	$F(x_1, x_2) = 5x_1^2 + x_2^2 + 2x_1x_2 + x_1 + 16$
30	$F(x_1, x_2) = 3x_1^2 + 2x_2^2 + x_1x_2 + x_1 + 2x_2 + 9$

2. Содержание отчета

1) В форме комментариев:
   - Название лабораторной работы
   - ФИО студента, номер группы
   - № варианта
   - Индивидуальное задание

2) Протокол вычислений (сессии) в **Командном окне**, снабженный необходимыми комментариями.

2.6.3. Контрольные вопросы по теме

1) Какую функцию называют многомерной?
2) Что является достаточным условием существования минимума для многомерной функции?
3) Назначение функции **optim(),** ее входных и выходных параметров.
4) Можно ли с использованием функции **optim()** вычислить локальный максимум?
5) Назначение функции ***costf(),*** ее входные и выходные параметры.
6) Что служит результатом выполнения функции **optim()?**
7) Какие средства Scilab используются для описания целевой функции, и как это влияет на параметры функции **optim()?**

Список литературы

1. Кочегурова Е.А., Особенности системы Matlab для решения задач вычислительной математики, Учебное пособие/ ТПУ, 2013. - 7с.
2. Андриевский А.Б., Андриевский Б.Р., Капитонов А.А., Фрадков А.Л. Решение инженерных задач в среде Scilab. Учебное пособие/ СПб.: НИУ ИТМО, 2013. - 97с.
3. Шакин В.Н., Семенова Т.И. Основы работы с математическим пакетом Matlab, Учебное пособие/ МТУСИ, 2016. -133с.