Scilab

Scilab – одна из двух основных альтернатив MATLAB с открытым исходным кодом, вторая - GNU Octave. Его синтаксис немного отличается от синтаксиса MATLAB, чем Octave, но она достаточно похожа, что позволяет легко переносить навыки между этими двумя системами. Scilab – это система компьютерной математики, которая предназначена для выполнения инженерных и научных вычислений, таких как:

- решение нелинейных уравнений и систем;
- решение задач линейной алгебры;
- решение задач оптимизации;
- дифференцирование и интегрирование;
- обработка экспериментальных данных (интерполяция и аппроксимация, метод наименьших квадратов);
- решение обыкновенных дифференциальных уравнений и систем.

Кроме того, Scilab предоставляет широкие возможности по созданию и редактированию различных видов графиков и поверхностей.

Несмотря на то, что система Scilab содержит достаточное количество встроенных команд, операторов и функций, отличительная ее черта – это гибкость. Пользователь может создать любую новую команду или функцию, а затем использовать ее наравне со встроенными. К тому же, система имеет достаточно мощный собственный язык программирования высокого уровня, что говорит о возможности решения новых задач.

Установка Scilab на ПК

Свободно распространяемую версию пакета вместе с полной документацией на английском языке можно получить на сайте программы www.scilab.org.

Среда Scilab

После запуска Scilab на экране появиться основное окно приложения. Окно содержит меню, панель инструментов и рабочую область. Признаком

того, что система готова к выполнению команды, является наличие знака приглашения “-->”, после которого расположен активный (мигающий) курсор. Рабочую область со знаком приглашения обычно называют командной строкой. Ввод команд в Scilab осуществляется с клавиатуры. Нажатие клавиши Enter заставляет систему выполнить команду и вывести результат (рис. 1).

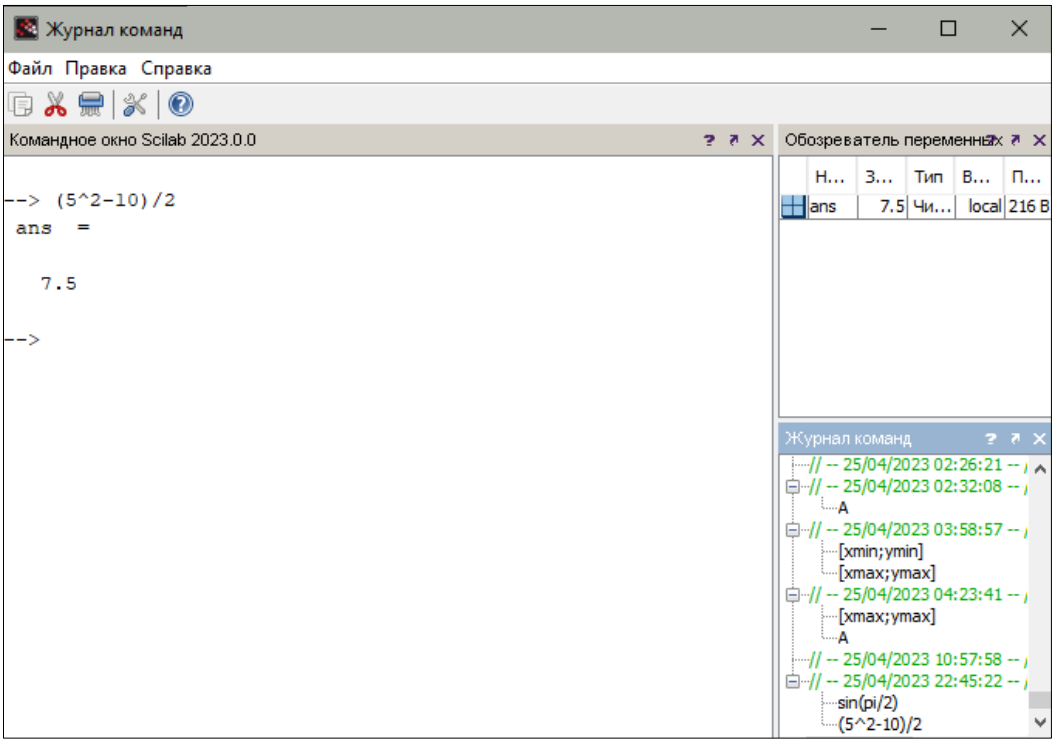

Рис. 1. Выполнение элементарной команды в Scilab

Понятно, что все выполняемые команды не могут одновременно находиться в поле зрения пользователя. Поэтому, просмотреть ту информацию, которая покинула видимую часть окна, можно в Журнале команд (рис. 1).

Кроме того, существуют особенности ввода команд. Если команда заканчивается точкой с запятой «;», то результат ее действия не отображается в командной строке. В противном случае, при отсутствии знака «;», результат действия команды сразу же выводится в рабочую область (рис. 2).

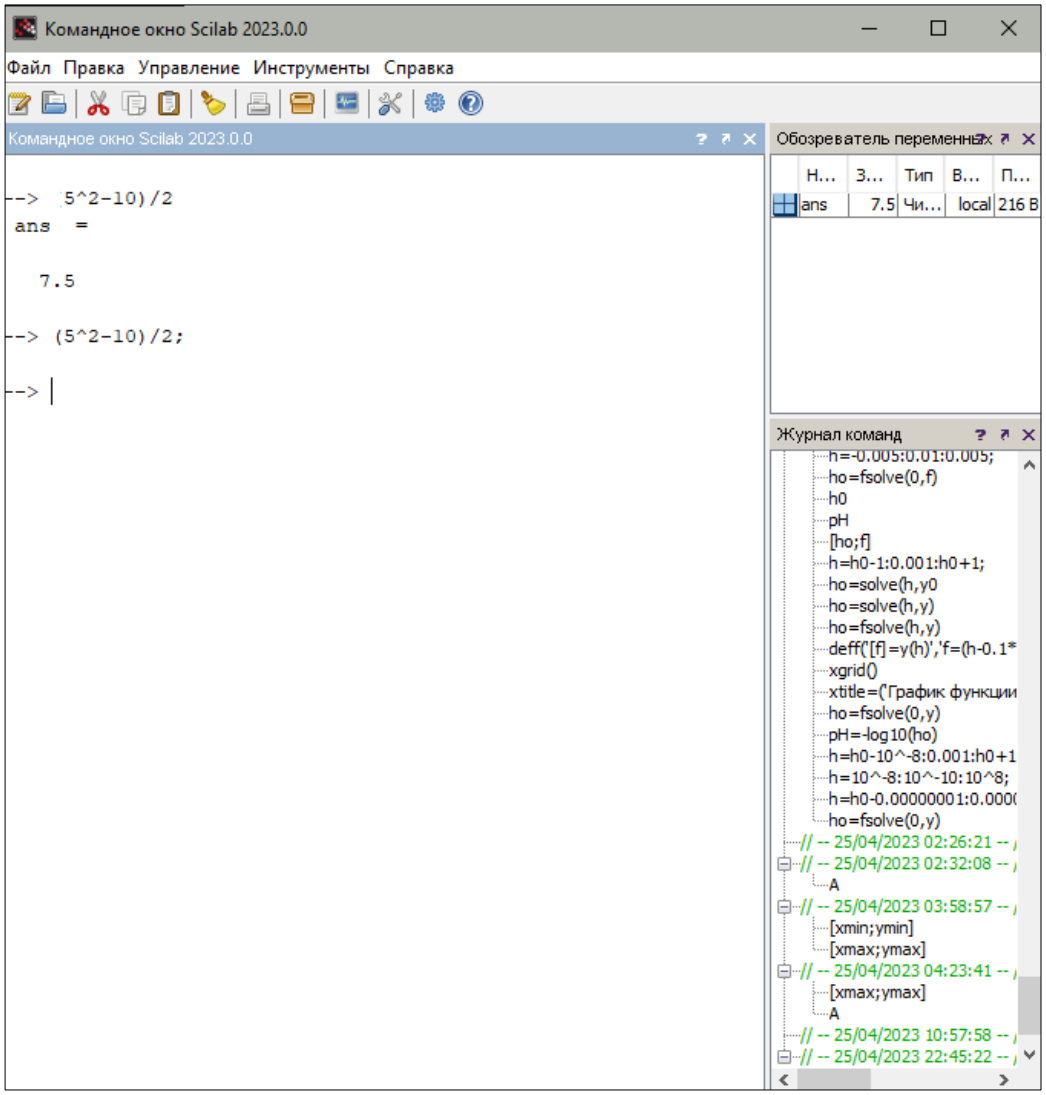

Рис. 2. Различие между командами с точкой с запятой и её отсутствием

Текущий документ, отражающий работу пользователя с системой Scilab, содержащий строки ввода, вывода и сообщения об ошибках, принято называть сессией. Значения всех переменных, вычисленные в течение текущей сессии, сохраняются в специально зарезервированной области памяти, называемой рабочим пространством системы. При желании определения всех переменных и функций, входящих в текущую сессию можно сохранить в виде файла, саму сессию сохранить нельзя.

Основные команды главного меню Scilab

Главное меню системы содержит команды, предназначенные для работы с файлами, настройки среды, редактирования команд текущей сессии и получения справочной информации. Кроме того, с помощью главного меню можно создавать, редактировать, выполнять отладку и запускать на

выполнение так называемые файлы-сценарии Scilab, а также работать с графическими приложениям пакета.

Справочная система

Команда главного меню ? открывает доступ к справочной системе Scilab. В справочной системе информацию можно искать, воспользовавшись содержанием, в списке, упорядоченном по алфавиту, по ключевому слову или фразе. С помощью команды Scilab Demos можно осуществить просмотр демонстрационных примеров.

Редактирование и отладка файлов-сценариев

Файл-сценарий – это список команд Scilab, сохраненный на диске. Для подготовки, редактирования и отладки файлов-сценариев служит специальный редактор SciNotes, который можно вызвать, выполнив команду главного меню Editor. В результате работы этой команды будет создан новый файл-сценарий. По умолчанию он имеет имя Untitled1.sce (рис. 3).

Окно редактора файлов-сценариев выглядит стандартно, т.е. имеет заголовок, меню, панели инструментов, строку состояния. Ввод текста в окно редактора файла-сценария осуществляется по правилам, принятым для команд Scilab.

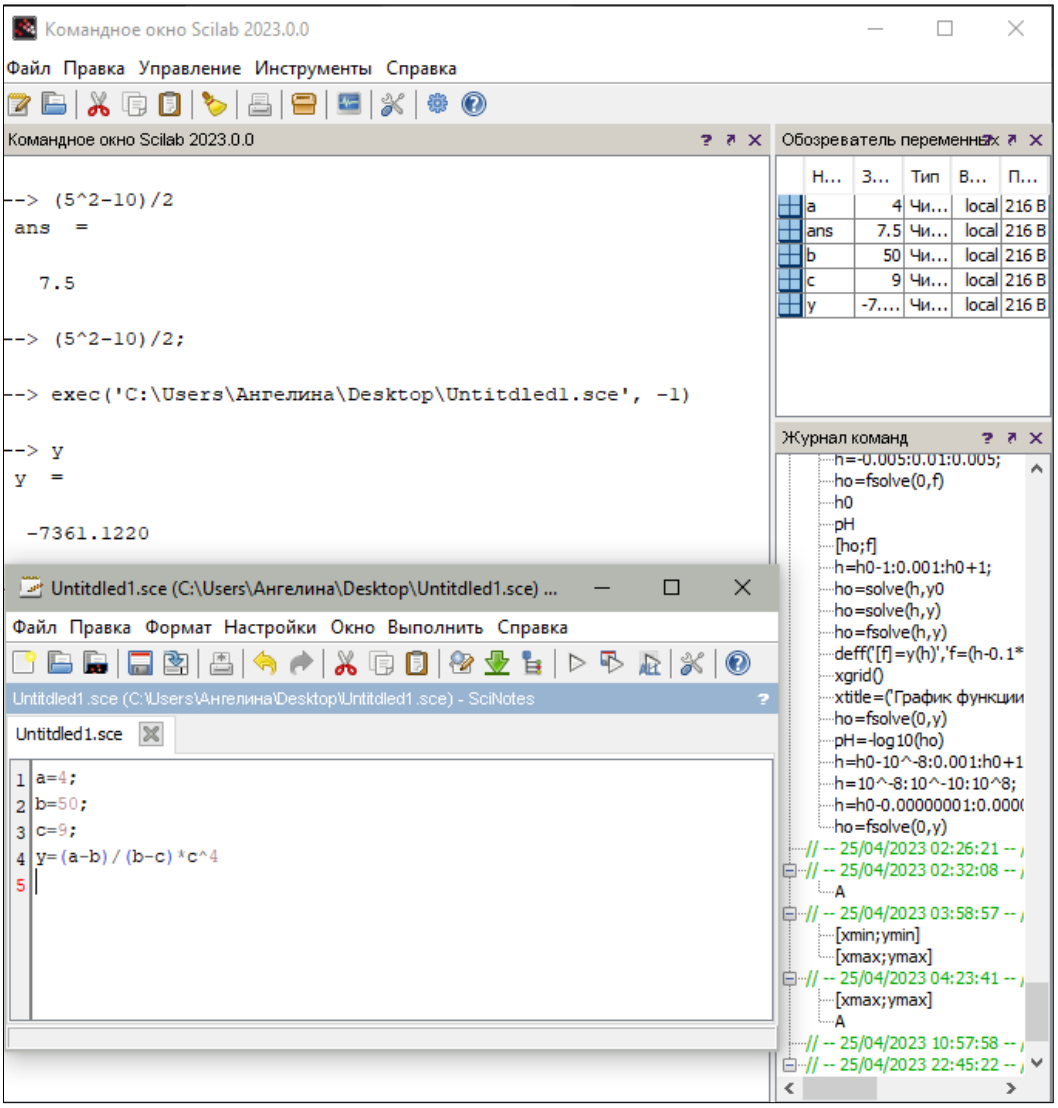

Рис. 3. Пример ввода команд для решения уравнения

Нетрудно заметить, что точка с запятой «;» ставится после тех команд, которые не требуют вывода значений. Для сохранения введенной информации необходимо выполнить команду File – Save из меню редактора. Если информация сохраняется впервые, то появится окно Save file As. . . . Ввод имени в поле File Name и щелчок по кнопке Save. приведет к сохранению информации, находящейся в окне редактора. Файлы-сценарии сохраняют с расширением .sce . Открывает ранее созданный файл команда главного меню File – Open.

Выполнить операторы файла-сценария можно несколькими способами:

• из меню редактора SciNotes вызвать команду Execute – Load into Scilab;

• из главного меню Scilab вызвать команду Exec и указать имя файла-

сценария.

Все эти действия приведут к появлению в рабочей области результатов вычислений команд файла-сценария (рис. 2).

Отметим, что редактор SciNotes имеет возможность работы с множеством окон (пункт меню Windows), обладает принятыми для текстовых редакторов приемами редактирования (пункт меню Edit) и поиска (пункт меню Search). Кроме того, можно выполнить настройку среды редактора SciNotes (пункт меню Options), вызвать справочную информацию (пункт меню Help) и осуществить отладку программы, набранной в редакторе (пункт меню Debug).

Выйти из режима редактирования можно, просто закрыв окно SciNotes или выполнив команду File – Exit.

Решение задач с использованием пакета Scilab.

Задача 1. Операции с матрицами

1. Определить переменную M как массив 3x4 элементов:

$$\begin{matrix} -25 & 13 & -21 \\ 26 & -25 & 23 \\ -7 & 7 & 1 \\ 1 & -29 & -10 \end{matrix}$$

2. Извлечь вторую строку в отдельный массив, присвоив его в новую переменную.

3. Вычислить максимальное значение и его порядковый номер в новом массиве.

4. Извлечь третий столбец исходного массива в отдельную переменную.

5. Вычислить треть суммы элементов этого массива, присвоить в переменную k.

6. Подсчитать количество элементов в M, больших по абсолютному значению чем k.

Реализация в SciLab:

В исполняемом файле в SciNotes запишем команды, для решения

задания.

Введем данные для матрицы, присвоив ей переменную M. Для создания матрицы в квадратных скобках запишем значения, где они выделяются между собой запятыми для строк, и точкой с запятой для столбцов:

M=[-25,13,-21;26,-25,23;-7,7,1;1,-29,-10];

Далее извлечем вторую строку массива M в переменную M2, для этого введем:

M2=M(2,:);

Вычислим максимальное значение массива M2 с помощью функции **max()**:

M2_max=max(M2);

Затем определим порядковый номер максимального значения в массиве M2 с помощью функции **find()**:

max_index=find(M2==M2_max);

Извлечем третий столбец массива M в переменную M3:

M3=M(:,3);

Далее найдем треть суммы(**sum()**) элементов массива M3:

k=sum(M3)/3;

Определяем количество элементов в M, больших по абсолютному значению, чем k:

n=sum(abs(M(:))>k)

Далее мы должны сохранить файл с командами и в командном окне SciLab выведем значения переменных, которые мы находили: M, M2, M2_max, max_index, M3, k, n:

```
--> M
 M  =
  -25.   13.  -21.
   26.  -25.   23.
  -7.    7.    1.
   1.   -29.  -10.
--> M2
```

```
 M2  =
   26.  -25.   23.
--> M2_max
 M2_max  =
   26.
--> max_index
 max_index  =
   1.
 --> M3
 M3  =
  -21.
   23.
   1.
  -10.
--> k
 k  =
  -2.3333333
--> n
 n  =
   12.
```

Задача 2. Решение систем неравенств

Дана функция:

$$y = \begin{cases} \sqrt{1 + |x|}, x \leq 0 \\ \dfrac{1 + 3x^2}{\sqrt[3]{1 + sinx} + 2}, x > 0 \end{cases}$$

1. Создать *.m-файл с определением функции по своему варианту.

2. Вычислить минимальные и максимальные значения функции на интервале [–8;8].

3. Построить график функции на этом интервале.

4. Нанести подписи осей графика и заголовок всего графика на изображение.

Реализация в SciLab:

В SciNotes пишем код с определения функции с помощью **function** с использованием **for**, **if**, **else**:

```
function [y]=zad2(x)
   for i=1:lenght(x)
      if x(i)<=0
         y(i)=sqrt(1+abs(x(i)));
```

```
        else
            y(i)=(1+3*(x(i))^2)/((1+sin(x(i)))^(1/3)+2)
        end
    end
end
```

Описание кода:

Пишем кодовое слово ***function****, где в квадратных скобках указываем имя функции. После равенства указываем название файла, где находится код. Далее указываем цикл от* ***1*** *до длины строки в переменной* ***x****, то есть цикл будет повторяться до тех пор, пока не найдет оптимальное решение* ***x****. Затем указываем допустимый промежуток для* ***x*** *в виде неравенства, тем самым указывая, что если* ***x*** *будет удовлетворять неравенству, то будет выполняться данное уравнение, в противном случае будет выполняться следующее уравнение.*

Вычислим максимальное и минимальное значение функции на промежутке от -8 до 8. Для этого сначала введем переменную x и укажем промежуток с шагом 0,01:

```
x=-8:0.01:8;
```

Далее находим значение для заданной функции на интервале **x**, при этом указывая, в каком файле изначально была указана формула для **y**:

```
y=zad2(x);
```

С помощью функции **max()** и **min()** найдем минимум и максимум функции на данном промежутке:

```
ymin=min(y);
ymax=max(y);
```

Теперь найдем значение x, соответствующее минимальному значению функции:

```
xmin=x(find(y==ymin));
xmax=x(find(y==ymax));
```

Строим график функции y=f(x), также строим точки минимальных и максимальных значений, с указанием типа точек (рис. 4):

```
plot(x,y);
```

plot2d(xmin,ymin,-6);
plot2d(xmax,ymax,-4);

Описание функции plot():

plot(x1, y1, s1, x2, y2, s2, ..., xn, yn, sn)

где x1, x2, ..., xn – массивы абсцисс графиков;

y1, y2, ..., yn – массивы ординат графиков;

s1, s2, ..., sn – строка, состоящая из трех символов, которые определяют соответственно цвет линии, тип маркера и тип линии графиков (табл. 1–3), в строке могут использоваться один, два или три символа одновременно в любой желаемой комбинации.

Таблица 1

Символы, определяющие цвет линии графика

Символ	Описание
y	желтый
m	розовый
c	голубой
r	красный
g	зеленый
b	синий
w	белый
k	черный

Таблица 2

Символы, определяющие тип маркера

Символ	Описание
.	точка
o	кружок
x	крестик
+	знак «плюс»
*	звездочка
s	квадрат
d	ромб
v	треугольник вершиной вниз
^	треугольник вершиной вверх
<	треугольник вершиной влево
>	треугольник вершиной вправо
p	пятиконечная звезда

Таблица 3

Символы, определяющие тип линии графика

Символ	Описание
-	сплошная (по умолчанию)
:	штрих, чередующийся с двумя точками
-.	штрих, чередующийся с одной точками
--	штриховая

Описание функции plot2d():

plot2d([logflag],x,y,[key1=value1,key2=value2,...,keyn=valuen]

logflag – строка из двух символов, каждый из которых определяет тип осей (n – нормальная ось, l – логарифмическая ось), по умолчанию – «nn»;

x – массив абсцисс;

y – массив ординат или матрица, каждый столбец которых содержит массив ординат очередного графика – в случае, если необходимо построить графики нескольких функций y1, y2, ..., yn, когда все они зависят от одной и той же переменной x. При этом количество элементов в массиве x и y должно быть одинаковым. Если x и y – матрицы одного размера, то каждый столбец матрицы y отображается относительно соответствующего столбца матрицы x;

keyi=valuei – последовательность значений свойств графика key1=value1, key2=value2, ..., keyn=valuen, определяющих его внешний вид. Возможные значения свойств графика будут подробно описаны ниже.

Возможны следующие значения параметра keyn=valuen:

style – определяет массив числовых значений цветов графика. Количество элементов массива совпадает с количеством изображаемых графиков. Можно воспользоваться функцией color, которая по названию (color("имя цвета")).

Функцию plot2d можно использовать для построения точечных графиков. В этом случае обращение к функции имеет вид:

plot2d(x,y,d)

d – отрицательное число, определяющее тип маркера (табл. 4).

Таблица 4

Числа, определяющие тип маркера

Число	Описание
-0	точка
-1	плюс
-2	крестик
-3	плюс, вписанный в окружность
-4	закрашенный ромб
-5	незакрашенный ромб
-6	треугольник вершиной вверх
-7	треугольник вершиной вниз
-8	плюс, вписанный в ромб
-9	кружок
-10	звездочка
-11	квадрат
-12	треугольник вершиной вправо
-13	треугольник вершиной влево
-14	пятиконечная звезда

Указываем функцию, которая выводит размерную сетку графика:

xgrid();

Используем функцию **xtitle()** для указания названия графика и названия осей:

xtitle('График функции y=f(x)','Ось x','Ось y')

Подписываем для графиков и точек легенду, с указанием места на графике:

legend('y=f(x)','Минимум функции','Максимум функции',2)

В случаях, когда в одной координатной плоскости изображаются графики нескольких функций, как в нашем примере, возникает необходимость в «легенде». Ее можно вывести с помощью команды legend:

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

leg1 – имя первого графика, leg2 – имя второго графика, legn – имя n-го графика;

pos – месторасположение легенды: 1 – в верхнем правом углу (по умолчанию), 2 – в верхнем левом углу, 3 – в нижнем левом углу, 4 – в нижнем правом углу, 5 – определяется пользователем после изображения графика;

boxed – логическая переменная, которая определяет, прорисовывать (значение по умолчанию – %t) или нет (значение %f) рамку вокруг легенды

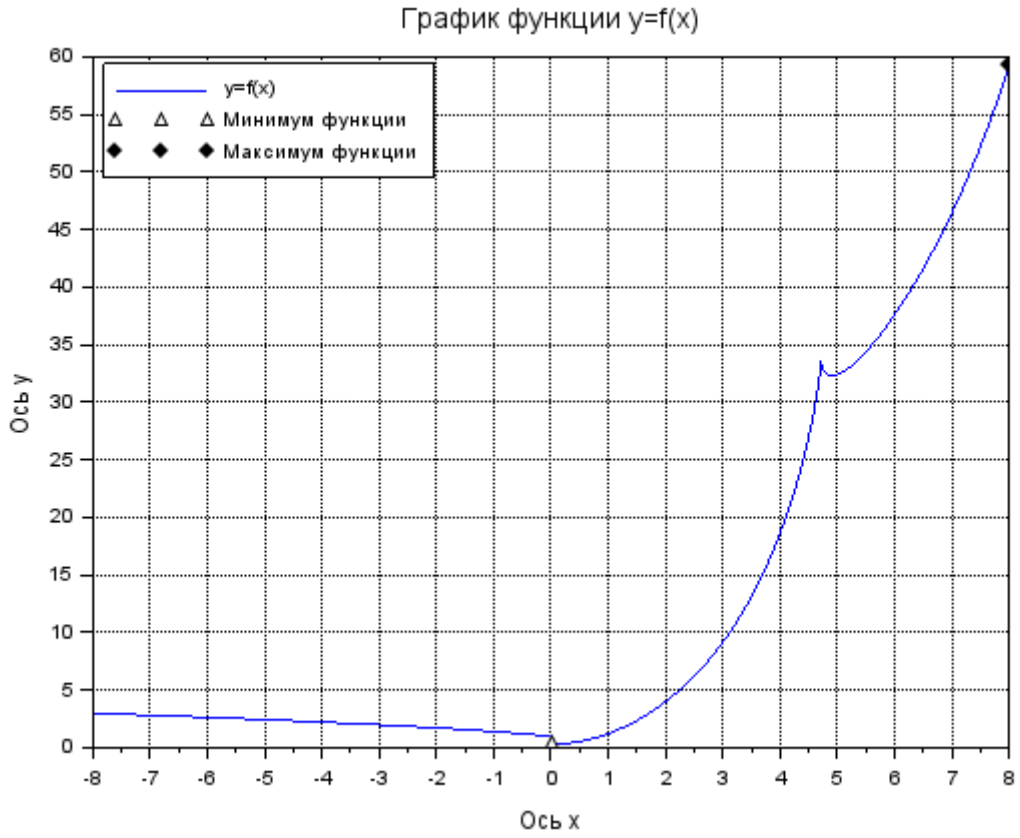

Рис. 4. График функции y=f(x), с указанием точек минимума и максимума функции

После того, как сохранили файл с кодом, в командном окне вывод значения максимальных и минимальных значений:

```
--> [xmin;ymin]
 ans  =
   0.0200000
   0.3329983
--> [xmax;ymax]
 ans  =
   8.
   59.244571
```

Задача 3. Запись данных в текстовый файл

Дана функция:

$$y = \begin{cases} \sqrt{1+|x|}, x \le 0 \\ \dfrac{1+3x^2}{\sqrt[3]{1+sinx}+2}, x > 0 \end{cases}$$

1. Протабулировать (вычислить значения функции в точках) функцию на интервале [- 10, 10] с шагом 0,01.

2. Не выводя результаты на экран, записать данные в текстовой файл с

тремя колонками: №, *x*, *f(x)*. Где № – номер точки по счету (всего должно получиться 3000 строк).

3. Считать получившийся файл в новую переменную как массив символов.

Реализация в SciLab:

Для определения заданной функции будем использовать файл zad2, описанный в задаче 2.

Создаем переменную **x** с заданным интервалом [-10;10] с шагом 0,01:

x=[-10:0.01:10];

Далее вычисляем значение заданной функции на заданном интервале, ссылаясь на файл с командами предыдущей задачи:

y=zad2(x);

Результаты табулирования будем вносить в файл zad3.txt. Для этого с помощью функции **mopen()**(см. описание) открываем этот файл, с указанием имени файла, в которой будут записаны данные, и режима работы с файлом “a+”:

FileID=mopen('zad3.txt','a+');

Описание функции mopen():

FileID=mopen(file,mode)

file – строка, в которой хранится имя файла,

mode – режим работы с файлом:

’r’ – текстовый файл открывается в режиме чтения;

’rb’ – двоичный файл открывается в режиме чтения;

’w’ – открывается пустой текстовый файл, который предназначен только для записи информации;

’wb’ – открывается пустой двоичный файл, который предназначен только для записи информации;

’a’ – открывается текстовый файл, который будет использоваться для добавления данных в конец файла; если файла нет, он будет создан;

’ab’ – открывается двоичный файл, который будет использоваться

для добавления данных в конец файла; если файла нет, он будет создан;

'r+' – открывается текстовый файл, который будет использоваться в режиме чтения и записи;

'rb+' – открывается двоичный файл, который будет использоваться в режиме чтения и записи;

'w+' – создаваемый пустой текстовый файл предназначен для чтения и записи информации;

'wb+' – создаваемый пустой двоичный файл предназначен для чтения и записи информации;

'a+' – открываемый текстовый файл будет использоваться для добавления данных в конец файла и чтения данных; если файла нет, он будет создан;

'ab+' – открываемый двоичный файл будет использоваться для добавления данных в конец файла и чтения данных; если файла нет, он будет создан.

Далее в файл записываем данные с помощью функции **mfprintf()** (см. описание). Сначала мы записываем параметры (табл. 1) для заголовков будущей таблицы данных, далее указываем цикл от 1 до длины строки переменной **x**, затем записываем параметры для вывода значений (табл. 5, 6) и выводимые переменные, и закрываем файл с помощью функции **mclose()**:

```
mfprintf(FileID,'%4s%5s%9s\n','№','x','f(x)');
for i=1:length(x)
   mfprintf(FileID,'%4.0f%8.4f%10.4f\n',(i),x(i),y(i));
end
mclose(FileID);
```

Описание функции mfprintf():

mfprintf(FileID, s1, s2);

FileID – идентификатор файла (значение идентификатора возвращается функцией topen), s1 – строка вывода, s2 – список выводимых переменных.

В строке вывода вместо выводимых переменных указывается строка

преобразования следующего вида:

%[ширина][.точность] тип.

Далее автоматически создается текстовый файл с результатами табулирования (рис. 5).

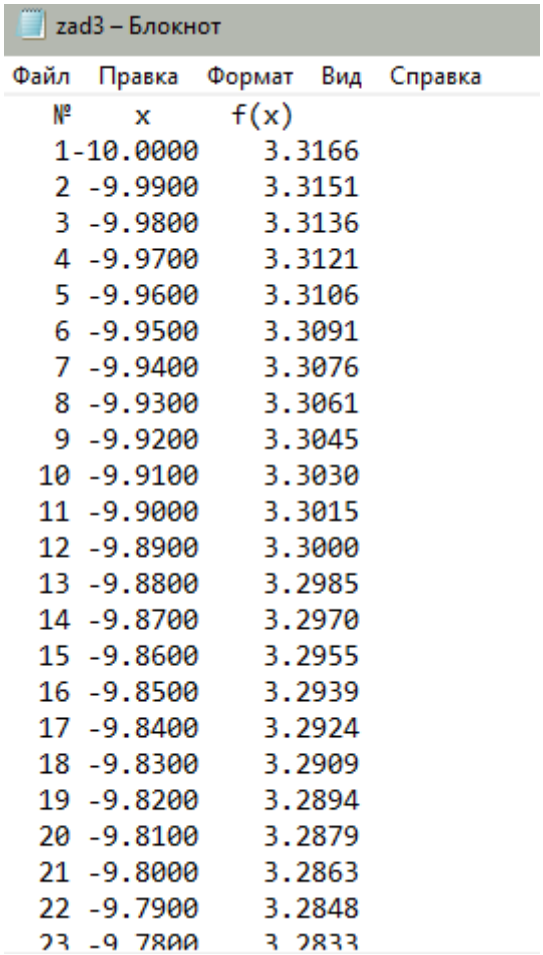

Рис. 5. *.txt-файл с результатами табулирования

Таблица 5

Некоторые специальные символы

Символ	Назначение
\b	Сдвиг текущей позиции влево
\n	Перевод строки
\r	Перевод в начало строки, не переходя на новую строку
\t	Горизонтальная табуляция
\’	Символ одинарной кавычки
\’’	Символ двойной кавычки
\?	Символ «?»

Таблица 6

Значение параметров строки преобразования

Ширина	
n	Ширина поля вывода. Если *n* позиций недостаточно, то поле вывода расширяется до минимально необходимого. Незаполненные позиции заполняются пробелами.
0n	То же, что и n, но незаполненные позиции заполняются нулями.
Точность	
ничего	Точность по умолчанию
n	Для типов e, E, f выводить *n* знаков после десятичной точки
Тип	
c	При вводе символьный тип char, при выводе один байт.
d, i	Десятичное со знаком
i	Десятичное со знаком
o	Восьмеричное int unsigned
u	Десятичное без знака
x, X	Шестнадцатеричное int unsigned, при x используются символы a-f, при X — A-F.
f	Значение со знаком вида [-]dddd.dddd
e	Значение со знаком вида [-]d.ddddе[+|-]ddd
E	Значение со знаком вида [-]d.ddddE[+|-]ddd
g	Значение со знаком типа e или f в зависимости от значения и точности
G	Значение со знаком типа E или F в зависимости от значения и точности
s	Строка символов

Задача 4. Поиск корней функции

Найти глобальный максимум целевой функции вида:

$$Y = -2 * x^4 + x^3 - 3 * x + 10$$

на интервале [–15; 15].

Найти один из корней этой функции.

Построить её график, отобразив точками местоположение максимума и корня.

Реализация в SciLab:

Создаем файл с кодом, определяющий заданную функцию:

```
function [y]=zad4(x)
   for i=1^length(x)
y(i)=-2*x(i)^4+x(i)^3-3*x(i)+10;
end
end
```

Для задания интервала [-15;15] создаем переменную x и присваиваем ей значение вектора-строки, представляющего собой ряд чисел от -15 до 15 с

шагом 0.01:

x=-15:0.1:15;

Вычисляем значение заданной функции на интервале x:

y=zad4(x);

Находим максимум функции и максимальное значение x, соответствующее точке y:

ymax=max(y);
xmax=x(find(y==ymax));

Для того, чтобы вычислить корень уравнения, запишем оператор **deff** в виде:

deff('[f]=y(x)','f=-2*x^4+x^3-3*x+10');

Оператор deff:

deff('[имя1,...,имяN] = имя_функции(переменная_1,...,переменная_M)', 'имя1=выражение1;...;имяN=выражениеN')

имя1,...,имяN – список выходных параметров, то есть переменных, которым будет присвоен конечный результат вычислений,

имя_функции – имя с которым эта функция будет вызываться,

переменная_1,...,переменная_M – входные параметры.

С помощью функции fsolve() найдем корень уравнения:

A=fsolve(0,y);

Описание функции fsolve:

fsolve(x0,f)

x0 – начальное приближение,

f – функция, описывающая левую часть уравнения y(x) = 0.

Строим график функции, точки корня и максимума функции (рис. 6):

plot(x,y);
plot2d(0,A,-3);
plot2d(xmax,ymax,-4);

Указываем вывод сетки, название графика и осей, подписываем легенду:

xgrid();

```
xtitle('График функции f(x)','ось x','ось y');
legend('y=f(x)','Корень функции','Максимум функции')
```

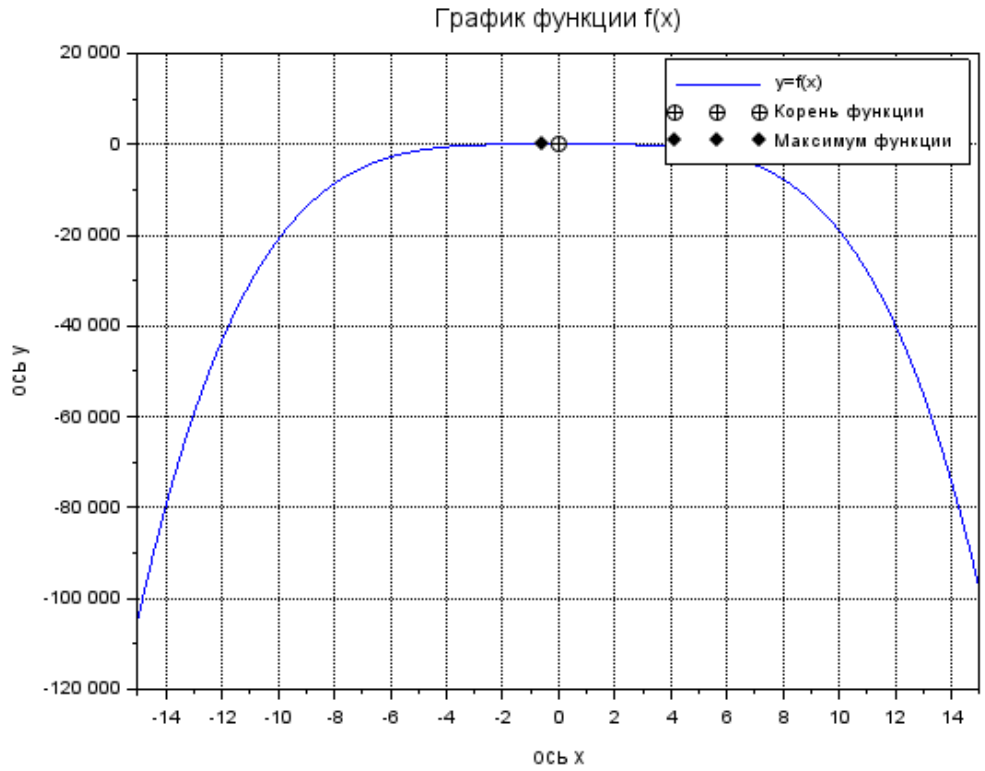

Рис. 6. График функции f(x)с указанием точек максимума и корней функции

После исполнения кода, в командном окне выводим координаты точки максимума [xmax; ymax] и корня функции (A, 0):

```
-> [xmax,ymax]
 ans  =
  -0.6   11.3248
--> A
 A  =
   1.4432002
```

Задача 5.

Найти коэффициенты полиномов 3-й степени: $y = a_0 + a_1 x + a_2 x^2 + a_3 x^3$ и 2-й степени: $y = b_0 + b_1 x + b_2 x^2$ методом наименьших квадратов (МНК).

Построить графики подобранных зависимостей (в одном окне) и подписать оси и легенду.

Обозначить на графиках экспериментальные точки.

Таблица 7

Исходные данные

X (время замера)					Y (эксперимент)				
0	50	100	150	200	12	25	23	54	149
250	300	350	400	450	88	230	200	226	295
500	550	600	650	700	511	351	440	594	532
750	800	850	900	950	602	592	590	590	667

Описание задачи:

Идея метода наименьших квадратов заключается в том, что функцию $Y = f(x, a_0, a_1, \ldots, a_k)$ необходимо подобрать таким образом, чтобы сумма квадратов отклонений измеренных значений y_i от расчетных Y_i была наименьшей:

$$S = \sum_{i=1}^{n}(y_i - f(x_i, a_0, a_1, \ldots, a_k))^2 \rightarrow min \quad (1)$$

Задача сводится к определению коэффициентов a_i из условия (1). Для реализации этой задачи в Scilab предусмотрена функция [a,S]=datafit(F,z,c).

Описание функции [a,S]=datafit(F,z,c):

F – функция, параметры которой необходимо подобрать;

z – матрица исходных данных;

c – вектор начальных приближений;

a – вектор коэффициентов;

S – сумма квадратов отклонений измеренных значений от расчетных.

Реализация в SciLab:

Создадим файл в SciNotes.

Введем исходные данные (табл. 7) в виде векторов:

x=[0,50,100,150,200,250,300,350,400,450,500,550,600,650,700,750,800,850,900,950];

y=[12,25,23,54,149,88,230,200,226,295,511,351,440,594,532,602,592,590,590,667];

Далее данные объединим в матрицу:

z=[x;y];

Запишем функцию полинома 2-й степени:

```
function [mr]=F(r, z)
mr=z(2)-r(1)-r(2)*z(1)-r(3)*z(1)^2
```

endfunction

Запишем вектор начальных приближений для полинома 2-й степени (размерность вектора должна совпадать с количеством искомых коэффициентов):

r=[0;0;0];

И запишем функцию для нахождения коэффициентов полинома:

[a2]=datafit(F,z,r)

Запишем функцию полинома 3-й степени:

```
function [zr]=G(c, z)
   zr=z(2)-c(1)-c(2)*z(1)-c(3)*z(1)^2-c(4)*z(1)^3
endfunction
```

Запишем вектор начальных приближений для полинома 3-й степени (размерность вектора должна совпадать с количеством искомых коэффициентов):

c=[0;0;0;0];

И запишем функцию для нахождения коэффициентов полинома:

[a3]=datafit(G,z,c)

Далее строим графики и точки с помощью функции **plot2d()**.

Для начала указываем интервал по оси x:

t=0:50:950;

Вводим формулу полинома 3-й и 2-й степеней в виде:

```
Ptc=[a3](1)+[a3](2)*t+[a3](3)*t^2+[a3](4)*t^3;
n=[a2](3)*t^2+[a2](2)*t+[a2](1);
```

Строим экспериментальные точки, с указанием переменных и типа точки:

plot2d(x,y,-4);

Строим графики для полиномов 3-й степени и 2-й степени соответственно (рис. 7):

```
plot2d(t,Ptc,style=[color("red")]),
plot2d(t,n,style=[color("green")]);
```

Указываем функцию, которая выводит размерную сетку графика:

xgrid()

Используем функцию **xtitle()** для указания названия графика и названия осей:

xtitle('График полиномов 2й и 3й степени','Ось x','Ось y',2)

Подписываем для полиномов и точек легенду:

legend("Экспериментальные точки","Полином 3й степени","Полином 2й степени")

Далее файл сохраняем и выполняем.

В командном окне вводим переменную [**a2]** для вывода коэффициентов полинома 2-й степени:

```
--> a
 a  =
  -46.835827
   0.9119183
  -0.0001547
```

Следовательно, уравнение полинома второй степени будет иметь вид:

$$Y_b = -0.0002 * x^2 + 0.9119 * x - 46.801 \quad (2)$$

Вводим в командное окно переменную [**a3]** для вывода коэффициентов полинома 3-й степени:

```
-->a
 a  =
   2.018D-08
   0.0000076
   0.0026385
  -0.0000021
```

Тогда уравнения полинома 3-й степени будет выглядеть следующим образом:

$$Y_a = 2{,}018 * 10^{-8} * x^3 + 0.0000076 * x^2 + 0.0026x - 0.0000021 \quad (3)$$

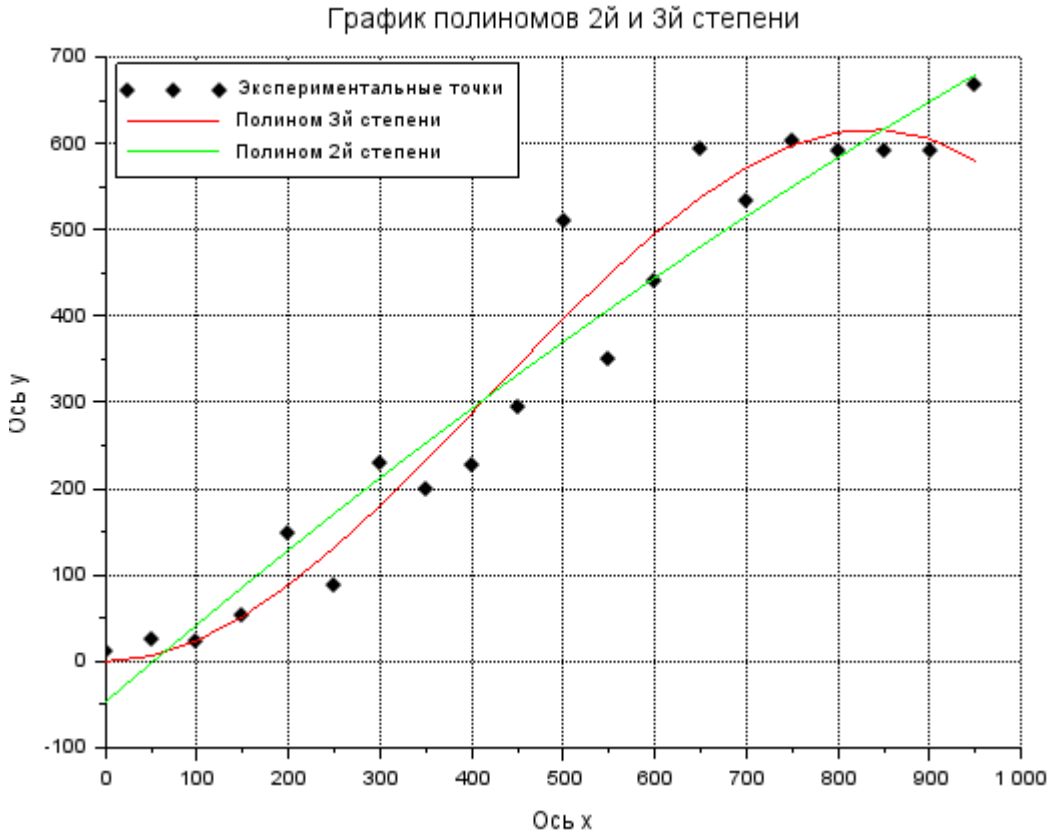

Рис. 7. График полиномов 2-й и 3-й степени

Задача 6. Линейная регрессия

В «Основах химии» Д. И. Менделеева приводятся данные о растворимости азотнокислого натрия $NaNO_3$ в зависимости от температуры воды. Число условных частей $NaNO_3$, растворяющихся в 100 частях воды при соответствующих температурах, представлено в табл. 8. Требуется определить растворимость азотнокислого натрия при температуре 32 градуса в случае линейной зависимости и найти коэффициент и индекс корреляции.

Таблица 8

Данные о растворимости $NaNO_3$ в зависимости от температуры воды

t, ˚C	0	4	10	15	21	29	36	51	68
Реализация	66,7	71	76,3	80,6	85,7	92,9	99,4	113,6	125,1

Описание задачи:

Одной из наиболее часто используемых в методе наименьших квадратов функций является прямая, описываемая уравнением вида $y =$

$a_1 + a_2 * x$, которая называется линией регрессии y на x. Параметры a_1 и a_2 являются коэффициентами регрессии. Показатель, характеризующий тесноту линейной связи между x и y, называемый коэффициентом корреляции, рассчитывается по формуле:

$$r = \frac{\sum_{i=1}^{n}(x_i - M_x)*(y_i - M_y)}{\sqrt{\sum_{i=1}^{n}(x_i - M_x)^2 * \sum_{i=1}^{n}(y_i - M_y)^2}}, M_x = \frac{\sum_{i=1}^{n} x_i}{n} \quad (4)$$

Значение коэффициента корреляции удовлетворяет соотношению $-1 \leq r \leq 1$. Чем меньше отличается абсолютная величина r от единицы, тем ближе к линии регрессии располагаются экспериментальные точки. Если коэффициент корреляции близок к нулю, то это означает, что между x и y отсутствует линейная связь, но может существовать другая, нелинейная, зависимость.

Аналогом коэффициента корреляции r для нелинейных зависимостей является индекс корреляции, рассчитываемый по формуле:

$$R = \sqrt{1 - \frac{\sum_{i=1}^{n}(y_i - Y_I)^2}{\sum_{i=1}^{n}(y_i - M_y)^2}}, \quad (5)$$

где y – экспериментальные значения, Y – значения, найденные методом наименьших квадратов, M_y – среднее значение y. Индекс корреляции по своему абсолютному значению колеблется в пределах от 0 до 1. При функциональной зависимости индекс корреляции равен 1. В случае отсутствии связи R = 0. Если коэффициент корреляции r является мерой тесноты связи только для линейной формы, то индекс корреляции R – и для линейной, и для криволинейной. При прямолинейной связи коэффициент корреляции по своей абсолютной величине равен индексу корреляции.

Для расчета коэффициентов регрессии в Scilab предназначена функция: reglin ().

Описание функции reglin():

a=reglin(x,y)

x и y – экспериментальные данные,

a – вектор коэффициентов линии регрессии a_1 и a_2.

Реализация в SCiLab:

Создаем в SciNotes файл для кода решения задачи.

Присваиваем переменным экспериментальные данные (табл. 8) в виде векторов:

```
x=[0,4,10,15,21,29,36,51,68];
y=[66.7,71,76.3,80.6,85.7,92.9,99.4,113.6,125.1];
```

Рассчитаем коэффициенты регрессии с помощью функции **reglin()**:

```
[a(2),a(1)]=reglin(x,y);
```

Растворимость азотного натрия при температуре 32 градуса:

```
t=32;
a(1)+a(2)*t;
```

Коэффициент корреляции (4):

```
r=sum((x-mean(x)).*(y-mean(y)))/sqrt(sum((x-mean(x))^2)*sum((y-mean(y))^2))
```

Индекс корреляции (5):

```
R=sqrt(1-sum((y-(a(1)+a(2)*x))^2)/sum((y-mean(y))^2))
```

Построение графика экспериментальных данных и линии регрессии (рис. 8), с указанием размерной сетки, названия графика и осей, легенды:

```
t=0:70;
Yt=a(1)+a(2)*t;
plot2d(x,y,-5); plot2d(t,Yt);
xgrid()
xtitle('Зависимость растворимости азотнокислого натрия от температуры воды','Ось x','Ось y')
legend("Экспериментальные точки","Линия регрессии",2)
```

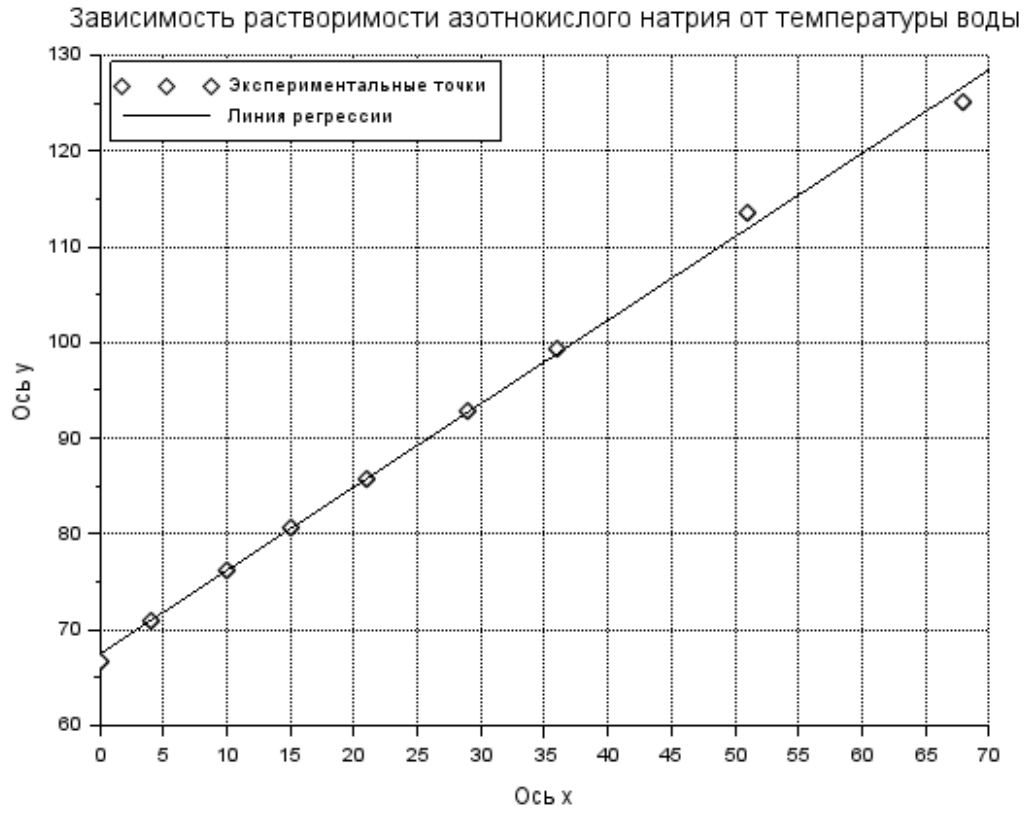

Рис. 8. Зависимость растворимости азотнокислого натрия от температуры воды

После сохранения файла и выполнения кода в командном окне выводим коэффициенты регрессии, коэффициент корреляции и индекс корреляции:

```
-> [a(2),a(1)]
 ans  =
   0.8706404   67.507794
--> r
 r  =
   0.9989549
--> R
 R  =
   0.9989549
```