МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГБОУ ВО "Пензенский государственный университет
архитектуры и строительства"

Прикладная математика

Методические указания к выполнению лабораторных работ для студентов высших учебных заведений, обучающихся по направлению 08.04.01 «Строительство»

Пенза 2020

УДК 681.32
ББК 32.973-018.1
М74

Авторы: Т.А. Глебова, доцент;
М.А. Чиркина, к.т.н., доцент
И.С. Пышкина, к.т.н., доцент

Рецензент: к.т.н., доцент кафедры ИВС ПГУАС Васин Л.А.

М74 Прикладная математикаи: Методические указания / Т.А. Глебова, М.А. Чиркина, И.С. Пышкина.– Пенза: ПГУАС, 2020. –37 с.: ил.

Рассмотрены математические методы решения задач в научных исследованиях. Пособие направлено на формирование профессиональных компетенций, предусмотренных рабочей программой по дисциплине «Прикладная математика». Материал способствует формированию знаний и практических навыков в области применения численных методов решения научных задач.
Предназначено для студентов высших учебных заведений, обучающихся по направлению 08.04.01 «Строительство».

.

Оглавление

Предисловие

Цель дисциплины ”Прикладная математика” - изучение мет
приложений фундаментальных научных достижений в матем
информатике к решению прикладных задач с учетом возра
возможностей компьютерной техники и современных комп

технологий.

Выполнение лабораторных работ направлено на формирование и проверку освоения компетенций, предусмотренных рабочей программой дисциплины

В результате изучения дисциплины (модуля) обучающийся должен:

Знать:

- – о точных и приближенных методах решения;
- – о связи задач дифференциального и интегрального исчисления;
- – о математическом моделировании;
- – современные тенденции развития, научные и прикладные достижения прикладной математики и информатики

Уметь:

- – применять теорию дифференциальных уравнений в частных производных в решении научных задач;
- – применять математические модели для типичных базовых задач и проводить необходимые расчеты в рамках построенных моделей;
- – осуществлять концептуальный анализ и формирование онтологического базиса при решении научных и прикладных задач в области строительства

Иметь навыки*:*

- – использования математической символики для выражении количественных и качественных отношений объектов;
- – применения математических методов в прикладных задачах;
- – использования аналитического и приближенного решения дифференциальных уравнений в частных производных;
- – владения основами методологии и научного познания и системного подхода при изучении различных уровней организации материи, информации, пространства и времени.

Материал дисциплины предназначен для использования в курсах, связанных с постановкой и решением реальных задач (например, различные разделы теоретической и прикладной механики, расчет строительных конструкций), с построением математических моделей физических процессов, верификацией гипотез, теоретических моделей и т.д. Знания, полученные в процессе изучения дисциплины, могут быть использованы в курсах профильной направленности.

К **основным задачам**, дисциплины следует отнести:

– изучение основных понятий дисциплины;

– изучение современных численных методов;

– изучение возможностей различных численных алгоритмов, методов, особенностей их применения при решении прикладных задач;

– различные численные методы, их характеристики и свойства, особенности применения этих методов для решения практических задач.

Объектом изучения научной и учебной дисциплины " Прикладная математикаи " являются исследование практических задач в области строительства.

Лабораторная работа №1

Тема: **Выполнения математических, инженерных и технических расчетов в системе компьютерной математики SciLab.**

1. **Цель работы**

Закрепление, углубление и совершенствование знаний и практических навыков работы на персональном компьютере с использованием современных компьютерных технологий.

2. **Учебные вопросы, подлежащие рассмотрению:**

– Основы работы в Scilab
– Функции в Scilab
– Массивы в Scilab
– Построение двумерных графиков

3. **Порядок выполнения работы**

Основы работы в Scilab

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.

После запуска Scilab на экране появится основное окно приложения. Окно содержит меню, панель инструментов и рабочую область. Признаком того, что система готова к выполнению команды, является наличие знака приглашения > в командной строке, после которого расположен активный (мигающий) курсор. Ввод команд в Scilab осуществляется с клавиатуры. Нажатие клавиши Enter заставляет систему выполнить команду и вывести результат (рис. 1).

Исправить что-либо в области просмотра уже выполненных команд нельзя. Однако все ранее вводимые команды сохраняются в специальной области памяти, и их можно просмотреть с помощью клавиш клавиатуры | |. Например, нажатие клавиши | один раз в пустой в командной строке отразит предыдущую выполненную команду, которую можно отредактировать и запустить заново.

Если набираемое выражение очень длинное, его можно продолжить на следующей строке, для этого в месте прерывания нужно набрать три точки без пробелов, а затем продолжить набор выражения на следующей строке.

Для подавления вывода на экран результатов промежуточных вычислений, в конце команды используется точку с запятой «;».

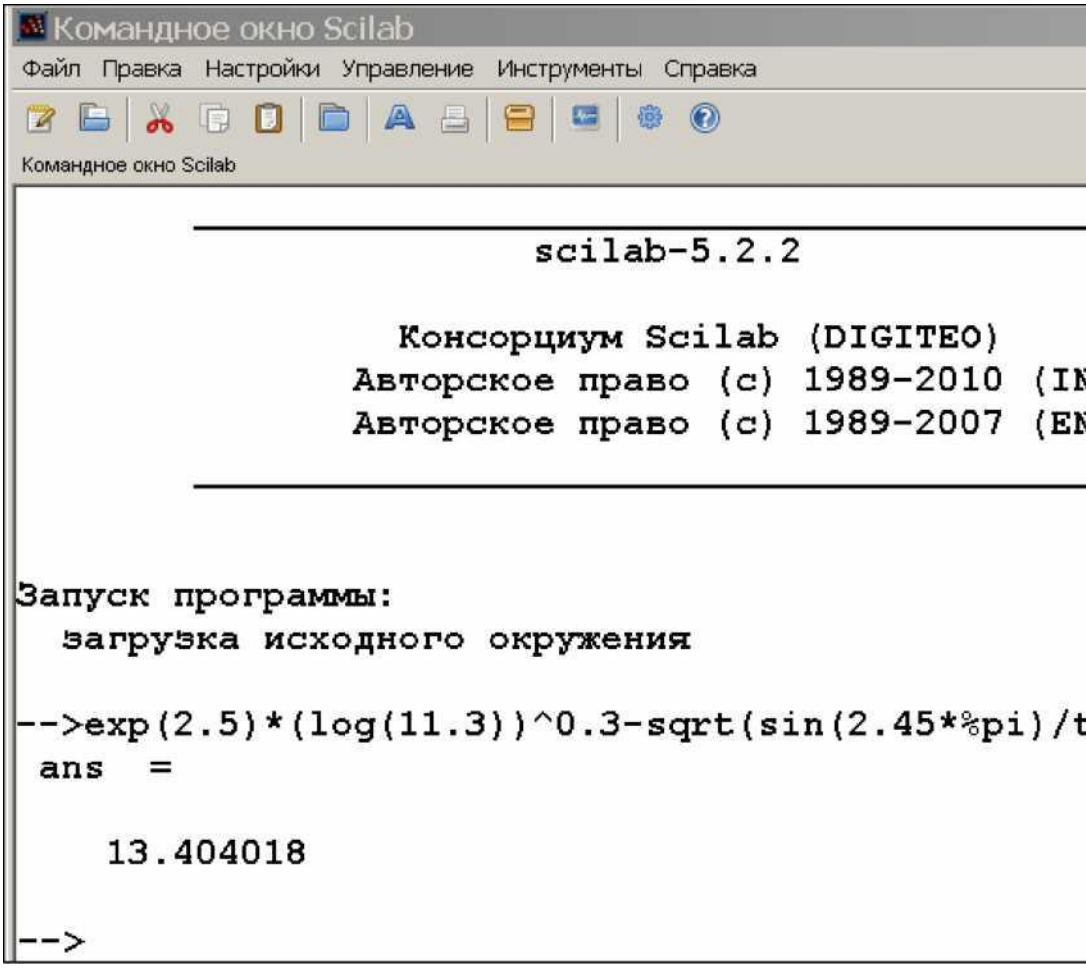


Рис. 1. Командная строка SciLab

Арифметические операции выполняются в обычном порядке: свойственном языкам программирования:

- действия в скобках;
- вычисление функции;
- возведение в степень^;
- умножение * и деление слева направо / (5/2=2.5) и справа налево \ (5\2=0.4);
- сложение и вычитание +, -.

Для изменения порядка вычислений используйте скобки.

Переменные в Scilab. Любая переменная до использования в формулах и выражениях должна быть определена. Для этого используется оператор присваивания «=», который в общем виде записывается

Имя переменной = Значение выражения

Действие оператора: в переменную, имя которой указано слева, будет записано значение выражения, указанного справа.

Примечание 1. Имя переменной не должно совпадать с именами встроенных процедур, функций и встроенных переменных системы и может содержать до 24 символов.

Примечание 2. Система различает большие и малые буквы в именах переменных, т.е. ABC, abc, Abc, aBc - это имена разных переменных.

Примечание 3. Выражение в правой части оператора присваивания может быть числом, арифметическим выражением, строкой символов или символьным выражением. Если переменная является символьной, то выражение в правой части оператора присваивания следует брать в одинарные кавычки.

Примечание 4. Если для сохранения результата операции переменная пользователем не назначена, то SciLab определяет временную переменную *ans*, которую можно использовать в дальнейших вычислениях.

Системные переменные в SciLab начинаются с символа %:

%i - мнимая единица;

%pi - число пи (3.1415926);

%e - экспонента 1 (2.7182818);

%inf - машинный символ бесконечности;

%NaN - неопределенный результат (0/0 и т.п.);

%eps - условный ноль (2.220E-16)

Вывод в Scilab. По умолчанию результат выводится с восемью значащими цифрами после запятой. Для контроля вывода применяют команду printf с заданным форматом, который соответствует правилам, принятым для этой команды в языке С (рис. 2).

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. Определения

всех переменных и функций, входящих в текущую сессию можно сохранить в виде файла, саму сессию сохранить нельзя.

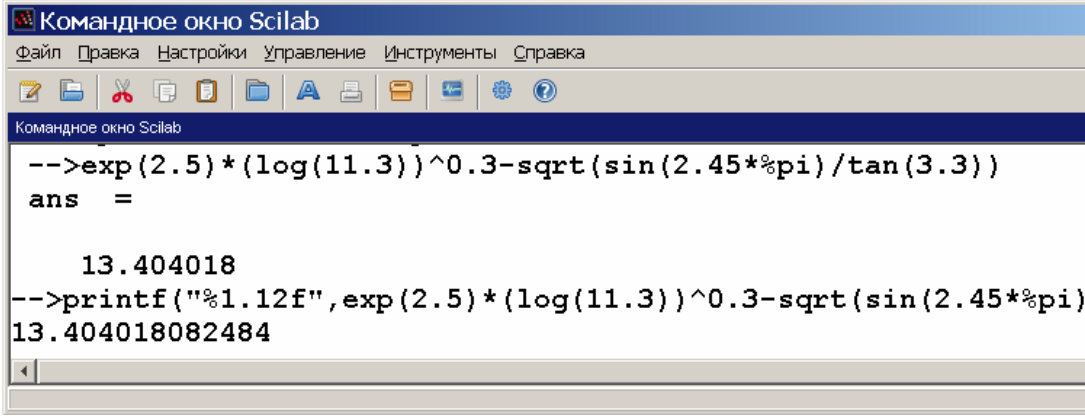


Рис. 2. Форматированный вывод (12 знаков после запятой)

Редактирование и отладка файлов-сценариев

Файл-сценарий - это список команд Scilab, сохраненный на диске. Для подготовки, редактирования и отладки файлов-сценариев служит специальный редактор SciPad, который можно вызвать, нажав кнопку на панели инструментов (рис. 3). В результате работы этой команды будет создан новый файл-сценарий. По умолчанию он имеет имя Untitledl. sce.

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.

Для сохранения введенной информации необходимо выполнить команду File →Save As... (Файл →Сохранить как...) из меню редактора. Файлы-сценарии сохраняют с расширением . sce.

Открыть ранее созданный файл можно с помощью команды главного меню File → Open (Файл→ Открыть).

Выполнить операторы файла-сценария можно из меню редактора SciPad с помощью команды Execute → Load into Scilab (Выполнение →Загрузить в Scilab).

’V Console

1Ш

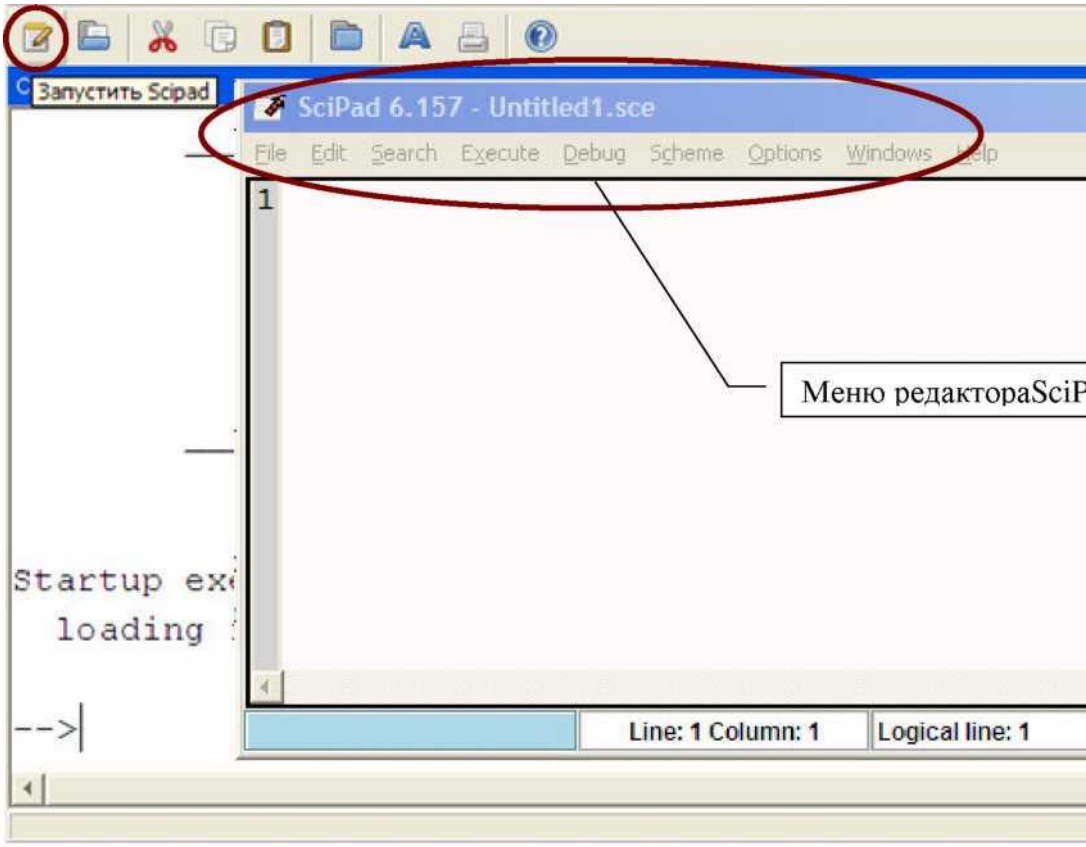


Рис. 3. Редактор SciPad

Примечание 1. Точка с запятой «;» ставится после тех команд, которые не требуют вывода значений.

Примечание 2. Строка после символов // не воспринимается как команда - это текстовый комментарий.

Встроенные функции

В общем виде обращение к функции в Scilab имеет вид:

имя_переменной = имя_функции(пер_1 [, пер_2, ...])

- имя_переменной - переменная, в которую будут записаны результаты работы функции: этот параметр может отсутствовать, тогда значение, вычисленное функцией, будет присвоено системной переменной *ans*;

- имя_функции - имя встроенной функции или ранее созданной пользователем;

- пер_1, пер_2,... - список аргументов функции.

В табл. 1 приведены наиболее часто используемые элементарные математические функции.

Табл. 1. Элементарные математические функции

Функция	Описание функции	Функция	Описание функ
Тригонометрические			
sin(x)	синус числа x	asin(x)	арксинус числа
cos(x)	косинус числа x	acos(x)	арккосинус чис
tan(x)	тангенс числа x	atan(x)	арктангенс числ
cotg(x)	котангенс числа x		
Экспоненциальные			
exp(x)	экспонента числа x	log(x)	натуральный ло числа x
Другие			
sqrt(x)	корень квадратный из числа x	log10(x)	десятичный лог от числа x
abs(x)	модуль числа x	log2(x)	логарифм по ос два от числа x

Функции, определенные пользователем

Функция - это именованная логически законченная группа команд, которую можно вызывать для выполнения по имени, и предназначена для неоднократного использования. Функция имеет входные параметры и не выполняется без их предварительного задания. Задать вид функции можно с помощью конструкции function ... endfunction:

function[имя1,...,имяN] = имя_функции
(переменная_1,...,переменная_M)
тело функции
endfunction

- 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.

Все имена переменных внутри функции, а также имена из списка входных и выходных параметров воспринимаются системой как локальные, т.е. считаются определенными только внутри функции.

Пример. Вычислить площадь четырехгранника, если даны длины его ребер. Решение приведено на рис. 4.

```
-->function [r]=area(a,b,c)
-->  p=(a+b+c)/2;
-->  //Формула Герона
-->  r=sqrt(p*(p-a)*(p-b)*(p-c));
-->endfunction

-->S=area(2,3,3)+area(3,2,3)+area(3,3,2)+area
 S  =

    10.217332
```

Рис.4

Массивы в Scilab

Массив - пронумерованная совокупность однородных данных, состоящая из фиксированного числа элементов, обозначенная одним именем. Доступ к отдельным элементам массива осуществляется по целочисленному индексу, то есть по номеру элемента в массиве. В зависимости от количества индексов, определяющих положение элемента в массиве, массивы разделяют на одномерные (вектора- строки, вектора-столбцы), двумерные (матрицы) и многомерные.

SciLab представляет все данные в виде массивов, даже переменная - это двумерный массив с размерностью один на один.

Работа с векторами. Вектора - это одномерные (линейные) массивы, в которых позиция каждого элемента задается единственным числом - его номером. При задании векторов элементы разделяются пробелами, запятой (,) или точкой с запятой (;):

a1=[3 4 9 2] - вектор-строка

a1=[3, 4, 9, 2]- вектор-строка

a1=[3; 4; 9; 2] - вектор-столбец.

Доступ к элементам вектора осуществляется заданием его индекса в круглых скобках после имени.

Если значения элементов вектора являются арифметической прогрессией, то их можно задать с помощью операции «:». С помощью функции length можно определить, сколько элементов попало в вектор.

Пример. Сформировать одномерный массив чисел в диапазоне от 3.7 до 8.947 с приращением 0.3. Решение приведено на рис. 5.

```
-->Mas=3.7:0.3:8.947
 Mas  =

         column 1 to 8

    3.7    4.    4.3    4.6    4.9    5.2    5.

         column  9 to 16

    6.1    6.4    6.7    7.    7.3    7.6    7.

         column 17 to 18

    8.5    8.8

-->length(Mas)
 ans  =

    18.
```

Рис. 5. Формирование одномерного массива

Поэлементные операции с векторами

Чтобы выполнить поэлементное умножение, деление, возведение в степень векторов, используют следующие знаки «.*», «./», «.^» (без пробелов!). В результате выполнения этих операций получается тоже вектор.

Примечание. Умножение и деление вектора на число выполняется с помощью обычных знаков «*» «/» без точки.

Пример. Протабулировать функцию $y(x) = e^{-x} \sin 10x$ на отрезке [0,1] с шагом 0.05.

Для решения задачи необходимо вначале задать вектор, содержащий значения аргумента X, а затем вычислить элементы вектора функции Y, используя поэлементное умножение!!! Решение представлено на рис. 6.

```
-->x=[0:0.2:1]
 x  =
    0.    0.2    0.4    0.6    0.8    1.
-->y=exp(-x).*sin(10*x)
 y  =
         column 1 to 4
    0.    0.7444698  - 0.5072999  - 0.1533465
         column 5 to 6
    0.4445473  - 0.2001342
```

Рис. 6. Табулирование функции

Работа с матрицами

Для хранения матриц в системе SciLab используются двумерные массивы, имеющие уникальное имя. Доступ к элементам массива осуществляется при помощи двух индексов: номера строки и номера столбца, указанных в круглых скобках, например C(2,3).

Матрицы можно ввести либо по строкам

```
a = [3  1  -1; 2  4  3]
```

либо по столбцам

```
a = [[3;2] [1;4] [-1;3]]
```

Для того чтобы узнать размеры двумерного массива и «геометрию» векторов (вектор-столбцы или вектор-строки), нужно использовать функцию size:

```
size(a)
 ans =
```

2 3

Заданная матрица a содержит две строки и три столбца.

Примечание. Над массивами одинаковых размеров допускаются операции сложения (+) и вычитания (-). Для поэлементного перемножения используется знак «.*». Для поэлементного деления массивов - знаки «./» и «\.». Для поэлементного возведения в степень - знак «.^».

Привычные знаки «*», «/» предназначены в системе SciLab для матричных операций.

Транспонирование матрицы, так же как и векторов производится с помощью символов «.’».

Построение двумерных графиков

В двумерных графиках положение каждой точки задается двумя величинами (координатами по оси X и по оси Y), поэтому прежде, чем строить график необходимо:

1) сформировать массив x;

2) создать массив y, вычислив значение функции для каждого значения массива x.

Использование функции plot. Обращение к функции имеет вид:

plot(x,y,[xcap, ycap, caption]),

где

x - массив абсцисс;

y - массив ординат;

xcap, *ycap*, *caption* - подписи осей *X, Y* и графика соответственно.

Пример. Построить график функции $y = sin(cos(x))$ на интервале $[-2\pi; 2\pi]$ с шагом 0,1.

Необходимо:

1) сформировать массив *x*;

2) создать массив *y*, вычислив значение функции для каждого значения массива *x*;

3) построить график функции с помощью функции plot (рис. 7).

```
-->x=[-2*%pi:0.1:2*
-->y=sin(cos(x));
-->plot(x,y)
```

```
-->x=[-2*%pi:0.1:2*%pi];
-->y=sin(cos(x));
-->plot(x,y)
```

Рис. 7

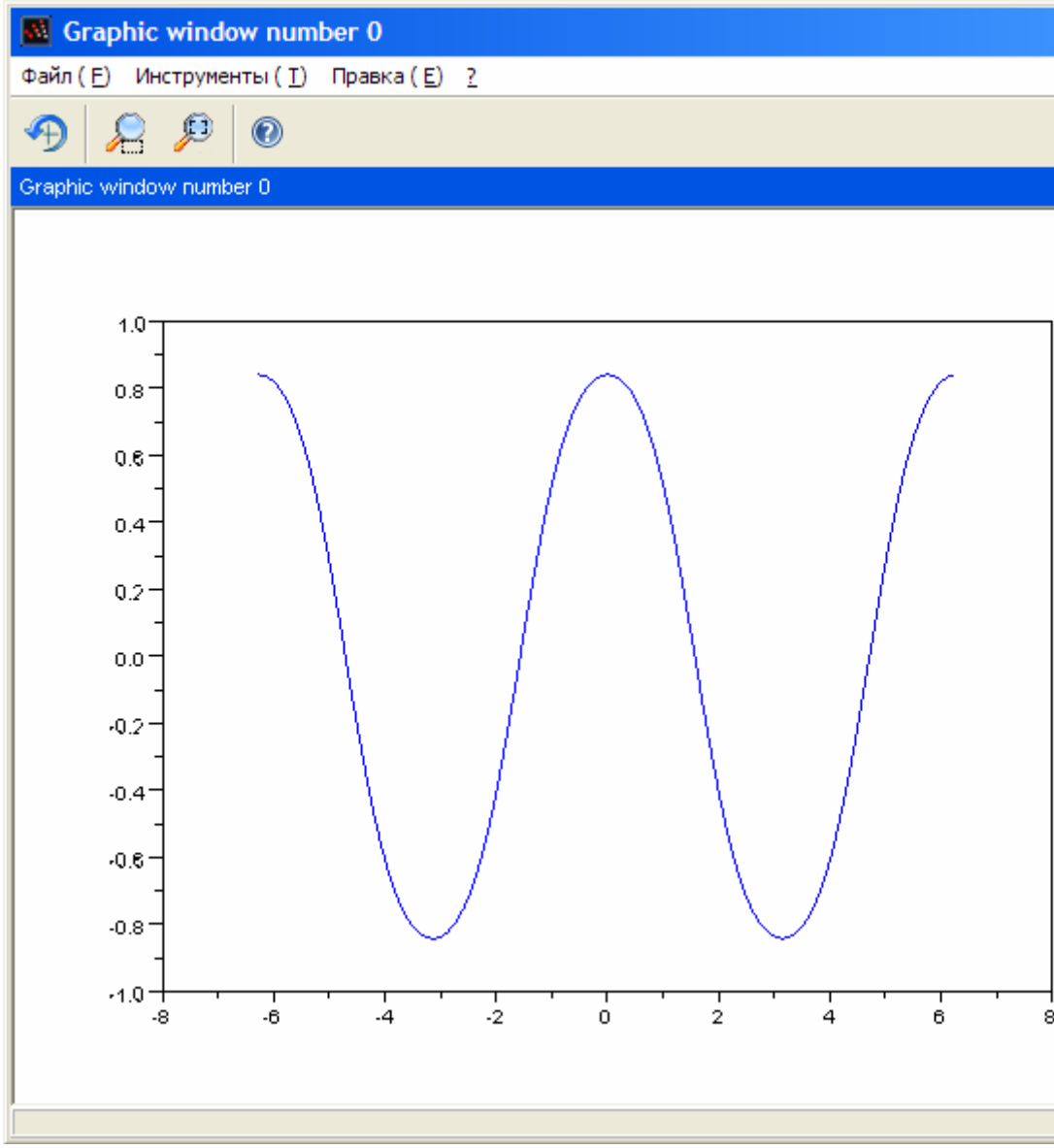


Рис. 8. Построенный график функции

Форматирования графика. В Scilab внешний вид графика можно изменять, используя дополнительный параметр

функции plot: строку из трех символов, заключенных в апострофы. Эти символы задают цвет линии, тип маркера и тип линии соответственно (табл. 2).

Табл. 2. Некоторые символы для задания внешнего вида функции

Цвет		Маркер		Линия	
Символ	Цвет	Символ	Тип маркера	Символ	Тип ли
r	красный	*	звездочка	-	сплош
g	зеленый	s	квадрат	-..	пунктир дв
b	синий	d	ромб	-.	пунктир
k	черный	o	кружок	- -	пунктир

Построение пунктирного графика красного цвета с маркерами в виде *:

```
-->plot(x,y,'r*--')
```

Внешний вид графика можно также изменять, используя функцию *xgrid(),* чтобы задать линии сетки и функцию *xtitle(),* чтобы задать название графика и его осей:

```
-->plot(x,y,'r')
-->xgrid();
-->xtitle('sin(cos(x))','X','Y')
```

Расположение осей графика можно изменить, получив доступ к параметрам осей с помощью команды *gca()* и используя соответствующие команды. Например, для перемещения оси X в начало координат следует использовать:

```
-->a=gca();
-->a.x_location="origin";
```

На рис. 9 приведен отформатированный график.

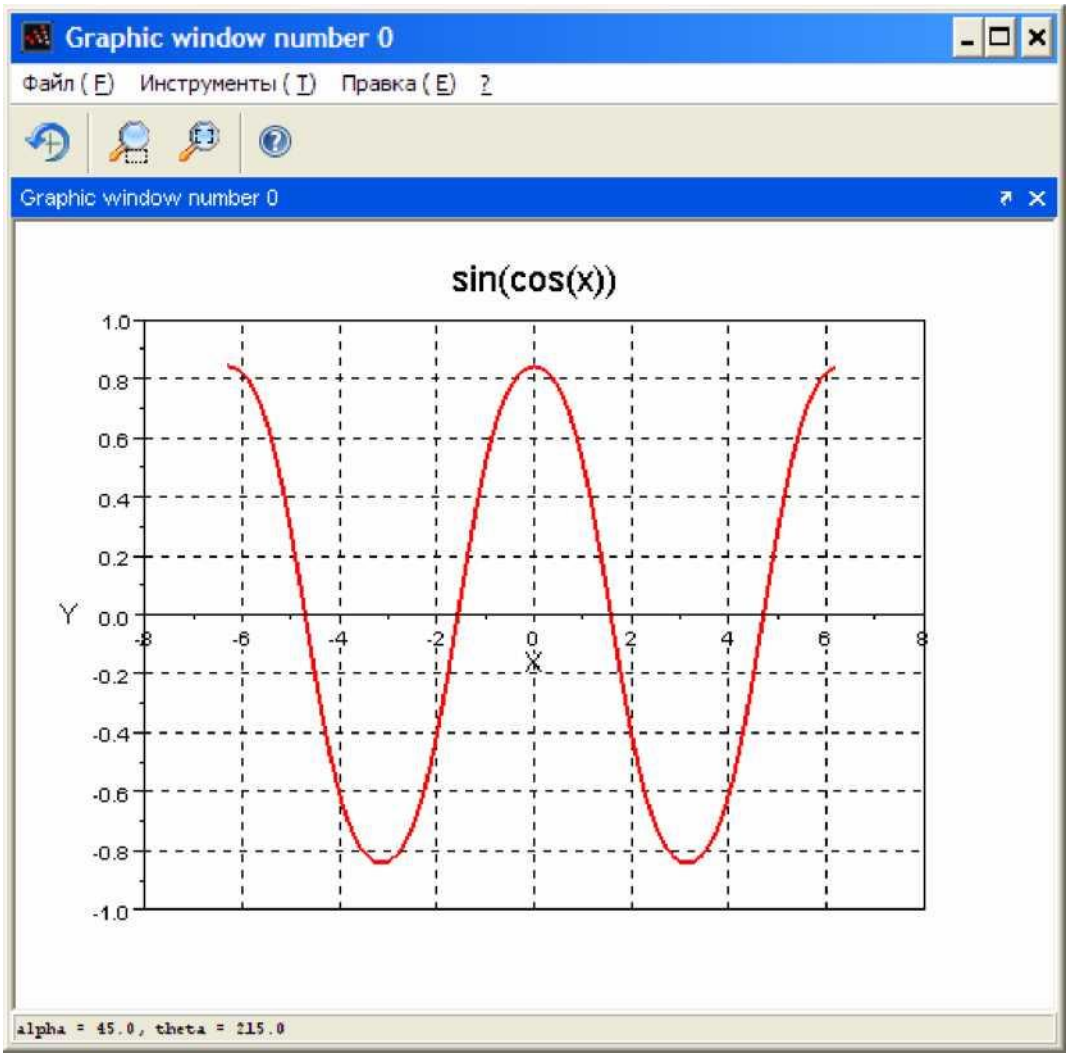


Рис. 9. Отформатированный график

На рис. 10 приведен файл-сценарий построения графика рассматриваемой функции.

```
//Задание значений аргумента
x=[-2*%pi:0.1:2*%pi];
//Вычисление значений функции
y=sin(cos(x));
//Построение графика (цвет красный)
plot(x,y,'r')
//
//Нанесение на график сетки
xgrid();
//Задание заголовков графика и осей
xtitle('sin(cos(x))','X','Y')
//доступ к параметрам осей графика
a=gca();
//Расположение оси X в начале координат
a.x_location="origin";
```

Рис. 10 Файл-сценарий построения графика рассматриваемой функции.

Лабораторная работа №2

Тема: **Решение нелинейных уравнений и систем**.

1. **Цель работы**

Использование методов решения нелинейных уравнений и систем для решения конкретных производственных задач.

2. **Учебные вопросы, подлежащие рассмотрению:**

– Отделение корней уравнения.
– Решение нелинейных уравнений:
✓ метод половинного деления,
✓ метод простых итераций,
✓ метод Ньютона.

3. **Порядок выполнения работы**

Задание 1.

Вычислить наименьший положительный корень заданного уравнения с точностью $\varepsilon=10^{-3}$. Работу провести в три этапа:

1) Провести графическое отделение корней уравнения.

2) Сузить отрезок, полученный графическим способом до отрезка длиной 0.1.

3) Вычислить приближенное решение методом половинного деления.

По итогам выполнения заданий представить корень уравнения, вычисленный с указанной точностью.

Номер варианта соответствует порядковому номеру в списке.

Задание 2.

Найти решение уравнения с точностью $\varepsilon=10^{-4}$, используя метод простой итерации и один из методов Ньютона.

Номер варианта соответствует порядковому номеру в списке.

Исполнение: Освоить реализацию итерационных процессов с использованием логических функций в MS Excel.

Оценка: Использование инструментальных пакетов для решения трансцендентных уравнений.

Методические указания

Общие сведения о решении нелинейного уравнения

Как правило, нелинейное уравнения общего вида $f(x)=0$ невозможно решить аналитически. Для практических задач достаточно найти приближенное значение x, в определенном смысле близкое к точному решению уравнения $x_{точн}$.

В большинстве случаев поиск приближенного решения включает два этапа. На первом этапе *отделяют* корни, т. е. находят такие отрезки, внутри которых находится строго один корень. На втором этапе *уточняют* корень на одном из таких отрезков, т. е. находят его значение с требуемой точностью.

Достигнутая точность может оцениваться либо «по функции» (в найденной точке x, функция достаточно близка к 0, т. е. выполняется условие $|f(x)|\le ef$, где ef требуемая точность по оси ординат), либо «по аргументу» (найден достаточно маленький отрезок $[a,b]$, внутри которого находится корень, т. е. $|b-a|\le ex$, где ex требуемая точность по оси абсцисс).

Графическое отделение корней.

Для того чтобы провести графическое отделение корней, надо построить график функции $y = F(x)$ и визуально определить интервал, на котором находится ровно один корень уравнения.

Уточнение корня методом половинного деления.

Основная идея нахождения приближенного решения заключается в сокращении первоначального интервала, определенного при графическом отделении, до интервала длиной 2 ε. После того, как удалось сократить интервал до заданной величины, можно определить $\tilde{x} = (a + b) / 2$. В этом случае условие $|x^* - \tilde{x}| \leq \varepsilon$ выполнено.

В методе половинного деления сокращение интервала происходит делением отрезка $[a, b]$ пополам и выбора той из половин, которой принадлежит искомый корень уравнения.

Итак, алгоритм численного решения уравнения (1) методом половинного деления заключается в выполнении следующих шагов:

1. определить начальный отрезок $[a, b]$;
2. найти точку с – середину отрезка $[a, b]$, $c = (a + b) / 2$;
3. проверить, какому из отрезков $[a, c]$ или $[c, b]$ принадлежит корень. Легко

видеть, что проверка выполняется так:

если $f(a) * f(c) < 0$, то корень принадлежит отрезка $[a, c]$ и в дальнейшем надо

положить $b = c$, в противном случае корень принадлежит отрезку $[c, b]$ и следует положить $a = c$;

4. если длина отрезка $[a, b]$ больше 2ε, то перейти к пункту 2;
5. закончить вычисления, положив $\tilde{x} = (a + b) / 2$.

Теория электронных таблиц

Для реализации данного алгоритма необходимо воспользоваться логическими функциями. Логические функции предназначены для проверки выполнения условия или для проверки нескольких условий. В отличие от математических функций, при проведении вычислений с логическими функциями мы оперируем понятиями ИСТИНА и ЛОЖЬ.

В общем виде функция, позволяющая учесть при вычислениях условия выглядит так:

ЕСЛИ(*логическое выражение; значение1; значение2*)

логическое выражение – это выражение, принимающее значения ИСТИНА или ЛОЖЬ. Например, С15=1 – это логическое выражение. Если значение в ячейке С15 равно 1, то выражение принимает значение ИСТИНА, иначе – ЛОЖЬ.

Значение1 – это значение, которое заносится в ячейку, если *логическое выражение* равно ИСТИНА; *значение2* – это значение, которое заносится в ячейку, если *логическое выражение* равно ЛОЖЬ.

До семи функций ЕСЛИ могут быть вложены друг в друга в качестве значений аргументов *значение1* и *значение2* для конструирования более сложных проверок. Если любой из аргументов функции ЕСЛИ является массивом, все элементы массива вычисляются при выполнении функции ЕСЛИ.

Microsoft Excel предлагает дополнительные функции, которые можно применять для анализа данных с использованием условий. Например, для вычисления числа появлений текстовой строки или числа в диапазоне ячеек используется функция СЧЁТЕСЛИ. Для вычисления суммы значений, попадающих в интервал, заданный текстовой строкой или числами, используется функция СУММАЕСЛИ.

Для записи логических выражений (условий) служат стандартные операции

сравнения: = (равно), > (больше), < (меньше), >= (больше или равно), <= (меньше или равно), <> (не равно). Если требуется записать более сложные условия, включающие в себя несколько простых условий, то приходится применять такие логические функции, как И, ИЛИ, НЕ.

И(*логическое выражение1; логическое выражение2*; ...)

Функция И будет иметь значение ИСТИНА, если все логические выражения,

перечисленные в скобках, истинны. В противном случае результатом функции И будет значение ЛОЖЬ. Всего можно указать до 30 различных условий.

ИЛИ(*логическое выражение1; логическое выражение2*; ...)

Функция ИЛИ возвращает значение ИСТИНА, если хотя бы один из аргументов имеет значение ИСТИНА и возвращает ЛОЖЬ, если все аргументы имеют значение ЛОЖЬ.

НЕ(логическое выражение)

Если логическое выражение имеет значение ЛОЖЬ, то функция НЕ возвращает значение ИСТИНА; если *логическое выражение* имеет значение ИСТИНА, то функция НЕ возвращает значение ЛОЖЬ.

Методика выполнения

1.1 Отделение корней

Отделение корней может производиться сочетанием графического и аналитического исследования функции. Такое исследование опирается на теорему Вейерштрасса, в соответствии с которой для непрерывной на отрезке *[a,b]* функции *f(x)* и любого числа *y*, отвечающего условию $f(a) \leq y \leq f(b)$, существует на этом отрезке точка *x*, в которой функция равна *y*. Следовательно, для непрерывной функции достаточно найти отрезок, на концах которого функция имеет разные знаки, и можно быть уверенным, что на этом отрезке есть корень уравнения $f(x)=0$.

Для ряда методов уточнения желательно, чтобы найденный на первом этапе отрезок содержал только один корень уравнения. Это условие выполняется, если функция на отрезке монотонна. Монотонность, можно проверить либо по графику функции, либо по знаку производной.

Пример Найти с точностью до целых все корни нелинейного уравнения $y(x)=x^3 - 10x + 7=0$

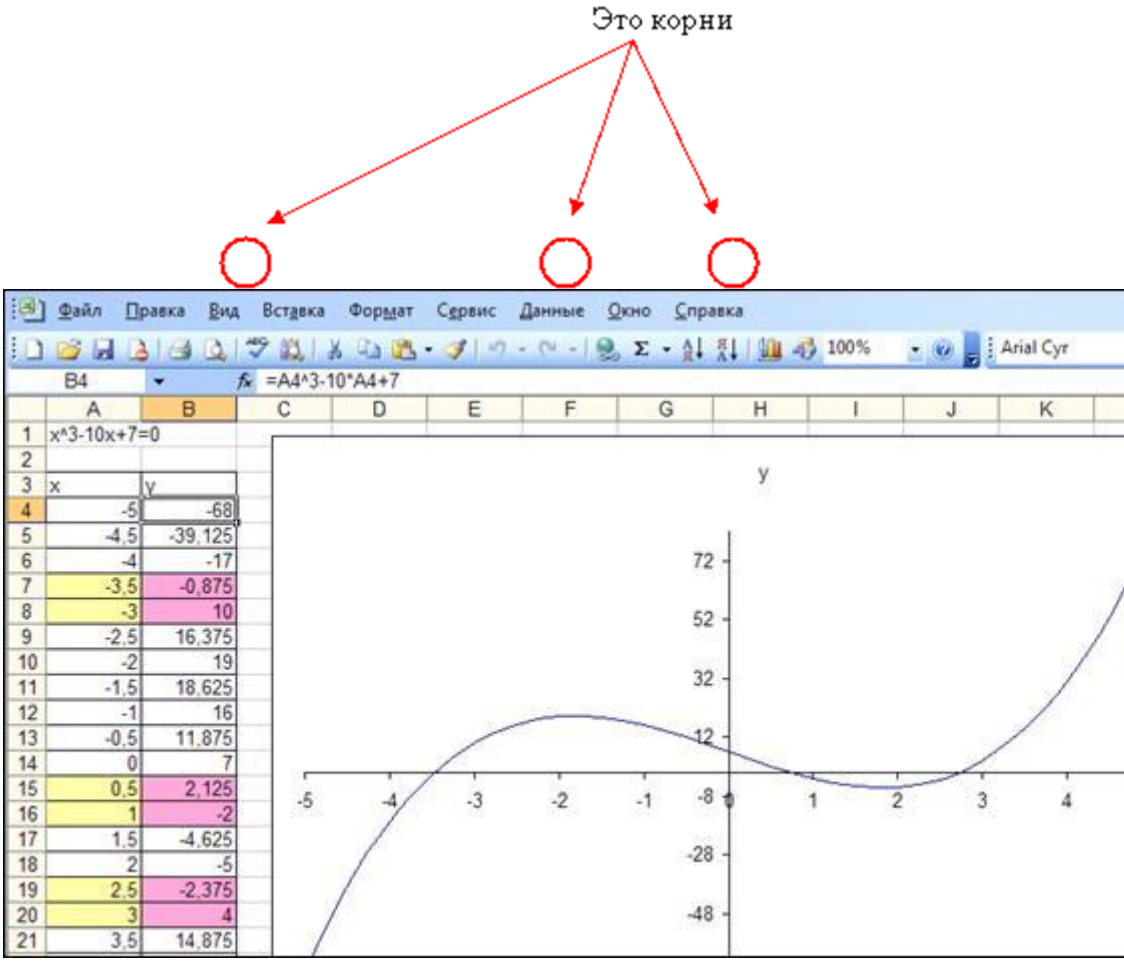
а) построив таблицу и

б) построив график.

Решение

Создадим в Excel таблицу, содержащую аргументы и значения функции и по ней построим *точечную диаграмму*. На рисунке 1 приведен снимок решения.

На графике видно, что уравнение имеет три корня, принадлежащие отрезкам [-4, -3], [0, 1] и [2, 3]. Эти отрезки можно выявить и наблюдая за сменой знаков функции в таблице. По построенному графику можно сделать вывод, что на указанных отрезках функция *f(x)* монотонна и, следовательно, на каждом из них содержится только по одному корню.



	A	B
1	x^3-10x+7=0	
2		
3	x	y
4	-5	-68
5	-4,5	-39,125
6	-4	-17
7	-3,5	-0,875
8	-3	10
9	-2,5	16,375
10	-2	19
11	-1,5	18,625
12	-1	16
13	-0,5	11,875
14	0	7
15	0,5	2,125
16	1	-2
17	1,5	-4,625
18	2	-5
19	2,5	-2,375
20	3	4
21	3,5	14,875

Рисунок 1 – Таблица и график для отделения корней нелинейного уравнения

Найти корень уравнения на выделенном отрезке, используя опции «Подбор параметра» и «Поиск решения».

1.2 Метод деления отрезка пополам

1.2 Метод деления отрезка пополам

В этом методе на каждом шаге отрезок делится на две равные части. Затем сравнивают знаки функции на концах каждой из двух половинок (например, по знаку произведения значений функций на концах), определяют ту из них, в которой содержится решение (знаки функции на концах должны быть разные), и. сужают отрезок, перенося в найденную точку его границу (*a* или *b*). Условием окончания служит малость отрезка, где содержится корень («точность по *x*»), либо близость к 0 значения функции в средине отрезка («точность по y»). Решением уравнения считают середину отрезка, найденного на последнем шаге.

Пример. Построить таблицу для уточнения корня уравнения $x^3 - 10x + 7 = 0$ на отрезке *[-4, -3]* методом деления отрезка пополам. Определить сколько шагов надо сделать методом деления отрезка пополам и какая при этом достигается точность по *x,* для достижения точности по *y*, равной 0,01.

Решение

Для решения можно использовать табличный процессор Excel, позволяющий автоматически продолжать строки. На первом шаге заносим в таблицу значения левого и правого концов выбранного начального отрезка и вычисляем значение середины отрезка $c=(a+b)/2$, а затем вводим формулу для вычисления функции в точке a ($f(a)$) и растягиваем (копируем) её для вычисления $f(c)$ и $f(b)$. В последнем столбца вычисляем выражение $(b-a)/2$, характеризующего степень точности вычислений. Все набранные формулы можно скопировать во вторую строку таблицы.

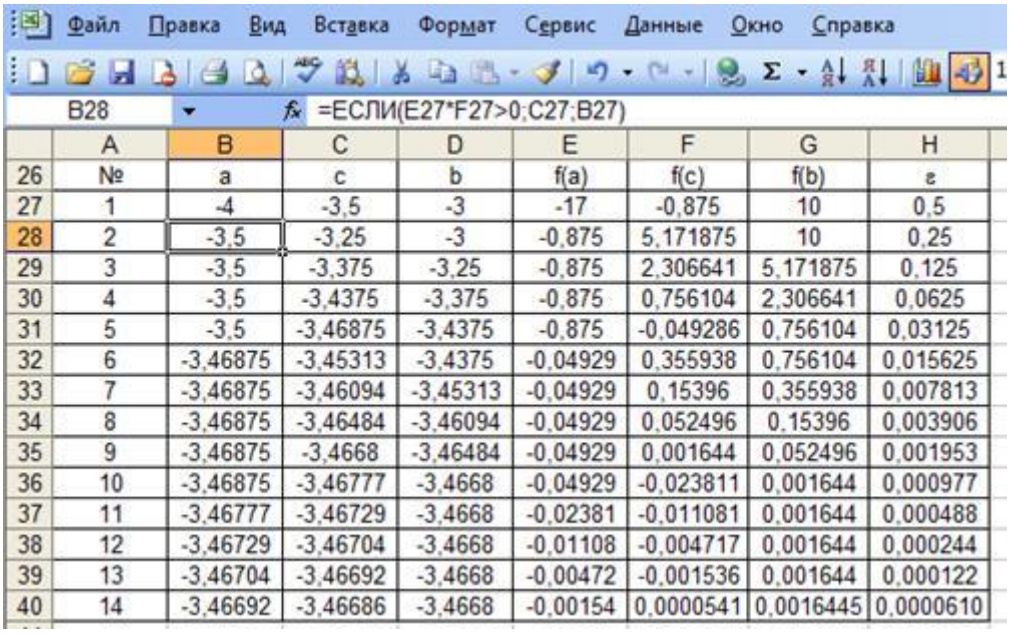
На втором шаге нужно автоматизировать процесс поиска той половины отрезка, где содержится корень. Для этого испльзуется логическая функция ЕСЛИ (*Меню*: Вставка→Функция→Логические). Для нового левого края отрезка мы проверяем истинность условия $f(a)*f(c)>0$, если

оно верно, то мы в качестве нового значения левого конца отрезка берем число *c* (т. к. это условие показывает, что корня на отрезке [*a, c*] нет), иначе оставляем значение *a*. Аналогично, для нового правого края отрезка мы проверяем истинность условия *f(c)*f(b)*>0, если оно верно, то мы в качестве нового значения правого конца отрезка берем число *c* (т. к. это условие показывает, что корня на отрезке [*c, b*] нет), иначе оставляем значение *b*.

Вторую строку таблицы можно продолжить (скопировать) на необходимое число последующих строк.

Итерационный процесс завершается, когда очередное значение в последнем столбце становится меньшим, чем заданный показатель точности ε. При этом, значение середины отрезка в последнем приближении, принимается в качестве приближенного значения искомого корня нелинейного уравнения. На рисунке 6 приведен снимок решения.

Итак, одним из трех корней нелинейного уравнения $x^3 - 10x + 7=0$, найденным с точностью e=0,0001, является $x= - 3,46686$. Как мы видим, он действительно принадлежит отрезку [-4; -3].



	A	B	C	D	E	F	G	H
26	№	a	c	b	f(a)	f(c)	f(b)	ε
27	1	-4	-3,5	-3	-17	-0,875	10	0,5
28	2	-3,5	-3,25	-3	-0,875	5,171875	10	0,25
29	3	-3,5	-3,375	-3,25	-0,875	2,306641	5,171875	0,125
30	4	-3,5	-3,4375	-3,375	-0,875	0,756104	2,306641	0,0625
31	5	-3,5	-3,46875	-3,4375	-0,875	-0,049286	0,756104	0,03125
32	6	-3,46875	-3,45313	-3,4375	-0,04929	0,355938	0,756104	0,015625
33	7	-3,46875	-3,46094	-3,45313	-0,04929	0,15396	0,355938	0,007813
34	8	-3,46875	-3,46484	-3,46094	-0,04929	0,052496	0,15396	0,003906
35	9	-3,46875	-3,4668	-3,46484	-0,04929	0,001644	0,052496	0,001953
36	10	-3,46875	-3,46777	-3,4668	-0,04929	-0,023811	0,001644	0,000977
37	11	-3,46777	-3,46729	-3,4668	-0,02381	-0,011081	0,001644	0,000488
38	12	-3,46729	-3,46704	-3,4668	-0,01108	-0,004717	0,001644	0,000244
39	13	-3,46704	-3,46692	-3,4668	-0,00472	-0,001536	0,001644	0,000122
40	14	-3,46692	-3,46686	-3,4668	-0,00154	0,0000541	0,0016445	0,0000610

Рисунок 12 – Уточнение корня методом деления отрезка пополам в Excel

Уточнение корней методом касательных (Ньютона)

Обширную группу методов уточнения корня представляют *итерационные методы* - методы последовательных приближений. Здесь в отличие от метода дихотомии задается не начальный интервал местонахождения корня, а его начальное приближение.

Наиболее популярным из итерационных методов является *метод Ньютона (метод касательных)*.

Этот метод обеспечивает сходящийся процесс приближений лишь при выполнении некоторых условий (например при непрерывности и знакопостоянстве первой и второй производной функции в окрестности корня) и при их нарушении либо дает расходящийся процесс, либо приводит к другому корню.

Очевидно, что для функций, производная от которых в окрестности корня близка к нулю, использовать метод Ньютона не рекомендуется. Расчетная формула:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

Пример 1.3. Решить уравнение $x^3 + x - 1 = 0$ на отрезке **[0; 1]** методом Ньютона с точностью $\varepsilon = 0{,}001$ с помощью программы Excel.

Порядок решения (рис. 1.8).

1) Ввести в ячейки **A1:D1** заголовки столбцов.
2) В ячейку **A2** – значение начального приближения **1**
3) В ячейку **B3** – формулу функции **=A2^3+A2-1**
4) В ячейку **C3** – формулу производной функции **=3*A2^2+1**
5) В ячейку **A3** – формулу первого приближения **=A2-B3/C3**
6) В ячейку **D3** – погрешность **=ABS(A3-A2)**
7) Выделить ячейки **A3:D3** и скопировать формулы в соседние ячейки расположенных ниже строк **A4:D4**, **A5:D5**, и т.д. при помощи маркера заполнения. Каждая новая строка содержит результаты очередного приближения.
8) В столбце **A** найти значение корня, соответствующее заданной точности.

Приближенное решение данного уравнения $x = 0{,}68233 \approx 0{,}682$ содержится в ячейке **A6** (погрешность $0{,}00001 < 0{,}001$ в ячейке **D6**).

	A	B	C	D
1	x	F(x)	F'(x)	погрешность
2	1,00000			
3	0,75000	1,00000	4,00000	0,25000
4	0,68605	0,17188	2,68750	0,06395
5	0,68234	0,00894	2,41198	0,00371
6	0,68233	0,00003	2,39676	0,00001

Рис. 1.8. Решение уравнения методом Ньютона с помощью программы Excel.

Уточнение корней методом простой итерации

Другим представителем итерационных методов является *метод простой итерации*.

Здесь уравнение $f(x)=0$ заменяется равносильным уравнением $x=\varphi(x)$ и строится последовательность значений. Если функция $\varphi(x)$ определена и дифференцируема на некотором интервале, причем $|\varphi'(x)|<1$, то эта

последовательность сходится к корню уравнения $x = \varphi(x)$ на этом интервале.

Если $f'(x) > 0$, то подбор равносильного уравнения можно свести к замене $x = x - l * f(x)$, т.е. к выбору $\varphi(x) = x - l * f(x)$, где $l > 0$ подбирается так, чтобы в окрестности корня $0 < \varphi'(x) = 1 - l * f'(x)$. Отсюда может быть построен итерационный процесс

$$X_{n+1} = X_n - \frac{f(X_n)}{M}, \quad n = 0, 1, 2, \ldots$$

где $M = max\,|f'(x)|$ (в случае $f'(x) < 0$ возьмите функцию $f(x)$ с противоположным знаком).

Возьмем для примера уравнение $x^3 + x - 1000 = 0$. Очевидно, что корень данного уравнения несколько меньше 10. Если переписать это уравнение в виде $x = 1000 - x^3$ и начать итерационный процесс при $x_0 = 10$, то из первых же приближений очевидна его расходимость. Если же учесть $f'(x) = 3x^2 + 1 > 0$ и принять за приближенное значение максимума $f'(x)$ $M = 300$, то можно построить сходящийся итерационный процесс на основе представления

$$X = X - \frac{X^3 + X - 1000}{300}.$$

Пример 1.4. Решить уравнение $x^3 + x - 1 = 0$ на отрезке [0; 1] методом простой итерации с точностью $\varepsilon = 0{,}01$ с помощью программы Excel.

Порядок решения (рис. 1.11).

1) Ввести в ячейки **A1:D1** заголовки столбцов.
2) В ячейку **A2** – значение начального приближения **1**
3) В ячейку **B3** – формулу функции **=A2^3+A2-1**
4) В ячейку **C2** – значение ***M*** **5**
5) В ячейку **A3** – формулу первого приближения **=A2-B3/$C$2**
6) В ячейку **D3** – погрешность **=ABS(A3-A2)**
7) Выделить ячейки **A3:D3** и скопировать формулы в соседние ячейки расположенных ниже строк **A4:D4**, **A5:D5**, и т.д. при помощи маркера заполнения. Каждая новая строка содержит результаты очередного приближения.
8) В столбце **A** найти значение корня, соответствующее заданной точности.

Приближенное решение данного уравнения $x = 0{,}68427 \approx 0{,}68$ содержится в ячейке **A9** (погрешность **0,00179463 < 0,01** в ячейке **D9**).

	A	B	C	D
1	x	f(x)	M	погрешность
2	1		5	
3	0,8	1		0,2
4	0,7376	0,312		0,0624
5	0,70982	0,13889		0,02777881
6	0,69633	0,06746		0,01349237
7	0,68954	0,03396		0,00679209
8	0,68606	0,01738		0,0034769
9	0,68427	0,00897		0,00179463

Рис.1.11. Решение уравнения методом простой итерации с помощью программы Excel.

Варианты функции $y(x)$ к Заданию 1

Таблица 1

№ варианта	Функция $y(x)$
0	$sin^2 3x - lg(x+2)$
1	$sin^2\left(\frac{x}{2}+2\right)^2 - \frac{x}{5}$
2	$0.2\,x - cos^2 3x$
3	$\log_2(x+4) - \frac{1}{2}\left(\sin\left(\frac{x+6}{3}\right)^3 + 5\right)$
4	$\frac{sin(x-8)^2}{3} - \frac{x-2}{5}$
5	$\frac{cos(x-8)^2}{3} + ln(x+3) - 1.5$
6	$\frac{cos(x-8)^2 - sin(x+1)}{2} - 0.2$
7	$\frac{1}{2}\,sin\frac{(x+5)^2}{2} + \left(\frac{x-2}{4}\right)^2$
8	$sin\,(x^2+3) + \frac{sin^2 x}{2}$
9	$\frac{2\,sin\,8x}{3} + \frac{3\,cos\,x}{4}$
10	$cos\,5x\;\; ln\frac{x+3}{2}$
11	$\frac{3}{x+3}\,sin\left(\frac{2x+6}{3}\right)^2$
12	$cos\,\frac{(x+2)^2}{2}\;\; ln\left(\frac{x+3}{2}\right)$
13	$\frac{cos\left((x-5)^2+1\right)}{2}\,ln(x+2.5)$
14	$\frac{\sin(x-1)^2}{e^{x/4}} - 0.1$
15	$\frac{1}{2}\,sin\,\frac{(x+3)^2}{2}\,ln\,(x+2)$

№ варианта	Функция $y(x)$
16	$\frac{1}{2}\sin\frac{(x+3)^2}{2}+\frac{\ln(x+2)}{2}-1$
17	$\frac{\sin\left(2(x-2)^2\right)}{e^{x/4}}-0.1x$
18	$\frac{\sin 10x}{e^{\frac{(x-2)^2}{5}}}+0.5$
19	$\frac{\sin(10x-20)}{e^{\frac{(x-2)^2}{5}}}+0.25x$
20	$\frac{\cos(5x+10)}{\ln(4x+8)}$
21	$\frac{2\cos 5x}{3}-\frac{x}{6}+\frac{1}{3}$
22	$2\sqrt{2\cos^2\left(\frac{x}{2}+2\right)^2+2}-\frac{x}{5}-3$
23	$\frac{\sin(10x-20)}{e^{\frac{(x-2)^2}{5}}}-0.25x$
24	$\frac{1}{2}\cos\left(\frac{x}{3}\right)-\sin^2(3x)$
25	$\sin\left(\frac{x}{3}\right)-\cos^2(3x)$

Решение нелинейных уравнений в SciLab

Пример. Найти корни уравнения $\frac{e^x}{5} - 2 \cdot (x-1)^2 = 0$

Этап 1. Отделение корней. С помощью конструкции function ... endfunction определяется вид функции, задается вектор диапазона значений аргумента x, вычисляется вектор значений функции y и строится график функции (рис. 11).

```
-->function [y]=F(x)
-->y=exp(x)/5-2*(x-1).^2
-->endfunction

-->x=[-1:0.01:6];

-->y=F(x);

-->plot(x,y)
```

Рис. 13. Команды для задания вида функции и построения графика функции

Этап 2. По графику функции (рис. 12) определяются начальные приближения корней (0, 2, 5). Уточнить корни можно: либо последовательно вызывая функцию fsolve с различными начальными приближениями (рис. 13); либо задав вектор начальных приближений, тогда fsolve вызывается один раз (рис. 14).

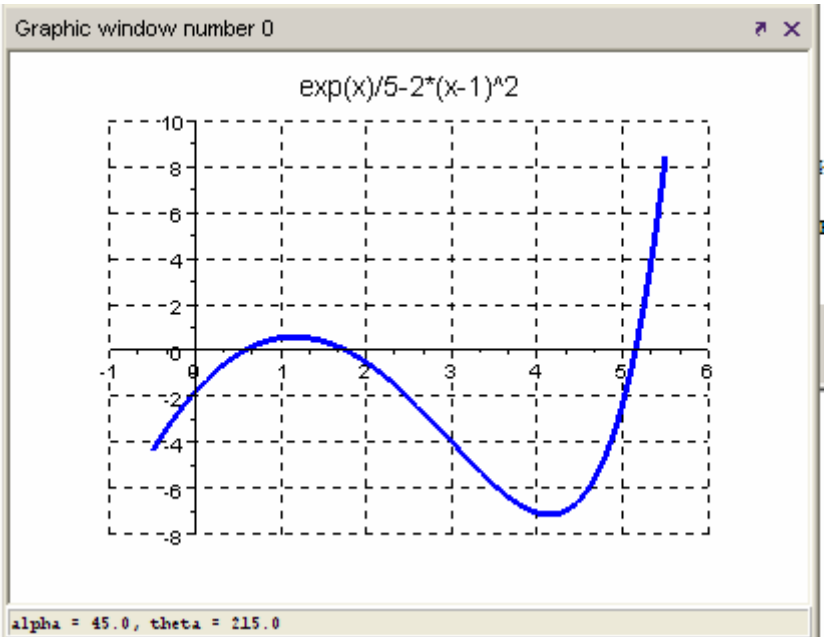


Рис. 14. Графическое решение трансцендентного уравнения

```
-->X(1)=fsolve(0,F); X(2)=fsolve(2,F); X(3)=fs

-->X
 X  =

    0.5778406
    1.7638701
    5.1476865
```

Рис. 13. Последовательный вызов fsolve с различным начальным приближением

```
-->X=fsolve([0;2;5],F)
 X  =

    0.5778406
    1.7638701
    5.1476865
```

Рис. 15. Вызов fsolve при задании начальных приближений в виде вектора

На рис. 15 приведен файл-сценарий нахождения корней трансцендентного уравнения.

```
//Задание вида функции
function [y]=F(x)
y=exp(x)/5-2*(x-1).^2
endfunction
//
//Задание значений аргумента
x=[-0.5:0.01:5.5];
//
//Вычисление значений функции
y=F(x);
//
//Изменение параметров графика
a=gca();
a.x_location="origin";
a.y_location="origin";
xgrid();
xtitle('exp(x)/5-2*(x-1)^2')
//
//Построение графика функции
plot(x,y)
//
//Нахождение трех корней трансцендентного
//уравнения при их заданных начальных
//приближениях 0 2 5
fsolve([0;2;5],F)
```

Рис. 16. Файл-сценарий построения графика трансцендентного уравнения
и нахождения его корней

Контрольные вопросы.

1) Какие точные методы решения нелинейных уравнений вы знаете?
2) Для чего нужен первый этап - отделение корней?
3) Сформулируйте условия существования решения уравнения. Являются ли эти требования необходимыми и достаточными?

4) Что можно сказать о точности методов половинного деления, хорд, касательных и комбинированного? По каким параметрам их еще можно сравнить?

Лабораторная работа №3
Массивы и матрицы. Решение задач линейной алгебры

Для работы с множеством данных удобно использовать массивы. Например, можно создать массив для хранения числовых или символьных данных. В этом случае вместо создания переменной для хранения каждого данного достаточно создать один массив, где каждому элементу будет присвоен порядковый номер.
Таким образом, массив - множественный тип данных, состоящий из фиксированного числа элементов. Как и любой другой переменной, массиву должно быть присвоено имя. Переменную, представляющую собой просто список данных, называют одномерным массивом, или вектором. Для доступа к данным, хранящимся в определенном элементе массива, необходимо указать имя массива и порядковый номер этого элемента, называемый индексом.
Если возникает необходимость хранения данных в виде таблиц, в формате строк и столбцов, то необходимо использовать двумерные массивы (матрицы). Для доступа к данным, хранящимся в таком массиве, необходимо указать имя массива и два индекса: первый должен соответствовать номеру строки, а второй
- номеру столбца, в которых хранится необходимый элемент. Значение нижней границы индексации в Scilab равно единице. Индексы могут быть только целыми положительными числами.

Ввод и формирование массивов и матриц

Задать одномерный массив в Scilab можно следующим образом:

name=Xn:dX:Xk

где name - имя переменной, в которую будет записан сформированный массив, Xn - значение первого элемента массива, Xk - значение последнего элемента массива, dX - шаг, с помощью которого формируется каждый следующий элемент массива, т.е. значение второго элемента составит Xn+dX, третьего Xn+ dX+dX и так далее до Xk.
Если параметр dX в конструкции отсутствует, это означает, что по умолчанию он принимает значение, равное единице, т.е. каждый следующий элемент массива равен значению предыдущего плюс один:

name=Xn:Xk

Переменную, заданную как массив, можно использовать в арифметических выражениях и в качестве аргумента математических функций. Результатом работы таких операторов являются массивы:

Листинг 3.1. Примеры работы с массивами

```
--> Xn=-3.5; dX=1.5; Xk=4.5;
-->  X=Xn:dX:Xk
X =
-3.5000 -2.0000 -0.5000 1.0000 2.5000 4.0000
-->    Y=sin(X/2)
Y =
-0.9840       -0.8415       -0.2474       0.4794       0.9490
        0.9093
-->  A=0:5
A =
0 1 2 3 4 5
--> 0:5
ans =
```

```
0        1        2 3 4 5
--> ans/2+%pi
 ans =
3.1416        3.6416        4.1416        4.6416        5.1416
        5.6416
```

Еще один способ задания векторов и матриц в Scilab - это их поэлементный ввод.
Так, для определения вектора-строки следует ввести имя массива, а затем после знака присваивания, в квадратных скобках через пробел или запятую, перечислить элементы массива:
name=[x1 x2 ... xn] или name=[x1, x2, ..., xn]
Пример ввода вектора-строки:

Листинг 3.2. Определение вектора-строки

```
--> V=[1 2 3 4 5]
V =
1 2 3 4 5
-->      W=[1.1,2.3,-0.1,5.88]
W =
1.1000 2.3000 -0.1000 5.8800
```

Элементы вектора-столбца вводятся через точку с запятой:
name=[x1; x2; ...; xn]
Пример ввода вектора-столбца:

Листинг 3.3. Определение вектора-столбца

```
-->    X=[1;2;3]
X =
1
2
3
```

Обратиться к элементу вектора можно, указав имя массива и порядковый номер элемента в круглых скобках:
name(индекс)
Например:
Листинг 3.4. Пример обращения к элементу массива

```
--> W=[1.1,2.3,-0.1,5.88];
-->  W(1)+2*W(3)
 ans =
0.9000
```

Ввод элементов матрицы также осуществляется в квадратных скобках, при этом элементы строки отделяются друг от друга пробелом или запятой, а строки разделяются между собой точкой с запятой:
name=[x11, x12, ..., x1n; x21, x22, ..., x2n; ...;
xm1, xm2, ..., xmn;]

Обратиться к элементу матрицы можно, указав после имени матрицы, в круглых скобках через запятую, номер строки и номер столбца на пересечении которых элемент расположен:
name(индекс1, индекс2)
Далее приведен пример задания матрицы и обращение к ее элементам:
Листинг 3.5. Пример обращения к элементам матрицы

```
--> A=[1 2 3;4 5 6;7 8 9]
A =
1       2       3
4       5       6
7       8       9
-->      A(1,2)^A(2,2)/A(3,3)
ans =  3.5556
```

Кроме того, матрицы и векторы можно формировать, составляя их из ранее заданных матриц и векторов:

Листинг 3.6. Пример конкатенации матриц

```
--> v1=[1 2 3]; v2=[4 5 6]; v3=[7 8 9];
--> //Горизонтальная конкатенация векторов-строк:
--> V=[v1 v2 v3]
V = 1 2 3 4 5 6 7 8 9
-->//Вертикальная конкатенация векторов-строк,
-->//результат матрица:
-->  V=[v1;  v2;  v3]
V =
1 2 3
4 5 6
7 8 9
-->//Горизонтальная конкатенация матриц:
--> M=[V V V]
       M  =
       1 2 3 1 2 3 1 2 3
       4 5 6 4 5 6 4 5 6
       7 8 9 7 8 9 7 8 9
-->//Вертикальная конкатенация матриц:
-->  M=[V;V]
M =
1 2 3
4 5 6
7 8 9
1 2 3
4 5 6
7 8 9
```

Важную роль при работе с матрицами играет знак двоеточия <:>. Указывая его вместо индекса при обращении к массиву, можно получать доступ к группам его элементов. Например:

Листинг 3.7. Примеры использования операции <:>

```
-->//Пусть  задана  матрица  A
--> A=[5 7 6 5; 7 10 8 7;6 8 10 9;5 7 9 10]
--> //Выделить из матрицы A второй столбец
--> A(:,2)
ans =
```

```
7
10
8
7
--> //Выделить из матрицы A третью строку
--> A(3,:)
ans =
 6 8 10 9
--> //Выделить из матрицы A подматрицу M
-->      M=A(3:4,2:3)
M =
8 10
7 9
--> //Удалить из матрицы A второй столбец
-->   A(:,2)=[]
A =
5 8 10
7 7 9
6 10 9
5 9 10
--> //Удалить из матрицы A третью строку
-->   A(3,:)=[]
A =
5 8 10
7 7 9
5 9 10
--> //Представить матрицу M в виде вектора-столбца
--> v=M(:)
v  = 8
7
10
9
--> //Выделить из вектора v элементы со второго по четвертый
--> b=v(2:4)
b  =
7
10
```

9
--> //Удалить из массива b второй элемент
--> b(2)=[];

Действия над матрицами

Для работы с матрицами и векторами в Scilab предусмотрены следующие операции:

+ - сложение;
ации сложения и вычитания определены для матриц одной размерности или векторов одного типа, т.е. суммировать (вычитать) можно либо векторы-столбцы, либо векторы-строки одинаковой длины);

’ – транспонирование (Если в некоторой матрице заменить строки соответствующими столбцами, то получится транспонированная матрица);

- матричное умножение (Операция умножения вектора на вектор определена только для векторов одинакового размера, причем один из них должен быть вектором-столбцом, а второй вектором-строкой. Матричное умножение выполняется по правилу «строка на столбец» и допустимо, если количество строк во второй матрице совпадает с количеством столбцов в первой. Кроме того, переместительный закон на произведение матриц не распространяется);

- умножение на число;

^ - возведение в степень (Возвести матрицу в n-ю степень значит умножить ее саму на себя n раз. При этом целочисленный показатель степени может быть как положительным, так и отрицательным. В первом случае выполняется алгоритм умножения матрицы на себя указанное число раз, во втором умножается на себя матрица, обратная к данной);

\ - левое деление(A\B) ⇒ (A−1B), операция может быть применима для решения матричного уравнения вида A · X = B, где X - неизвестный вектор);
/ - правое деление ((B/A) ⇒ (B · A−1), используют для решения матричных уравнений вида X · A = B);

.* - поэлементное умножение матриц;

.^ - поэлементное возведение в степень;

.\ - поэлементное левое деление;

./ - поэлементное правое деление.

Пример действий над матрицами:

Листинг 3.8. Примеры матричных операций

```
-->A=[1 2 0;-1 3 1;4 -2 5];
-->B=[-1 0 1;2 1 1;3 -1 -1];
-->//Вычислить (A^T+B)^2 - 2A(0.5B^T-A)
-->(A'+B)^2-2*A*(1/2*B'-A)
ans    =
           10.  8.      24.
           11.  20.     35.
           63.  - 30.   68.
--> //Решить матричные уравнения A•X=B и X•A=B.
-->A=[3 2;4 3];
-->B=[-1 7;3 5];
-->//Решение матричного уравнения AX=B:
-->X=A\B
X      =
- 9.    11.
13.    - 13.
-->//Решение матричного уравнения XA=B:
-->X=B/A
```

```
X =
- 31.   23.
- 11.   9.
-->//Проверка
-->X*A-B
ans    =
0.      0.
0.      0.
```

Кроме того, если к некоторому заданному вектору или матрице применить математическую функцию, то результатом будет новый вектор или матрица той же размерности, но элементы будут преобразованы в соответствии с заданной функцией:

Листинг 3.9. Пример применения функции к массиву

```
--> x=[0.1 -2.2 3.14 0 -1];
-->   sin(x)
ans =
0.0998  -0.8085  0.0016  0  -0.8415
```

Специальные матричные функции

Для работы с матрицами и векторами в Scilab существуют специальные функции. Рассмотрим наиболее часто используемые из них.
Функции определения матриц:

matrix(A [,n,m]) - преобразует матрицу A в матрицу другого размера;

Листинг 3.10. Использование функции matrix

```
-->D=[1 2;3 4;5 6];
-->matrix(D,2,3)
ans    =
```

1. 5. 4.
3. 2. 6.
-->matrix(D,3,2)
ans =
1. 2.
3. 4.
5. 6.
-->matrix(D,1,6)
ns =
1. 3. 5. 2. 4. 6.
-->matrix(D,6,1)
ans =
1.
3.
5.
2.
4.
6.

ones(m,n) - создает матрицу единиц из m строк и n столбцов ;

Листинг 3.11. Использование функции ones

-->ones(1,3) //Формируется вектор-строка
ans =
1. 1. 1.
-->ones(2,2) //Формируется квадратная матрица
ans =
1. 1.
1. 1.
-->m=3; n=2;
-->X=ones(m,n) //Формируется матрица размерности m на n
X =
1. 1.
1. 1.
1. 1.

```
-->M=[1 2 3;4 5 6]
M      =
1. 2. 3.
4. 5. 6.
-->//Формируется матрица Y, состоящая из единиц,
-->//той же размерности, что и матрица M
-->Y=ones(M)
Y      =
1. 1. 1.
1. 1. 1.
```

zeros(m,n) - создает нулевую матрицу из m строк и n столбцов;

Листинг 3.12. Использование функции zeros

```
-->zeros(3,2)
ans    =
0. 0.
0. 0.
0.
-->M=[1 2 3 4 5];
-->Z=zeros(M)
 Z      =
0. 0. 0. 0. 0.
```

eye(m,n) - формирует единичную матриз m строк и n столбцов;

Листинг 3.13. Примеры использования функции eye

```
-->eye(3,3)
ans    = 1.  0.  0.
0. 1. 0.
0. 0. 1.
-->eye(5,1)
ans    =
1.
0.
0.
0.
0.
-->m=3; n=4;
-->E=eye(m,n) E   =
1. 0. 0. 0.
0. 1. 0. 0.
0. 0. 1. 0.
-->M=[0 1;2 3];
-->//Формируется единичная матрица E
-->//той же размерности, что и матрица M
-->E=eye(M) E     =
0.
0. 1.
-->//Функцию можно использовать без параметров eye().
-->//В этом случае задается матрица с неопределенными
-->//размерами, которые будут определены после
суммирования
-->//с другой, определенной ранее, матрицей.
-->M=[1 2;3 4;5 6]; E=eye();
-->A=E+M A        =
2.
5.
5. 6.
-->M-E
ans    = 0.  2.
```

3. 3.
5. 6.

rand(n1,n2,...nn[,fl]) - формирует многомерную матрицу случайных чисел. Необязательный параметр p - это символьная переменная, с по- мощью которой можно задать тип распределения случайной величи-
ны (’uniform’ - равномерное, ’normal’ - гауссовское); rand(m,n) - фор- мирует матрицу m на n случайных чисел; rand(M) - формирует матрицу случайных чисел, размер которой совпадает с размером матрицы M; ре- зультат функции rand() - случайное скалярное число;

Листинг 3.14. Примеры использования функции rand

```
-->rand(2,2)//Матрица 2 на 2 случайных чисел ans =
0.2113249 0.0002211
0.7560439 0.3303271
--> R=rand(2,2,2)//Многомерный массив случайных чисел
R(:,:,1) =
0.9355  0.4103
0.9169  0.8936
R(:,:,2)  = 0.0579 0.8132
0.3529  0.0099
-->rand()//Случайное число ans =
0.6653811
```

sparse([i1 j1;i2 j2;...;in jn],[n1,n2,...,nn]) - формирует разре- женную матрицу1. Для создания матрицы такого типа необходимо указать
индексы ее ненулевых элементов - [i1 j1,i2 j2,...,in jn], и их значе- ния - [n1,n2,...,nn]. Индексы одного элемента отделяются друг от друга либо пробелом, либо запятой, а пары индексов - соответственно точкой с запятой, значения элементов разделяются запятыми. При попытке просмотреть матрицу подобного типа пользователю будет предоставлено сообщение о ее размерности, а также значения ненулевых элементов и их местоположение в матрице;

full(M) - вывод разреженной матрицы M в виде таблицы;
Листинг 3.15. Использование функций sparse и full

```
-->A=sparse([1  3;3  2;3  5],[4,5,6])
A       =
(       3,      5) sparse matrix
(       1,      3)      4.
(       3,      2)      5.
(       3,      5)      6.
-->full(A) ans       =
        0.      0.  4.  0.  0.
        0.      0.  0.  0.  0.
        0.      5.  0.  0.  6.
```

---

hypermat(D[,V]) - создание многомерной матрицы с размерностью, задан- ной вектором D и значениями элементов, хранящихся в векторе V (исполь- зование параметра V необязательно);

Листинг 3.16. Использование функции hypermat

```
-->//Пример  создания  матрицы  M,
-->//состоящей из трех матриц размерности два на два,
-->//каждый элемент матрицы - член последовательности
-->//целых чисел от 0 до 11.
-->M=hypermat([2   2   3],0:11) M     =
(:,:,1)
2.
3.
(:,:,2)
6.
7.
(:,:,3)
8. 10.
9. 11.
```

diag(V[,k]) - возвращает квадратную матрицу с элементами V на главной или на k-й диагонали1; функция diag(A[,k]), где A - ранее определенная матрица, в качестве результата выдаст вектор-столбец, содержащий эле- менты главной или k-ой диагонали матрицы A;

Листинг 3.17. Использование функции diag

```
--> V=[1,2,3];
--> diag(V)//Диагональная матрица, V на главной
диагонали ans = 10 0
0       2 0
0       0 3
-->//Диагональная матрица,
-->//V на первой диагонали (выше главной)
--> diag(V,1)
          ans = 0 1  0 0
          0       0  2 0
          0       0  0 3
          0       0  0 0
-->//Диагональная матрица,
-->//V на первой диагонали (ниже главной)
--> diag(V,-1)
          ans =  0 0  0 0
               1    0  0 0
               0    2  0 0
               0    0  3 0
--> A=[-1 2 0 ;2 1 -1 ;2 1 3]
A  =
-1 2   0
2 1 -1
2 1    3
--> diag(A)  //Главная диагональ матрицы A ans =
-1
1
3
```

cat(n, A, B, [C, ...]) - объединяет матрицы A и B или все входящие матрицы, при n=1 по строкам, при n=2 по столбцам; то же что [A; B] или [A, B];

Листинг 3.18. Использование функции cat

```
--> A=[1 2;3 4]; B=[5 6 ;7 8];
-->  cat(2,A,B)//Объединение  матриц ans =
1 2 5 6
3 4 7 8
-->  cat(1,A,B)  //Объединение  матриц ans =
1 2
3 4
5 6
7 8
```

tril(A[,k]) - формирует из матрицы A нижнюю треугольную матрицу , начиная с главной или с k-й диагонали;

Листинг 3.19. Использование функции tril

```
--> A=[1 2 3;4 5 6;7 8 9] A =
1 2 3
4 5 6
7 8 9
-->//Нижняя треугольная матрица, начиная с главной диагонали
-->   tril(A) ans =
1 0 0
4 5 0
7 8 9
--> tril(A,0)//Тоже что и tril(A) ans =
1 0 0
4 5 0
7 8 9
--> tril(A,1)//Нижняя треугольная матрица,
--> //начиная с первой диагонали (выше главной) ans =
```

1 2 0
4 5 6
7 8 9
--> tril(A,-2))//Нижняя треугольная матрица,
--> //начиная со второй диагонали (ниже главной) ans =
0 0 0
0 0 0
7 0 0

triu(A[,k]) - формирует из матрицы A верхнюю треугольную [1]
матрицу , начиная с главной или с k-й диагонали;

Листинг 3.20. Использование функции triu

--> A=[1 2 3;4 5 6;7 8 9];
--> triu(A)//Верхняя треугольная матрица ans =
1 2 3
0 5 6
0 0 9
--> triu(A,2))//Верхняя треугольная матрица,
--> //начиная со второй диагонали (выше главной) ans =
0 0 3
0 0 0
0 0 0
--> triu(A,-1))//Верхняя треугольная матрица,
--> //начиная с первой диагонали (ниже главной) ans =
1 2 3
4 5 6
0 8 9

sort(X) - выполняет упорядочивание массива X; если X - матрица, сорти- ровка выполняется по столбцам;

Листинг 3.21. Использование функции sort

-->b=[2 0 1]; sort(b) //Сортировка по убыванию ans =
2. 1. 0.

```
-->-sort(-b)  //Сортировка  по  возрастанию ans  =
0.      1.      2.
-->A=[1 2 0;-1 3 1;4 -2 5];
-->sort(A) //Сортировка матрицы ans        =
                5.   2.    0.
                4.   1.    - 1.
                3.   1.    - 2.
```

Функции вычисления различных числовых характеристик матриц:

size(V[,fl]) - определяет размер массива V; если V - двумерный мас- сив, то size(V,1) или size(V,'r') определяют число строк матрицы V, а size(V,2) или size(V,'c') - число столбцов;

Листинг 3.22. Использование функции size

```
-->M=[1 2;3 4;5 6;7 8];
-->[n,m]=size(M) m       =
2.
n       =
4.
-->size(M,1) ans    =
4.
-->size(M,2) ans    =
2.
```

length(X) - определяет количество элементов массива X; если X - вектор, его длину; если X - матрица, вычисляет общее число ее элементов;

Листинг 3.23. Использование функции length

```
-->  V=[-1  0  3  -2  1  -1  1];//Вектор-строка
-->  length(V)//Длина  вектора ans =
7
-->[1 2 3;4 5 6];//Матрица
-->length(ans)//Количество  элементов  матрицы ans   =
```

6.

sum(X[,fl]) - вычисляет сумму элементов массива X, имеет необязатель- ный параметр fl. Если параметр fl отсутствует, то функция sum(X) воз- вращает скалярное значение, равное сумме элементов массива. Если fl=’r’
или fl=1, что то же самое, то функция вернет строку, равную поэлементной сумме столбцов матрицы X. Если fl=’c’ или fl=2, то результатом работы функции будет вектор-столбец, каждый элемент которого равен сумме эле- ментов строк матрицы X. Частный случай применения функции sum - это вычисление скалярного произведения векторов1;

Листинг 3.24. Примеры использования функции sum

```
-->M=[1 2 3;4 5 6;7 8 9];
-->Y=sum(M) //Сумма элементов матрицы Y     =  45.
-->S1=sum(M,1)  //Сумма  элементов  матрицы  по  столбцам
S1      =
12 15 18
-->S2=sum(M,2) // Сумма элементов матрицы по строкам S2
         =
6
15
24
--> V=[-1 0 3 -2 1 -1 1];
--> sum(V) //Сумма элементов вектора ans = 1
-->//Частный случай. Вычисление скалярного произведения
--> a=[1 2 3];b=[2 0 1];
-->    sum(a.*b) ans = 5
```

prod(X[,fl]) - вычисляет произведение элементов массива X, работает аналогично функции sum;

Листинг 3.25. Использование функции prod

```
-->prod(M)
ans    = 362880.
-->p1=prod(M,1) p1       =
28 80 162
-->p2=prod(M,2) p2       =
6
120
504
--> V=[1,2,3];
-->  prod(V)  //Произведение  элементов  вектора ans = 6
```

max(M[,fl]) - вычисляет наибольший элемент в массиве M, имеет необя- зательный параметр fl. Если параметр fl отсутствует, то функция max(M) возвращает максимальный элемент массива M; если fl=’r’, то функция
вернет строку максимальных элементов столбцов матрицы M; если fl=’c’, то результатом работы функции будет вектор-столбец, каждый элемент ко- торого равен максимальному элементу соответствующих строк матрицы M. Функция [x, nom]=max(M[,fl]) вернет значение максимального элемента x и его номер в массиве nom;

Листинг 3.26. Использование функции max

```
-->M=[5 0 3;2 7 1;0 4 9];
-->max(M) ans      =   9.
-->max(M,’r’) ans =
5.       7.       9.
-->max(M,’c’) ans =
```

5.
7.
9.
-->[x,nom]=max(M) nom=

3. 3.
x = 9.

min(M[,fl]) - вычисляет наименьший элемент в массиве M, работает аналогично функции max;

Листинг 3.27. Использование функции min

-->A=[5 10 3 2 7 1 25 4 0];
-->[x,nom]=min(A) nom =
7.
x =
25.

mean(M[,fl]) - вычисляет среднее значение массива M; если M двумерный массив, то mean(M,1) или mean(M,'r') определяют среднее значение строк матрицы M, а mean(M,2) или mean(M,'c') - среднее значение столбцов;

Листинг 3.28. Использование функции mean

-->mean(M) ans =
3.4444444
-->mean(M,1) ans =
2.3333333 3.6666667 4.3333333
-->mean(M,2) ans =
2.6666667
3.3333333
4.3333333

median(M[,fl]) - вычисляет медиану массива M, работает аналогично функции mean;

Листинг 3.29. Использование функции median

```
-->M=[5 0 3;2 7 1;0 4 9];
-->median(M) ans =  3.
-->median(M,1) ans        =
2.       4.       3.
-->median(M,2) ans        =
3.
2.
4.
```

det(M) - вычисляет определитель квадратной матрицы M;
Листинг 3.30. Использование функции det

```
-->M=[1 0 2;3 2 1;0 3 1];
-->det(M) ans        = 17.
-->Z=[1 2 2;0 1 3;2 4 4];
-->det(Z) ans        = 0.
```

rank(M[,tol]) - вычисление ранга матрицы M с точностью tol.

Листинг 3.31. Использование функции rank

```
-->M=[1 0 2;3 2 1;0 3 1];
-->rank(M) ans      =  3.
-->Z=[1 2 2;0 1 3;2 4 4];
-->rank(Z) ans      = 2.
```

norm(M[,fl]) - вычисление нормы квадратной матрицы M; тип нормы определяется необязательной строковой переменной fl, по умолчанию fl=2. Функции norm(M) и norm(M,2) эквивалентны и вычисляют вторую норму матрицы M2. Первая норма3 определяется функцией norm(M,1). Функции norm(M,’inf’) и norm(M,’fro’) вычисляют соответственно бес- конечную4 и евклидову нормы5. Если V - вектор, то результатом работы функции norm(V,1) будет сумма модулей всех элементов вектора V. С по- мощью функции norm(V,2) можно вычислить модуль вектора V6.

Значение norm(V,’inf’) равно модулю максимального элемента вектора по модулю;

Листинг 3.32. Использование функции norm

```
-->M=[1 0 2;3 2 1;0 3 1];
-->norm(M,1) ans  =  5.
-->norm(M,2) ans  =
4.5806705
-->norm(M,’inf’) ans      = 6.
-->norm(M,’fro’) ans      = 5.3851648
-->X=[5 -3 4 -1 2];
-->norm(X,1) ans  = 15.
-->sum(abs(X))//То  же,  что  и  norm(X,1) ans    = 15.
-->norm(X,2)
ans     =  7.4161985
-->sqrt(sum(X^2))  //То  же,  что  и  norm(X,2) ans       =
7.4161985
-->norm(X,’inf’) ans       =
5.
-->max(abs(X))//То   же,  что  и  norm(X,’inf’) ans       =
5.
```

cond(M) - вычисляет число обусловленности матрицы M по второй норме;

Листинг 3.33. Использование функции cond

```
-->A=[5 7 6 5;7 10 8 7;6 8 10 9;5 7 9 10];
-->cond(A) ans      =
2984.0927
```

Функции, реализующие численные алгоритмы решения задач линейной алгебры:

spec(M) - вычисляет собственные значения и собственные векторы ратной матрицы M.

Использование функции spec

```
-->M=[3 -4;-2 1]
3. - 2.
- 4. 1.
-->spec(M) //Собственные числа матрицы ans  =
- 1.
5.
//X - собственные векторы,
-->соответствующие собственным значениям из матрицы Y.
-->[X,Y]=spec(M) Y  =
! - 1. 0   !
! 0    5.  !
X  =
! - 0.4472136  - 0.7071068 !
! - 0.8944272    0.7071068 !
```

inv(A) - вычисляет матрицу, обратную к A3;

Листинг 3.35. Использование функции inv

```
-->//Пример вычисления обратной матрицы.
-->A=[1 2 3 5;0 1 3 2;4 2 1 1;2 3 0 1];
-->inv(A) ans        =

    !    0.0285714   - 0.1428571     0.3428571   - 0.2 !
    ! - 0.1428571     0.2142857   - 0.2142857     0.5 !
    ! - 0.2             0.5           0.1         - 0.1 !
    !    0.3714286   - 0.3571429   - 0.0428571   - 0.1 !

-->//При умножении обратной матрицы на исходную,
-->//получилась матрица, близкая к единичной.
-->inv(A)*A ans    =
1. - 1.110D-16 0. 0.
0.       1.       - 5.551D-17 5.551D-17
0.       0. 1. 1.388D-17
0.       0. 6.939D-17 1.
-->//При попытке обратить вырожденную матрицу
-->//(определитель равен или близок к нулю)
-->//пользователь получит сообщение об ошибке.
-->B=[1 2 3;1 4 5;1 6 7];
-->inv(B)
!--error        19
Problem is singular
```

pinv(A[,tol]) - вычисляет псевдообратную 1 для матрицы A с точностью tol (необязательный параметр);

Листинг 3.36. Использование функции pinv

```
-->pinv(A) ans      =
            0.0285714 -             0.3428571 - 0.2
            -           0.2142857 -             0.5
            - 0.2   0.5   0.1 - 0.1
            0.3714286 -             -           - 0.1
```

linsolve(A,b) - решает систему линейных алгебраических уравнений ви- да $A \cdot x - b = 0$.

Листинг 3.37. Пример использования функции linsolve

```
-->//Решение  системы  линейных  уравнений
-->//{x1+2x2-7=0;  x1+x2-6=0}.
-->//Свободные коэффициенты вводятся как вектор-столбец
-->//и с учетом знаков.
-->A=[1 2;1 1];b=[-7;-6];
-->x=linsolve(A,b) x       =
5.
1.
-->//Результатом  операции  A*x+b является  вектор,
достаточно
-->//близкий к нулю, это значит, что система решена верно.
-->A*x+b ans       =
1.0D-14 *
- 0.6217249
0.0888178
-->//Решение  системы  {x1+x2-1=0;  x1+x2-3=0}
-->A=[1 1;1 1]; b=[-1;-3];
-->//Система не имеет решений:
-->linsolve(A,b)
WARNING:Conflicting  linear  constraints! ans    =
[]
-->//Решение системы {3x1-x2-1=0; 6x1-2x2-2=0}.
-->//В случае, когда система имеет бесконечное
-->//множество решений, SCILAB выдаст одно из них.
-->A=[3 -1;6 -2];
-->b=[-1;-2];
-->x=linsolve(A,b) x       =
```

0.3
- 0.1
-->//Проверка неверна
-->A*x+b ans =
1.0D-15 *
- 0.1110223
- 0.2220446

rref(A) - осуществляет приведение матрицы A к треугольной форме, ис- пользуя метод исключения Гаусса;

Листинг 3.38. Пример использования функции rref

--> A=[3 -2 1 5;6 -4 2 7;9 -6 3 12] A =
3 -2 1 5
6 -4 2 7
9 -6 3 12
--> rref(A) ans =
1.0000 -0.6667 0.3333 0
0 0 0 1.0000
0 0 0 0
lu(M) - выполняет треугольное разложение матрицы M ;

Листинг 3.39. Использование функции lu

-->A=[2 -1 5;3 2 -5;1 1 -2] A =
2. - 1. 5.
3. 2. - 5.
1. 1. - 2.
-->[L,U]=lu(A) U =
3. 2. - 5.
0. - 2.3333333 8.3333333 !
0. 0. 0.8571429 ! L =
0.6666667 1. 0.
1. 0. 0.
0.3333333 - 0.1428571 1.
-->LU=L*U LU =

2. - 1. 5.
3. 2. - 5.
1. 1. - 2.

qr(M) - выполняет разложение ^{2} на ортогональную и верхнюю
матрицы M
треугольную матрицы;

Листинг 3.40. Использование функции qr

```
-->[Q,R]=qr(A) R =
 - 3.7416574   - 1.3363062    1.8708287
    0.         - 2.0528726    7.0632734
    0.            0.          0.7811335
  Q  =
 - 0.5345225    0.8350668    0.1301889
 - 0.8017837   - 0.4523279   - 0.3905667
 0.2672612   - 0.3131501     0.9113224

 -->Q*R

ans    =


    2.  - 1.    5.
    3.  2.    - 5.
    1.  1.    - 2.
```

svd(M) - выполняет сингулярное разложение[1] размером n×m; результатом работы функции может быть либо сингулярное разложение, либо вектор, содержащий сингулярные значения матрицы;

Листинг 3.41. Использование функции svd

```
-->[U,S,V]=svd(A) V     =
- 0.1725618 0.9641403   - 0.2016333
- 0.3059444 0.1421160   0.9413825
0.9362801   0.2241352   0.2704496
S       =
7.8003308   0.       0.
0.       3.6207331   0.
0.       0.       0.2124427
U       =
0.5951314   0.8028320   0.0357682
- 0.7449652 0.5678344   - 0.3501300
```

- 0.3014060 0.1817273 0.9360180
-->U*S*V’
ans =
2. - 1. 5.
3. 2. - 5.
1. 1. - 2.
-->s=svd(A) s =
7.8003308
3.6207331
0.2124427

kernel(M[,tol[,fl]]) - определение ядра матрицы [2]

M, параметры tol и
fl являются необязательными. Первый задает точность вычислений, второй - используемый при вычислении алгоритм и принимает значения ’qr’ или ’svd’.

Листинг 3.42. Использование функции kernel

```
-->A=[4 1 -3 -1;2 3 1 -5;1 -2 -2 3]
            A  =
            4.     1.  - 3.   - 1.
            2.     3.  1.     - 5.
            1.  -  2.  - 2.    3.
-->X=kernel(A)
X       =
0.3464102
0.5773503
0.4618802
0.5773503
```

Символьные матрицы и операции над ними

В Scilab можно задавать символьные матрицы, то есть матрицы, элементы которых имеют строковый тип. При этом необходимо помнить, что строковые элементы должны быть заключены в двойные или одинарные кавычки.

Листинг 3.43. Формирование символьных матриц

```
-->M=[’a’  ’b’;’c’  ’d’] M         =
a       b
c       d
-->P=[’1’  ’2’;’3’  ’4’] P =
1       2
3       4
```

Символьные матрицы можно складывать (результат сложения - конкатена ция соответствующих строк) и транспонировать:

Листинг 3.44. Операции над символьными матрицами

```
-->M+P
ans             = a1   b2
c3      d4
-->M’
```

ans = a c
b d

Кроме того, операции сложения и умножения можно проводить над от- дельными элементами символьных матриц при помощи функций addf(a,b) и mulf(a,b):

Листинг 3.45. Использование функций addf и mulf

```
-->addf(M(1,1),P(2,2))
ans  = a+4
-->mulf(M(1,2),P(2,1))
ans  = b*3
```

Контрольные вопросы.

5) Какие точные методы решения нелинейных уравнений вы знаете?
6) Для чего нужен первый этап - отделение корней?
7) Сформулируйте условия существования решения уравнения. Являются ли эти требования необходимыми и достаточными?
8) Что можно сказать о точности методов половинного деления, хорд, касательных и комбинированного? По каким параметрам их еще можно сравнить?

Лабораторная работа №4

Тема: **Численные методы решения систем линейных уравнений.**

1. **Цель работы**

Использование методов решения систем линейных уравнений для решения конкретных научных задач.

2. **Учебные вопросы, подлежащие рассмотрению:**

– Метод простой итерации
– Оценка погрешности решения системы линейных алгебраических уравнений.
– Понятие об обусловленности. Метод прогонки, трехдиагональная матрица.
– Релаксация.

3. **Порядок выполнения работы**

Задание

Дана система трех линейных уравнений с тремя неизвестными:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \end{cases}$$

1. Решите систему методом Гаусса:

а) используя «ручную» схему единственного деления, двумя способами: без перестановки строк; с перестановкой строк; расчеты выполняйте с тремя знаками после запятой (с применением калькулятора); подставьте найденные решения в исходную систему, вычислите невязки и сравните полученные решения; выбрав ведущие элементы схемы единственного деления, найдите значение определителя системы;

б) с помощью программы для ЭВМ с пооперационным учетом ошибок.

2. Решите систему методом простой итерации с точностью $\varepsilon = 10^{-4}$ с помощью программы для ЭВМ.

Исполнение: применить а) метод Гаусса; б) метод простой итерации используя любой инструментальный пакет.

Оценка: Сопоставление полученных результатов, решаемых различными методами.

Методические указания

В табличном процессоре Excel для решения систем уравнений есть два варианта: реализация алгоритмов в электронной таблице с помощью основных средств табличного процессора и использование специальных средств.

Первый вариант проиллюстрирован на примере системы уравнений

$$\begin{cases} x_1 + 2x_2 + 2x_4 = 5 \\ 2x_1 + 4x_2 - x_3 + 5x_4 = -1 \\ x_1 + 3x_2 + 5x_4 = -3 \\ 3x_1 + 7x_2 - 3x_3 + 9x_4 = -13 \end{cases}$$

На рис. приведены идентичные тексты в Excel, но один — в режиме отображения формул, а другой — значений. Из них прекрасно видно устройство алгоритма.

	A	B	C	D	E
1	Решение системы линейных алгебраических уравнений Ax=b				
2	методом Гаусса				
3		A			b
4					
5	1	1	0	2	5
6	2	4	-1	5	-1
7	1	3	0	5	-3
8	3	7	-2	9	-13
9		Первое преобразование			
10	1	=B5/$A$5	=C5/$A$5	=D5/$A$5	=E5/$A$5
11	0	=B6-B5*$A6	=C6-C5*$A6	=D6-D5*$A6	=E6-E5*$A6
12	0	=B7-B5*$A7	=C7-C5*$A7	=D7-D5*$A7	=E7-E5*$A7
13	0	=B8-B5*$A8	=C8-C5*$A8	=D8-D5*$A8	=E8-E5*$A8
14		Второе преобразование			
15	1	1	0	2	5
16	0	1	=C11/$B$11	=D11/$B$11	=E11/$B$11
17	0	0	=C12-C11*$B$12/$B$11	=D12-D11*$B$12/$B$11	=E12-E11*$B$12/$B$11
18	0	0	=C13-C11*$B$13/$B$11	=D13-D11*$B$12/$B$11	=E13-E11*$B$13/$B$11
19		Третье преобразование			
20	1	1	0	2	5
21	0	1	-0,5	0,5	-5,5
22	0	0	1	=D17/$C$17	=E17/$C$17
23	0	0	0	=D18-D17*$C$18/$C$17	=E18-E17*$C$18/$C$17
24		Четвертое преобразование			
25	1	1	0	2	5
26	0	1	-0,5	0,5	-5,5
27	0	0	1	2	3
28	0	0	0	1	=E23/D23
29		Обратный ход			
30	x4=	=E28			
31	x3=	=E27-D27*B30			
32	x2=	=E26-D26*B30-C26*B31			
33	x1=	=E25-D25*B30-C25*B31-B25*B32			

	A	B	C	D	E
1	Решение системы линейных алгебраических уравнений Ax=b				
2	методом Гаусса				
3		A			b
4					
5	1	1	0	2	5
6	2	4	-1	5	-1
7	1	3	0	5	-3
8	3	7	-3	9	-13
9	Первое преобразование				
10	1	1	0	2	5
11	0	2	-1	1	-11
12	0	2	0	3	-8
13	0	4	-3	3	-28
14	Второе преобразование				
15	1	1	0	2	5
16	0	1	-0,5	0,5	-5,5
17	0	0	1	2	3
18	0	0	-1	1	-6
19	Третье преобразование				
20	1	1	0	2	5
21	0	1	-0,5	0,5	-5,5
22	0	0	1	2	3
23	0	0	0	3	-3
24	Четвертое преобразование				
25	1	1	0	2	5
26	0	1	-0,5	0,5	-5,5
27	0	0	1	2	3
28	0	0	0	1	-1
29	Обратный ход				
30	x4=	-1			
31	x3=	5			
32	x2=	-2,5			
33	x1=	9,5			

Рис.1

Второй вариант не столь очевиден. Среди встроенных в Excel математических программ программы решения систем уравнений, строго говоря, нет.

Неточно так же, как для решения уравнений было использовано средство Подбор параметра, для решения систем может быть использовано средство, предназначенное совсем для другой цели (для решения задач оптимизации).

Это средство - Поиск решения. Поясним его использование и приведем примеры применения.

Средство Поиск решения активируется в меню Сервис (если это средство не установлено, то это необходимо сделать).

Предварительно проводят следующую подготовительную работу.

Отводят для каждой переменной по ячейке.

Вводят формулы для вычисления правых частей уравнений системы (по одной формуле в ячейку).

После этого запускают Поиск решения. Возможный вид экрана для подготовки к решению той же системы, что и выше, приведен на рис.

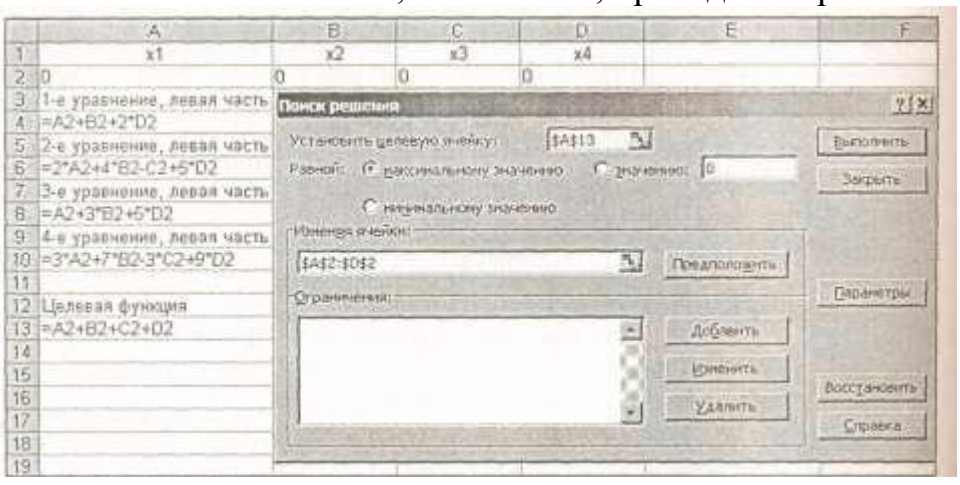


Рис.2

На этом рисунке под переменные отведены ячейки A2:D2, под формулы — ячейки A4, A6, A8 и A10. Какие именно числовые значения переменных будут введены в ячейки A2:D2, при решении системы линейных уравнений значения не имеет (итерационная процедура, заложенная в Поиск решения, стартует с этих значений).

Еще один элемент в таблице — целевая функция. Она в данном случае особой роли не играет, но какую-нибудь формулу ввести необходимо, иначе Поиск решения работать не может (напомним, что эта программа нацелена на другой класс задач). Точно так же неважно, как установлен флажок: «максимальному значению» или «минимальному значению».

Теперь необходимо ввести то, что в форме на рис. 2 именуется ограничениями.

Щелкнув по кнопке Добавить, получают другую форму (рис.3).

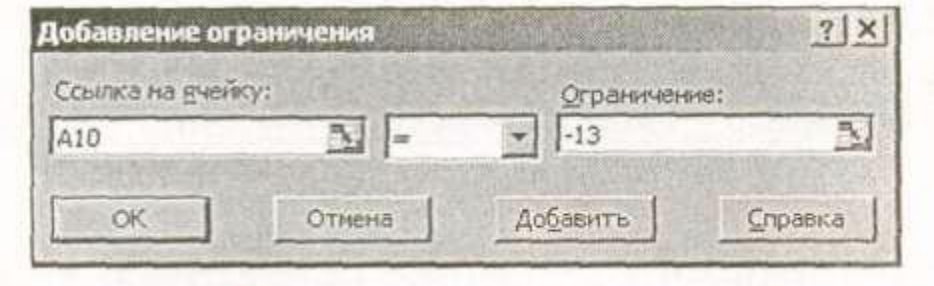


Рис3.

В форму, изображенную на рис. 3. надо ввести четыре условия (по числу уравнений системы). На рисунке отражено последнее условие. Ссылка на ячейку A10 обусловлена тем, что в ней -формула для левой части 4-го уравнения, знак '=' выбран из меню, число (-13) введено с клавиатуры (правая часть 4-го уравнения). После ввода последнего ограничения нажимают кнопку ОК и возвращаются в основную форму Поиск решения. Щелкнув по кнопке Выполнить, получают результат (рис. 4).

Рис4.

Каким методом это решение получено, можно узнать, щелкнув по кнопке Параметры (см. рис. 2). Там, в частности, есть знакомый нам метод Ньютона (наверняка сильно видоизмененный, поскольку применяется на самом деле к решению гораздо более сложной задачи).

Поиск решения можно попытаться применить и к решению систем нелинейных уравнений. В этом случае выбор начального приближения очень важен, в

зависимости от него решение может быть получено или не получено и могут быть получены разные решения.

Контрольные вопросы и задания

1) Запустить программу Excel, открыть рабочую книгу лабораторных работ. Создать в ней новый лист, дать ему имя «Мет.итераций».

2) Оформить рабочий лист.

Значения коэффициентов матрицы P и вектора b взять по своему варианту, таблица 4 Приложения.

3) Привести систему к виду (4).

4) Умножить матрицу P и вектор b на транспонированную матрицу P^T

Преобразовать коэффициенты по формулам (4'). В формулах вычисления коэффициентов a_{ij} и b_i ($i \neq j$) использовать абсолютную адресацию. Коэффициенты a_{ii} задать равными нулю.

5) Для начала итерационного процесса задать значение вектора x^0 равным b.

Ввести формулы для вычисления x^{k+1} и d. При ссылке на матрицу A можно использовать абсолютные адреса, для вектора b удобно ячейкам с элементами вектора присвоить имена и в формулах ссылаться на них:

$x^{k+1}=$ =H24+v_b1
=H25+v_b2
=H26+v_b3
=H27+v_b4

Ячейкам со значениями величин ε и d также следует присвоить имена для использования в формулах.

Для переноса значений x^{k+1} на следующий шаг в ячейки x^k необходимо использовать функцию ЕСЛИ с условием $\varepsilon<d$.

6) Копировать ячейки, относящиеся к одному шагу итераций до тех пор, пока $\varepsilon<d$.

7) Выделить в отдельные ячейки окончательный ответ с 3 десятичными знаками. Сравнить полученные значения с точным решением системы уравнений и решением, полученным выше.

СЛАУ можно записать в матричном виде

$$Ax=b$$

где A – матрица коэффициентов при неизвестных;

x – вектор неизвестных;

b – вектор свободных членов.

При решении СЛАУ можно использовать метод обратной матрицы. Тогда решение можно найти по формуле

$$x=A^{-1}\cdot b$$

Пример. Решить систему линейных алгебраических уравнений (СЛАУ)

$$\begin{cases} 6x_1 - 2x_2 + x_3 = 4 \\ 3x_1 + 2x_3 = 6 \\ 6x_1 - 2x_2 + 12x_3 = 1 \end{cases}$$

Для решения СЛАУ методом обратной матрицы необходимо:

1. Задать матрицу коэффициентов при неизвестных *A* и вектор свободных членов *b*.
2. Проверить детерминант матрицы А.
3. Если детерминант не равен 0, найти обратную матрицу A^{-1}
4. Умножить обратную матрицу на вектор свободных членов.

Решение СЛАУ в системе Scilab.

Специальные матричные функции необходимые для решения СЛАУ методом обратной матрицы:

- Функция *det(A)* вычисляет определитель квадратной матрицы А.
- Функция *inv(A)*вычисляет обратную матрицу к матрице А.

Решение СЛАУ с помощью этого метода приведено на рис. 20.

```
-->A=[6 -2 1;3 0 2; 6 -2 12];

-->b=[4;6;1];

-->det(A)
 ans  =

    66.

-->A1=inv(A);

-->x=A1*b
 x  =

    2.1818182
    4.4090909
  - 0.2727273
```

Рис. 20. Решение СЛАУ с использованием обратной матрицы

Решить СЛАУ, заданную в виде уравнения $A\bar{x} - \vec{b} = 0$ используя функцию *linsolve(A, b)* можно, (рис. 21).

```
-->A=[6 -2 1;3 0 2; 6 -2 12];

-->b=[-4;-6;-1];

-->x=linsolve(A,b)
 x  =

    2.1818182
    4.4090909
  - 0.2727273
```

Рис. 21. Решение СЛАУ с использованием функции linsolve

Решите систему методом обратной матрицы.

Таблица 4 Варианты СЛАУ

№ варианта	Система линейных алгебраических уравнений	корни системы	№ варианта	Система линейных алгебраических уравнений	корни системы
1.	$\begin{cases} 3x-4y+4z-2k=4 \\ 6x+2y-3k=-5 \\ -9x+5y-2z+k=-2 \\ x-6y+z+3k=8 \end{cases}$	2 5 9 9	2.	$\begin{cases} 7x-2y+3z+5k=-15 \\ 3x+y-4z+3k=31 \\ x-4y+4z-k=-21 \\ x+y+4z-5k=-53 \end{cases}$	-2 -4 -8 3
3.	$\begin{cases} x-6y+3z-k=-17 \\ 5y-3z+2k=15 \\ -4x+y+z-k=-6 \\ x+y-2z+3k=2 \end{cases}$	3 4 1 -1	4.	$\begin{cases} x-y-3z-4k=10 \\ -5x-3y-z-3k=-1 \\ 4x-3y+3z+9k=24 \\ 2x-y+5z+k=2 \end{cases}$	3 -5 -2 1
5.	$\begin{cases} 2x+3y-3z+5k=-2 \\ -2x+2y-z+4k=4 \\ 5x+y+4z+9k=-24 \\ -7x-9y-2z-7k=-8 \end{cases}$	-2 4 0 -2	6.	$\begin{cases} -2x+4y-2z-4k=-4 \\ 2x+2y-z-k=9 \\ 4x-2y+3z+4k=0 \\ 5x+3y-z+3k=15 \end{cases}$	4 -2 -4 -1
7.	$\begin{cases} -x+4y+5z+3k=-2 \\ -2x-y-z-5k=-13 \\ 4x-y+z+k=-9 \\ 4x+3y-2z+k=7 \end{cases}$	-2 1 -4 4	8.	$\begin{cases} 5x+y-2z-4k=2 \\ 3x-4y+3z-5k=-10 \\ -3x-4y+4z-5k=3 \\ -4x-y+5z-4k=-4 \end{cases}$	-3 -1 -5 -2
9.	$\begin{cases} x+4y-z-5k=12 \\ -2x+5y-z+4k=-8 \\ -x+5y-4z-3k=0 \\ 2x+5y+2z+4k=11 \end{cases}$	4 1 1 -1	10.	$\begin{cases} 4x-3y-2z-3k=-12 \\ -5x+3y+3z+3k=8 \\ 2x-3y-5z-4k=0 \\ -2x+2y-4z-4k=4 \end{cases}$	-1 1 -5 5
11.	$\begin{cases} 4x-4y-z+5k=-9 \\ -3x+4y-5z-4k=-12 \\ 3x-2y+z-k=6 \\ 2x-5y-5z-5k=3 \end{cases}$	-1 -2 3 -2	12.	$\begin{cases} -3x-3y-5z+5k=7 \\ 2x+y+5z+k=-6 \\ -5x-2z-4z-5k=6 \\ -2x+y+2z-5k=3 \end{cases}$	-3 4 -1 1
13.	$\begin{cases} x-2y+3z+2k=9 \\ x-2y-3z-k=9 \\ x-5y-2z-4k=1 \\ -4x+4y+2z+5k=-4 \end{cases}$	3 -2 -2 4	14.	$\begin{cases} -5x+2y+2z+4k=10 \\ 3x+3y-2z-2k=-1 \\ 2x+y-z+k=8 \\ 5x+3y+4z+k=21 \end{cases}$	2 1 1 4

№ варианта	Система линейных алгебраических уравнений	корни системы	№ варианта	Система линейных алгебраических уравнений	корни системы
15.	$\begin{cases} 3x+3y+4z-k=-33 \\ x-2y-z+5k=14 \\ 5x-2z+k=-10 \\ x-y+5z+5k=-3 \end{cases}$	-3 -5 -2 1	16.	$\begin{cases} -4x+y-2z+2k=7 \\ -5x-y+3z-4k=20 \\ 2x-3y-2z+5k=4 \\ 5x-5y+3z-k=5 \end{cases}$	-3 -3 2 1
17.	$\begin{cases} 4x+4y+5z+4k=5 \\ -5x+2y-z-2k=11 \\ -2x+3y-2z-7k=-9 \\ x-2y+z+5k=9 \end{cases}$	-2 3 -3 4	18.	$\begin{cases} 4x-4y-2z+3k=13 \\ -5x-y+4z+3k=-8 \\ -4x-2y-5z+5k=-23 \\ 2x-2y-3z+4k=0 \end{cases}$	3 -2 2 -1
19.	$\begin{cases} 3x+3y-5z-k=11 \\ 4x-2y-z-5k=-5 \\ -3x-2y-5z+5k=7 \\ -4x-4y+2z+5k=-9 \end{cases}$	-2 2 -2 -1	20.	$\begin{cases} -2x-2y-3z-2k=-9 \\ -x-3y+z-5k=7 \\ -x+y+5z-k=5 \\ 5x-5y-4z+5k=8 \end{cases}$	5 -2 3 -3
21.	$\begin{cases} -5x-2y+2z-k=-10 \\ -4x-4y+z+k=7 \\ 2x+y-4z-3k=-13 \\ -4x+3y-z+4k=0 \end{cases}$	2 -3 -1 4	22.	$\begin{cases} -5x-2y-2z+3k=6 \\ 3x-4y+3z+2k=-5 \\ -x+y-5z-k=-12 \\ -x+y-z-k=4 \end{cases}$	-5 -2 4 -5
23.	$\begin{cases} 3x-y+2z-3k=13 \\ -2x+5y+3z+4k=-9 \\ -4x-5y-z+2k=-7 \\ 4x+3y+4z-5k=8 \end{cases}$	5 -2 1 2	24.	$\begin{cases} -2x+3y-2z+4k=2 \\ -4x-y+2z+4k=0 \\ -4x-5y-z+2k=-7 \\ 4x+3y+4z-5k=8 \end{cases}$	-3 4 2 -3
25.	$\begin{cases} 3x-5y-z-4k=8 \\ -5x-4y+4z-2k=-24 \\ -2x-5y-z-4k=-2 \\ -4x+3y-2z-2k=-12 \end{cases}$	2 -2 -4 3	26.	$\begin{cases} 5x-y-3z+k=-7 \\ -4x+5y+4z-3k=0 \\ 3x-5y-4z+k=6 \\ x+y-z-3k=3 \end{cases}$	-2 -2 -1 -1

Контрольные вопросы

1. Основные понятия алгебры матриц. Действия с матрицами. Норма матрицы. Транспонированная и обратная матрицы.
2. Методы решения систем линейных алгебраических уравнений. Метод исключения Гаусса.
3. Вычисление обратной матрицы методом Гаусса.
4. Вычисление определителя методом Гаусса.
5. Итерационные методы решения систем линейных алгебраических уравнений. Метод Якоби. Условие сходимости итераций.
6. Итерационные методы решения систем линейных алгебраических уравнений. Метод Гаусса-Зейделя. Условие сходимости итераций.
7. Метод релаксации решения системы линейных алгебраических уравнений.

Лабораторная работа №5

Тема: **Численное дифференцирование и интегрирование.**

1. **Цель работы**

Использование численных методов решения задач дифференцирования и интегрирования.

2. **Учебные вопросы, подлежащие рассмотрению:**

– Постановка задачи.

– Интерполяционный многочлен Лагранжа и интерполяционный многочлен Ньютона.

– Численное дифференцирование с помощью многочленов Лагранжа и Ньютона.

– Метод трапеции, метод Симпсона

– Численное интегрирование методами трапеции и Симпсона (парабол)

3. **Порядок выполнения работы**

Задание 1.

Вычислить значение производной функции, заданной таблично, используя интерполяционные формулы Лагранжа или Ньютона.

Задание 2.

Вычислить интеграл от заданной функции f(x) на отрезке [a,b] при делении отрезка на 10 равных частей следующими способами 1) по формуле трапеций; 2) по формуле Симпсона.

Исполнение: применить интерполяционные формулы Лагранжа или Ньютона используя любой инструментальный пакет для вычисления производной. Использовать формулы трапеции и формулы Симпсона для вычисления определенного интеграла.

Оценка: Сопоставление полученных результатов, решаемых различными методами.

Методические указания

Контрольные вопросы и задания

1. Вычислить значение производной функции, заданной таблично, используя интерполяционные формулы Лагранжа или Ньютона.

Вариант

X	$\sin x$
0,60	0,56464
0,65	0,60519
0,70	0,64422
0,75	0,68164
0,80	0,71736
0,85	0,75128
0,90	0,78333
0,95	0,81342
1,00	0,84147
1,05	0,86742
1,10	0,89121

Задание 2.

Вычислить интеграл от заданной функции f(x) на отрезке [a,b] при делении отрезка на 10 равных частей следующими способами 1) по формуле трапеций; 2) по формуле Симпсона.

Отрезок интегрирования разбивается на 10 равных частей. Для расчетов удобно составить единую таблицу значений по схеме:

Xi	y /2 (1=0, 10)	y_i (i=l, 2, 3, ..., 9)	2 y_i (i=1, 3, 5, 7, 9)

По каждому из трех столбцов таблицы находятся суммы соответствующих значений подынтегральной функции (при этом по столбцу *y,- для* формулы трапеций находится сумма всех элементов столбца, а для формулы Симпсона — только с четными индексами).

Вариант

Вариант	*m*	*a*	*b*
1	$0{,}37e^{\sin x}$	0	1
2	$0,5 + x \lg x$	1	2
3	$(x+1{,}9)\sin(x/3)$	1	2
4	$\frac{1}{x}\ln(x+2)$	2	3
5	$\frac{3\cos x}{2x+1,7}$	0	1
6	$(2x+0{,}6)\cos(x/2)$	1	2
7	$2,6x^2 \ln x$	1,2	2,2
8	$(x^2+1)\sin(x-0{,}5)$	0,5	1,5
9	$x^2 \cos(x/4)$	2	3
10	$\frac{\sin(0,2x-3)}{x^2+1}$	3	4
11	$3x + \ln x$	1	2
12	$4xe^{x^2}$	-1	0

Рассчитать значение определенного интеграла можно с помощью формулы Ньютона-Лейбница

$$\int_a^b f(x)dx = F(b) - F(a)$$

где $F(x)$ – первообразная подинтегральной функции.

В функциях интегрирования в Scilab реализованы различные численные алгоритмы. Наиболее универсальной командой интегрирования в Scilab является

[I, err] = intg(a, b, name [,er1 [,er2]]),

где *name* — имя функции, задающей подынтегральное выражение (функция может быть задана в виде набора дискретных точек, т.е. таблицей или с

помощью внешней функции);

a и b – пределы интегрирования;

erl и er2 — абсолютная и относительная точность вычислений (необязательные параметры).

Пример. Вычислить значение интеграла.

$$\int_{2}^{5} \sin(x) \cdot \cos(2x)$$

Первообразной подинтегральной функции является функция

$$F(x) = -\frac{\cos(3x)}{6} + \frac{\cos(x)}{2}.$$

Решение, полученное по формуле Ньютона-Лейбница, приведено на рис. 22.

```
-->int=-cos(15)/6+cos(5)/2-(-cos(6)/6+cos(2)/2)
 int  =

    0.6365475
```

Рис. 22. Вычисление интеграла по формуле Ньютона-Лейбница

При использовании функции *intg* необходимо вначале задать подинтегральную функцию как внешнюю. Это можно сделать с помощью конструкции function … endfunction.

Решение, полученное с помощью функции *intg*, приведено на рис. 23.

```
-->function y=f(x),y=sin(x).*cos(2*x),endfunction;

-->[Int,er]=intg(2,5,f)
 er  =

   5.348D-14
 Int  =

   0.6365475
```

Рис. 23. Использование функции intg

На рис. 24 приведен файл-сценарий вычисления рассматриваемого интеграла.

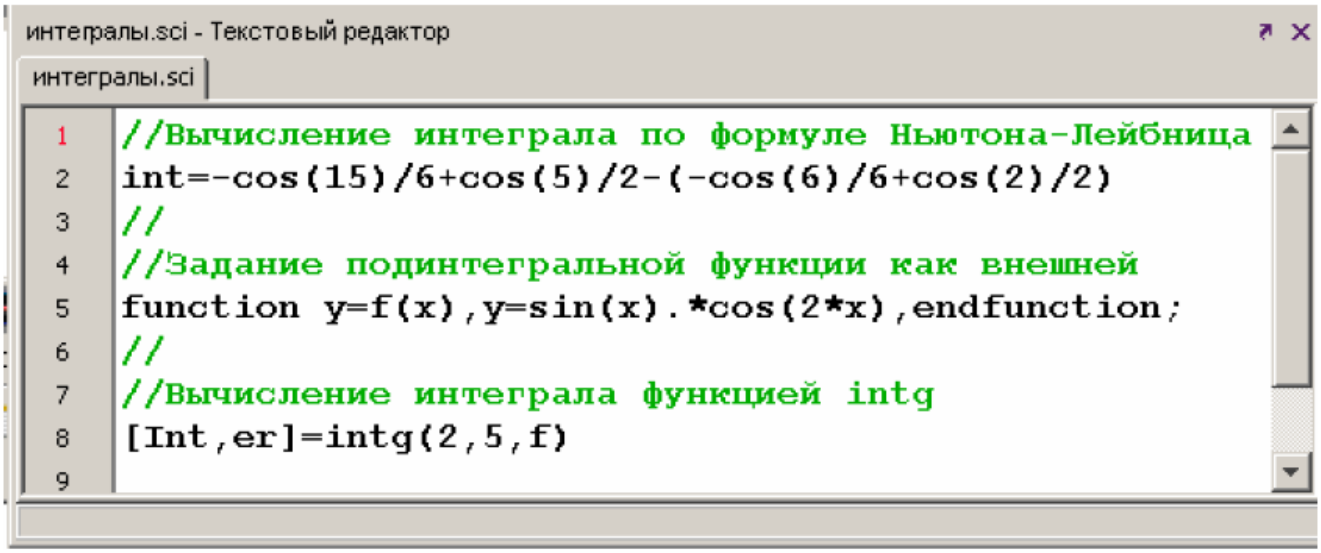


Рис. 24. Файл-сценарий вычисления рассматриваемого интеграла

Аппроксимация экспериментальных данных

Очень часто, особенно при анализе эмпирических данных возникает необходимость найти в явном виде функциональную зависимость между величинами *x* и *y*, которые получены в результате измерений.

При аналитическом исследовании взаимосвязи между двумя величинами x и y производят ряд наблюдений и в результате получается таблица значений:

x	x_1	x_2	…	x_i	…	x_n
y	y_1	y_2	…	y_i	…	y_n

Эта таблица обычно получается как итог каких-либо экспериментов, в которых x_i (независимая величина) задается экспериментатором, а y_i получается в результате опыта. Поэтому эти значения y_i будем называть эмпирическими или опытными значениями.

Между величинами x и y существует функциональная зависимость, но ее аналитический вид обычно неизвестен, поэтому возникает практически важная задача - найти эмпирическую формулу

$$y = f(x; a_1, a_2, \dots, a_m),$$

(где $a_1, a_2, \dots, a_m$ - параметры), значения которой при $x = x_i$ возможно мало отличались бы от опытных значений y_i $(i = 1, 2, \dots, n)$.

Метод наименьших квадратов позволяет по экспериментальным данным подобрать такую аналитическую функцию, которая проходит настолько близко к экспериментальным точкам, насколько это возможно.

Согласно методу наименьших квадратов наилучшими коэффициентами $a_1, a_2, \dots, a_m$ считаются те, для которых сумма квадратов отклонений найденной эмпирической функции от заданных значений функции

$$S(a_1, a_2, \dots, a_m) = \sum_{i=1}^{n} \left[f(x_i; a_1, a_2, \dots, a_m) - y_i \right]^2$$

будет минимальной.

Нахождение коэффициентов a_i сводится к решению системы.

$$a_0 n + a_1 \Sigma x_i + a_2 \Sigma x_i^2 + \dots + a_k \Sigma x_i^k = \Sigma y_i$$
$$a_0 \Sigma x_i + a_1 \Sigma x_i^2 + a_2 \Sigma x_i^3 + \dots + a_k \Sigma x_i^{k+1} = \Sigma x_i y_i$$
…………………………………………………..
$$a_0 \Sigma x_i^k + a_1 \Sigma x_i^{k+1} + a_2 \Sigma x_i^{k+2} + \dots + a_k \Sigma x_i^{2k} = \Sigma x_i^k y_i$$

Построение эмпирической формулы состоит из двух этапов: выяснение общего вида этой формулы и определение ее наилучших параметров.

Эта система упрощается, если эмпирическая формула линейна относительно параметров a_i, тогда система - будет линейной.

Конкретный вид системы зависит от того, из какого класса эмпирических формул мы ищем зависимость. В случае линейной зависимости $y = a_1 + a_2 x$ система примет вид:

$$\begin{cases} a_1 n + a_2 \sum\limits_{i=1}^{n} x_i = \sum\limits_{i=1}^{n} y_i , \\ a_1 \sum\limits_{i=1}^{n} x_i + a_2 \sum\limits_{i=1}^{n} x_i^2 = \sum\limits_{i=1}^{n} x_i \, y_i . \end{cases}$$

Эта линейная система может быть решена любым известным методом (методом Гаусса, простых итераций, формулами Крамера).

Для реализации этой задачи в Scilab предусмотрена функция

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

Где *F* – аппроксимирующая функция, параметры которой необходимо подобрать;

z – матрица исходных данных;

c – вектор начальных приближений;

a – вектор коэффициентов;

S – сумма квадратов отклонений измеренных значений от расчетных

Вид аппроксимирующей функции, подбирается как наиболее подходящий для заданных экспериментальных данных, это может быть:

- Линейная,
- Логарифмическая,
- Полиномиальная,
- Экспоненциальная и др.

Чаще в качестве аппроксимирующей функции выбирают полином необходимой степени.

Пример. Пусть в результате эксперимента были получены некоторые данные, отображенные в виде таблицы. Требуется построить аналитическую зависимость, наиболее точно описывающую результаты эксперимента.

x	0.0	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0
y	0.57	0.70	0.89	1.10	1.32	1.50	1.58	1.40	1.32	1.10	0.90

В качестве функции аппроксимирующей данные эксперимента следует взять полином третьей степени, у которого четыре коэффициента, которые необходимо найти.

На рис. 25 приведен файл-сценарий подбора аппроксимирующего полинома с помощью функции *datafit.*

*аппроксимация_2.sci

```
1  //Задание функции, вычисляющей разность между
2  //экспериментальными и теоретическими значениями
3  function [y]=P(c,z)
4    y=z(2)-c(1)-c(2)*z(1)-c(3)*z(1)^2-c(4)*z(1)^3;
5  endfunction
6  //Задание исходных данных
7  x=[0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0];
8  y=[0.57 0.7 0.89 1.1 1.32 1.5 1.58 1.4 1.32 1.1 0.9];
9  //Построение графика экспериментальных данных
10 plot(x,y,'rx')
11 //
12 //Формирование матрицы исходных данных
13 z=[x;y];
14 //
15 //Формирование вектора начальных значений
   коэффициентов,размерность которого должна
16 //совпадать с количеством искомых коэффициентов
17 //искомых коэффициентов
18 c=[0;0;0;0];
19 //
20 //Определение коэффициентов полинома - a и суммы
   квадратов
21 //отклонений err
22 [a,err]=datafit(P,z,c)
23 //
24 //Задание вектора значений аргумента в диапазоне 0; 2
25 //Вычисление вектора значений найденного полинома
   для
26 //заданного вектора значений аргумента
27 t=[0:0.05:2];
28 pl=a(1)+a(2)*t+a(3)*t^2+a(4)*t^3;
29 //
30 //Построение графика найденного полинома
31 plot(t,pl)
32 xgrid()
```

Рис. 25. Файл-сценарий аппроксимации экспериментальных данных полиномом 3 степени

В результате работы функции *datafit* была подобрана аналитическая зависимость в виде полинома

$$P = -0{,}3x3 + 0{,}17x2 + 1{,}03x + 0{,}52\ ,$$

а сумма квадратов отклонений измеренных значений от расчетных составила 0,038 (рис. 26).

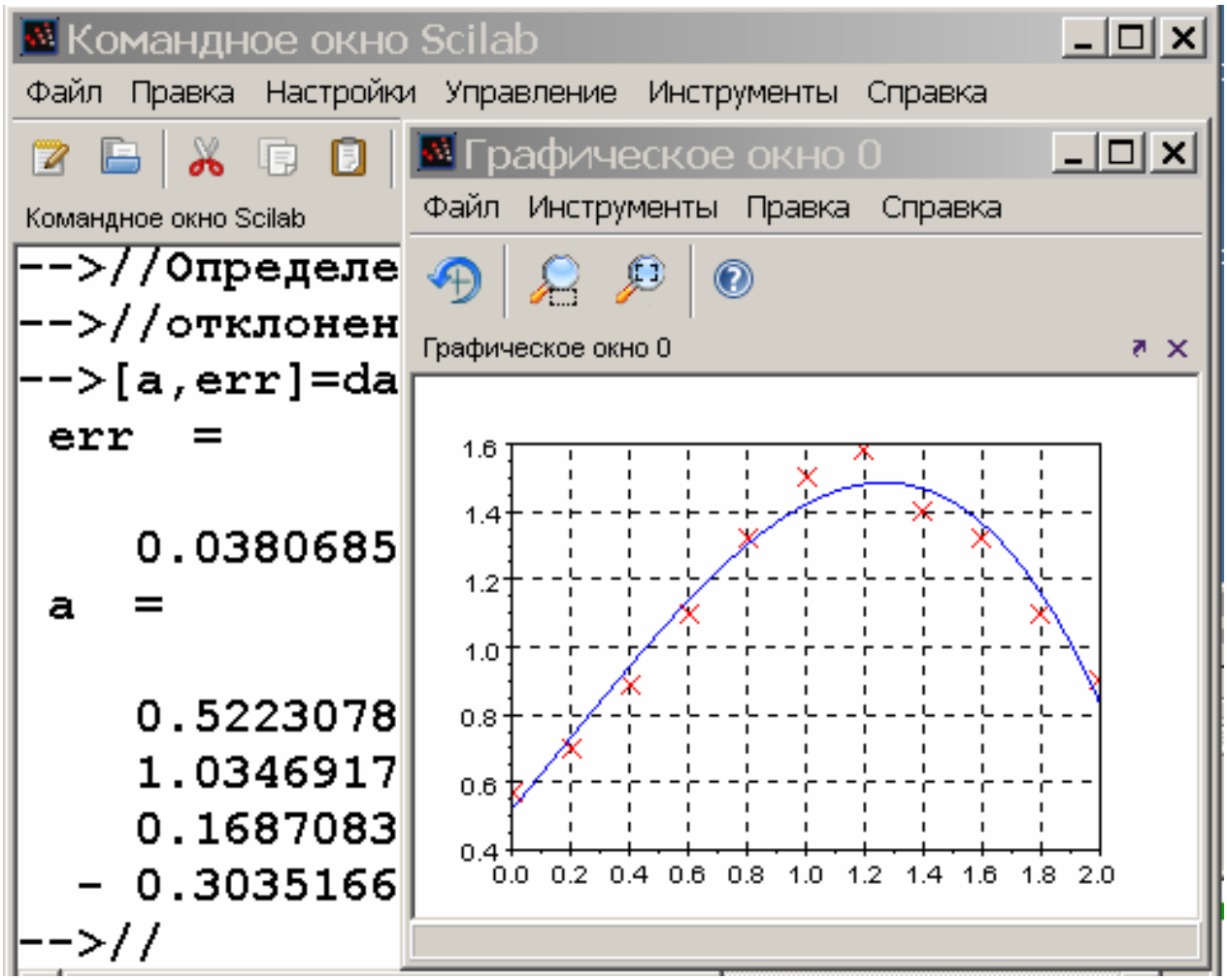


Рис. 26. Решение задачи аппроксимации

Контрольные вопросы

1. Квадратурные формулы прямоугольников (с оценкой точности).
2. Квадратурная формула трапеций (с оценкой точности).
3. Квадратурная формула Симпсона (с оценкой точности).
4. Квадратурные формулы гауссова типа.

Лабораторная работа №6

Тема: **Решение обыкновенных дифференциальных уравнений.**

1. **Цель работы**

Использование методов решения обыкновенных дифференциальных уравнений для решения конкретных задач строительства.

2. **Учебные вопросы, подлежащие рассмотрению:**

Решение дифференциального уравнения:

- ✓ методом Коши,
- ✓ методом Эйлера,
- ✓ Методом Эйлера-Коши,
- ✓ Рунге-Кутта 4-го порядка
- ✓ Адамса.

3. **Порядок выполнения работы**

Задание

Решить задачу Коши для дифференциального уравнения $y=f(x,y)$ на отрезке $[a,b]$ при заданном начальном условии и шаге интегрирования h.

Номер варианта соответствует порядковому номеру в списке.

Исполнение: С помощью инструментальных пакетов MS Office методами Эйлера, Рунге-Кутта 4-го порядка и Адамса, предусмотрев вывод полученных решений в виде таблиц и графиков.

Оценка: Сопоставление полученных результатов, решаемых различными методами

Методические указания

Дана система дифференциальных уравнений:

$$\begin{cases} \frac{dy_1}{dx} = F_1(x, y_1, y_2, \ldots, y_n) \\ \frac{dy_2}{dx} = F_2(x, y_1, y_2, \ldots, y_n) \\ \ldots \\ \frac{dy_n}{dx} = F_n(x, y_1, y_2, \ldots, y_n) \end{cases}$$

, где n – *размерность системы*.

Рассмотрим задачу Коши для данной системы. Пусть известны начальные условия при $x_0 = a$: $y_1(x_0) = y_{10}$, $y_2(x_0) = y_{20}$, …, $y_n(x_0) = y_{n0}$. Требуется найти $y_1(x)$, $y_2(x)$,…, $y_n(x)$, проходящие через заданные точки: (x_0,y_{10}), (x_0,y_{20}), …, (x_0,y_{n0}).

Методы решения одного дифференциального уравнения можно обобщить и на их системы.

Метод Рунге-Кутта 4-го порядка для системы ОДУ 1-го порядка

Расчетные формулы метода Рунге-Кутта 4-го порядка для системы ОДУ 1-го порядка:

$$y_{i,j+1} = y_{i,j} + \frac{h}{6}\cdot(K_{0i} + 2K_{1i} + 2K_{2i} + K_{3i})$$

где $h = \frac{b-a}{m}$;

$a = x_0$;

$b = x_m$;

m – количество узлов;

$i = \overline{1,n}$ – номер функции;

$j = 0,1,\ldots,m-1$ – номер узла;

$K_{0i} = F_i(x_j, y_{1j}, y_{2j}, \ldots, y_{nj})$;

$$K_{1i} = F_i\left(x_j + \frac{h}{2}, y_{1j} + \frac{h}{2}\cdot K_{0i}, y_{2j} + \frac{h}{2}\cdot K_{0i}, \ldots, y_{nj} + \frac{h}{2}\cdot K_{0i}\right);$$

$$K_{2i} = F_i\left(x_j + \frac{h}{2}, y_{1j} + \frac{h}{2}\cdot K_{1i}, y_{2j} + \frac{h}{2}\cdot K_{1i}, \ldots, y_{nj} + \frac{h}{2}\cdot K_{1i}\right);$$

$$K_{3i} = F_i(x_j + h, y_{1j} + hK_{2i}, y_{2j} + hK_{2i}, \ldots, y_{nj} + hK_{2i}).$$

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.

Для того чтобы решить обыкновенное дифференциальное уравнение вида

$$\frac{dy}{dt} = f(t, y),\ y(t_0) = y_0$$

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_Кутта четвертого порядка;
- "rkf" - указывают при выборе пятиэтапного метода Рунге_Кутта четвертого порядка; "fix" - тот же метод Рунге_Кутта, 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 на примере следующих задач.

ЗАДАЧА 8.1. Решить задачу Коши

$$\frac{dx}{dt}+x=\sin(xt),x(0)=1.5.$$

Перепишем уравнение следующим образом:

$$\frac{dx}{dt}=-x+\sin(xt),x(0)=1.5.$$

Далее представим его в виде внешней функции, так как показано в листинге и применим функцию *y=ode(x0,t0,t,f)*, в качестве параметров которой будем использовать

f - ссылка на предварительно созданную функцию *f(t,x)*;

t - координаты сетки;

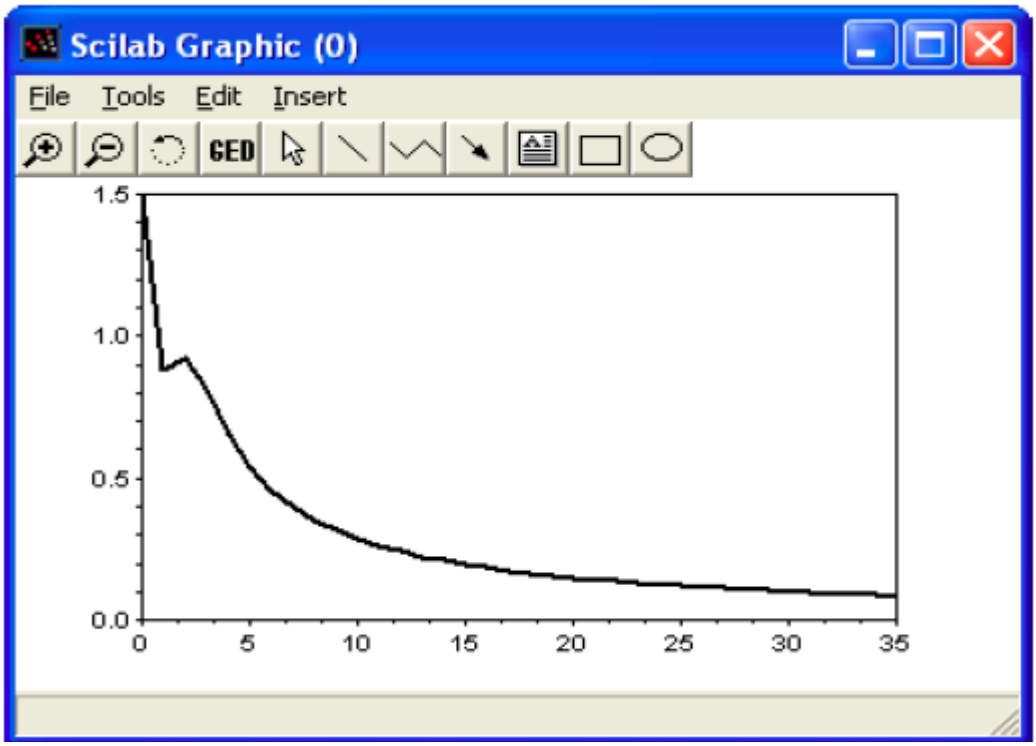
x0,t0 - начальное условие *x(0)=1.5*;

y - результат работы функции.

График, моделирующий процесс, описанный заданным уравнением, представлен на рис.8.1.

```
-->function yd=f(t,x),yd=-x+sin(t*x),endfunction;
-->x0=1.5;t0=0;t=0:1:35;
-->y=ode(x0,t0,t,f);
-->plot(t,y)
```

Листинг 8.1



из задачи 8.1

ЗАДАЧА 8.2. Решить задачу Коши

$$x' = \cos(x\,y),$$
$$y' = \sin(x + t\,y),$$
$$x(0) = 0, y(0) = 0.$$

на интервале [0; 10].

Листинг 8.2 содержит функцию, описывающую заданную систему обыкновенных дифференциальных уравнений и команды Sciab необходимые для ее численного и графического решения (рис.8.2).

```
>>%Функция, описывающая систему дифференциальных уравнений
-->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 дифференциальных уравнений
-->x0=[0;0];t0=0;t=0:1:10;
-->y=ode(x0,t0,t,syst);
>>%Формирование графического решения
-->plot(t,y)
```

Листинг 8.2

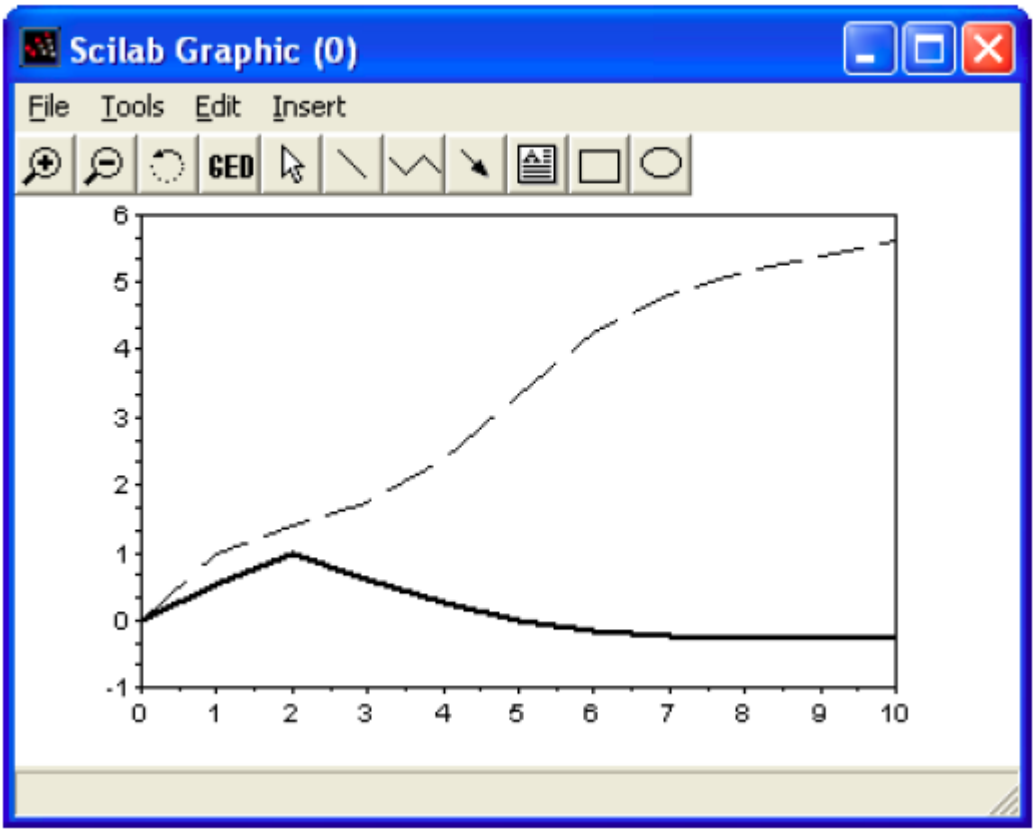


Рис. 8.2. Решение задачи 8.2

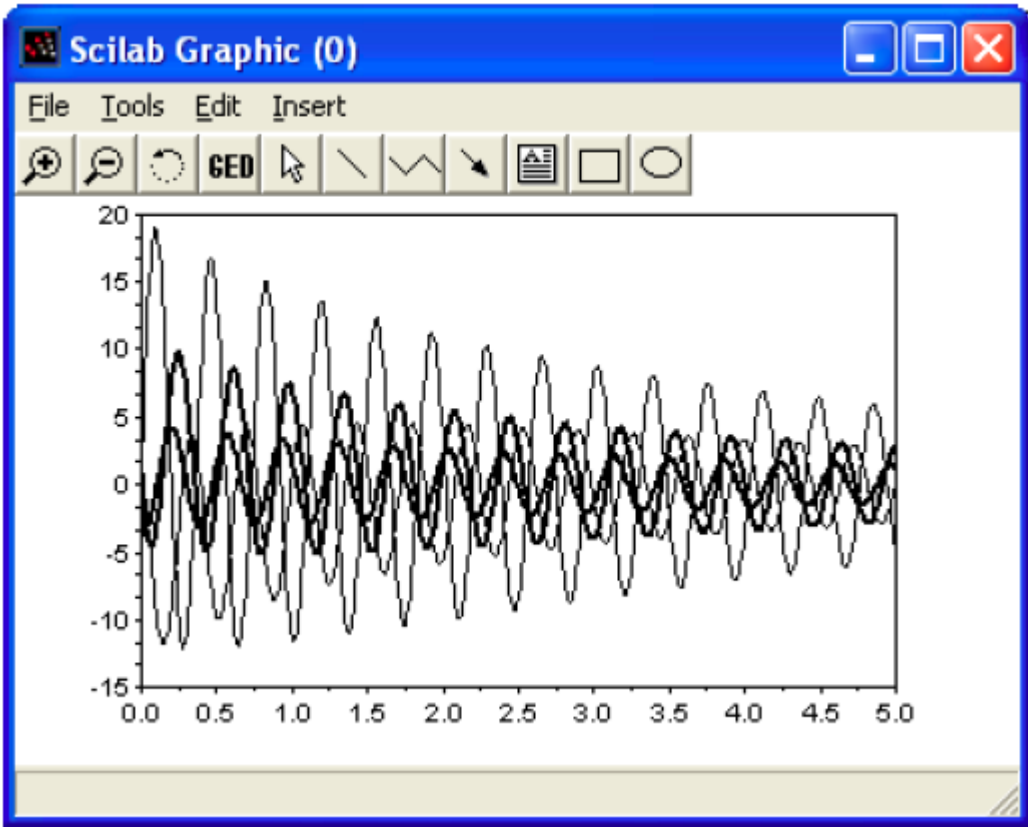


Рис. 8.3. Графическое решение жесткой системы

ЗАДАЧА 8.3. Найти решение задачи Коши для следующей жесткой системы:

$$\frac{dX}{dt}=\begin{pmatrix}119.46 & 185.38 & 126.88 & 121.03\\ -10.395 & -10.136 & -3.636 & 8.577\\ -53.302 & -85.932 & -63.182 & 54.211\\ -115.58 & -181.75 & -112.8 & -199\end{pmatrix}X;X(0)=\begin{pmatrix}1\\1\\1\\1\end{pmatrix}$$

Решение системы показано в листинге 8.3.
Графическое решение показано на рис. 8.3.

```
-->B=[119.46 185.38 126.88 121.03;-10.395 -10.136 -3.636 8.577;
-->-53.302 -85.932 -63.182 -54.211;-115.58 -181.75 -112.8 -199];
-->function dx=syst1(t,x), dx=B*x,endfunction
-->function J=Jac(t,y),J=B,endfunction
-->x0=[1;1;1;1]; t0=0; t=0:0.01:5;
-->y=ode("stiff",x0,t0,t,syst1,Jacobian);
-->plot(t,y)
```

Листинг 8.3

ЗАДАЧА 8.5. Решить следующую краевую задачу

$$\frac{d^2x}{dt^2}+4\frac{dx}{dt}+13=e^{\sin(t)},\ x(0.25)=-1,\ x'(0.25)=1.$$

на интервале [0.25; 2].
Преобразуем уравнение в систему, сделав замену:

$$y=\frac{dx}{dt}$$

$$\frac{dy}{dt}=-4y-13x+e^{\sin(t)},\ \frac{dx}{dt}=y,\ y(0.25)=1,\ x(0.25)=-1.$$

Составим функцию вычисления системы и решим ее так, как показано в листинге 8.5. График решения приведен на рис. 8. 5.

```
function F=FF(t,x)
F=[-4*x(1)-13*x(2)+exp(t);x(1)];
endfunction
-->//Решение системы дифференциальных уравнений
-->X0=[1;-1];t0=0.25;t=0.25:0.05:2;
-->y=ode("stiff",X0,t0,t,FF);
-->//Вывод графика решения
-->plot(t,y)
```

Листинг 8.5

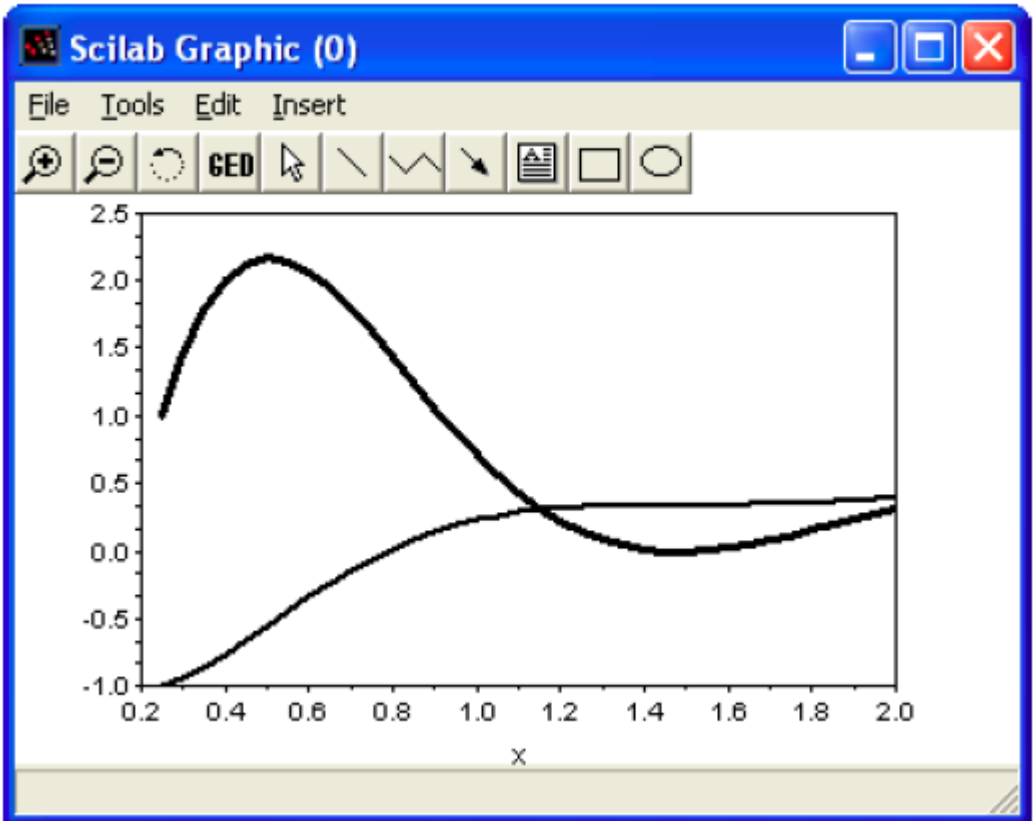


Рис.8.5. Решение задачи 8.5

Контрольные вопросы

1) Сформулируйте задачу Коши для обыкновенных дифференциальных уравнений первого порядка.

2) Что является решением дифференциального уравнения: а) в высшей математике, б) в прикладной математике?

3) Какие методы решения дифференциальных уравнений называются одношаговыми, многошаговыми? Приведите примеры.

4) Сравните решения, полученные на первом, втором шаге методами Эйлера, Рунге-Кутта и разложением в ряд Тейлора (трудоемкость, погрешность…).

5) Как оценить погрешность применяемого метода? Как ее уменьшить?

6) Сравните одношаговые и многошаговые методы решения дифференциальных уравнений, указав достоинства и недостатки первых и вторых.

7) Что такое экстраполяционные и интерполяционные методы (формулы) Адамса?

8) Можно ли применять: а) только экстраполяционные методы Адамса, б) только интерполяционные?

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В каком виде представляются все данные в Scilab ?
2. Как вводятся элементы вектора-строки ?
3. Как обратиться к блоку последовательно расположенных элементов вектора?
4. Какие знаки используются для поэлементного умножения, деления, возведения в степень векторов?

5. Каким образом можно описать функцию в Scilab?
6. Какой знак используется В Scilab в качестве оператора присваивания?
7. Какая функция используется для приближенного вычисления корня уравнения по заданному начальному приближению?
8. Как задать полином в Scilab?
9. Какую функцию используют для определения значений полинома?
10. Какую функцию используют для нахождения корней полинома?
11. Какую функцию используют для решения нелинейного уравнения?
12. Какие функции используются для нахождения интегралов в Scilab?
13. Где набираются команды Scilab?
14. Как можно узнать имена всех категорий встроенных функций Scilab?
15. Какой знак используется для того, чтобы продолжить длинное выражение на следующей строке?
16. Как используется команда plot для построения графика?
17. Как задать стиль и цвета линий при построении графика?.
18. С помощью какой команды можно построить график в полярных координатах?
19. Как построить график функции, заданной параметрически?
20. Какие операции допускаются для массивов?
21. Как решить систему линейных уравнений?
22. Какую функцию используют для аппроксимации экспериментальных данных полиномом?

СПИСОК ЛИТЕРАТУРЫ

1. Глебова Т.А., Чиркина М.А, Пышкина И.С. Прикладная математикаи: учебное пособие,— Пенза, ПГУАС, 2020.— 96 с.

2. Сулейманов Р.Р. Компьютерное моделирование математических задач [Электронный ресурс]: учебное пособие/ Сулейманов Р.Р.— Электрон. текстовые данные.— М.: БИНОМ. Лаборатория знаний, 2012.— 381 с.— Режим доступа: http://www.iprbookshop.ru/12228.— ЭБС «IPRbooks», по паролю

3. Алябьева В.Г. Теория алгоритмов [Электронный ресурс]: учебное пособие / Алябьева В.Г., Пастухова Г.В.— Электрон. текстовые данные.— Пермь: Пермский государственный гуманитарно-педагогический университет, 2013.— 125 с.— Режим доступа: http://www.iprbookshop.ru/32100.— ЭБС «IPRbooks», по паролю

4. Информационные системы. Часть III [Электронный ресурс]: практикум.— М.: Московский городской педагогический университет, 2013.— 204 с.—

Режим доступа: http://www.iprbookshop.ru/26490.— ЭБС «IPRbooks»

5. Алексеев Е. Р. Scilab: Решение инженерных и математических задач / Е. Р. Алексеев. О. В. Чеснокова. Е. А. Рудченко. — М. : ALT Linux : БИНОМ. Лаборатория знаний. 2008. — 260с.