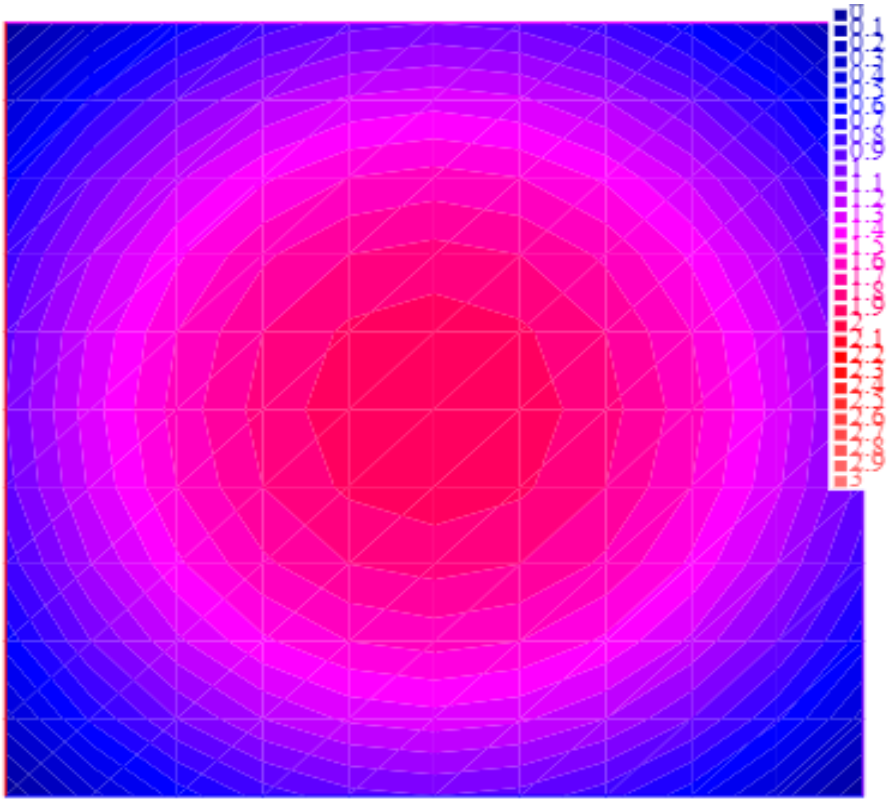
УНИВЕРСИТЕТ ИТМО

В. Ю. Захарова, М. И. Волкович

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ ПО КУРСУ «МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ ФИЗИЧЕСКИХ ПРОЦЕССОВ»

ЧАСТЬ 2



Санкт-Петербург
2021

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

УНИВЕРСИТЕТ ИТМО

В. Ю. Захарова, М. И. Волкович

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ ПО КУРСУ «МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ ФИЗИЧЕСКИХ ПРОЦЕССОВ» ЧАСТЬ 2

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

РЕКОМЕНДОВАНО К ИСПОЛЬЗОВАНИЮ В УНИВЕРСИТЕТЕ ИТМО
по направлению подготовки 14.03.01 Ядерная энергетика и теплофизика в качестве Учебно-методического пособия для реализации основных профессиональных образовательных программ высшего образования бакалавриата

Санкт-Петербург
2021

Захарова В.Ю., Волкович М.И., Методические указания к выполнению лабораторных работ по курсу "Математическое моделирование физических процессов". Часть 2 – СПб: Университет ИТМО, 2021. – 97 с.

Рецензент(ы):
Трифанова Екатерина Станиславовна, кандидат физико-математических наук, доцент, преподаватель (квалифицированная категория «преподаватель») факультета систем управления и робототехники, Университета ИТМО.

Учебно-методическое пособие адресовано студентам, обучающимся по направлению 14.03.01 «Ядерная энергетика и теплофизика».

В пособие входит описание трех лабораторных работ, посвященных математическому моделированию физических процессов, в частности, процессов теплопроводности. Моделирование осуществляется в математических пакетах Scilab и FreeFem++:

- Моделирование стационарного процесса в математическом пакете Scilab,
- Моделирование нестационарного процесса в математическом пакете Scilab,
- Моделирование двумерного процесса в среде FreeFem++ с применением метода конечных элементов.

Работы снабжены пошаговыми примерами выполнения задач и описанием заданий для самостоятельной работы.

УНИВЕРСИТЕТ ИТМО

Университет ИТМО – ведущий вуз России в области информационных и фотонных технологий, один из немногих российских вузов, получивших в 2009 году статус национального исследовательского университета. С 2013 года Университет ИТМО – участник программы повышения конкурентоспособности российских университетов среди ведущих мировых научно-образовательных центров, известной как проект «5 в 100». Цель Университета ИТМО – становление исследовательского университета мирового уровня, предпринимательского по типу, ориентированного на интернационализацию всех направлений деятельности.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Одним из важнейших этапов математического моделирования физических процессов является этап поиска решения для разработанной математической модели. Современные программные средства позволяют существенно ускорить и автоматизировать данный процесс, уменьшить количество ошибок вне зависимости от того, используются аналитические или численные методы расчета. Среди всего многообразия программных средств большой интерес представляют те прикладные пакеты программ для математических расчетов, которые распространяются свободно по лицензиям, допускающим их применение для учебных и научных целей без дополнительной платы.

В настоящем учебном пособии представлено описание трех лабораторных работ по курсу «Математическое моделирование физических процессов», предназначенных для обучающихся бакалавриата по направлению подготовки 14.03.01 «Ядерная энергетика и теплофизика». В процессе выполнения работ студенты получают навыки решения для ряда типичных математических постановок, характерных для задач теплопроводности. Однако, в связи со схожестью математических постановок, данные решения могут применяться и при решении ряда задач прочности, гидродинамики и аэродинамики, электростатики. Для реализации используются программный мультифункциональный математический пакет Scilab и пакет FreeFem++, реализующий решение методом конечных элементов для произвольных математических моделей.

В пособии представлены следующие лабораторные работы:

- Моделирование стационарного процесса в математическом пакете Scilab,
- Моделирование нестационарного процесса в математическом пакете Scilab,
- Моделирование двумерного процесса в среде FreeFem++ с применением метода конечных элементов.

Каждая лабораторная работа состоит из теоретических сведений, пошагового практического примера или нескольких примеров, выполнение которых является обязательным, а также требований для получения допуска и контрольного задания по вариантам.

В пособии используются следующие обозначения.

Курсивом выделяются те действия, порядок выполнения которых уже был описан ранее. Эти действия подробно не поясняются, их необходимо выполнить самостоятельно.

Жирным шрифтом выделяются те положения, на которые необходимо обратить особое внимание.

Шрифтом Tahoma выделяются названия пунктов меню, переменных, параметров, а также те выражения, которые необходимо ввести с командной строки или в файл. Шрифтом `Courier New` выделяются сообщения в окне вывода результатов.

При выборе пунктов меню используется следующая сокращенная запись: «Пункт 1 > Подпункт 10», что означает, что необходимо вначале выбрать в меню Пункт 1, а затем из появившегося подменю выбрать Подпункт 10.

ЛАБОРАТОРНАЯ РАБОТА №3. МОДЕЛИРОВАНИЕ СТАЦИОНАРНОГО ПРОЦЕССА В МАТЕМАТИЧЕСКОМ ПАКЕТЕ SCILAB

1 Основные сведения о работе

1.1 Цель работы

Целью работы является получение навыков работы со встроенным в математический пакет Scilab языком программирования и применение этих навыков для решения задач моделирования физических процессов.

1.2 Порядок выполнения работы

В данной работе теоретическая часть изложена совместно с примерами, разделенными на три части. В начале каждой части описаны особенности программирования для задания математических моделей пакета Scilab, которые дальше иллюстрируются примерами. Все указанные примеры необходимо выполнить.

Первая глава посвящена функциям, создаваемым пользователем, вторая – операторам цикла и ветвления, третья – организации ввода-вывода в интерактивном режиме.

Для получения допуска ко второй части работы необходимо предъявить пять файлов, полученных в результате выполнения практических примеров, ответить на теоретические вопросы

В качестве контрольного задания необходимо вычислить скалярное поле физической величины.

1.3 Используемое программное обеспечение

Для выполнения лабораторной работы необходим компьютер с установленным математическим пакетом Scilab версии 5 и выше.

1.4 О совместимости с математическим пакетом MATLAB

Языки программирования в математических пакетах Scilab и MATLAB схожи. Однако некоторые операторы могут иметь немного отличающийся синтаксис или название. В целом принципы, используемые для программирования в математических пакетах Scilab и MATLAB, одинаковы. Программа для математического пакета Scilab может быть достаточно просто переделана в программу для пакета MATLAB. Заметные отличия наблюдаются, как правило, только при организации ввода-вывода, создании графического интерфейса и взаимодействии с пользователем.

2 Функции пользователя в пакете Scilab

2.1 Функции в пакете Scilab и в математике

В предыдущих лабораторных работах уже были рассмотрены некоторые функции, встроенные в пакет Scilab. Например, функции sin(), cos(), ndgrid(), plot2d().

Понятие функции в пакете Scilab шире, чем понятие функции в математике, однако, между ними много общего.

Известно, что в математике функция должна иметь один или несколько аргументов, т. е. переменных, от которых зависит значение функции. В математике функция может иметь и несколько значений. Такие функции называются вектор-функциями.

Аналогично в пакете Scilab функция может иметь несколько аргументов – входных переменных. Однако функция в пакете Scilab может не иметь входных переменных или иметь переменное число входных переменных. Функции в пакете Scilab также возвращают значение. Значение, аналогично вектор-функциям, может быть вектором или матрицей. Однако функция в пакете Scilab может и не возвращать никакого значения. Более того, помимо вычисления значения функция может осуществлять и другие действия-строить график, выводить какие-либо значения на экран.

В пакете Scilab существует множество встроенных функций, но пользователь может при необходимости сам определить (задать) собственные функции.

2.2 Файлы сценария и функции

Все возможности программирования в пакете Scilab, как правило, реализуются посредством создания файлов сценария или функций пользователя. Между файлами сценария и функциями пользователя есть много общего:

– Файлы сценария и функции могут быть созданы (определены) как из командной строки, так и в отдельном файле.

– Для выполнения файлов сценария или функций пользователя их необходимо вначале загрузить в пакет Scilab.

– В файлах сценария, функциях пользователя и командной строке используются одни и те же команды, операторы и функции.

Существенное отличие между файлами сценария и функциями пользователя заключается в следующем. Все переменные в файлах сценария являются глобальными. Т. е. любая переменная, определенная в файле сценария и не удаленная в нем, далее будет доступна из командной строки и из других файлов сценария. Команды внутри файла сценария могут изменить уже существующие переменные, заданные вне этого файла. Это может быть опасно,

так как неизвестно, какие переменные и с какой целью были определены до запуска файла сценария.

Напротив, все переменные, определенные внутри функций, являются локальными. Т. е. они не видны ниоткуда, кроме самой функции. Более того, функция «не видит» других переменных, определенных в среде, кроме встроенных переменных. Даже если имя переменной внутри функции совпадет с именем переменной в среде, эти две переменные будут считаться разными. Функция взаимодействует с переменными в среде и в других функциях только посредством входных и выходных параметров. Возможно определить глобальные переменные, видимые внутри других функций, но, как правило, в этом нет необходимости. Такое поведение функций типично для большинства языков программирования.

Если функция вызывает функцию, находящуюся в одном с ней файле (подфункцию), то нет необходимости загружать эту подфункцию в среду. Подфункции могут быть размещены и в файлах сценария.

2.3 Синтаксис при задании функции

Вне зависимости от того, где определяется функция-в командной строке или внутри файла, она записывается следующим образом:

```
function [y1,y2,…,yn]=fun1(x1,x2,…,xm)
операторы
endfunction[1]
```

Здесь y1, y2, …, yn – имена выходных параметров (значений) функции[2]. Выходные параметры могут отсутствовать. Всем выходным параметрам внутри функции должны быть присвоены значения. Тип выходных параметров автоматически определяется их значениями. fun1 – имя функции, требования к имени функции такие же, как к имени переменной. x1, x2, …, xm – входные параметры[3]. Тип входных параметров определяется значениями, переданными в функцию при ее вызове. Значения входных параметров можно использовать в выражениях внутри функции. Входные параметры рассматриваются внутри функции как константы, т. е. их значения нельзя изменить. операторы – команды, выполняемые внутри функции. endfunction – ключевое слово, которое указывает на конец функции.

Функции пользователя вызываются из командной строки и другой функции таким же способом, как и встроенные функции пакета Scilab.

[1] В пакете MATLAB функции записываются также, но ключевое слово endfunction в конце функции не требуется.

[2] Число выходных параметров может быть переменным. Работа с ними здесь не рассматривается. При необходимости смотрите помощь по параметру varargout.

[3] Число входных параметров может быть переменным. Работа с ними здесь не рассматривается. При необходимости смотрите помощь по параметру varargin.

Возможно использование функции в выражениях. Для вызова функции fun1, описанной выше, необходимо набрать:

```
[a1,a2,…,a3]=fun1(b1,b2,…,b3)
```

a1, a2, ..., a3 – переменные, которым будут присвоены вычисленные значения функции, b1, b2, ..., b3 – входные параметры (аргументы) функции.

2.4 Создание функции из командной строки

Создадим функцию sinexp, которая вычисляет значение выражения $\sin(x)\cdot\exp(x)$. Входной параметр функции – значение аргумента x, выходной параметр – значение функции $\sin(x)\cdot\exp(x)$.

2.4.1. *Войдите в среду Scilab.*

2.4.2. Задайте функцию из командной строки, как указано ниже. Не забывайте нажимать ENTER после каждой строки.

```
→function y=sinexp(x)
→y=sin(x)*exp(x);
→endfunction
```

2.4.3. Теперь функцию можно использовать. Вычислите значение функции для аргумента $x = 0$.

```
→sinexp(0)
 ans  =    0.
```

2.4.4. Присвойте значение функции в точке $x = \pi$ переменной v1.

```
→v1=sinexp(%pi)
 v1  =    2.834D-15
```

2.4.5. Значение функции можно использовать в выражениях, так же, как это делалось со встроенными функциями. В качестве аргумента функции также могут быть использованы выражения.

```
→5*sinexp(%pi/2)
 ans  =    24.052387
```

2.4.6. Среда Scilab сама не проверяет, является ли тип входных переменных в функции правильным. Например, при попытке указать массив в качестве аргумента функции sinexp, среда Scilab будет работать с ним так, как указано в функции. В данном случае она попытается умножить матрицы значений sin(x) и exp(x) друг на друга по правилам матричного умножения.

```
→x=0:%pi/40:%pi;
→sinexp(x)
 !--error 10
Некорректное умножение.
at line       2 of function sinexp called by :
sinexp(x)
```

Ошибка вызвана тем, что матрицы имеют размеры, неподходящие для матричного умножения.

2.4.7. Можно заставить работать функцию sinexp правильно и в том случае, когда в качестве аргумента служит массив. Для этого, очевидно, нужно заменить матричное умножение поэлементным. Определите функцию sinexp2.

```
→function y=sinexp2(x)
→y=sin(x).*exp(x);
→endfunction
```

2.4.8. Новая функция sinexp2 работает с числами так же, как и функция sinexp.

```
→sinexp2(%pi)
 ans  =    2.834D-15
```

2.4.9. Функция sinexp2 корректно работает с массивами значений. Убедитесь, что вычисления происходят поэлементно.

```
→sinexp2(x)^4
```

2.4.10. Если вычисления в функции происходят поэлементно, то ее можно использовать для построения графика.

```
→plot2d(x,sinexp2(x))
```

Результат приведен на рисунке 1.1.

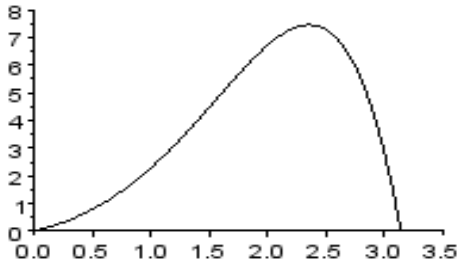


Рис. 1.1 – Результат построения графика посредством поэлементного вычисления функции

2.4.11. ***Сохраните полученный график в графическом файле exl3_1 и предъявите его преподавателю для получения допуска к выполнению лабораторной работы.***

2.4.12. Закройте построенный график.

2.5 Создание функции в файле

Сложные функции неудобно создавать из командной строки. Функции, созданные в командной строке, неудобно редактировать. В большинстве случаев рекомендуется создавать функции в файле и загружать их из него.

[4] Здесь результаты работы не приведены, так как они занимают много места.

2.5.1. *Откройте текстовый редактор SciNotes, встроенный в среду Scilab.*

2.5.2. Наберите текст функции, приведенный ниже, и сохраните его в файле graph1.sci.

```
function graph1(x0,xk,N,col)
  x=x0:((xk-x0)/N):xk;
  y=2*x;
  plot2d(x,y,style=col)
endfunction
```

Эта функция не имеет выходных параметров. Она строит график функции $y = 2x$ на отрезке $[x0, xk]$, причем значения функции вычисляются в $N + 1$ точке, распределенной по отрезку равномерно. Параметры x0, xk и N являются входными аргументами функции. Также в качестве входного аргумента задается цвет линии на графике col.

2.5.3. Перед использованием функции ее надо загрузить в среду. Для этого выберите из главного меню текстового редактора пункт Выполнение > ...файла без отображения команд[5] (В старых версиях программы Выполнение > Загрузить в Scilab[6]).

2.5.4. Теперь можно использовать функцию. Для построения графика функции $y = 2x$ на отрезке [1; 5] по шести (5+1) точкам зеленым цветом наберите в командной строке:

```
→graph1(1,5,5,3)
```

Полученный график приведен справа (рис.1.2). Закройте его.

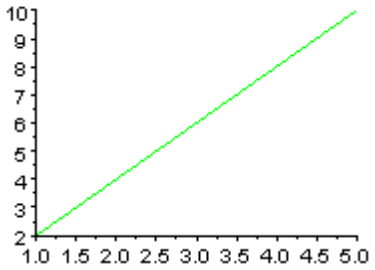


Рис. 1.2 - График функции на отрезке [1,5]

2.5.5. Для построения графика функции $y = 2x$ на отрезке [2; 4] по семи (6+1) точкам красным цветом наберите:

```
→graph1(2,4,6,5)
```

Полученный график функции приведен справа (рис. 1.3).

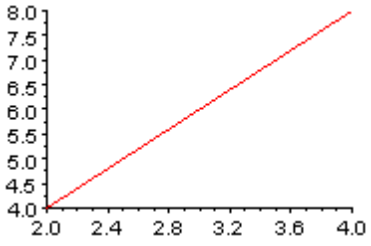


Рис. 1.3 - График функции на отрезке [2; 4]

2.5.6. ***Сохраните полученный график в графическом файле exl3_2 и предъявите его преподавателю для получения допуска к выполнению лабораторной работы.***

2.5.7. Закройте окно с графиком.

---

[5] Файлы функций могу быть очень большими, поэтому не рекомендуется отображать команды в командном окне.

[6] Функции в пакете MATLAB доступны без загрузки в среду. Они должны находиться в рабочем каталоге.

2.6 Вложенные функции

Функции могут быть определены внутри файла сценария или файла функции. Для примера будет рассмотрена функция, осуществляющая построение двух графиков функций в одном окне.

Описание задачи

Необходимо составить функцию graph2, которая вычисляет два значения функции $y = \frac{2}{2+8x+x^2} exp(x)$ для заданных значений аргумента $x_1, x_2 > 0$ и строит два графика этой функции один под другим в одном окне на отрезках $[0;\ x_1,]$ и $[0; x_2]$ соответственно.

2.6.1. Наберите или скопируйте текст файла graph2.sci в текстовый редактор SciNotes (не забудьте переключиться на новую вкладку, команда главного меню Файл > Создать). Разберитесь в том, как работает функция.

```
function [y1,y2]=graph2(x1,x2)
  //Функция для простроения графика
y(x)=2/(2+8x+x^2)*exp(x)
  // на отрезках [0,x1] и [0,x2]
  //y1=y(x1), y2=y(x2)
  function[7] y0=ygr(x0)
    //Подфункция, вычисляет значение y0=y(x0)
    // и строит график y(x) на отрезке [0,x0]
    N=25; //Задание количества интервалов, на
которые
    // разбивается отрезок для построения графика
функции
    x=0:(x0/N):x0; //Задание аргумента функции y(x)
    y=(2)[8]./(2+(8).*x+x.^2).*exp(x); //Вычисление
значений
                                       //функции y(x)
    plot2d(x,y); //Построение графика функции в
выбранных
                 // осях координат
    y0=y(N+1); //Вычисление значения функции ygr
  endfunction //Конец подфункции
  scf(1); //Создание нового графического окна
  subplot(211); //Выбор верхних осей координат
  y1=ygr(x1); //Вычисление параметра y1 функции
graph2
              // с одновременным построением графика
```

[7] Подфункция может быть определена в любом месте основной функции до ее вызова.

[8] Если не поставить скобки вокруг цифры, то Scilab воспримет выражение, как 2.0/.., т.е. как матричное деление, а не поэлементное.

```
  subplot(212); //Выбор нижних осей координат
  y2=ygr(x2); //То же, но для y2
endfunction
```

2.6.2. *Сохраните и загрузите функцию* graph2 *в среду*.

2.6.3. Простой вызов функции, без указания переменных, в которые необходимо поместить выходные параметры, дает следующее.

```
→graph2(2,5)
 ans  =    0.6717324
```

Т.е. в переменную ans помещено значение только первого выходного параметра. При этом строится необходимый график (рис. 1.4). Закройте его.

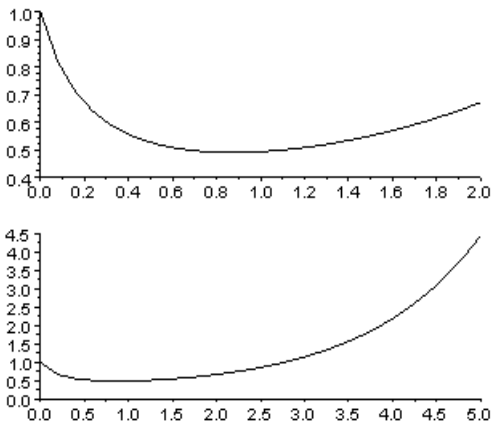


Рис. 1.4 – Результирующий график

2.6.4. Чтобы получить значения всех выходных параметров, нужно указать соответствующие переменные при вызове функции. Значения будут помещены в эти переменные.

Сравните следующее с пунктом 2.6.3.

```
→[a1,a2]=graph2(2,5)
 a2  =    4.4302436
 a1  =    0.6717324
```

2.6.5. ***Сохраните полученный график в графическом файле exl3_3 и предъявите его преподавателю для получения допуска к выполнению лабораторной работы***.

2.6.6. Закройте окно с графиком.

2.6.7. Обратите внимание, что подфункция ygr, определенная внутри функции graph2, доступна только из этой функции. Попытка ее вызова из командной строки приводит к ошибке.

```
→ygr(5)
        !--error 4
Неизвестная переменная: ygr
```

3 Операторы цикла и ветвления

3.1 Операторы в пакете Scilab

К основным операторам встроенного в пакет Scilab языка программирования можно отнести, в том числе, следующие:

– оператор присваивания =,
– условный оператор if,

– оператор альтернативного выбора select[9],
– оператор цикла for,
– оператор цикла с предусловием while.

Перечисленные операторы отличаются от подобных операторов в других языках программирования лишь синтаксисом, поэтому далее рассматривается только синтаксис этих операторов. Операторы рассматриваются на соответствующих примерах. Оператор присваивания был рассмотрен в предыдущих лабораторных работах.

3.2 Условный оператор if и логические выражения

Существует обычная и расширенная форма оператора if. В обычной форме он имеет следующий синтаксис:

```
if условие then
операторы 1
else
операторы 2
end
```

Работает оператор следующим образом: если условие истинно, то выполняются операторы 1, в противном случае выполняются операторы 2. Условие – это переменная логического типа или выражение, возвращающее значение логического типа.

Переменная логического типа может иметь два значения: истина (T) или ложь (F). Эти переменные могут использоваться в логических выражениях. Вещественное или целое число, не равное нулю, в логических выражениях заменяется на T, а ноль – на F.

В логических выражениях используются стандартные логические операторы, которые имеют по два равноценных обозначения:

- and, & – логическое «и»,
- or, | – логическое «или»,
- not, ~ – логическое «не».

Также используются операторы отношения, результат выполнения которых имеет логический тип. Операторы отношения применяются к числовым значениям. В пакете Scilab применяются следующие операторы:

- < – меньше,
- > – больше,
- == – равно,
- ~=, <> – не равно,
- <= – меньше или равно,
- >= – больше или равно.

[9] В пакете MATLAB этот оператор записывается как switch

Расширенная форма оператора if имеет вид, указанный далее. Она позволяет проверять несколько условий.

```
if условие 1 then
операторы 1
elseif условие 2 then
операторы 2
elseif условие 3 then
операторы 3
…
else
операторы
end
```

Если условие 1 - истина, то выполняются операторы 1, иначе проверяется условие 2. Если условие 2 - истина, то выполняются операторы 2, иначе проверяется условие 3. Если ни одно из условий не выполнено, то выполняются операторы после слова else.

3.2.1. Рассмотрим возможные логические выражения (наберите их в командной строке).

```
→(1>2)&(2==2)
 ans  =  F
```

То же самое можно записать другим образом. Аргументом функции and служит матрица, над элементами которой нужно произвести логическую операцию «и».

```
→and([1>2 2==2])
 ans  =  F
```

3.2.2. При наличии нескольких логических операций их порядок такой же, как в математике[10]. Следующие два выражения эквивалентны.

```
→1&0|1
 ans  =  T
→1|1&0
 ans  =  T
```

3.2.3. Логическое «не»

```
→~1
 ans  =  F
```

3.2.4. Вернитесь к функции graph2, созданной в предыдущем пункте. В этой функции предполагается, что ее входные аргументы – положительные. Однако ничто не мешает вызвать ее с отрицательными аргументами.

```
→graph2(-2,3)
```

[10] В пакете MATLAB порядок логических операций не такой, как в математике, что крайне неудобно.

```
 ans  =  - 0.0270671
```

3.2.5. Для исправления проблемы, описанной в пункте 3.2.4, необходимо организовать проверку входных аргументов функции. В случае если хотя бы один из них отрицательный – графики не должны строиться, и в качестве значений функции возвращается Nan. Создайте функцию graph3, изменив тело основной функции graph2 следующим образом[11] (не забудьте изменить название функции):

```
function [y1,y2]=graph3(x1,x2)
  …
  if (x1>0)&(x2>0) then //проверка, являются ли оба
                        // аргумента положительными[12]
    scf(1);
    subplot(211);
    y1=ygr(x1);
    subplot(212);
    y2=ygr(x2);
  else //выполняется, если хотя бы один из
аргументов
       // отрицателен
    y1=%inf/%inf; //в y1 помещается значение Nan
    y2=%inf/%inf; //в y2 помещается значение Nan
  end
endfunction
```

3.2.6. Теперь при отрицательном аргументе функции график не строится. Не забудьте загрузить функцию в среду.

```
→graph3(-2,4)
 ans  =   Nan
```

3.2.7. Удобнее было бы, чтобы график не строился только для того аргумента функции, который является отрицательным. Создайте функцию graph4, изменив функцию graph3.

```
function [y1,y2]=graph4(x1,x2)
  …
  scf(1);
  if x1>0 then //Проверка, является ли первый
аргумент
               // положительным
    subplot(211);
    y1=ygr(x1);
  else //Если первый аргумент отрицательный
```

[11] Приведена только изменяемая часть функции.

[12] Комментарии можно не вводить.

```
        y1=%inf/%inf; //в y1 помещается значение Nan
    end
    if x2>0 then //Проверка, является ли второй
аргумент
                 // положительным
      subplot(212);
      y2=ygr(x2);
    else //Если второй аргумент отрицательный
       y2=%inf/%inf; //в y2 помещается значение Nan
    end
```

3.2.8. Теперь один отрицательный аргумент влияет только на построение соответствующего графика. График для положительного аргумента функции строится (рис. 1.5).

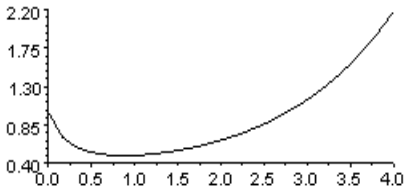


Рис. 1.5 - График для положительного аргумента функции

```
→[a1, a2]=graph4(-2,4)
 a2  =    2.183926
 a1  =   Nan
```

3.2.9. Закройте построенный график.

3.3 Оператор альтернативного выбора select

Вместо оператора альтернативного выбора select можно использовать оператор if, но при использовании первого оператора программа будет компактнее, если действия зависят от значения одной переменной. Синтаксис оператора select следующий:

```
select параметр
case значение 1 then операторы 1
case значение 2 then операторы 2
…
else операторы
end
```

Если значение параметра равно значению 1, то выполняются операторы 1, иначе, значение параметра равно значению 2, то выполняются операторы 2 и т.д. Если значение параметра не совпадает ни с одним значением в группах case, то выполняются операторы, которые идут после ключевого слова else.

3.3.1. Добавим в качестве входного параметра в функцию graph4 параметр c, который показывает, какую палитру следует установить в графическом окне. Значение 0 соответствует палитре graycolormap, 1 – springcolormap, 2 – summercolormap, 3 – autumncolormap. Если значение

параметра какое-то другое, то палитра – hsvcolormap. Пусть график выводится цветом 18 палитры.

Создайте функцию graph5, изменив функцию graph4 следующим образом. Разберитесь, как работает функция.

```
function [y1,y2]=graph5(x1,x2,c)
  // c - параметр, определяющий палитру
```

Внутри функции ygr

```
…
plot2d(x,y,style=18); //Построение графика функции
…
```

В начале основного тела функции graph5

```
f1=scf(1); //Создание нового графического окна и
получение его дескриптора
select c //Выбор палитры в зависимости от значения
переменной c
case 0 then f1.color_map=graycolormap(32);
case 1 then f1.color_map=springcolormap(32);
case 2 then f1.color_map=summercolormap(32);
case 3 then f1.color_map=autumncolormap(32);
else f1.color_map=hsvcolormap(32);
end
…
```

3.3.2. Убедитесь, что палитра изменяется в зависимости от параметра, не забудьте закрывать графические окна.

→`graph5(5,3,1)`
→`graph5(5,3,0)`

3.4 Оператор цикла for

Оператор цикла for предназначен для повторения одного и того же блока операторов заданное число раз. Он имеет следующий вид:

```
for x=xn:hx:xk
операторы
end
```

x – параметр цикла (матрица 1x1), xn – начальное значение параметра цикла, xk – конечное значение параметра цикла, hx – шаг цикла. Если hx=1, то этот параметр можно опустить.

Выполнение цикла начинается с присвоения параметру стартового значения (x=xn). Затем следует проверка, не превосходит ли параметр конечное значение (x>xk). Если x>xk, то цикл считается завершенным, и управление передается следующему за телом цикла оператору. Если же x<xk,

то выполняются операторы в цикле (тело цикла). Далее параметр цикла увеличивается на hx (x=x+hx). После этого снова производится проверка значения параметра цикла, и алгоритм повторяется. При выходе из цикла значение параметра цикла x равно его значению до последнего суммирования параметра hx.

3.4.1. В функции ygr, входящей в graph5, заменим экспоненту суммой ряда $1+\sum_{i=1}^{\infty}\frac{x^i}{i}$. Поскольку бесконечное число членов ряда не вычислить численно, то ограничимся числом n членов ряда, где параметр n будет задаваться внутри функции. Результат работы функции graph6 будем выводить в графическое окно № 2, чтобы можно было сравнить его с результатами выполнения других функций.

Для вычисления числа членов ряда используется подфункция внутри функции ygr. Она должна быть определена раньше, чем вычисляется значение функции y(x).

Создайте функцию graph6, изменив функцию graph5. Вывод графиков осуществляйте в графическое окно № 2.

```
function [y1,y2]=graph6(x1,x2,c)
…
f1=scf(2); //Создание нового графического окна
           // и получение его дескриптора
```

Измените функцию ygr в graph6, добавив функцию se, и заменив в выражении для вычисления y(x) экспоненту на функцию se.

```
function s=se(x)
  //Подфункция вычисляет сумму ряда для каждого
элемента
  // вектора x
  s=1; //задание первого слагаемого в сумме
  n=20; //задание числа вычисляемых членов ряда
  for i=1:n //цикл
    s=s+x.^i/gamma(i+1)[13];
  end
endfunction //конец подфункции
y=(2)./(2+(8).*x+x.^2).*se(x); //Вычисление значений
                               // функции y(x)
```

3.4.2. Убедитесь, что функции graph5 и graph 6 дают схожий результат (при необходимости загрузите нужные функции в среду).

```
→graph6(5,2,1)
 ans  =    4.4302432
→graph5(5,2,1)
```

[13] Факториал в пакете Scilab можно вычислить через гамма-функцию $n!=\gamma(n+1)$.

```
 ans  =    4.4302436
```

3.4.3. Закройте оба графических окна.

3.5 Оператор цикла while

Если не известно, сколько раз нужно повторять тело цикла, то удобнее использовать оператор цикла с предусловием while[14]. Он имеет вид:

```
while условие
операторы
end
```

операторы будут выполняться циклически пока логическое условие истинно.

3.5.1. Теперь можно изменить функцию для вычисления суммы ряда так, чтобы вычислялось не конкретное число членов ряда, а такое число, чтобы последний член ряда был меньше заданного значения. Создайте новую функцию graph7 на основе функции graph6.

```
          function s=se(x)
            //Подфункция вычисляет сумму ряда для каждого
элемента
            // вектора x
            //Вычисления проводятся отдельно для каждого
элемента
            ep=0.01; //задание минимально допустимой
разницы
                     // между членами ряда
            s=ones(x)[15]; //задание первого слагаемого
                        // в сумме для всех элементов x
            for j=1:length(x) //индекс по числу элементов
                              // вектора x
              sp=0; //задание предыдущей суммы ряда
              i=0; //номер текущего члена ряда
              while abs(s(j)-sp)>ep //повторять, пока
разница двух
                             // последующих членов ряда
больше, чем ep
                sp=s(j); //Предыдущий член ряда
                i=i+1; //Индекс для следующего члена ряда
                s(j)=s(j)+x(j)^i/gamma(i+1);
              end
            end
```

[14] К сожалению, ни в пакете MATLAB, ни в пакете Scilab нет цикла с постусловием. Его можно реализовать, используя существующие операторы, но это удлиняет программу.

[15] Создает вектор s, имеющий такой же размер, как вектор x, и состоящий из единиц

```
endfunction
```

3.5.2. Проверьте, что функция graph7 дает результат, схожий с результатом, даваемым функцией graph5. Вычисления могут занять некоторое время на маломощных компьютерах.

```
→graph7(5,2,1)
 ans  =    4.4301555
```

3.5.3. Программу, реализованную в п. 3.5.1, можно написать, используя операторы управления циклом break (прерывает цикл) и continue. *Просмотрите справку по ним.*

3.5.4. ***Файл graph7 необходимо предъявить преподавателю для получения допуска к выполнению лабораторной работы.***

3.5.5. Закройте все простроенные графики.

4 Организация простейшего ввода-вывода

4.1 Вывод в командном окне

4.1.1. Для вывода переменных в командную строку используется функция disp. Можно выводить и строковые переменные. Строковые переменные заключаются в апострофы.

```
→disp('Hello')
 Hello
```

4.1.2. Если нужно вывести несколько переменных в одну строку, то можно сцепить строки, используя оператор **+.** В результате получится одна строка. Сравните два варианта вывода:

```
→st1='One ';
→st2='string';
→disp(st1,st2)
string
One
→disp(st1+st2)
 One string
```

4.1.3. Если выводимые переменные имеют числовой тип, то их можно преобразовать в строку функцией string. Если выводятся только числовые переменные, то их можно не преобразовывать в строку. Сравните:

```
→a1=15;
→disp(a1)
    15.
→disp('x='+string(a1))
 x=15
```

4.1.4. Теперь можно изменить функцию graph7, создав функцию graph8. Новая функция не будет иметь выходных параметров, а результат вычисления значений y1 и y2 будет выводиться в командное окно. В случае если аргумент отрицателен, будет выводиться соответствующее сообщение. Измените функцию следующим образом:

```
function graph8(x1,x2,c)
  //Функция для построения графика
y(x)=2/(2+8x+x^2)*exp(x) на отрезках [0,x1] и [0,x2]
  //и вычисления значений y1=y(x1), y2=y(x2), c -
параметр, определяющий палитру
…
  if x1>0 then
    subplot(211);
    y1=ygr(x1);
    disp('y1('+string(x1)+')='+string(y1)) //вывод
y1 в командное окно
  else //Если первый аргумент отрицательный
    disp('First parameter is not positive')[16]
  end
  if x2>0 then
    subplot(212);
    y2=ygr(x2);
    disp('y2('+string(x2)+')='+string(y2)) //вывод
y2 в командное окно
  else //Если второй аргумент отрицательный
    disp('Second parameter is not positive')
  end
```

4.1.5. Результат работы функции graph8 (не забудьте загрузить ее в среду).

```
→graph8(5,2,2)
 y1(5)=4.4301555
 y2(2)=0.6715729
→graph8(-5,2,2)
First parameter is not positive
y2(2)=0.6715729
→graph8(-5,-2,2)
First parameter is not positive
Second parameter is not positive
```

[16] В программе можно использовать символы кириллицы при выводе в командное окно. Однако попытка их ввода в командной строке может привести к «зависанию» программы, особенно в старых версиях.

4.2 Вывод в отдельном окне

4.2.1. Сообщение можно вывести не только в командном окне, но и в отдельном окне. Для этого используется функция messagebox.

Ее первый обязательный параметр определяет текст сообщения. Если это вектор, состоящий из строковых переменных, то каждая переменная выводится на новой строке. Остальные параметры не обязательны. Второй параметр определяет заголовок окна. Третий – тип иконки окна. Эти типы приведены в Табл. 1. Четвертый параметр определяет надписи на кнопках в окне, каждая кнопка определяется как отдельный элемент вектора, состоящего из строк. По умолчанию выводится одна кнопка OK. Функция возвращает номер нажатой кнопки.

Табл. 1 Параметр, определяющий тип иконки в окне сообщения

Значение параметра	Обычная область применения	Обычный вид иконки
'error'	Сообщение об ошибке	Белый крест на красном фоне
'info'	Информация к сведению	Буква i
'passwd'	Сообщение системы контроля доступа	Замок
'question'	Вопрос	Знак ?
'warning'	Предупреждение	Восклицательный знак в желтом треугольнике
'scilab'	Любые варианты	Иконка Scilab (параметр по умолчанию)

Сравните различные варианты однострочных сообщений (рис. 1.6 – 1.10). Не забывайте закрывать появляющиеся сообщения.

```
→messagebox('Hello');
```

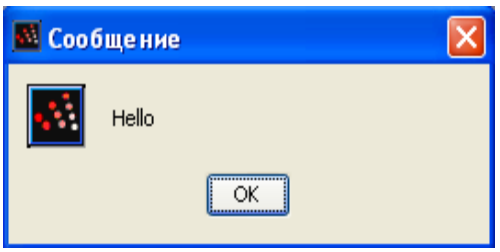


Рис 1.6 – Иконка Scilab. Вывод сообщения

```
→a2=5;
→messagebox('x='+string(a2)
, 'Results','info','Close');
```

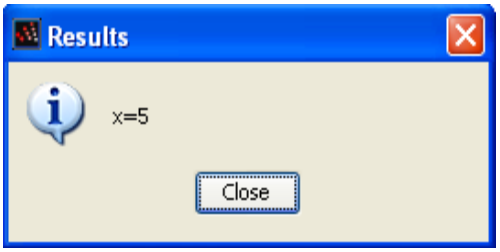


Рис. 1.7 – Информация к сведению

```
→messagebox('No
results','Error','error','Exit');
```

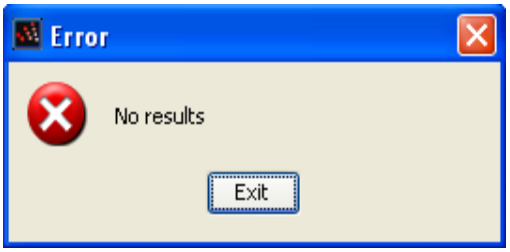


Рис. 1.8 – Сообщение об ошибке

4.2.2. Вывод сообщения в несколько строк

```
→messagebox(['Sum'
'x='+string(a2)],'Results');
```

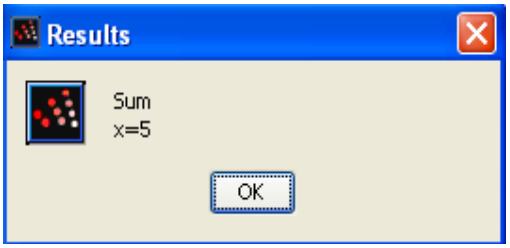


Рис. 1.9 – Вывод сообщения в несколько строк

4.2.3. Окно сообщения с несколькими кнопками. Обратите внимание на значение переменной ans после нажатия кнопок.

Параметр 'modal' необходим, чтобы выполнение следующих команд было остановлено до нажатия кнопки в окне.

```
messagebox('Division by
zero','Warning','warning',['Cance
l' 'OK'],'modal')
```

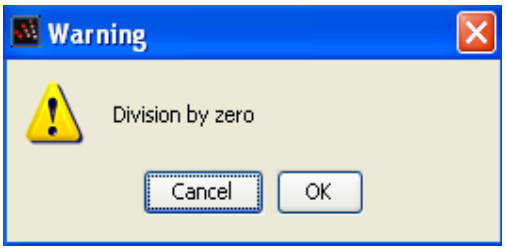


Рис. 1.10 – Предупреждение

Без параметра 'modal' функция всегда возвращает значение 0, выполнение других команд не прерывается.

4.2.4. Теперь можно изменить функцию graph8 так, чтобы все результаты выводились в отдельных окнах. Создайте функцию graph9, например, следующим образом:

```
if x1>0 then
  subplot(211);
  y1=ygr(x1);
  //вывод y1
```

```
messagebox('y1('+string(x1)+')='+string(y1),'Results')
       else //Если первый аргумент отрицательный
         messagebox(['First parameter is not positive'
'No graph'], 'Error in parameter 1', 'warning', 'Close');
       end
       if x2>0 then
         subplot(212);
         y2=ygr(x2);
         //вывод y2

messagebox('y2('+string(x2)+')='+string(y2),'Results')
       else //Если второй аргумент отрицательный
         messagebox(['Second parameter is not positive'
'No graph'], 'Error in parameter 2', 'warning', 'Close');
       end
```

4.2.5. *Проверьте работу новой функции*.

4.3 Ввод в командной строке

4.3.1. Для ввода значения переменной с клавиатуры из командной строки используется функция input. Она возвращает значение переменной. Строковые переменные при вводе нужно заключить в кавычки.

4.3.2. Наберите в командной строке:

```
→x=input('Enter x: ')
```

Появится подсказка:

```
Enter x:
```

Введите число 56

```
x  =    56.
```

4.3.3. Теперь можно изменить функцию, описанную в п. 4.3.2, так, чтобы ввод значений x1, x2 и c осуществлялся не в качестве параметров функции, а из командной строки. Функция не будет иметь входных параметров.

```
function graph10()
…
  x1=input('Enter x1: '); //Ввод x1
  x2=input('Enter x2: '); //Ввод x2
  c=input('Enter colormap number: '); //Ввод c
  f1=scf(2); //Создание нового графического окна и
получение его дескриптора
```

4.3.4. Пример работы функции graph10 представлен ниже (введите самостоятельно требующиеся значения). Обратите внимание, что при

отсутствии у функции параметров, при ее вызове все равно нужно указывать круглые скобки.

```
→graph10()
Enter x1: 5
Enter x2: 6
Enter colormap number: 2
```

4.3.5. В принципе, теперь функцию graph10 можно переделать в файл сценария, однако в этом случае его подфункция ygr станет доступна из командной строки, что не всегда нужно. *При желании переделайте функцию в файл сценария.*

4.4 Ввод в отдельном окне

4.4.1. Для организации ввода в отдельном окне может быть использована функция x_dialog. Ее первым параметром является подсказка, выводимая в окне (что надо ввести), а вторым параметром – значение по умолчанию. Эта функция возвращает строковое значение. Чтобы преобразовать строку в число, можно использовать функцию evstr. Например,

```
→a=x_dialog('Enter
a','2+2');
```

В появившемся окне (рис. 1.11) нажмите кнопку OK.

Значение переменной имеет строковый тип.

```
→a
 a   = 2+2
```

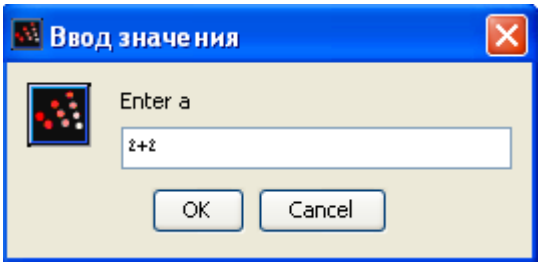


Рис. 1.11 – Ввод переменных в отдельном окне

Если теперь использовать преобразование в число, то выражение будет вычислено

```
→a2=evstr(a)
 a2   =     4.
```

4.4.2. Можно сразу использовать преобразование в число.

```
→s3=evstr(x_dialog('Enter summ','5+6+8*2'))
```

Если принять значение по умолчанию

```
 s3   =     27.
```

4.4.3. Теперь можно изменить функцию graph10 так, чтобы ввод осуществлялся из отдельных окон. Создайте на ее основе функцию или файл сценария egraph. **Эта функция (файл сценария) должна быть предъявлена преподавателю для получения допуска к лабораторной работе. Функция должна корректно работать.** Измените ввод переменных в функции на следующий:

```
x1=evstr(x_dialog('Enter x1','5'));
x2=evstr(x_dialog('Enter x2','2'));
c=evstr(x_dialog('Enter colormap number','0'));
```

4.4.4. *Проверьте работу функции egraph и предъявите ее преподавателю.* Может потребоваться демонстрация работоспособности функции.

5 Требования для получения допуска

Для получения допуска к выполнению лабораторной работы необходимо предъявить преподавателю файлы со следующим содержимым, требования к отдельным файлам описаны выше:

1. Графический файл с результатом построения графика функции sinexp, полученный в п. 2.4.
2. Графический файл с результатом построения графика функции $y = 2x$, полученный в п. 2.5.
3. Графический файл с результатом работы функции graph2, полученный в п. 2.6.
4. Файл graph7.sci, полученный в п. 3.5.
5. Файл сценария или функцию egraph, содержащую окончательную программу, полученную по результатам работы с методическими указаниями. Работа программы должна быть продемонстрирована.

Все файлы должны быть помещены в каталог с именем, имеющим формат: Д_ИВАНОВ_5212_3 или D_IVANOV_PETROV_5212_3. Здесь буква Д (D) указывает на то, что это допуск, далее указываются фамилии студентов (студента), выполнявших работу на одном компьютере, далее указывается номер группы и номер лабораторной работы.

Максимальное число студентов, одновременно участвующих в получении допуска – 2 человека.

6 Контрольное задание к лабораторной работе №3

6.1 Описание задания

Необходимо составить файл сценария или файл функции (далее в тексте - программа) для математического пакета Scilab, в котором будет производиться вычисление стационарного двумерного поля некоторой физической величины U для различных условий окружающей среды.

Поле физической величины U описывается функцией $U(x, y)$, $U(x, y) = g(x, y) + k$, где x и y – пространственные координаты (в декартовой системе

координат), k – константа, определяющая условия работы, $g(x,y)$ – функция, приведенная в таблице 2 в зависимости от варианта[17].

Поле физической величины необходимо вычислить для области Ω, имеющей форму прямоугольника, координаты вершин которого также приведены в таблице 2.

Вычисления должны проводиться в соответствии с исходными данными, заданными пользователем программы в интерактивном режиме, и исходными данными в зависимости от варианта задания.

В качестве исходных данных в программе должна быть задана функция $g(x,y)$ **в виде отдельной функции (подфункции)**, а также координаты прямоугольной области **в виде отдельных переменных или констант**.

Следующие данные, необходимые для расчета, должны вводиться пользователем программы в интерактивном режиме (не в виде параметров функции!):

- Константа k
- Координаты точки (x_0, y_0), в которой необходимо вычислить значение физической величины $U = U_0$ и вывести на экран результат.

Взаимодействие с пользователем может осуществляться как посредством командной строки, так и с использованием окон и графического интерфейса. Выбор способа не влияет на количество баллов, начисляемых за лабораторную работу. Интерфейс должен удовлетворять следующим условиям:

- При запуске файла сценария на экран должны выводиться фамилии авторов(а) работы, номер группы, номер варианта задания. Возможен также вывод исходных данных.
- Также при запуске должны быть выведены параметры, которые заданы по умолчанию, если пользователь не введет параметры k и (x_0, y_0). В качестве параметров, заданных по умолчанию, необходимо использовать значения $k = 0$, а для координаты точки – координаты центра прямоугольника, задающего область Ω.
- Далее должно быть выведено меню, в котором пользователю предлагается выбрать одно из указанных ниже действий:

1. ввести новые параметры взамен параметров, заданных ранее (k и (x_0, y_0)),
2. рассчитать значение U_0 в заданной точке (x_0, y_0),
3. вывести на экран график поля физической величины в заданной области,
4. завершить работу с программой.

- После выполнения действий 1-2 (и 3 по желанию) должен производиться возврат в меню с возможностью выбора всех перечисленных действий.

[17] Функция g используется в лабораторной работе №2. Ее можно скопировать в программу из выполненной лабораторной работы.

- Программа может иметь дополнительные функции, они не влияют на число начисленных баллов.

Также предъявляются следующие требования к графику, отображающему поле физической величины U в заданной области Ω:

- График должен быть контурным графиком с заливкой контуров цветом.
- В графике должна быть использована приемлемая цветовая палитра с плавными переходами между соседними цветами. Желательно отображение более высоких значений оттенками красного цвета, а более низких – оттенками синего и фиолетового. Возможно использование черно-белой палитры.
- По графику должно быть понятно, какие цвета соответствуют каким значениям, должны быть подписи осей.
- Для построения графика должно быть использовано достаточное число разбиений осей, чтобы не было значительных искажений поля физической величины (желательно более 100 на каждую ось).

К самой программе также предъявляются следующие требования:

- Программа должна корректно работать.
- **Все основные действия**, выполняемые в программе, должны иметь соответствующие комментарии. Первый комментарий должен содержать номер лабораторной работы, номер варианта и фамилию автора или авторов.
- При вводе исходных данных пользователем должна проверяться их адекватность. В случае если, например, введенные координаты точки выходят за границы области Ω, пользователю должно быть предложено повторно ввести эти координаты с сообщением о причине ошибки.

6.2 Форма отчета

Перед выполнением задания **необходимо получить допуск** по результатам выполнения заданий в методических указаниях.

В качестве отчета по лабораторной работе представляется исходный файл сценария (или файл функции) для математического пакета Scilab **в электронном виде.** Программа должна корректно работать и удовлетворять условиям, описанным в п. 2.

Файл должен быть помещен в отдельный каталог Название каталога должно состоять из фамилии студента (или фамилий студентов), номера группы и номера работы, например, `ПЕТРОВ_3460_3` или `PETROV_3460_3`.

При защите лабораторной работы могут быть заданы вопросы по любой строке программе и ее работе. Разрешается пользоваться только комментариями внутри файла. Также при защите лабораторной работы должна быть продемонстрирована работоспособность программы, в противном случае работа не засчитывается. Результаты расчетов, выдаваемые программой, должны совпасть с правильными ответами. В противном случае работа не засчитывается или начисляются штрафные баллы.

6.3 Указания по решению

При задании функции $g(x, y)$ следует обратить особое внимание на следующие моменты, которые обычно вызывают сложности у студентов:

1. Для удобства построения графика функцию необходимо организовать так, чтобы ее входные параметры могли быть векторами, содержащими значения набора координат x и y, в этом случае на выходе функции должен получаться вектор, содержащий значения функции для указанных наборов координат. Тогда контурный график можно строить способом, описанным в лабораторной работе № 2.

2. В альтернативном случае, когда входными параметрами функций не могут быть вектора, контурный график тоже можно построить. В этом случае массив Z координат точек необходимо получать вручную, используя цикл for для каждой координаты.

3. Выбор между способами 1 и 2 остается на усмотрение студента.

6.4 Критерии оценки и баллы

Далее везде указывается число баллов, начисляемое по курсу «Математические методы моделирования физических процессов», для групп, изучающих курс «Математическое моделирование физических процессов», баллы даны в скобках.

Максимальное число баллов, которое может быть получено за работу, составляет 7,5 (15), а минимальное число баллов 6 (12).

7,5 (15) баллов начисляется, если выполнены все условия сдачи программы. При этом предусмотрены следующие штрафы за погрешности в выполнении лабораторной работы:

- 0,2 (0,5) балла за отсутствие комментариев в тексте программы или их несоответствие тексту программы;
- 1 (2) балл, если в программе не строится график поля физической величины, а вычисляется только ее значение в заданной точке;
- 0,5 (1) балла, если в программе не вычисляется значение физической величины в заданной точке, а строится только ее график;
- 0,2 (0,5) балла за отсутствие вывода на экран исходных данных и комментариев;
- 0,5 (1) балла за отсутствие проверки корректности введенных значений координат;
- 0,2 (0,5) балла за интерфейс программы, в котором сложно разобраться;
- 0,5 (1) балл за задание функции $g(x)$ не в виде отдельной функции;
- 0,5 (1) балла за отсутствие задания координат области Ω в виде переменных или констант;
- 0,2 (0,5) балла за отсутствие подписей на графике;

- 0,2 (0,5) балла за неудачную цветовую палитру на графике и отсутствие легенды;

- 1 (2) балл за построение графика других типов взамен указанных в задании;

- 1,5 (3) балла, если результат расчетов не совпадает с правильным, и найденная ошибка связана с неверным заданием исходных данных.

Также штрафы начисляются за прочие погрешности в выполнении работы.

Следует также учитывать, что невыполнение отдельных требований к работе может **существенно осложнить процесс выполнения домашнего задания № 1**.

6.5 Варианты задания

Варианты задания приведены в таблице 2.

Таблица 2. Задание по вариантам: функция g(x,y) и координаты противоположных вершин области Ω

№ варианта	вершина 1		вершина 2		$g(x,y)$
	x	y	x	y	
1	-4	-4	4	4	$x^3 \sin y + 1$
2	-4	-4	4	4	$x^2 \sin y - 1$
3	-4	-4	4	4	$\left(1+(y+5)^2\right)^{-1} + x^2$
4	-4	-4	4	4	$\sqrt{\lvert y^2 + x^3 \rvert}$
5	-2	-5	-2	4	$\sqrt{\lvert y^3 + x^2 \rvert}$
6	-2	-5	6	6	$\cos y + xy$
7	-2	-5	6	6	$\left(1+(y+5)^2\right)^{-1} + xy$
8	0	0	1	π	$e^x + 3y$
9	0	0	1	π	$x^2 \cos y + 0.1$
10	-4	- π	4	π	$x^3 \cos y + 0.1$
11	0	0	2	π	$\cos(xy) - 0.5$
12	-4	-4	4	4	$e^{-y} + e^x - 2$
13	-4	-4	4	4	$e^{-y} - e^x - 0.1$
14	-1	-1	1	1	$e^{-xy} + 1$
15	-4	-4	4	4	$\sqrt{y^2 + x^4}$
16	0	-4	4	4	$y\sqrt{(x+1)}$
17	-4	-4	4	4	$e^y(x+1)$

№ варианта	вершина 1		вершина 2		$g(x,y)$
18	-4	-4	4	4	$\frac{x-3}{y+10}$
19	-4	-4	4	4	$2x+y^2$
20	-4	-4	4	4	$2x-y^3$
21	-4	-4	4	4	e^x-3y
22	-4	-4	4	4	$\cos y - xy$
23	-1	-1	1	1	$-e^{-xy}+1$
24	-4	-4	4	4	$(y-1)\sqrt{\lvert x+1\rvert}$
25	0	0	1	1	$\frac{2x-3}{y+10}$

ЛАБОРАТОРНАЯ РАБОТА №4. МОДЕЛИРОВАНИЕ НЕСТАЦИОНАРНОГО ПРОЦЕССА В МАТЕМАТИЧЕСКОМ ПАКЕТЕ SCILAB

1 Основные сведения о работе

1.1 Цель работы

Целью работы является получение навыков решения обыкновенных дифференциальных уравнений в математических пакетах и, в частности, в математическом пакете Scilab, и применение этих навыков для решения задач моделирования нестационарных физических процессов.

1.2 Порядок выполнения работы

В теоретической части методических указаний представлены сведения, необходимые для решения обыкновенных дифференциальных уравнений в математическом пакете Scilab.

В практической части методических указаний рассматриваются два примера решения обыкновенных дифференциальных уравнений в математическом пакете Scilab, которые необходимо выполнить:

- решение обыкновенного дифференциального уравнения первого порядка,
- решение дифференциального уравнения третьего порядка, сводящегося к системе уравнений первого порядка.

Для получения допуска к выполнению второй части работы должны быть предъявлены два графических файла, полученные в результате выполнения практических примеров.

Во второй части работы необходимо выполнить контрольное задание по вычислению значений физической величины в нестационарном процессе.

2 Теоретические сведения

2.1 Вектор-функции как способ записи систем уравнений

Математические модели задач технической физики могут быть представлены как в виде одного уравнения, так и в виде системы уравнений. Системы уравнений удобно представлять, используя **вектор-функции**. Значением вектор-функции является вектор, в то время как аргументами вектор-функции могут быть как скалярные переменные, так и вектора. Понятие вектор-функции широко используется и в математическом программном обеспечении. Суть этого понятия раскрывается следующим образом.

Обычно функция f представляется в виде

$$u = f(x), \qquad (2.1.1)$$

где u – значение функции, x – аргумент функции. Аргумент функции x может быть скалярной величиной, а может быть вектором $\vec{x}$. Вектор принято записывать в виде

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_d \end{pmatrix}, \tag{2.1.2}$$

где $x_1, x_2 \dots, x_d$ – координаты вектора. Обычно полагают, что зависимость функции от вектора $\vec{x}$ равнозначна ее зависимости от d скалярных переменных $x_1, x_2 \dots, x_d$. Т. е. рассматривается функция нескольких аргументов:

$$u = f(x_1, x_2, \dots, x_d). \tag{2.1.3}$$

Если f – **вектор-функция,** то ее значение $\vec{u}$ представляется в виде вектора:

$$\vec{u} = \begin{pmatrix} u_1 \\ u_2 \\ \dots \\ u_m \end{pmatrix}. \tag{2.1.4}$$

В этом случае уравнение (2.1.1) равносильно следующей системе:

$$\begin{cases} u_1 = f_1(x_1, x_2, \dots, x_d) \\ u_2 = f_2(x_1, x_2, \dots, x_d) \\ \dots \\ u_m = f_m(x_1, x_2, \dots, x_d) \end{cases}, \tag{2.1.5}$$

т. е. саму функцию f можно представить в виде вектора, состоящего из m строк:

$$f = \begin{pmatrix} f_1(x_1, x_2, \dots, x_d) \\ f_2(x_1, x_2, \dots, x_d) \\ \dots \\ f_m(x_1, x_2, \dots, x_d) \end{pmatrix}. \tag{2.1.6}$$

Форма представления (2.1.5) часто используется для записи систем уравнений. Для математического программного обеспечения эта форма удобна

тем, что позволяет одним и тем же способом (2.1.1) описывать как одиночные уравнения, так и системы уравнений.

2.2 Дифференциальные уравнения: основные понятия

Дифференциальными называют уравнения, в которых неизвестная функция или вектор-функция входит под знаком производной или дифференциала.

Очевидно, что если неизвестная функция является вектор-функцией, то ее можно представить в виде (2.1.5) и, таким образом, перейти к системе дифференциальных уравнений. Или наоборот, систему уравнений можно представить в виде одного уравнения, в которое входит вектор-функция.

Дифференциальное уравнение называется **линейным**, если оно представляется в виде линейной комбинации производных неизвестной функции.

Решением дифференциального уравнения называется функция, которая при подстановке в дифференциальное уравнение обращает его в тождество для всех значений аргументов функции, принадлежащих **области определения** дифференциального уравнения. Т. е. решение дифференциального уравнения находится только для некоторого множества Ω значений аргументов функций, называемого областью определения.

В общем случае дифференциальное уравнение может иметь бесконечное множество решений. Процесс нахождения решений дифференциального уравнения называется **интегрированием дифференциального уравнения**.

Порядком дифференциального уравнения называется максимальный порядок входящей в уравнение производной (или дифференциала) неизвестной функции.

Дифференциальное уравнение называется **обыкновенным**, если входящие в него неизвестные функции или вектор-функции являются функциями одной переменной. Если неизвестная функция, входящая в дифференциальное уравнение, является функцией двух или большего числа переменных, то дифференциальное уравнение называется **уравнением в частных производных**.

Как правило, дифференциальное уравнение имеет бесконечное число решений, если не заданы условия однозначности. Поскольку в задачах технической физики необходимо найти значение физической величины, соответствующее рассматриваемому физическому процессу, то задание условий однозначности в математической модели является обязательным.

Условия однозначности делятся на два вида: **начальные условия** и **граничные условия**. Граничные условия задаются на границе области определения $\Gamma = \partial\Omega$. Граничные условия в задачах технической физики обычно встречаются, когда независимые переменные в дифференциальном

уравнении являются пространственными координатами. Граничные условия бывают нескольких видов.

Начальные условия задаются в начальной точке области определения $x_i = x_0$. В этом случае должно быть задано значение неизвестной функции $u = u(x_0)$ и всех ее производных до порядка n включительно, где n – порядок дифференциального уравнения. Как правило, начальные условия задаются в тех случаях, когда независимой переменной в дифференциальном уравнении является время. Но следует отметить, что начальные условия в некоторых случаях могут быть заданы и для пространственных координат. В отдельных случаях начальные условия рассматривают как особый вид граничных условий.

Задачу нахождения решения дифференциального уравнения с начальными условиями называют **задачей Коши**.

В уравнения в частных производных в качестве независимых переменных могут одновременно входить пространственные координаты и время. В этих случаях требуется задавать одновременно как начальные условия, так и граничные условия, которые в совокупности называются **краевыми условиями**.

В технической физике, как правило, необходимо найти функцию, значения которой зависят от пространственных координат[18] x, y, z и времени τ. Такой функцией может быть, например, температура, электрический потенциал, скорость, деформация, давление и пр.

Задача называется **одномерной**, если искомая функция зависит только от одной пространственной координаты (например, от x), если функция зависит только от двух пространственных координат (например, x и y), то задача называется **двумерной**, если от трех координат, то задача называется **трехмерной**.

Если функция не зависит от времени, то задача называется **статической** (стационарной, установившейся), иначе задача называется **переходной** (нестационарной, неустановившейся). В свою очередь, каждая их этих задач может быть одномерной, двумерной или трехмерной.

Следует обратить особое внимание, что если рассматривается переходный процесс, то в одномерной задаче функция зависит не от одной, а от двух переменных, и, следовательно, дифференциальное уравнение является уравнением в частных производных.

Численные алгоритмы и методы, используемые для решения дифференциальных уравнений, различаются в зависимости от того, рассматривается ли обыкновенное дифференциальное уравнение или дифференциальное уравнение в частных производных. Значение имеет также, является ли уравнение линейным или нет. Также значение имеет и то, рассматривается ли задача Коши или задача с граничными условиями. В некоторых случаях определяющим является и вид граничных условий.

[18] Координаты могут быть выражены не только в декартовой системе координат.

2.3 Основные численные методы решения

Используемые в тех или иных программных продуктах численные методы решения дифференциальных уравнений различаются для уравнений различных типов. Как правило, методы, предназначенные для решения дифференциальных уравнений в частных производных, применимы и для решения обыкновенных дифференциальных уравнений. Однако численная реализация таких методов намного сложнее, чем методов, предназначенных только для решения обыкновенных дифференциальных уравнений.

Численные методы, применяемые для решения обыкновенных дифференциальных уравнений, выбираются в зависимости от того, рассматривается ли задача Коши или задача с граничными условиями. Решение задачи с граничными условиями в целом сложнее. Подобная задача либо сводится к задаче Коши, либо решается теми же методами, что и задача, включающая дифференциальные уравнения в частных производных.

Для решения нестационарных одномерных, двумерных или трехмерных задач, которые, очевидно, описываются дифференциальными уравнениями в частных производных, обычно используется комбинация численных методов. Комбинируются методы, предназначенные для решения уравнений в частных производных и задачи Коши для обыкновенных дифференциальных уравнений. На первом этапе решения задача сводится к решению системы обыкновенных дифференциальных уравнений, независимой переменной в которой является время. Для получения этой системы используются численные методы решения уравнений в частных производных, применяемые относительно пространственных координат.

Наиболее часто в программном обеспечении для решения задачи Коши для обыкновенного дифференциального уравнения или системы уравнений используется метод Рунге-Кутты 4-го порядка. Иногда пользователю предоставляется возможность самостоятельного выбора численного метода. Особенности численных методов можно найти в соответствующей литературе [2]. Метод Рунге-Кутты реализован и в математическом пакете Scilab. Также в пакете Scilab реализованы и другие численные методы, например, метод Адамса.

Для сведения граничной задачи к задаче Коши обычно используется метод стрельбы [2].

Для решения обыкновенных дифференциальных уравнений с граничными условиями в математическом программном обеспечении часто используется метод конечных разностей. Однако для решения таких задач может применяться и метод конечных элементов.

Оба рассмотренных метода, метод конечных разностей (иногда называемый методом сеток) и метод конечных элементов, применяются для решения дифференциальных уравнений в частных производных. Более простой в реализации метод конечных разностей накладывает заметные ограничения на

форму области определения дифференциального уравнения. Границы области в этом случае должны быть параллельны координатным плоскостям.

При решении дифференциальных уравнений в частных производных для области определения сложной или произвольной формы обычно используют метод конечных элементов (МКЭ). Решению подобных задач посвящена лабораторная работа № 5 и домашнее задание № 3.

В данной лабораторной работе рассматривается только решение задачи Коши для системы обыкновенных дифференциальных уравнений.

2.4 Вид уравнения для численного решения в математическом пакете Scilab

В пакете Scilab, как и в большинстве других подобных программных продуктов, численное решение находится для обыкновенного дифференциального уравнения, представленного в виде

$$\frac{du}{d\tau} = f(x,u), \tag{2.4.1}$$

т. е. первая производная должна быть выражена явно. В (2.4.1) u – неизвестная (искомая) функция, зависящая от τ, f– произвольная функция, зависящая от u и τ, и не зависящая от производных функции u. При невозможности явного выражения первой производной, что бывает редко, необходимо искать другие методы решения.

Начальное условие для уравнения (2.4.1), имеет вид

$$u(\tau = \tau_0) = u_0, \tag{2.4.2}$$

где u_0 – известное число. Т. е. рассматривается случай, когда решается задача Коши.

Аналогично система N дифференциальных уравнений должна быть сведена к виду

$$\begin{cases} \frac{du_1}{d\tau} = f_1(\tau, u_1, u_2, ..., u_N) \\ \frac{du_2}{d\tau} = f_2(\tau, u_1, u_2, ..., u_N) \\ ... \\ \frac{du_N}{d\tau} = f_n(\tau, u_1, u_2, ..., u_N) \end{cases}. \tag{2.4.3}$$

Начальные условия для системы (2.4.3) :

$$\begin{cases} u_1(\tau_0) = u_{10} \\ u_2(\tau_0) = u_{20} \\ \dots \\ u_N(\tau_0) = u_{N0} \end{cases}. \qquad (2.4.4)$$

Используя понятие вектор-функции, можно представить систему (2.4.3) следующим образом:

$$\frac{d\vec{u}}{d\tau} = F(\tau, \vec{u}), \qquad (2.4.5)$$

где искомая вектор-функция $\vec{u}$:

$$\vec{u} = \begin{pmatrix} u_1 \\ u_2 \\ \dots \\ u_N \end{pmatrix}, \qquad (2.4.6)$$

а

$$F(\tau, \vec{u}) = \begin{pmatrix} f_1(\tau, \vec{u}) \\ f_2(\tau, \vec{u}) \\ \dots \\ f_N(\tau, \vec{u}) \end{pmatrix}. \qquad (2.4.7)$$

В этом случае начальное условие также представляется в виде вектора:

$$\vec{u}(\tau_0) = \vec{u}_0, \text{ где } \vec{u}_0 = \begin{pmatrix} u_{10} \\ u_{20} \\ \dots \\ u_{N0} \end{pmatrix}. \qquad (2.4.8)$$

Выражения (2.4.5) и (2.4.8) описывают также уравнение (2.4.1) с начальным условием (2.4.2), если $N = 1$. Таким образом, случаи одиночного уравнения и системы уравнений далее могут рассматриваться вместе.

Если необходимо решить уравнение порядка n, большего, чем первый, его необходимо свести к системе уравнений первого порядка. Это делается известным из математики способом, а именно, путем введения $n - 1$ новых переменных (функций).

$$k_{i+1} = \frac{d^i u}{d\tau^i}, \text{ где } i = 1,2,...,(n-1). \quad (2.4.9)$$

Для удобства обозначения полагают

$$k_1 = u. \quad (2.4.10)$$

Первые $n-1$ уравнений новой системы получают из (2.4.9) и (2.4.10). Они имеют вид

$$\frac{dk_i}{d\tau} = k_{i+1}, \text{ где } i = 1,2,...,(n-1). \quad (2.4.11)$$

Последнее уравнение системы получается путем подстановки в исходное дифференциальное уравнение вместо производных функции u и самой функции u соответствующих им функций k. В результате получается система n дифференциальных уравнений первого порядка. Начальные условия, заданные для производных, преобразуются в начальные условия для n функций. Если действовать по указанной схеме, то сам вектор, задающий числовые значения начальных условий, останется неизменным.

Аналогичным образом можно свести к системе обыкновенных дифференциальных уравнений первого порядка систему обыкновенных дифференциальных уравнений порядка большего, чем первый.

2.5 Функции математического пакета Scilab для решения обыкновенных дифференциальных уравнений

В пакете Scilab, в принципе, можно решать не только задачу Коши, но и задачу с граничными условиями. Для решения задачи с граничными условиями существует функция bvode[19], однако в данной лабораторной работе она не рассматривается. В работе рассматривается только решение задачи Коши.

Для решения обыкновенных дифференциальных уравнений или систем обыкновенных дифференциальных уравнений первого порядка в пакете Scilab используется функция ode. В простейшем случае она имеет следующий синтаксис:

```
u=ode(u0,x0,x,F);
```

Здесь

- u0 – вектор начальных условий, соответствующий (2.4.8).

[19] В пакете MATLAB решение обыкновенного дифференциального уравнения с граничными условиями можно найти при помощи функции bvp4c, реализующей метод конечных разностей. Ее аналога в пакете Scilab нет. Подробнее о работе функции bvp4c можно прочитать в [3].

• x0 – значение аргумента функции, для которого заданы начальные условия, т. е. в обозначениях (2.4.8) $x0 = \tau_0$.

• x – вектор-столбец, содержащий координаты узлов сетки, в которых должны быть вычислены значения искомых функций. Выбор этих значений влияет на погрешность расчета.

• u – имя переменной, в которую записываются вычисленные значения искомой функции в узлах. u представляет собой матрицу, число строк которой равно числу неизвестных в системе уравнений, а число столбцов равно числу столбцов вектора x. Каждая строка i матрицы содержит значения искомой функции u_i, вычисленные в узлах x_j, заданных в векторе x.

• F – имя вектор-функции среды Scilab, которая задает правую часть системы, состоящей из N уравнений, в соответствии с (2.4.7). Эта функция должна иметь следующий вид:[20]

```
function du=F(x,u);
...
du(1)=...;
du(2)=...;
...
du(n)=...;
endfunction
```

В качестве первого параметра функции ode можно указать необязательный параметр, который определяет метод решения системы уравнений. Если этот параметр опущен, то выбор метода будет производиться автоматически, что может быть не всегда корректно. Некоторые значения указанного параметра и соответствующие ему методы численного решения указаны ниже:

• 'adams' – метод Адамса;

• 'stiff' – используется для решения жестких систем;

• 'rk' – адаптивный метод Рунге-Кутта четвертого порядка.

Функция ode имеет и другие дополнительные входные и выходные параметры, их описание можно найти в [4].

3 Примеры решения обыкновенных дифференциальных уравнений в математическом пакете Scilab

3.1 Дифференциальное уравнение первого порядка

Описание задачи

Необходимо численно решить следующее нелинейное дифференциальное уравнение на интервале [0; 1]:

[20] Функции подобного вида задаются и в других программах, например, в пакете MathCAD.

$$\frac{dy}{dx} + x^2 - \sin(x + y) = 0, \tag{3.1.1}$$

с начальным условием:

$$y(x = 0) = 0. \tag{3.1.2}$$

Исходные данные для численного расчета

Вначале необходимо преобразовать уравнение к виду (2.4.1). В данном случае очевидно, что

$$\frac{dy}{dx} = \sin(x + y) - x^2. \tag{3.1.3}$$

Для численного расчета необходимы следующие данные, они задаются в качестве аргументов функции ode:

- Начальное условие, $y(x_0) = 0$;
- Точка, в которой вычисляется начальное условие, $x_0 = 0$;
- Вектор узлов сетки x. В данном случае зададим его как вектор, состоящий из $N = 51$ точек, равномерно распределенных на интервале $[0; 1]$ области определения уравнения, включая его граничные точки;
- Функция, задающая правую часть уравнения, $f(x, y) = \sin(x + y) - x^2$.

Решение уравнения

3.1.1. *Войдите в среду Scilab.*

3.1.2 Вначале задайте функцию f, определяющую правую часть уравнения (3.1.3). Ее можно задать в файле (не забудьте загрузить функцию в среду) или из командной строки. Обратите внимание, что при вычислении значений функции должны использоваться поэлементные операции, так как ее входными параметрами могут быть массивы значений узлов.

```
→function dy=f(x,y)
→dy=sin(x+y)-x.^2;
→endfunction
```

3.1.3 Задайте начальное условие и точку, в которой оно вычисляется. Конечно, эти условия можно задать сразу в параметрах функции ode, но приведенный ниже вариант делает программу более понятной и упрощает ее изменение в дальнейшем.

```
→y0=0;
→x0=0;
```

3.1.4. Задайте значения вектора, содержащего узлы сетки, в которых проводятся вычисления.

```
→N=50;
→x=0:(1/N):1;
```

3.1.5. Теперь найдите решение заданной системы уравнений методом Рунге-Кутта четвертого порядка.

```
→y=ode("rk",y0,x0,x,f);
```

Значения искомой функции, вычисленные в узлах, заданных вектором x, были помещены в вектор y.

3.1.6. Постройте график найденной функции, он приведен на рисунке 2.1.

```
→plot2d(x,y,style=-2)
```

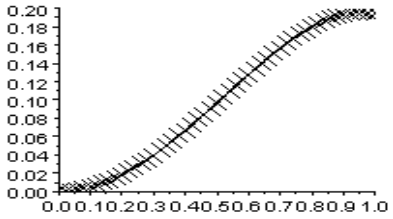


Рис. 2.1 - Результирующий график

3.1.7. ***Сохраните полученный график в графическом файле exl4_1 и предъявите его преподавателю для получения допуска к выполнению лабораторной работы***.

3.1.8. Выведите значения функции, найденной в результате решения уравнения, на границах рассматриваемого интервала.

```
→y(1) //Значение функции при x=0
 ans = 0.
→y(N+1) //Значение функции при x=1
 ans = 0.1943212
```

3.1.9. *Закройте построенный график и удалите все неиспользуемые переменные.*

3.2 Уравнение третьего порядка и система уравнений

Описание задачи

Необходимо численно решить следующее нелинейное дифференциальное уравнение на интервале [2; 5]:

$$\frac{d^3y}{dx^3} = 2y^{-5/2} + x^2 . \qquad (3.2.1)$$

Начальное условие к этому уравнению имеет вид

$$y(x=2)=1, \quad \left.\frac{dy}{dx}\right|_{x=2} = 3, \quad \left.\frac{d^2y}{dx^2}\right|_{x=2} = 0. \qquad (3.2.2)$$

Исходные данные для численного расчета

Уравнение (3.2.1) является уравнением третьего порядка, поэтому для численных расчетов его необходимо преобразовать в систему уравнений первого порядка. Вначале производится замена переменных в соответствии с (2.4.9) и (2.4.10):

$$y1 = y, \quad y2 = \frac{dy}{dx}, \quad y3 = \frac{d^2y}{dx^2} . \qquad (3.2.3)$$

Далее, в соответствии с (2.4.11), а также после подстановки (3.2.3) в исходное уравнение (3.2.1) получается:

$$\begin{cases} \dfrac{dy1}{dx} = y2 \\ \dfrac{dy2}{dx} = y3 \\ \dfrac{dy3}{dx} = 2y1^{-5/2} + x^2 \end{cases}. \tag{3.2.4}$$

Начальные условия также требуют преобразования, в (3.2.2) подставляются новые переменные (3.2.3). В результате получаются начальные условия для системы уравнений:

$$\begin{cases} y1(x = 2) = 1 \\ y2(x = 2) = 3 \\ y3(x = 2) = 0 \end{cases}. \tag{3.2.5}$$

Функции $y1, y2$ и $y3$ являются неизвестным в системе уравнений (3.2.4) с начальными условиями (3.2.5). Причем функция $y1$ и есть искомая функция y, а $y2$ и $y3$ – ее первая и вторая производные соответственно. Все эти функции можно представить как одну вектор-функцию Y:

$$Y = \begin{pmatrix} y1 \\ y2 \\ y3 \end{pmatrix}. \tag{3.2.6}$$

Для численного расчета необходимы следующие данные, которые задаются в качестве аргументов функции ode:

- Вектор начальных условий, который определяется из (3.2.5);

$$Y0 = \begin{pmatrix} 1 \\ 3 \\ 0 \end{pmatrix}; \tag{3.2.7}$$

- Точка, в которой вычисляется начальное условие, $x_o = 2$;
- Вектор узлов сетки x. В данном случае зададим его как вектор, состоящий из $N = 61$ точек, равномерно распределенных на интервале $[2; 5]$ области определения уравнения;
- Вектор-функция, задающая правую часть системы уравнений (3.2.4),

$$F = \begin{pmatrix} y2 \\ y3 \\ 2y1^{-5/2} + x^2 \end{pmatrix}. \qquad (3.2.8)$$

Используя запись (3.2.6), последнее выражение можно представить в виде:

$$F = \begin{pmatrix} Y_2 \\ Y_3 \\ 2Y_1^{-5/2} + x^2 \end{pmatrix}. \qquad (3.2.9)$$

Решение системы уравнений

3.2.1. Вначале задайте вектор начальных условий Y0 в соответствии с (3.2.7). Ввод можно осуществлять в командной строке или в файле сценария.

```
→Y0=[1; 3; 0];
```

3.2.2. Задайте начальное и конечное значение интервала, на котором определена искомая функция (область определения).

```
→x0=2; //Начальное значение интервала
→xk=5; //Конечное значение интервала
```

3.2.3. Задайте вектор узловых значений x.

```
→N=60;
→x=x0:(xk-x0)/N:xk;
```

3.2.4. Задайте вектор-функцию F в соответствии с (3.2.9). Она зависит от переменной x и вектора Y, соответствующего (3.2.6).

```
→function dY=F(x,Y)
→dY(1)=Y(2);
→dY(2)=Y(3);
→dY(3)=2*Y(1).^(-5/2)+x.^2;
→endfunction
```

3.2.5. Решите систему уравнений, используя метод, выбранный по умолчанию.

```
→Y=ode(Y0,x0,x,F);
```

3.2.6. Выведите на экран значения искомой функции и ее первой и второй производной в начальной точке интервала. Очевидно, что они должны совпадать с начальными условиями.

```
→Y(1,1) //Значение функции
 ans = 1.
→Y(2,1) //Значение первой производной
 ans = 3.
```

```
→Y(3,1) //Значение второй производной
 ans = 0.
```

3.2.7. Выведите на экран значения искомой функции и ее первой и второй производной в конечной точке интервала.

```
→Y(1,N+1)
 ans = 47.01117
→Y(2,N+1)
 ans = 46.802175
→Y(3,N+1)
 ans = 39.383196
```

3.2.8. Выведите значения искомой функции в виде графика. Они сохранены в первой строке матрицы Y. Результат приведен на рисунке 2.2.

```
→plot2d(x,Y(1,:))
```

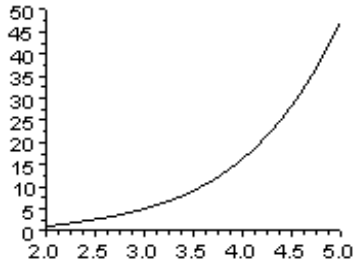


Рис. 2.2 – График искомой функции

3.2.9. ***Сохраните полученный график в графическом файле exl4_2 и предъявите его преподавателю для получения допуска к выполнению лабораторной работы.***

4 Требования для получения допуска

Для получения допуска к выполнению лабораторной работы необходимо предъявить преподавателю файлы со следующим содержимым:

1. Графический файл exl4_1 с результатом построения графика функции, являющейся решением дифференциального уравнения в п. 3.1.
2. Графический файл exl4_2 с результатом построения графика функции, являющейся решением дифференциального уравнения в п. 3.2.

Все файлы должны быть помещены в каталог с именем.

5 Контрольное задание к лабораторной работе №4

5.1 Описание задания

Необходимо составить файл сценария или файл функции (далее в тексте называется как программа) для математического пакета Scilab. В программе должно производиться вычисление некоторой физической величины U в зависимости от времени для системы, состоящей из двух связанных между собой тел. Полагается, что поле физической величины U в каждом теле равномерное, то есть не зависит от координат. Вычисления должны проводиться в соответствии с исходными данными, заданными пользователем в интерактивном режиме, и исходными данными в зависимости от варианта задания.

Математическая модель, описывающая рассматриваемый процесс, представляет собой систему двух нелинейных обыкновенных дифференциальных уравнений, неизвестными в которой являются значения величины U для двух тел ($u1$ и $u2$ для тела 1 и тела 2 соответственно) в зависимости от времени τ. Математическая модель приведена в таблице 3 в зависимости от варианта. Начальные условия в момент времени $\tau = 0$ задаются пользователем программы. Также пользователем задается конечный момент времени, до которого необходимо проводить расчеты.

Взаимодействие с пользователем может осуществляться как посредством командной строки, так и с использованием окон и графического интерфейса. Выбор способа не влияет на количество баллов, начисляемых за лабораторную работу. Интерфейс должен удовлетворять следующим условиям:

- При запуске программы на экран должны выводиться фамилии авторов работы, номер группы, номер варианта. Возможен также вывод исходных данных.
- После этого должен выводиться запрос на ввод начальных условий и конечного момента времени, до которого должны производиться расчеты. Значения начальных условий и конечного момента времени, принимаемых по умолчанию, приведены в таблице 3.
- Далее должны быть вычислены и выведены на экран результаты расчетов величины U для каждого тела в конечный момент времени.
- Должен строиться график, показывающий на одних осях изменение величины U каждого тела с начального момента времени до указанного времени.
- Программа может иметь дополнительные функции, они, кроме специально оговоренных случаев, не влияют на число начисленных баллов.

Также предъявляются следующие требования к графику, отображающему изменение физической величины U:

- Графики для всех тел системы должны быть построены на одних осях.
- Разные графики для разных тел должны отображаться различным цветом.
- По графику должно быть понятно, какая кривая соответствует какому телу, график должен иметь подписи осей.
- Для построения графика должно быть использовано достаточное число точек, чтобы не было его заметных искажений.

К самой программе также предъявляются следующие требования:

- Все действия, выполняемые в программе, должны иметь соответствующие комментарии. Первый комментарий должен содержать номер лабораторной работы, номер варианта и фамилию автора или авторов.
- При вводе исходных данных пользователем должна проверяться их адекватность. В случае если, например, время является отрицательным,

пользователю должно быть предложено повторно ввести его с сообщением о причине ошибки.

5.2 Форма отчета

В качестве отчета по лабораторной работе представляется исходный файл сценария (файл функции) для математического пакета Scilab и файл графического формата с результатами построения графика для времени и начальных условий, заданных по умолчанию. Результаты построения можно сохранять в файле графического формата вручную. Все файлы представляются **в электронном виде.** Файл сценария (функция) должен корректно работать и удовлетворять условиям, описанным в п. 2.

Все файлы должны быть помещены в отдельный каталог.

При защите лабораторной работы могут быть заданы вопросы по любой строке файла сценария и его работе. Разрешается пользоваться комментариями в файле сценария. Также при защите лабораторной работы должна быть продемонстрирована работоспособность файла сценария, в противном случае работа не засчитывается. Результаты расчетов, выдаваемые программой, должны совпасть с правильными ответами. В противном случае работа не засчитывается или начисляются штрафные баллы.

5.3 Критерии оценки и баллы

Следующие погрешности в выполнении работы приводят к снижению оценки за нее:

- отсутствие комментариев в тексте программы или их несоответствие тексту программы;
- если в программе не строятся графики изменения физической величины, а вычисляются только ее значения в указанный момент времени;
- если в программе не вычисляются значения физической величины в указанный момент времени;
- отсутствие вывода на экран комментариев, результатов;
- отсутствие проверки корректности введенного значения времени;
- интерфейс программы, в котором сложно разобраться;
- отсутствие подписей на графике;
- неудачную цветовую палитру на графике и отсутствие легенды;
- построение графика других типов, взамен указанных в задании;
- недостаточное число точек на графике;
- результат расчетов не совпадает с правильным, и найденная ошибка связана с неверным заданием исходных данных.

5.4 Варианты задания

Варианты задания приведены в таблице 3.

Таблица 3 Варианты заданий

№ вар.	Система уравнений	Начальные условия по умолчанию		Конечный момент времени по умолчанию
		$u1$	$u2$	
1	$\begin{cases} 7 = 500\frac{du_1}{d\tau} + 0.045(u_1 - u_2) + 0.000081(u_1^3 - u_2^3) \\ 1 = 2830\frac{du_2}{d\tau} + 0.045(u_2 - u_1) + 0.000081(u_2^3 - u_1^3) \end{cases}$	20	120	500
2	$\begin{cases} 14 = 6700\frac{du_1}{d\tau} + 0.0195(u_1 - u_2) + 0.000081(u_1^2 - u_2^2) \\ 2.2 = 2860\frac{du_2}{d\tau} + 0.0195(u_2 - u_1) + 0.000081(u_2^2 - u_1^2) \end{cases}$	30	10	1500
3	$\begin{cases} 12 = 2000\frac{du_1}{d\tau} + 0.0285(u_1 - u_2) + 0.000026(u_1^{1.5} - u_2^{1.5}) \\ 0.3 = 4080\frac{du_2}{d\tau} + 0.0285(u_2 - u_1) + 0.000026(u_2^{1.5} - u_1^{1.5}) \end{cases}$	30	140	1300
4	$\begin{cases} 8 = 3900\frac{du_1}{d\tau} + 0.0355(u_1 - u_2) + 0.000041(u_1^3 - u_2^3) \\ 0.5 = 280\frac{du_2}{d\tau} + 0.0355(u_2 - u_1) + 0.000041(u_2^3 - u_1^3) \end{cases}$	30	70	2200
5	$\begin{cases} 15 = 8300\frac{du_1}{d\tau} + 0.034(u_1 - u_2) + 0.000036(u_1^2 - u_2^2) \\ 1.4 = 2800\frac{du_2}{d\tau} + 0.034(u_2 - u_1) + 0.000036(u_2^2 - u_1^2) \end{cases}$	50	45	1500
6	$\begin{cases} 13 = 5900\frac{du_1}{d\tau} + 0.0205(u_1 - u_2) + 0.000069(u_1^4 - u_2^4) \\ 1.5 = 620\frac{du_2}{d\tau} + 0.0205(u_2 - u_1) + 0.000069(u_2^4 - u_1^4) \end{cases}$	20	0	2300
7	$\begin{cases} 11 = 4800\frac{du_1}{d\tau} + 0.007(u_1 - u_2) + 0.000077(u_1^3 - u_2^3) \\ 1.7 = 3640\frac{du_2}{d\tau} + 0.007(u_2 - u_1) + 0.000077(u_2^3 - u_1^3) \end{cases}$	70	135	1400
8	$\begin{cases} 7 = 2200\frac{du_1}{d\tau} + 0.025(u_1 - u_2) + 0.000036(u_1^2 - u_2^2) \\ 2.2 = 1340\frac{du_2}{d\tau} + 0.025(u_2 - u_1) + 0.000036(u_2^2 - u_1^2) \end{cases}$	60	125	1700
9	$\begin{cases} 16 = 8400\frac{du_1}{d\tau} + 0.021(u_1 - u_2) + 0.000077(u_1^4 - u_2^4) \\ 1.3 = 2730\frac{du_2}{d\tau} + 0.021(u_2 - u_1) + 0.000077(u_2^4 - u_1^4) \end{cases}$	40	65	500

№ вар.	Система уравнений	Начальные условия по умолчанию		Конечный момент времени по умолчанию
		$u1$	$u2$	
10	$\begin{cases} 10 = 1200\frac{du_1}{d\tau} + 0.043(u_1 - u_2) + 0.000055\left(u_1^{1.5} - u_2^{1.5}\right) \\ 0.9 = 4940\frac{du_2}{d\tau} + 0.043(u_2 - u_1) + 0.000055\left(u_2^{1.5} - u_1^{1.5}\right) \end{cases}$	40	110	2000
11	$\begin{cases} 23 = 2900\frac{du_1}{d\tau} + 0.0145(u_1 - u_2) + 0.00001\left(u_1^3 - u_2^3\right) \\ 1 = 3700\frac{du_2}{d\tau} + 0.0145(u_2 - u_1) + 0.00001\left(u_2^3 - u_1^3\right) \end{cases}$	30	20	1600
12	$\begin{cases} 1 = 86000\frac{du_1}{d\tau} + 0.0125(u_1 - u_2) + 0.000006\left(u_1^4 - u_2^4\right) \\ 2.3 = 2000\frac{du_2}{d\tau} + 0.0125(u_2 - u_1) + 0.000006\left(u_2^4 - u_1^4\right) \end{cases}$	60	20	1300
13	$\begin{cases} 12 = 85000\frac{du_1}{d\tau} + 0.017(u_1 - u_2) + 0.000002\left(u_1^5 - u_2^5\right) \\ 2.4 = 6000\frac{du_2}{d\tau} + 0.017(u_2 - u_1) + 0.000002\left(u_2^5 - u_1^5\right) \end{cases}$	40	80	600
14	$\begin{cases} 7 = 5300\frac{du_1}{d\tau} + 0.0195(u_1 - u_2) + 0.000001\left(u_1^2 - u_2^2\right) \\ 0.9 = 1550\frac{du_2}{d\tau} + 0.0195(u_2 - u_1) + 0.000001\left(u_2^2 - u_1^2\right) \end{cases}$	100	65	1500
15	$\begin{cases} 10 = 9900\frac{du_1}{d\tau} + 0.0235(u_1 - u_2) + 0.000082\left(u_1^3 - u_2^3\right) \\ 0.9 = 1280\frac{du_2}{d\tau} + 0.0235(u_2 - u_1) + 0.000082\left(u_2^3 - u_1^3\right) \end{cases}$	25	90	1800
16	$\begin{cases} 7 = 65000\frac{du_1}{d\tau} + 0.017(u_1 - u_2) + 0.000003\left(u_1^5 - u_2^5\right) \\ 1.8 = 13000\frac{du_2}{d\tau} + 0.017(u_2 - u_1) + 0.000003\left(u_2^5 - u_1^5\right) \end{cases}$	40	140	600
17	$\begin{cases} 3 = 9900\frac{du_1}{d\tau} + 0.0265(u_1 - u_2) + 0.000097\left(u_1^3 - u_2^3\right) \\ 0.7 = 580\frac{du_2}{d\tau} + 0.0265(u_2 - u_1) + 0.000097\left(u_2^3 - u_1^3\right) \end{cases}$	10	15	800
18	$\begin{cases} 19 = 500\frac{du_1}{d\tau} + 0.01(u_1 - u_2) + 0.000066\left(u_1^3 - u_2^3\right) \\ 1.2 = 3060\frac{du_2}{d\tau} + 0.01(u_2 - u_1) + 0.000066\left(u_2^3 - u_1^3\right) \end{cases}$	25	60	1500

№ вар.	Система уравнений	Начальные условия по умолчанию		Конечный момент времени по умолчанию
		$u1$	$u2$	
19	$\begin{cases} 5 = 7500\frac{du_1}{d\tau} + 0.008(u_1 - u_2) + 0.000024\left(u_1^3 - u_2^3\right) \\ 0.7 = 3390\frac{du_2}{d\tau} + 0.008(u_2 - u_1) + 0.000024\left(u_2^3 - u_1^3\right) \end{cases}$	10	140	1000
20	$\begin{cases} 3 = 4100\frac{du_1}{d\tau} + 0.001(u_1 - u_2) + 0.000053\left(u_1^3 - u_2^3\right) \\ 1.3 = 1660\frac{du_2}{d\tau} + 0.001(u_2 - u_1) + 0.000053\left(u_2^3 - u_1^3\right) \end{cases}$	75	5	1600
21	$\begin{cases} 17 = 6100\frac{du_1}{d\tau} + 0.0205(u_1 - u_2) + 0.000085\left(u_1^3 - u_2^3\right) \\ 1.3 = 130\frac{du_2}{d\tau} + 0.0205(u_2 - u_1) + 0.000085\left(u_2^3 - u_1^3\right) \end{cases}$	5	140	1300
22	$\begin{cases} 4 = 85000\frac{du_1}{d\tau} + 0.0005(u_1 - u_2) + 0.000009\left(u_1^4 - u_2^4\right) \\ 0.3 = 10000\frac{du_2}{d\tau} + 0.0005(u_2 - u_1) + 0.000009\left(u_2^4 - u_1^4\right) \end{cases}$	35	140	500
23	$\begin{cases} 17 = 6000\frac{du_1}{d\tau} + 0.01(u_1 - u_2) + 0.000003\left(u_1^4 - u_2^4\right) \\ 0.6 = 5000\frac{du_2}{d\tau} + 0.01(u_2 - u_1) + 0.000003\left(u_2^4 - u_1^4\right) \end{cases}$	20	135	1600
24	$\begin{cases} 21 = 83000\frac{du_1}{d\tau} + 0.0135(u_1 - u_2) + 0.000009\left(u_1^5 - u_2^5\right) \\ 1.6 = 12000\frac{du_2}{d\tau} + 0.0135(u_2 - u_1) + 0.000009\left(u_2^5 - u_1^5\right) \end{cases}$	20	120	300
25	$\begin{cases} 10 = 9300\frac{du_1}{d\tau} + 0.017(u_1 - u_2) + 0.000035\left(u_1^4 - u_2^4\right) \\ 1.9 = 2530\frac{du_2}{d\tau} + 0.017(u_2 - u_1) + 0.000035\left(u_2^4 - u_1^4\right) \end{cases}$	30	5	2500

ЛАБОРАТОРНАЯ РАБОТА №5. МОДЕЛИРОВАНИЕ ДВУМЕРНОГО ПРОЦЕССА В СРЕДЕ FREEFEM++ С ПРИМЕНЕНИЕМ МЕТОДА КОНЕЧНЫХ ЭЛЕМЕНТОВ

1 Основные сведения о работе

1.1 Цель работы

Целью работы является получение навыков моделирования линейных стационарных двумерных процессов в среде FreeFem++ с использованием метода конечных элементов.

1.2 Порядок выполнения работы

Теоретические сведения в методических указаниях состоят из четырех частей:

- В первой части приведены общие сведения о среде FreeFem++ и даны рекомендации по ее установке на компьютер для дальнейшего комфортного использования.
- Во второй части теоретических сведений рассматриваются некоторые проблемы, связанные с заданием области определения дифференциального уравнения в частных производных.
- В третьей части приведены некоторые сведения о методе конечных элементов.
- В четвертой части теоретических сведений описана реализация метода конечных элементов в среде FreeFem++, и приведены функции, требующиеся для задания вариационной формулировки дифференциального уравнения.

В практической части методических указаний приведены примеры задания области определения дифференциального уравнения в среде FreeFem++, примеры построения сетки конечных элементов, а также примеры решения стационарных двумерных задач. Все примеры необходимо выполнить.

Для получения допуска ко второй части работы должны быть предъявлены 3 файла программ для среды FreeFem++ и графический файл, полученные в результате выполнения практических примеров.

Во второй части работы необходимо самостоятельно решить задачу моделирования стационарного двумерного процесса теплопроводности, описываемого дифференциальным уравнением в частных производных, при этом также необходимо корректно построить геометрию области определения и сетку.

1.3 Используемое программное обеспечение

Для выполнения лабораторной работы необходим компьютер с установленной средой FreeFem++. Можно использовать любую версию

приложения, как с графическим интерфейсом, так и без него. Поскольку в некоторые версии приложения не входит GUI (графический интерфейс), то рекомендуется для таких версий использовать специальные редакторы программного кода.

Все примеры, описанные в данных методических указаниях, относятся к FreeFem++ версии 3.20 при работе совместно с Crimson Editor SVN286M. Рекомендации по установке для данной конфигурации описаны далее в п. 2.1 теоретических сведений. Однако более удобной является работа в интегрированной среде разработки FreeFem++-cs, снабженной интуитивным графическим интерфейсом. Эта конфигурация и является рекомендуемой.

Описанные конфигурации не являются единственно возможными. Выполнение лабораторной работы возможно в любой из описанных или неописанных конфигураций. Выбор конфигурации зависит от предпочтений студента и возможностей используемого компьютера.

Поскольку программы, написанные для среды FreeFem++, представляют собой текстовые файлы, то устанавливать специальный редактор программного кода или графический интерфейс нет необходимости. Можно редактировать программный код в «Блокноте» или прочих подобных программах. Однако в этом случае работа со средой FreeFem++ будет менее комфортной.

В зависимости от выбранной конфигурации среды FreeFem++ некоторые приемы работы с ней могут отличаться от приемов, приведенных в данных методических указаниях. Обычно это не представляет особых трудностей для студентов.

2 Теоретические сведения

2.1 Общие сведения о среде FreeFem++

Общие сведения и назначение

Среда FreeFem++[21] является высокоуровневой средой разработки для численного решения дифференциальных уравнений в частных производных[22]. По сути, она является компилятором языка программирования высокого уровня, базирующимся на языке программирования C++, т. е. корректнее говорить не о среде FreeFem++, а о языке программирования FreeFem++.

Среда FreeFem++ является свободно распространяемым программным обеспечением. Дистрибутивы программы и исходный код можно найти на

[21] Название среды составлено из двух частей. Первая часть «free» – от англ. «свободный». Вторая часть – аббревиатура FEM, от англ. «finite element method», т. е. «метод конечных элементов». «++» в конце названия означает, что программа написана на языке программирования C++.

[22] В англоязычной литературе часто используется аббревиатура PDE – Partial Differential Equations, что означает «дифференциальное уравнение в частных производных».

сайте https://github.com/FreeFem/FreeFem-sources/releases. Существуют версии среды для различных операционных систем: Microsoft Windows, MacOS и др.

Для языка программирования FreeFem++ разработаны различные альтернативные оболочки с графическим интерфейсом, облегчающие создание и редактирование программного кода. Они используют исходный компилятор FreeFem++. Одна из таких оболочек, FreeFem++-cs, также является свободно распространяемой и доступна для скачивания по адресу: https://www.ljll.math.upmc.fr/lehyaric/ffcs/install.htm (не забудьте предварительно установить freefem++).

Среда FreeFem++, в первую очередь, предназначена для изучения метода конечных элементов (МКЭ) и проведения исследований, относящихся к самому МКЭ. Среда FreeFem++ предназначена для использования ее студентами-магистрантами, исследователями и инженерами. Для ее использования необходимо понимать, как реализуется МКЭ.

Возможно использование среды FreeFem++ и для решения прикладных задач. В последнем случае традиционные CAE-системы[23], основанные на МКЭ, имеют существенные преимущества.

Среду FreeFem++ удобно, в основном, использовать только для решения двумерных дифференциальных уравнений в частных производных. Трехмерные задачи в среде FreeFem++ решать можно, но с большими сложностями из-за необходимости описания трехмерной геометрии при помощи уравнений. Используя среду FreeFem++, можно решить эллиптические, параболические и гиперболические дифференциальные уравнения второго порядка. Однако для решения параболических и гиперболических уравнений необходимо использовать итерационные методы.

Среда FreeFem++ использует вариационную формулировку МКЭ. Можно решить **любое** дифференциальное уравнение, для которого возможно задать вариационную формулировку, однако надо ее знать. Это отличает среду FreeFem++ от CAE-систем, в которых нет необходимости непосредственного задания уравнения. Необходимо лишь выбрать вид этого уравнения из предложенных вариантов, но среди них может и не быть нужного. Т. е. среду FreeFem++ можно применить для решения даже тех задач, стандартные методы решения которых в CAE-системах не предусмотрены.

Особенности

Можно выделить следующие особенности среды FreeFem++:

- Решаемая задача в среде FreeFem++ описывается вариационной формулировкой, но можно непосредственно (самостоятельно) задать матрицы системы уравнений для расчета узловых значений. Т. е. можно использовать и формулировку на основе метода Галеркина, но вручную.
- Возможно решение систем дифференциальных уравнений с несколькими неизвестными функциями.

[23] Подобная система будет рассмотрена в лабораторных работах 6 – 10.

- Возможно решение двумерных задач в декартовой и цилиндрической системе координат (осесимметричных задач).
- Возможно решение нестационарных и нелинейных задач, но соответствующие алгоритмы, сводящие задачу к системе линейных уравнений или к серии стационарных задач, должны быть написаны пользователем.
- Среда FreeFem++ не включает в себя CAD-систему и не позволяет импортировать геометрию области определения из CAD-систем. Это означает, что геометрия области определения дифференциального уравнения должна описываться математическими формулами, а не экспортироваться из редактора[24].
- Сетка конечных элементов строится автоматически, параметры сетки задаются пользователем.
- Возможна адаптация сетки (выбор наиболее подходящего размера элементов для каждой части области определения) для конкретной задачи.
- Возможно задание функций в аналитической форме.
- Имеется большой выбор различных видов конечных элементов, но, к сожалению, только треугольных и тетраэдрических.
- Наличие различных алгоритмов решения систем линейных алгебраических уравнений, как прямых, так и итерационных.
- Высокая скорость работы, сравнимая со скоростью работы программы, напрямую реализующей метод конечных элементов на языке программирования C++.
- Возможность экспорта построенных программой графиков и сеток конечных элементов в другие программы (как правило, при работе под операционную систему Linux).

К основным преимуществам среды FreeFem++ можно отнести возможность решения нестандартных задач и высокую скорость расчетов, а к недостаткам - сложности при работе со средой и неудобный интерфейс, особенно при решении трехмерных задач.

Язык программирования FreeFem++

Среда Freefem++ написана на языке C++, при этом язык программирования FreeFem++ не просто похож на язык программирования C++, а прямо использует его синтаксис и многие операторы. Самые последние версии среды стали включать некоторые операторы пакетов MATLAB/Scilab (например, оператор двоеточие для задания массивов). Однако следует учитывать некоторые особенности языка программирования FreeFem++, которые позволяют ему быть легко используемым для решения уравнений методом конечных элементов.

Подобно языку программирования C++:

[24] Это **очень** неудобно. Это можно отнести к одному из существенных недостатков среды FreeFem++.

• В языке FreeFem++ любая переменная должна быть предварительно объявлена.

• Операторы отделяются друг от друга точкой с запятой ;.

• Многие операторы языка C++ работают в среде FreeFem++ и имеют тот же синтаксис.

• Комментарии начинаются с //.

Однако!

• Имя переменной может содержать буквы и цифры, но не может содержать знака подчеркивания _.

• Помимо общепринятых типов переменных - действительных (real), целых (int), строковых (string), массивов, существуют такие встроенные типы как сетка конечных элементов (mesh), двумерное пространство конечных элементов (fespace) и аналитическая функция (func).

• Переменные x, y и z являются встроенными. Они всегда соответствуют пространственным координатам. Эти переменные не нуждаются в определении.

Эти и другие особенности языка будут проиллюстрированы на конкретных примерах в лабораторных работах.

Среда FreeFem++ не работает в режиме интерпретатора. Все программы компилируются. Программы представляют собой текстовый файл. Файл, в принципе, может иметь любое расширение. «Родным» для среды FreeFem++ расширением является *.edp. Файл можно создать в любом текстовом редакторе и сразу же запустить после компиляции в режиме командной строки.

Более удобным способом является использование интегрированной среды с графическим интерфейсом. Она включает в себя текстовый редактор. Для редактирования программ FreeFem++ можно также использовать различные редакторы программного кода.

Рекомендации по установке

Наиболее новую версию среды FreeFem++ можно найти на сайте http://www.freefem.org/ff++/index.htm, где можно выбрать версию в зависимости от операционной системы. После установки программы, в принципе, можно выполнять лабораторные работы, используя текстовый редактор для набора программ с последующим их ручным запуском. Однако это не очень удобно. Удобно использовать специальные редакторы программного кода, которые автоматически будут вызывать компилятор FreeFem++ прямо из редактора.

Альтернативным вариантом является использование графической оболочки, которую можно найти на сайте https://www.ljll.math.upmc.fr/lehyaric/ffcs/install.htm. В этом случае редактор программного кода устанавливать не нужно.

Компилятор FreeFem++ может работать со следующими редакторами программного кода (для операционной системы Windows):

- Crimson Editor, http://www.crimsoneditor.com/. Является свободно распространяемым, бесплатным для некоммерческого использования.
- Winedit, http://www.winedt.com/. Является наиболее мощным и многофункциональным, но платным.
- TeXnicCenter, http://www.texniccenter.org/. Является наиболее простым.

Для выполнения лабораторных работ рекомендуется использовать графическую оболочку или редактор Crimson Editor, пошаговое руководство по его подготовке к использованию совместно с компилятором FreeFem++ приведено ниже.

2.1.1. Установите среду FreeFem++ и редактор Crimson Editor.

2.1.2. Запустите редактор Crimson Editor. Выберите из главного меню редактора пункт Tools > Preferences. В раскрывшемся окне в меню слева выберите пункт File >Association. В появившемся списке справа выберите расширение .edp и нажмите кнопку Associate, чтобы при выборе файла с расширением .edp запускался редактор Crimson Editor.

2.1.3. В том же окне в меню слева выберите пункт Tools > User Tools. Справа в окне появятся поля, которые надо заполнить следующим образом. В поле Menu Text введите FreeFem++. В поле Command введите путь к файлу FreeFem++.exe, который находится в том каталоге, куда вы установили программу. Путь можно выбрать, нажав на кнопку ... справа от поля ввода. В поле Argument введите $(FilePath), в поле Initial Dir введите $(FileDir). Поставьте галочку у пункта Use short filename (8.3). Нажмите кнопку OK.

2.1.4. Закройте редактор Crimson Editor. Откройте каталог, в который установлена среда FreeFem++. Найдите в нем архив crimson-freefem++.zip. Распакуйте архив. Скопируйте распакованные файлы в соответствующие каталоги каталога, в который установлен редактор Crimson Editor. Т. е., например, все файлы из каталога link архива нужно скопировать в каталог link каталога с редактором Crimson Editor. Это необходимо для подсветки разными цветами команд в редакторе, все остальное будет работать и без этих действий.

2.2 Задание области определения дифференциального уравнения

Термины и определения

Если неизвестная функция, входящая в дифференциальное уравнение, является функцией двух или большего числа переменных, то дифференциальное уравнение называется **уравнением в частных производных**.

Рассмотрим физический смысл области определения в задачах технической физики. Для трехмерных задач пространственную часть области определения можно представить как тело или совокупность тел. Для двумерных задач область определения – плоская область или их совокупность.

Для одномерных задач область определения – отрезок прямой или их совокупность.

Например, если нужно найти температурное поле в активном элементе лазера, то для задания области определения необходимо задать форму активного элемента как твердого тела.

Область определения в случае переходных процессов также должна включать в себя отрезок на оси времени.

В принципе, область определения может быть неограниченной. Однако численные методы решения дают не функциональные зависимости, а значения искомой функции в виде совокупности чисел, поэтому они неудобны для работы с неограниченными областями определения.

Математическое описание области определения

Несложно понять, что для численного решения дифференциального уравнения при помощи компьютера его область определения должна быть описана математическими выражениями.

Математическое описание области определения для одномерных задач не представляет особой сложности. Отрезок на оси x, показанный на рисунке 3.1, например, можно задать как

$$x_1 < x < x_2\,, \tag{2.2.1}$$

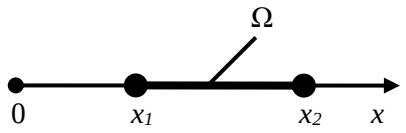


Рис. 3.1 – Отрезок на оси x

где x_1 и x_2 координаты граничных точек отрезка.

Аналогично задается и отрезок на оси времени:

$$\tau_1 < \tau < \tau_2\,, \tag{2.2.2}$$

где τ_1 – начальный рассматриваемый момент времени, τ_2– конечный рассматриваемый момент времени.

Сложнее обстоит дело для двумерных или трехмерных задач. Если записать область определения аналогично (2.2.1):

$$\begin{cases} x_1 \le x \le x_2 \\ y_1 \le y \le y_2 \end{cases}, \tag{2.2.3}$$

то таким образом можно задать только прямоугольник для двумерного случая (см. рис. 3.2) или прямоугольный параллелепипед для трехмерного случая. Однако область определения может иметь сложную форму, как, например, показано на рисунке 3.3.

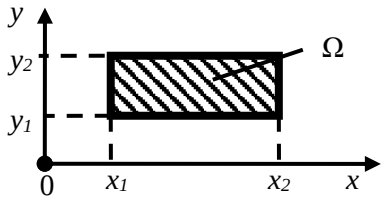


Рис. 3.2 - Прямоугольная область определения

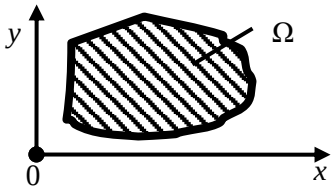


Рис. 3.3 - Область определения произвольной формы

Математическое описание такой области уже представляет собой отдельную задачу. В случае, когда область является трехмерной, задача еще больше усложняется.

При математическом описании области определения дифференциального уравнения обычно описывают границы этой области в параметрическом виде. Указывается, с какой стороны от границы находится рассматриваемая область, если это не очевидно. Границами области для двумерной задачи являются линии, а для трехмерной – поверхности. Линии могут быть криволинейными, а поверхность может быть неплоской.

В параметрической форме линия на плоскости описывается как

$$\begin{cases} x = f_x(t) \\ y = f_y(t) \end{cases}, \text{ где параметр } t_1 \le t \le t_2 . \qquad (2.2.4)$$

Вид функций f_x и f_y в (2.2.4) зависит от формы линий. Аналогичным образом описываются и поверхности в трехмерном пространстве.

Примеры параметрического описания области определения

В качестве примера рассмотрим часто встречающиеся параметрические представления окружности и прямой линии.

Окружность (см. рис. 3.4) с радиусом r, центр которой имеет координаты (x_0, y_0), в параметрической форме задается следующим уравнением (для плоского случая):

$$\begin{cases} x = x_0 + r\cos(t) \\ y = y_0 + r\sin(t) \end{cases}, \text{ где } 0 \le t \le 2\pi . \qquad (2.2.5)$$

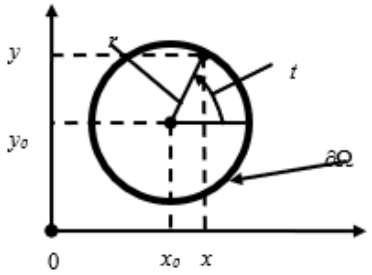


Рис. 3.4 - Параметрическое представление окружности

Поскольку параметр t в этом случае соответствует углу дуги окружности, то часть окружности можно задать, изменив значения параметра t. Например, ¼ окружности можно задать, если $0 \le t \le \frac{\pi}{2}$.

Прямая линия с уравнением $y = kx + b$ в параметрическом виде задается как

$$\begin{cases} y = t + b \\ x = t/k \end{cases}, \text{ где } -\infty \le t \le +\infty . \qquad (2.2.6)$$

Отрезок прямой линии на плоскости, ограниченный двумя точками с координатами (x_1, y_1) и (x_2, y_2), показанный на рис. 3.5, в параметрическом представлении описывается следующим образом:

$$\begin{cases} x = (x_2 - x_1)\cdot t + x_1 \\ y = (y_2 - y_1)\cdot t + y_1 \end{cases}, \ 0 \le t \le 1. \qquad (2.2.7)$$

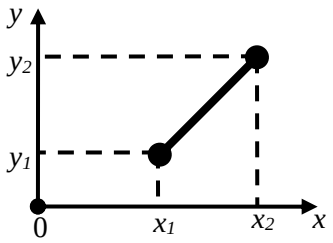


Рис. 3.5 - Отрезок на плоскости

Уравнения других кривых можно найти в справочниках по математике или вывести самостоятельно.

Область определения и численное решение задачи

Область определения в задачах технической физики, как правило, является непрерывной. В то же время одним из условий применения численных методов решения является сведение математической задачи к конечномерной [4]. Это означает переход от функции непрерывного аргумента к функции дискретного аргумента.

В свою очередь, переход к функции дискретного аргумента означает, что множество, представляющее собой непрерывную область определения дифференциального уравнения, заменяется дискретным множеством. Это множество представляется, как правило, в виде конечного числа точек исходной области определения, обычно называемых **узлами**. Совокупность узлов, образующая новую дискретную область определения дифференциального уравнения, называется **сеткой**. Во многих численных методах для двумерных и трехмерных задач имеет значение, как соединены между собой узлы. В этом случае под сеткой может пониматься совокупность узлов и связей между ними. Переход от непрерывной области определения к сетке для одномерного и двумерного случая показан на рис. 3.6.

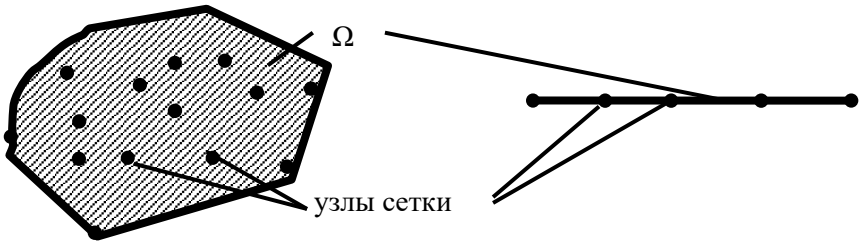


Рис. 3.6 - Переход от непрерывной области определения к сетке в двумерном (слева) и одномерном (справа) случае

При решении нестационарных задач, как правило, отдельно выделяют узлы пространственной области определения и узлы на оси времени. Конечно, их можно представить как единые узлы в четырехмерном пространстве, но это представление не всегда удобно.

В зависимости от используемого численного метода могут накладываться те или иные ограничения на расположение узлов, но в некоторых случаях расположение узлов является произвольным. Следует, однако, учитывать, что погрешность расчетов с использованием численных методов в первую очередь определяется количеством и расположением узлов. Анализ погрешности численного метода всегда связан с характеристиками сетки.

При корректной постановке задачи увеличение числа узлов ведет к уменьшению погрешности, но неудачный выбор расположения узлов может привести, наоборот, к ее увеличению. Требования к сетке различаются в зависимости от используемого численного метода решения.

Значение неизвестной функции при численном решении задачи находится только для узлов сетки, однако многие численные методы одновременно предоставляют способ аппроксимации, который дает возможность вычислить значение функции для произвольной точки области определения.

В данной лабораторной работе предполагается создание сетки для дальнейшего решения задачи методом конечных элементов. При необходимости сведения о методе конечных элементов и некоторые требования к сетке можно найти в [5, 6].

Задание области определения в среде FreeFem++

В среде FreeFem++ нет функций, позволяющих импортировать геометрию области определения, построенную в CAD-системе. Геометрию области определения необходимо задавать непосредственным образом, в параметрической форме.

Область определения задается путем задания ее границ и указания, с какой стороны границы находится рассматриваемая область. Отдельная линия границы задается следующим образом:

```
border bname(t=t0,tk){x=f1(t);y=f2(t);label=lab;}
```

Здесь:

- bname – имя границы, задаваемое пользователем. Оно используется для обращения к этой границе в дальнейшем.
- t – имя параметра в параметрическом представлении границы.
- t0, tk – начальное и конечное значения параметра соответственно (числа).
- f1, f2 – аналитические функции, описывающие параметрическое представление. Их можно задать непосредственно в строке или отдельно.
- lab – необязательный параметр (число), задающий номер, который будет присвоен границе. К границе можно обращаться, используя как ее имя bname, так и ее номер. Номер нужен при сохранении геометрии в файле, так как имя границы при сохранении теряется.

В среде FreeFem++ считается, что область определения находится слева по направлению изменения параметра t. Таким образом, если в параметрическом представлении окружности (2.2.5) параметр t меняется от $t0 = 0$ до $tk = 2\pi$, то описывается внутренняя область окружности, т. е. круг, как показано на рис. 3.7. Если же, напротив, параметр t меняется от $t0 = 2\pi$ до $tk = 0$, то описывается внешняя часть окружности, т. е. плоскость без круга, как показано на рис. 3.8.

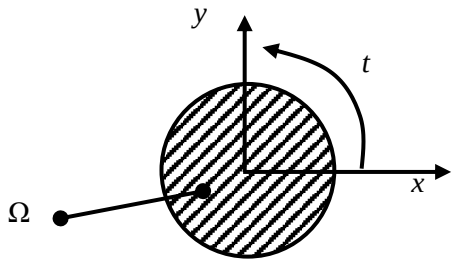


Рис. 3.7 – Внутренняя область окружности

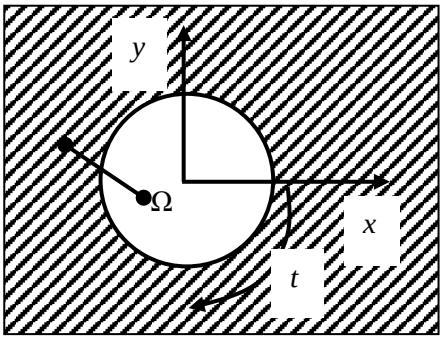


Рис. 3.8 – Внешняя часть окружности

Границы можно складывать для получения области, состоящий из нескольких линий, описывающихся различным образом. Граница, вычитающаяся из суммы, рассматривается как граница с противоположным направлением изменения параметра t. Точки пересечения линий, описывающих составную границу, необходимо рассчитывать самостоятельно. Соответствующие значения параметра t должны быть заданы пользователем.

Построение сетки конечных элементов в среде FreeFem++

При решении дифференциальных уравнений в частных производных при помощи МКЭ исходная область определения должна быть разбита сеткой конечных элементов. Элементы имеют, как правило, простую форму, например, треугольную или четырехугольную для двумерных задач. Элементы могут иметь криволинейные стороны, если используются нелинейные функции

формы. Поскольку допустимая форма элемента определяется функцией формы, то обычно сразу выбирают и ее. Выбор типа конечного элемента влияет на точность расчетов и скорость решения задачи.

Криволинейные границы при выборе линейных конечных элементов аппроксимируются ломаными линиями. В задачах теплопроводности это обычно не вносит существенной погрешности в результаты расчета. Однако в задачах прочности такая аппроксимация часто неприемлема. В таких случаях необходимо использовать элементы высших порядков.

В среде FreeFem++ для построения сетки можно выбрать элементы различных порядков, использующие различные аппроксимации, но только треугольной формы. Четырехугольная форма элементов не поддерживается.

В среде FreeFem++ сетка конечных элементов строится автоматически для выбранной области определения. Для построения сетки используется алгоритм триангуляции Делоне-Вороного, он описан, например, в [6]. Пользователю только необходимо выбрать тип конечных элементов, разбиваемую область, описываемую своими границами, и число разбиений каждой части границы области.

Чем больше число разбиений границ области, тем меньше размер элементов. Чем меньше размер элементов, тем, в общем случае, меньше погрешность расчетов, однако тем меньше скорость расчета. Основные правила, которых следует придерживаться при выборе числа разбиений границ, следующие:

- В тех областях, где решение представляет наибольший интерес, сетка должна быть плотнее;
- В тех областях, где ожидается сильное изменение (большие градиенты) искомой функции, сетка должна быть плотнее;
- Желательно не строить излишне плотную сетку, так как это замедляет расчеты. Может не хватить памяти компьютера для проведения расчетов.

Иногда для получения заданного разбиения приходится разбивать границу на несколько частей, чтобы задать разное число разбиений для каждой части.

В среде FreeFem++ сетка конечных элементов представляет собой особый тип переменных, имя этого типа mesh[25]. Для определения[26] переменной me типа mesh необходимо ввести следующее:

```
mesh me;
```

Сетка строится при помощи функции buildmesh. Часто удобно сразу присвоить соответствующее значение переменной при ее определении. Тогда для построения сетки и помещения ее в переменную me в простейшем случае необходимо задать следующее:

[25] От англ. mesh – сетка.

[26] В среде FreeFem++ подобно языку программирования C++ все переменные, кроме встроенных переменных, должны быть предварительно определены.

```
me=buildmesh(b1(nb1)+b2(nb2)+…+bk(nbk));
```

Здесь me – имя переменной типа mesh, в которую помещается построенная сетка, b1, b2, …, bk – границы, описывающие область определения дифференциального уравнения, nb1, nb2, …, nbk – соответствующее число разбиений каждой границы.

2.3 Общие сведения о методе конечных элементов

Введение

Метод конечных элементов (МКЭ[27]) является численным методом решения дифференциальных уравнений. В настоящее время он практически не используется для ручных расчетов, так как его реализация требует проведения большого объема вычислительных операций.

На настоящий момент МКЭ является одним из самых популярных численных методов решения дифференциальных уравнений. Он является основой большого числа CAE-систем [6], в том числе основой наиболее популярных на настоящий момент систем NASTRAN и ANSYS, поэтому понимание основ МКЭ является важным для проведения инженерных расчетов на современном уровне.

Не удивительно, что изучению особенностей МКЭ и его применению для решения технических задач посвящено множество работ [например, 7 – 9]. В указанных работах описана вариационная формулировка МКЭ. В настоящее время довольно часто используется формулировка МКЭ на основе метода Галеркина, в связи с тем, что она позволяет охватить более широкий спектр решаемых задач. Эта формулировка описана в [5]. Использование МКЭ применительно к задачам теплообмена описано в [10].

Основные положения МКЭ

Далее рассматриваются лишь некоторые основные идеи, лежащие в основе МКЭ.

В МКЭ, как и в других численных методах, исходная непрерывная область определения дифференциального уравнения заменяется дискретной областью, состоящей из конечного числа элементов. Решение дифференциального уравнения находится лишь для конечного числа точек – узлов, которые, как правило, но не всегда, расположены в вершинах элементов. Решение уравнений для точек, не являющихся узловыми, может быть получено путем интерполяции вычисленных значений функции для узлов.

Форма конечных элементов может быть различной, но обычно используются элементы простой формы: отрезки для одномерной задачи, треугольники и четырехугольники для двумерной задачи, шестигранники,

[27] В английском языке используется аббревиатура FEM – finite element method. Эту аббревиатуру можно встретить и в русскоязычных изданиях.

тетраэдры, пирамиды с четырехугольником в основании для трехмерной задачи.

Рассмотрим основные положения МКЭ на примере дифференциального уравнения для двумерного стационарного случая. В этом случае необходимо найти функцию u_a, которая зависит от двух координат x и y.

Функция $u_a(x, y)$, являющаяся приближенным решением дифференциального уравнения для двумерного стационарного случая, в МКЭ ищется в виде

$$u_a(x, y) = \sum_{i=1}^{N} \varphi_i(x, y) \cdot u_{ai} , \qquad (2.3.1)$$

где x и y – пространственные координаты, являющиеся независимыми переменными в дифференциальном уравнении, N – число узлов в разбиении области определения, u_{ai} – значение функции u_a в узловой точке i. Функции φ_i, входящие в выражение (2.3.1), называются **функциями формы**. Они определяются только геометрией конечных элементов и, следовательно, являются известными. Неизвестными величинами в (2.3.1) являются узловые значения функции u_{ai}.

Одна из основных идей метода конечных элементов состоит в том, что функции формы имеют локальный характер. Т. е. функция формы φ_i принимает ненулевое значение только для точек, находящихся в пределах элементов, которым принадлежит узел i. Также для выполнения равенства (2.3.1) достаточно, чтобы выполнялось следующее условие для значений функций формы в узлах:

$$\text{для } u_{ai} \begin{cases} \varphi_i = 1 \\ \varphi_k = 0, \quad если \quad i \neq k \end{cases} . \qquad (2.3.2)$$

Т. е. функция формы равна единице в том узле, к которому она относится, и равна нулю во всех остальных узлах (но не обязательно между ними).

Обычно функции формы выбираются так, чтобы обеспечивалась непрерывность функции u_a (но не обязательно ее производных) по всей области определения, так как в физических задачах функции обычно непрерывны.

Как правило, функции формы представляют собой кусочно-полиномиальные функции[28]. Порядок полинома определяет необходимое число узлов в элементе для однозначного задания функции. Чаще всего используется полином первого порядка. Такая функция формы называется линейной, а конечный элемент должен содержать узлы только в вершинах. Для полинома

[28] В т. ч. может рассматриваться функция нескольких переменных для двумерных и трехмерных задач.

второго порядка – квадратичной функции формы, элемент должен дополнительно содержать узлы на каждой стороне элемента.

Неизвестные узловые значения функции u_{ai} определяются после подстановки выражения (2.3.1) в исходное дифференциальное уравнение с применением метода Галеркина или вариационного метода Релея-Ритца. Оба этих метода являются приближенными методами решения дифференциальных уравнений. В результате получается система N алгебраических уравнений, содержащих u_{ai} как неизвестные. Полученная матрица системы уравнений является разреженной, т. е. большая часть элементов этой матрицы равна нулю. Для решения таких уравнений используются специальные методы, позволяющие существенно уменьшить число операций, требующихся для расчетов, по сравнению с методом Гаусса. После того, как узловые значения определены, можно найти значение искомой функции в любой точке области определения, используя формулу (2.3.1).

Решение одномерных и трехмерных задач происходит аналогичным образом. При решении нестационарных задач после подстановки (2.3.1) в исходное дифференциальное уравнение получается не система алгебраических уравнений, а система обыкновенных дифференциальных уравнений. Для ее решения используют, как правило, метод Эйлера или метод Рунге-Кутты.

В случае если исходное дифференциальное уравнение является нелинейным, или если нелинейными являются функции формы, то полученная система алгебраических или обыкновенных дифференциальных уравнений также будет нелинейной.

МКЭ обобщается и на случай системы дифференциальных уравнений в частных производных.

Решение задач методом конечных элементов

Решение задачи методом конечных элементов для простейшего стационарного случая происходит в следующей последовательности.

1. Вначале разрабатывается математическая модель. Здесь необходимо задать дифференциальные уравнения (систему уравнений), краевые условия и область определения дифференциального уравнения. На этом этапе область определения может быть представлена в графическом виде.

2. Проводятся математические преобразования исходного уравнения, чтобы привести его к стандартному виду. Требующийся вид зависит от используемого программного обеспечения.

3. Составляется математическое описание области определения, если она была представлена в графическом виде.

4. Выбирается тип функции формы и тип конечного элемента. Эти два выбора связаны, так как функция формы определяет число узлов и форму элемента.

5. Проводится разбиение области определения на конечные элементы. Полученная совокупность элементов называется конечно-элементной сеткой

или просто сеткой. В процессе разбиения каждому узлу или элементу присваивается свой номер.

6. Формируется матрица системы уравнений, необходимая для расчетов. Матрица формируется с использованием метода Галеркина или метода Релея-Ритца. В системе уравнений неизвестными являются узловые значения функции.

7. Решается система алгебраических уравнений, находятся узловые значения искомой функции.

8. Обрабатываются результаты расчетов. Здесь, например, может осуществляться построение графиков или вычисление средних значений. Также могут быть вычислены значения функции между узлами и производные величины.

Используя практически любое математическое программное обеспечение можно автоматизировать выполнение этапа 7 (решение системы алгебраических уравнений). При наличии в этом программном обеспечении графической оболочки можно автоматизировать выполнение этапа 8.

Автоматизированное построение сетки конечных элементов с выбором соответствующего типа конечного элемента возможно при помощи специальных сеточных генераторов, например, генераторов Grid и Gmesh. Они дополнительно требуют построения геометрии области определения.

Однако намного удобнее использовать специальные пакеты расширения (toolbox) для математического программного обеспечения. Они реализуют полный набор средств для решения задач методом конечных элементов. Например, для математических сред MATLAB и Scilab существует свободно распространяемый пакет расширения OpenFEM. В пакет MATLAB входит коммерческий пакет расширения PDE. Для различных языков программирования также созданы библиотеки, реализующие средства для решения задач методом конечных элементов. Эти средства включают сеточные генераторы, алгоритмы, формирующие матрицу системы уравнений, алгоритмы для решения систем алгебраических уравнений, для построения графиков и пр.

Отдельно такие средства реализованы в среде FreeFem++, они и рассматриваются в данной лабораторной работе.

Недостатком подобных средств является необходимость самостоятельного объединения их пользователем в единый алгоритм для получения решения. Т. е. для получения решения задачи все равно нужно хорошо представлять алгоритм расчета в МКЭ.

Существенно облегчают процесс решения задачи CAE-системы, в основе которых лежит МКЭ. К наиболее популярным на настоящий момент CAE-системам можно отнести ANSYS, MD Nastran, COMSOL, LS-DYNA, Femap и пр. Во многих случаях эти системы позволяют решать задачи, вообще не концентрируясь на МКЭ. Пользователь, в основном, должен думать о том, чтобы корректно поставить задачу, а ее решение автоматизируется. В целом, такие системы будут наилучшим выбором для решения задачи с

использованием МКЭ. Однако решение сложных задач в таких системах все же требует понимания МКЭ. В них существует большая вероятность получения правдоподобного, но неправильного решения при отсутствии понимания МКЭ.

Вторым недостатком CAE-систем является отсутствие возможности выхода за рамки тех математических моделей, которые заложены в системах. В случае необходимости решения дифференциального уравнения в частных производных нестандартного вида могут возникнуть сложности при попытке использовать такие системы. В этом случае пакеты расширения, содержащие отдельные средства для реализации метода конечных элементов, являются более гибкими. Они часто накладывают меньшие ограничения на вид решаемого уравнения, позволяя пользователю задавать уравнение нужного вида.

2.4 Метод конечных элементов в среде FreeFem++

Последовательность решения задачи методом конечных элементов в среде Freefem++

Применение интегрированной среды разработки FreeFem++ позволяет автоматизировать большинство операций (они описаны в п. 2.3), выполняемых при решении задачи методом конечных элементов.

Последовательность расчетов методом конечных элементов в среде FreeFem++ в простейшем случае следующая:

1. Перед работой на компьютере необходимо составить математическое описание задачи (математическую модель) и выяснить, какие величины требуется найти. Убедиться, что задача включает уравнения в частных производных и может быть решена в среде FreeFem++.

2. Далее в среде FreeFem++ составляется математическое описание границ рассматриваемой области определения дифференциального уравнения. Используется оператор border..

3. Строится разбиение области определения дифференциального уравнения сеткой треугольников. Это еще не сетка конечных элементов, а только разбиение на треугольники. Полученная сетка сохраняется в переменной специального типа mesh.

4. Создается пространство конечных элементов, т. е. сетка конечных элементов. При этом выбирается тип элемента, который связан с определенной функцией формы, и сетка из треугольников. Пространство конечных элементов сохраняется в переменной специального вида fespace.

5. В созданном пространстве конечных элементов определяются две функции – искомая неизвестная функция и функция формы.

6. Задается вариационная формулировка решаемого дифференциального уравнения в частных производных. Она включает в себя и граничные условия. Вариационная формулировка задается в виде переменной специального типа problem.

7. Решается уравнение, используется оператор solve. В качестве альтернативы можно напрямую сформировать матрицу системы уравнений и решить ее.

8. Выводятся результаты расчетов. Для построения графиков используется функция plot. При необходимости рассчитываются и выводятся на экран производные величины.

Типы конечных элементов в среде FreeFem++

В среде FreeFem++ используются конечные элементы только треугольной формы. Различия между элементами определяются используемой функцией формы и соответствующим числом узлов в элементе.

Основные типы конечных элементов, используемые в среде Freefem++, и соответствующие им функции формы приведены в таблице 4, остальные элементы можно найти в руководстве [2].

Наиболее часто используются элементы P1 и P2. Они представлены на рис. 3.9. Жирными точками обозначены узлы элементов. Элемент P2 имеет более высокий порядок аппроксимации, что приводит к получению более точного решения за счет уменьшения скорости расчетов. Опыт показывает, что для решения большинства задач теплопроводности достаточно применения элемента P1.

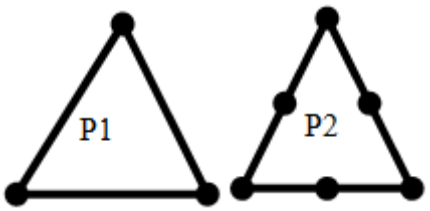


Рис. 3.9 - Конечные элементы в среде Freefem++

Табл. 4 Типы конечных элементов в среде Freefem++

Обозначение	Вид функции формы	Число узлов
P0	кусочно-постоянная, имеет разрывы на границах элементов	1
P1	кусочно-линейная, непрерывна на границах элементов	3 узла в вершинах
P1b	кусочно-линейная, непрерывна на границах элементов	4 узла: 3 в вершинах, 1 в центре
P2	кусочно-квадратичная, непрерывна на границах элементов	6 узлов: 3 в вершинах, 3 в центре каждой стороны

Задание вариационной формулировки в среде FreeFem++

Решаемая задача в среде FreeFem++ определяется посредством ее вариационной формулировки, характерной для метода Релея-Ритца. Способы получения такой формулировки и сами формулировки можно найти в соответствующей литературе [7 – 9].

В среде FreeFem++ решение можно получить несколькими способами. В одном из них вначале определяется вариационная формулировка уравнения как переменная особого типа. Используется ключевое слово `problem`. Затем

уравнение автоматически решается. Для этого достаточно обратиться к заданной ранее переменной по имени. Используется следующий синтаксис.

```
//Определяется задача
problem problem_name(u,phi)=f(u,phi);
//Решается задача
problem_name
```

Здесь problem_name – имя задачи (переменная), присваиваемое пользователем, u – имя искомой функции, phi – имя функций формы, f – аналитическая функция, описывающая вариационную формулировку задачи. Функции u и phi предварительно должны быть определены для некоторого пространства конечных элементов, которое и задает область определения дифференциального уравнения.

Уравнение можно решить и сразу, не используя предварительное задание. Для этого используется оператор solve следующим образом:

```
solve problem_name(u,phi)=f(u,phi);
```

Для задания операций интегрирования и производных, необходимых в вариационной формулировке, в среде FreeFem++ существуют специальные встроенные функции. Они приведены в таблице 5. При этом необходимо помнить, что переменные x и y являются встроенными, т. е. нет необходимости указывать эти переменные в качестве аргументов функций.

Табл. 5 Функции среды FreeFem++ для задания интегралов и производных

Выражение	Функция среды Freefem++	Комментарии
$\frac{df1}{dx}$	dx(f1)	производная от функции f1 по координате x
$\frac{df1}{dy}$	dy(f1)	производная от функции f1 по координате y
$\iint_D f1dxdy$	int2d(D)(f1)	двойной интеграл от функции f1 по области определения D (должна иметь тип mesh)
$\iint_{D1\subset D} f1dxdy$	int2d(D,D1)(f1)	двойной интеграл от функции f1 по подобласти D1 области определения D, $D1\subset D$
$\int_{g1} f1ds$	int1d(D,g1)(f1)	криволинейный интеграл от функции f1 по границе g1 области определения D

Операторы интегрирования можно также использовать для численного интегрирования функций, заданных в пространстве конечных элементов. Таким образом, среду FreeFem++ **можно использовать для численного расчета криволинейных интегралов и двойных интегралов по областям и линиям**

произвольной формы. Эти возможности реализованы лишь в малом числе программных продуктов.

Если есть необходимость задания граничного условия Дирихле ($u = u0$) на границе g1, то это условие задается как on(g1,u=u0).

Вариационная формулировка задачи теплопроводности в среде FreeFem++

В общем виде линейная двумерная стационарная задача теплопроводности описывается следующим дифференциальным уравнением в частных производных для некоторой области D на плоскости (x, y):

$$\lambda\left(\frac{\partial^2 t}{\partial x^2}+\frac{\partial^2 t}{\partial y^2}\right)+W=0\text{, для всех }(x,y)\in D. \qquad (2.4.1)$$

Здесь $t(x,y)$ - температура в точке (x,y), которая является искомой функцией, λ – коэффициент теплопроводности материала области, W – объемная плотность мощности источников теплоты внутри тела.

В общем случае на границе области определения ∂D могут быть заданы различные граничные условия. Пусть на границе *g1* задано граничное условие первого рода (условие Дирихле), на границе *g2* задано граничное условие второго рода (условие Неймана), а на границе *g3* задано граничное условие третьего рода (смешанное граничное условие). Причем $g1\cap g2\cap g3=\partial D$, где ∂D – вся граница области D. Таким образом,

$$\begin{cases} t=t1 \quad для \quad g1 \\ -\lambda\dfrac{\partial t}{\partial n}=q2 \quad для \quad g2 \\ \alpha3(t-ta)=-\lambda\left(\dfrac{\partial t}{\partial n}\right) \quad для \quad g3 \end{cases}, \qquad (2.4.2)$$

где *t1* – заданная постоянная температура на границе области, *q2* – заданная плотность теплового потока на границе области, $\alpha 3$и ta – заданные коэффициент теплоотдачи и температура окружающей среды на границе области соответственно, n – вектор нормали к соответствующей границе области.

Без учета граничных условий Дирихле вариационная формулировка для этой задачи имеет вид [2] (φ - функция формы).

$$\iint_D \lambda\left(\frac{\partial t}{\partial x}\frac{\partial \varphi}{\partial x}+\frac{\partial t}{\partial y}\frac{\partial \varphi}{\partial y}\right)dxdy-\iint_D W\varphi dxdy+ \\ +\int_{g3}\alpha 3\cdot t\cdot\varphi ds-\int_{g3}\alpha 3\cdot ta\cdot\varphi ds-\int_{g2}q2\cdot\varphi ds=0 \qquad (2.4.3)$$

Используя функции, приведенные в Табл. 2, вариационную формулировку (2.4.3) с граничными условиями Дирихле на границе *g1* можно представить в среде FreeFem++ следующим образом:[29]

```
problem conduct(t,phi)=
int2d(D)(lam*(dx(t)*dx(phi)+ dy(t)*dy(phi)))
-int2d(D)(W*phi)
+int1d(D,g3)(a3*t*phi)-int1d(D,g3)(a3*ta*phi)
-int1d(D,g2)(q2*phi)
+on(g1,t=t1);
```

Значения переменных или констант `lam`, `W`, `a3`, `ta`, `t1` должны быть предварительно заданы.

3 Задание области определения и построение сетки в среде FreeFem++

3.1 Первое знакомство

Далее описано, как работать со средой FreeFem++ с установленным редактором Crimson Editor. Работа с графической оболочкой напоминает работу с редактором Crimson Editor, работа с графической оболочкой также описана в п. 1. Если не установлен редактор или графическая оболочка, то прочитайте приложение (раздел 6).

3.1.1. Запустите редактор Crimson Editor, выбрав иконку  на рабочем столе, или иным способом. Появится окно редактора (рис. 3.10). В окне редактора можно набирать текст программ для различных сред и языков программирования.

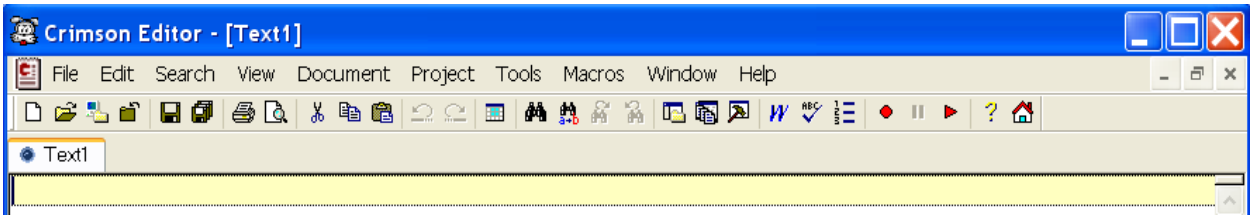


Рис. 3.10 – Окно редактора Crimson Editor

3.1.2. Попробуйте ввести и выполнить простейшую программу для среды FreeFem++. В среде FreeFem++ можно проводить арифметические операции над переменными и выводить их значения в окно вывода точно так же, как в языке программирования C++. Наберите в окне текстового редактора следующее:

```
real a=5,b=0; //Объявление переменных
```

[29] Переносы строки сделаны для лучшего понимания. В реальной программе их делать не обязательно.

```
      b=2*pi*a; //Вычисление значения переменной b
      cout <<"a="<<a<<" b="<<b<<endl; //Вывод значений a и
b
      cout <<"sin(b)="<<sin(b)<<endl; //Вычисление и вывод
sin(b)
```

В первой строке программы объявляются две новые переменные a и b типа real (действительные). Им присваиваются начальные значения.

Во второй строке вычисляется значение переменной $b = 2\pi a$.

В третьей строке в окно вывода выводится значение переменной a. Используется стандартный поток вывода языка программирования C++ cout. Строковые переменные заключаются в двойные кавычки ". endl возвращает конец строки (т. е. служит для перевода на новую строку).

3.1.3. Перед запуском программы ее следует сохранить[30]. Выберите из меню редактора пункт File>Save и сохраните файл под именем exl5_1.edp. Если редактор был настроен правильно, то после сохранения файла в формате Freefem++ загрузятся синтаксические правила, и текст программы будет подсвечен разными цветами (рис. 3.11).

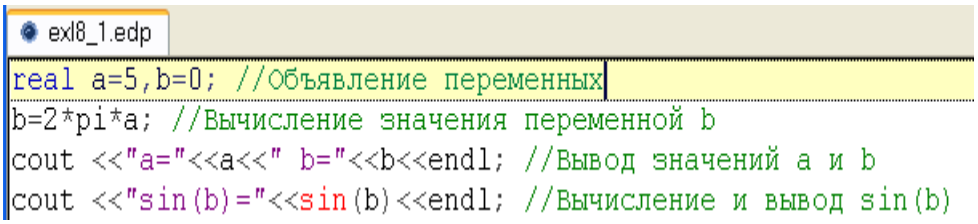


Рис. 3.11 – Результат подсвечивания текста программы

3.1.4. Теперь можно запустить программу в среде FreeFem++. Для этого выберите из меню редактора пункт Tools>Freefem++. Программа выполнится, результаты ее выполнения будут выведены в командном окне, которое появится после запуска программы. Обратите внимание на сообщения, выводимые в командном окне (рис. 3.12).

[30] Графический интерфейс FreeFem++-cs позволяет запускать программы без их сохранения.

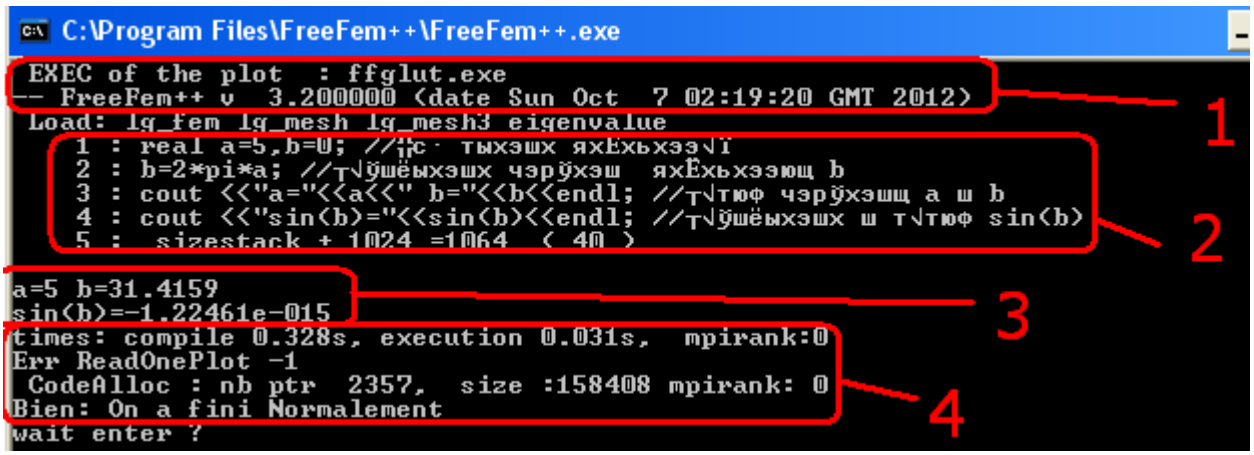


Рис. 3.12 – Вывод сообщений в командном окне

1 – Версия программы и загруженные модули,

2 – Текст программы, кириллица не воспринимается[31], поэтому комментарии могут отображаться неправильно,

3 – Результаты работы программы (те, которые мы выводим с помощью cout),

4 – Сообщения компилятора (время выполнения, ошибки и пр.)

Нажмите клавишу ENTER, чтобы закрыть командное окно. Результаты, выведенные в командном окне, также сохраняются в файле[32] с расширением *.log. Он находится в том же каталоге, что и запускаемая программа.

3.1.5. В среде FreeFem++ можно задавать массивы, проводить поэлементные операции над ними, использовать операторы цикла и ветвления. Однако при решении простых задач без этих операторов можно обойтись. При желании операторы можно найти в разделе Syntax руководства по среде FreeFem++.

3.2 Построение простейшей сетки

3.2.1. Постройте сетку для круглой области. Пусть радиус круга составляет 20 мм. Вначале создайте новый файл в редакторе, выбрав пункт File > New из главного меню.

3.2.2. При построении удобнее задавать все единицы измерения в системе СИ, чтобы избежать проблем в задании различных единиц. Поэтому радиус круга полагается равным 0,02 м. Наберите следующую программу:

```
border cir1(t=0,2*pi){x=0.02*cos(t);y=0.02*sin(t);}
plot(cir1(30));
```

В первой строке этой программы задается граница cir1 – окружность. Для ее задания используется параметрический вид (2.2.5). Во второй строке

[31] В интегрированной среде Freefem++-cs проблем с символами кириллицы отмечено не было.

[32] Этот файл может не записываться, зависит от установок программы.

строится график, на котором отображается граница cir1, разбитая на 30 линий. Команда plot используется в среде FreeFem++ для построения различных графиков.

3.2.3. *Сохраните программу под именем ex15_2.edp. Запустите ее.* Помимо командного окна появиться графическое окно с окружностью (окно может оказаться под другими окнами), как показано на рисунке 3.13. *Закройте графическое окно и командное окно*.

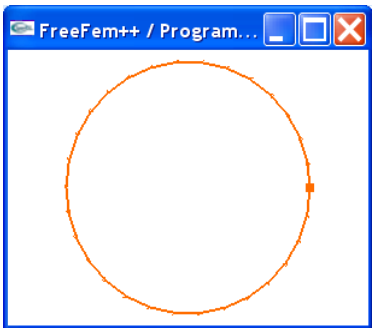


Рис. 3.13 – Графическое окно с окружностью

3.2.4. Теперь можно построить сетку. Добавьте в программу строку, создающую сетку конечных элементов m1, используя 20 разбиений окружности. Команда plot может использоваться и для вывода на экран сетки.

```
mesh
m1=buildmesh(cir1(20));
plot(m1);
```

3.2.5. Сохраните программу под тем же именем. Запустите ее. В графическом окне появится построенная сетка.

3.2.6. Оба рисунка в графическом окне – граница и сетка – строятся без паузы между построениями (рис. 3.14). Из-за этого первый рисунок не виден. Решить эту проблему можно путем введения дополнительного параметра в функцию plot.

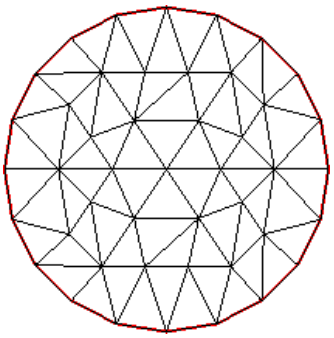


Рис. 3.14 – Построение границы и сетки в одном окне

Параметр wait=1 показывает, что необходимо приостановить выполнение программы до нажатия клавиши на мыши или клавиатуре.

Измените две строки программы, содержащие функцию plot, следующим образом:

```
…
plot(cir1(30),wait=
1);
…
plot(m1,wait=1,grey
=1);
```

Параметр grey=1 указывает, что график нужно выводить, используя только оттенки серого цвета, что удобно для печати на принтере.

3.2.7. *Сохраните программу под именем ex15_3.edp. Запустите ее.*

3.2.8. В графическом окне вначале появится первый рисунок (рис. 3.15) – граница области. Выполнение программы будет приостановлено.

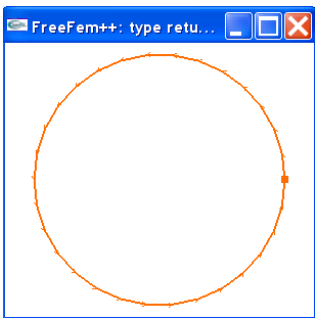


Рис. 3.15 – Граница области

В паузе можно изменять масштаб рисунка.

Для уменьшения рисунка нажмите клавишу минус -, для увеличения – клавишу плюс +, стрелки позволяют двигать увеличенное изображение.

3.2.9. Чтобы продолжить выполнение программы, нажмите клавишу ENTER.

3.2.10.Появится следующий рисунок (рис. 3.16). Он черно-белый, так как в параметрах функции plot был указан параметр grey=1.

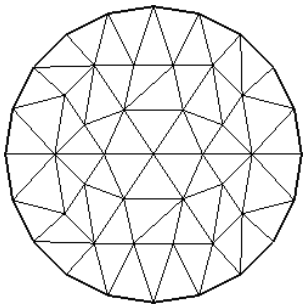


Рис. 3.16 – Результат построения черно-белого рисунка

Этот рисунок также можно масштабировать. По окончании необходимо нажать клавишу ESC, чтобы завершить выполнение программы.

3.3 Построение графика функции в пространстве конечных элементов

3.3.1. Построенная в п. 3.20 сетка является лишь геометрическим разбиением рассматриваемой области. Для решения задачи методом конечных элементов нужно создать пространство конечных элементов, выбрав тип элементов. Пространство конечных элементов – это тип переменной в FreeFem++, носящий название fespace.

Задайте пространство конечных элементов Fe1, используя построенную ранее сетку m1. Используйте линейные элементы. Для этого добавьте в полученную ранее программу следующую строку:

```
fespace Fe1(m1,P1);
```

Здесь параметр P1 определяет тип конечного элемента. В данном случае выбран линейный треугольный элемент с тремя узлами.

3.3.2. Для заданного пространства конечных элементов можно определить функцию и использовать разбиение на конечные элементы для построения графика этой функции. При построении графика функции значения в узлах будут вычислены как значения этой функции, а для остальных точек будет использоваться аппроксимация, соответствующая типу конечного элемента в рассматриваемом пространстве конечных элементов. Определите функцию $f1(x,y) = x^2 - y^2$ в пространстве конечных элементов Fe1.

Переменные x и y являются встроенными, поэтому их не нужно указывать как аргументы функции.

```
Fe1 f1=x^2-y^2;
```

3.3.3. Чтобы просмотреть график функции f1, спроецированной на пространство конечных элементов Fe1, достаточно опять воспользоваться командой plot. Добавьте ее в программу.

```
plot(f1);
```

3.3.4. *Сохраните программу под именем exl5_4.edp*. Запустите ее. После двух ранее построенных графиков, построится контурный график функции. Обратите внимание, что график построился только для той области, в которой определены элементы, т. е. для круга.

3.3.5. Изменяя параметры команды plot, можно изменить вид графика функции.

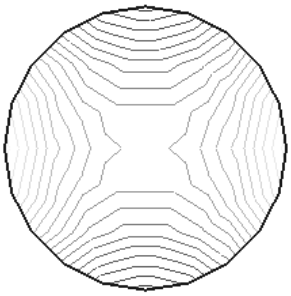


Рис. 3.17 – Контурный график функции

Параметр fill позволяет заливать пространство между контурными линиями, а параметр value позволяет выводить таблицу соответствия цветов и значений функции. Параметр nbiso определяет число контурных линий (20 по умолчанию). Измените последнюю команду plot следующим образом.

```
plot(f1,fill=1,value=1,nbiso=25);
```

Сохраните файл и запустите его.

Построится график с заливкой, 25 контурных цветов (рис. 3.18).

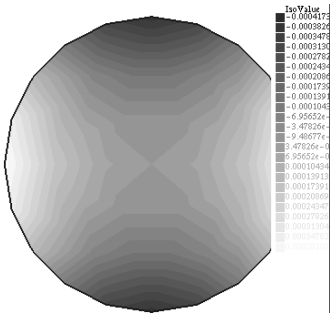


Рис. 3.18 – График с заливкой

3.3.6. Для более плавного перехода между элементами измените тип конечного элемента в команде создания пространства конечных элементов на

P2. Это квадратичный элемент, использующий аппроксимацию многочленом второго порядка, и имеющий 8 узлов.

fespace Fe1(m1,P2);

3.3.7. *Сохраните и запустите программу*.

Посмотрите, как изменился график проекции функции на пространство конечных элементов (рис. 3.19). Он стал более гладким, так как для аппроксимации используется функция второго порядка.

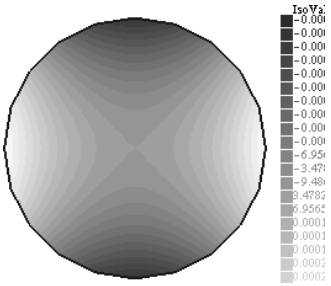


Рис. 3.19 – Конечный график

3.3.8. Для экспорта рисунка в файл формата *.eps[14]

Введите имя этого файла в параметр ps функции plot.

```
plot(f1,fill=1,value=1,nbiso=25,ps="ex15_4.eps");
```

3.3.9. *Сохраните программу и запустите ее. Проверьте, что графический файл сохранился.*

3.3.10. Для просмотра графических файлов в формате *.eps могут использоваться различные коммерческие и некоммерческие продукты. Файлы в этом формате можно вставить в приложения Microsoft Office. Для просмотра файлов в формате *.eps можно использовать бесплатное приложение XnView. Возможно, также потребуется установить скрипт Ghostscript.

3.3.11. *Закройте открытые окна.* ***Файлы ex15_4.eps и ex15_4.edp необходимо предъявить преподавателю для получения допуска к выполнению лабораторной работы***.

3.3.12. *Для желающих предлагается проверить, как изменится график функции при уменьшении или увеличении числа элементов в сетке (за счет изменения разбиения границ области)*.

4 Моделирование процессов теплопроводности в среде FreeFem++

4.1 Теплопроводность без источников теплоты

Описание задачи

Необходимо найти распределение температур в двумерной области, приведенной на рисунке 3.20. Правая и левая границы области теплоизолированы, сверху и снизу области задана постоянная температура 600 К. В центре области задана постоянная температура 500 К. Коэффициент теплопроводности материала области составляет 200 Вт/(м·К).

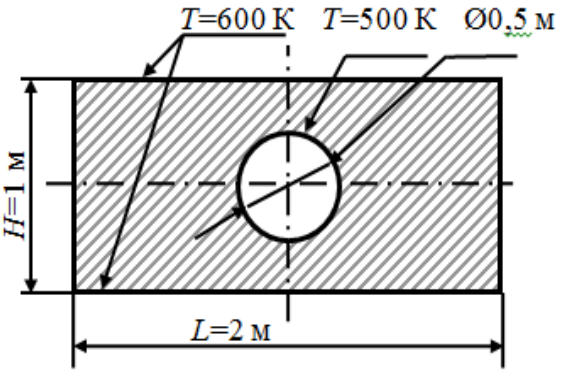


Рис. 3.20 – Двумерная область

Анализ задачи и тепловая модель

Рассматриваемая область определения имеет две линии симметрии. Следовательно, задачу можно упростить, рассматривая только часть области, ограниченную линиями симметрии. В этом случае на линиях симметрии задается граничное условие симметрии. В задачах теплопроводности граничное условие симметрии соответствует условию адиабаты или однородному граничному условию Неймана.

Моделируемая область определения изображена на рисунке 3.21.

Границы *g1*, *g2*, *g3*, *g4* представляют собой отрезки прямых. Граница *g5* – дуга окружности.

На границе *g3* задана постоянная температура 600 К, на границе *g5* – постоянная температура 500 К. На остальных границах задано условие адиабаты.

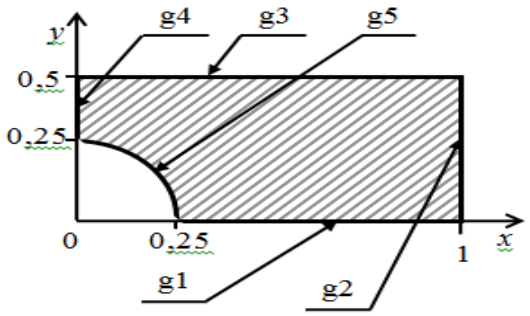


Рис. 3.21 – Моделируемая область определения

Решение задачи

4.1.1 *Создайте новый файл программы для среды FreeFem++.*

4.1.2. Вначале задайте границу области определения. Направление изменения параметра *t* выбрано так, чтобы область определения лежала слева по направлению изменения параметра. Добавьте в программу следующие строки

```
border g1(t=0.25,1){x=t;y=0;} //Граница g1
border g2(t=0,0.5){x=1;y=t;} //Граница g2
border g3(t=1,0){x=t;y=0.5;} //Граница g3
border g4(t=0.5,0.25){x=0;y=t;} //Граница g4
//Граница g5
border g5(t=pi/2,0){x=0.25*cos(t);y=0.25*sin(t);}
//Вывод границ на экран
plot(g1(10)+g2(10)+g3(10)+g4(10)+g5(10),grey=1);
```

4.1.3. *Сохраните программу под именем* ex5_5.edp *и запустите ее*. Результат ее работы (построенные границы области определения) приведен на рисунке 3.22.

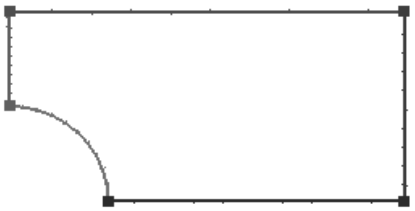


Рис.3.22 – Построение границы области определения

4.1.4. Постройте сетку и задайте пространство конечных элементов. Для этого воспользуйтесь линейным элементом P1, который рекомендуется для решения большинства задач теплопроводности.

Число разбиений границ выбирается так, чтобы получить сетку с неискаженной формой элементов. Добавьте в программу следующие строки.

```
//Создание сетки
mesh
m1=buildmesh(g1(20)+g2(10)+g3(20)+g4(10)+g5(10));
plot(m1); //Вывод сетки на экран
//Создание пространства конечных элементов
fespace fe1(m1,P1);
```

4.1.5 Сохраните программу под именем ex5_6.edp и запустите ее. Полученная сетка приведена на рисунке 3.23.

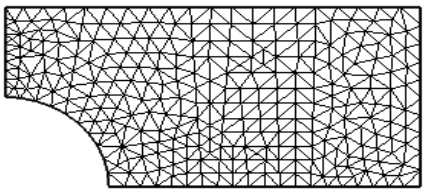


Рис. 3.23 – Результат построения сетки

4.1.6. Теперь задайте несколько переменных, определяющих коэффициенты уравнения. Коэффициенты можно задать и сразу в уравнении, но использование переменных делает программу более понятной.

Задайте переменные:

T1=600 – температура на верхней и нижней границе, T2=500 – температура в центре области, lam=200 – коэффициент теплопроводности. Также задайте функции t и phi, определенные в пространстве конечных элементов fe1. Функция t будет соответствовать искомой температуре, а функция phi – функции формы в методе конечных элементов.

```
real T1=600;//Температура на верхней и нижней
границе
real T2=500; //Температура в центре области
real lam=200; //Коэффициент теплопроводности области
fe1 t,phi;//Функции в пространстве конечных
элементов
```

4.1.7. Сохраните программу под именем ex5_7.edp, запустите ее для проверки.

4.1.8. Теперь можно перейти к заданию вариационной формулировки задачи. В данном случае удобно сразу запустить решение. Условие адиабаты на границах задавать нет необходимости, оно прикладывается автоматически. Необходимо задать соответствующее условие Дирихле на границах g3 и g5. Правила задания вариационной формулировки описаны в пункте 2.4. Добавьте следующие строки в программу:

```
//Задание и решение уравнения
solve conduct(t,phi)=
int2d(m1)(lam*(dx(t)*dx(phi)+dy(t)*dy(phi)))
+on(g3,t=T1)+on(g5,t=T2);
//Построение графика с залитыми
```

```
//контурами,10 контуров
plot(t,fill=1,value=1,nbiso=10);
```

4.1.9. *Сохраните файл (под прежним именем) и запустите его.* Построится найденное распределение температур (рис. 3.24).

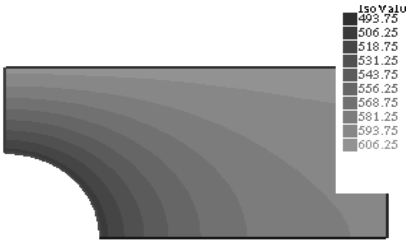


Рис. 3.24 – Распределение температур

4.1.10. Теперь можно вывести значение температуры в любой точке области определения. Достаточно лишь знать ее координаты. Например, чтобы вывести температуру в центре правой стороны прямоугольника, нужно указать его координаты: x=1, y=0.

Добавьте следующую строку в программу.

```
//Вывод значения температуры в точке с координатами (1,0)
cout << "Temperature " << t(1,0) << endl;
```

4.1.11. *Сохраните программу и запустите ее.* После запуска в окне вывода выведется значение температуры, как показано на рисунке 3.25.

```
--  mesh:  Nb of Triangles =    560, Nb of Vertices 316
-- Solve :          min 500  max 600
Temperature 582.061
times: compile 0.078s, execution 0.297s,  mpirank:0
```

Рис. 3.25 – Вывод значения температуры

Предлагается самостоятельно просмотреть значения температуры в других точках. ***Файл ex5_7.edp необходимо предъявить преподавателю для получения допуска к выполнению лабораторной работы.***

4.2 Теплопроводность с источником теплоты

Описание задачи

По медной шине, имеющей прямоугольное поперечное сечение A=5x20 мм, протекает постоянный электрический ток I=500 А. Шина имеет фторопластовую изоляцию толщиной 1,5 мм. Шина свободно охлаждается в воздухе с температурой 20°С.

Поперечное сечение шины представлено на рисунке 3.26.

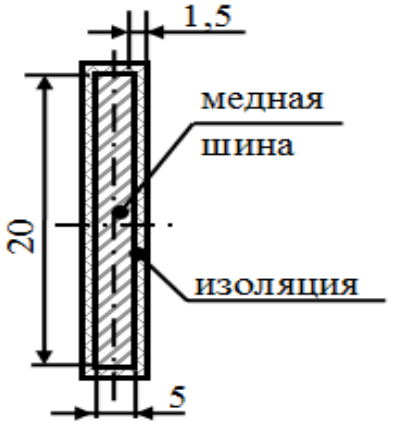


Рис. 3.26 – Поперечное сечение шины

Необходимо построить распределение температур в поперечном сечении шины с изоляцией и определить ее среднеобъемную температуру.

Коэффициент теплоотдачи с поверхности шины в окружающую среду полагать равным 10 Вт/(м^{2}К).

Анализ задачи и тепловая модель

Как и в предыдущей рассмотренной задаче, здесь имеется две линии симметрии, что позволяет упростить модель. На линиях симметрии задается условие адиабаты (границы *g1*, *g2*, *g5*, *g6* на рисунке 3.27).

На внешних границах *g3* и *g4*, являющихся внешними поверхностями изоляции шины, задается конвективный теплообмен, т. е граничное условие третьего рода (смешанное граничное условие).

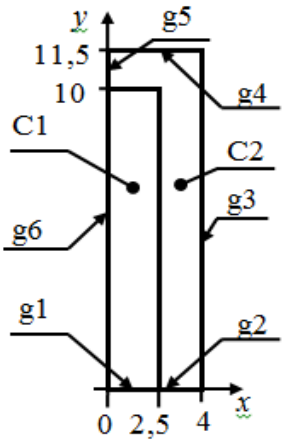


Рис. 3.27 – Условие адиабаты на границе модели

Коэффициент теплопроводности в областях *C1* (медь) и *C2* (фторопласт) различен. В области *C1* коэффициент теплопроводности 396 Вт/(м·К), а в области *C2* – 0,26 Вт/(м·К).

В области *C1* также задан объемный источник теплоты. При протекании постоянного тока можно считать, что тепловые потери распределены равномерно по объему шины.

Удельное электрическое сопротивление меди при 20°С $\rho=1{,}78\cdot10^{-8}$ Ом·м, полагаем его постоянным, не зависящим от температуры.

Тепловые потери на единицу объема шины, *W*, в данном случае можно вычислить следующим образом:

$$W=\left(\frac{I}{A}\right)^2\rho, \qquad (4.2.1)$$

где I – сила тока, A – площадь поперечного сечения шины, ρ – ее удельное электрическое сопротивление.

Для удобства также введем параметры $y1$ – длина наибольшей стороны поперечного сечения шины, $x1$ – длина наименьшей стороны поперечного сечения шины и $s2$ – толщина изоляции. Все геометрические параметры будем задавать в метрах. Тогда:

$$A = x1\cdot y1. \qquad (4.2.2)$$

Решение задачи

4.2.1. *Создайте новый файл программы для среды FreeFem++.*

4.2.2. Вначале задайте параметры, необходимые для расчетов, в виде переменных типа real (действительные числа). Здесь задаются как

геометрические параметры, так и свойства материалов, и коэффициенты в граничных условиях.

```
real y1=0.02; //Длина наибольшей стороны сечения
шины
real x1=0.005;//Длина наименьшей стороны сечения
шины
real s2=0.0015; //Толщина изоляции
real I=500; //Сила тока, А
real lam1=396; //Коэффициент теплопроводности шины
real lam2=0.26; //Коэфф. теплопроводности изоляции
real ro1=1.78E-8; //Удельное электрическое сопр.
шины
real ta=20; //Температура окружающей среды
real alf=10; //Коэффициент теплоотдачи
```

4.2.3. Задайте границы области определения. Необходимо задавать как внешние границы, так и границы раздела двух материалов. Добавьте следующие строки в программу:

```
border g1(t=0,x1/2){x=t;y=0;} //Граница g1
border g2(t=x1/2,x1/2+s2){x=t;y=0;} //Граница g2
border g3(t=0,y1/2+s2){x=x1/2+s2;y=t;} //Граница g3
border g4(t=x1/2+s2,0){x=t;y=y1/2+s2;} //Граница g4
border g5(t=y1/2+s2,y1/2){x=0;y=t;} //Граница g5
border g6(t=y1/2,0){x=0;y=t;} //Граница g6
//Внутренние границы
border gin1(t=0,y1/2){x=x1/2;y=t;}
border gin2(t=x1/2,0){x=t;y=y1/2;}
//Вывод границ на экран
plot(g1(10)+g2(3)+g3(10)+g4(10)+g5(3)+g6(10)+gin1(10
)+gin2(10),grey=1);
```

4.2.4. *Сохраните программу под именем* ***ex5_8.edp*** *и запустите ее.* Построятся границы области определения. Они приведены на рисунке 3.28.

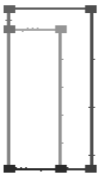


Рис. 3.28 – Границы области определения

4.2.5. Постройте сетку и задайте пространство конечных элементов.

```
//Построение сетки
mesh D=buildmesh(g1(10)+g2(8)+g3(48)+g4(18)
+g5(8)+g6(40)+gin1(40)+gin2(10));
plot(D,grey=1); //Вывод сетки на экран
//Создание пространства конечных элементов
fespace fe1(D,P1);
```

4.2.6 *Сохраните программу под прежним именем и запустите ее*. Построится сетка. Она приведена на рисунке 3.29.

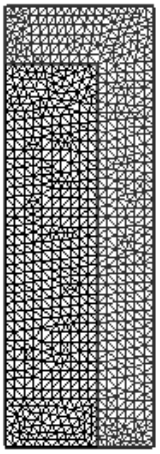


Рис. 3.29 – Построение сетки

4.2.7. Теперь необходимо задать коэффициенты в вариационной формулировке задачи, которые соответствуют физическим свойствам материалов. Некоторые свойства материалов (коэффициент теплопроводности, объемная плотность мощности) различаются для областей C1 и C2. Это различие можно представить в виде зависимости свойств от координат. Как известно, зависимость свойств от координат не делает задачу нелинейной. Однако необходимо задать соответствующие функции. Все функции должны быть определены в пространстве конечных элементов fe1.

Зададим функцию для определения коэффициента теплопроводности. Для его определения как функции координат воспользуемся тем свойством, что операции сравнения (>, <, = и т. п.) в среде FreeFem++, подобно языку программирования C++, можно использовать в арифметических выражениях. Истина соответствует значению «1», а ложь – «0».

В результате выражение, приведенное ниже, дает значение λ_1, если для координат рассматриваемой точки (x, y) одновременно выполняются неравенства $x \le \frac{x_1}{2}$ и $y \le \frac{y_1}{2}$. В противном случае выражение, приведенное ниже, дает значение λ_2.

$$\lambda = (\lambda_1 - \lambda_2)\cdot\left(x \le \frac{x}{2}\right)\cdot\left(y \le \frac{y}{2}\right) + \lambda_2 \qquad (4.2.3)$$

Выражение (4.2.3) позволяет задать для области с $x \le \frac{x_1}{2}$ и $y \le \frac{y_1}{2}$ коэффициент теплопроводности λ_1, а для остальной области – коэффициент теплопроводности λ_2. Таким образом, задаются два разных материала.

Задайте функцию lam в пространстве конечных элементов fe1, соответствующую коэффициенту теплопроводности (4.2.3). Постройте график функции, чтобы убедиться, что она задана правильно.

```
//Задание коэффициента теплопроводности как функции
//координат в пространстве fe1
fe1 lam=(lam1-lam2)*(x<=x1/2)*(y<=y1/2) +lam2;
//Вывод распределения коэффициента
//теплопроводности
plot(lam,fill=1,value=1);
```

4.2.8. *Сохраните программу и запустите ее*. Построится график функции в пространстве конечных элементов, он приведен на рисунке 3.30.

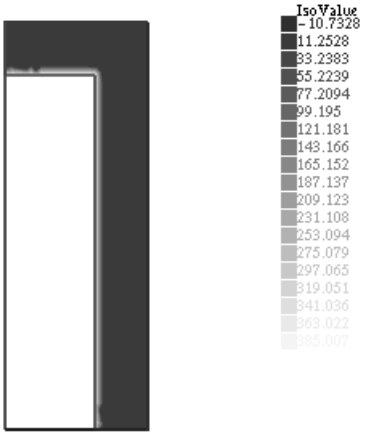


Рис. 3.30 – График функции в пространстве конечных элементов

4.2.9. Аналогичным образом задайте функцию, определяющую распределение объемной плотности мощности. При этом учитываются соотношения (4.2.1) и (4.2.2). Добавьте в программу следующее:

```
//Задание объемной плотности мощности
fe1 W=ro1*(I/x1/y1)^2*(x<=x1/2)*(y<=y1/2);
//Вывод объемной плотности мощности
plot(W,fill=1,value=1);
```

4.2.10. *Сохраните программу и запустите ее*. Построится график функции в пространстве конечных элементов.

4.2.11. Определите искомую функцию и функцию формы в пространстве конечных элементов fe1.

```
fe1 t,phi; //Задание искомой функции и функции формы
```

4.2.12. Задайте вариационную формулировку задачи и решите ее. Условия симметрии прикладываются автоматически, также автоматически задаются условия на границе раздела двух материалов. В случае если подынтегральное выражение одинаковое, можно указать сразу несколько границ при интегрировании.

```
//Задание и решение уравнения
solve cond1(t,phi)=
int2d(D)(lam*(dx(t)*dx(phi)+ dy(t)*dy(phi)))-
int2d(D)(W*phi)
+int1d(D,g3,g4)(alf*t*phi)-
int1d(D,g3,g4)(alf*ta*phi);
//Вывод графика температурного поля
plot(t,fill=1,value=1);
```

4.2.13. *Сохраните программу и запустите ее*. Построится график распределения температур, он приведен на рисунке 3.31.

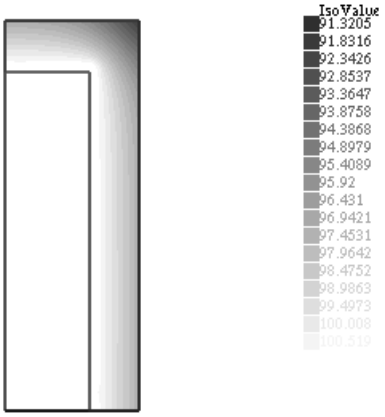


Рис. 3.31 – График распределения температур

4.2.14. Далее по условию задачи необходимо рассчитать среднеобъемную температуру медной шины. Для нахождения средней температуры t_v по объему V воспользуемся известной формулой $t_V = \frac{1}{V}\iiint\limits_V t dV$.

Эта формула в нашем случае двумерной задачи превращается в $t_V = \frac{4}{x1 \cdot y1}\iint\limits_{C1} t dx dy$, где $C1$ – область, в которой ищется среднеобъемная температура, а $x1/2 \cdot y1/2$ – площадь поперечного сечения области.

Для выделения из области D области $C1$ воспользуемся тем же приемом, что и в п. 4.2.7. Добавьте в программу следующие строки:

```
real tv; //Задание переменной
//Вычисление среднеобъемной температуры
tv=4/x1/y1*int2d(D)(t*(x<=x1/2)*(y<=y1/2));
//Вывод температуры на экран
cout << "Average bus rod temperature " << tv <<endl;
```

4.2.15. *Сохраните программу и запустите ее*. Значение температуры $t_v = 100{,}8°C$ выведется в окне вывода (рис. 3.32).

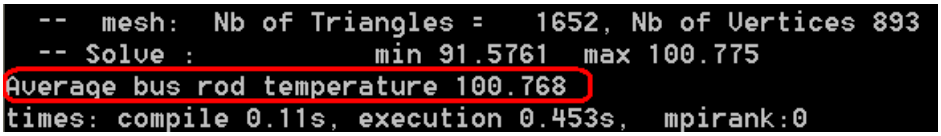


Рис. 3.32 – Вывод значения температуры

4.2.16. Аналогичным образом, меняя поверхностный интеграл на интеграл по границе области, можно найти среднеповерхностную температуру.

В данном случае средняя температура поверхности изоляции $t_s = \frac{1}{x1/2 + y1/2 + 2 \cdot s2} \int_{g3+g4} t d\gamma$. Добавьте в программу следующие строки:

```
real ts; //Задание переменной
//Вычисление среднеповерхностной температуры
ts=1/(x1/2+y1/2+2*s2)*int1d(D,g3,g4)(t);
//Вывод результатов расчетов
cout << "Average surface temperature " << ts <<
endl;
```

4.2.17. Сохраните программу и запустите ее. Значение температуры $t_v = 96.2°C$ выведется в окне вывода (рис. 3.33). Задача решена. ***Файл ex5_8.edp необходимо предъявить преподавателю для получения допуска к выполнению лабораторной работы.***

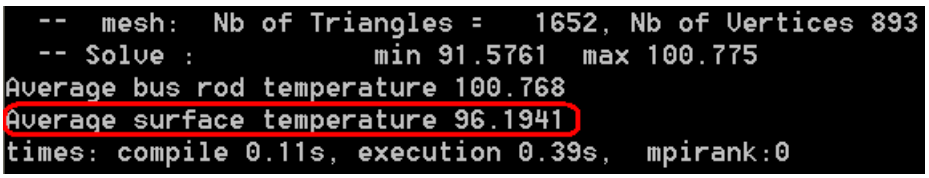


Рис. 3.33 – Вывод значения температуры

5 Требования для получения допуска

Для получения допуска к выполнению лабораторной работы необходимо предъявить преподавателю файлы со следующим содержимым, требования к отдельным файлам описаны выше:

1. Окончательный файл программы для среды FreeFem++, выполняющий все действия по построению области определения, описанные в части 3 методических указаний (exl5_4.edp).
2. Графический файл, получающийся при работе программы, описанной в предыдущем пункте (exl5_4.eps).
3. Окончательный файл программы для среды FreeFem++, выполняющий все действия по решению задачи теплопроводности без источника теплоты, описанные в п. 4.1. (exl5_7.edp).
4. Окончательный файл программы для среды FreeFem++, выполняющий все действия по решению задачи теплопроводности с источником теплоты, описанные в п. 4.2. (exl5_8.edp).

Все файлы должны быть помещены в каталог.

5.1 Приложение. Работа со средой FreeFem++ без редакторов программного кода и GUI

Если на компьютере не установлены никакие редакторы программного кода и графический интерфейс для среды FreeFem++, то любую программу, описанную в методических указаниях можно запустить следующим образом:

1. Наберите текст программы в «Блокноте» или ином редакторе, поддерживающим формат *.txt (обычный текст). Сохраните полученный файл с расширением *.edp.

2. Запустите компилятор FreeFem++ и выберите сохраненный ранее файл. Программа запустится.

6 Контрольное задание к лабораторной работе № 5

6.1 Информация о задании

В данной лабораторной работе необходимо решить задачу теплопроводности, но нет необходимости составления тепловой модели, так как она приведена в задании практически в готовом виде. В силу этого задача не должна представлять проблемы для студентов нетеплофизических специальностей. Однако, как всегда в подобных случаях, предлагаются альтернативы данному заданию. Они приведены в пункте «Дополнительные и альтернативные задания».

6.2 Описание задания

Необходимо составить программу для среды FreeFem++, в которой будет производиться расчет стационарного температурного поля в соответствии с тепловой моделью, приведенной в следующем пункте задания.

Программа должна удовлетворять следующим условиям:

- Должна выводиться на экран сетка конечных элементов, используемая для решения задачи. Построенная сетка должна быть приемлемой для рассматриваемой задачи.
- Должно выводиться на экран в виде графика температурное поле, полученное в результате решения задачи.
- Полученный график температурного поля должен сохраняться средствами самой программы в графическом файле.
- На экран также должно выводиться значение среднеобъемной температуры рассматриваемого тела.
- Все строки должны содержать соответствующие комментарии. Первый комментарий должен содержать номер лабораторной работы, номер варианта и фамилию автора или авторов.

Для разработки программы можно использовать как графический интерфейс, так и командную строку. Можно использовать альтернативные

оболочки FreeFem++. Однако программа должна быть работоспособна на программном обеспечении, установленном в компьютерном классе.

По результатам выполнения работы должен быть составлен **отчет.**

6.3 Тепловая модель

В этом пункте приводится практически полная тепловая модель для задачи, однако в ходе решения задачи разрешены ее модификации, влияющие только на вычислительную погрешность (такие как упрощение геометрической модели: учет симметрии, изменение системы координат и пр.). Некоторые модификации могут упростить решение задачи, однако их применение не является обязательным для **данной** лабораторной работы.

Требуется определить **стационарное** температурное поле двумерной области, представляющей собой поперечное сечение длинной балки. Геометрия области приведена на рис. 1 – 25 в зависимости от варианта задания, область заштрихована. Размеры на рисунках приведены **в миллиметрах**. Материал балки приведен в табл. 1 в зависимости от варианта задания. Требующиеся свойства материалов необходимо определить самостоятельно.

В области происходит выделение теплоты с объемной плотностью мощности W, указанной в табл. 6 в зависимости от варианта. В качестве граничного условия на внешних границах области задается условие конвективного теплообмена – теплоотдача в среду с температурой t_a и коэффициентом теплоотдачи α, эти параметры приведены в табл. 6 в зависимости от варианта задания. На внутренних границах области в тех вариантах, где они есть, задается условие адиабаты.

При решении задачу можно считать **линейной.**

6.4 Рекомендации по выполнению работы

Рекомендуется выполнять данную работу в указанном порядке, хотя отклонения от него вполне допустимы:

1. Определите недостающие коэффициенты тепловой модели (свойства материалов). Выберите удобное расположение системы координат. Подумайте, нельзя ли как-то упростить модель. Если можно, то упростите ее и опишите модифицированную модель в отчете.

2. Разработайте и опишите в отчете математическую модель задачи. Проведите необходимые математические преобразования выражений к стандартному виду. За основу рекомендуется взять п. 2.4 методических указаний. Сюда же необходимо включить математическое описание области определения. Его рекомендуется составлять следующим образом:

2.1. Мысленно, на бумаге или при помощи компьютера разделите границу рассматриваемой области на простейшие линии: отрезки прямой линии, окружности и дуги окружностей.

2.2. Составьте уравнения, описывающие параметрически каждую из полученных кривых. Используйте теоретические сведения в методических указаниях и лекциях.

2.3. Далее необходимо найти диапазон изменения параметра для каждой кривой. Иногда этот диапазон очевиден. В противном случае необходимо найти точки пересечения каждой из пар соседних кривых. Это можно сделать, решив систему уравнений для каждой из пар кривых. Система уравнений получается путем приравнивания соответствующих координат *x* и *y*, выраженных параметрически, через уравнения, полученные в пункте 2.2. Результатом решения системы уравнений будут два значения параметра для двух рассмотренных кривых. Таким образом, можно найти минимальное и максимальное значение параметра для каждой кривой. Для решения уравнений можно использовать и компьютер, в том числе и саму среду FreeFem++, вставляя предварительные вычисления в основную программу.

2.4. Определите, в какую сторону (возрастание, убывание) должен изменяться параметр для каждой линии.

3. Используя модель, полученную в п. 2, составьте программу для среды FreeFem++. Начните с задания исходных данных и области определения. Затем постройте сетку, выведите ее на экран. Задайте вариационную формулировку задачи и решите ее.

4. Далее необходимо обработать результаты расчета. Включите в программу для среды FreeFem++ операторы для построения графика температурного поля. График сохраните в файл и вставьте в отчет. Включите в программу операторы для расчета среднеобъемной температуры и вставьте результаты в отчет.

Также следует обратить внимание на следующие моменты:

- Обращайте внимание на соответствие размерностей физических величин. Наибольшее число ошибок возникает именно из-за этой проблемы.
- Не забудьте указать граничные условия на всех границах. Граничное условие адиабаты прикладывается автоматически.
- Обратите внимание, что площадь области может быть рассчитана средствами среды FreeFem++, а не вручную.
- Обратите внимание нао выбор расположения системы координат.

Постарайтесь выбрать расположение системы координат так, чтобы упростить математическое описание границ области.

6.5 Форма отчета

Перед выполнением задания **необходимо получить допуск** по результатам выполнения заданий в методических указаниях.

Отчет может быть представлен в печатном или электронном виде. Отчеты, написанные «от руки», не принимаются. В электронном виде отчет может быть представлен в виде файла в формате *.doc (должен читаться в Office 2003), *.rtf или *.pdf. По согласованию с преподавателем возможно представление отчета в другом виде. Дополнительно должен быть представлен файл(ы) с программой для среды FreeFem++, осуществляющей решение задачи. Все файлы, включая файл отчета, если он представляется в электронном виде, должны быть помещены в отдельный каталог.

Отчет должен содержать следующие сведения:

1. ФИО студента или студентов, выполнявших работу, номер группы, номер задания, вариант задания.

2. Тепловая модель задачи (описание геометрии задачи, включая расположение системы координат, свойств материалов, граничных условий **в текстовой, графической или табличной форме**). Описывается только конечная тепловая модель, используемая в решении с обоснованием принятых дополнительных допущений (если проводилась модификация модели). При указании физических свойств материалов необходимо указать источник(и), из которого взяты свойства.

3. Математическая модель задачи (вариационная формулировка уравнения теплопроводности для рассматриваемого случая с соответствующими граничными условиями, математическое описание области определения в виде уравнений или неравенств).

4. Результаты расчетов (полученный график температурного поля, результаты расчетов среднеобъемной температуры).

Отчет может содержать и другие сведения, например, результаты расчетов при выполнении дополнительных заданий, описанных далее. Эти сведения являются необязательными. Остальные сведения являются обязательными, и их непредставление ведет к снижению оценки за работу.

Файл программы для FreeFem++ должен удовлетворять условиям, приведенным в п. 2.

При защите лабораторной работы могут быть заданы вопросы по любому пункту отчета или тексту программы. Также при защите лабораторной работы должна быть продемонстрирована работоспособность программы, в противном случае задание не засчитывается. Должно наблюдаться соответствие результатов, выдаваемых программой, и результатов, представленных в отчете. Результаты **должны совпасть с правильными ответами.**

6.6 Критерии оценки

Следующие погрешности в выполнении работы приводят к снижению оценки за нее:

- отсутствие комментариев в тексте программы или их несоответствие тексту программы;

- отсутствие ссылок на литературу в тех случаях, когда это требуется;

- отсутствие отчета и предоставление только программы;

- отсутствие в отчете тепловой или математической модели (за каждую);

- погрешности в составлении тепловой или математической модели, не влияющие на результаты работы;

-не выводится график температурного поля или среднеобъемная температура (за каждый);

- отсутствие в отчете графика температурного поля, результатов расчетов среднеобъемной температуры (за каждый), при условии наличия их в программе;

- отсутствие в программе вывода сетки конечных элементов;

- неразумное разбиение сеткой области, приводящее к искажению результатов расчетов;

6.7 Варианты задания

Варианты задания приведены на рисунках 1 – 25 и в таблице 6, обозначения даны в п. 3.

Таблица 6 Исходные данные

№ вар.	материал	W, кВт/м^2	t_a, °C	α, Вт/(м^2·К)
1	сталь 12Х18Н10Т	18	5	9
2	константан	60	10	10
3	манганин	30	15	11
4	нихром	45	20	12
5	сплав ВТ1	18	25	13
6	сплав ХН60Ю	15	30	14
7	сталь 12Х18Н10Т	25	35	15
8	константан	20	40	16
9	манганин	9	45	17
10	нихром	20	40	15
11	сплав ВТ1	50	35	13
12	сплав ХН60Ю	25	30	11
13	сталь 12Х18Н10Т	30	25	9
14	константан	23	20	7
15	манганин	20	15	30
16	нихром	80	10	35
17	сплав ВТ1	80	5	40
18	сплав ХН60Ю	25	20	25
19	сталь 12Х18Н10Т	55	25	20
20	константан	35	30	18
21	манганин	15	35	16
22	нихром	40	40	14

23	сплав ВТ1	15	45	12
24	сплав ХН60Ю	10	50	10
25	манганин	1,5	55	8

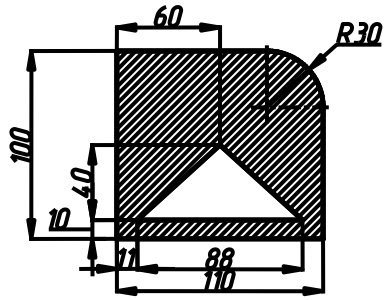


Рисунок 1 Вариант 1

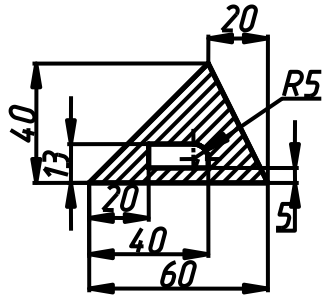


Рисунок 2 Вариант 2

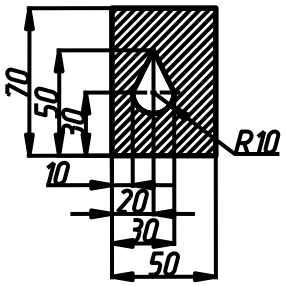


Рисунок 3 Вариант 3

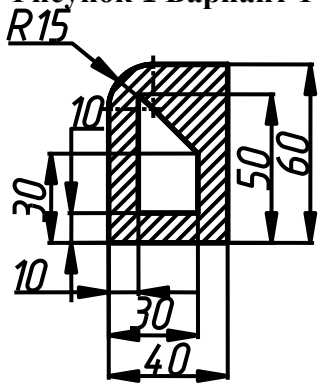


Рисунок 4 Вариант 4

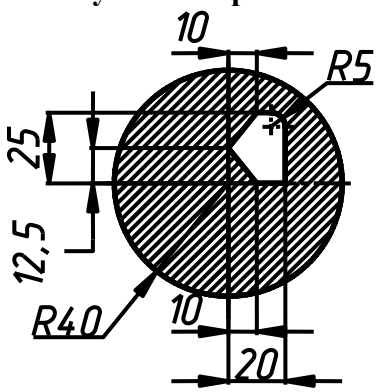


Рисунок 5 Вариант 5

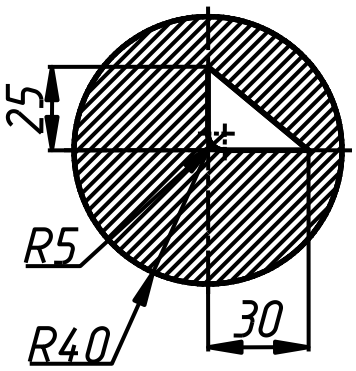


Рисунок 6 Вариант 6

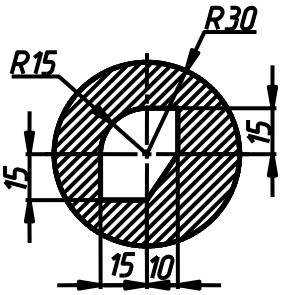


Рисунок 7 Вариант 7

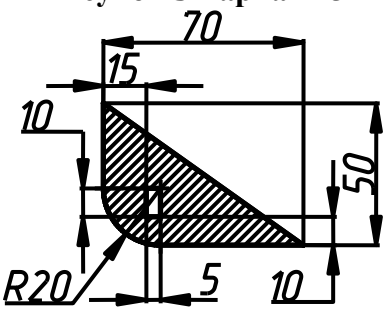


Рисунок 8 Вариант 8

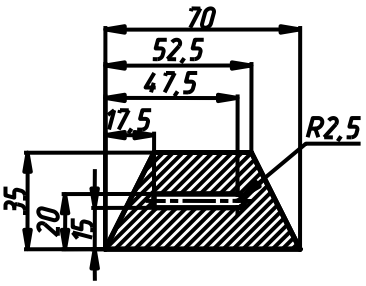


Рисунок 9 Вариант 9

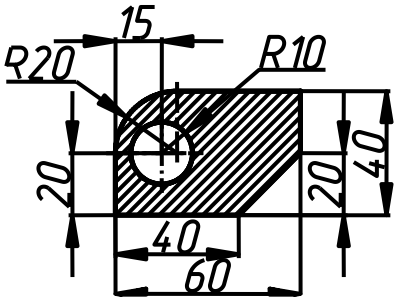


Рисунок 10 Вариант 10

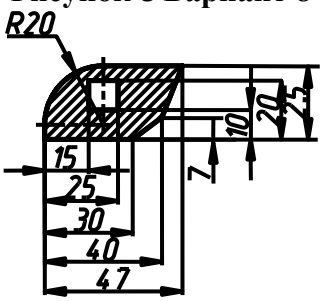


Рисунок 11 Вариант 11

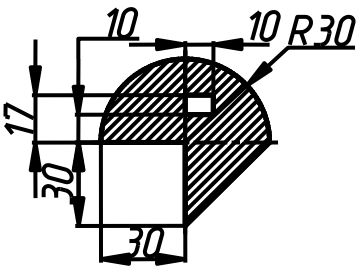


Рисунок 12 Вариант 12

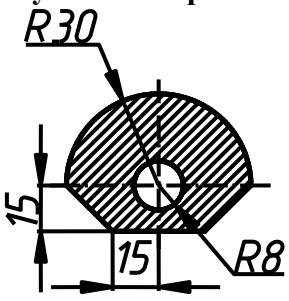


Рисунок 13 Вариант 13

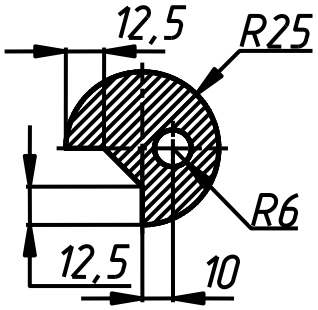


Рисунок 14 Вариант 14

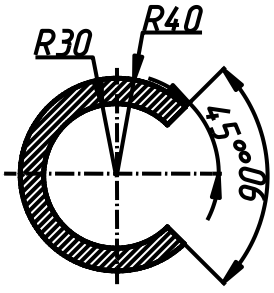


Рисунок 15 Вариант 15

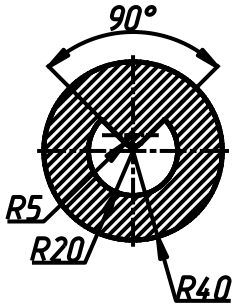


Рисунок 16 Вариант 16

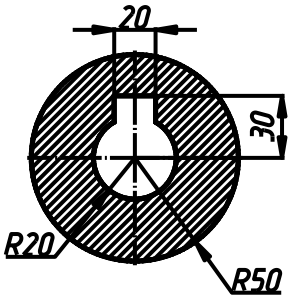


Рисунок 17 Вариант 17

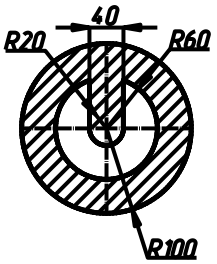


Рисунок 18 Вариант 18

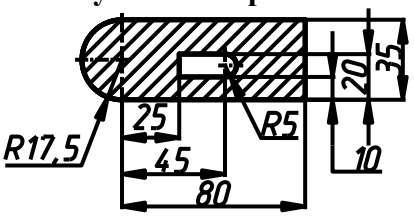


Рисунок 19 Вариант 19

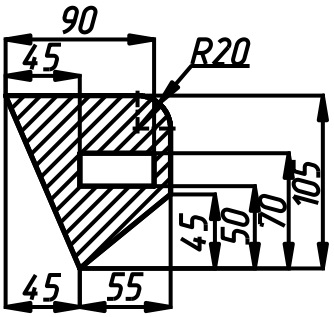


Рисунок 20 Вариант 20

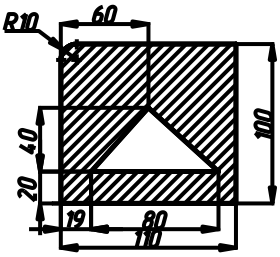


Рисунок 21 Вариант 21

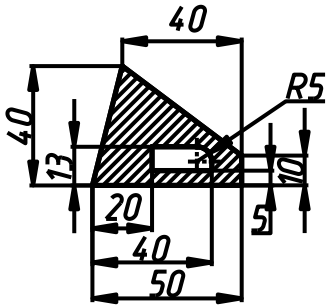


Рисунок 22 Вариант 22

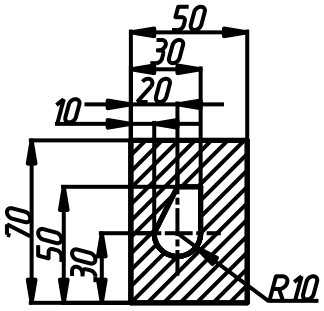


Рисунок 23 Вариант 23

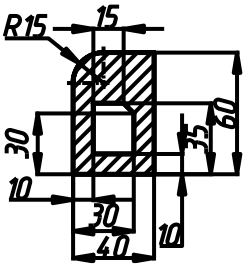


Рисунок 24 Вариант 24

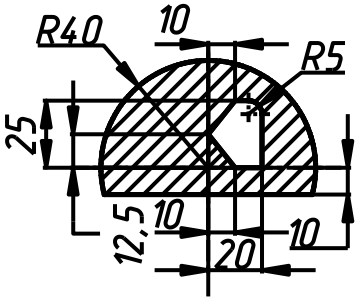


Рисунок 25 Вариант 25

СПИСОК ЛИТЕРАТУРЫ

1) Сабитов, К.Б. Уравнения математической физики [Электронный ресурс] : учеб. — Электрон. дан. — Москва : Физматлит, 2013. — 352 с. — Режим доступа: https://e.lanbook.com/book/59660. — Загл. с экрана.

2) Треногин, В.А. Уравнения в частных производных [Электронный ресурс] : учеб. пособие / В.А. Треногин, И.С. Недосекина. — Электрон. дан. — Москва : Физматлит, 2013. — 228 с. — Режим доступа: https://e.lanbook.com/book/59744. — Загл. с экрана.

3) Ковеня В. М., Чирков Д. В. Методы конечных разностей и конечных объемов для решения задач математической физики. Учебное пособие. – Новосибирск: НГУ, 2013. – 86 с.

4) Горлач, Б.А. Математическое моделирование. Построение моделей и численная реализация [Электронный ресурс] : учеб. пособие / Б.А. Горлач, В.Г. Шахов. — Электрон. дан. — Санкт-Петербург : Лань, 2018. — 292 с. — Режим доступа: https://e.lanbook.com/book/103190. — Загл. с экрана.

1.Эльсгольц Л. Э. Дифференциальные уравнения и вариационное исчисление. – Изд. 4-е. – М.: Едиториал УРСС, 2000. – 320 с.

2. Ахмеров Р. Р. Численные методы решения ОДУ [электронный ресурс]. – Электрон. учебник. – Новосибирск, 2005. – Режим доступа: http://www.sbras.ru/rus/textbooks/akhmerov/

3. Ануфриев И. Е. Самоучитель MatLab 5.3/6.x. – СПб.: БХВ-Петербург, 2002. – 710 с.

4. 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. – 272 с.

5. Захарова В. Ю. Компьютерные технологии в науке и образовании. Программное обеспечение для решения задач технической физики.— СПб.: СПбГУ ИТМО, 2010 .— 78 с.

6. Hecht F. Freefem++ [electronic resource] /F.Hecht, O. Pironneau, J. Morice, A. Le Hyaric, K. Ohtsuka. – Third Edition, Version 3.20. – Electronic book (390 p.). – Paris, 2012. – Mode of access: http://www.freefem.org/ff++/index.htm.

7. Жуков М. Ю., Ширяева Е. В. Использование пакета конечных элементов FreeFem++ для задач гидродинамики, электрофореза и биологии. – Ростов н/Д: Изд-во ЮФУ. 2008. – 256 с.

8. Самарский А. А. Введение в численные методы. – М.: Наука, 1978. – 269 с.

9. Флетчер К. Численные методы на основе метода Галеркина: Пер. с англ. – М.: Мир, 1988. – 352 с.

10. Ли К. Основы САПР (CAD/CAM/CAE). – СПб.: Питер, 2004. – 560 с.

11. Норри Д., Фриз Ж. Введение в метод конечных элементов. – М.: Мир, 1981. – 304 с.

12. Зенкевич О. Метод конечных элементов в технике: Пер. с англ. – М.: Мир, 1975. – 544 с.

13. Коннор Дж., Бреббиа К. Метод конечных элементов в механике жидкости: Пер. с англ. – Л.: Судостроение, 1979. – 264 с.

14. Дульнев Г. Н., Парфенов В. Г., Сигалов А. В. Методы расчета теплового режима приборов. – М.: Радио и связь, 1990. – 312 с.

Захарова Виктория Юрьевна, Волкович Мария Игоревна

Методические указания к выполнению лабораторных работ по курсу "Математическое моделирование физических процессов". Часть 2

Учебно-методическое пособие

В авторской редакции
Редакционно-издательский отдел Университета ИТМО
Зав. РИО Н.Ф. Гусарова
Подписано к печати
Заказ №
Тираж
Отпечатано на ризографе

Редакционно-издательский отдел
Университета ИТМО
197101, Санкт-Петербург, Кронверкский пр., 49, литер А