МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

Высшего образования

«Нижегородский государственный университет им. Н.И. Лобачевского»

Национальный исследовательский университет

**ОТЧЕТ**

**о проделанной работе по курсу «Численные методы» на тему**

**«Качественное исследование уравнения теплопроводности»**

Подготовили учащиеся гр. 381906-3 Олюнин А., Шабунин В., Михеев И., Сорокина Е., Тюлькина О.

Проверил преподаватель кафедры

Дифференциальных уравнений, математического и численного анализа Эгамов А. И.

2022 год

Содержание:

[1. Введение 3](#_Toc103772179)

[2. Постановка задачи 4](#_Toc103772180)

[3. Описание физического процесса 5](#_Toc103772181)

[4. Аналитическое решение уравнения теплопроводности при помощи метода Фурье 7](#_Toc103772182)

[5. 10](#_Toc103772183)

[6. Разностная аппроксимация уравнения теплопроводности при помощи неявной схемы 10](#_Toc103772184)

[6. Метод прогонки 13](#_Toc103772185)

[7. Руководство пользователя 15](#_Toc103772186)

[8. Вывод 18](#_Toc103772187)

[9. Список литературы и источников информации 20](#_Toc103772188)

[10. Приложение. Код программы. 21](#_Toc103772189)

1. Введение

Большинство реальных систем можно описывать, по крайней мере в известном приближении, в виде динамических систем. Динамической системой принято называть такую систему, для которой можно указать некоторый набор величин, называемый набором динамических переменных и полностью характеризующий состояние этой системы в любой последующий момент времени, и задан закон, называемый законом эволюции, и описывающий эволюцию начального состояния с течением времени. На основе динамических систем строится модель - упрощенное представление действительности в какой-либо форме, повторяющей основные характеристики реального объекта и имитирующей его поведение. В общем виде система дифференциальных уравнений принимает вид:

где – вектор состояния, – оператор эволюции.

Стоит отметить, что в современном представлении, понятие оператора эволюции вышло за рамки ассоциации только с дифференциальными уравнениями, теперь это более широкое понятие, включающее в том числе, например, точечные отображения.

Как известно, теория обыкновенных дифференциальных уравнений начала развиваться еще давно, сразу, после открытия дифференциального и интегрального исчисления. Более же сложная теория дифференциальных уравнений с частными производными начала свое развитие гораздо позже и в виде конкретных уравнений, которые находили свое отражение в физических процессах. Поэтому в современной математической физике уравнения с частными производными и способы их решения занимают главенствующую роль. Для анализа таких уравнений нередко применяют аналитические методы решения. Например, метод разделения переменных, также известный как метод Фурье, является одним из наиболее распространенных методов решения уравнений с частными производными, предполагающий нахождение аналитического решения уравнения. Но нахождение решения в таком виде может стать серьезной проблемой, особенно в таких системах, как нелинейные системы или системы с переменными коэффициентами.

Поэтому, нередко для изучения дифференциальных уравнений с частными производными используется аппарат нахождения численного решения. Тут стоит выделить метод конечных разностей. Он демонстрирует свою эффективность при решении уравнений в частных производных гиперболического и эллиптического вида. Кроме того, данный метод можно применять к уравнениям параболического типа, одним из которых является уравнение теплопроводности, подробно рассматривающийся в этой работе.

2. Постановка задачи

В данной лабораторной работе стоит задача рассмотреть одно из дифференциальных уравнений в частных производных параболического типа – уравнение теплопроводности. Для этой цели применить метод Фурье, получить аналитическое решение, и метод конечных разностей, а именно неявную схему, получить численное решение. При исследовании уравнения учесть начальные условия и наложенные на функции ограничения. Рассмотреть условия устойчивости численного решения неявной схемы. Оформить численное решение в виде программы на языке высокого уровня с дружеским интерфейсом.

3. Описание физического процесса

Рассмотрим одно из главных уравнений параболического типа - уравнение теплопроводности. В общем смысле оно описывает физический процесс распределения тепла в различных телах (в нашем случае стержень). Причем, в разных задачах он может иметь разную форму, но мы ограничимся рассмотрением тонкого стержня. Это означает, что его можно представить в виде одномерного объекта. Тогда и уравнение теплопроводности будет являться линейным одномерным по пространству уравнением в частных производных параболического типа.

Отметим, что уравнение теплопроводности является уравнением параболического вида, потому как при рассмотрении общего вида линейных уравнений относительно старших производных, записывающихся в общем виде следующим образом:

оно удовлетворяет условию: точке этого уравнения выполняется:

Перейдем к рассмотрению самого уравнения теплопроводности. В неоднородном виде оно записывается, как:

где – функция распределения тепла по стержню, значение функции соответствует значению температуры стержня в точке , в момент времени . Непрерывная функция – функция управления обратной связью. – температуропроводность стержня, константа, в качестве ее значения принято брать , не будем далеко уходить от этого правила, но учтем возможность ее изменения в программе.

Переменные принято рассматривать в следующих отрезках: , где – длина стержня, – конечное время рассмотрения нагревания.

Для того, чтобы уравнение теплопроводности имело единственное решение достаточно задать три условия. Сначала зададим начальное распределение температуры по стержню. Представим его в виде дважды дифференцируемой на отрезке функции:

Причем, на функцию наложим дополнительное условие:

Еще два условия возьмем из условия теплоизолированности концов стержня. Оно задается в виде однограничных условий первого рода, и называется условием Неймана:

Теперь условия заданы, следовательно рассматриваемая нами задача имеет единственное решение.

Дополним задачу еще несколькими ограничениями.

Во-первых, так как далее мы будем приводить аналитическое решение с помощью метода Фурье, то, для того чтобы оно существовало достаточно наложить на функцию условие непрерывной дифференцируемости по и дважды непрерывной дифференцируемости по .

Во-вторых, на функцию обратной связи необходимо наложить такое условие, чтобы выполнялось:

И при этом функцию необходимо рассмотреть в двух видах:

.

где – управляющая функция.

4. Аналитическое решение уравнения теплопроводности при помощи метода Фурье

Рассмотрим однородное уравнение теплопроводности для тонкого стержня:

где переменная находится в отрезке . Переменная . А также задано начальное условие:

И два граничных условия:

Метод Фурье заключается в нахождении суммы частных решений дифференциального уравнения путем разделения переменных. Эта сумма также будет являться решением уравнения. Таким образом, будем искать частное решение уравнения в виде:

В таком случае имеем систему:

которую подставляем в исходное однородное уравнение, тем самым получая:

Переписав данное уравнение в виде:

понимаем, что, так как частные производные берутся по разным независимым друг от друга переменным, то такое равенство возможно только в случае равенства констант. Введем обозначение для этой константы.

Далее следует система уравнений:

*–* из граничных условий

Вплоть до кратных постоянных единственное решение краевой задачи для *:*

Тогда:

Следовательно, должна удовлетворять уравнению:

Отсюда имеем:

где – коэффициенты ряда Фурье функции . Таким образом, мы получили собственные функции, перемножив которые, можно получить собственные колебания уравнения теплопроводности:

Применяя принцип суперпозиции, получаем ряд:

Подставляя начальное условие, имеем:

Полученное выражение представляет собой разложение функции в синусоидальный ряд. Следовательно:

Получаем, что если известно разложение функции в синусоидальный ряд, то можно получить решение простым включением экспоненты. Также, несложно увидеть, что такое решение будет единственно. Выявим характерные для данного уравнения особенности:

* Собственные колебания системы экспоненциально стремятся к нулю при .
* Температура экспоненциально стремится к 0 в каждой точке при .
* При повышении параметра температуропроводности температура стержня стремится к 0 еще быстрее.

Таким образом, однородное уравнение теплопроводности можно решить аналитически с помощью метода Фурье. Однако получить такое же решение для неоднородного уравнения теплопроводности гораздо труднее. Поэтому далее мы рассмотрим метод конечных разностей – численный метод, позволяющий упростить анализ неоднородного уравнения теплопроводности.

5. Рассмотрение специальных условий краевой задачи

Начальное условие. Рассмотрим условие, которое нужно наложить на начальное распределение температуры по стержню.

Возьмем в качестве функции следующую функцию:

где – длина стержня, – некоторые константы, для любого значения которых функция удовлетворяет краевым условиям. Тогда в силу ортогональности косинусов можно записать интеграл:

Таким образом, если функция константа, то есть:

Таким образом, далее в работе можно рассматривать функцию как:

Так же, функцию можно рассматривать, как:

Условие функции решения. Теперь рассмотрим следующее условие, которое также необходимо наложить:

Возьмем производную по от обоих сторон.

Удачно, что можно выполнить подстановку из исходно уравнения теплопроводности, то есть

Выполним вычисление интеграла по частям.

Так как соглаcно условиям Неймана для нашей краевой задачи, то имеем:

Перед нами стоит задача рассмотреть функцию . Тогда выражение примет вид:

Преобразуем выражение.

где равно 1 исходя из логики рассматриваемого условия. Отсюда можно найти чему равна :

Таким образом, выходит, что для того, чтобы краевая задача удовлетворяла всем представленным условиям необходимо ее рассматривать в следующем расширенном виде:

где функцию возьмем в виде, удовлетворяющим краевым условиям, так как к значениям данной управляющей функции будет стремиться целевая температура.

Далее, для того чтобы найти интеграл в правой части уравнения воспользуемся методом Симпсона. Суть метода проста – вычислить определенный интеграл функции с помощью приближения подынтегральной функции на отрезке интерполяционным многочленом второй степени, то есть обычной параболой. Тогда формула Симпсона имеет вид:

6. Разностная аппроксимация уравнения теплопроводности при помощи неявной схемы

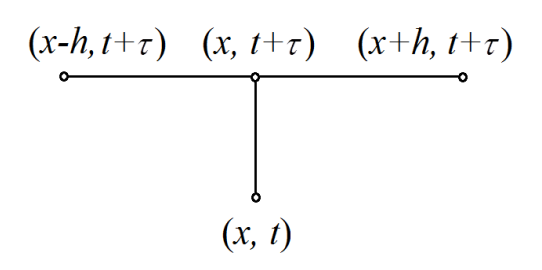
Рассмотрим разностную аппроксимацию при помощи неявной схемы линейного неоднородного, одномерного по пространству уравнения теплопроводности вида:

Для этого введем в рассматриваемой области равномерную сетку с шагом по координате и по времени .

Так как исходное уравнение содержит оба вида частных производных: по пространственной переменной и по временной переменной , то для построения его разностной аппроксимации придется использовать узлы сетки, соответствующие различным , где – временной слой, т.е. момент времени .

Устойчивость схемы зависит от того, какой вид схемы был выбран для аппроксимации, от параметров , а также на каком слое по времени аппроксимируется выражение .

Шаблон, используемый для аппроксимации при помощи неявной схемы приведен на рис. 5.1 (источник: [2], стр. 3, см. список литературы).



*Рис.5.1 Шаблон неявной схемы для уравнения теплопроводности*

Таким образом, используя приближения первой и второй производной, несложно записать аппроксимацию с помощью шаблона неявной схемы для уравнения теплопроводности. В качестве сеточной аппроксимации возьмем, например, функцию . Тогда получим разностное уравнение:

Оценим его погрешность. Для этого учтем свойства достаточно гладкой функции :

Отсюда следует, что погрешность аппроксимации равна .

Далее, повторив данную операцию по всему отрезку и взяв в качестве сеточной аппроксимации функцию , получим систему линейных разностных уравнений, имеющую следующий вид:

Для удобства дальнейшей записи введем следующее обозначение:

Тогда, система уравнений аппроксимации примет вид:

Возьмем во внимание еще условие Дирихле:

В отличие от явной схемы, о которой можно узнать в [2] стр. 1 (см. список литературы), мы не можем постепенно вычислять значения функции , так как они неявно связаны системой линейных уравнений. Причем, для ее решения удобно воспользоваться методом прогонки, так как она представима в виде трехдиагональной матрицы:

7. Метод прогонки

Метод прогонки или алгоритм Томаса используется для решения систем линейных уравнений вида:

где – трехдиагональная матрица. Данный алгоритм является частным способом решения метода последовательного исключения переменных. Его и очень удобно использовать для численного решения уравнения теплопроводности.

Первый шаг для решения системы — это представить матрицу в следующем виде:

где -двухдиагональная матрица, а – транспонированная матрица . В матричном виде это записывается следующим образом:

Тогда коэффициенты могут быть найдены с помощью следующих формул:

и далее по рекуррентным формулам:

После нахождения матрицы , необходимо рассмотреть решение следующей системы:

которую нетрудно решить, воспользовавшись следующими формулами:

и рекуррентная формула примет вид:

Заключительным шагом в решении метода прогонки – рассмотреть систему следующего вида:

Решить которую можно с помощью формул:

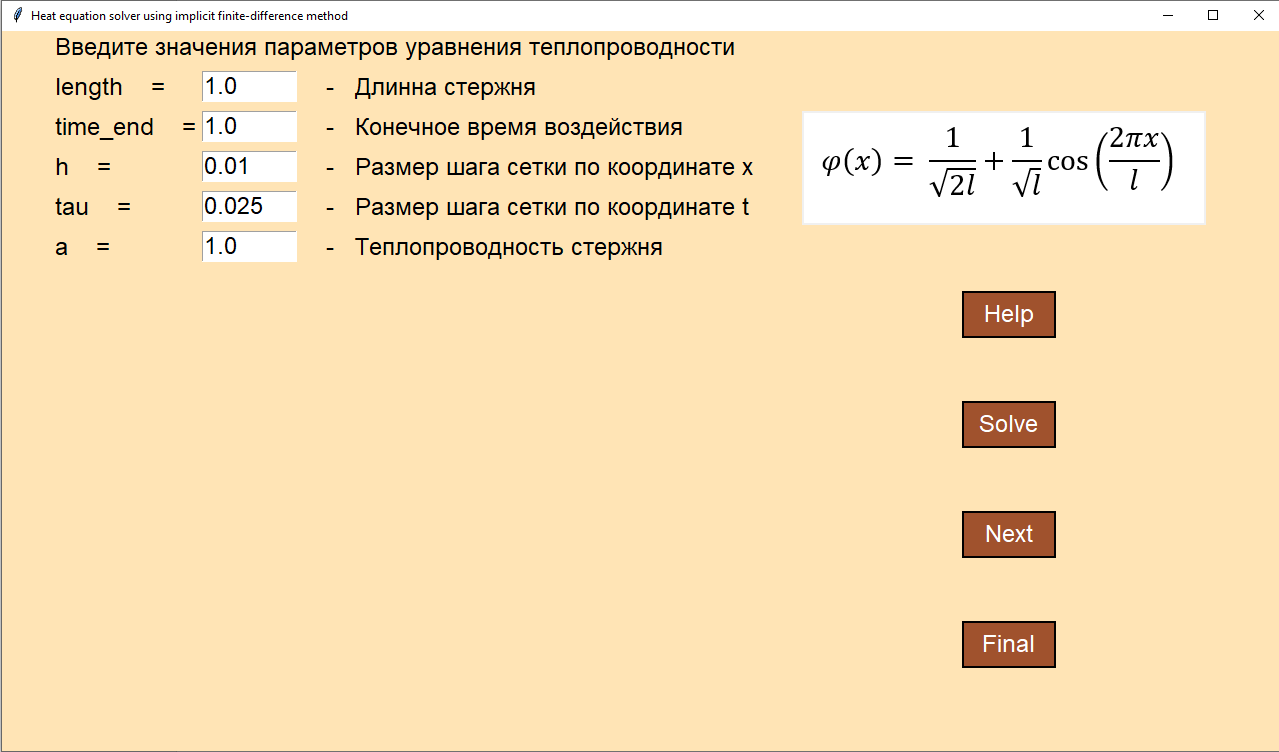
и рекуррентная формула примет вид:

Таким образом, исходное уравнение теплопроводности, рассмотренное в п. 5 сводится к решению линейной системы уравнений. Причем, левая часть может быть представлена в виде трехдиагональной матрицы. Это очень удобно и дает возможность использовать метод прогонки, который упрощает применение метода поэтапного исключения переменной. Найденное решение этой системы уравнений представляет собой распределение температуры по стрежню в момент времени .

8. Руководство пользователя

Данная программа предназначена для решения уравнения теплопроводности путем аппроксимации температуры стержня с помощью численного решения методом конечных разностей, а также графической интерпретации полученного результата.

Для работы приложения запустите исполняемый файл (расширение .exe) с названием «Heat\_equation\_with\_implicit\_method\_solver.exe». По завершению запуска (ожидайте около 10 с.) программы перед Вами предстанет интерфейс пользователя окна главного меню программы (рис. 7.1, источник: код программы Python, см. приложение).



*Рис.7.1 Главное меню программы*

Окно программы содержит пять полей ввода значений параметров, необходимых для задания параметров для численного решения уравнения теплопроводности и кнопок «Help» и «Solve».

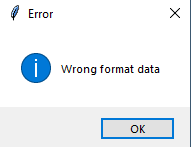
Также к каждому из полей ввода приложено описание смысла вводимого значения переменной, нетрудно догадаться о типе принимаемых значений.

Так, пользователь может ввести значения переменных:

1. – длина стержня
2. − количество шагов сетки по
3. - конечное время нагрева стержня
4. – количество шагов сетки по
5. – температуропроводность стержня

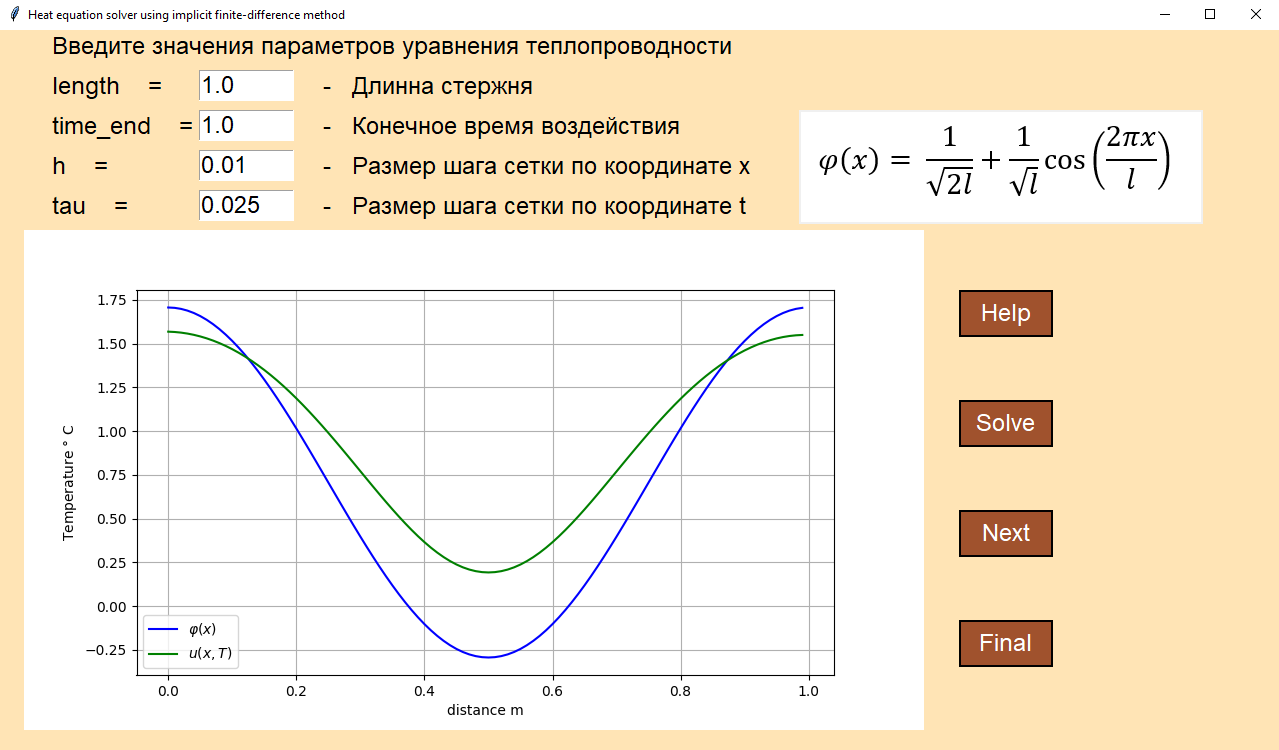
При этом, при запуске решения программы (кнопка «Solve»), некоторые параметры можно не задавать, тогда им будет присвоено значение по умолчанию.

В случае введения недопустимых значений переменных, программа выдаст ошибку о неверном формате введенных данных (рис. 7.2, источник: код программы Python, см. приложение).



*Рис.7.2 Ошибка: неверный формат данных*

Для получения решения нажмите кнопку «Solve». Перед вами появится окно с графиком. (рис. 7.3, источник: код программы Python, см. приложение).

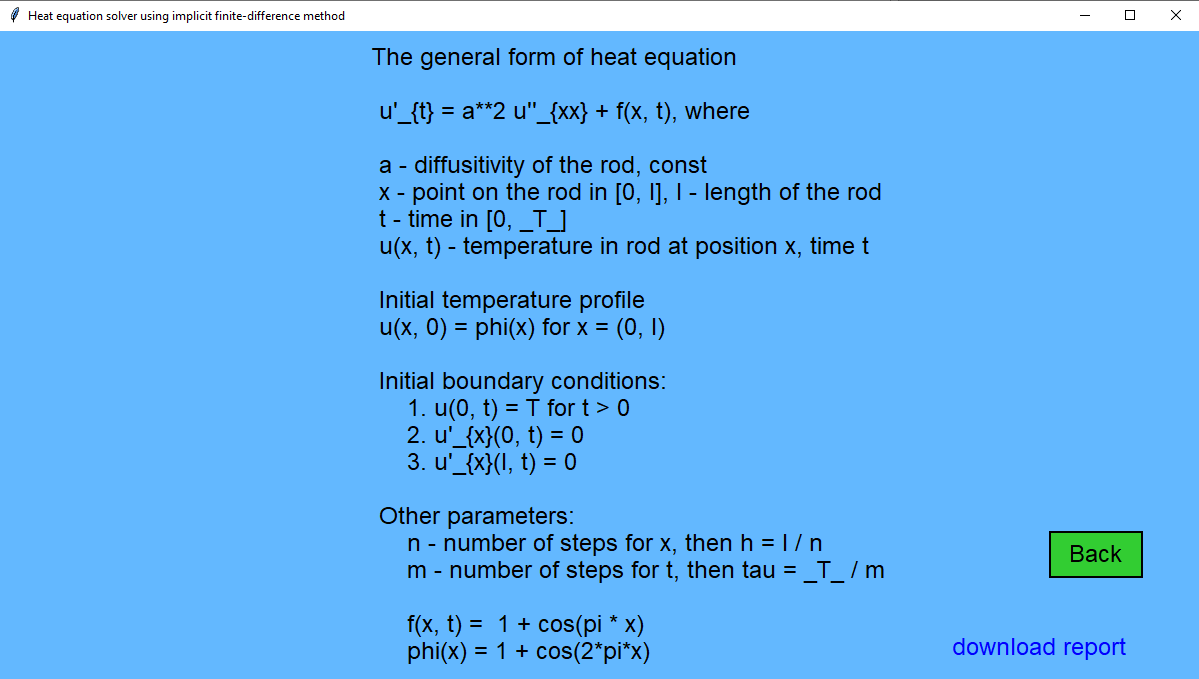


*Рис.7.3 График решения после нажатия кнопки «Solve»*

Он представляет собой решение уравнения теплопроводности с помощью неявной схемы. На оси абсцисс расположена координата тонкого стержня, на оси ординат – значение температуры в условных единицах. Синим цветом обозначено начальное распределение температуры по стержню в момент времени , а красным – распределение температуры в следующий момент времени .

Для того чтобы получить решение в момент времени нажмите кнопку «Next». Для того, чтобы вернуться в главное меню программы нажмите «Back».

Более того, Вы можете изучить справочную информацию. Для этого в главном меню программы нажмите кнопку «Help». Перед Вами появится справка о работе программы, включающая в себя подробную информацию о решаемом уравнении теплопроводности и ссылку на исчерпывающую документацию (интерактивная надпись «download report»), оформленную в виде отчета (рис. 7.4, источник: код программы Python, см. приложение).

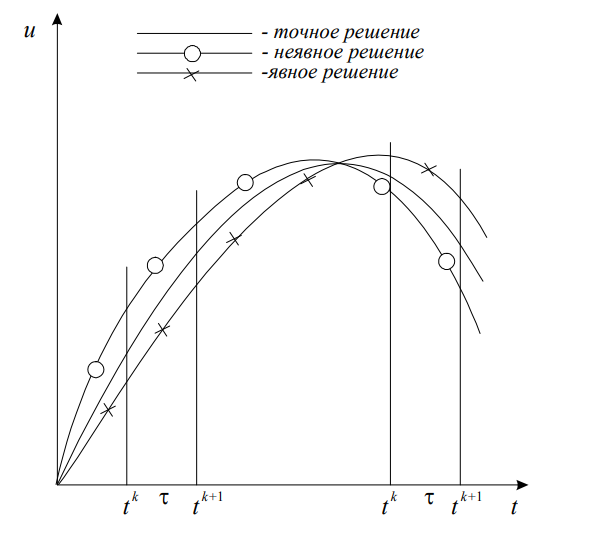
**

*Рис.7.4 Справочная информация*

9. Вывод

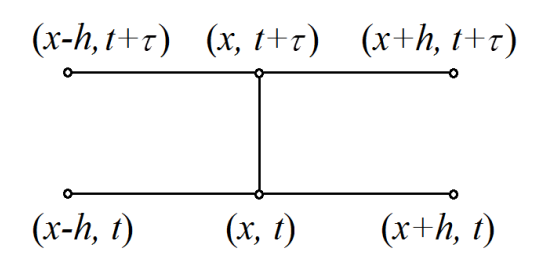
В данной лабораторной работе был проведен качественный анализ одного из основных уравнений в частных производных параболического типа – уравнения теплопроводности. Был рассмотрен физический смысл уравнения, сформулированы условия задачи. Решение уравнения было произведено в двух видах: в аналитическом и численном. Первое, аналитическое решение, было получено для однородного уравнения теплопроводности, начальные краевые условия которого были заданы в самом простом виде – в виде начальных условий Дирихле. Также на функцию, задающую температуру стержня, были наложены специальные ограничения для того, чтобы аналитическое решение можно было получить с помощью метода разделения переменных, или просто метода Фурье. Второе, численное решение, было получено в ходе анализа более сложного неоднородного уравнения теплопроводности, для которого начальные краевые условия были взяты также в усложненном виде – в виде краевых условий Неймана. Для этой цели было решено использовать численный метод конечных разностей. В качестве шаблона был выбран шаблон неявной схемы.

Таким образом, становится ясно, что аналитическое решение Фурье уступает численному методу тем, что его иногда сложно применить к нелинейным динамическим системам или системам с непостоянными коэффициентами. Однако рассмотренные численные методы тоже не являются панацеей от всех проблем. Один из существенных минусов приближенного численного решения, бросающийся в глаза, это его погрешность – несовершенная точность решения. Так, явная схема, о которой можно узнать подробнее в [2] стр. 1 (см. список литературы), и неявная схема при решении разных задач показывают неточность (см. рис. 8.1). На рисунке видно, что явная схема на больших демонстрирует тенденцию завышения результата, в то время как неявная схема демонстрирует тенденцию занижения результата.



*Рис. 8.1 Неточность явной и неявной схемы*

Поэтому, если в задаче стоит приоритет точности решения, то хорошим выбором может стать использование смешанной схемы – неявной схемы с весами, шаблон которой представлен на рис. 8.2 (источник: [2], стр. 4, см. список литературы).



*Рис. 8.2 Шаблон неявной схемы с весами для уравнения теплопроводности*

10. Список литературы и источников информации

* Эгамов, А. И. Лабораторная работа «Численное решение начально-краевой задачи для интегро-дифференциального уравнения в частных производных»: Учебно-методическое пособие — Нижний Новгород, 2019. URL: <http://www.lib.unn.ru/students/src/Lab_rab_VM.pdf>
* «Разностная аппроксимация начально-краевой задачи для уравнения теплопроводности. Понятие явной и неявной схемы.» URL: <http://math.phys.msu.ru/data/374/tema_2.pdf>
* Daileda, R. C. «The One-Dimensional Heat Equation: Partial Differential Equations Lecture 9» – Trinity University. URL: <http://ramanujan.math.trinity.edu/rdaileda/teach/s17/m3357/lectures/lecture9.pdf>
* Денисов, А. М. «Уравнения математической физики» - ВМК МГУ. URL: <https://teach-in.ru/file/synopsis/pdf/urmatfiz-denisov-M.pdf>
* «Implicit Scheme for the Heat Equation». URL: <https://people.sc.fsu.edu/~jpeterson/4-Implicit.pdf>
* «Численное решение дифференциальных уравнений с частными производными». URL: <https://mainfodotru.files.wordpress.com/2017/09/numeric-methods-part5.pdf>
* «Heat Equation». URL: <https://web.stanford.edu/class/math220b/handouts/heateqn.pdf>

11. Приложение. Код программы.

*Heat\_equation\_with\_implicit\_method\_solver.py*

*# Interface*

*import* tkinter *as* tk

*import* my\_style *as* ms

*import* webbrowser

*# Charts*

*import* matplotlib

*from* matplotlib.figure *import* Figure

*from* matplotlib.backends.backend\_tkagg *import* FigureCanvasTkAgg

*#Algotithms*

*import* implicit\_method\_for\_heat\_equation *as* alg

*import* numpy *as* np

*from* PIL *import* Image, ImageTk

def increase():

    global step

*if* step < tmsteps - 1:

        step += 1

*return* step

def last():

    global step

*while* step < tmsteps - 2:

        increase()

*return* step

def initstep():

    global step

    step = 0

*return* step

VarDef = ["Длинна стержня",

          "Конечное время воздействия",

          "Размер шага сетки по координате x",

          "Размер шага сетки по координате t",

          "Теплопроводность стержня"]

def ClearAll():

    global window

*for* part *in* window.winfo\_children():

        part.destroy()

def ThroughReport(*event*):

    webbrowser.open\_new(r"https://1drv.ms/w/s!AuklzTu4sjCrgUwweyhyq7KQ635f?e=nLffVW")

def Help():

    ClearAll()

    Lb\_info = ms.LabelT(window,

*text*="\

          The general form of heat equation \n \

          \n \

          u'\_{t} = a\*\*2 u''\_{xx} + f(x, t), where \n \

          \n \

          a - diffusitivity of the rod, const \n \

          x - point on the rod in [0, l], l - length of the rod \n \

          t - time in [0, \_T\_]\n \

          u(x, t) - temperature in rod at position x, time t \n \

          \n \

          Initial temperature profile \n \

          u(x, 0) = phi(x) for x = (0, l) \n \

          \n \

          Initial boundary conditions: \n \

              1. u(0, t) = T for t > 0 \n \

              2. u'\_{x}(0, t) = 0 \n \

              3. u'\_{x}(l, t) = 0 \n \

          \n \

          Other parameters: \n \

              n - number of steps for x, then h = l / n \n \

              m - number of steps for t, then tau = \_T\_ / m \n \

          \n \

              f(x, t) =  1 + cos(pi \* x) \n \

              phi(x) = 1 + cos(2\*pi\*x) \

    ")

    Lb\_info.place(*x* = 300, *y* = 10)

    B\_back = ms.Button(window, "back", "Back", main)

    B\_back.place(*x* = 1050, *y* = 500)

    Lb\_link = ms.LabelLink(window, *text*="download report", *callback*=report)

    Lb\_link.place(*x* = 950, *y* = 600)

def ReadData():

    global length, time\_end, h, tau, a

*try*:

        length = float(window.nametowidget(Names[0]).get())

        time\_end = float(window.nametowidget(Names[1]).get())

        h = float(window.nametowidget(Names[2]).get())

        tau = float(window.nametowidget(Names[3]).get())

        a = float(window.nametowidget(Names[4]).get())

*except*:

        tk.messagebox.showinfo("Error", "Wrong format of data")

def Solve(*\_*):

    global TotalTemp, length, h, time\_end, tau, xnsteps, tmsteps, xCoord, tCoord, step, a

    ReadData()

    xnsteps = int(length / h)

    tmsteps = int(time\_end / tau)

    xCoord = [i\*h *for* i *in* range(xnsteps)]

    tCoord = [j\*tau *for* j *in* range(tmsteps)]

    TotalTemp = [0] \* tmsteps

*for* j *in* range(tmsteps):

        TotalTemp[j] = [0] \* xnsteps

*for* i, x *in* enumerate(xCoord):

        TotalTemp[0][i] = alg.InitialConditions(x, length)

    TotalTemp = alg.ImplicitMethod(step, TotalTemp, length, time\_end, h, tau, a, xnsteps, tmsteps, xCoord, tCoord)

    ShowChart()

def ShowChart():

    figure = Figure(*figsize* = (9, 5), *dpi* = 100)

    plot = figure.add\_subplot(111)

    plot.set\_xlabel(r"distance m")

    plot.set\_ylabel(r"Temperature $\degree$ C")

    plot.plot(xCoord, TotalTemp[step], *color* = "blue")

    plot.plot(xCoord, TotalTemp[step+1], *color* = "green")

    plot.legend([r"$\varphi(x)$", r"$u(x, T)$"])

    plot.grid()

    canvas = FigureCanvasTkAgg(figure, *master* = window)

    canvas.draw()

    canvas.get\_tk\_widget().pack()

    canvas.get\_tk\_widget().place(*x* = 25, *y* = 200)

def main():

*# Window*

    ClearAll()

    ms.set\_window(window, "Heat equation solver using implicit finite-difference method")

    Lb\_intro = ms.Label(window, *name*="\_\_intro", *text*="Введите значения параметров уравнения теплопроводности")

    Lb\_intro.place(*x* = 50, *y* = 0)

*# Buttons*

    B\_help = ms.Button(window, *name*="help", *text*="Help", *command*=Help)

    B\_help.place(*x* = 960, *y* = 260)

    B\_solve = ms.Button(window, *name*="solve", *text*="Solve", *command*=lambda:Solve(initstep()))

    B\_solve.place(*x* = 960, *y* = 370)

    B\_next = ms.Button(window, *name*="next", *text*="Next", *command*=lambda:Solve(increase()))

    B\_next.place(*x* = 960, *y* = 480)

    B\_final = ms.Button(window, *name*="final", *text*="Final", *command*=lambda:Solve(last()))

    B\_final.place(*x* = 960, *y* = 590)

*# Text*

    Lb\_data = []

*for* i *in* range(5):

        Lb\_sample = ms.Label(window, *name*="\_\_{}".format(Names[i]), *text*="{}    =    ".format(Names[i]))

        Lb\_sample.place(*x* = 50, *y* = (i+1) \* 40)

        Lb\_data.append(Lb\_sample)

        Lb\_def\_sample = ms.Label(window, *name*="\_{}".format(Names[i]), *text*="   -   {}".format(VarDef[i]))

        Lb\_def\_sample.place(*x* = 300, *y* = (i+1) \* 40)

        Lb\_data.append(Lb\_def\_sample)

    E\_data = []

    Values = [1.0, 1.0, 0.01, 0.025, 1.0]

*for* i *in* range(5):

        E\_sample = tk.Entry(window, *name*="{}".format(Names[i]), *width*=7, *font*=ms.fontStyle())

        E\_sample.insert(0, str(Values[i]))

        E\_sample.place(*x* = 200, *y* = (i+1) \* 40)

        E\_data.append(E\_sample)

    canvas = tk.Canvas(window, *width*=400, *height*=110)

    image = Image.open("e:/Python\_Programs/NumericalMethods/phi.png")

    photo = ImageTk.PhotoImage(image)

    image = canvas.create\_image(0, 0, *anchor*='nw', *image*=photo)

    canvas.place(*x* = 800, *y* = 80)

    window.mainloop()

matplotlib.use("TkAgg")

window = tk.Tk()

length = 1.0

time\_end = 1.0

h = 0.01

tau = 0.025

a = 1.0

xnsteps = int(length / h)

tmsteps = int(time\_end / tau)

xCoord = [i\*h *for* i *in* range(xnsteps)]

tCoord = [j\*tau *for* j *in* range(tmsteps)]

step = 0

Names = ["length", "time\_end", "h", "tau", "a", "xnsteps", "tmsteps"]

TotalTemp = [0] \* tmsteps

*for* j *in* range(tmsteps):

    TotalTemp[j] = [0] \* xnsteps

main()

implicit\_method\_for\_heat\_equation.py

*import* numpy *as* np

*import* matplotlib.pyplot *as* plt

def InitialConditions(*func\_x*, *length*):

*return* 1/np.sqrt(2 \* length) + 1/np.sqrt(length) \* np.cos(2 \* np.pi \* func\_x / length)

def Bfunc(*func\_x*, *length*):

*return* 1/np.sqrt(2 \* length) + 1/np.sqrt(length) \* np.cos(3 \* np.pi \* func\_x / length)

def TridiagonalAlgorithm(*a*, *b*, *c*, *d*):

    size = len(d)

    alpha = [1] \* (size - 1)

    beta = [1] \* (size - 1)

    x = [1] \* size

    alpha[0] = -c[0] / b[0]

    beta[0] = d[0] / b[0]

*for* i *in* range(1, size - 1):

        alpha[i] = -c[i] / (a[i] \* alpha[i-1] + b[i])

        beta[i] = (d[i] - a[i] \* beta[i-1]) / (a[i] \* alpha[i-1] + b[i])

    x[size-1] = (d[size-1] - a[size-1] \* beta[size-2]) / (b[size-1] + a[size-1] \* alpha[size-2])

*for* i *in* range(size-2, -1, -1):

        x[i] = alpha[i] \* x[i+1] + beta[i]

*return* x

def SimpsonMethod(*TotalTemp*, *step*, *length*, *xnsteps*, *h*, *xCoord*):

    Sum = TotalTemp[step][0] \* Bfunc(xCoord[0], length) + TotalTemp[step][xnsteps-1] \* Bfunc(xCoord[xnsteps-1], length)

*for* j *in* range(1, xnsteps):

*if* (j % 2 == 1):

            Sum += 4 \* TotalTemp[step][j] \* Bfunc(xCoord[j], length)

*else*:

            Sum += 2 \* TotalTemp[step][j] \* Bfunc(xCoord[j], length)

    Sum \*= h / 3

*return* Sum

def ImplicitMethod(*step*, *TotalTemp*, *length*, *time\_end*, *h*, *tau*, *a*, *xnsteps*, *tmsteps*, *xCoord*, *tCoord*):

    A = [0] \* xnsteps

    B = [0] \* xnsteps

    C = [0] \* xnsteps

    D = [0] \* xnsteps

    factor = a \* tau / h\*\*2

*for* j *in* range(1, step + 2):

        D = TotalTemp[j-1]

        Sum = SimpsonMethod(TotalTemp, j-1, length, xnsteps, h, xCoord)

*for* i *in* range(xnsteps):

            A[i] = - factor

            B[i] = 1 + 2 \* factor - tau \* Bfunc(xCoord[i], length) \* TotalTemp[j-1][i] + TotalTemp[j-1][i] \* Sum

            C[i] = - factor

        A[0] = 0

        B[0] = 1 + factor - tau \* Bfunc(xCoord[0], length) \* TotalTemp[j-1][0] + TotalTemp[j-1][0] \* Sum

        C[0] = -factor

        A[xnsteps - 1] = -factor

        B[xnsteps - 1] = 1 + factor - tau \* Bfunc(xCoord[xnsteps - 1], length) \* TotalTemp[j-1][xnsteps-1] + TotalTemp[j-1][xnsteps-1] \* Sum

        C[xnsteps - 1] = 0

        TotalTemp[j] = TridiagonalAlgorithm(A, B, C, D)

*# Нормировка (чтобы все было в порядке, а то она уходит куда-то)*

        Sum = SimpsonMethod(TotalTemp, j, length, xnsteps, h, xCoord)

*for* i *in* range(xnsteps):

            TotalTemp[j][i] = TotalTemp[j][i] / Sum

*return* TotalTemp

my\_style.py

*import* tkinter *as* tk

*import* tkinter.font

*from* functools *import* partial

*#Style*

def fontStyle():

*return* tkinter.font.Font(*family*="lucida grande bold", *size*=18)

def global\_background(*num*):

*if* num == 1:

*return* "moccasin"

*# return "Steelblue1"*

*# fontStyle\_coordinate=tk.font.Font(family="Arial", size=12)*

def set\_window(*window* : tk.Tk, *text* : str):

    window.geometry('1280x720')

    window.title(text)

    window.configure(*background*=global\_background(1))

def Button(*window*, *name*, *text*, *command*):

    B\_temp = tk.Button(window,

*name*=name,

*text*=text,

*command* = command,

*width* = 6,

*height* = 1,

*compound* = tkinter.CENTER,

*font* = fontStyle(),

*background* = "sienna",

*foreground*="white",

*highlightbackground* = "black",

*relief* = "solid")

*return* B\_temp

def ButtonP(*window*, *name*, *text*, *command*, *p*):

    B\_temp = tk.Button(window,

*name*=name,

*text*=text,

*command* = partial(command, p),

*width* = 6,

*height* = 1,

*compound* = tkinter.CENTER,

*font* = fontStyle(),

*background* = "LimeGreen",

*highlightbackground* = "black",

*relief* = "solid")

*return* B\_temp

def Label(*window*, *name*="label", *text*=""):

    L\_temp = tk.Label(window,

*name*=name,

*text*=text,

*font*=fontStyle(),

*background*=global\_background(1))

*return* L\_temp

def LabelT(*window*, *text*=""):

    L\_temp = tk.Label(window,

*text*=text,

*font*=fontStyle(),

*background*=global\_background(1),

*justify*=tk.LEFT)

*return* L\_temp

def LabelLink(*window*, *text*="", *callback*=None):

    L\_temp = tk.Label(window,

*text*=text,

*font*=fontStyle(),

*background*=global\_background(1),

*fg*="blue",

*cursor*="hand2")

    L\_temp.bind("<Button-1>", callback)

*return* L\_temp