**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ**

**ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «МЭИ»**

**ИНСТИТУТ ИНФОРМАЦИОННЫХ И ВЫЧИСЛИТЕЛЬНЫХ ТЕХНОЛОГИЙ**

**КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ И ИСКУССТВЕННОГО ИНТЕЛЕКТА**

**Курсовая работа**

на тему

Решение краевой задачи в сложной области методом конечных элементов.

|  |
| --- |
| *Выполнила*: студентка Малышева Диана Даниловна 3 курс, группа № А-13а-20  Работа сдана 17 «февраля» 2023 г. |
| *Научный руководитель:*  Крымов Никита Евгеньевич |

**Москва 2023**

# Содержание

1. [Постановка задачи 3](#_bookmark0)
2. [Дискретизация задачи 4](#_bookmark1)
3. [Интерполяция 8](#_bookmark2)
4. [Подготовка тестовых примеров 9](#_bookmark3)
   1. [Подготовка первого тестового примера 9](#_bookmark4)
   2. [Подготовка второго тестового примера 9](#_bookmark5)
   3. [Подготовка третьего тестового примера 10](#_bookmark6)
5. [Решение тестовых примеров 11](#_bookmark7)
   1. [Решение первого тестового примера 11](#_bookmark8)
   2. [Решение второго тестового примера 13](#_bookmark9)
   3. [Решение третьего тестового примера 15](#_bookmark10)
6. [Заключение 17](#_bookmark11)
7. [Список литературы 17](#_bookmark12)
8. [Приложение 18](#_bookmark13)

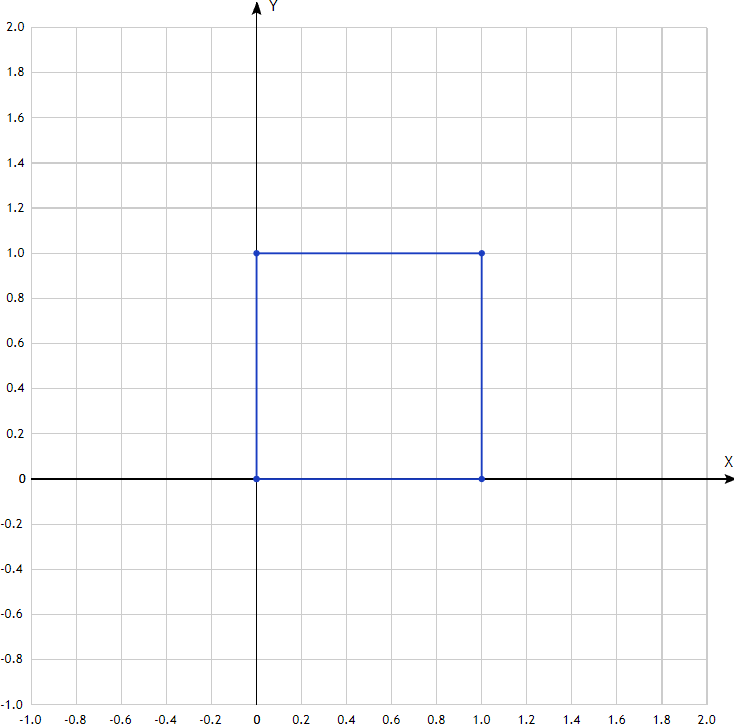
# Постановка задачи

Рассматривается задача Дирихле в сложной области.

𝜕2𝑢 𝜕2𝑢

− (𝜕𝑥2 + 𝜕𝑦2) = 𝑓, (𝑥, 𝑦) ∈ 𝐺, 𝑢 = 𝑔, (𝑥, 𝑦) ∈ 𝜕𝐺

Уравнение описывает установившееся в какой-то бесконечно далекий момент времени, при неизменности внешней среды и внутренних физикохимических процессов, распределение абсолютной температуры u в пластине G. Здесь f — плотность внутренних источников тепла, g — температура внешней среды.



Необходимо решить задачу методом конечных элементов. Продемонстрируйте работоспособность программы на тестовых примерах, решениями которых являются функции:

1.𝑢(𝑥, 𝑦) = −𝑥 + 2𝑦 + 20

2.𝑢(𝑥, 𝑦) = 2𝑥3𝑦 + 𝑥(𝑦 − 𝑥) + 20

𝑥+𝑦

3.𝑢(𝑥, 𝑦) = cos(4𝑥𝑦) sin(2𝑥) + 𝑒 π + 10

Результат должен быть представлен в виде термограммы, созданной с помощью matplotlib с применением цветовой карты gnuplot из matplotlib.colormaps.

# Дискретизация задачи

Рассмотрим исходную задачу Дирихле для области:

𝜕2𝑢 𝜕2𝑢

− (𝜕𝑥2 + 𝜕𝑦2) = 𝑓

Раскроем скобки и домножим обе части на некоторую функцию φ, равную нулю на границе:

𝜕2𝑢

− 𝜕𝑥2 φ −

𝜕2𝑢

𝜕𝑦2 φ = 𝑓φ, φ(𝑥, 𝑦) = 0, (𝑥, 𝑦) ∈ 𝜕𝐺

И добавим интегралы по области G в обе части равенства:

𝜕2𝑢

− ∫ 𝜕𝑥2 φdxdy − ∫

𝐺

𝐺

𝜕2𝑢

𝜕𝑦2 φdxdy = ∫ 𝑓φdxdy

𝐺

Проинтегрируем по частям интегралы в левой части (т.к. функция φ = 0 на границе):

∫

𝐺

∫

𝐺

Подставим полученные интегралы:

𝜕2𝑢

𝜕𝑥2 φdxdy = ∫

𝐺

𝜕2𝑢

𝜕𝑦2 φdxdy = ∫

𝐺

𝜕𝑢 𝜕φ

𝜕𝑥 𝜕𝑥 dxdy

𝜕𝑢 𝜕φ

𝜕𝑦 𝜕𝑦 dxdy

𝜕𝑢 𝜕φ

∫ 𝜕𝑥 𝜕𝑥 dxdy + ∫

𝐺

𝐺

𝜕𝑢 𝜕φ

𝜕𝑦 𝜕𝑦 dxdy = ∫ 𝑓φdxdy

𝐺

Дискретизируем функцию φ как φi,j, 1 ≤ i ≤ N, 1 ≤ j ≤ N, каждая из φi,j = 0 на границе. Будем искать решение задачи в виде:

𝑁 𝑁

𝑢̅ = ∑ ∑ 𝛼𝑖,𝑗𝜑𝑖,𝑗

𝑖=1 𝑗=1

Используя 𝑢̅ вычислим значения частных производных для u:

𝜕𝑢̅

𝜕𝑥

𝑁 𝑁

= ∑ ∑ 𝛼𝑖,𝑗

𝑖=1 𝑗=1

𝜕𝜑𝑖,𝑗

## 𝜕𝑥

𝜕𝑢̅

𝜕𝑦

𝑁 𝑁

= ∑ ∑ 𝛼𝑖,𝑗

𝑖=1 𝑗=1

𝜕𝜑𝑖,𝑗

## 𝜕𝑦

И подставим полученное в равенство:

𝑁 𝑁

## 𝜕𝜑

𝜕𝜑

𝑁 𝑁

## 𝜕𝜑

𝜕𝜑

∫ ∑ ∑ 𝛼𝑖,𝑗

𝐺 𝑖=1 𝑗=1

𝑖,𝑗

## 𝜕𝑥

𝜕𝑥

𝑑𝑥𝑑𝑦 + ∫ ∑ ∑ 𝛼𝑖,𝑗

𝐺 𝑖=1 𝑗=1

𝑖,𝑗

## 𝜕𝑦

𝜕𝑦 dxdy = ∫ 𝑓φdxdy

Поменяем местами знаки интегрирования и суммы:

𝐺

𝑁 𝑁

∑ ∑ ∫ 𝛼𝑖,𝑗

𝑖=1 𝑗=1 𝐺

𝜕𝜑𝑖,𝑗

## 𝜕𝑥

𝜕𝜑

## 𝜕𝑥

𝑑𝑥𝑑𝑦 +

𝑁 𝑁

∑ ∑ ∫ 𝛼𝑖,𝑗

𝑖=1 𝑗=1 𝐺

𝜕𝜑𝑖,𝑗

## 𝜕𝑦

𝜕𝜑

## 𝜕𝑦

𝑑𝑥𝑑𝑦 = ∫ 𝑓φdxdy

𝐺

Теперь вместо φ будем по очереди подставлять φk,l, 1 ≤ k ≤ N, 1 ≤ l ≤ N и вынесем 𝛼𝑖,𝑗 за знак интеграла:

𝑁 𝑁

## 𝜕𝜑

𝜕𝜑

𝑁 𝑁

## 𝜕𝜑

𝜕𝜑

∑ ∑ 𝛼𝑖,𝑗 ∫

𝑖,𝑗 𝑘,𝑙 𝑑𝑥𝑑𝑦 + ∑ ∑ 𝛼𝑖,𝑗 ∫

𝑖,𝑗 𝑘,𝑙

## 𝑑𝑥𝑑𝑦 =

∫ 𝑓φdxdy

𝑖=1 𝑗=1

## 𝐺 𝜕𝑥

𝜕𝑥

𝑖=1 𝑗=1

## 𝐺 𝜕𝑦

𝜕𝑦 𝐺

Вынесем знаки суммы и 𝛼𝑖,𝑗 за скобки:

𝑁 𝑁

## 𝜕𝜑 𝜕𝜑

𝜕𝜑 𝜕𝜑

∑ ∑ 𝛼𝑖,𝑗(∫

𝑖,𝑗 𝑘,𝑙 𝑑𝑥𝑑𝑦 + ∫

𝑖,𝑗 𝑘,𝑙

## 𝑑𝑥𝑑𝑦) =

∫ 𝑓φdxdy

𝑖=1 𝑗=1

## 𝐺 𝜕𝑥

𝜕𝑥

## 𝐺 𝜕𝑦

𝜕𝑦 𝐺

Рассмотрим функцию

𝑥

ﻟ 1 − (ℎ − 𝑖) ; 𝑥𝑖 ≤ 𝑥 ≤ 𝑥𝑖+1, 𝑦𝑗 ≤ 𝑦 ≤ 𝑦𝑗 + (𝑥 − 𝑥𝑖); (I)

𝑦

I 1 − (ℎ − 𝑗) ; 𝑥𝑖 ≤ 𝑥 ≤ 𝑥𝑖+1, 𝑦𝑗 + (𝑥 − 𝑥𝑖) ≤ 𝑦 ≤ 𝑦𝑗+1; (II)

I 𝑥 𝑦 ( )

𝜑(𝑥, 𝑦) =

1 1 + (ℎ − 𝑖) − (ℎ − 𝑗) ; 𝑥𝑖−1 ≤ 𝑥 ≤ 𝑥𝑖, 𝑦𝑗 ≤ 𝑦 ≤ 𝑦𝑗 +

ℎ ❪ 𝑥

𝑥 − 𝑥𝑖−1

; (III)

1 + (ℎ − 𝑖) ; 𝑥𝑖−1 ≤ 𝑥 ≤ 𝑥𝑖, 𝑦𝑗−1 + (𝑥 − 𝑥𝑖−1) ≤ 𝑦 ≤ 𝑦𝑗; (IV)

𝑦

I 1 + (ℎ − 𝑗) ; 𝑥𝑖−1 ≤ 𝑥 ≤ 𝑥𝑖, 𝑦𝑗−1 ≤ 𝑦 ≤ 𝑦𝑗−1 + (𝑥 − 𝑥𝑖−1); (V)

I 𝑥 𝑦 ( )

𝗅1 − (ℎ − 𝑖) + (ℎ − 𝑗) ; 𝑥𝑖 ≤ 𝑥 ≤ 𝑥𝑖+1, 𝑦𝑗−1 + + 𝑥 − 𝑥𝑖

Найдем для неё частные производные:

≤ 𝑦 ≤ 𝑦𝑗; (VI)

1

ﻟ − ℎ ; 𝑥𝑖 ≤ 𝑥 ≤ 𝑥𝑖+1, 𝑦𝑗 ≤ 𝑦 ≤ 𝑦𝑗 + (𝑥 − 𝑥𝑖); (I)

0; 𝑥𝑖 ≤ 𝑥 ≤ 𝑥𝑖+1, 𝑦𝑗 + (𝑥 − 𝑥𝑖) ≤ 𝑦 ≤ 𝑦𝑗+1; (II)

𝜕𝜑𝑖,𝑗 1

=

1

I ; 𝑥𝑖−1 ℎ

≤ 𝑥 ≤ 𝑥𝑖, 𝑦𝑗 ≤ 𝑦 ≤ 𝑦𝑗

+ (𝑥 − 𝑥

𝑖−1

); (III)

𝑥 ℎ ❪ 1 ; 𝑥

ℎ 𝑖−1

≤ 𝑥 ≤ 𝑥𝑖, 𝑦𝑗−1

+ (𝑥 − 𝑥

𝑖−1) ≤ 𝑦 ≤ 𝑦𝑗; (IV)

0; 𝑥𝑖−1 ≤ 𝑥 ≤ 𝑥𝑖, 𝑦𝑗−1 ≤ 𝑦 ≤ 𝑦𝑗−1 + (𝑥 − 𝑥𝑖−1); (V)

I 1

− ; 𝑥

≤ 𝑥 ≤ 𝑥

, 𝑦

+ +(𝑥 − 𝑥 ) ≤ 𝑦 ≤ 𝑦 ; (VI)

𝗅 ℎ 𝑖

𝑖+1

𝑗−1

𝑖 𝑗

ﻟ 0; 𝑥𝑖 ≤ 𝑥 ≤ 𝑥𝑖+1, 𝑦𝑗 ≤ 𝑦 ≤ 𝑦𝑗 + (𝑥 − 𝑥𝑖); (I) 1

− ℎ ; 𝑥𝑖 ≤ 𝑥 ≤ 𝑥𝑖+1, 𝑦𝑗 + (𝑥 − 𝑥𝑖) ≤ 𝑦 ≤ 𝑦𝑗+1; (II)

𝜕𝜑𝑖,𝑗 1

=

1

− ℎ ; 𝑥𝑖−1

I

≤ 𝑥 ≤ 𝑥𝑖, 𝑦𝑗 ≤ 𝑦 ≤ 𝑦𝑗

+ (𝑥 − 𝑥

𝑖−1

); (III)

𝑦 ℎ ❪ 0; 𝑥𝑖−1 ≤ 𝑥 ≤ 𝑥𝑖, 𝑦𝑗−1 + (𝑥 − 𝑥𝑖−1) ≤ 𝑦 ≤ 𝑦𝑗; (IV)

1

ℎ ; 𝑥𝑖−1 ≤ 𝑥 ≤ 𝑥𝑖, 𝑦𝑗−1 ≤ 𝑦 ≤ 𝑦𝑗−1 + (𝑥 − 𝑥𝑖−1); (V)

I 1

; 𝑥

≤ 𝑥 ≤ 𝑥

, 𝑦

+ +(𝑥 − 𝑥 ) ≤ 𝑦 ≤ 𝑦 ; (VI)

𝗅 ℎ 𝑖

𝑖+1

𝑗−1

𝑖 𝑗

Вычислим интегралы ∫ 𝜕𝜑𝑖,𝑗 𝜕𝜑𝑘,𝑙

𝑑𝑥𝑑𝑦 во всех случаях:

𝐺 𝜕𝑥 𝜕𝑥

1. Совпадают: i = k, j = l

𝜕𝜑𝑖,𝑗 𝜕𝜑

𝜕𝜑𝑖,𝑗 2

𝜕𝜑𝑖,𝑗 2

𝜕𝜑𝑖,𝑗 2

∫𝐺

𝜕𝑥

𝜕𝑥 𝑑𝑥𝑑𝑦 = ∫𝐺

( 𝜕𝑥 )

𝑑𝑥𝑑𝑦 = ∫I

( 𝜕𝑥 )

𝑑𝑥𝑑𝑦 + ∫II

( 𝜕𝑥 )

## 𝑑𝑥𝑑𝑦 + … +

𝜕𝜑𝑖,𝑗 2

1 2

∫VI ( 𝜕𝑥 )

𝑑𝑥𝑑𝑦 = 4 2 = 2

2ℎ ℎ

𝜕𝜑𝑖,𝑗 𝜕𝜑

𝜕𝜑𝑖,𝑗 2

𝜕𝜑𝑖,𝑗 2

𝜕𝜑𝑖,𝑗 2

∫𝐺

𝜕𝑦

𝜕𝑦 𝑑𝑥𝑑𝑦 = ∫𝐺

( 𝜕𝑦 )

𝑑𝑥𝑑𝑦 = ∫I

( 𝜕𝑦 )

𝑑𝑥𝑑𝑦 + ∫II

( 𝜕𝑦 )

## 𝑑𝑥𝑑𝑦 + … +

𝜕𝜑𝑖,𝑗 2

1 2

∫VI ( 𝜕𝑦 )

𝑑𝑥𝑑𝑦 = 4 2 = 2

2ℎ ℎ

𝜕𝜑𝑖,𝑗 𝜕𝜑

𝐺

𝜕𝜑𝑖,𝑗 𝜕𝜑 4

∫ 𝜕𝑥

𝐺

## 𝜕𝑥 𝑑𝑥𝑑𝑦 + ∫

𝜕𝑦

## 𝜕𝑦 𝑑𝑥𝑑𝑦 = ℎ2

1. Смещение вправо на 1: i = k + 1, j = l

𝜕𝜑𝑖+1,𝑗 𝜕𝜑 1

∫ 𝜕𝑥

𝐺

## 𝜕𝑥 𝑑𝑥𝑑𝑦 = − ℎ2

𝜕𝜑𝑖+1,𝑗 𝜕𝜑

∫ 𝜕𝑦

𝐺

## 𝜕𝑦 𝑑𝑥𝑑𝑦 = 0

𝜕𝜑𝑖+1,𝑗 𝜕𝜑 𝜕𝜑𝑖+1,𝑗 𝜕𝜑 1

𝐺

∫ 𝜕𝑥

𝐺

## 𝜕𝑥 𝑑𝑥𝑑𝑦 + ∫

𝜕𝑦

## 𝜕𝑦 𝑑𝑥𝑑𝑦 = − ℎ2

1. Смещение вправо-вверх на 1: i = k + 1; j = l + 1

𝜕𝜑𝑖+1,𝑗+1 𝜕𝜑

∫ 𝜕𝑥

𝐺

## 𝜕𝑥 𝑑𝑥𝑑𝑦 = 0

𝜕𝜑𝑖+1,𝑗+1 𝜕𝜑

∫ 𝜕𝑦

𝐺

## 𝜕𝑦 𝑑𝑥𝑑𝑦 = 0

𝜕𝜑𝑖+1,𝑗+1 𝜕𝜑 𝜕𝜑𝑖+1,𝑗+1 𝜕𝜑

𝐺

∫ 𝜕𝑥

𝐺

## 𝜕𝑥 𝑑𝑥𝑑𝑦 + ∫

𝜕𝑦

## 𝜕𝑦 𝑑𝑥𝑑𝑦 = 0

1. Смещение вверх на 1: i = k; j = l + 1

𝜕𝜑𝑖,𝑗+1 𝜕𝜑

∫ 𝜕𝑥

𝐺

## 𝜕𝑥 𝑑𝑥𝑑𝑦 = 0

𝜕𝜑𝑖,𝑗+1 𝜕𝜑 1

∫ 𝜕𝑦

𝐺

## 𝜕𝑦 𝑑𝑥𝑑𝑦 = − ℎ2

𝜕𝜑𝑖,𝑗+1 𝜕𝜑 𝜕𝜑𝑖,𝑗+1 𝜕𝜑 1

𝐺

∫ 𝜕𝑥

𝐺

## 𝜕𝑥 𝑑𝑥𝑑𝑦 + ∫

𝜕𝑦

## 𝜕𝑦 𝑑𝑥𝑑𝑦 = − ℎ2

1. Смещение влево на 1: i = k – 1; j = 1

𝜕𝜑𝑖−1,𝑗 𝜕𝜑 1

∫ 𝜕𝑥

𝐺

## 𝜕𝑥 𝑑𝑥𝑑𝑦 = − ℎ2

𝜕𝜑𝑖−1,𝑗 𝜕𝜑

∫ 𝜕𝑦

𝐺

## 𝜕𝑦 𝑑𝑥𝑑𝑦 = 0

𝜕𝜑𝑖−1,𝑗 𝜕𝜑 𝜕𝜑𝑖−1,𝑗 𝜕𝜑 1

𝐺

∫ 𝜕𝑥

𝐺

## 𝜕𝑥 𝑑𝑥𝑑𝑦 + ∫

𝜕𝑦

## 𝜕𝑦 𝑑𝑥𝑑𝑦 = − ℎ2

1. Смещение влево-вниз:

𝜕𝜑𝑖−1,𝑗−1 𝜕𝜑

∫ 𝜕𝑥

𝐺

## 𝜕𝑥 𝑑𝑥𝑑𝑦 = 0

𝜕𝜑𝑖−1,𝑗−1 𝜕𝜑

∫ 𝜕𝑦

𝐺

## 𝜕𝑦 𝑑𝑥𝑑𝑦 = 0

𝜕𝜑𝑖−1,𝑗−1 𝜕𝜑 𝜕𝜑𝑖−1,𝑗−1 𝜕𝜑

𝐺

∫ 𝜕𝑥

𝐺

## 𝜕𝑥 𝑑𝑥𝑑𝑦 + ∫

𝜕𝑦

## 𝜕𝑦 𝑑𝑥𝑑𝑦 = 0

1. Смещение вниз на 1: i = k; j = l – 1

𝜕𝜑𝑖,𝑗−1 𝜕𝜑

∫ 𝜕𝑥

𝐺

## 𝜕𝑥 𝑑𝑥𝑑𝑦 = 0

𝜕𝜑𝑖,𝑗−1 𝜕𝜑 1

∫ 𝜕𝑦

𝐺

## 𝜕𝑦 𝑑𝑥𝑑𝑦 = − ℎ2

𝜕𝜑𝑖,𝑗−1 𝜕𝜑 𝜕𝜑𝑖,𝑗−1 𝜕𝜑 1

𝐺

∫ 𝜕𝑥

𝐺

## 𝜕𝑥 𝑑𝑥𝑑𝑦 + ∫

𝜕𝑦

## 𝜕𝑦 𝑑𝑥𝑑𝑦 = − ℎ2

1. Во всех остальных случаях

𝜕𝜑𝑖,𝑗 𝜕𝜑𝑘,𝑙

∫ 𝜕𝑥

𝐺

## 𝜕𝑥 𝑑𝑥𝑑𝑦 = 0

𝜕𝜑𝑖,𝑗 𝜕𝜑𝑘,𝑙

∫ 𝜕𝑦

𝐺

## 𝜕𝑦 𝑑𝑥𝑑𝑦 = 0

𝜕𝜑𝑖,𝑗 𝜕𝜑𝑘,𝑙 𝜕𝜑𝑖,𝑗 𝜕𝜑𝑘,𝑙

𝐺

∫ 𝜕𝑥

𝐺

## 𝜕𝑥 𝑑𝑥𝑑𝑦 + ∫

𝜕𝑦

## 𝜕𝑦 𝑑𝑥𝑑𝑦 = 0

В случае ненулевой границы:

∑𝑁

∑𝑁

𝛼 (∫ 𝜕𝜑𝑖,𝑗 𝜕𝜑𝑘,𝑙 𝑑𝑥𝑑𝑦 + ∫ 𝜕𝜑𝑖,𝑗 𝜕𝜑𝑘,𝑙

## 𝑑𝑥𝑑𝑦) =

∫ 𝑓𝜑

dxdy −

𝑖=1

𝑗=1

𝑖,𝑗 𝐺

𝜕𝑥

𝜕𝑥

𝐺 𝜕𝑦

𝜕𝑦

𝐺 𝑘,𝑙

− 𝑢

(∫ 𝜕𝜑𝑖,𝑗 𝜕𝜑𝑘,𝑙 𝑑𝑥𝑑𝑦 + ∫ 𝜕𝜑𝑖,𝑗 𝜕𝜑𝑘,𝑙 𝑑𝑥𝑑𝑦)

𝑖,𝑗

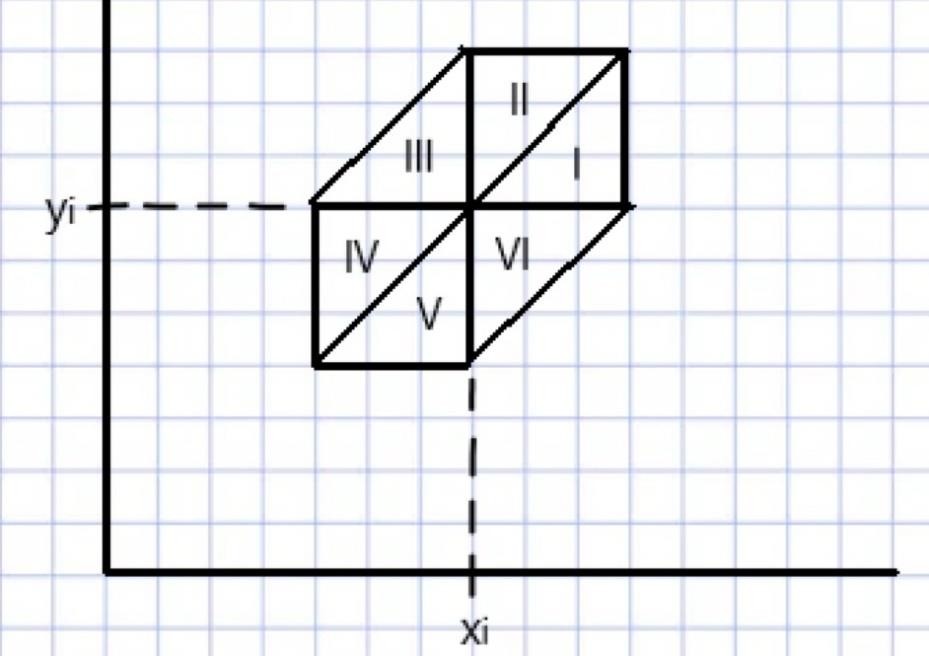
𝐺 𝜕𝑥

𝜕𝑥

𝐺 𝜕𝑦

𝜕𝑦

Где 𝑢𝑖,𝑗 – известное значение функции на границе.



# Интерполяция

Т.к. вычисление узлов (матрицы) с помощью метода конечных элементов требует больших вычислительных мощностей, и, как следствие, требует много времени, даже на мощных устройствах. Решено было использовать интерполяцию, чтобы не терять в качестве графика, но ускорить вычислительный процесс. В качестве метода для интерполяции была выбрана линейная интерполяция, а точнее её обобщение одной переменной для функций двух переменных (билинейная интерполяция). Обобщение основано на применении обычной линейной интерполяции сначала в направлении одной из координат (в нашем случае x), а затем в перпендикулярном направлении (в нашем случае y).

# Подготовка тестовых примеров

# Подготовка первого тестового примера

Рассмотрим краевую задачу

−𝑢′′ = 𝑓, 𝑥 ∈ [0,1], 𝑦 ∈ [0,1]

𝑢(0, 𝑦) = 𝑢𝑎, 𝑢(𝑥, 0) = 𝑢𝑏, 𝑢(1, 𝑦) = 𝑢𝑐, 𝑢(𝑥, 1) = 𝑢𝑑

Составим тестовую задачу для функции 𝑢(𝑥, 𝑦) = −𝑥 + 2𝑦 + 20

Так как

𝑓 = −(𝑢′′ + 𝑢′′ ) = −(−𝑥 + 2𝑦 + 20)′′ = 0

𝑥𝑥 𝑦𝑦

𝑢 (0, 𝑦) = −0 + 2 ∗ 𝑦 + 20 = 2𝑦 + 20

𝑢 (𝑥, 0) = −𝑥 + 2 ∗ 0 + 20 = −𝑥 + 20

𝑢 (1, 𝑦) = −1 + 2 ∗ 𝑦 + 20 = 2𝑦 + 19

𝑢 (𝑥, 1) = −𝑥 + 2 ∗ 1 + 20 = −𝑥 + 22

То функция 𝑢(𝑥, 𝑦) = −𝑥 + 2𝑦 + 20 является решением краевой задачи

−𝑢′′ = 0 , 𝑥 ∈ [0,1], 𝑦 ∈ [0,1]

𝑢 (0, 𝑦) = 2𝑦 + 20

𝑢 (𝑥, 0) = −𝑥 + 20

𝑢 (1, 𝑦) = 2𝑦 + 19

𝑢 (𝑥, 1) = −𝑥 + 22

Теперь эта задача может быть использована для проверки работоспособности численного метода решения краевой задачи, так как численное решение можно сравнить с аналитическим решением, которое известно.

# Подготовка второго тестового примера

Рассмотрим краевую задачу

−𝑢′′ = 𝑓, 𝑥 ∈ 𝑥 ∈ [0,1], 𝑦 ∈ [0,1]

𝑢(0, 𝑦) = 𝑢𝑎, 𝑢(𝑥, 0) = 𝑢𝑏, 𝑢(1, 𝑦) = 𝑢𝑐, 𝑢(𝑥, 1) = 𝑢𝑑

Составим тестовую задачу для функции 𝑢(𝑥, 𝑦) = 2𝑥3𝑦 + 𝑥(𝑦 − 𝑥) + 20

Так как

𝑓 = −(𝑢′′ + 𝑢′′ ) = −(2𝑥3𝑦 + 𝑥(𝑦 − 𝑥) + 20)′′ = −12𝑦𝑥 + 2

𝑥𝑥 𝑦𝑦

𝑢 (0, 𝑦) = 2 ∗ 03 ∗ 𝑦 + 0 ∗ (𝑦 − 0) + 20 = 20

𝑢 (𝑥, 0) = 2 ∗ 𝑥3 ∗ 0 + 𝑥 ∗ (0 − 𝑥) + 20 = −𝑥2 + 20

𝑢 (1, 𝑦) = 2 ∗ 13 ∗ 𝑦 + 1 ∗ (𝑦 − 1) + 20 = 3𝑦 + 19

𝑢 (𝑥, 1) = 2 ∗ 𝑥3 ∗ 1 + 𝑥 ∗ (1 − 𝑥) + 20 = 2𝑥3 − 𝑥2 + 𝑥 + 20

То функция 𝑢(𝑥, 𝑦) = 2𝑥3𝑦 + 𝑥(𝑦 − 𝑥) + 20 является решением краевой задачи

−𝑢′′ = −12𝑦𝑥 + 2 , 𝑥 ∈ [0,1], 𝑦 ∈ [0,1]

𝑢 (0, 𝑦) = 20

𝑢 (𝑥, 0) = −𝑥2 + 20

𝑢 (1, 𝑦) = 3𝑦 + 19

𝑢 (𝑥, 1) = 2𝑥3 − 𝑥2 + 𝑥 + 20

Теперь эта задача может быть использована для проверки работоспособности численного метода решения краевой задачи, так как численное решение можно сравнить с аналитическим решением, которое известно.

# Подготовка третьего тестового примера

Рассмотрим краевую задачу

−𝑢′′ = 𝑓, 𝑥 ∈ 𝑥 ∈ [0,1], 𝑦 ∈ [0,1]

𝑢(0, 𝑦) = 𝑢𝑎, 𝑢(𝑥, 0) = 𝑢𝑏, 𝑢(1, 𝑦) = 𝑢𝑐, 𝑢(𝑥, 1) = 𝑢𝑑

Составим тестовую задачу для функции 𝑢(𝑥, 𝑦) = 𝑐𝑜𝑠4𝑥𝑦 𝑠𝑖𝑛2𝑥 + 𝑒

Так как

𝑥+𝑦

𝜋 + 10

𝑓 = −(𝑢′′ + 𝑢′′ ) = − ( 𝑐𝑜𝑠4𝑥𝑦 𝑠𝑖𝑛2𝑥 + 𝑒

𝑥+𝑦 ′′

𝜋 + 10)

= 16𝑦2𝑐𝑜𝑠4𝑥𝑦 𝑠𝑖𝑛2𝑥 + 16𝑦 𝑠𝑖𝑛4𝑥𝑦 𝑐𝑜𝑠2𝑥 +

𝑥𝑥

𝑦𝑦

𝑥+𝑦

4𝑐𝑜𝑠4𝑥𝑦 𝑠𝑖𝑛2𝑥 − 2𝑒 𝜋

+ 16𝑥2𝑠𝑖𝑛2𝑥 𝑐𝑜𝑠4𝑥𝑦

𝜋2

0+𝑦 𝑦

𝑢 (0, 𝑦) = cos(4 ∗ 0 ∗ 𝑦) ∗ sin(2 ∗ 0) + 𝑒 𝜋 + 10 = 𝑒𝜋 + 10

𝑢 (𝑥, 0) = cos(4 ∗ 𝑥 ∗ 0) ∗ sin(2 ∗ 𝑥) + 𝑒

𝑢 (1, 𝑦) = cos(4 ∗ 1 ∗ 𝑦) ∗ sin(2 ∗ 1) + 𝑒

𝑥+0 𝑥

𝜋 + 10 = 𝑒𝜋 + 10

1+𝑦

𝜋 + 10 = 𝑐𝑜𝑠4𝑦 𝑠𝑖𝑛2 + 𝑒

𝑥+1

1+𝑦

𝜋 + 10

𝑥+1

𝑢 (𝑥, 1) = cos(4 ∗ 𝑥 ∗ 1) ∗ sin(2 ∗ 𝑥) + 𝑒

𝜋 + 10 = 𝑐𝑜𝑠4𝑥 𝑠𝑖𝑛2𝑥 + 𝑒

𝜋 + 10

То функция 𝑢(𝑥, 𝑦) = 𝑐𝑜𝑠4𝑥𝑦 𝑠𝑖𝑛2𝑥 + 𝑒

𝑥+𝑦

𝜋 + 10 является решением краевой задачи

−𝑢′′ = 16𝑦2𝑐𝑜𝑠4𝑥𝑦 𝑠𝑖𝑛2𝑥 + 16𝑦 𝑠𝑖𝑛4𝑥𝑦 𝑐𝑜𝑠2𝑥 + 4𝑐𝑜𝑠4𝑥𝑦 𝑠𝑖𝑛2𝑥 − 2𝑒

𝑥+𝑦

𝜋

+ 16𝑥2𝑠𝑖𝑛2𝑥 𝑐𝑜𝑠4𝑥𝑦,

𝑥 ∈ [0,1], 𝑦 ∈ [0,1]

𝜋2

𝑦

𝑢 (0, 𝑦) = 𝑒𝜋 + 10

𝑥

𝑢 (𝑥, 0) = 𝑒𝜋 + 10

𝑢 (1, 𝑦) = 𝑐𝑜𝑠4𝑦 𝑠𝑖𝑛2 + 𝑒

1+𝑦

𝜋 + 10

𝑥+1

𝑢 (𝑥, 1) = 𝑐𝑜𝑠4𝑥 𝑠𝑖𝑛2𝑥 + 𝑒 𝜋 + 10

Теперь эта задача может быть использована для проверки работоспособности численного метода решения краевой задачи, так как численное решение можно сравнить с аналитическим решением, которое известно.

# Решение тестовых примеров

# Решение первого тестового примера

График численного решения тестового примера методом конечных элементов с шагом h = 0.25

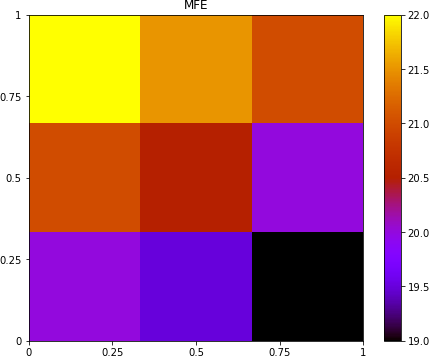


График проинтерполированного решения с шагом h = 0.001

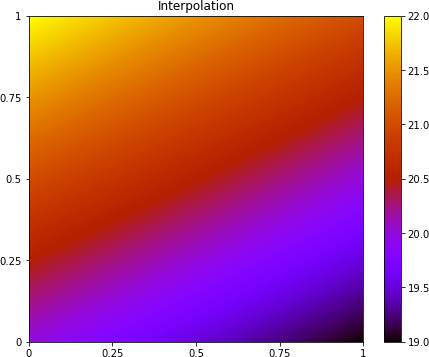
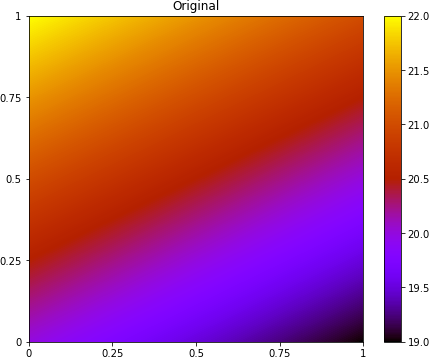
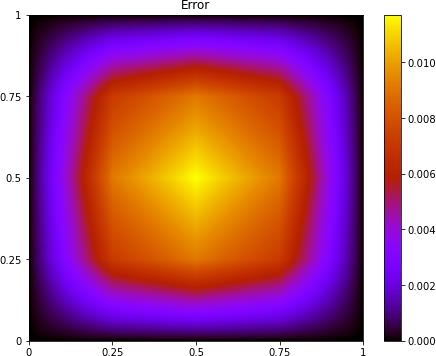


График аналитического решения с шагом h = 0.001



График погрешности с шагом h = 0.001

# Решение второго тестового примера

График численного решения тестового примера методом конечных элементов с шагом h = 0.25

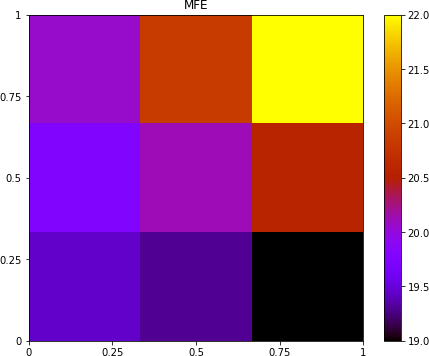


График проинтерполированного решения с шагом h = 0.001

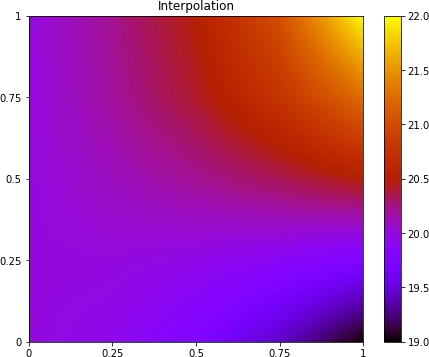


График аналитического решения с шагом h = 0.001

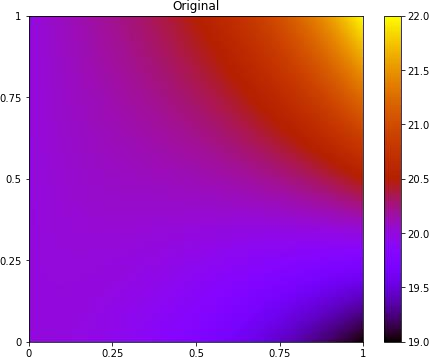
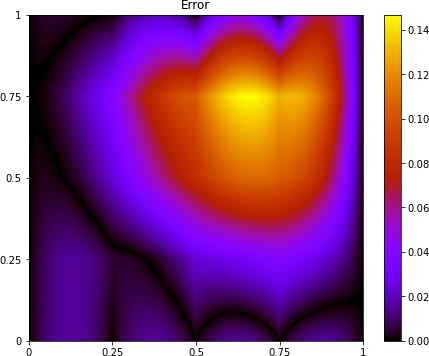


График погрешности с шагом h = 0.001



# Решение третьего тестового примера

График численного решения тестового примера методом конечных элементов с шагом h = 0.25

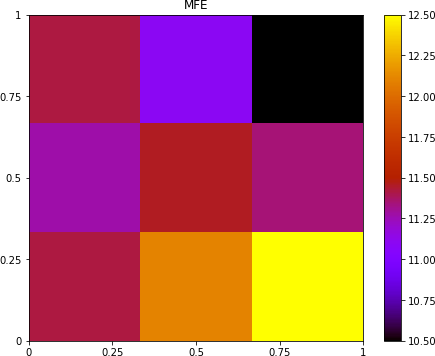


График проинтерполированного решения с шагом h = 0.001

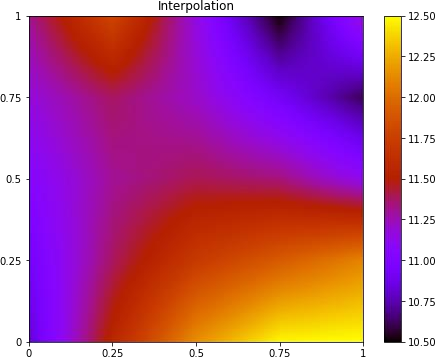


График аналитического решения с шагом h = 0.001

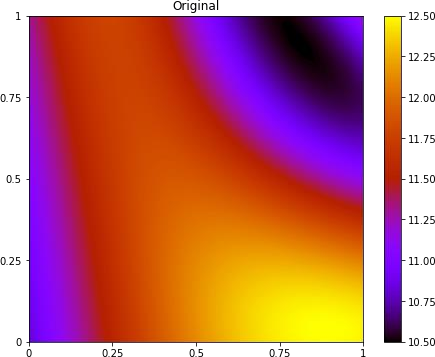
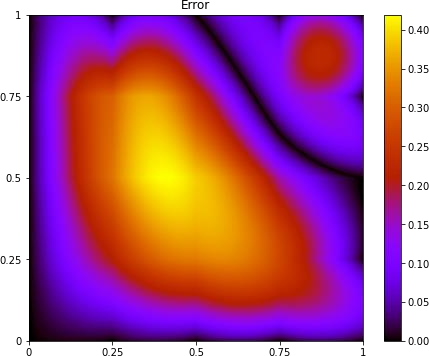


График погрешности с шагом h = 0.001



# Заключение

Мы рассмотрели решение задачи Дирихле в сложной области методом конечных элементов. Можем считать, что метод конечных элементов реализован успешно, т.к. даже после билинейной интерполяции графики численных решений тестовых примеров имеют хорошую точность, и, как следствие, небольшую погрешность относительно аналитических решений.

# Список литературы

1. Марчук Г. И., Агошков В. И. Введение в проекционно-сеточные методы. — 1981.
2. Амосов А. А., Дубинский Ю. А., Копченова Н. В. Вычислительные методы. — 2014.
3. Казёнкин К. О., Вестфальский А. Е., Амосова О. А. Численное решение стационарных уравнений математической физики. – 2018.
4. Самарский А. А., Вабищевич П. Н. Вычислительная теплопередача. — 2003.

# 8. Приложение

* + 1. **import** numpy **as** np
    2. **import** matplotlib
    3. **from** matplotlib **import** pyplot **as** plt
    4. **from** scipy.integrate **import** dblquad **as** Double 5.

6. **def** MFE(f, border):

7. h=0,25

8. n = int(1 / h) - 1

9. A = np.zeros((n\*\*2, n\*\*2)) 10.

1. **for** i **in** range (n):
2. **for** j **in** range (n):

13. A[i\*n + j, i\*n +j] = 4 / h\*\*2

14. **(** (i == 0 **and** j == 0):

15. A[i, j + 1] = -1 / h\*\*2

16. A[i, (i + 1) \* n + j] = -1 / h\*\*2

17. **(** ( == 0 **and** j == n - 1):

18. A[j, j - 1] = -1 / h\*\*2

19. A[j, n + j] = -1 / h\*\*2

20. **(** ( == n - 1 **and** j == 0):

21. A[i \* n, i \* n + 1] = -1 1 / h\*\*2 22. A[i \* n, (i - 1) \* n ] = -1 1 / h\*\*2

23. **(** ( == n - 1 **and** j == n - 1):

24. A[i \* n + j, i \* n + j - 1] = -1 / h\*\*2 25. A[i \* n + j, (i - 1) \* n + j] = -1 / h\*\*2 26. **(** ( == 0):

27. A[j, j - 1] = -1 / h\*\*2

28. A[j, j + 1] = -1 / h\*\*2

29. A[j, (i + 1) \* n + j] = -1 / h\*\*2

30. **(** (j == 0):

31. A[i \* n, (i - 1) \* n] = -1 / h\*\*2

32. A[i \* n, (i + 1) \* n] = -1 / h\*\*2

33. A[i \* n, i \* n + (j + 1)] = -1 / h\*\*2

34. **(** ( == n - 1):

35. A[i \* n + j, i \* n + j - 1] = -1 / h\*\*2

36. A[i \* n + j, i \* n + j + 1] = -1 / h\*\*2 37. A[i \* n + j, (i - 1) \* n + j] = -1 / h\*\*2 38. **(** (j == n - 1):

39. A[i \* n + j, (i - 1) \* n + j] = -1 / h\*\*2

40. A[i \* n + j, (i + 1) \* n + j] = -1 / h\*\*2

41. A[i \* n + j, i \* n + (j - 1)] = -1 / h\*\*2

42. **else**:

43. A[i \* n + j, (i - 1) \* n + j] = -1 / h\*\*2

44. A[i \* n + j, (i + 1) \* n + j] = -1 / h\*\*2

45. A[i \* n + j, i \* n + (j - 1)] = -1 / h\*\*2

46. A[i \* n + j, i \* n + (j + 1)] = -1 / h\*\*2

47.

48.

49. **def** Integrate(f, h, inti, intj):

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 50. | **def** | phi1(y, | x): |  | | | |
| 51. |  | **1** 1/h \* | (1 - x / | h | + | inti) |  |
| 52. | **def** | phi2(y, | x): |  |  |  |  |
| 53. |  | **1** 1 / h | \* f1(x, | y) | \* | (1 - (y | / h - intj)) |
| 54. | **def** | phi3(y, | x): |  |  |  |  |
| 55. |  | **1** 1 / h | \* f1(x, | y) | \* | (1 + (x | / h - inti) - (y / h - intj)) |
| 56. | **def** | phi4(y, | x): |  |  |  |  |
| 57. |  | **1** 1 / h | \* f1(x, | y) | \* | (1 + (x | / h - inti)) |
| 58. | **def** | phi5(y, | x): |  |  |  |  |
| 59. |  | **1** 1 / h | \* f1(x, | y) | \* | (1 + (y | / h - intj)) |
| 60. | **def** | phi6(y, | x): |  |  |  |  |

61. **1** 1 / h \* f1(x, y) \* (1 - (x / h - inti) + (y / h - intj))

1. I1 = Double(phi1, inti \* h, (inti + 1) \* h, intj \* h, **lambda** x: intj \* h + (x - inti \* h))[0]
2. I2 = Double(phi2, inti \* h, (inti + 1) \* h, **lambda** x: intj \* h + (x - inti \* h), (intj + 1) \* h)[0]
3. I3 = Double(phi3, (inti - 1) \* h, inti \* h, intj \* h, **lambda** x: intj \* h + (x - (inti - 1) \* h))[0]
4. I4 = Double(phi4, (inti - 1) \* h, inti \* h, **lambda** x: (intj - 1) \* h + (x - (inti - 1) \* h), intj \* h)[0]
5. I5 = Double(phi5, (inti - 1) \* h, inti \* h, (intj - 1) \* h, **lambda** x: (intj - 1) \* h + (x - (inti - 1) \* h))[0]
6. I6 = Double(phi6, inti \* h, (inti + 1) \* h, **lambda** x: (intj - 1) \* h + (x - inti \* h), intj \* h)[0] 68. **1** 1/h \* (I1 + I2 + I3 + I4 + I5 + I6)

69.

1. **def** Right(f, h, border):
2. b = np.zeros(n\*\*2)
3. **for** j **in** range(1, n + 1):
4. **for** i **in** range(1, n + 1):

74. **if** (i == 1 **and** j == 1):

75. b[(i - 1) \* n + (j - 1)] = Integrate(f, h, i, j) + 1 / h\*\*2 \* border(0, i \* h)

+ 1 / h\*\*2 \* border(j \* h, 0)

76. **elif** (i == 1 **and** j == n):

77. b[(i - 1) \* n + (j - 1)] = Integrate(f, h, i, j) + 1 / h\*\*2 \* border(1, i \* h)

+ 1 / h\*\*2 \* border(j \* h, 0)

78. **elif** (i == n **and** j == 1):

79. b[(i - 1) \* n + (j - 1)] = Integrate(f, h, i, j) + 1 / h\*\*2 \* border(0, i \* h)

+ 1 / h\*\*2 \* border(j \* h, 1)

80. **elif** (i == n **and** j == n):

81. b[(i - 1) \* n + (j - 1)] = Integrate(f, h, i, j) + 1 / h\*\*2 \* border(1, i \* h)

+ 1 / h\*\*2 \* border(j \* h, 1)

82. **elif** (i == 1):

83. b[(i - 1) \* n + (j - 1)] = Integrate(f, h, i, j)+ 1 / h\*\*2 \* border(j \* h, 0)

84. **elif** (j == 1):

85. b[(i - 1) \* n + (j - 1)] = Integrate(f, h, i, j)+ 1 / h\*\*2 \* border(0, i \* h)

86. **elif** (i == n):

87. b[(i - 1) \* n + (j - 1)] = Integrate(f, h, i, j)+ 1 / h\*\*2 \* border(j \* h, 1)

88. **elif** (j == n):

89. b[(i - 1) \* n + (j - 1)] = Integrate(f, h, i, j)+ 1 / h\*\*2 \* border(1, i \* h)

90. **else**:

91. b[(i - 1) \* n + (j - 1)] = Integrate(f, h, i, j)

92. **return** b 93.

1. rp = Right(f, h, border)
2. matrix = np.linalg.solve(A, rp)
3. matrix = np.reshape(matrix, (n, n)) 97.
4. **print**("MFE")
5. **print**(matrix)
6. **print**()
7. **print**() 102.

103. **return** matrix, h 104.

105. **def** Interpolation(matr, h, border):

106. n = int(1 / h) - 1

1. matrix = np.zeros((n + 2, n + 2))
2. **for** i **in** range(1, n + 1):
3. **for** j **in** range(1, n + 1):
4. matrix [i, j] = matr[i - 1, j - 1]
5. **for** i **in** range(n + 2):
6. matrix[i, 0] = border(0, i \* h)
7. matrix[i, n + 1] = border(1, i \* h)
8. **for** j **in** range(n + 2):
9. matrix[0, j] = border(j \* h, 0)
10. matrix[n + 1, j] = border(j \* h, 1)
11. **print**("MFE + board")
12. **print**(matrix)
13. **print**()
14. **print**() 121.

122. h\_int = 0.001

1. n\_int = int(1 / h\_int) - 1
2. matrix\_int = np.zeros((n + 2, n\_int + 2)) 125.
3. **for** i **in** range(n + 2):
4. sum = h\_int

128. j = 1

1. matrix\_int[i, 0] = matrix[i, 0]
2. matrix\_int[i, n\_int + 1] = matrix[i, n + 1]
3. **while** (sum <= 1):
4. lin = (matrix[i, j] - matrix[i, j - 1]) / h
5. **while** (sum <= h \* j):
6. matrix\_int[i, round(sum / h\_int)] =matrix[i, j - 1] + lin \* (sum - h \* (j - 1))
7. sum += h\_int

136. j +=1

137.

1. **print**("Intepolation Ox")
2. **print**(matrix\_int)
3. **print**()
4. **print**() 142.

143. matrix\_i = np.zeros((n\_int + 2, n\_int + 2)) 144.

1. **for** j **in** range(n\_int + 2):
2. sum = h\_int

147. i = 1

1. matrix\_i[0, j] = matrix\_int[0, j]
2. matrix\_i[n\_int + 1, j] = matrix\_int[n + 1, j]
3. **while** (sum <= 1):
4. lin = (matrix\_int[i, j] - matrix\_int[i - 1, j]) / h
5. **while** (sum <= h \* i):
6. matrix\_i[round(sum / h\_int), j] =matrix\_int[i - 1, j] + lin \* (sum - h \* (i - 1))
7. sum += h\_int

155. i +=1

156.

1. **print**("Intrpolation")
2. **print**(matrix\_i)
3. **print**()
4. **print**() 161.

162. **return**(matrix\_i) 163.

1. **def** DrawHeatMap(a, b\_min, b\_max):
2. plt.figure(figsize=(7.5, 6))
3. cmap = 'gnuplot'
4. plt.pcolormesh(a, cmap = cmap)
5. plt.title( "MFE" )

169. axes\_ticks = [0 \* len(a[0]), 0.25 \* len(a[0]), 0.5 \* len(a[0]), 0.75 \* len(a[0]), 1 \* len(a[0])]

170. axes\_labels = [0, 0.25, 0.5, 0.75, 1]

1. plt.xticks(ticks = axes\_ticks, labels = axes\_labels)
2. plt.yticks(ticks = axes\_ticks, labels = axes\_labels)
3. norm = matplotlib.colors.Normalize(vmin = b\_min, vmax = b\_max)
4. plt.colorbar(matplotlib.cm.ScalarMappable(norm = norm, cmap = cmap))
5. plt.show() 176.
6. **def** DrawInterpolation(a, b\_min, b\_max):
7. plt.figure(figsize=(7.5, 6))
8. cmap = 'gnuplot'
9. plt.pcolormesh(a, cmap = cmap)
10. plt.title( "Interpolation" )

182. axes\_ticks = [0 \* len(a[0]), 0.25 \* len(a[0]), 0.5 \* len(a[0]), 0.75 \* len(a[0]), 1 \* len(a[0])]

183. axes\_labels = [0, 0.25, 0.5, 0.75, 1]

1. plt.xticks(ticks = axes\_ticks, labels = axes\_labels)
2. plt.yticks(ticks = axes\_ticks, labels = axes\_labels)
3. norm = matplotlib.colors.Normalize(vmin = b\_min, vmax = b\_max)
4. plt.colorbar(matplotlib.cm.ScalarMappable(norm = norm, cmap = cmap))
5. plt.show() 189.

190. **def** DrawOriginal(func, b\_min, b\_max): 191. h=0.001

192. n = int(1 / h) - 1

1. matr = np.zeros((n + 2, n + 2))
2. **for** i **in** range(n + 2):
3. **for** j **in** range(n + 2):
4. matr[i, j] = func(i, j, h) 197.

198. **print**("Original")

|  |  |  |  |
| --- | --- | --- | --- |
| 199. | **print**(matr) | | |
| 200. | **print**() | | |
| 201. | **print**() | | |
| 202. |  | | |
| 203. | plt.figure(figsize=(7.5, 6)) | | |
| 204. | cmap = 'gnuplot' | | |
| 205. | plt.pcolormesh(matr, cmap = cmap) | | |
| 206. | plt.title( "Original" ) | | |
| 207. | axes\_ticks = [0 \* len(matr[0]), 0.25 \* len(matr[0]), 0.5 \* len(matr[0]), 0.75 \* len(matr[0]), | | |
| 1 \* len(matr[0])]  208. axes\_labels = [0, 0.25, 0.5, 0.75, 1]   1. plt.xticks(ticks = axes\_ticks, labels = axes\_labels) 2. plt.yticks(ticks = axes\_ticks, labels = axes\_labels) 3. norm = matplotlib.colors.Normalize(vmin = b\_min, vmax = b\_max) | | | |
| 212. | plt.colorbar(matplotlib.cm.ScalarMappable(norm = norm, cmap = cmap)) | |  |
| 213. | plt.show() | |  |
| 214. |  | |  |
| 215. | **return**(matr) | |  |
| 216.  217. **def** | DrawError(test, u): | |  |
| 218. | n = len(test[0]) | |  |
| 219. | error = np.zeros((n, n)) | |  |
| 220. | **for** i **in** range(n): | |  |
| 221. | **for** j **in** range(n): | |  |
| 222. | error[i, j] = np.abs(u[i, j] - test[i, j]) | |  |
| 223. |  | |  |
| 224. | **print**("Error") | |  |
| 225. | **print**(error) | |  |
| 226. | **print**() | |  |
| 227. | **print**() | |  |
| 228. |  | |  |
| 229. | plt.figure(figsize=(7.5, 6)) | |  |
| 230. | plt.pcolormesh(error, cmap = 'gnuplot') | |  |
| 231. | plt.title( "Error" ) | |  |
| 232. | axes\_ticks = [0 \* len(error[0]), 0.25 \* len(error[0]), 0.5 \* len(error[0]), | | 0.75 \* |
| len(error[0]), 1 \* len(error[0])] | | | |
| 233. | axes\_labels = [0, 0.25, 0.5, 0.75, 1] |  | |
| 234. | plt.xticks(ticks = axes\_ticks, labels = | axes\_labels) | |
| 235. | plt.yticks(ticks = axes\_ticks, labels = | axes\_labels) | |
| 236. | plt.colorbar() |  | |
| 237. | plt.show() |  | |
| 238. |  |  | |
| 239. **def** | f1(x, y): |  | |
| 240. | **return** 0 |  | |
| 241. |  |  | |
| 242. **def** | border1(x, y): |  | |
| 243. | **if** (x == 0): |  | |
| 244. | **return** 2 \* y + 20 |  | |
| 245. | **if** (y == 0): |  | |
| 246. | **return** -x + 20 |  | |
| 247. | **if** (x == 1): |  | |
| 248. | **return** 2 \* y + 19 |  | |
| 249. | **if** (y == 1): |  | |
| 250. | **return** -x + 22 |  | |
| 251. |  |  | |
| 252. **def** | u1(i\_1, j\_1, h\_1): |  | |
| 253. | **return** -(j\_1 \* h\_1) + 2 \* (i\_1 \* h\_1) + | 20 | |
| 254. |  |  | |

255. min1 = 19

256. max1 = 22 257.

1. test1, h1 = MFE(f1, border1)
2. DrawHeatMap(test1, min1, max1)
3. inter1 = Interpolation(test1, h1, border1)
4. DrawInterpolation(inter1, min1, max1)
5. orig1 = DrawOriginal(u1, min1, max1)
6. DrawError(inter1, orig1) 264.
7. **def** f2(x, y):
8. **return** -12 \* y \* x + 2
9. **def** border2(x, y):

|  |  |
| --- | --- |
| 269. | **if** (x == 0): |
| 270. | **return** 20 |
| 271. | **if** (y == 0): |
| 272. | **return** -x\*\*2 + 20 |
| 273. | **if** (x == 1): |
| 274. | **return** 3 \* y + 19 |
| 275. | **if** (y == 1): |
| 276. | **return** 2\*x\*\*3 - x\*\*2 + x + 20 |
| 277. |  |

278. **def** u2(i\_2, j\_2, h\_2):

279. **return** 2 \* (j\_2 \* h\_2)\*\*3 \* (i\_2 \* h\_2) + (j\_2 \* h\_2) \* (i\_2 \* h\_2 - j\_2 \* h\_2) + 20 280.

281. min2 = 19

282. max2 = 22 283.

1. test2, h2 = MFE(f2, border2)
2. DrawHeatMap(test2, min2, max2)
3. inter2 = Interpolation(test2, h2, border2)
4. DrawInterpolation(inter2, min2, max2)
5. orig2 = DrawOriginal(u2, min2, max2)
6. DrawError(inter2, orig2) 290.

291. **def** f3(x, y):

292. **return** -(-16 \* y \* np.cos(2 \* x) \* np.sin(4 \* x \* y) - ((16 \* np.pi \* np.pi \* y \* y + 4 \* np.pi \* np.pi) \* np.sin(2 \* x) \* np.cos(4 \* y \* x)) / (np.pi \* np.pi) + np.exp((x + y) / np.pi) / np.pi + np.exp((x + y) / np.pi) / (np.pi \* np.pi) - 16 \* x \* x \* np.sin(2 \* x) \* np.cos(4 \* x \* y))

293.

294. **def** border3(x, y):

295. **if** (x == 0):

296. **return** np.exp(y / np.pi) + 10 297. **if** (y == 0):

298. **return** np.sin(2 \* x) + np.exp(x / np.pi) + 10 299. **if** (x == 1):

300. **return** np.sin(2) \* np.cos(4 \* y) + np.exp((y + 1) / np.pi) + 10 301. **if** (y == 1):

302. **return** np.sin(2 \* x) \* np.cos(4 \* x) + np.exp((x + 1) / np.pi) + 10 303.

304. **def** u3(i\_3, j\_3, h\_3):

305. **return** np.cos(4 \* (j\_3 \* h\_3) \* (i\_3 \* h\_3)) \* np.sin(2 \* (j\_3 \* h\_3)) + np.exp(((j\_3 \* h\_3) + (i\_3 \* h\_3)) / np.pi) + 10

306.

307. min3 = 10.5

308. max3 = 12.5

309.

1. test3, h3 = MFE(f3, border3)
2. DrawHeatMap(test3, min3, max3)
3. inter3 = Interpolation(test3, h3, border3)
4. DrawInterpolation(inter3, min3, max3)
5. orig3 = DrawOriginal(u3, min3, max3)
6. DrawError(inter3, orig3)