НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ
"МЭИ"

Курсовая работа по
Численным методам

Нахождение распределения температуры на прямоугольной металлической пластине с вырезом

Студент:
Николаев Ю. С.

Преподаватель:
Амосова О. А.

Москва, 2021

Содержание

1 Введение 2
1.1 Задание 2
1.2 Постановка задачи 3
1.3 Методы решения 3

2 Построение тестовых примеров 6
2.1 Тестовый пример №1 7
2.2 Тестовый пример №2 9
2.3 Анализ тестовых примеров 11

3 Решение задачи 12
3.1 Вариант №1 12
3.2 Вариант №2 16
3.3 Вариант №3 20
3.4 Вывод 22

4 Приложение 23
4.1 Код для тестовых примеров 23
4.2 Код основной программы 26

Список литературы 29

1 Введение

1.1 Задание

Задание 3.1. Прямоугольная металлическая пластина с вырезом используется как теплоотводящий элемент. В угловом вырезе пластины (границы G_2 и G_3) расположен источник тепла. Распределение температуры $T(x, y)$ по площади пластины описывается уравнением Лапласа:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

Найдите распределение $T(x, y)$. Размеры A, B, C и граничные условия даны в таблице.

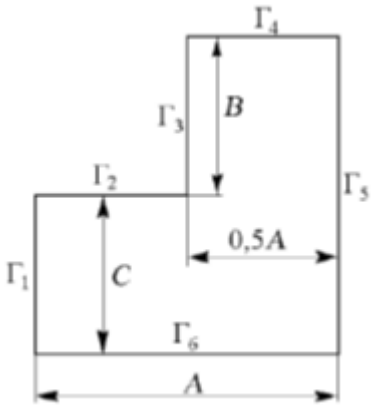

Рис. 1: Пластина

Параметр	Величина
A, мм	200
B, мм	45
C, мм	65
$T(G_1),^\circ C$	30
$T(G_2),^\circ C$	60
$T(G_3),^\circ C$	60
$T(G_4),^\circ C$	30
$T(G_5),^\circ C$	20
$T(G_6),^\circ C$	20

1.2 Постановка задачи

Во многих случаях для описания физических процессов используют уравнения с частными производными. В нашем случае мы проводим анализ статических тепловых деформаций, который описывается уравнением Лапласа:

$$\Delta T = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

Это эллиптический тип уравнений. В такой задаче используют только граничные условия по координатам x, y, а саму задачу называют краевой. У нас краевое условие задает распределение функции на границе, данное условие принято называть условием Дирихле. Таким образом, мы пришли к тому, что нужно найти решение уравнения Лапласа $\Delta T = 0$, удовлетворяющее условиям, заданным в таблице:

Условие	x	y
$T_{G_1} = 30$	0	$0 \le y \le C$
$T_{G_2} = 60$	$0 \le x \le \frac{A}{2}$	C
$T_{G_3} = 60$	$\frac{A}{2}$	$0 \le y \le C$
$T_{G_4} = 30$	$0 \le x \le A$	$C + B$
$T_{G_5} = 20$	A	$0 \le y \le C + B$
$T_{G_6} = 20$	$0 \le x \le A$	0

1.3 Методы решения

Данное дифференциальное уравнение будем решать часто используемым методом - конечных разностей. Этот метод заключается в том, что дифференциальное уравнение в частных производных заменяется соответствующей ему системой алгебраических уравнений. Решение этой системы дает приближенное решение для искомой функции.

Преобразование уравнения эллиптического типа для двумерной задачи производится путем замены в нем производных $\frac{\partial^2 T}{\partial x^2}$ и $\frac{\partial^2 T}{\partial y^2}$ конечно-разностными формулами:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \Rightarrow$$

$$\frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta x^2} + \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{\Delta y^2} = 0, \qquad (1)$$

В нашем случае $\Delta x = \Delta y = h$.

Уравнение (1) связывает между собой неизвестное значение $T_{i,j}$ с ее значениями в четырех соседних узлах. На сетке аппроксимации эти узлы образуют пятиточечный шаблон, позволяющий определить индексы в (1) для любого произвольно выбранного на сетке узла i, j.

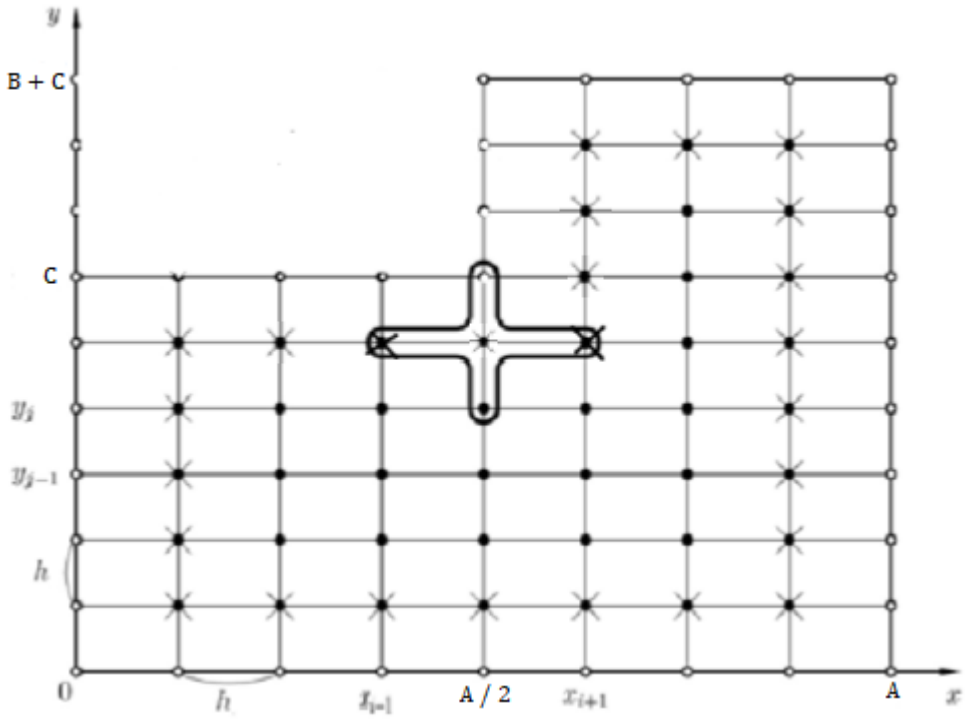

Рис. 2: Сетка аппроксимации

Записывая (1) для каждого узла $2 < i < n - 1$, $2 < j < m - 1$ и подставляя вместо i и j соответствующие номера, получим систему связанных уравнений. Количество уравнений будет равно количеству узлов, в которых необходимо найти неизвестные $T_{i,i}$. Иначе говоря, число неизвестных равно числу уравнений и система будет замкнутой. Значения функции T в узлах сетки, лежащих на границе рассматриваемой области, определяются заданными граничными условиями.

В качестве программируемого метода возьмем метод Гаусса-Зейделя, который является классическим итерационным методом. Мы будем вычислять новое приближение T^{k+1}, используя приближение с предыдущей итерации или, в случае первой итерации, используем начальное приближение T^k. Таким образом можем записать, как будем вычислять приближение:

$$\frac{T^k_{i+1,j} - 2T^{k+1}_{i,j} + T^{k+1}_{i-1,j}}{\Delta x^2} + \frac{T^k_{i,j+1} - 2T^{k+1}_{i,j} + T^{k+1}_{i,j-1}}{\Delta y^2} = 0, \Delta x^2 = \Delta y^2 = h^2$$

Выразим $T^{k+1}_{i,j}$:

$$T^{k+1}_{i,j} = \frac{T^k_{i+1,j} + T^{k+1}_{i-1,j} + T^k_{i,j+1} + T^{k+1}_{i,j-1}}{4}$$

Будем завершать процесс при достижении заданной точности ε, для этого используем разницу между двумя ближайшими итерациями в Евклидовой норме.

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} |T^{k+1}_i - T^k_i|^2}$$

Пример матрицы начального приближения (стоит отметить, что она зависит от геометрии пластины, шага и граничных условий):

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 30 & 30 & 30 & 30 & 20 \\ 0 & 0 & 0 & 0 & 60 & 0 & 0 & 0 & 20 \\ 0 & 0 & 0 & 0 & 60 & 0 & 0 & 0 & 20 \\ 60 & 60 & 60 & 60 & 60 & 0 & 0 & 0 & 20 \\ 30 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 20 \\ 30 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 20 \\ 20 & 20 & 20 & 20 & 20 & 20 & 20 & 20 & 20 \end{pmatrix}$$

2 Построение тестовых примеров

Тестовые примеры будем строить следующим образом: выберем некоторую функцию $u(x, y)$ и границы нашей фигуры (прямоугольника с вырезом) A, B, C. Для полученной функции вычислим граничные условия и функцию $f(x)$.

Уравнение Пуассона:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y)$$

Для того, чтобы получить приближенное решение, будем применять данные конечно-разностные формулы:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = f(x, y) \Rightarrow$$

$$\frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{h^2} + \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{h^2} = f(x, y)$$

Запишем, как будем вычислять приближение в программе (опять же, с помощью метода Гаусса-Зейделя):

$$\frac{T^k_{i+1,j} - 2T^{k+1}_{i,j} + T^{k+1}_{i-1,j}}{h^2} + \frac{T^k_{i,j+1} - 2T^{k+1}_{i,j} + T^{k+1}_{i,j-1}}{h^2} = f(i \cdot h, j \cdot h)$$

Выразим $T^{k+1}_{i,j}$:

$$T^{k+1}_{i,j} = \frac{-h^2 \cdot f(i \cdot h, j \cdot h) + T^k_{i+1,j} + T^{k+1}_{i-1,j} + T^k_{i,j+1} + T^{k+1}_{i,j-1}}{4}$$

Теперь построим несколько тестовых примеров.

2.1 Тестовый пример №1

Для построения тестового примера возьмем функцию:

$$u(x,y) = \frac{5}{12}\sin 4\pi x \cdot \sin \pi y$$

Граничные условия:

$$\begin{array}{l} \mathrm{A} = 1 \\ \mathrm{B} = 1 \\ \mathrm{C} = 1 \end{array} \Rightarrow \begin{array}{l} 0 \le x \le A = 1 \\ 0 \le y \le B + C = 2 \end{array}$$

Считаем производные второго порядка:

$$\frac{\partial^2 u}{\partial x^2} = -\frac{20\pi^2 \sin 4\pi x \cdot \sin \pi y}{3}$$

$$\frac{\partial^2 u}{\partial y^2} = -\frac{5\pi^2 \sin 4\pi x \cdot \sin \pi y}{12}$$

Получаем:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -\frac{85\pi^2 \sin 4\pi x \cdot \sin \pi y}{12} \Rightarrow f(x,y) = -\frac{85\pi^2 \sin 4\pi x \cdot \sin \pi y}{12}$$

Теперь запишем тестовый пример:

$$u(x,y) = \frac{5}{12}\sin 4\pi x \cdot \sin \pi y$$

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -\frac{85\pi^2 \sin 4\pi x \cdot \sin \pi y}{12}$$

$$\begin{array}{c} 0 \le x \le 1 \\ 0 \le y \le 2 \end{array}$$

$u(0,y) = 0$	$0 \le y \le 1$
$u(x,1) = 0$	$0 \le x \le 0.5$
$u(0.5,y) = 0$	$1 \le y \le 2$
$u(x,1) = 0$	$0.5 \le x \le 1$
$u(1,y) = 0$	$0 \le y \le 2$
$u(x,0) = 0$	$0 \le x \le 1$

Построим графики:

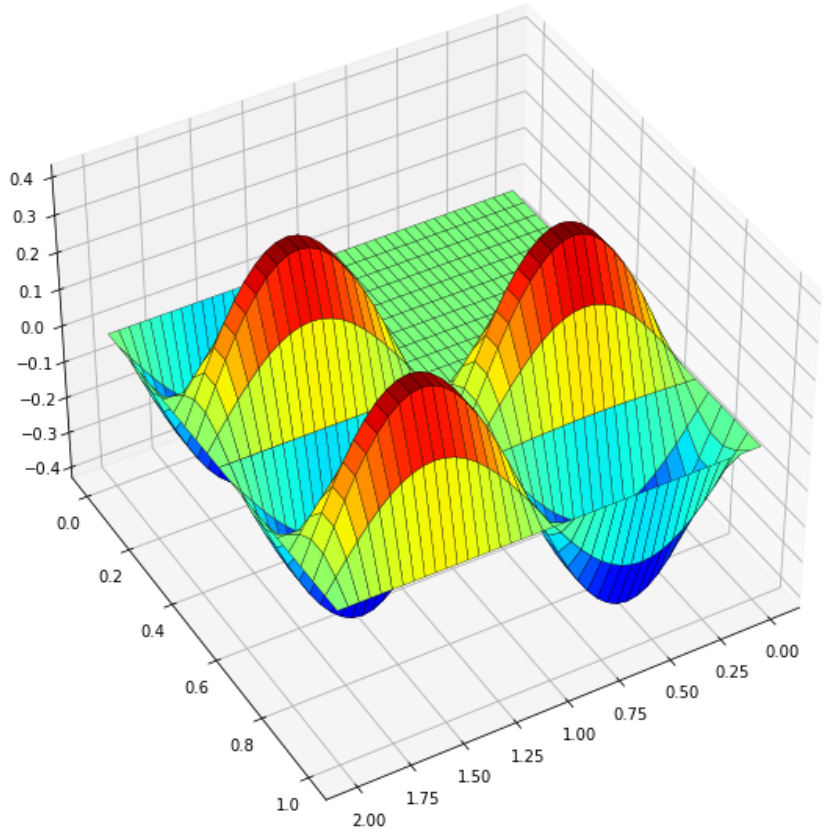

Рис. 3: Тест №1, приближенное решение

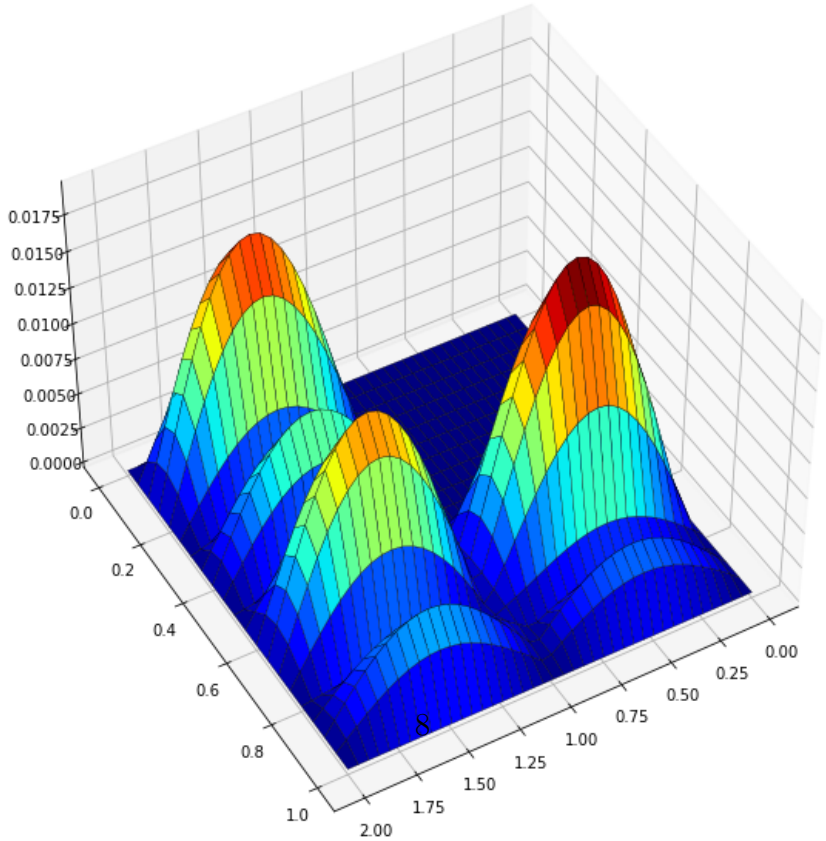

Рис. 4: Тест №1, погрешность

2.2 Тестовый пример №2

Для построения тестового примера возьмем функцию:

$$u(x,y) = \frac{1}{7}\sin 4\pi x \cdot \sin 2\pi y$$

Граничные условия:

$$\begin{array}{l} \text{A} = 1 \\ \text{B} = 0.5 \\ \text{C} = 0.5 \end{array} \Rightarrow \begin{array}{l} 0 \le x \le A = 1 \\ 0 \le y \le B + C = 1 \end{array}$$

Считаем производные второго порядка:

$$\frac{\partial^2 u}{\partial x^2} = -\frac{16\pi^2 \sin 4\pi x \cdot \sin 2\pi y}{7}$$

$$\frac{\partial^2 u}{\partial y^2} = -\frac{4\pi^2 \sin 4\pi x \cdot \sin 2\pi y}{7}$$

Получаем:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -\frac{20\pi^2 \sin 4\pi x \cdot \sin 2\pi y}{7} \Rightarrow f(x,y) = -\frac{20\pi^2 \sin 4\pi x \cdot \sin 2\pi y}{7}$$

Теперь запишем тестовый пример:

$$u(x,y) = \frac{1}{7}\sin 4\pi x \cdot \sin 2\pi y$$

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -\frac{20\pi^2 \sin 4\pi x \cdot \sin 2\pi y}{7}$$

$$\begin{array}{c} 0 \le x \le 1 \\ 0 \le y \le 1 \end{array}$$

$$\begin{array}{cc} u(0,y) = 0 & 0 \le y \le 0.5 \\ u(x,1) = 0 & 0 \le x \le 0.5 \\ u(0.5,y) = 0 & 0.5 \le y \le 1 \\ u(x,2) = 0 & 0.5 \le x \le 1 \\ u(1,y) = 0 & 0 \le y \le 1 \\ u(x,0) = 0 & 0 \le x \le 1 \end{array}$$

Построим графики:

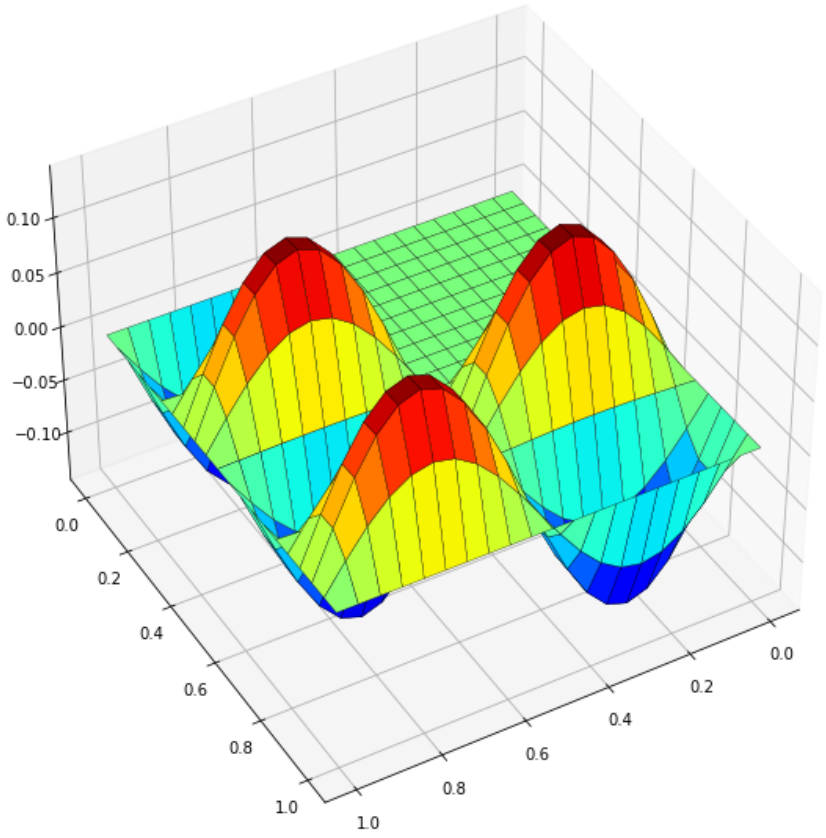

Рис. 5: Тест №2, приближенное решение

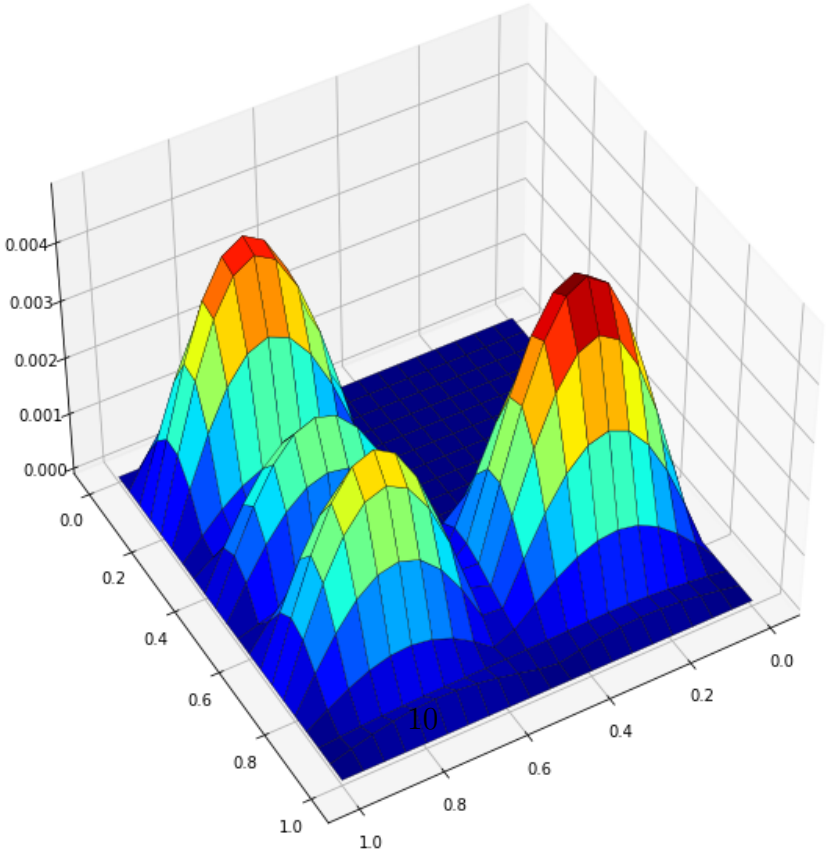

Рис. 6: Тест №2, погрешность

2.3 Анализ тестовых примеров

Были составлены два тестовых примера для решения задачи Дирихле в прямоугольнике с вырезом. При решении задачи были использованы методы, о которых мы говорили в пункте "Методы решения". В обоих примерах шаг сетки был равен $h = 0.05$, а $\varepsilon = 0.001$. Для наглядности были построены графики приближенного решения и погрешности, по которым видно, что задача решается верно. Погрешность была посчитана как модуль разности полученного и точного решения.

3 Решение задачи

Мы протестировали выбранные методы решения задачи, теперь будем решать нашу основную задачу - находить распределение температуры на прямоугольной пластине с вырезом.

У нас заданы начальные данные, поэтому для решения задачи нам нужно лишь подбирать шаг h и точность ε. Сделаем несколько вариантов и занесем их в таблицу, помимо этого будем выводить нужные нам графики.

3.1 Вариант №1

Пусть $h = 10mm$, а вот ε будем брать несколько - 0.01 и 0.001.
Получили, что количество итераций по норме:

$iterations = 101$, где $\varepsilon = 0.01$
$iterations = 138$, где $\varepsilon = 0.001$

Выведем графики изменения нормы и 2D, 3D графики распределения температуры на пластине:

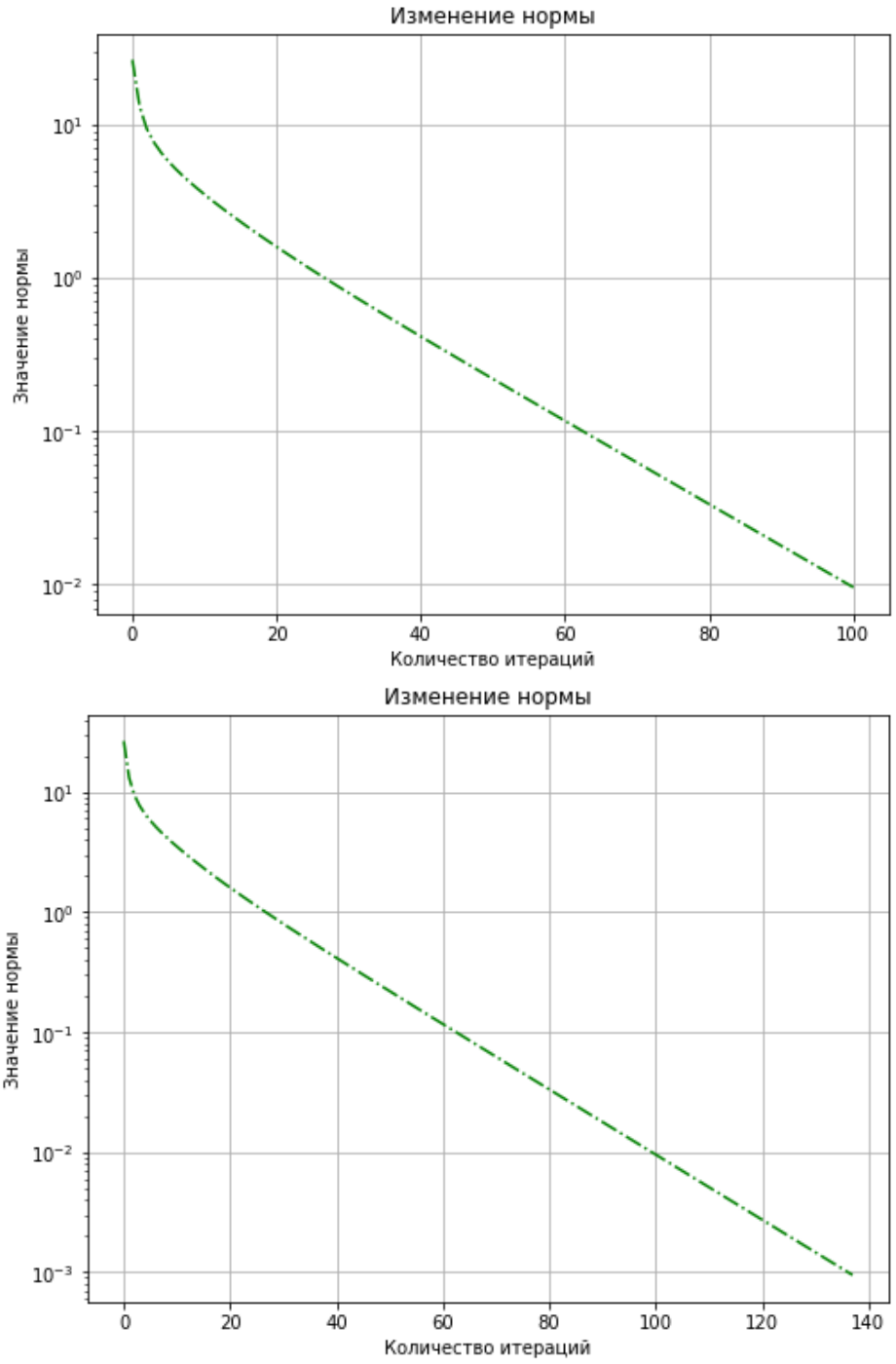

Рис. 7: Графики изменения нормы

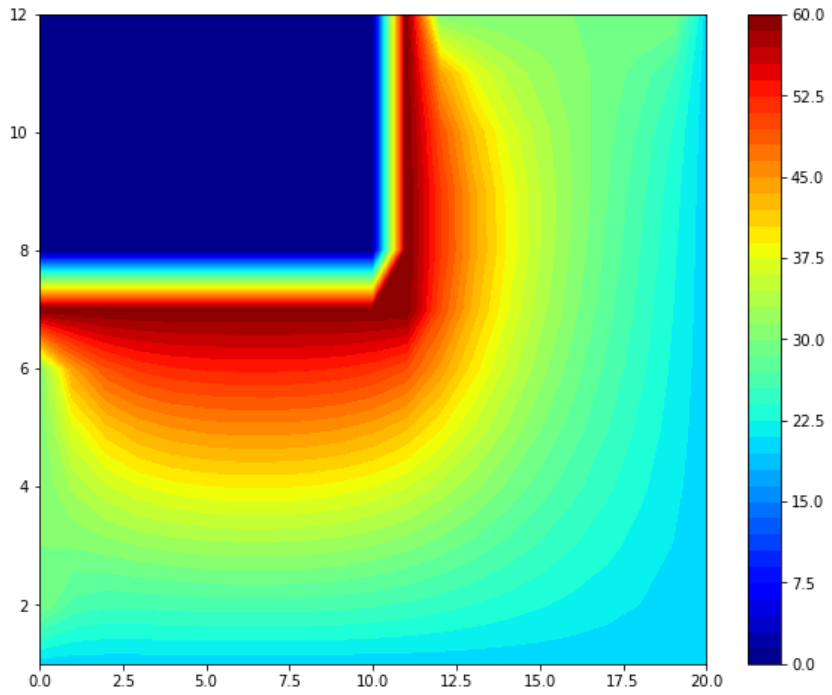

Рис. 8: График 2D распределения температуры, eps = 0.01

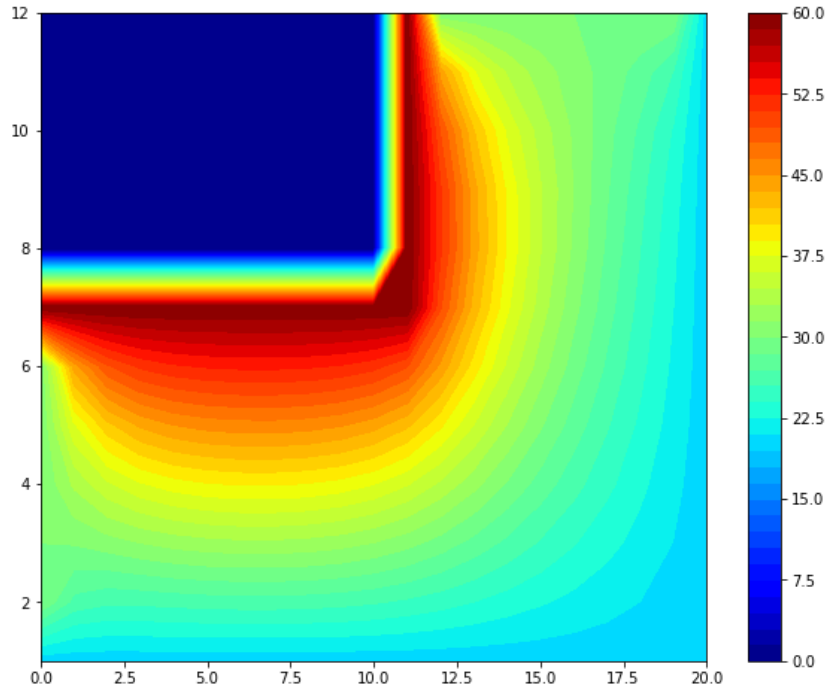

Рис. 9: График 2D распределения температуры, eps = 0.001

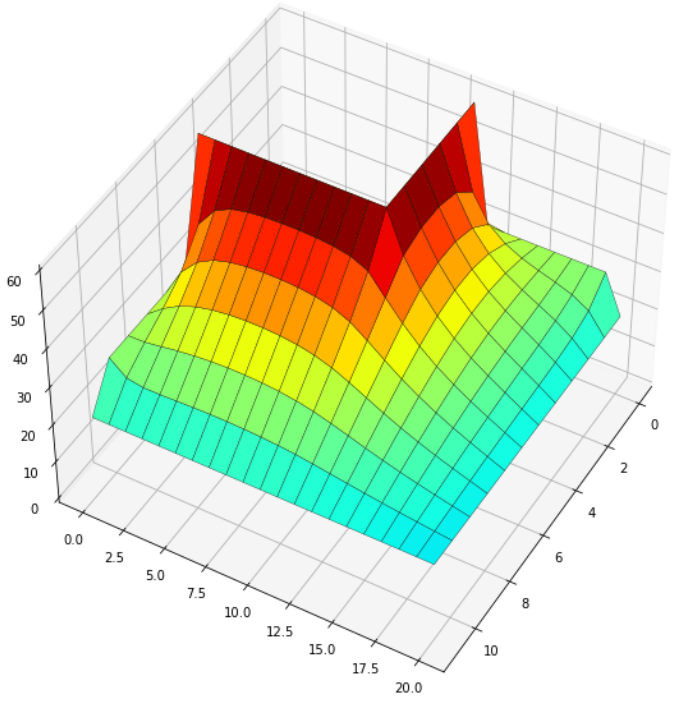

Рис. 10: График 3D распределения температуры, eps = 0.01

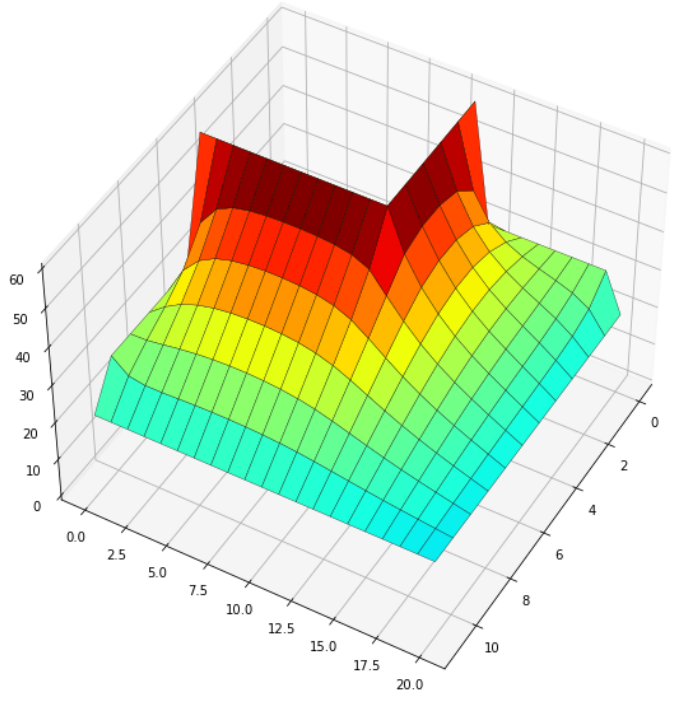

Рис. 11: График 3D распределения температуры, eps = 0.001

3.2 Вариант №2

Пусть $h = 5mm$, а вот ε будем брать несколько - 0.01 и 0.001.
Получили, что количество итераций по норме:

$iterations = 356$, где $\varepsilon = 0.01$
$iterations = 514$, где $\varepsilon = 0.001$

Выведем графики изменения нормы и 2D, 3D графики распределения температуры на пластине:

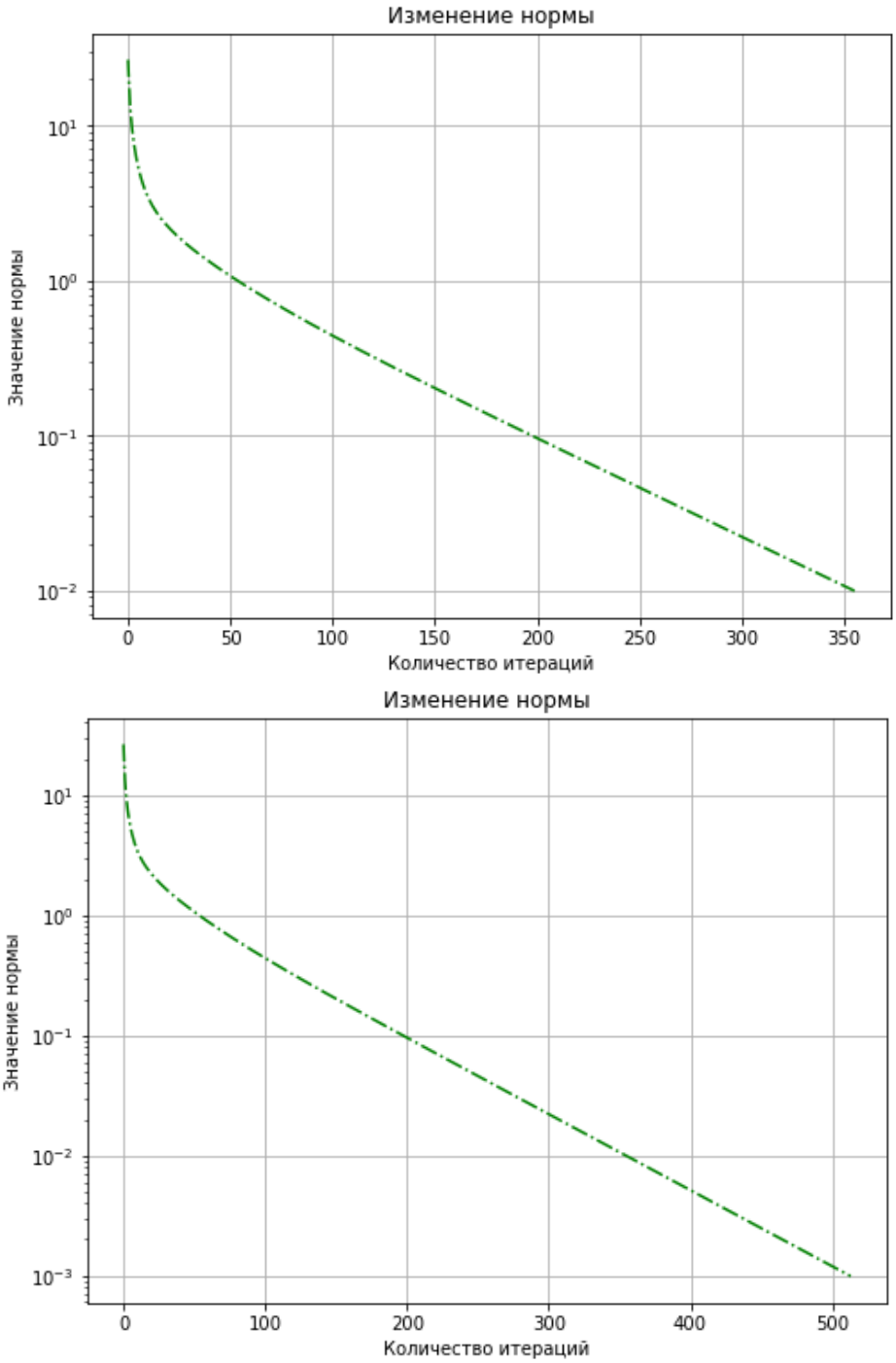

Рис. 12: Графики изменения нормы

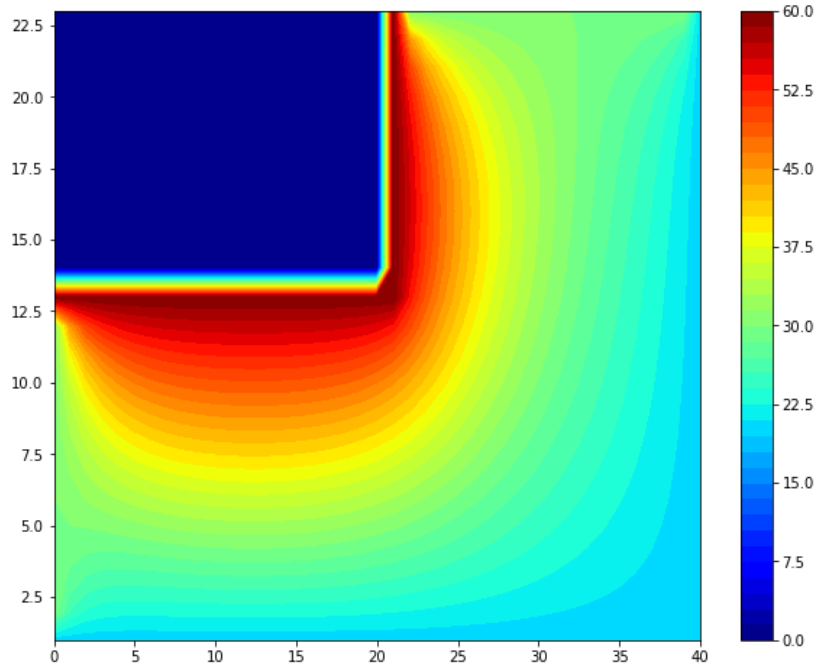

Рис. 13: График 2D распределения температуры, eps = 0.01

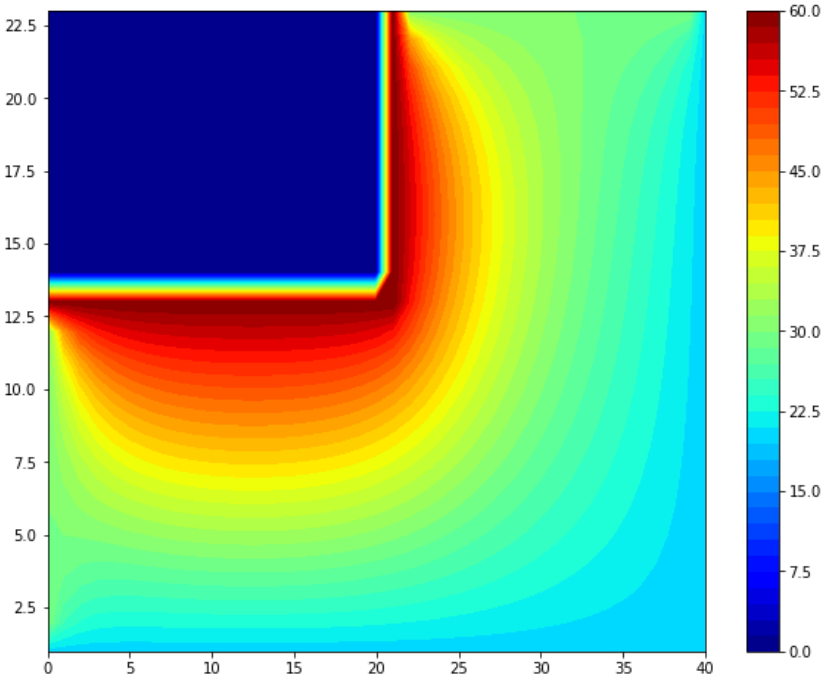

Рис. 14: График 2D распределения температуры, eps = 0.001

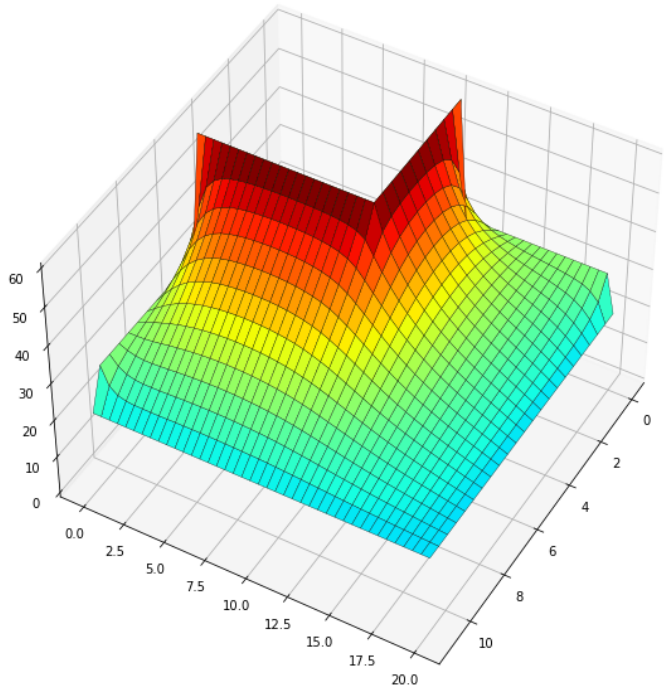

Рис. 15: График 3D распределения температуры, eps = 0.01

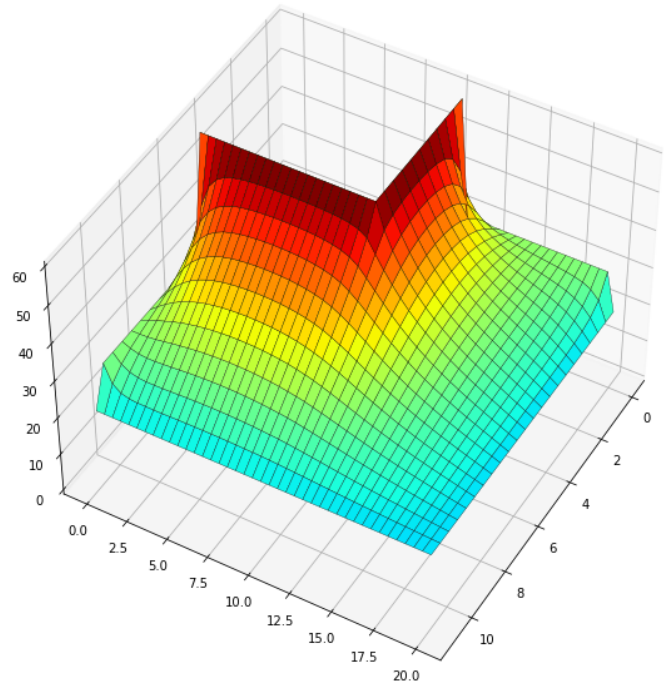

Рис. 16: График 3D распределения температуры, eps = 0.001

3.3 Вариант №3

Пусть $h = 1mm$, а вот ε будем брать несколько - 0.01 и 0.001. Но по ходу решения задачи получили, что уже при $\varepsilon = 0.01$ количество итераций по норме достаточно большое (5078 итераций). Поэтому остановимся на выборе $\varepsilon = 0.01$.

Выведем графики изменения нормы и 2D, 3D графики распределения температуры на пластине:

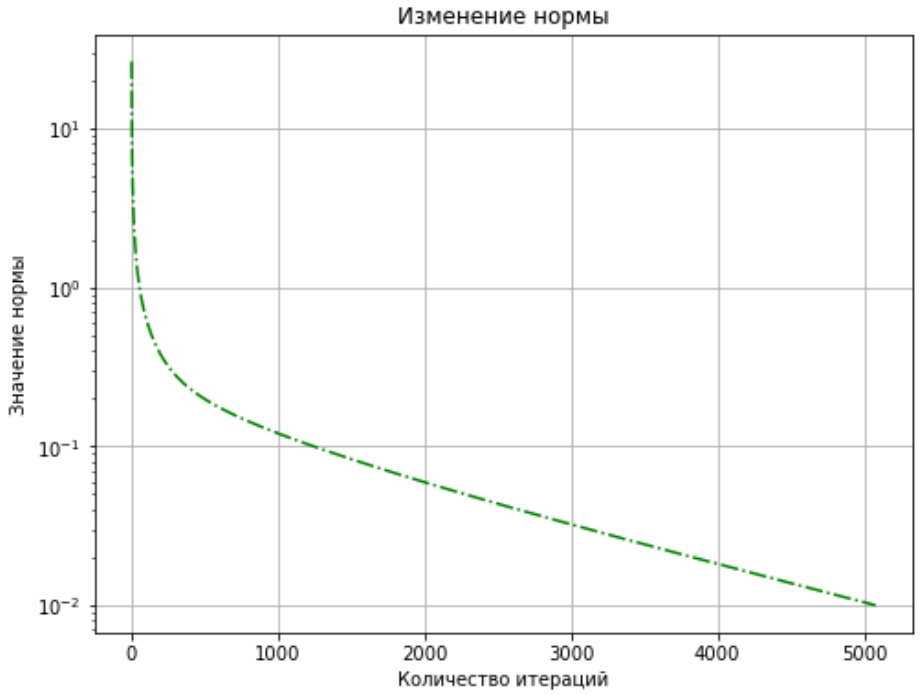

Рис. 17: График изменения нормы

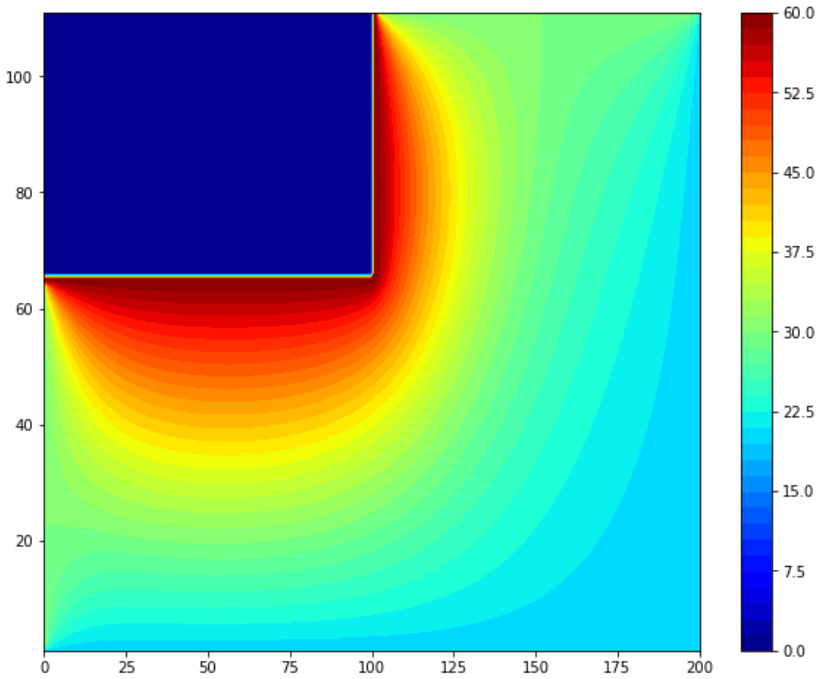

Рис. 18: График 2D распределения температуры, eps = 0.01

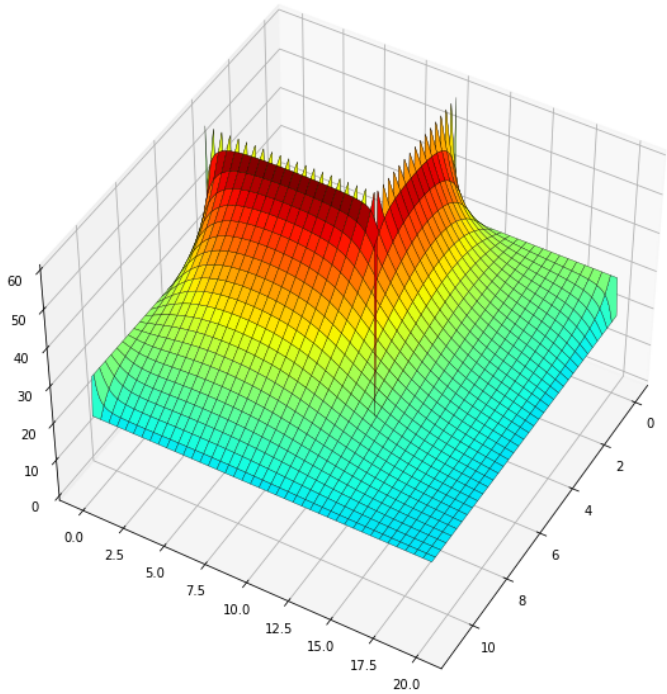

Рис. 19: График 3D распределения температуры, eps = 0.01

3.4 Вывод

Составим таблицу вариантов выбора шага и точности и получаемое количество итераций:

h, mm	eps	iterations
10	0.01	101
10	0.001	138
5	0.01	356
5	0.001	514
1	0.01	5078

По графикам прослеживается, что при уменьшении шага распределение температуры происходит более плавно, что неудивительно. А вот при уменьшении точности не прослеживается каких-либо улучшений. На мой взгляд, в данном случае оптимальнее всего взять шаг $h = 5mm$ и точность $\varepsilon = 0.01$, такое решение не нагружает систему, быстро вычисляется и дает хорошее представление о распределении температуры на пластине.

4 Приложение

4.1 Код для тестовых примеров

```
import numpy as np
from matplotlib import pyplot as plt

A_test = 1
B_test = 0.5
C_test = 0.5
h_test = 0.05

x_count_test = int(A_test / h_test) + 1
y_count_test = int((C_test + B_test) / h_test) + 1

boundary_values_test = np.array([0., 0., 0., 0., 0., 0.])

def initialize_matrix(A, B, C, boundary_values, x_count, y_count
    ):
    matrix = np.zeros((y_count, x_count))

    matrix[int(C / h) + 1:, 0] = boundary_values[0]
    matrix[int(C / h) + 1, :int(A / (2 * h)) + 1] =
        boundary_values[1]
    matrix[:int(C / h) + 2, int(A / (2 * h)) + 1] =
        boundary_values[2]
    matrix[0, int(A / (2 * h)) + 2:] = boundary_values[3]
    matrix[:, int(A /  h)] = boundary_values[4]
    matrix[int((C + B) / h), :] = boundary_values[5]

    return matrix

matrix_test = initialize_matrix(A_test, B_test, C_test,
    boundary_values_test, x_count_test, y_count_test)

def test_f(x, y):
#     return -85 * (np.pi ** 2) * np.sin(4 * np.pi * x) * np.sin
    (np.pi * y) / 12
    return -20 * (np.pi ** 2) * np.sin(4 * np.pi * x) * np.sin(2
        * np.pi * y) / 7

def exact_f(x, y):
#     return 5 * np.sin(4 * np.pi * x) * np.sin(np.pi * y) / 12
```

```
    return np.sin(4 * np.pi * x) * np.sin(2 * np.pi * y) / 7

def Norm(T_k1, T_k):
        diff = np.abs(T_k1 - T_k)
        return np.sqrt(np.sum(diff ** 2) / len(diff))

def T1_exact(A, B, C, h, T0, y_count, x_count, f):

    T1_exact = np.copy(T0)
    for i in range(1, y_count):
        for j in range(1, x_count):
            if j > (int(A / (2 * h))) or i > (int(C / h)):
                T1_exact[i, j] = f(j * h, i * h)

    return T1_exact

def exact_solution(A, B, C, h, y_count, x_count, eps, matrix, f)
    :
    T_k = T_k1 = matrix
    T_k1 = T1_exact(A, B, C, h, T_k, y_count, x_count, f)

    return T_k1

def T1_test(A, B, C, h, T0, y_count, x_count, f):
        T1_test = np.copy(T0)
        for i in range(1, y_count):
            for j in range(1, x_count):
                if j > (int(A / (2 * h))) or i > (int(C / h)):
                    T1_test[i, j] = (-f(j * h, i * h) * (h ** 2)
                        + T0[i - 1, j] + T1_test[i + 1, j] +
                                    T0[i, j + 1] + T1_test[i, j
                                        - 1]) / 4

        return T1_test

def method_test(A, B, C, h, y_count, x_count, eps, matrix, f):
    T_k = T_k1 = matrix
    iterations = 0
    norm = 1000
    norms = []
    while norm >= eps:
        iterations += 1
        T_k = T_k1
```

```
        T_k1 = T1_test(A, B, C, h, T_k, y_count, x_count, f)
        norm = Norm(T_k1, T_k)
        norms.append(norm)
    return T_k1, iterations, np.array(norms)

eps = 0.001

T_test, iterations_test, norms_test = method_test(A_test, B_test
    , C_test, h_test, y_count_test - 1, x_count_test - 1,
                                                    eps,
                                                       matrix_test
                                                       , test_f)

exact = exact_solution(A_test, B_test, C_test, h_test,
    y_count_test - 1, x_count_test - 1, eps, matrix_test, exact_f
    )

xgrid_test, ygrid_test = np.meshgrid(np.linspace(0., A_test,
    x_count_test), np.linspace(0., B_test + C_test , y_count_test
    ))

# test
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(projection='3d')
ax.plot_surface(ygrid_test, xgrid_test, T_test, cmap = 'jet',
    linewidth=0.3, edgecolors='k')
ax.view_init(elev=40, azim=60)

# exact
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(projection='3d')
ax.plot_surface(ygrid_test, xgrid_test, exact, cmap = 'jet',
    linewidth=0.3, edgecolors='k')
ax.view_init(elev=40, azim=60)

# difference
diff_test = np.abs(T_test - exact)

fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(projection='3d')
ax.set_title('                        ')
ax.plot_surface(ygrid_test, xgrid_test, diff_test, cmap = 'jet',
    linewidth=0.3, edgecolors='k')
```

```
ax.view_init(elev=45, azim=60)

plt.show()
```

4.2 Код основной программы

```
import numpy as np
from matplotlib import pyplot as plt

A = 20.
B = 6.5
C = 4.5
h = 0.5

boundary_values = np.array([30., 60., 60., 30., 20., 20.])

x_count = int(A / h) + 1
y_count = int((C + B) / h) + 1

eps = 0.01

def initialize_matrix(A, B, C, boundary_values, x_count, y_count
    ):
    matrix = np.zeros((y_count, x_count))

    matrix[int(C / h) + 1:, 0] = boundary_values[0]
    matrix[int(C / h) + 1, :int(A / (2 * h)) + 1] =
        boundary_values[1]
    matrix[:int(C / h) + 2, int(A / (2 * h)) + 1] =
        boundary_values[2]
    matrix[0, int(A / (2 * h)) + 2:] = boundary_values[3]
    matrix[:, int(A /  h)] = boundary_values[4]
    matrix[int((C + B) / h), :] = boundary_values[5]

    return matrix

matrix = initialize_matrix(A, B, C, boundary_values, x_count,
    y_count)

def T1(A, B, C, h, T0, y_count, x_count):
        T1 = np.copy(T0)
        for i in range(1, y_count):
```

```
            for j in range(1, x_count):
                if j > (int(A / (2 * h)) + 1) or i > (int(C / h)
                    + 1):
                    T1[i, j] = (T0[i - 1, j] + T1[i + 1, j] + T0
                        [i, j + 1] + T1[i, j - 1]) / 4

        return T1

def Norm(T_k1, T_k):
        diff = np.abs(T_k1 - T_k)
        return np.sqrt(np.sum(diff ** 2) / len(diff))

def method_Zeidel(A, B, C, h, y_count, x_count, eps, matrix):
    T_k = T_k1 = matrix
    iterations = 0
    norm = 1000
    norms = []
    while norm >= eps:
        iterations += 1
        T_k = T_k1
        T_k1 = T1(A, B, C, h, T_k, y_count, x_count)
        norm = Norm(T_k1, T_k)
        norms.append(norm)
    return T_k1, iterations, np.array(norms)

T, iterations, norms = method_Zeidel(A, B, C, h, y_count - 1,
    x_count - 1, eps, matrix)

i = np.arange(len(norms))
plt.figure(figsize=(8, 6))
plt.title("                                ")
plt.plot(i, norms[i], 'g-.')
plt.xlabel('                                          ')
plt.ylabel('                               ')
plt.yscale('log')
plt.grid()
plt.show()

X, Y = np.meshgrid(np.arange(0, x_count), np.arange(y_count, 0,
    -1))
plt.figure(figsize = (10, 8))
colorinterpolation = 50
colourMap = plt.cm.jet
```

```
plt.contourf(X, Y, T, colorinterpolation, cmap=colourMap)
plt.colorbar()
plt.show()

xgrid, ygrid = np.meshgrid(np.linspace(0., A, x_count), np.
    linspace(0., B + C , y_count))

fig = plt.figure(figsize=(15, 10))
axes = fig.add_subplot(projection='3d')
axes.set_title('3D temperature map')

axes.plot_surface(ygrid, xgrid, T, cmap = 'jet',linewidth=0.3,
    edgecolors='k')
axes.view_init(elev=45, azim=30)

plt.show()
```

Список литературы

[1] Амосов А. А., Дубинский Ю. А., Копченова Н. В. Вычислительные методы: учебное пособие //СПб.: Лань. – 2014. – Т. 672. – С. 4.

[2] Левицкий А. А. Информатика. Основы численных методов: Лабораторный практикум //Красноярск: ИПЦ КГТУ - 2005. 111. - С.