Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «МЭИ»

Институт информационных и вычислительных технологий

Кафедра математического и компьютерного моделирования

КУРСОВАЯ РАБОТА

по дисциплине «Численные методы»

ТЕМА: НАЧАЛЬНО-КРАЕВАЯ ЗАДАЧА ДЛЯ ДВУМЕРНОГО УРАВНЕНИЯ КОЛЕБАНИЙ

Студенты
гр. А-18-21 _____ Липкин Г.М.
(подпись)

_____ Перьков А.М.
(подпись)

Руководитель _____ Амосова О.А.
(подпись)

Москва, 2023

ЗАДАНИЕ

Первая краевая задача для уравнения с постоянными коэффициентами. Схема с весами. Моделирование нестационарных процессов в зависимости от правой части уравнения.

$$\delta^2 u / \delta t^2 = \Delta u + f(x, y, t),\ 0 < x < a,\ 0 < y < b,\ t > 0$$
$$u(x, y, 0) = \varphi(x, y),\ \delta u/\delta t = \psi(x, y),\ 0 < x < a,\ 0 < y < b$$
$$u|\Gamma = 0,\ t \geq 0$$

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Задача заключается в численном решении уравнения и создания 3д отображения получившейся функции в зависимости от времени. Также необходимо оценить абсолютную погрешность и погрешность по Рунге полученного решения.

Так как аналитическое решение задачи выходит за рамки курса, задача вычислялась только численно на тестовых примерах.

Для получения решения составим разностную схему первого порядка по методу разностных производных.

1. ДИСКРЕТЕЗАЦИЯ ЗАДАЧИ

1.1. Составление разностной схемы

Пусть $x_i \in [0, a]$, $y_i \in [0, b]$, $t_k \in [0, T)$, где i = 0,1,…,n, j = 0,1,…,n, k = 0,1,…,m.

$$\frac{\delta^2 u}{\delta t^2} = \Delta u + f(x,y,t), 0 < x < a, 0 < y < b, t > 0$$
$$u(x,y,0) = \varphi(x,y), \delta u/\delta t = \psi(x,y), 0 < x < a, 0 < y < b$$
$$u|\Gamma = 0, t \geq 0$$

Начальное условие:
$$u(x,y,0) = \varphi(x,y)$$
$$u_{i,j,0} = \varphi(x,y)$$

$$\frac{\big(u(x,y,h_t) - u(x,y,0)\big)}{h_t} = \psi(x,y)$$
$$u(x,y,h_t) = h_t * \psi(x,y) + u(x,y,0)$$
$$u_{i,j,1} = h_t * \psi(x,y) + \varphi(x,y)$$

Разностная схема:
$$u''_{tt} = u''_{xx} + u''_{yy} + f(x,y,t)$$

$$u''_{tt} = \frac{u_{i,j,k-1} - 2u_{i,j,k} + u_{i,j,k+1}}{{h_t}^2}$$
$$u''_{xx} = \frac{u_{i-1,j,k} - 2u_{i,j,k} + u_{i+1,j,k}}{{h_x}^2}$$
$$u''_{yy} = \frac{u_{i,j-1,k} - 2u_{i,j,k} + u_{i,j+1,k}}{{h_y}^2}$$
$$\frac{u_{i,j,k-1} - 2u_{i,j,k} + u_{i,j,k+1}}{{h_t}^2} = \frac{u_{i-1,j,k} - 2u_{i,j,k} + u_{i+1,j,k}}{{h_x}^2} + \frac{u_{i,j-1,k} - 2u_{i,j,k} + u_{i,j+1,k}}{{h_y}^2} + f_{i,j,k}$$
$${h_x}^2{h_y}^2\big(u_{i,j,k-1} - 2u_{i,j,k} + u_{i,j,k+1}\big) =$$
$$= {h_y}^2{h_t}^2\big(u_{i-1,j,k} - 2u_{i,j,k} + u_{i+1,j,k}\big) + {h_x}^2{h_t}^2\big(u_{i,j-1,k} - 2u_{i,j,k} + u_{i,j+1,k}\big)$$
$$+ {h_x}^2{h_y}^2{h_t}^2 * f_{i,j,k}$$
$$u_{i,j,k+1}\big({h_x}^2{h_y}^2\big) =$$
$$= -u_{i,j,k-1}\big({h_x}^2{h_y}^2\big) + (2u_{i,j,k}\big({h_x}^2{h_y}^2 - {h_y}^2{h_t}^2 - {h_x}^2{h_t}^2\big)$$
$$+ {h_y}^2{h_t}^2\big(u_{i-1,j,k} + u_{i+1,j,k}\big) + {h_x}^2{h_t}^2\big(u_{i,j-1,k} + u_{i,j+1,k}\big) + {h_x}^2{h_y}^2{h_t}^2 f_{i,j,k}$$
$$C_1 = {h_x}^2{h_y}^2$$
$$C_2 = -{h_y}^2{h_t}^2$$
$$C_3 = -{h_x}^2{h_t}^2$$
$$C_4 = {h_x}^2{h_y}^2{h_t}^2$$
$$C_0 = -2(C_1 + C_2 + C_3)$$
$$u_{i,j,k+1}C_1 = -u_{i,j,k-1}C_1 - \Big(u_{i,j,k}C_0 + C_2\big(u_{i-1,j,k} + u_{i+1,j,k}\big) + C_3\big(u_{i,j-1,k} + u_{i,j+1,k}\big)\Big)$$
$$+ C_4 f_{i,j,k}$$
$$u_{i,j,k+1} = -u_{i,j,k-1} - 1/C_1\Big(u_{i,j,k}C_0 + C_2\big(u_{i-1,j,k} + u_{i+1,j,k}\big) + C_3\big(u_{i,j-1,k} + u_{i,j+1,k}\big)\Big)$$
$$+ C_4 f_{i,j,k}$$

1.2. Подготовка тестовых примера

Тестовый пример 1:

$$u(x,y,t) = sin\left(\frac{x\pi}{l}\right) sin\left(\frac{y\pi}{l}\right) sin\left(\frac{t\pi}{l}\right)$$

Найдём значение правой части

$$-\frac{\pi^2}{l^2}\left(sin\left(\frac{x\pi}{l}\right) sin\left(\frac{y\pi}{l}\right) sin\left(\frac{t\pi}{l}\right)\right)$$
$$= -\frac{\pi^2}{l^2}\left(sin\left(\frac{x\pi}{l}\right) sin\left(\frac{y\pi}{l}\right) sin\left(\frac{t\pi}{l}\right)\right) - \frac{\pi^2}{l^2}\left(sin\left(\frac{x\pi}{l}\right) sin\left(\frac{y\pi}{l}\right) * sin\left(\frac{t\pi}{l}\right)\right)$$
$$+ f(x,y,t)$$
$$f(x,y,t) = -\frac{\pi^2}{l^2}\left(sin\left(\frac{x\pi}{l}\right) sin\left(\frac{y\pi}{l}\right) sin\left(\frac{t\pi}{l}\right)\right)$$

Подставим граничные условия, чтобы найти 2 первых слоя

$$\varphi = sin(x)sin(y)$$
$$u_{i,j,0} = \varphi(x,y)$$
$$u_{i,j,0} = sin(x)sin(y)$$
$$\psi(x,y) = 0$$
$$u_{i,j,1} = h_t\psi(x,y) + \varphi(x,y)$$
$$u_{i,j,1} = sin(x)sin(y)$$

Полученный тестовый пример:

$$\frac{\delta^2 u}{\delta t^2} = \Delta u + f(x,y,t), 0 < x < 2, 0 < y < 2, t > 0$$
$$f(x,y,t) = -\frac{\pi^2}{l^2}\left(sin\left(\frac{x\pi}{l}\right) sin\left(\frac{y\pi}{l}\right) sin\left(\frac{t\pi}{l}\right)\right)$$
$$u(x,y,0) = sin(x)sin(y), \frac{\delta u}{\delta t}(x,y,0) = 0, 0 < x < 2, 0 < y < 2$$

Тестовый пример 2:

$$u(x,y,t) = \left|x - \frac{a}{2}\right| \left|y - \frac{a}{2}\right| \sin(t)$$

Найдём значение правой части

$$-\left|x - \frac{a}{2}\right| \left|y - \frac{a}{2}\right| \sin(t) = 0 + 0 + f(x,y,t)$$
$$f(x,y,t) = -\left|x - \frac{a}{2}\right| \left|y - \frac{a}{2}\right| \sin(t)$$

Подставим граничные условия, чтобы найти 2 первых слоя

$$\varphi = \left|x - \frac{a}{2}\right| \left|y - \frac{a}{2}\right| - \frac{a}{2}$$
$$u_{i,j,0} = \varphi(x,y)$$

$$u_{i,j,0} = \left|x - \frac{a}{2}\right| \left|y - \frac{a}{2}\right| - \frac{a}{2}$$
$$\psi(x,y) = 0$$
$$u_{i,j,1} = h_t * \psi(x,y) + \varphi(x,y)$$
$$u_{i,j,1} = \left|x - \frac{a}{2}\right| \left|y - \frac{a}{2}\right| - \frac{a}{2}$$

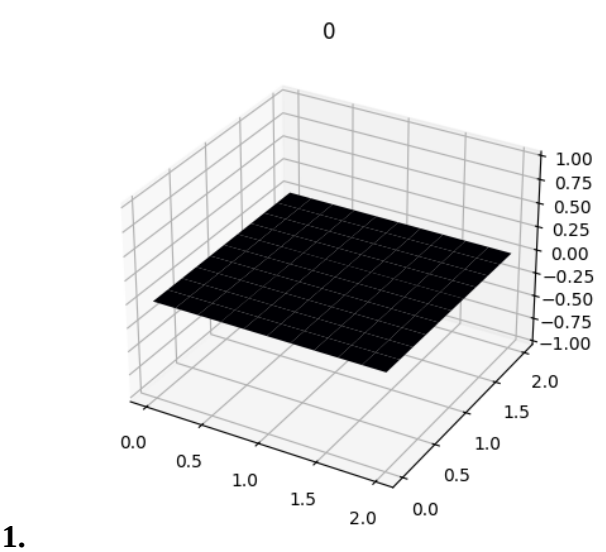
Полученный тестовый пример:

$$\frac{\delta^2 u}{\delta t^2} = \Delta u + f(x,y,t), 0 < x < 2, 0 < y < 2, t > 0$$
$$f(x,y,t) = -\left|x - \frac{a}{2}\right| \left|y - \frac{a}{2}\right| \sin(t)$$
$$u(x,y,0) = \left|x - \frac{a}{2}\right| \left|y - \frac{a}{2}\right| - \frac{a}{2}, \frac{\delta u}{\delta t}(x,y,0) = 0, 0 < x < 2, 0 < y < 2$$

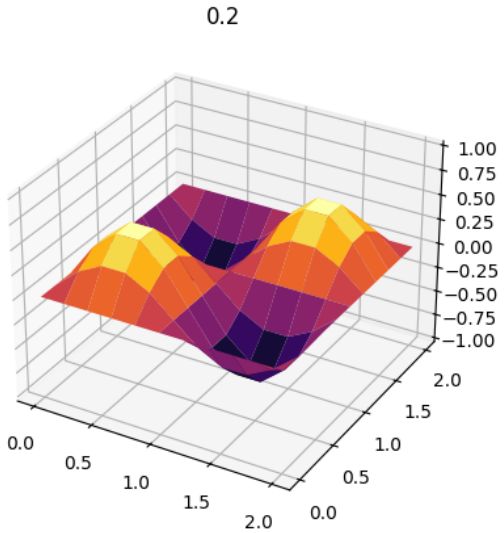
2. РЕШЕНИЕ ТЕСТОВОГО ПРИМЕРА 1

2.1. Результат решения

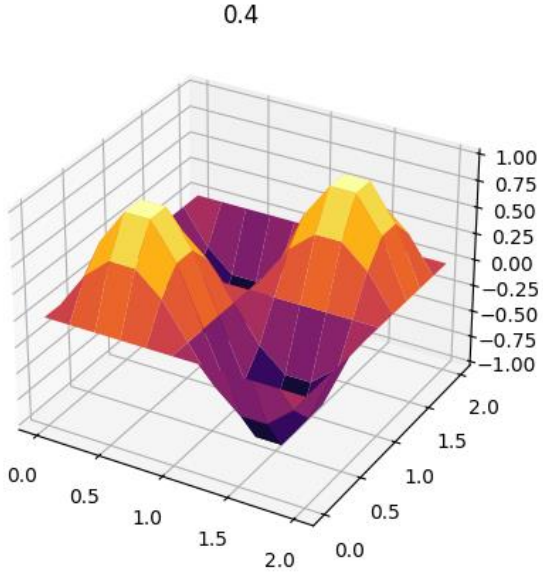
1.



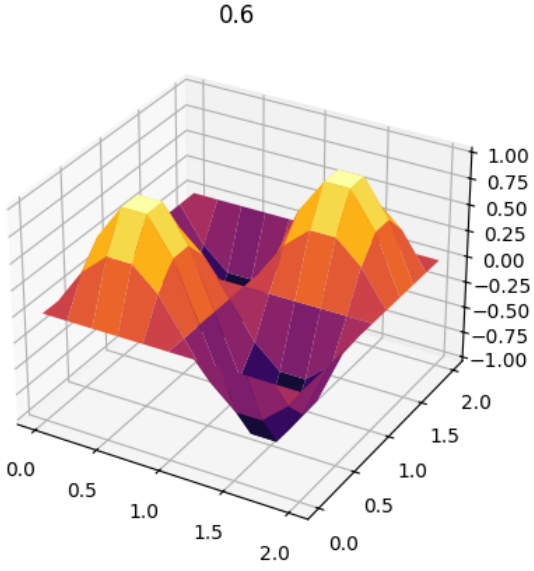
2.



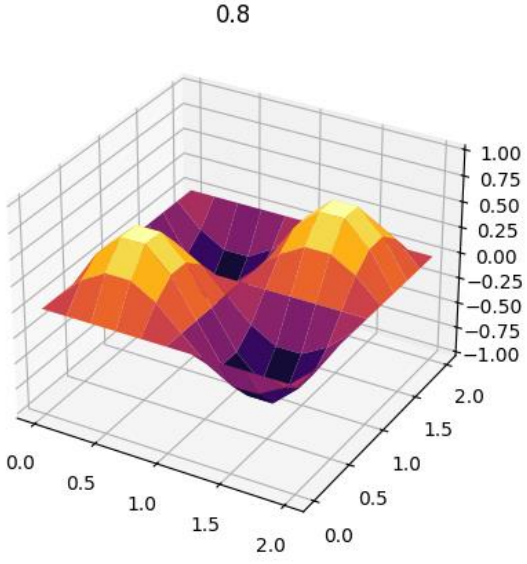
3.



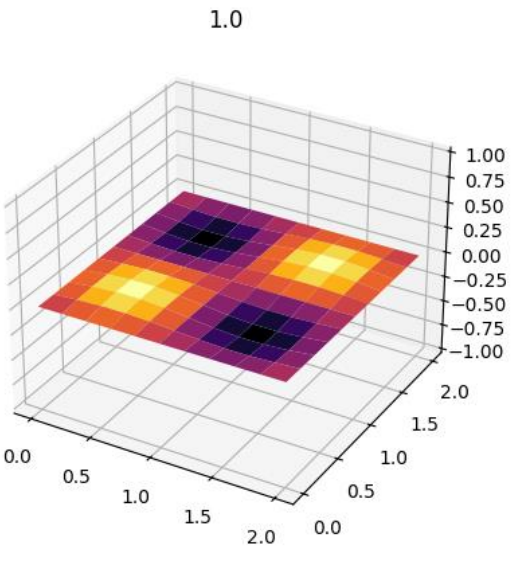
4.



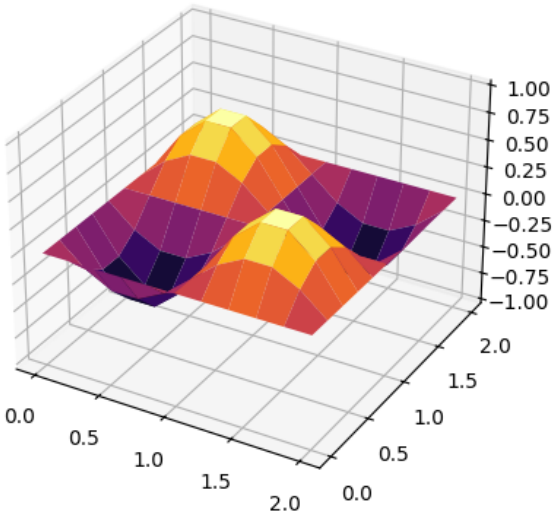
5.



6.

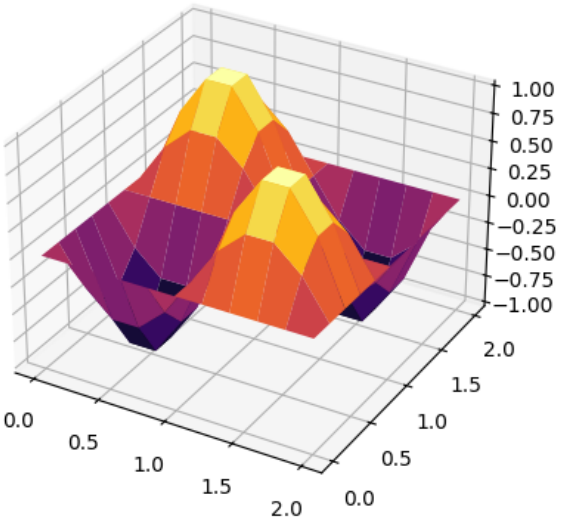


1.2



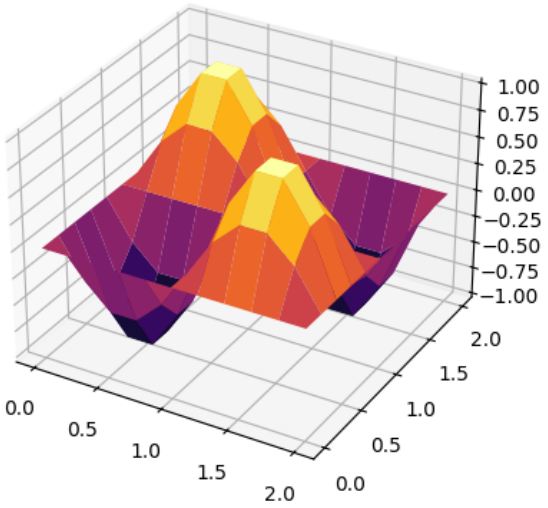
7.

1.4



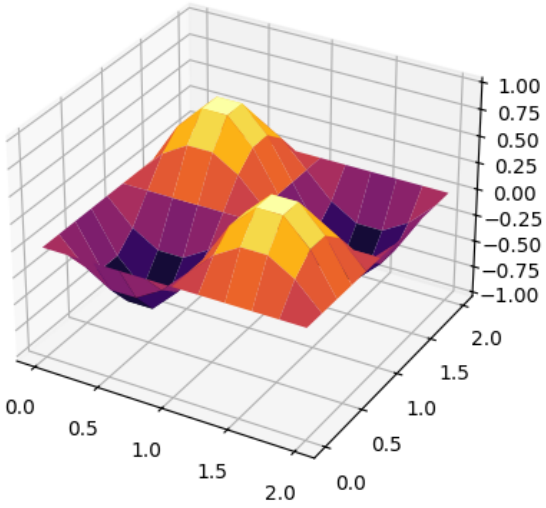
8.

1.6



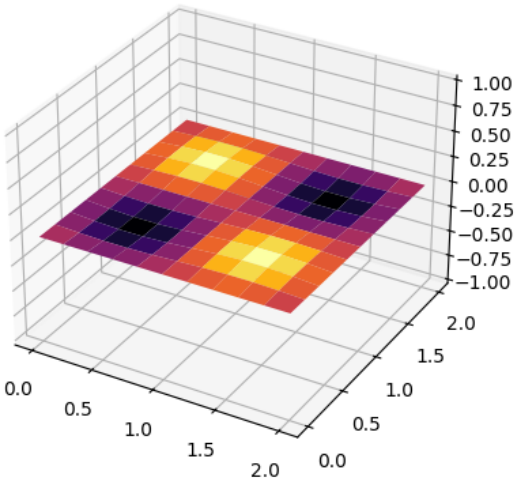
9.

1.8



10.

2.0



11.

На изображениях 1-11 показан результат одного полного колебания, полученного в результате работы функции на временном промежутке [0, 2] секунд вычисляемый с шагом 0.001 с сеткой 10 на 10.

2.2. Расчёт погрешностей

Абсолютная погрешность решения: 0.10787115412417425

Погрешность по Рунге: 0.43224571446686944

Данные результаты погрешности получены при разбиении пластины на 100 частей (10 на 10) с шагом по времени 0.001.

При увеличении числа разбиений до 400 (20 на 20) погрешность значительно уменьшается

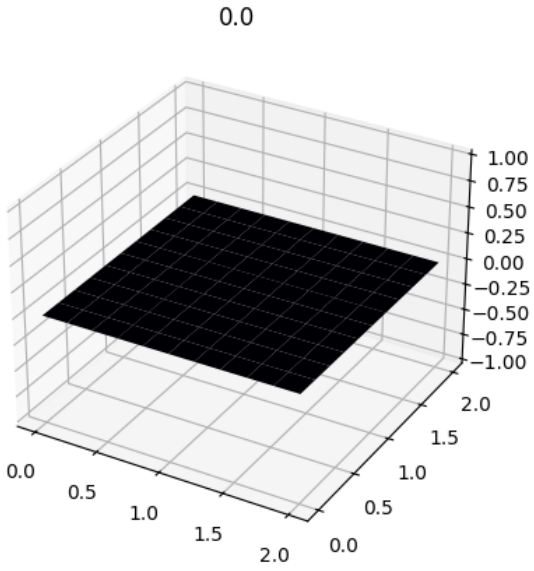
Абсолютная погрешность: 0.028703794186701437

Погрешность по Рунге: 0.08222248858192371

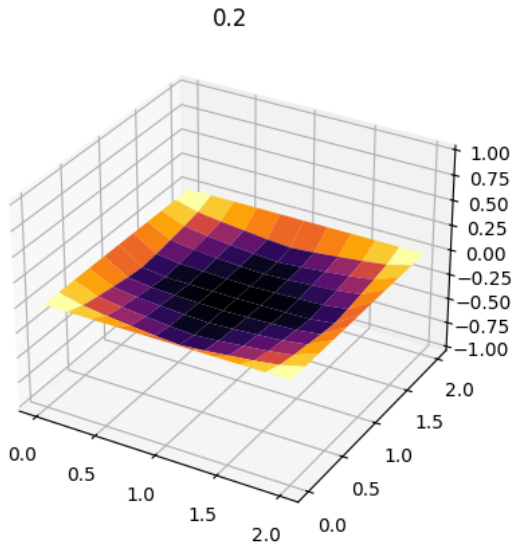
3. РЕШЕНИЕ ТЕСТОВОГО ПРИМЕРА 2

3.1. Результат решения

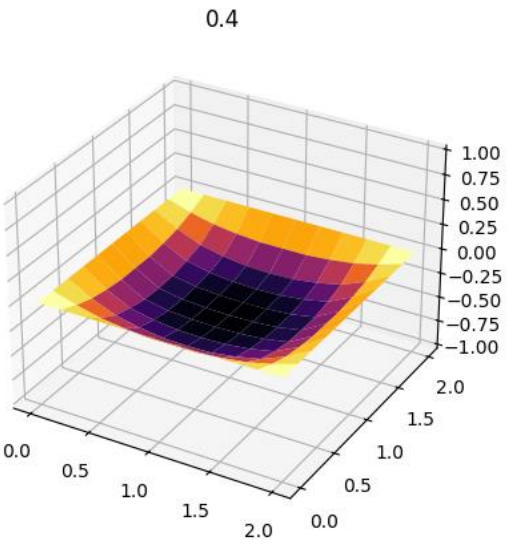
1.



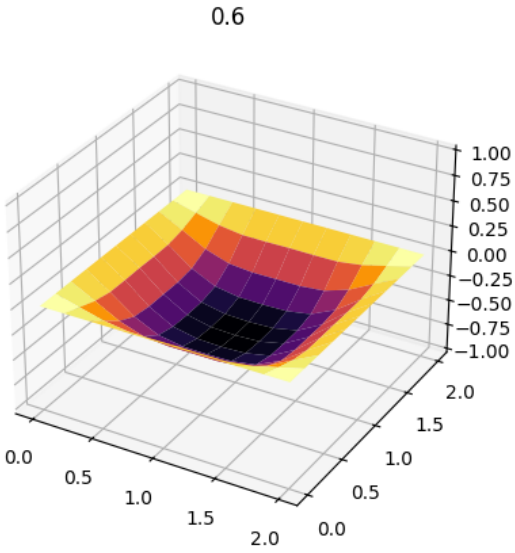
2.



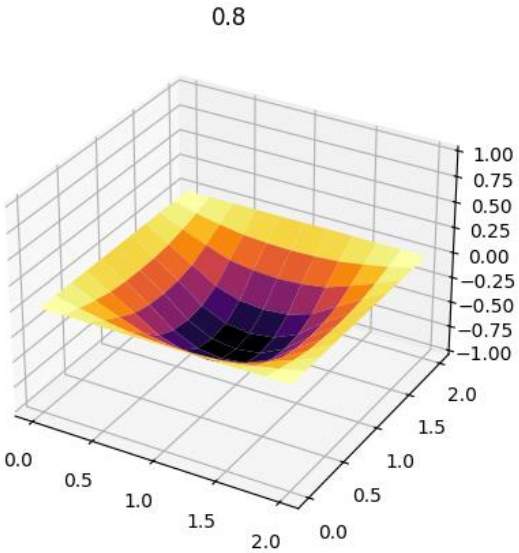
3.



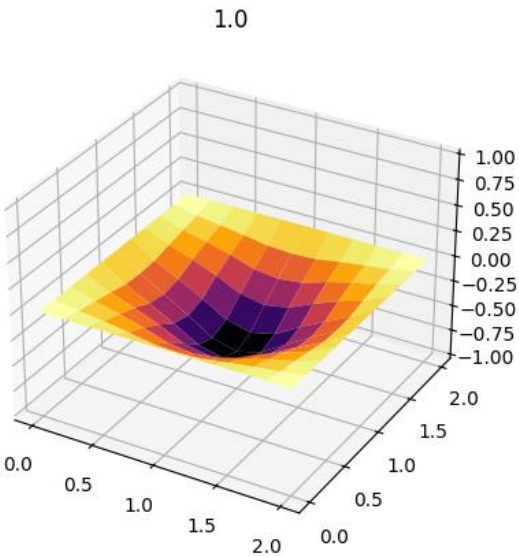
4.



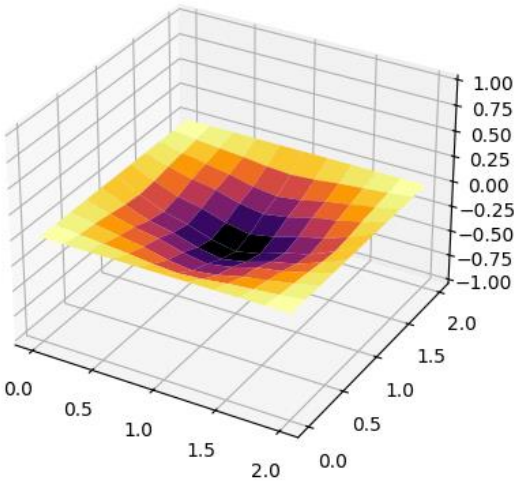
5.



6.

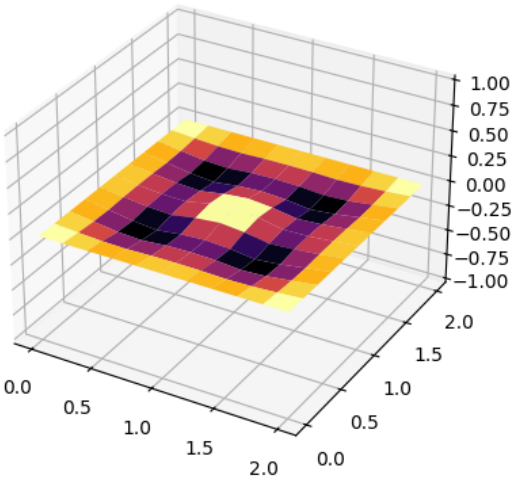


1.2



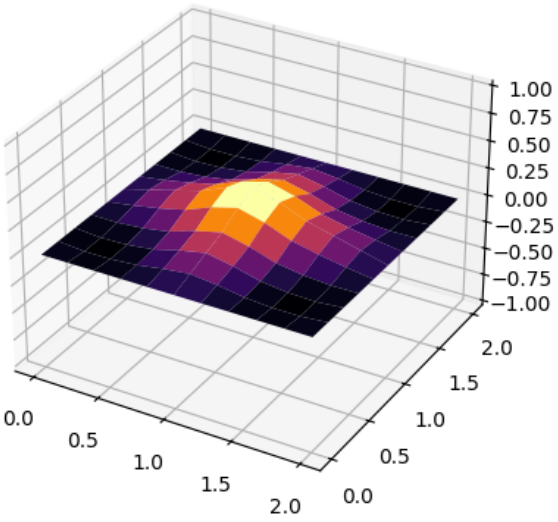
7.

1.4



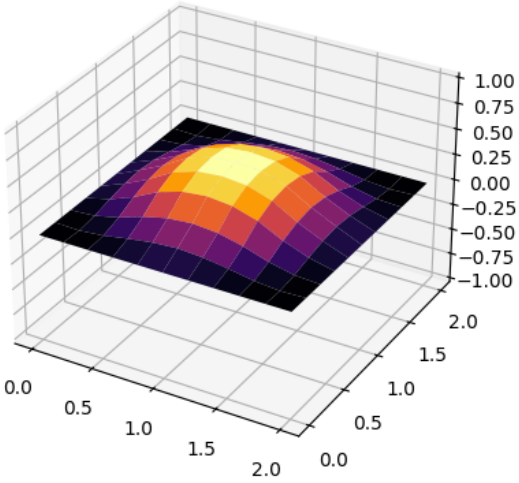
8.

1.6



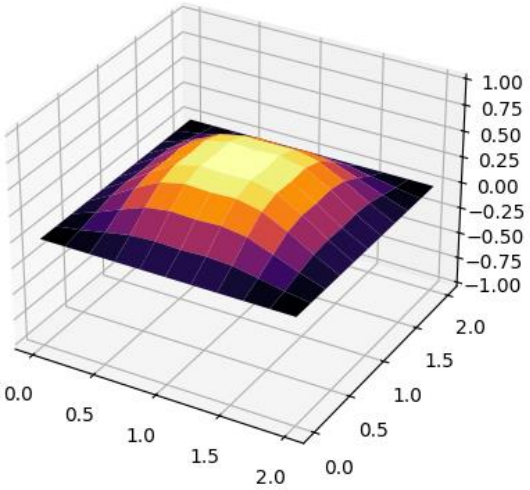
9.

1.8



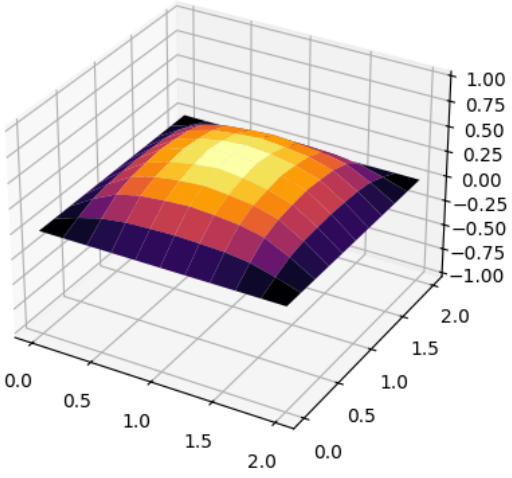
10.

2.0

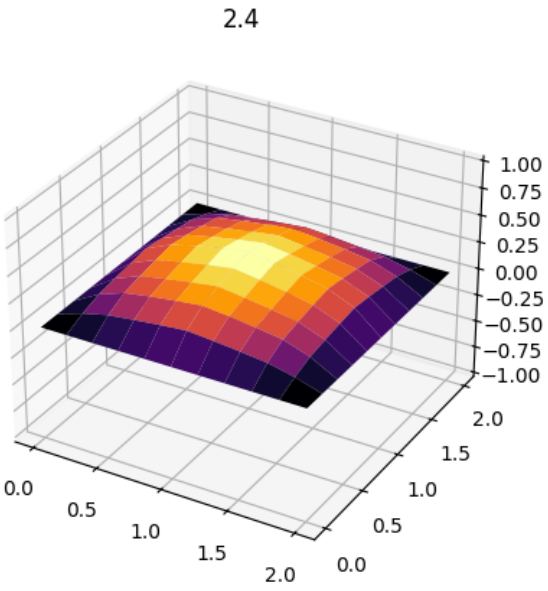


11.

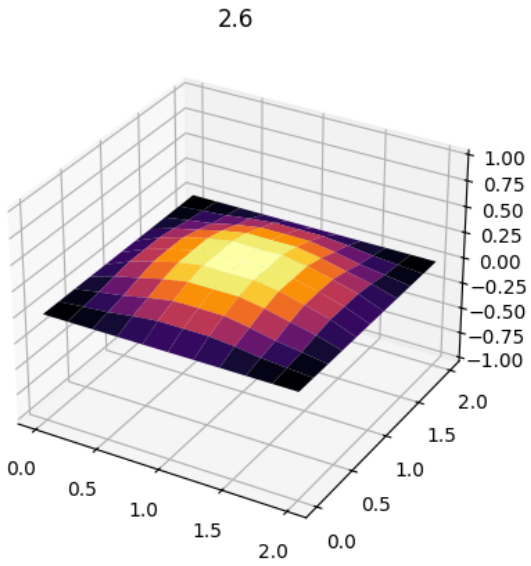
2.2



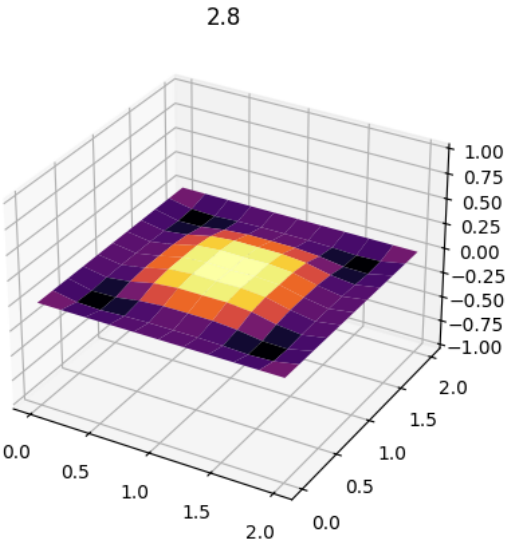
12.



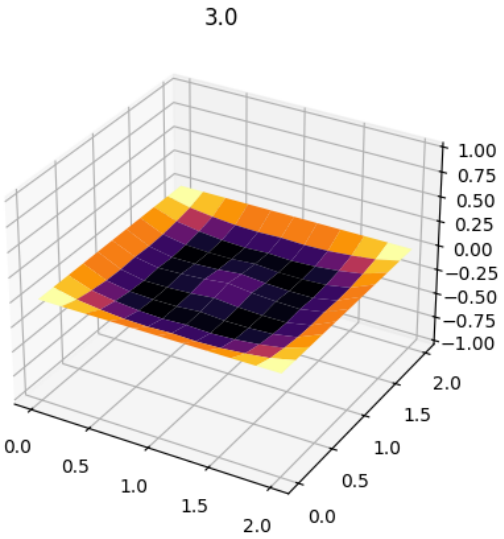
13.



14.



15.



16.

На изображениях 1-16 показан результат одного полного колебания, полученного в результате работы функции на временном промежутке [0, 3] секунд вычисляемый с шагом 0.001 с сеткой 10 на 10.

3.2. Расчёт погрешностей

Абсолютная погрешность решения: 0.7420149739404868

Погрешность по Рунге: 0.15128943133513562

Данные результаты погрешности получены при разбиении пластины на 100 частей (10 на 10) с шагом по времени 0.001.

При увеличении числа разбиений до 400 (20 на 20) погрешность увеличивается

Абсолютная погрешность: 0.8450817828317182

Погрешность по Рунге: 0.18106919441173686

4. ЗАКЛЮЧЕНИЕ

При решении задачи использовалась разностная схема, составленная с помощью разностных производных малого порядка, поэтому при численном решении наблюдается большая погрешность. Также из-за выбранного метода решения тестовый пример 2, содержащий модуль, был получен с большой погрешностью. Вычисление решения задачи трудоёмко по времени и по памяти $O(m * n^2)$, где n - количество разбиений прямоугольной области, m - количество разбиений по t.

5. ПРИЛОЖЕНИЕ

test1.py

```
from experiments import main
from numpy import pi, sin
import numpy as np
from glob import glob

l = 1.0

def real_y(x, y, t):
    return sin(x*pi/l) * sin(y*pi/l) * sin(t*pi/l)

def f(x, y, t):
    return pi*pi/l/l * sin(pi*x/l) * sin(pi*y/l) * sin(pi*t/l)

def phi(x, y):
    return 0

def psi(x, y):
    return pi/l * sin(pi*x/l) * sin(pi*y/l)

if __name__ == "__main__":
    ht = 0.001
    n = 10
    matrix_update = False
    prev_update = False
    prev_filename = f"./test1_diff_with_ht_{ht * 2}_n_{n / 2}.npy"
    prev_matrix_filename = f"./test1_diff_with_ht_{ht*2}_n_{n/2}_matrix.npy"
    filename = f"./test1_diff_with_ht_{ht}_n_{n}.npy"
    matrix_filename = f"./test1_diff_with_ht_{ht}_n_{n}_matrix.npy"
    file_present = glob(filename) and glob(matrix_filename)
    prev_present = glob(prev_filename) and glob(prev_matrix_filename)

    if matrix_update or not file_present:
        matrix, matrix1 = main(
            ht=ht, a=2, b=2, n=n, f=f, phi=phi, psi=psi, real_y=real_y
        )
        np.save(filename, matrix1)
        np.save(matrix_filename, matrix)
    else:
        matrix1 = np.load(filename)
        matrix = np.load(matrix_filename)
    if prev_update or not prev_present:
        prev_matrix, prev_matrix1 = main(
            ht=ht * 2,
            a=2,
            b=2,
            n=int(n/2),
            f=f,
            phi=phi,
            psi=psi,
            real_y=real_y,
        )
        np.save(prev_filename, prev_matrix1)
        np.save(prev_matrix_filename, prev_matrix)
    else:
        prev_matrix1 = np.load(prev_filename)
        prev_matrix =  np.load(prev_matrix_filename)
    # print(matrix1)

    print(f"Runge: {np.max(matrix[:-1:2, ::2, ::2] - prev_matrix)}")

    print(np.max(matrix1[:]))
```

test2.py

```
from experiments import main
from numpy import pi, sin
import numpy as np
from glob import glob

def phi(_x: float, _y: float):
    return 0

def psi(x: float, y: float):
    return abs(x - a / 2) * abs(y - a / 2) - a / 2

def f(x: float, y: float, t: float):
    return -abs(x - a / 2) * abs(y - a / 2) * sin(t)

def real_y(x: float, y: float, t: float):
    return abs(x - a / 2) * abs(y - a / 2) * sin(t)

if __name__ == "__main__":
    ht = 0.001

    a = 2
    b = 2
    n = 20
    matrix_update = False
    prev_update = False
    prev_filename = f"./test2_diff_with_ht_{ht * 2}_n_{n / 2}.npy"
    prev_matrix_filename = f"./test2_diff_with_ht_{ht*2}_n_{n/2}_matrix.npy"
    filename = f"./test2_diff_with_ht_{ht}_n_{n}.npy"
    matrix_filename = f"./test2_diff_with_ht_{ht}_n_{n}_matrix.npy"
    file_present = glob(filename) and glob(matrix_filename)
    prev_present = glob(prev_filename) and glob(prev_matrix_filename)

    if matrix_update or not file_present:
        matrix, matrix1 = main(
            ht=ht, a=2, b=2, n=n, f=f, phi=phi, psi=psi, real_y=real_y
        )
        np.save(filename, matrix1)
        np.save(matrix_filename, matrix)
    else:
        matrix1 = np.load(filename)
        matrix = np.load(matrix_filename)
    if prev_update or not prev_present:
        prev_matrix, prev_matrix1 = main(
            ht=ht * 2,
            a=2,
            b=2,
            n=int(n/2),
            f=f,
            phi=phi,
            psi=psi,
            real_y=real_y,
        )
        np.save(prev_filename, prev_matrix1)
        np.save(prev_matrix_filename, prev_matrix)
    else:
        prev_matrix1 = np.load(prev_filename)
        prev_matrix =  np.load(prev_matrix_filename)
    # print(matrix1)

    print(f"Runge: {np.max(matrix[:-1:2, ::2, ::2] - prev_matrix)}")

    print(np.max(matrix1[:]))
```

main.py

```
import numpy as np
from numpy import pi
import matplotlib.pyplot as plt
from typing import Callable


def method_step(*, prev_matrix, curr_matrix, f, k, deltas_matrix, hx, hy, ht,
real_y):
    C1 = hx**2 * hy**2
    C2 = -(ht**2) * hy**2
    C3 = -(ht**2) * hx**2
    C4 = ht**2 * hx**2 * hy**2
    C0 = -2 * (C1 + C2 + C3)
    next_matrix = np.zeros(prev_matrix.shape)
    for i in range(1, next_matrix.shape[0] - 1):
        for j in range(1, next_matrix.shape[1] - 1):
            next_matrix[i][j] = (
                -prev_matrix[i][j]
                - (
                    C0 * curr_matrix[i][j]
                    + C2 * (curr_matrix[i + 1][j] + curr_matrix[i - 1][j])
                    + C3 * (curr_matrix[i][j + 1] + curr_matrix[i][j - 1])
                    - C4 * f(j * hx, i * hy, (k - 1) * ht)
                )
                / C1
            )

            deltas_matrix[i][j] = abs(
                next_matrix[i][j] - real_y(j * hx, i * hy, (k - 1) * ht)
            )
    return next_matrix


def show_deltas(*, deltas_to_show, a, b):
    fig = plt.figure()
    ax = fig.add_subplot(projection="3d")
    # ax.set_zlim(-1, 1)
    x, y = np.meshgrid(
        np.linspace(0, a, deltas_to_show.shape[0], endpoint=True),
        np.linspace(0, b, deltas_to_show.shape[1], endpoint=True),
    )
    ax.plot_surface(x, y, deltas_to_show, cmap="inferno", rstride=1,
cstride=1)
    plt.show()
    return 0


def show_plot(*, matrix_to_show, i, a, b, ht):
    fig = plt.figure()
    ax = fig.add_subplot(projection="3d")
    ax.set_title(ht * i)
    ax.set_zlim(-1, 1)
    x, y = np.meshgrid(
        np.linspace(0, a, matrix_to_show.shape[0], endpoint=True),
        np.linspace(0, b, matrix_to_show.shape[1], endpoint=True),
    )
    ax.plot_surface(x, y, matrix_to_show, cmap="inferno", rstride=1,
cstride=1)
    plt.show()
    return 0
```

```
def main(
    *,
    a: float,
    b: float,
    ht: float,
    n: int,
    real_y: Callable[[float, float, float], float],
    f: Callable[[float, float, float], float],
    phi: Callable[[float, float], float],
    psi: Callable[[float, float], float],
):
    fps = int(1 / ht)
    t_steps = int(pi * 2 * fps)

    x = np.linspace(0, a, n + 1, endpoint=True)
    y = np.linspace(0, b, n + 1, endpoint=True)
    x, y = np.meshgrid(x, y)
    hx = a / n
    hy = b / n
    matrix = np.zeros((t_steps, n + 1, n + 1))
    deltas = np.zeros((t_steps, n + 1, n + 1))
    for i in range(1, n):
        for j in range(1, n):
            matrix[0][i][j] = phi(j * hx, i * hy)
            matrix[1][i][j] = ht * psi(j * hx, i * hy) + matrix[0][i][j]
    show_plot(matrix_to_show=matrix[0], ht=0, a=a, b=b, i=0)
    show_plot(matrix_to_show=matrix[1], ht=ht, a=a, b=b, i=1)

    for k in range(2, t_steps):
        matrix[k] = method_step(
            prev_matrix=matrix[k - 2],
            curr_matrix=matrix[k - 1],
            f=f,
            k=k,
            deltas_matrix=deltas[k],
            hx=hx,
            hy=hy,
            ht=ht,
            real_y=real_y,
        )
        show_plot(matrix_to_show=matrix[k], i=k, ht=ht, a=a, b=b)

    return matrix, deltas
```