Московский авиационный институт
(национальный исследовательский университет)

Институт №8 "Информационные технологии и прикладная математика"
Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа №7 по
дисциплине:

Численные методы

Вариант №3

Выполнил: студент группы М8О-409Б-20
Чибисов М.Р.
Принял: Пивоваров Е.Д.
Оценка:____

Москва, 2023 г.

1. Задание

Решить краевую задачу для дифференциального уравнения эллиптического типа. Аппроксимацию уравнения произвести с использованием центрально-разностной схемы. Для решения дискретного аналога применить следующие методы: метод простых итераций (метод Либмана), метод Зейделя, метод простых итераций с верхней релаксацией. Вычислить погрешность численного решения путем сравнения результатов с приведенным в задании аналитическим решением $U(x, y)$. Исследовать зависимость погрешности от сеточных параметров h_x,h_y.

Уравнение:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0,$$
$$u(0, y) = \cos y,$$
$$u(1, y) = e \cos y,$$
$$u_y(x, 0) = 0,$$
$$u_y(x, \frac{\pi}{2}) = -\exp(x).$$

Аналитическое решение:

$$U(x, y) = \exp(x) \cos y$$

2. Решение

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h_x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h_y^2} = -u_{i,j}$$

/Пусть $h_x = h_y$/

- Метод простых итераций:

$$u_{i,j}^{k+1} = \frac{u_{i+1,j}^k + u_{i-1,j}^k + u_{i,j+1}^k + u_{i,j-1}^k}{4 - h^2}$$

- Метод Зейделя:

$$u_{i,j}^{k+1} = \frac{u_{i+1,j}^k + u_{i-1,j}^{k+1} + u_{i,j+1}^k + u_{i,j-1}^{k+1}}{4 - h^2}$$

- Метод простых итераций с релаксацией:

$$u_{i,j}^{k+1} = \left(\frac{u_{i+1,j}^k + u_{i-1,j}^k + u_{i,j+1}^k + u_{i,j-1}^k}{4 - h^2}\right) \mathrm{C} + u_{i,j}^k (1 - \mathrm{C})$$

- Метод Зейделя с релаксацией:

$$u_{i,j}^{k+1} = \left(\frac{u_{i+1,j}^{k} + u_{i-1,j}^{k+1} + u_{i,j+1}^{k} + u_{i,j-1}^{k+1}}{4 - h^2}\right)\mathrm{C} + u_{i,j}^{k}(1 - \mathrm{C})$$

Аппроксимация граничных условий:

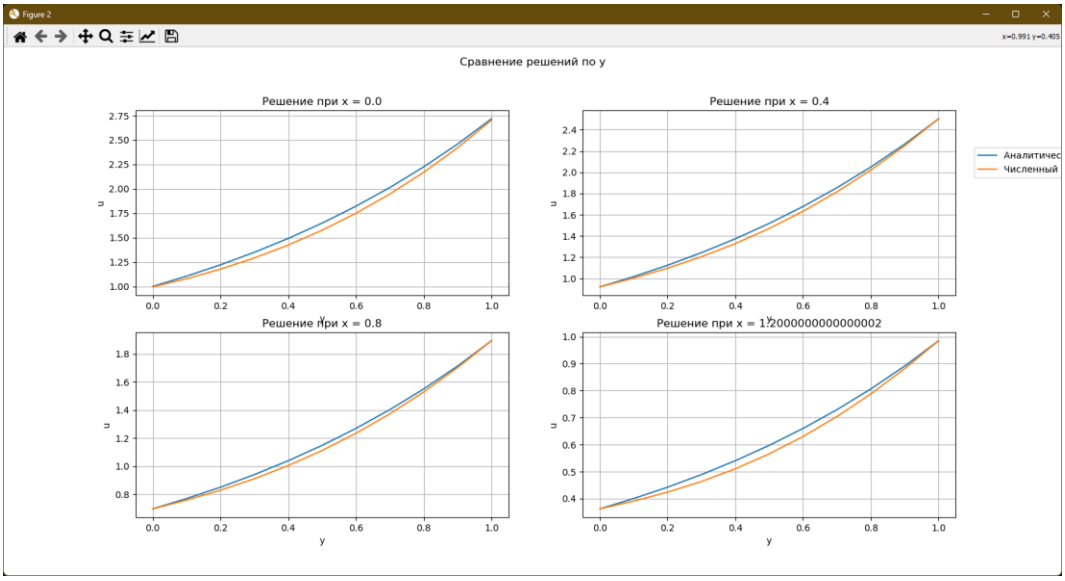
$$u_{0,j}^{k} = u_{1,j}^{k} - h_x \cos(y)$$

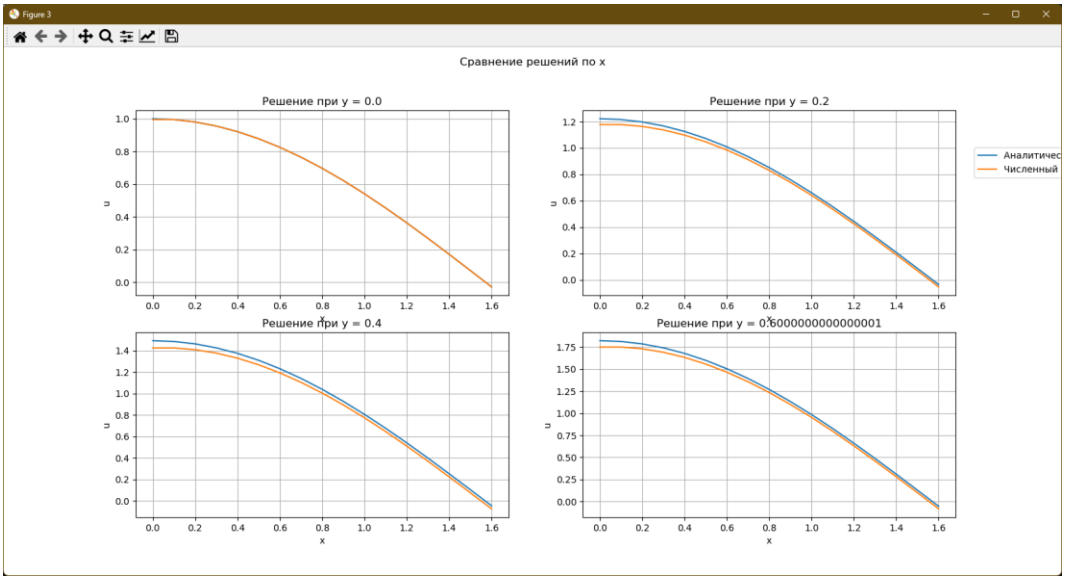
$$u_{N,j}^{k} = \frac{u_{N-1,j}^{k}}{(1 - h_x)}$$

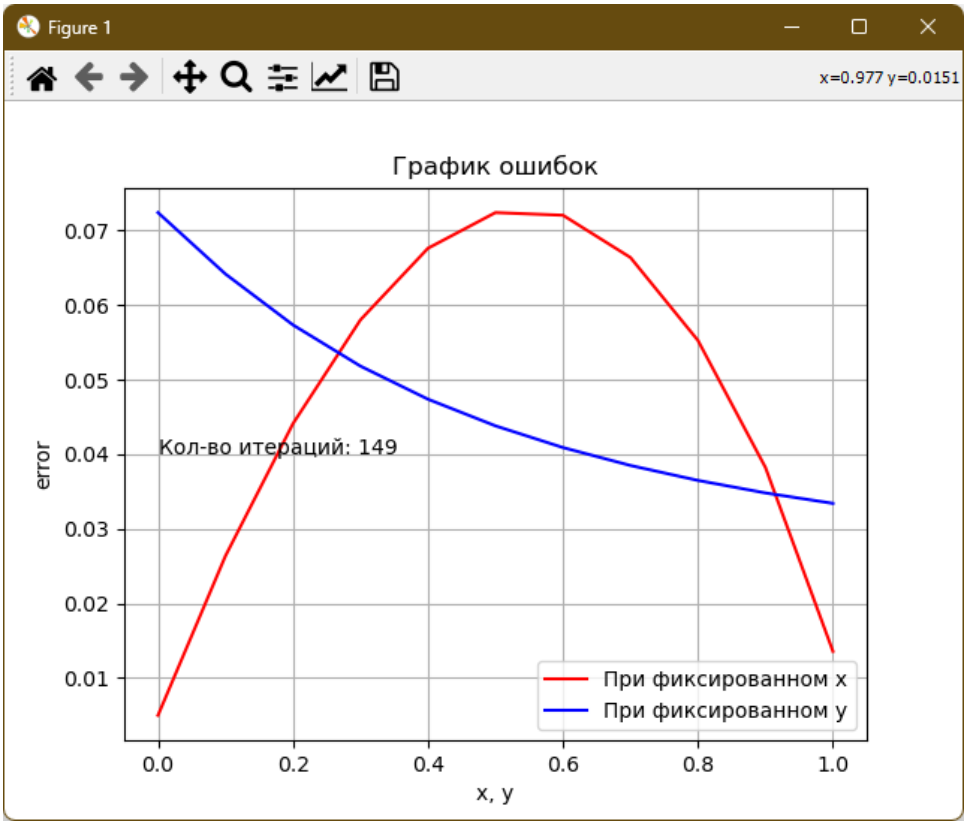
Погрешность между численным и аналитическим решением рассчитывается как абсолютная.

3. Вывод программы

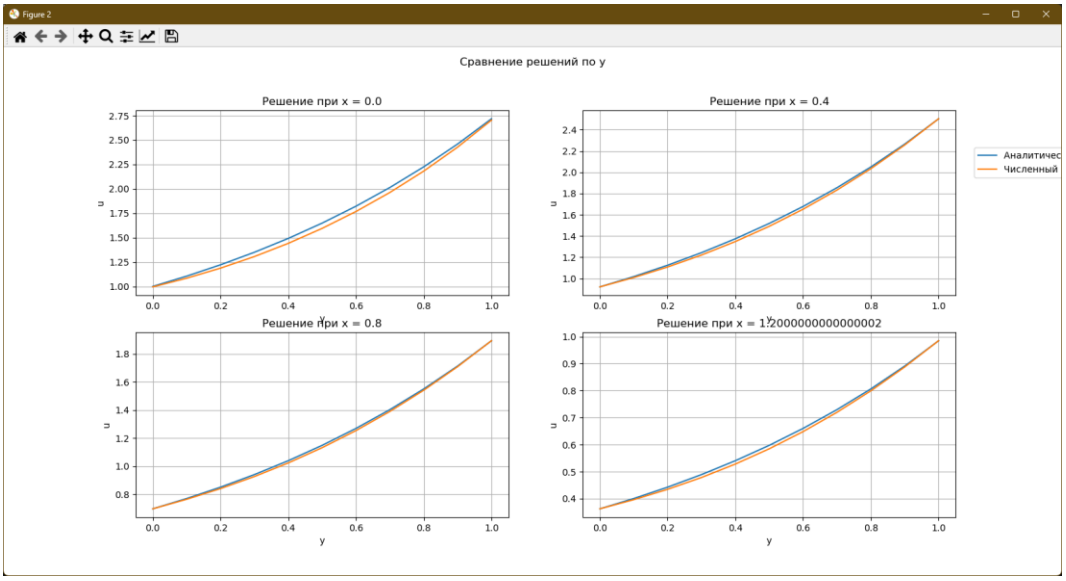
- Метод Либмана - простых итераций (k = 149)

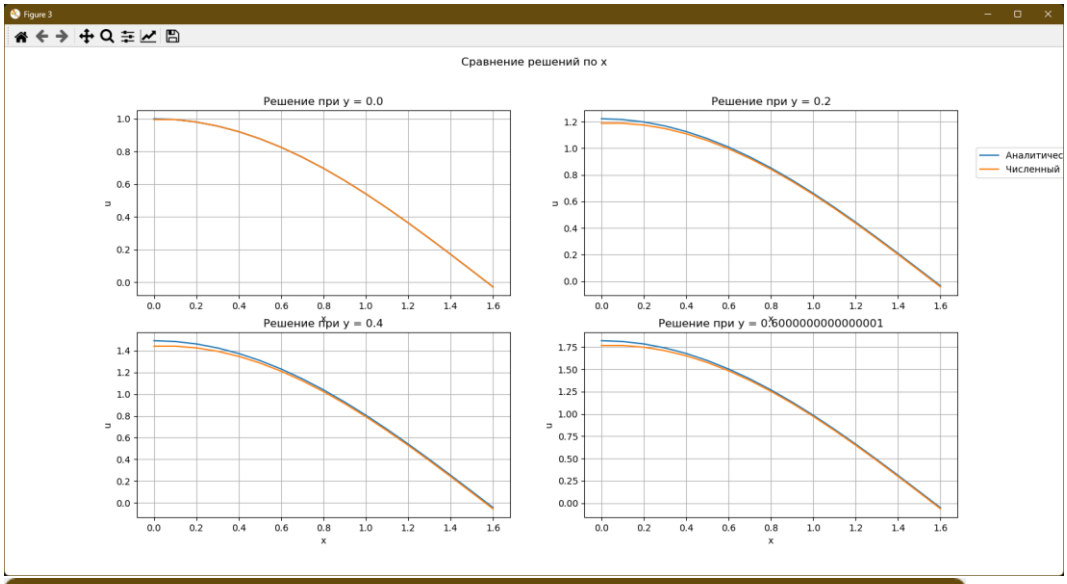


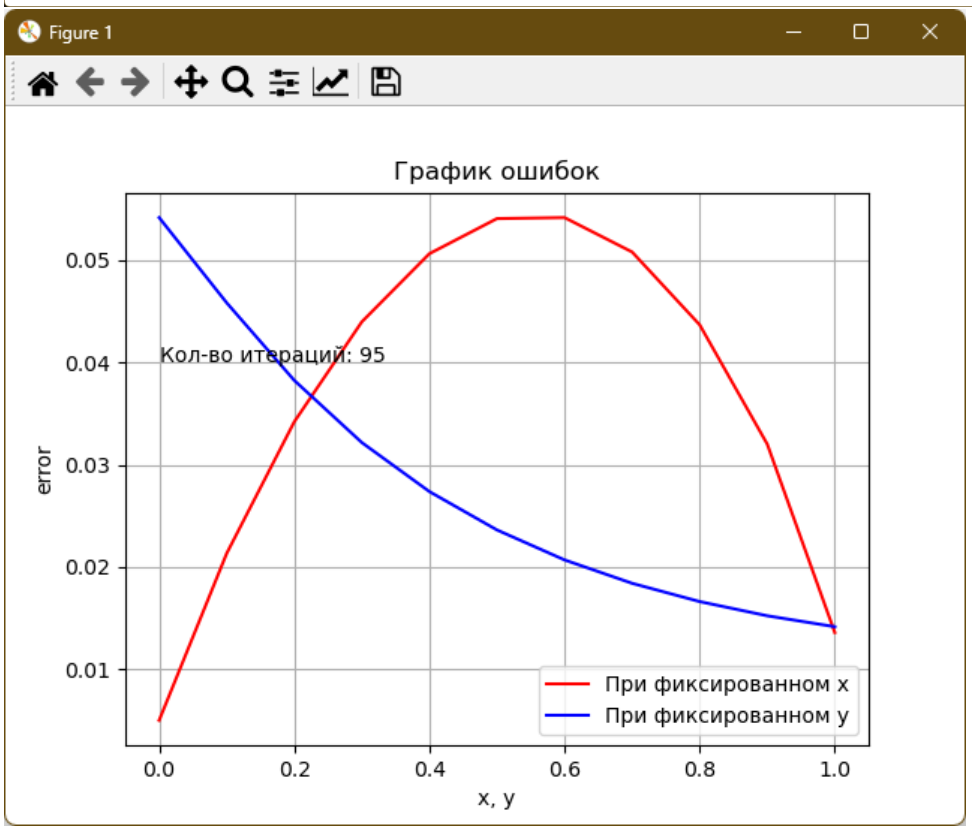




- Метод Зейделя (k = 95)







- Метод простых итераций с релаксацией (w = 1.8, k = 28)

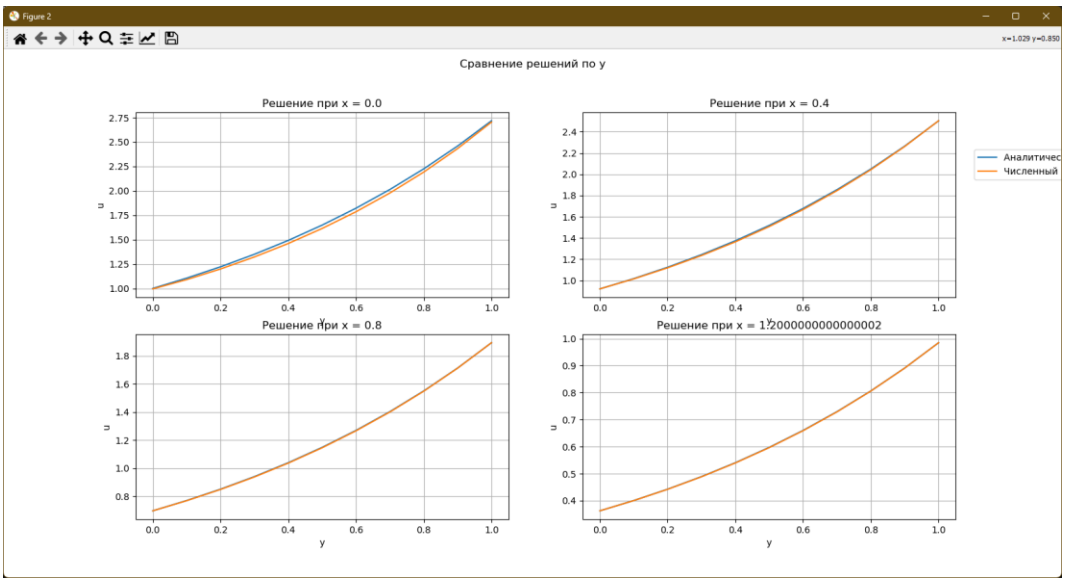


Figure 2
Сравнение решений по y
Решение при x = 0.0
Решение при x = 0.4
Решение при x = 0.8
Решение при x = 1.2000000000000002
Аналитичес
Численный
u
y

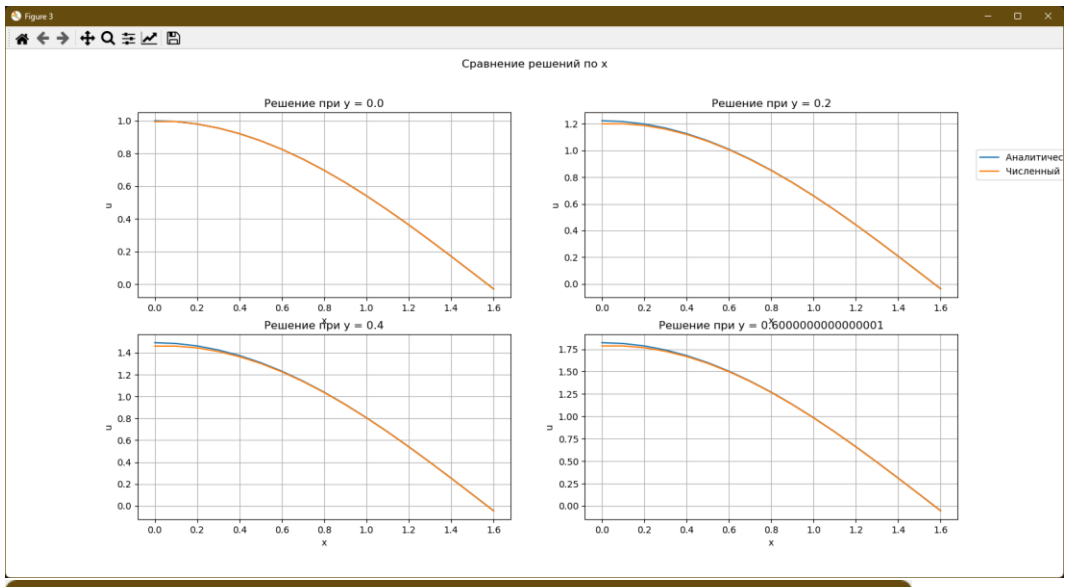


Figure 3
Сравнение решений по x
Решение при y = 0.0
Решение при y = 0.2
Решение при y = 0.4
Решение при y = 0.6000000000000001
Аналитичес
Численный
u
x

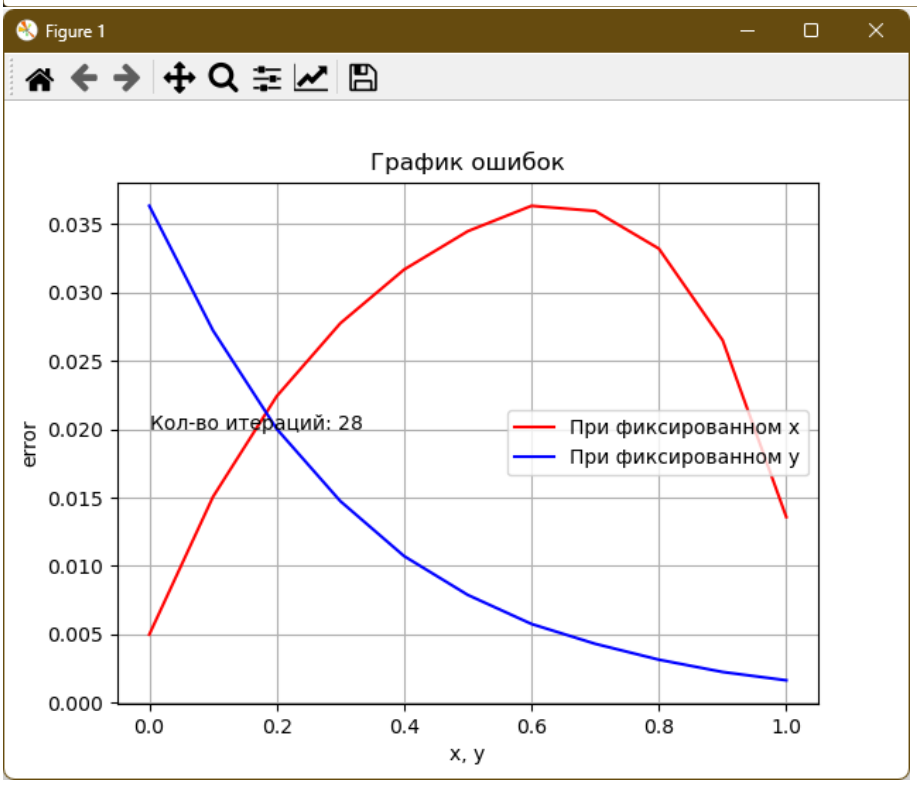


Figure 1
График ошибок
Кол-во итераций: 28
При фиксированном x
При фиксированном y
error
x, y

Листинг

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

def analyt_func(x, y):
    return np.exp(x) * np.cos(y)
def func_borderX0(y):
    return np.cos(y)
def func_borderXl(y):
    return np.e * np.cos(y)
def func_borderY0(x):  # dU/dy
    return 0
def func_borderYl(x):  # dU/dy
    return -np.exp(x)
def norm(cur_u, prev_u):
    max = 0
    for i in range(cur_u.shape[0]):
        for j in range(cur_u.shape[1]):
            if abs(cur_u[i, j] - prev_u[i, j]) > max:
                max = abs(cur_u[i, j] - prev_u[i, j])

    return max
def liebman(x, y, h, eps):
    N = len(x)
    M = len(y)
    count = 0
    prev_u = np.zeros((N, M))
    cur_u = np.zeros((N, M))
    cur_u[0] = func_borderX0(y)
    cur_u[-1] = func_borderXl(y)

    for j in range(M):
        for i in range(1, N - 1):
            cur_u[i][j] = cur_u[i][0] + (cur_u[i][-1] - cur_u[i][0]) /
(x[-1] - x[0]) * (x[i] - x[0])

    while norm(cur_u, prev_u) > eps:
        count += 1
        prev_u = np.copy(cur_u)
        for i in range(1, N - 1):
            for j in range(1, M - 1):
                cur_u[i][j] = (h[0] ** 2 * (prev_u[i - 1][j] +
prev_u[i + 1][j]) +
                               h[1] ** 2 * (prev_u[i][j - 1] +
prev_u[i][j + 1])) / (2 * (h[0] ** 2 + h[1] ** 2))
```

```
        cur_u[:, 0] = cur_u[:, 1] - h[1] * func_borderY0(x)
        cur_u[:, -1] = cur_u[:, -2] + h[1] * func_borderYl(x)

    U = np.copy(cur_u)
    return U, count

def relaxation(x, y, h, eps, w=1.8):
    N = len(x)
    M = len(y)
    count = 0
    prev_u = np.zeros((N, M))
    cur_u = np.zeros((N, M))

    cur_u[0] = func_borderX0(y)
    cur_u[-1] = func_borderXl(y)

    for j in range(M):
        for i in range(1, N - 1):
            cur_u[i][j] = cur_u[i][0] + (cur_u[i][-1] - cur_u[i][0]) /
(x[-1] - x[0]) * (x[i] - x[0])

    while norm(cur_u, prev_u) > eps:
        count += 1
        prev_u = np.copy(cur_u)
        for i in range(1, N - 1):
            for j in range(1, M - 1):
                cur_u[i][j] = (h[0]**2*(cur_u[i-1][j]+prev_u[i+1][j])
+ h[1]**2 * (cur_u[i][j-1] + prev_u[i][j+1])) / (2 * (h[0]**2 +
h[1]**2))
                cur_u[i][j] *= w
                cur_u[i][j] += (1 - w) * prev_u[i][j]
        cur_u[:, 0] = cur_u[:, 1] - h[1] * func_borderY0(x)
        cur_u[:, -1] = cur_u[:, -2] + h[1] * func_borderYl(x)

    U = np.copy(cur_u)
    return U, count


def Zeidel(x, y, h, eps):
    return relaxation(x, y, h, eps, w=1)

hx = 0.1
hy = 0.1
h = [hx, hy]
x = np.arange(0, 1 + h[0] / 2 - 1e-4, h[0])
y = np.arange(0, np.pi / 2 + h[1] / 2 - 1e-4, h[1])
```

4. Вывод

В ходе выполнения лабораторной работы были изучены метод простых итераций и метод Зейделя решений начально-краевой задачи для дифференциального уравнения эллиптического типа. Была применена двухточечная аппроксимация первого порядка граничных условий и линейная интерполяция для инициализации итерационных методов. Были получены результаты в графическом представлении и подсчитаны погрешности для каждого варианта решения.