Московский авиационный институт

(Национальный исследовательский университет)

Институт №8 «Компьютерные науки и прикладная математика»

**Отчет по лабораторной работе №7**

по курсу «Численные методы»

Вариант - 2

**Выполнила студентка**: Бондарева Е. Е.

**Группа:** М8О-405Б-21

# Преподаватель: Демидова О. Л.

Оценка: !

Дата: 7.11.2024\_\_\_\_\_\_ !

Москва, 2024

**Лабораторная работа №7**

**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 . Исследовать зависимость погрешности от сеточных параметров .

,



,

.

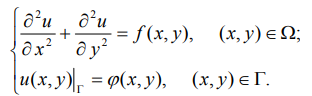
Аналитическое решение: .

**Теоретические сведения:**

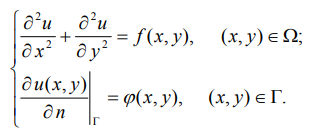
Классическим примером уравнения эллиптического типа является уравнение Пуассона : , в данной лабораторной работе то есть, уравнение имеет вид - уравнение Лапласа.

Здесь функция u(x,y) имеет различный физический смысл, а именно: стационарное, независящее от времени, распределение температуры, скорость потенциального (безвихревого) течения идеальной (без трения и теплопроводности) жидкости, распределение напряженностей электрического и магнитного полей, потенциала в силовом поле тяготения и т.п.

Если на границе Г расчетной области Ω = Ω + Γ задана искомая функция, то соответствующая первая краевая задача для уравнения Лапласа или Пуассона называется задачей Дирихле:



Если на границе Г задается нормальная производная искомой функции, то соответствующая вторая краевая задача называется задачей Неймана для уравнения Лапласа или Пуассона:



На прямоугольнике наложим сетку

.

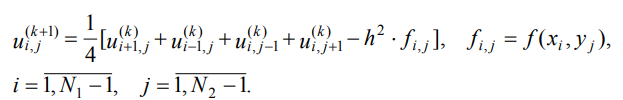
Для начальной инициализации значений в сетке можно использовать линейную интерполяцию при фиксированном для улучшения сходимости:

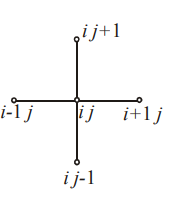
На этой сетке аппроксимируем дифференциальную задачу во внутренних узлах с помощью отношения конечных разностей по следующей схеме:

Выражаем из этого соотношения интересующий нас член, который мы можем найти одним из трех представленных ниже методов:

Метод Либмана:

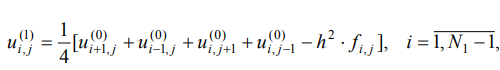
Рассмотрим разностно-итерационный метод Либмана численного решения задачи Дирихле. Для простоты изложения этого метода примем h1=h2=h, тогда из схемы получим (k-номер итерации).





Центрально-симметричный шаблон для уравнения Лапласа.

Распределение на первой итерации



Процесс Либмана прекращается, когда



**Код программы:**

import numpy as np, matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import axes3d

def analyt\_func(x, y):

    return x \*\* 2 - y \*\* 2

def func\_border1(x, y):

    return -y\*\*2

def func\_border2(x, y):

    return 1 - y \*\* 2

def func\_border3(x, y):

    return x\*\*2

def func\_border4(x, y):

    return x \*\* 2 - 1

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)

    count = 0

    prev\_u = np.zeros((N, N))

    cur\_u = np.zeros((N, N))

    for j in range(len(y)):

        coeff = (func\_border4(x[j], y[j]) - func\_border3(x[j], y[j])) / (N - 1)

        addition = func\_border3(x[j], y[j])

        for i in range(len(x)):

            cur\_u[i][j] = coeff \* i + addition

    for i in range(1, N - 1):

        cur\_u[i, 0] = func\_border1(x[i], y[i])

        cur\_u[i, -1] = func\_border2(x[i], y[i])

    for j in range(1, N - 1):

        cur\_u[0, j] = func\_border3(x[j], y[j])

        cur\_u[-1, j] = func\_border4(x[j], y[j])

    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, N - 1):

                cur\_u[i, j] = (prev\_u[i + 1, j] + prev\_u[i, j + 1] + prev\_u[i - 1, j] + prev\_u[i, j - 1] )/ 4

    U = np.copy(cur\_u)

    return U, count

def relaxation(x, y, h, eps, tau):

    N = len(x)

    count = 0

    prev\_u = np.zeros((N, N))

    cur\_u = np.zeros((N, N))

    for j in range(len(y)):

        coeff = (func\_border4(x[j], y[j]) - func\_border3(x[j], y[j])) / (N - 1)

        addition = func\_border3(x[j], y[j])

        for i in range(len(x)):

            cur\_u[i][j] = coeff \* i + addition

    for i in range(1, N - 1):

        cur\_u[i, 0] = func\_border1(x[i], y[i])

        cur\_u[i, -1] = func\_border2(x[i], y[i])

    for j in range(1, N - 1):

        cur\_u[0, j] = func\_border3(x[j], y[j])

        cur\_u[-1, j] = func\_border4(x[j], y[j])

    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, N - 1):

                cur\_u[i, j] = (1 - tau) \* prev\_u[i, j] + tau \*((prev\_u[i + 1, j] + prev\_u[i, j + 1] + prev\_u[i - 1, j] + prev\_u[i, j - 1]) / 4)

    U = np.copy(cur\_u)

    return U, count

def Zeidel(x, y, h, eps, tau):

    N = len(x)

    count = 0

    prev\_u = np.zeros((N, N))

    cur\_u = np.zeros((N, N))

    for j in range(len(y)):

        coeff = (func\_border4(x[j], y[j]) - func\_border3(x[j], y[j])) / (N - 1)

        addition = func\_border3(x[j], y[j])

        for i in range(len(x)):

            cur\_u[i][j] = coeff \* i + addition

    for i in range(1, N - 1):

        cur\_u[i, 0] = func\_border1(x[i], y[i])

        cur\_u[i, -1] = func\_border2(x[i], y[i])

    for j in range(1, N - 1):

        cur\_u[0, j] = func\_border3(x[j], y[j])

        cur\_u[-1, j] = func\_border4(x[j], y[j])

    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, N - 1):

                cur\_u[i, j] = (1 - tau) \* prev\_u[i, j] + tau \* (

                        (prev\_u[i + 1, j] + prev\_u[i, j + 1] + cur\_u[i - 1, j] + cur\_u[i, j - 1]) / 4)

    U = np.copy(cur\_u)

    return U, count

eps = 0.0001

N = 50

tau = 0.5

h = 1 / N

h2 = h / 2

h3 = h / 4

x = np.arange(0, 1 + h / 2 - 1e-4, h)

y = np.arange(0, 1 + h / 2 - 1e-4, h)

x2 = np.arange(0, 1 + h2 / 2 - 1e-4, h)

y2 = np.arange(0, 1 + h2 / 2 - 1e-4, h)

x3 = np.arange(0, 1 + h3 / 2 - 1e-4, h)

y3 = np.arange(0, 1 + h3 / 2 - 1e-4, h)

X, Y = np.meshgrid(x, y)

X2, Y2 = np.meshgrid(x2, y2)

X3, Y3 = np.meshgrid(x3, y3)

U\_analytic = analyt\_func(X, Y)

print("Лабораторная работа №7")

print("Выберите метод:\n"

      "1) Метод простых итераций (метод Либмана)\n"

      "2) Метод Зейделя\n"

      "3) Метод простых итераций с верхней релаксацией")

method = int(input())

if method == 1:

    U1, count1 = liebman(x, y, h, eps)

    U2 = liebman(x, y, h2, eps)

    U3 = liebman(x, y, h3, eps)

    fig = plt.figure()

    ax = fig.add\_subplot(projection='3d')

    ax.set\_title("Метод Либмана")

    ax.plot\_wireframe(X, Y, U\_analytic, color="red", label="Точное решение")

    ax.plot\_wireframe(X, Y, U1, color="blue", label="Численное решение")

    ax.set\_xlabel("x")

    ax.set\_ylabel("y")

    ax.set\_zlabel("U")

    ax.legend()

    plt.show()

if method == 2:

    U1, count2 = Zeidel(x, y, h, eps, tau)

    U2 = Zeidel(x, y, h2, eps, tau)

    U3 = Zeidel(x, y, h3, eps, tau)

    fig = plt.figure()

    ax = fig.add\_subplot(projection='3d')

    ax.set\_title("Метод Зейделя")

    ax.plot\_wireframe(X, Y, U\_analytic, color="red", label="Точное решение")

    ax.plot\_wireframe(X, Y, U1, color="blue", label="Численное решение")

    ax.set\_xlabel("x")

    ax.set\_ylabel("y")

    ax.set\_zlabel("U")

    ax.legend()

    plt.show()

if method == 3:

    U1, count3 = relaxation(x, y, h, eps, tau)

    U2 = relaxation(x, y, h2, eps, tau)

    U3 = relaxation(x, y, h3, eps, tau)

    fig = plt.figure()

    ax = fig.add\_subplot(projection='3d')

    ax.set\_title("Метод простых итераций с верхней релаксацией")

    ax.plot\_wireframe(X, Y, U\_analytic, color="red", label="Точное решение")

    ax.plot\_wireframe(X, Y, U1, color="blue", label="Численное решение")

    ax.set\_xlabel("x")

    ax.set\_ylabel("y")

    ax.set\_zlabel("U")

    ax.legend()

    plt.show()

H = np.zeros(3)

E1 = np.zeros(3)

E2 = np.zeros(3)

E3 = np.zeros(3)

n = 1

for i in range(3):

    n = int(n \* 2)

    h = 1 / n

    print("h = " , h)

    x = np.arange(0, 1 + h / 2 - 1e-4, h)

    y = np.arange(0, 1 + h / 2 - 1e-4, h)

    X, Y = np.meshgrid(x, y)

    U\_analytic = analyt\_func(X, Y)

    U1, count1 = liebman(x, y, h, eps)

    U2, count2 = Zeidel(x, y, h, eps, tau)

    U3, count3 = relaxation(x, y, h, eps, tau)

    H[i] = h

    error\_x1 = []

    error\_x2 = []

    error\_x3 = []

    error\_x4 = []

    for j in range(len(x)):

        error\_x1.append(max(abs(U\_analytic[:, j] - U1[:, j])))

        error\_x2.append(max(abs(U\_analytic[:, j] - U2[:, j])))

        error\_x3.append(max(abs(U\_analytic[:, j] - U3[:, j])))

    E1[i] = max(error\_x1)

    E2[i] = max(error\_x2)

    E3[i] = max(error\_x3)

from scipy.interpolate import PchipInterpolator

X\_reverse = H[::-1]

pchip\_reverse1 = PchipInterpolator(X\_reverse, E1)

pchip\_reverse2 = PchipInterpolator(X\_reverse, E2)

pchip\_reverse3 = PchipInterpolator(X\_reverse, E3)

xnew\_reverse = np.linspace(min(X\_reverse), max(X\_reverse), 1000)

ynew\_reverse1 = pchip\_reverse1(xnew\_reverse)

ynew\_reverse2 = pchip\_reverse2(xnew\_reverse)

ynew\_reverse3 = pchip\_reverse3(xnew\_reverse)

if method == 1:

    plt.figure(figsize=(10, 6))

    plt.scatter(X\_reverse, E1, label='', zorder=6,c = 'r')

    plt.plot(xnew\_reverse, ynew\_reverse1, label='', linewidth=2,c = 'b')

    plt.title('График погрешности метода Либмана')

    plt.xlabel('h')

    plt.ylabel('error')

    plt.grid(True)

    print("Количество итераций методом Либмана:", count1)

if method == 2:

    plt.figure(figsize=(10, 6))

    plt.scatter(X\_reverse, E2, label='', zorder=6,c = 'r')

    plt.plot(xnew\_reverse, ynew\_reverse2, label='', linewidth=2,c = 'b')

    plt.title('График погрешности метода Зейделя')

    plt.xlabel('h')

    plt.ylabel('error')

    plt.grid(True)

    print("Количество итераций методом Зейделя:", count2)

if method == 3:

    plt.figure(figsize=(10, 6))

    plt.scatter(X\_reverse, E3, label='', zorder=6,c = 'r')

    plt.plot(xnew\_reverse, ynew\_reverse3, label='', linewidth=2,c = 'b')

    plt.title('График погрешности метода простых итераций с верхней релаксацией')

    plt.xlabel('h')

    plt.ylabel('error')

    plt.grid(True)

    print("Количество итераций методом простых итераций с верхней релаксацией:", count3)

plt.show()

**Результат работы программы:**

Лабораторная работа №7

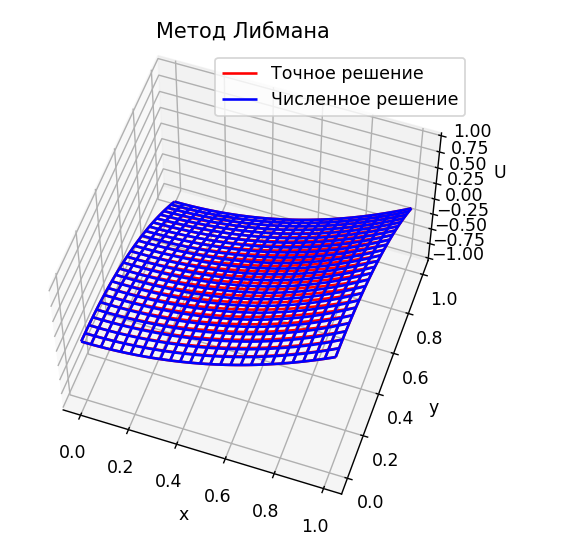
Выберите метод:

1) Метод простых итераций (метод Либмана)

2) Метод Зейделя

3) Метод простых итераций с верхней релаксацией

1

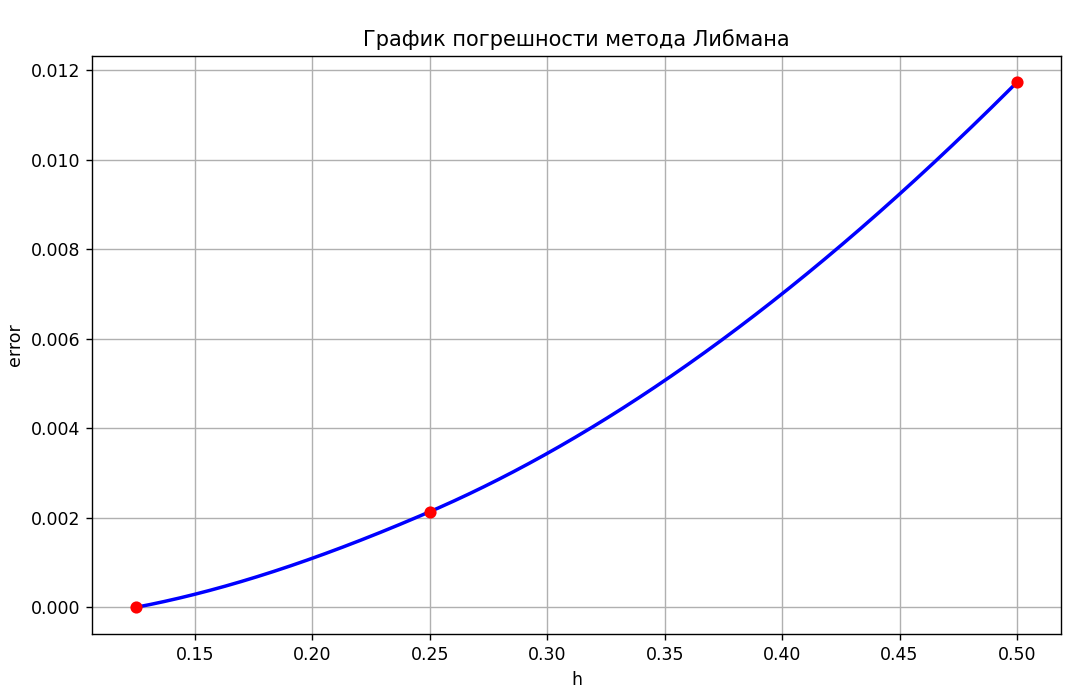


h = 0.5

h = 0.25

h = 0.125

Количество итераций методом Либмана: 947



Лабораторная работа №7

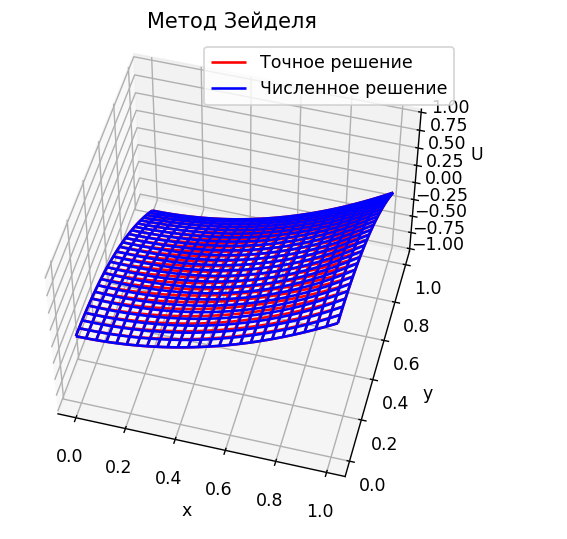
Выберите метод:

1) Метод простых итераций (метод Либмана)

2) Метод Зейделя

3) Метод простых итераций с верхней релаксацией

2

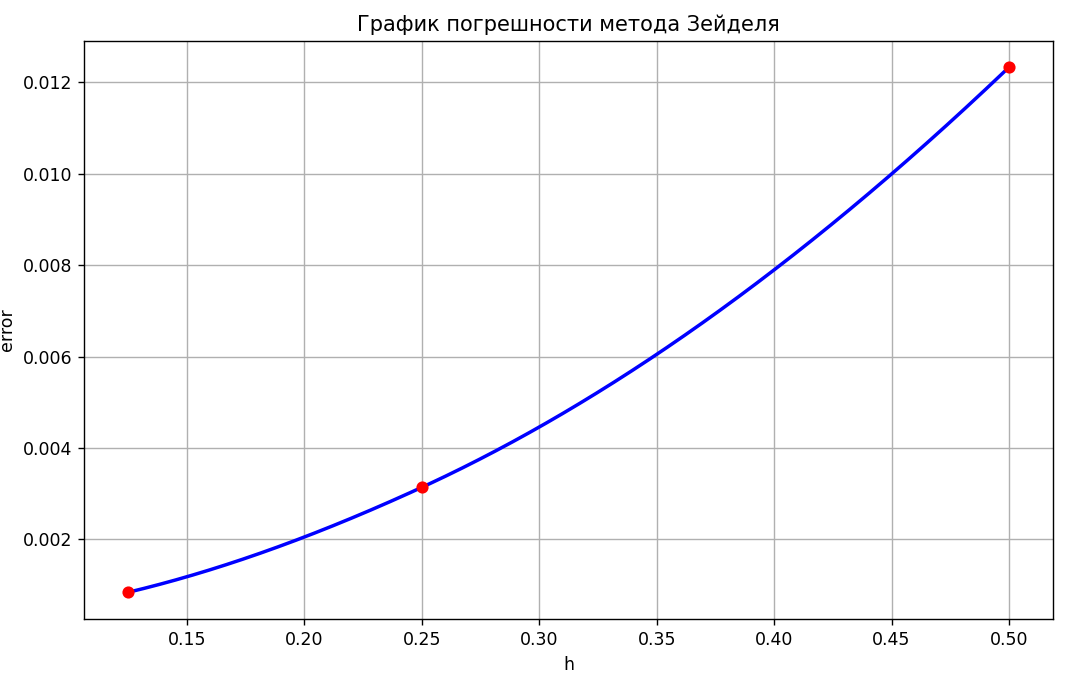


h = 0.5

h = 0.25

h = 0.125

Количество итераций методом Либмана: 670



Лабораторная работа №7

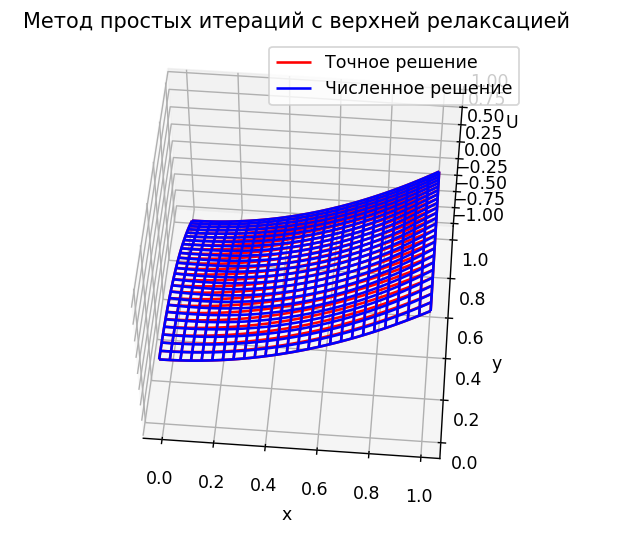
Выберите метод:

1) Метод простых итераций (метод Либмана)

2) Метод Зейделя

3) Метод простых итераций с верхней релаксацией

3

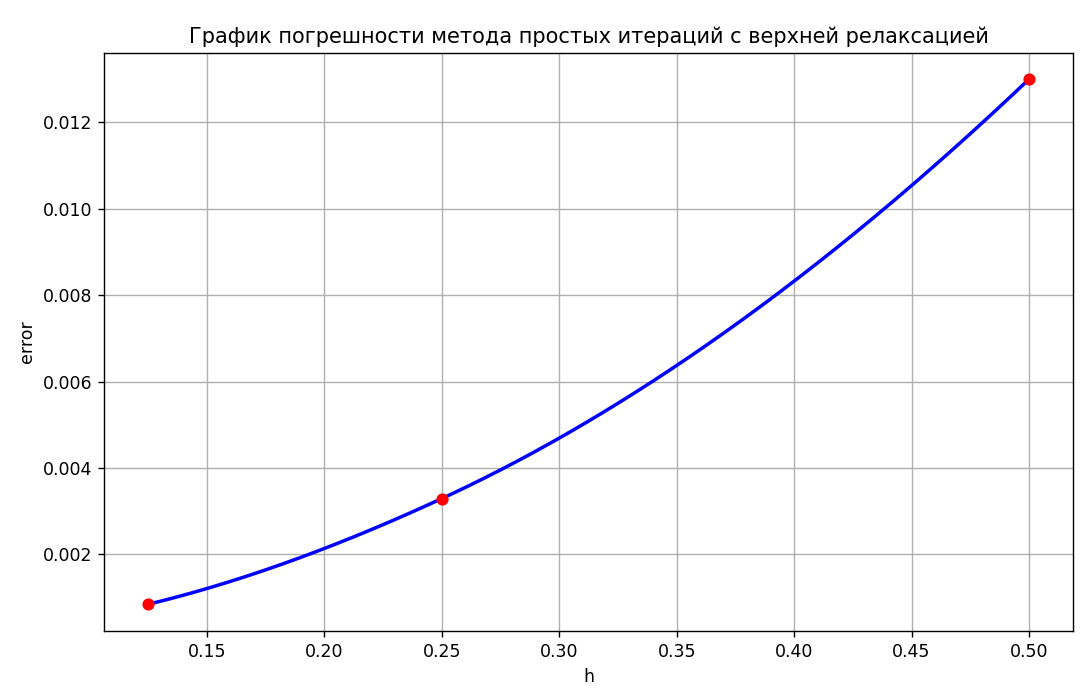


h = 0.5

h = 0.25

h = 0.125

Количество итераций методом Либмана: 664



**Вывод:**

В результате выполнения лабораторной работы была решена краевая задача для дифференциального уравнения эллиптического типа методами Либмана, Зейделя и простых итераций с верхней релаксацией. Было проведено исследование зависимости погрешности от h.

Из результатов выполненной работы можно сделать вывод, что все методы Зейделя и простых итераций с верхней релаксациией примерно в равной степени точны, и количество итераций, за которое эта точность достигается, у каждого из них примерно одинаковы. А вот метод Либмана имеет большую точность, но большое количество итераций. Самым быстрым оказался метод простых итераций с верхней релаксацией, за ним следует метод Зейделя, а самым медленным оказался метод простых итераций.

Погрешность наименьшая в методе Либмана, а наибольшая в методе простых итераций с верхней релаксацией. Также из графика погрешностей можно заметить четкую тенденцию – при небольшой длине шага зависимость погрешности от длины шага наименьшая, а наибольшая при увеличении длины шага.