Московский авиационный институт

(Национальный исследовательский университет)

Факультет прикладной математики и физики

Кафедра вычислительной математики и программирования

Отчет по лабораторным работам

по курсу «Численные методы»

Вариант 2

Выполнил: Баранов А.Д.

Группа: М8О-408Б-20

Проверил: проф. Пивоваров Д.Е.

Дата:

Оценка:

Москва, 2023

ЛАБОРАТОРНАЯ РАБОТА №3. РЕШЕНИЕ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ ЭЛЛИЧЕСКОГО ТИПА

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.

Вариант 2

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

$$u_x(0, y) = 0,$$

$$u(1, y) = 1 - y^2,$$

$$u_y(x, 0) = 0,$$

$$u(x, 1) = x^2 - 1.$$

Аналитическое решение: $U(x, y) = x^2 - y^2$.

Ход решения

На прямоугольнике $x \in [0, l_x], y \in [0, l_y]$ наложим сетку $\omega_{h_1,h_2} = \{x_i = ih_x, i = \overline{0, N_x}, y_j = jh_y, j = \overline{0, N_y}\}$. Согласно граничным условиям мы можем сразу явно заполнить верхнюю строку и последний столбец сетки, так они задаются граничными условиями первого рода. На этой сетке аппроксимируем дифференциальную задачу во внутренних узлах с помощью отношения конечных разностей:

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h_x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h_y^2} + O\left(h_x^2 + h_y^2\right) = 0$$

Выражаем из этого соотношения интересующий нас член $u_{i,j}$, который мы можем найти одним из трех представленных ниже методов:

Метод Либмана

$$u_{ij}^{(k)} = \frac{1}{\frac{2}{h_x^2} + \frac{2}{h_y^2}} \left(\frac{u_{i-1j}^{(k-1)} + u_{i+1j}^{(k-1)}}{h_x^2} + \frac{u_{ij-1}^{(k-1)} + u_{ij+1}^{(k-1)}}{h_y^2} \right)$$

Метод Зейделя

$$u_{ij}^{(k)} = \frac{1}{\frac{2}{h_x^2} + \frac{2}{h_y^2}} \left(\frac{u_{i-1j}^{(k)} + u_{i+1j}^{(k-1)}}{h_x^2} + \frac{u_{ij-1}^{(k-1)} + u_{ij+1}^{(k)}}{h_y^2} \right)$$

Метод простых итераций с верхней релаксацией

$$u_{ij}^{(k)} = \theta \frac{1}{\frac{2}{h_x^2} + \frac{2}{h_y^2}} \left(\frac{u_{i-1j}^{(k)} + u_{i+1j}^{(k-1)}}{h_x^2} + \frac{u_{ij-1}^{(k-1)} + u_{ij+1}^{(k)}}{h_y^2} \right) + (1 - \theta) *$$

$$* \frac{1}{\frac{2}{h_x^2} + \frac{2}{h_y^2}} \left(\frac{u_{i-1j}^{(k-1)} + u_{i+1j}^{(k-1)}}{h_x^2} + \frac{u_{ij-1}^{(k-1)} + u_{ij+1}^{(k-1)}}{h_y^2} \right)$$

Точки в первом столбце и нижней строке сетки получаются из аппроксимации граничных условий:

$$u_{i0} = u_{i1} - h_y * 0$$

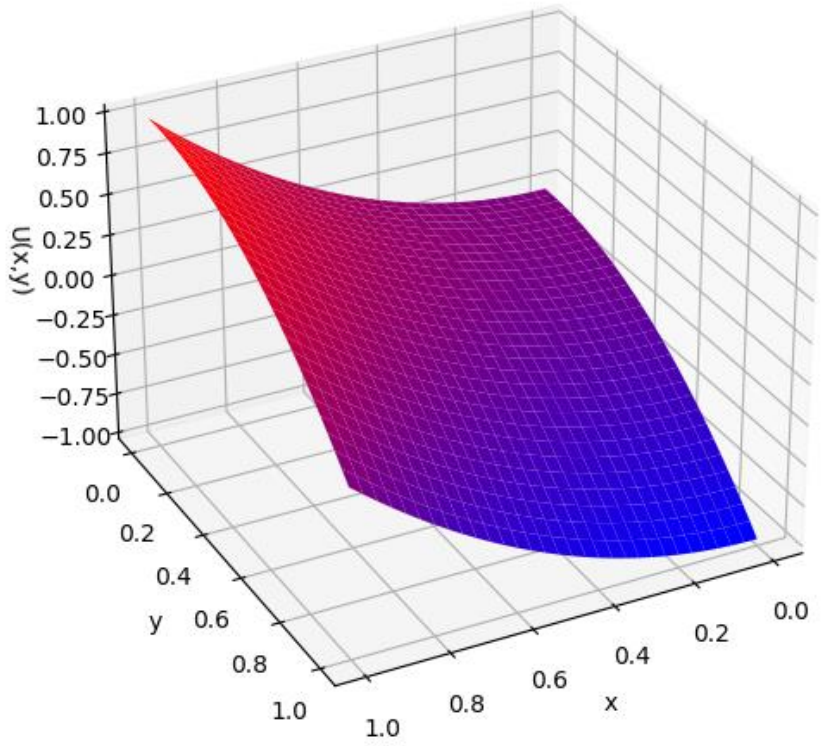
$$u_{0j} = u_{1j} - h_x * 0$$

Результат работы программы

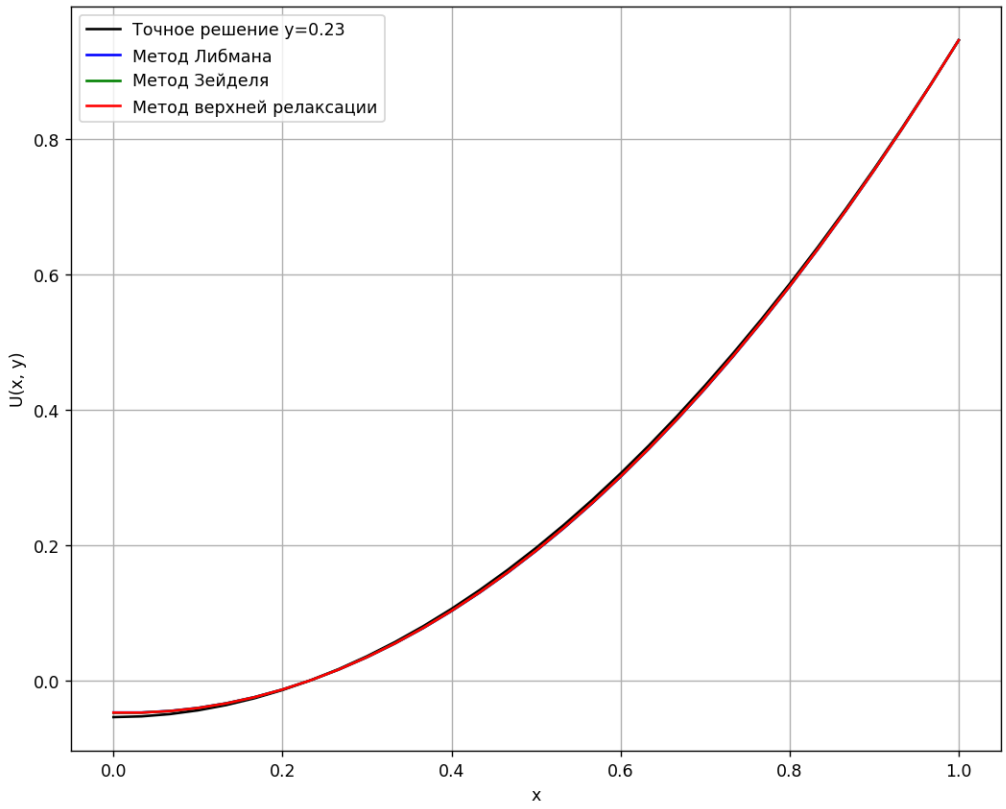
Кол-во итераций метода Либмана: 851

Кол-во итераций метода Зейделя: 498

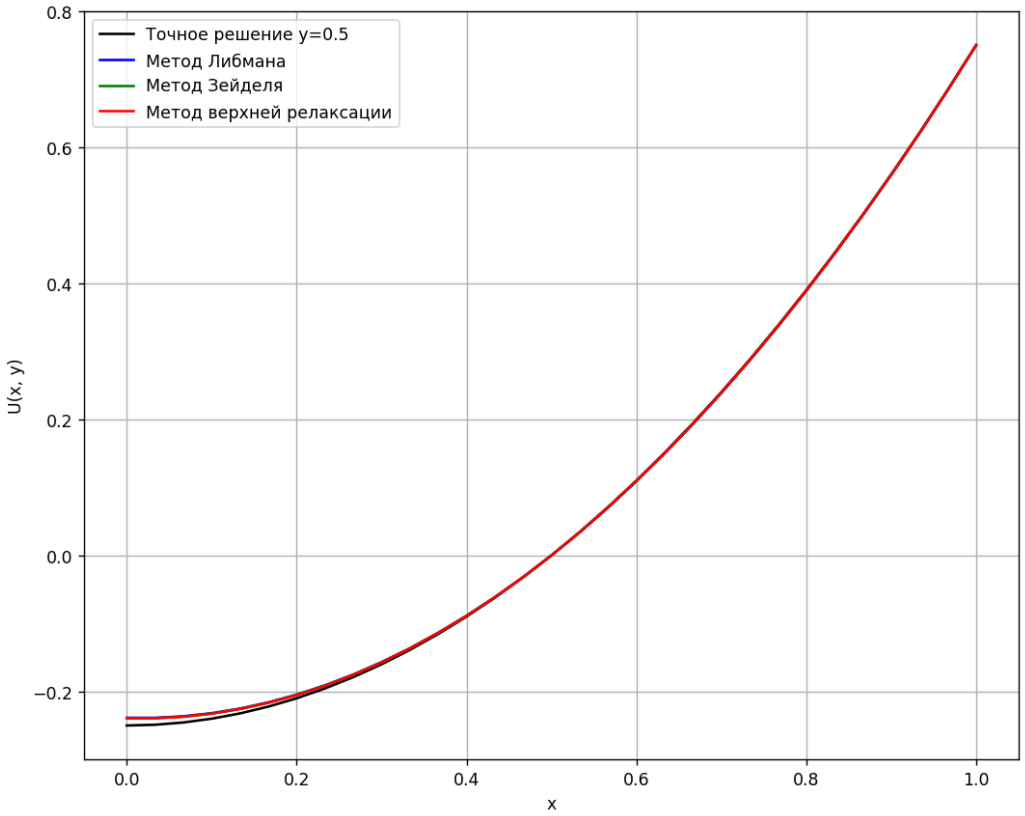
Кол-во итераций верхней релаксации: 411



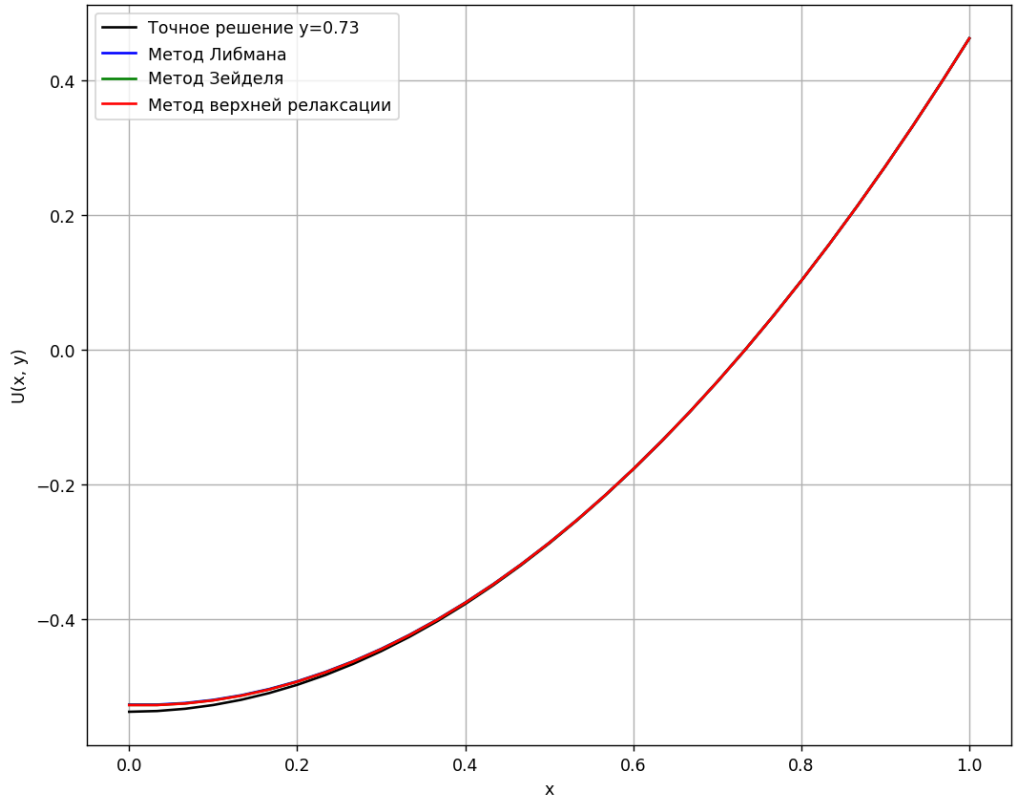
Поверхность, которая задается точным решением



Точное и численные решения, полученные при фиксированном $y = 0{,}23$



Точное и численные решения, полученные при фиксированном $y = 0{,}5$



Точное и численные решения, полученные при фиксированном $y = 0{,}73$

Вывод

Из результатов выполненной лабораторной работы можно заключить, что все рассмотренные методы примерно в равной степени точны, однако количество итераций, за которое эта точность достигается, у каждого из них отличается. Самым быстрым оказался метод простых итераций с верхней релаксацией, за ним следует метод Зейделя, а самым медленным оказался метод простых итераций. Также из графика погрешностей можно заметить четкую тенденцию – при увеличении длины шага увеличивается и погрешность используемого метода.

Код программы

```
import matplotlib.pyplot as plt
import numpy as np
from copy import deepcopy
from matplotlib.colors import LinearSegmentedColormap

FIGSIZE = 10
COLORS = ['blue', 'green', 'red']

plt.rcParams['figure.figsize'] = [FIGSIZE, FIGSIZE]

# Параметры задачи
lx = 1
ly = 1
Nx = 30
Ny = 30
eps = 0.00001
theta = 1.25
hx = lx / Nx
hy = ly / Ny

def phi1(y):
    return 0

def phi2(y):
    return 1 - y * y

def phi3(x):
    return 0

def phi4(x):
    return x * x - 1

def U(x, y):
    return x * x - y * y

# Норма
def norm(v1, v2):
    return np.amax(np.abs(v1 - v2))

def error(lx, hy, Ny, eps, theta):
    NxArray = [5, 10, 20, 40]
    size = np.size(NxArray)
    hxArray = np.zeros(size)
    errors1 = np.zeros(size)
    errors2 = np.zeros(size)
```

```
    errors3 = np.zeros(size)
    for i in range(0, size):
        hxArray[i] = lx / NxArray[i]
        xArray = np.arange(0, lx + hxArray[i], hxArray[i])
        u1, tmp = Liebman(hxArray[i], hy, NxArray[i], Ny, eps)
        u2, tmp = Seidel(hxArray[i], hy, NxArray[i], Ny, eps)
        u3, tmp = SOR(hxArray[i], hy, NxArray[i], Ny, eps, theta)
        if (np.size(xArray) != NxArray[i] + 1):
            xArray = xArray[:NxArray[i] + 1]
        y = hy * Ny / 5
        uCorrect = np.zeros(np.size(xArray))
        for j in range(np.size(xArray)):
            uCorrect[j] = U(xArray[j], y)
        u1Calc = u1[:, int(Ny / 5)]
        u2Calc = u2[:, int(Ny / 5)]
        u3Calc = u3[:, int(Ny / 5)]
        errors1[i] = norm(uCorrect, u1Calc)
        errors2[i] = norm(uCorrect, u2Calc)
        errors3[i] = norm(uCorrect, u3Calc)
    return NxArray, errors1, errors2, errors3


def showErrors(lx, hy, Ny, eps, theta):
    NxArray, errors1, errors2, errors3 = error(lx, hy, Ny, eps, theta)
    colors = ['blue', 'green', 'red']
    deltaX = np.zeros(np.size(NxArray))
    for i in range(np.size(NxArray)):
        deltaX[i] = lx / NxArray[i]
    fig, ax = plt.subplots()
    plt.plot(deltaX, errors1, color=colors[0], label='Метод Либмана')
    plt.plot(deltaX, errors2, color=colors[1], label='Метод Зейделя')
    plt.plot(deltaX, errors3, color=colors[2], label='Метод верхней
релаксации')
    ax.set_xlabel('delta X')
    ax.set_ylabel('Epsilon')
    plt.grid()
    ax.legend()
    plt.show()

def showSolution(Nx, Ny, hx, hy, U, ulieb, usei, usor):
    xArray = np.array([i * hx for i in range(Nx + 1)])
    y = [int(Ny * 0.25), int(Ny * 0.5), int(Ny * 0.75)]
    colors = ['black', 'blue', 'green', 'red']
    for i in range(len(y)):
        fig, ax = plt.subplots()
        uCorrect = U(xArray, y[i] * hy)
        uLiebman = ulieb[y[i]]
        uSeidel = usei[y[i]]
        uSor = usor[y[i]]
        plt.plot(xArray, uCorrect, color=colors[0], label='Точное решение y
= %s' % round(y[i] * hy, 2))
        plt.plot(xArray, uLiebman, color=colors[1], label='Метод Либмана')
        plt.plot(xArray, uSeidel, color=colors[2], label='Метод Зейделя')
        plt.plot(xArray, uSor, color=colors[3], label='Метод верхней
релаксации')
        ax.set_xlabel('x')
        ax.set_ylabel('U(x, y)')
        plt.grid()
        ax.legend()
        plt.show()


def showSolution3d(Nx, Ny, hx, hy, u, elev=45, azim=45):
    x = np.array([i * hx for i in range(Nx + 1)])
    y = np.array([j * hy for j in range(Ny + 1)])
```

```
    xgrid, ygrid = np.meshgrid(x, y)
    z = u
    fig = plt.figure()
    axes = fig.add_subplot(projection='3d')
    cmap = LinearSegmentedColormap.from_list('red_blue', ['b', 'r'], 256)
    axes.view_init(elev=elev, azim=azim)
    axes.set_xlabel('x')
    axes.set_ylabel('y')
    axes.set_zlabel('U(x,y)')
    axes.plot_surface(xgrid, ygrid, z, color='#11aa55', cmap=cmap,
rcount=Nx + 1, ccount=Ny + 1)
    plt.show()


def Liebman(hx, hy, Nx, Ny, eps):
    u = np.zeros((Nx + 1, Ny + 1))
    for i in range(Nx + 1):
        u[i][Ny] = phi4(i * hx)
    for j in range(Ny + 1):
        u[Nx][j] = phi2(j * hy)
    iterNum = 0
    while True:
        uNext = deepcopy(u)
        for i in range(1, Nx):
            for j in range(1, Ny):
                uNext[i][j] = 1 / (2 / hx ** 2 + 2 / hy ** 2) * ((u[i -
1][j] + u[i + 1][j]) / hx ** 2 + (u[i][j - 1] + u[i][j + 1]) / hy ** 2)
        for i in range(Nx):
            uNext[i][0] = uNext[i][1] - hy * phi3(i * hx)
        for j in range(Ny):
            uNext[0][j] = uNext[1][j] - hx * phi1(j * hy)
        if norm(u, uNext) < eps:
            break
        u = deepcopy(uNext)
        iterNum += 1
    return u, iterNum


def Seidel(hx, hy, Nx, Ny, eps):
    u = np.zeros((Nx + 1, Ny + 1))
    for i in range(Nx + 1):
        u[i][Ny] = phi4(i * hx)
    for j in range(Ny + 1):
        u[Nx][j] = phi2(j * hy)
    iterNum = 0
    while True:
        uPrev = deepcopy(u)
        for i in range(1, Nx):
            for j in range(1, Ny):
                u[i][j] = 1 / (2 / hx ** 2 + 2 / hy ** 2) * ((u[i - 1][j] +
u[i + 1][j]) / hx ** 2 + (u[i][j - 1] + u[i][j + 1]) / hy ** 2)
        for i in range(Nx):
            u[i][0] = u[i][1] - hy * phi3(i * hx)
        for j in range(Ny):
            u[0][j] = u[1][j] - hx * phi1(j * hy)
        if norm(u, uPrev) < eps:
            break
        iterNum += 1
    return u, iterNum


def SOR(hx, hy, Nx, Ny, eps, theta):
    u = np.zeros((Nx + 1, Ny + 1))
    for i in range(Nx + 1):
        u[i][Ny] = phi4(i * hx)
```

```
    for j in range(Ny + 1):
        u[Nx][j] = phi2(j * hy)
    iterNum = 0
    while True:
        uPrev = deepcopy(u)
        for i in range(1, Nx):
            for j in range(1, Ny):
                u[i][j] = 1 / (2 / hx ** 2 + 2 / hy ** 2) * ((u[i - 1][j] +
u[i + 1][j]) / hx ** 2 + (u[i][j - 1] + u[i][j + 1]) / hy ** 2)
        for i in range(Nx):
            u[i][0] = u[i][1] - hy * phi3(i * hx)
        for j in range(Ny):
            u[0][j] = u[1][j] - hx * phi1(j * hy)
        u = theta * u + (1 - theta) * uPrev
        if norm(u, uPrev) < eps:
            break
        iterNum += 1
    return u, iterNum


def main():
    u1, iterNum1 = Liebman(hx, hy, Nx, Ny, eps)
    print("Кол-во итераций метода Либмана:", iterNum1)
    u2, iterNum2 = Seidel(hx, hy, Nx, Ny, eps)
    print("Кол-во итераций метода Зейделя:", iterNum2)
    u3, iterNum3 = SOR(hx, hy, Nx, Ny, eps, theta)
    print("Кол-во итераций верхней релаксации:", iterNum3)
    showSolution3d(Nx, Ny, hx, hy, u1.T, elev=20, azim=110)
    showSolution(Nx, Ny, hx, hy, U, u1.T, u2.T, u3.T)
    showErrors(lx, hy, Ny, eps, theta)


main()
```