**Отчет по лабораторной работе №7 по курсу «Численные методы»**

Студент группы 8О-406: Полюбин А.И. Работа выполнена: 20.12.2022

Преподаватель: Пивоваров Д.Е. Отчет сдан: 4.01.2023

Итоговая оценка:

Подпись преподавателя:

Москва 2023

**1. Тема работы**

МЕТОД КОНЕЧНЫХ РАЗНОСТЕЙ ДЛЯ РЕШЕНИЯ УРАВНЕНИЙ ЭЛЛИПТИЧЕСКОГО ТИПА

**2. Цель работы**

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 .

Вариант №4:

,

,

.

Аналитическое решение: .

**3. Ход выполнения работы**

В данной лабораторной реализованы метод простых итераций, метод Зейделя и метод простых итераций с верхней релаксацией, реализована аппроксимация с использованием центрально-разностной схемы. Реализованы функции вычисления погрешности. Графики выводятся при помощи библиотеки matplotlib.

**4. Листинг**

import random

import matplotlib.pyplot as plt

import sys

import warnings

import numpy as np

from functools import reduce

from mpl\_toolkits.mplot3d import Axes3D

#Граничные условия

def psi\_0(x):

return np.sin(x)

def psi\_1(x):

return np.sin(x) \* np.e

def phi\_0(y):

return np.exp(y)

def phi\_1(y):

return -np.exp(y)

#Истинное решение

def u(x, y):

return np.exp(y)\*np.sin(x)

class Schema:

def \_\_init\_\_(self, psi0 = psi\_0, psi1 = psi\_1, phi0 = phi\_0, phi1 = phi\_1, lx0 = 0,

lx1 = np.pi, ly0 = 0, ly1 = 1, solver="zeidel\_method", relax=0.1, epsilon = 0.01):

self.psi1 = psi1

self.psi0 = psi0

self.phi0 = phi0

self.phi1 = phi1

self.lx0 = lx0

self.ly0 = ly0

self.lx1 = lx1

self.ly1 = ly1

self.eps = epsilon

self.method = None

if solver == "zeidel\_method":

self.method = self.zeidel\_step

elif solver == "simple\_iterations\_method":

self.method = self.simple\_eiler\_step

elif solver == "simple\_iterations\_relaxation\_method":

self.method = lambda x, y, m: self.relaxation\_step(x, y, m, relax)

else:

raise ValueError("Wrong solver name")

# зейдель

def zeidel\_step(self, X, Y, M):

return self.relaxation\_step(X, Y, M, w=1)

# релаксация

def relaxation\_step(self, X, Y, M, w):

norm = 0.0

hx2 = self.hx \* self.hx

hy2 = self.hy \* self.hy

for i in range(1, self.Ny - 1):

dif = w \* ((-2 \* self.hx \* self.phi0(Y[i][0]) + 4 \* M[i][1] - M[i][2]) / 3 - M[i][0])

M[i][0] += dif

dif = abs(dif)

norm = dif if dif > norm else norm

for j in range(1, self.Nx - 1):

dif = hy2 \* (M[i][j - 1] + M[i][j + 1])

dif += hx2 \* (M[i - 1][j] + M[i + 1][j])

dif /= 2 \* (hy2 + hx2)

dif -= M[i][j]

dif \*= w

M[i][j] += dif

dif = abs(dif)

norm = dif if dif > norm else norm

dif = w \* ((2 \* self.hx \* self.phi1(Y[i][-1]) + 4 \* M[i][-2] - M[i][-3]) / 3 - M[i][-1])

M[i][-1] += dif

dif = abs(dif)

norm = dif if dif > norm else norm

return norm

# простые итерации

def simple\_eiler\_step(self, X, Y, M):

temp = [[0.0 for \_ in range(self.Nx)] for \_ in range(self.Ny)]

norm = 0.0

hx2 = self.hx \* self.hx

hy2 = self.hy \* self.hy

for i in range(1, self.Ny - 1):

temp[i][0] = (-2 \* self.hx \* self.phi0(Y[i][0]) + 4 \* M[i][1] - M[i][2]) / 3

dif = abs(temp[i][0] - M[i][0])

norm = dif if dif > norm else norm

for j in range(1, self.Nx - 1):

temp[i][j] = hy2 \* (M[i][j - 1] + M[i][j + 1])

temp[i][j] += hx2 \* (M[i - 1][j] + M[i + 1][j])

temp[i][j] /= 2 \* (hy2 + hx2)

dif = abs(temp[i][j] - M[i][j])

norm = dif if dif > norm else norm

temp[i][-1] = (2 \* self.hx \* self.phi1(Y[i][-1]) + 4 \* M[i][-2] - M[i][-3]) / 3

dif = abs(temp[i][0] - M[i][0])

norm = dif if dif > norm else norm

for i in range(1, self.Ny - 1):

M[i] = temp[i]

return norm

def set\_l0\_l1(self, lx0, lx1, ly0, ly1):

self.lx0 = lx0

self.lx1 = lx1

self.ly0 = ly0

self.ly1 = ly1

def \_compute\_h(self):

self.hx = (self.lx1 - self.lx0) / (self.Nx - 1)

self.hy = (self.ly1 - self.ly0) / (self.Ny - 1)

@staticmethod

def nparange(start, end, step = 1):

now = start

e = 0.00000000001

while now - e <= end:

yield now

now += step

def init\_values(self, X, Y):

ans = [[0 for \_ in range(self.Nx)] for \_ in range(self.Ny)]

for j in range(self.Nx):

coeff = (self.psi1(X[-1][j]) - self.psi0(X[0][j])) / (self.ly1 - self.ly0)

addition = self.psi0(X[0][j])

for i in range(self.Ny):

ans[i][j] = coeff\*(Y[i][j] - self.ly0) + addition

return ans

def \_\_call\_\_(self, Nx=10, Ny=10):

self.Nx, self.Ny = Nx, Ny

self.\_compute\_h()

x = list(self.nparange(self.lx0, self.lx1, self.hx))

y = list(self.nparange(self.ly0, self.ly1, self.hy))

X = [x for \_ in range(self.Ny)]

Y = [[y[i] for \_ in x] for i in range(self.Ny)]

ans = self.init\_values(X, Y)

self.itters = 0

while(self.method(X, Y, ans) >= self.eps):

self.itters += 1

return X, Y, ans

def real\_z(lx0, lx1, ly0, ly1, f):

x = np.arange(lx0, lx1 + 0.005, 0.005)

y = np.arange(ly0, ly1 + 0.005, 0.005)

X = np.ones((y.shape[0], x.shape[0]))

Y = np.ones((x.shape[0], y.shape[0]))

Z = np.ones((y.shape[0], x.shape[0]))

for i in range(Y.shape[0]):

Y[i] = y

Y = Y.T

for i in range(X.shape[0]):

X[i] = x

for i in range(Z.shape[0]):

for j in range(Z.shape[1]):

Z[i, j] = f(X[i, j], Y[i, j])

return X, Y, Z

def All\_plot(Nx=10, Ny=10, eps=0.001, met="simple\_iterations\_relaxation\_method"):

schema = Schema(epsilon=eps, solver=met)

x, y, z = schema(Nx, Ny)

fig = plt.figure(num=1, figsize=(19, 12), clear=True)

ax = fig.add\_subplot(1, 1, 1, projection='3d')

ax.plot\_wireframe(\*real\_z(0, np.pi, 0, 1, u), color="blue")

ax.plot\_surface(np.array(x), np.array(y), np.array(z))

ax.set(xlabel='x', ylabel='y', zlabel='z',

title=f'График решения и реальной функции методом {met}')

fig.tight\_layout()

plt.show()

#Метод релаксации

All\_plot()

None

#Метод Зейделя

All\_plot(met="zeidel\_method")

None

#Метод простых итераций

All\_plot(met="simple\_iterations\_method")

None

schema = Schema(epsilon=0.001, solver="simple\_iterations\_method")

schema(10, 10)

print("Количество итераций метода простых иттераций:", schema.itters)

schema = Schema(epsilon=0.001, solver='zeidel\_method')

schema(10, 10)

print("Количество иттераций метода Зейделя:", schema.itters)

schema = Schema(epsilon=0.001, solver='simple\_iterations\_relaxation\_method', relax=1.5)

schema(10, 10)

print("Количество итераций метода простых итераций с релаксацией:", schema.itters)

def epsilon(x, y, z, f):

ans = 0.0

for i in range(len(z)):

for j in range(len(z[i])):

ans += (z[i][j] - f(x[i][j], y[i][j]))\*\*2

return (ans / (len(z) \* len(z[0])))\*\*0.5

def graphic\_h(solver, real\_f):

h = []

e = []

for N in range(4, 80, 3):

x, y, z = solver(N, N)

h.append(solver.hx)

e.append(epsilon(x, y, z, real\_f))

return h, e

explict = Schema(epsilon=0.00001)

plt.figure(figsize = (16, 10))

plt.subplot(2, 1, 1)

plt.title("Зависимость погрешности от длины шага")

h, e = graphic\_h(explict, u)

plt.plot(h, e, label="Значение погрешности", color = "blue")

plt.xlabel("h")

plt.ylabel("e")

plt.legend()

plt.grid()

plt.subplot(2, 1, 2)

#В логарифмической шкале

plt.plot(list(map(np.log, h)), list(map(np.log, e)), label="Полученная зависимость")

plt.plot([-3, -0.5], [-3, -0.5], label="Зависимость $O(h)$")

plt.plot([-3, -0.5], [-3, 2], label="Зависимость $O(h^2)$")

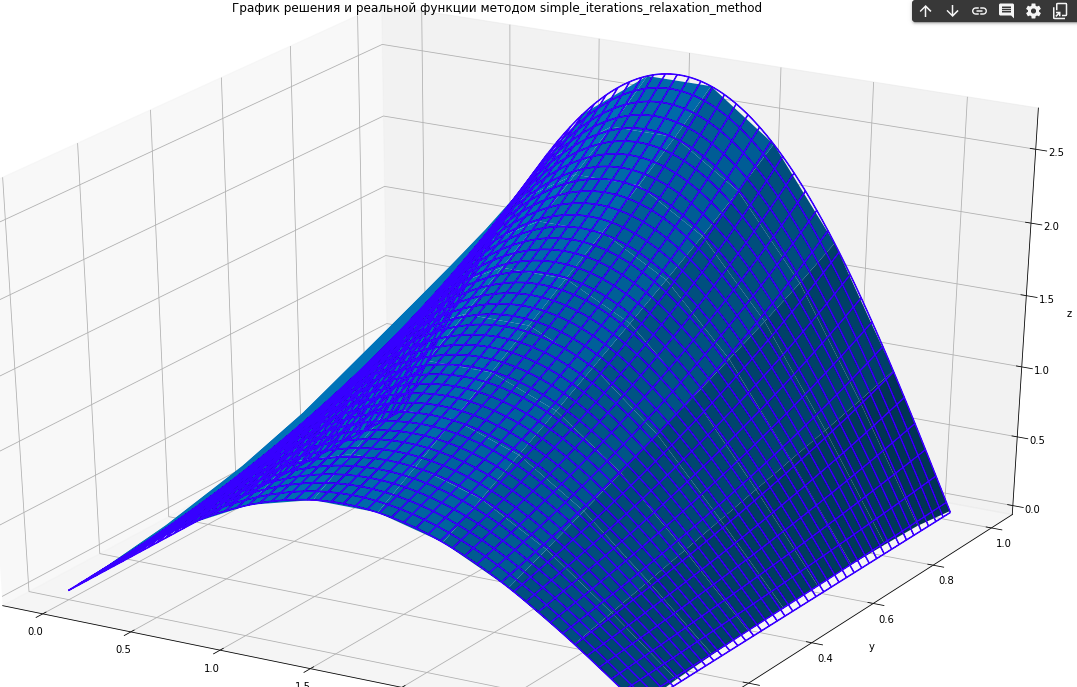
plt.xlabel("log h")

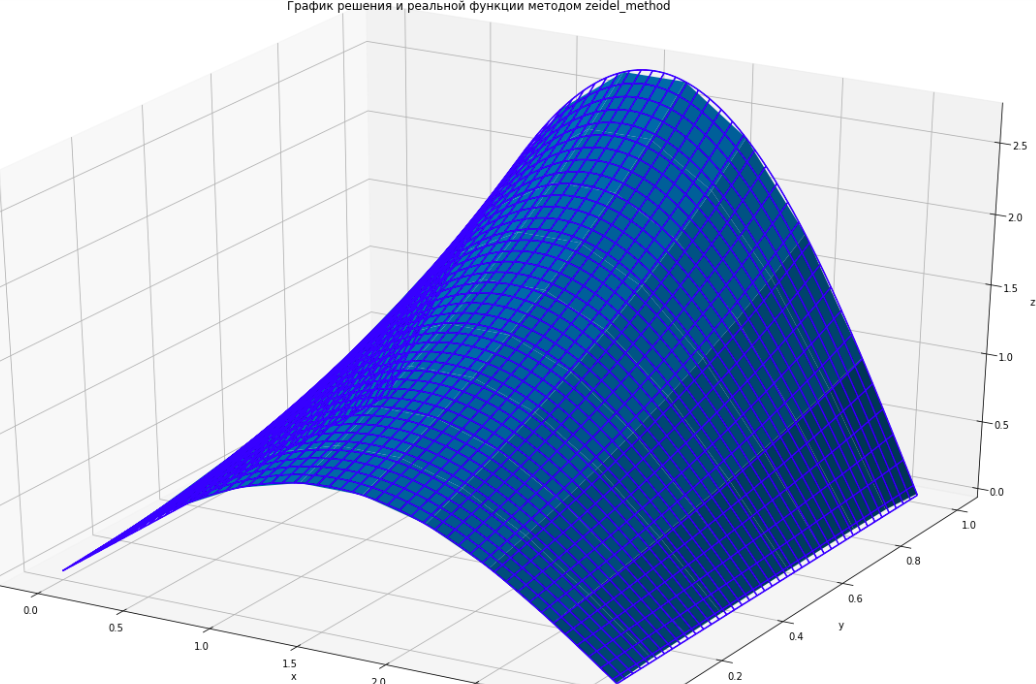
plt.ylabel("log e")

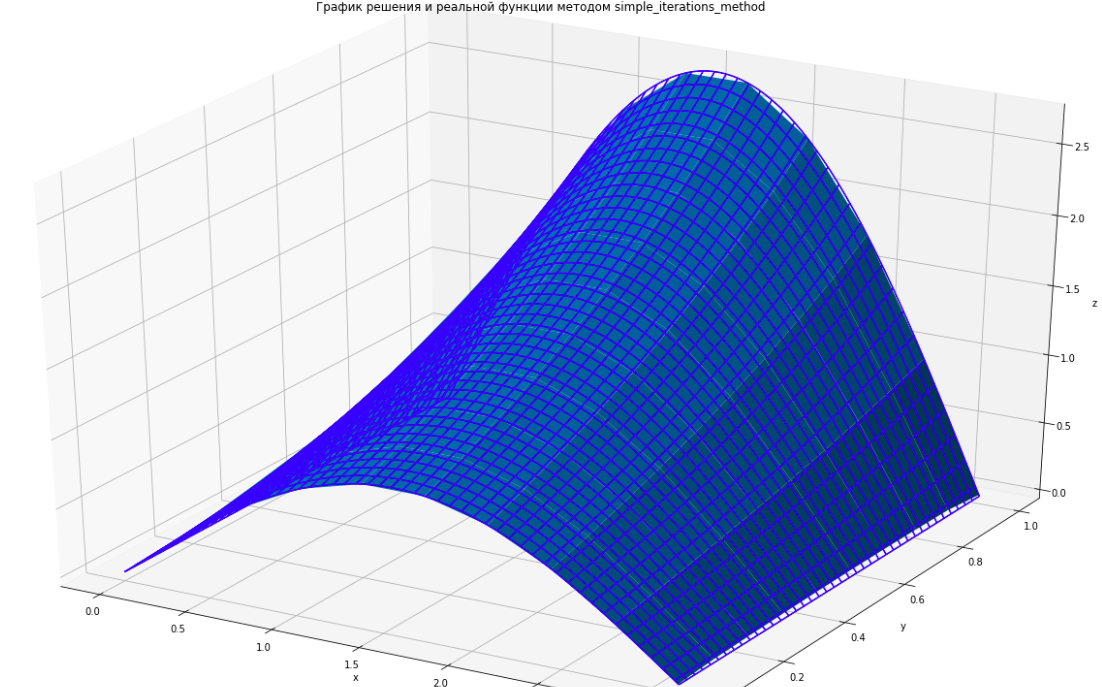
plt.legend()

plt.grid()

**5. Результаты**

****

****

****

Сходимость:

Количество итераций метода простых иттераций: 41

Количество иттераций метода Зейделя: 27

Количество итераций метода простых итераций с релаксацией: 13

**6. Выводы**

При выполнении данной лабораторной работы я научился находить численное решение уравнений эллиптического типа методами простых итераций и метода Зейделя, а также аппроксимировать граничные значения разными способами. Как мы видим по результатам метод простых итераций с релаксацией сходится за наименьшее число шагов.