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

Студент группы 8О-406: Киреев А. К.
Работа выполнена: 13.11.2022
Преподаватель: Пивоваров Д.Е.
Отчет сдан:
Итоговая оценка:
Подпись преподавателя:

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 U(x, y). Исследовать зависимость погрешности от сеточных параметров hx, hy.

(d^2u)/(dx^2) + (d^2u)/(dy^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

3. Ход выполнения работы

Реализованы метод простых итераций, метод Зейделя и метод простых итераций с верхней релаксацией, реализована аппроксимация с использованием центрально-разностной схемы. Реализована функция вычисления погрешности. Графики выводятся при помощи библиотеки matplotlib.

Код на Python:

main.py:

```
import numpy as np
import sys
import matplotlib.pyplot as plt
from matplotlib import cm

from methods import simple_iter_method, relaxation_iter_method, seidel_method

sys.path.append(".")


def analytical_solution(x: float, y: float) -> float:
   return x**2 - y**2


def analytical_grid(x: np.ndarray, y: np.ndarray) -> np.ndarray:
   grid: np.ndarray = np.zeros(shape=(len(y), len(x)))
   for i in range(len(y)):
       for j in range(len(x)):
           grid[i, j] = analytical_solution(x[j], y[i])
   return grid


def u_y_initial_0_dx(y: np.ndarray) -> np.ndarray:
   return np.zeros(len(y))


def u_y_initial_1(y: np.ndarray) -> np.ndarray:
   return 1.0 - y**2


def u_x_initial_0_dy(x: np.ndarray) -> np.ndarray:
   return np.zeros(len(x))


def u_x_initial_1(x: np.ndarray) -> np.ndarray:
   return x**2 - 1.0


def error(numeric: np.ndarray, analytical: np.ndarray) -> np.ndarray:
   return np.abs(numeric - analytical).max()


def draw(numerical: np.ndarray, analytical: np.ndarray,
        x: np.ndarray, y: np.ndarray,
        title_lhs: str, title_rhs: str):
   fig = plt.figure(figsize=plt.figaspect(0.7))
   xx, yy = np.meshgrid(x, y)

   ax = fig.add_subplot(1, 2, 1, projection='3d')
   plt.title(title_lhs)
   ax.set_xlabel('x', fontsize=20)
   ax.set_ylabel('y', fontsize=20)
   ax.set_zlabel('u', fontsize=20)
   ax.plot_surface(xx, yy, numerical, cmap=cm.coolwarm, linewidth=0, antialiased=True)

   ax = fig.add_subplot(1, 2, 2, projection='3d')
   ax.set_xlabel('x', fontsize=20)
   ax.set_ylabel('y', fontsize=20)
   ax.set_zlabel('u', fontsize=20)
   plt.title(title_rhs)
   ax.plot_surface(xx, yy, analytical, cmap=cm.coolwarm, linewidth=0, antialiased=True)

   plt.show()


if __name__ == "__main__":
   h = float(input("Enter step 'h': "))
   x: np.ndarray = np.arange(0, 1.0 + h/2.0, step=h)
   y: np.ndarray = np.arange(0, 1.0 + h/2.0, step=h)

   kwargs = {
       "u_y_initial_0_dx": u_y_initial_0_dx,
```

```
        "u_y_initial_1": u_y_initial_1,
        "u_x_initial_0_dy": u_x_initial_0_dy,
        "u_x_initial_1": u_x_initial_1,
        "h": h,
        "l": 0.0,
        "r": 1.0
    }

    analytical = analytical_grid(x, y)

    print("---------------- SIMPLE ITER ----------------")
    sol = simple_iter_method(**kwargs)
    print(np.round(sol, 2))
    print("\nError: ", error(sol, analytical))
    print("---------------------------------------------")
    print("--------------- ANALYTICAL ------------------")
    print(np.round(analytical, 2))
    draw(sol, analytical, x, y, 'simple iter', 'analytic')

    print("\n\n---------------- SEIDEL ----------------")
    sol = seidel_method(**kwargs)
    print(np.round(sol, 2))
    print("\nError: ", error(sol, analytical))
    print("---------------------------------------")
    print("-------------- ANALYTICAL ---------------")
    print(np.round(analytical, 2))
    draw(sol, analytical, x, y, 'seidel', 'analytic')

    print("\n\n---------------- RELAXATION ITER ----------------")
    sol = relaxation_iter_method(**kwargs, w=1.5)
    print(np.round(sol, 2))
    print("\nError: ", error(sol, analytical))
    print("-------------------------------------------------")
    print("------------------ ANALYTICAL -------------------")
    print(np.round(analytical, 2))
    draw(sol, analytical, x, y, 'relaxation iter', 'analytic')
```

methods.py:

```
import copy
import numpy as np
from typing import List, Callable
from functools import partial

from logger import base_logger


def init_u(x: np.ndarray, y: np.ndarray,
           u_x_initial_1: Callable, u_y_initial_1: Callable,
           l: float, r: float) -> np.ndarray:
    u: np.ndarray = np.zeros(shape=(len(y), len(x)))
    u[:, -1] = u_y_initial_1(y)
    u[-1, :] = u_x_initial_1(x)
    for j in range(len(x) - 2, -1, -1):
        for i in range(len(y) - 2, -1, -1):
            u[i, j] = u[i + 1, j] * x[j] / (x[j] + y[i] + 0.0001)
            u[i, j] += u[i, j + 1] * y[i] / (x[j] + y[i] + 0.0001)
    return u


def simple_iter_method(u_y_initial_0_dx: Callable, u_y_initial_1: Callable,
                       u_x_initial_0_dy: Callable, u_x_initial_1: Callable,
                       h: float, l: float, r: float) -> np.ndarray:
    x: np.ndarray = np.arange(0, 1.0 + h/2.0, step=h)
    y: np.ndarray = np.arange(0, 1.0 + h/2.0, step=h)
    u: np.ndarray = init_u(x, y, u_x_initial_1, u_y_initial_1, l, r)

    eps: float = 1e-4
    prev: np.ndarray = np.zeros(shape=(len(y), len(x)))
    curr_iter: int = 0
    max_iter: int = 100
    diff: float = np.abs(u - prev).max()
    diverge_coef: float = 1.5
    while diff > eps and curr_iter <= max_iter:
        prev_diff: float = diff
        prev = copy.deepcopy(u)
```

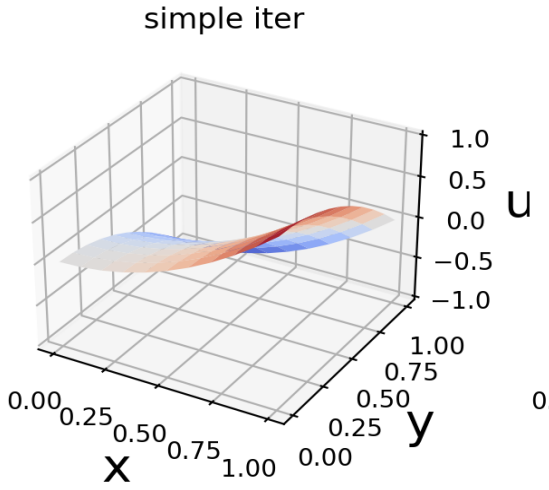
```
        u[0, :-1] = (-2.0 * h * u_x_initial_0_dy(x[:-1]) + 4.0 * prev[1, :-1] - prev[2, :-1]) /
3.0
        u[:-1, 0] = (-2.0 * h * u_y_initial_0_dx(y[:-1]) + 4.0 * prev[:-1, 1] - prev[:-1, 2]) /
3.0
       for i in range(1, len(y) - 1):
           for j in range(1, len(x) - 1):
               u[i, j] = (prev[i-1, j] + prev[i+1, j] + prev[i, j-1] + prev[i, j+1]) / 4.0

       diff = np.abs(u - prev).max()
       curr_iter += 1
       if diff > diverge_coef * prev_diff:
           base_logger.warning("WARNING : Max_iter starts diverge on iter = %s", curr_iter)
           break

   if curr_iter >= max_iter:
       base_logger.warning("WARNING : Max_iter was reached")

   base_logger.info("INFO : iters count = %s", curr_iter)

   return u


def relaxation_iter_method(u_y_initial_0_dx: Callable, u_y_initial_1: Callable,
                           u_x_initial_0_dy: Callable, u_x_initial_1: Callable,
                           h: float, l: float, r: float,
                           w: float) -> np.ndarray:
   x: np.ndarray = np.arange(0, 1.0 + h/2.0, step=h)
   y: np.ndarray = np.arange(0, 1.0 + h/2.0, step=h)
   u: np.ndarray = init_u(x, y, u_x_initial_1, u_y_initial_1, l, r)

   eps: float = 1e-4
   prev: np.ndarray = np.zeros(shape=(len(y), len(x)))
   curr_iter: int = 0
   max_iter: int = 100
   diff: float = np.abs(u - prev).max()
   diverge_coef: float = 1.5
   while diff > eps and curr_iter <= max_iter:
       prev_diff: float = diff
       prev = copy.deepcopy(u)

        u[0, :-1] += w * ((-2.0 * h * u_x_initial_0_dy(x[:-1]) + 4.0 * u[1, :-1] - u[2, :-1]) /
3.0 - u[0, :-1])
        u[:-1, 0] += w * ((-2.0 * h * u_y_initial_0_dx(y[:-1]) + 4.0 * u[:-1, 1] - u[:-1, 2]) /
3.0 - u[:-1, 0])
       for i in range(1, len(y) - 1):
           for j in range(1, len(x) - 1):
                 u[i, j] += w * ((u[i-1, j] + prev[i+1, j] + u[i, j-1] + prev[i, j+1]) / 4.0 -
prev[i, j])

       diff = np.abs(u - prev).max()
       curr_iter += 1
       if diff > diverge_coef * prev_diff:
           base_logger.warning("WARNING : Max_iter starts diverge on iter = %s", curr_iter)
           break

   if curr_iter >= max_iter:
       base_logger.warning("WARNING : Max_iter was reached")

   base_logger.info("INFO : iters count = %s", curr_iter)

   return u


seidel_method = partial(relaxation_iter_method, w=1.0)
```

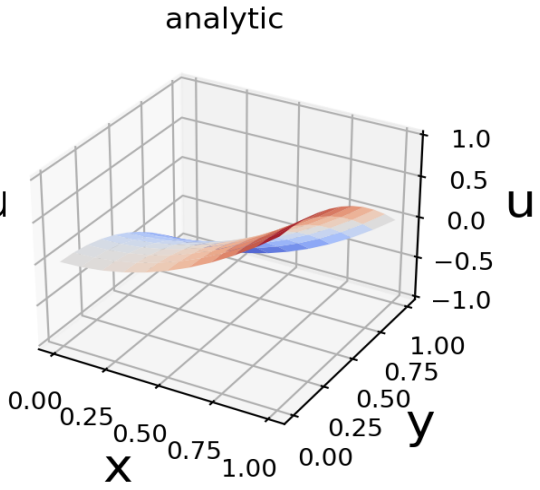
4. Результаты

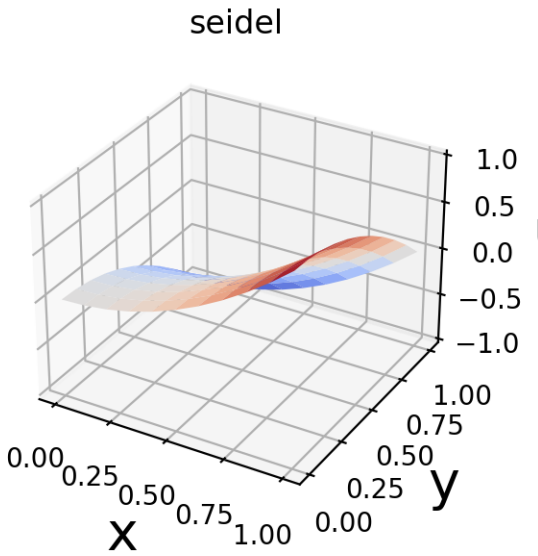
```
(venv) ak@MacBook-Air-AK lab7 % python3 main.py
Enter step 'h': 0.1
---------------- SIMPLE ITER ----------------
INFO : iters count = 70
[[-0.    0.01  0.04  0.09  0.16  0.25  0.36  0.49  0.64  0.81  1.  ]
[-0.01 -0.    0.03  0.08  0.15  0.24  0.35  0.48  0.63  0.8   0.99]
[-0.04 -0.03 -0.    0.05  0.12  0.21  0.32  0.45  0.6   0.77  0.96]
```

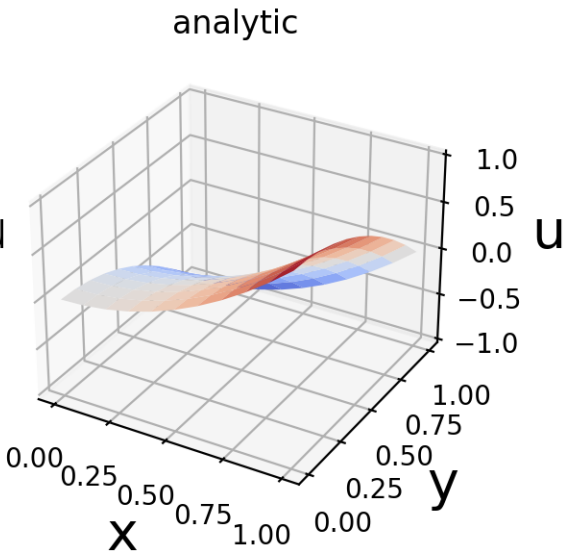
```
[-0.09 -0.08 -0.05 -0.    0.07  0.16  0.27  0.4   0.55  0.72  0.91]
[-0.16 -0.15 -0.12 -0.07  0.    0.09  0.2   0.33  0.48  0.65  0.84]
[-0.25 -0.24 -0.21 -0.16 -0.09  0.    0.11  0.24  0.39  0.56  0.75]
[-0.36 -0.35 -0.32 -0.27 -0.2  -0.11  0.    0.13  0.28  0.45  0.64]
[-0.49 -0.48 -0.45 -0.4  -0.33 -0.24 -0.13  0.    0.15  0.32  0.51]
[-0.64 -0.63 -0.6  -0.55 -0.48 -0.39 -0.28 -0.15  0.    0.17  0.36]
[-0.81 -0.8  -0.77 -0.72 -0.65 -0.56 -0.45 -0.32 -0.17  0.    0.19]
[-1.   -0.99 -0.96 -0.91 -0.84 -0.75 -0.64 -0.51 -0.36 -0.19  0.  ]]

Error:  0.000836917537235804
---------------------------------------------
--------------- ANALYTICAL ------------------
[[ 0.    0.01  0.04  0.09  0.16  0.25  0.36  0.49  0.64  0.81  1.  ]
[-0.01  0.    0.03  0.08  0.15  0.24  0.35  0.48  0.63  0.8   0.99]
[-0.04 -0.03  0.    0.05  0.12  0.21  0.32  0.45  0.6   0.77  0.96]
[-0.09 -0.08 -0.05  0.    0.07  0.16  0.27  0.4   0.55  0.72  0.91]
[-0.16 -0.15 -0.12 -0.07  0.    0.09  0.2   0.33  0.48  0.65  0.84]
[-0.25 -0.24 -0.21 -0.16 -0.09  0.    0.11  0.24  0.39  0.56  0.75]
[-0.36 -0.35 -0.32 -0.27 -0.2  -0.11  0.    0.13  0.28  0.45  0.64]
[-0.49 -0.48 -0.45 -0.4  -0.33 -0.24 -0.13  0.    0.15  0.32  0.51]
[-0.64 -0.63 -0.6  -0.55 -0.48 -0.39 -0.28 -0.15  0.    0.17  0.36]
[-0.81 -0.8  -0.77 -0.72 -0.65 -0.56 -0.45 -0.32 -0.17  0.    0.19]
[-1.   -0.99 -0.96 -0.91 -0.84 -0.75 -0.64 -0.51 -0.36 -0.19  0.  ]]


---------------- SEIDEL -----------------
INFO : iters count = 37
[[-0.    0.01  0.04  0.09  0.16  0.25  0.36  0.49  0.64  0.81  1.  ]
[-0.01  0.    0.03  0.08  0.15  0.24  0.35  0.48  0.63  0.8   0.99]
[-0.04 -0.03  0.    0.05  0.12  0.21  0.32  0.45  0.6   0.77  0.96]
[-0.09 -0.08 -0.05  0.    0.07  0.16  0.27  0.4   0.55  0.72  0.91]
[-0.16 -0.15 -0.12 -0.07  0.    0.09  0.2   0.33  0.48  0.65  0.84]
[-0.25 -0.24 -0.21 -0.16 -0.09  0.    0.11  0.24  0.39  0.56  0.75]
[-0.36 -0.35 -0.32 -0.27 -0.2  -0.11  0.    0.13  0.28  0.45  0.64]
[-0.49 -0.48 -0.45 -0.4  -0.33 -0.24 -0.13 -0.    0.15  0.32  0.51]
[-0.64 -0.63 -0.6  -0.55 -0.48 -0.39 -0.28 -0.15  0.    0.17  0.36]
[-0.81 -0.8  -0.77 -0.72 -0.65 -0.56 -0.45 -0.32 -0.17  0.    0.19]
[-1.   -0.99 -0.96 -0.91 -0.84 -0.75 -0.64 -0.51 -0.36 -0.19  0.  ]]

Error:  0.0008244650397923881
-----------------------------------------
-------------- ANALYTICAL ---------------
[[ 0.    0.01  0.04  0.09  0.16  0.25  0.36  0.49  0.64  0.81  1.  ]
[-0.01  0.    0.03  0.08  0.15  0.24  0.35  0.48  0.63  0.8   0.99]
[-0.04 -0.03  0.    0.05  0.12  0.21  0.32  0.45  0.6   0.77  0.96]
[-0.09 -0.08 -0.05  0.    0.07  0.16  0.27  0.4   0.55  0.72  0.91]
[-0.16 -0.15 -0.12 -0.07  0.    0.09  0.2   0.33  0.48  0.65  0.84]
[-0.25 -0.24 -0.21 -0.16 -0.09  0.    0.11  0.24  0.39  0.56  0.75]
[-0.36 -0.35 -0.32 -0.27 -0.2  -0.11  0.    0.13  0.28  0.45  0.64]
[-0.49 -0.48 -0.45 -0.4  -0.33 -0.24 -0.13  0.    0.15  0.32  0.51]
[-0.64 -0.63 -0.6  -0.55 -0.48 -0.39 -0.28 -0.15  0.    0.17  0.36]
[-0.81 -0.8  -0.77 -0.72 -0.65 -0.56 -0.45 -0.32 -0.17  0.    0.19]
[-1.   -0.99 -0.96 -0.91 -0.84 -0.75 -0.64 -0.51 -0.36 -0.19  0.  ]]


---------------- RELAXATION ITER ----------------
INFO : iters count = 16
[[ 0.    0.01  0.04  0.09  0.16  0.25  0.36  0.49  0.64  0.81  1.  ]
[-0.01 -0.    0.03  0.08  0.15  0.24  0.35  0.48  0.63  0.8   0.99]
[-0.04 -0.03  0.    0.05  0.12  0.21  0.32  0.45  0.6   0.77  0.96]
[-0.09 -0.08 -0.05  0.    0.07  0.16  0.27  0.4   0.55  0.72  0.91]
[-0.16 -0.15 -0.12 -0.07  0.    0.09  0.2   0.33  0.48  0.65  0.84]
[-0.25 -0.24 -0.21 -0.16 -0.09  0.    0.11  0.24  0.39  0.56  0.75]
[-0.36 -0.35 -0.32 -0.27 -0.2  -0.11 -0.    0.13  0.28  0.45  0.64]
[-0.49 -0.48 -0.45 -0.4  -0.33 -0.24 -0.13  0.    0.15  0.32  0.51]
[-0.64 -0.63 -0.6  -0.55 -0.48 -0.39 -0.28 -0.15  0.    0.17  0.36]
[-0.81 -0.8  -0.77 -0.72 -0.65 -0.56 -0.45 -0.32 -0.17 -0.    0.19]
[-1.   -0.99 -0.96 -0.91 -0.84 -0.75 -0.64 -0.51 -0.36 -0.19  0.  ]]

Error:  0.00016582723471558758
-------------------------------------------------
------------------ ANALYTICAL -------------------
[[ 0.    0.01  0.04  0.09  0.16  0.25  0.36  0.49  0.64  0.81  1.  ]
[-0.01  0.    0.03  0.08  0.15  0.24  0.35  0.48  0.63  0.8   0.99]
[-0.04 -0.03  0.    0.05  0.12  0.21  0.32  0.45  0.6   0.77  0.96]
[-0.09 -0.08 -0.05  0.    0.07  0.16  0.27  0.4   0.55  0.72  0.91]
```

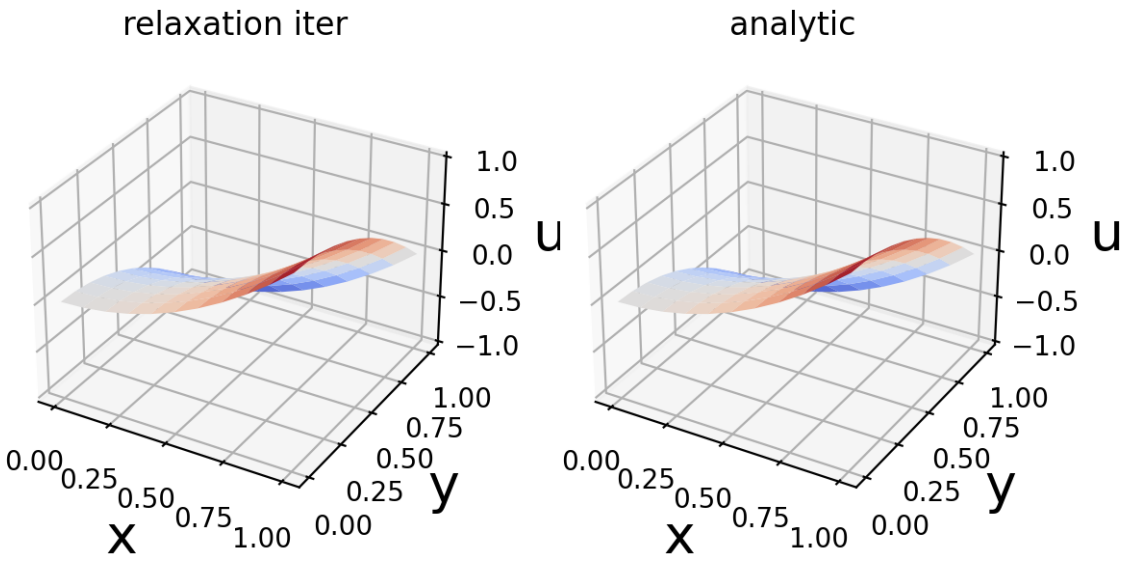
```
[-0.16 -0.15 -0.12 -0.07  0.    0.09  0.2   0.33  0.48  0.65  0.84]
[-0.25 -0.24 -0.21 -0.16 -0.09  0.    0.11  0.24  0.39  0.56  0.75]
[-0.36 -0.35 -0.32 -0.27 -0.2  -0.11  0.    0.13  0.28  0.45  0.64]
[-0.49 -0.48 -0.45 -0.4  -0.33 -0.24 -0.13  0.    0.15  0.32  0.51]
[-0.64 -0.63 -0.6  -0.55 -0.48 -0.39 -0.28 -0.15  0.    0.17  0.36]
[-0.81 -0.8  -0.77 -0.72 -0.65 -0.56 -0.45 -0.32 -0.17  0.    0.19]
[-1.   -0.99 -0.96 -0.91 -0.84 -0.75 -0.64 -0.5 1 -0.36 -0.19  0. ]]
```











5. Выводы

В данной лабораторной работе я научился находить численное решение уравнений эллиптического типа при помощи метода простых итераций и метода Зейделя, а также аппроксимировать граничные значения разными способами.