Московский авиационный институт

(Национальный исследовательский университет)

Институт №8 «Компьютерные науки и прикладная математика»

Кафедра вычислительной математики и программирования

Лабораторные работы

по курсу «Численные методы»

Вариант 17

Выполнил: Черных С. Д.

Группа: М8О-405Б-20

Проверил: доц. Иванов И. Э.

Дата:

Оценка:

Москва, 2023

Лабораторная работа №7

Используя центрально-разностную схему и метод Либмана, решить краевую задачу для дифференциального уравнения эллиптического типа. Вычислить погрешность численного решения путем сравнения результатов с приведенным в задании аналитическим решением U(x,y). Исследовать зависимость погрешности от сеточного параметра h. Реализовать ускорение сходимости итерационной процедуры методом Зейделя и методом верхней релаксации.

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

$u_x(0, y) = y$

$u_x\left(\frac{\pi}{2}, y\right) = 0$

$u(x, 0) = x$

$u(x, 1) = \sin(x)$

Аналитическое решение: $U(x, y) = y\sin(\text{x})$

Теоретические сведения

На прямоугольнике $x \in [0, l_x], y \in \left[0, l_y\right]$ 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 и нижнюю строки и первый столбец сетки, так как они задаются граничными условиями первого рода. На этой сетке аппроксимируем дифференциальную задачу во внутренних узлах с помощью отношения конечных разностей:

$$\frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h_x^2} + \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{h_y^2} + c\, u_{ij} = 0$$

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

Метод Либмана:

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

Метод Зейделя:

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

Метод простых итераций с верхней релаксацией:

В общем случае метод основан на следующей схеме:

Пусть $u^{k+1} = \alpha u^k + \beta$, тогда $u^{k+1} = \omega(\alpha u^k + \beta) + (1 - \omega)u^k$, где $\omega \in (1, 2)$

В данной работе:

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

Код программы

Программа выполнена на языке python, в среде разработки Jupyter Notebook. Код программы будет в виде незапущенного кода в Jupyter Notebook, вывод программы, сокращён до графиков, построенных в среде Jupyter Notebook.

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.

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

$ u_x(0, y) = y \ u_x(\frac{\pi}{2}, y) = 0 \ u(x, 0) = 0 \ u(x, 1) = sin(x) $

$$U(x, y) = ysin(x)$$

```
import math
import typing
from typing import List
import matplotlib.pyplot as plt
import copy
import plotly.graph_objects as go
```

Входные условия

```
x_left = 0
x_right = math.pi/2
y_left = 0
y_right = 1
a = 1
b = 1
c = 1
d = 0
alpha_1, betta_1 = 1, 0
alpha_2, betta_2 = 1, 0
alpha_3, betta_3 = 0, 1
alpha_4, betta_4 = 0, 1

def phi_1(y:float) -> float:
    return y

def phi_2(y:float) -> float:
    return 0

def phi_3(x:float) -> float:
    return 0

def phi_4(x:float) -> float:
    return math.sin(x)

def f(x:float, y:float) -> float:
    return 0

def U(x:float, y:float) -> float:
    return y * math.sin(x)

def real_U(X:list, Y:list) -> list:
```

```python
    n = len(X)
    m = len(Y)
    U_true = [[0] * n for _ in range(m)]
    for k in range(m):
        for j in range(n):
            U_true[k][j] = U(X[j], Y[k])
    return U_true
```

Вспомогательные функции

графики

```python
def plot_graphs(new_X:list, new_T:list, found_U:list, U_true:list, s:str='') -> None:
    plt.plot(new_X, U_true[len(new_T) // 4 ],  label='Аналитическое решение')
    plt.plot(new_X, found_U[len(new_T) // 4 ], label=s, linestyle='dashdot')
    plt.plot(new_X, U_true[len(new_T) // 2 ], label='Аналитическое решение')
    plt.plot(new_X, found_U[len(new_T) // 2 ], label=s, linestyle='dashdot')
    plt.plot(new_X, U_true[len(new_T) - 1], label='Аналитическое решение')
    plt.plot(new_X, found_U[len(new_T) - 1],label=s, linestyle='dashdot')
    plt.legend()
```

график ошибки от h

```python
def local_error (U_my:list, U_true:list) -> float:
    return sum([(a - b) ** 2 for a, b in zip(U_my, U_true)]) / len(U_my)

def get_error_array_with_h(N:list, x_left:float, x_right:float, y_left:float, y_right:
float, a:float, b:float, c:float, d:float,
                           find_u:typing.Callable, eps:float=0.1, omega:float=-10)  ->
(list, list): # H, error


    H_x = [x_right/(n - 1) for n in N]
    H_y = [y_right/(n - 1) for n in N]
    ERROR = []
    for n in N:
        if omega == -10:
            XX, YY, UU = find_u(x_left, x_right, y_left, y_right, a, b, c, d, eps=eps,
n_x=n, n_y=n)
        else:
            XX, YY, UU = find_u(x_left, x_right, y_left, y_right, a, b, c, d=0, eps=ep
s, n_x=n, n_y=n, omega=omega)
        U_true = real_U(XX, YY)
        t = len(YY) // 2
        ERROR.append(local_error(UU[t], U_true[t]))

    return H_x, H_y ,ERROR

def h_error_plot(H:list, ERROR:list, s:str=' x') -> None:
    plt.plot(H, ERROR)
    plt.xlabel("h" + s)
    plt.ylabel("error")
    plt.show()

def frange(start:float, stop:float, step:float) -> float:
    while start < stop:
        yield start
        start += step

def get_y(y0:float, y_end:float, h:float) -> list:
    return [i for i in frange(y0, y_end+h, h)]
```

```python
def get_x(x_0:float, x_l:float, h:float) -> list:
    return [i for i in frange(x_0, x_l+h, h)]

def interpol(U_values:List[List[float]], X:List[float], Y:List[float], y_right:float)
-> None:
    for i in range(1, len(Y) - 1):
        for j in range(1, len(X) - 1):
            alpha = Y[i] / y_right
            U_values[i][j] = phi_1(X[j]) * (1 - alpha) + phi_2(X[j]) * alpha

def border_conds(U_values:List[List[float]], X:List[float], Y:List[float]) -> None:
    for i in range(len(Y)):
        U_values[i][0] = phi_2(Y[i])
        U_values[i][1] = phi_2(Y[i]) * (X[1] - X[0])
        U_values[i][-1] = phi_1(Y[i])
    for i in range(len(X)):
        U_values[0][i] = phi_3(X[i])
        U_values[-1][i] = phi_4(X[i])

def error(U_prev:List[List[float]], U_values:List[List[float]]) -> float:
    error_score = 0
    for i in range(len(U_values)):
        for j in range(len(U_values[0])):
            error_score = max(error_score, abs(U_prev[i][j] - U_values[i][j]))
    return error_score
```

Отношение конечной разности по схеме

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h_1^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h_2^2} + O(h_1^2 + h_2^2) = f(x_i, y_i)$$

мой случай

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h_1^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h_2^2} + cu_{i,j} = f(x_i, y_i)$$

Метод Лимбмана

$$u_{i,j}^{k+1} = \left(\frac{u_{i-1,j}^k + u_{i+1,j}^k}{h_1^2} + \frac{u_{i,j-1}^k + u_{i,j+1}^k}{h_2^2} + cu_{i,j}^k + f_{i,j}\right)\left(\frac{2}{h_1^2} + \frac{2}{h_2^2}\right)^{-1}$$

```python
def limbman(x_left:float, x_right:float, y_left:float, y_right:float, a:float, b:float
, c:float, d:float=0, eps:float=0.1, n_x:float=10, n_y:float=10) -> (List[float], List
[float], List[List[float]]):
    hx = x_right / (n_x - 1)
    hy = y_right / (n_y - 1)
    X = get_x(x_left, x_right, hx)
    Y = get_y(y_left, y_right, hy)

    U_values = [[0.0] * len(X) for _ in range(len(Y))]
    U_prev = [[10.0] * len(X) for _ in range(len(Y))]


    border_conds(U_values, X, Y)
    interpol(U_values, X, Y, y_right)

    cnt = 0
    while (error(U_prev, U_values) > eps):
        U_prev = copy.deepcopy(U_values)
        for i in range(1, len(Y) - 1):
            for j in range(1, len(X) - 1):
                U_values[i][j] = ((U_prev[i-1][j] +  U_prev[i+1][j]) / hx**2 + (U_prev
```

```
[i][j-1] + U_prev[i][j+1]) / hy**2  + c * U_prev[i][j] - f(X[j], Y[i])) * ((hx**2 * hy
**2) / (2 * (hx**2 + hy**2)))

        cnt +=1

    print('Iterations', cnt)
    return X, Y, U_values

X, Y, UU = limbman(x_left, x_right, y_left, y_right, a, b, c, d=0, eps=0.0005, n_x=10,
n_y=10)
U_true = real_U(X, Y)
```

Моя поверхность

```
fig = go.Figure(data=[go.Surface(z=UU, x=X, y=Y)])
fig.show()
```

Точное решение

```
fig = go.Figure(data=[go.Surface(z=U_true, x=X, y=Y)])
fig.show()
```

Две поверхности вместе

```
fig = go.Figure(data=[go.Surface(z=UU, x=X, y=Y, colorscale='Reds', name='Метод Лимбма
на'), go.Surface(z=U_true, x=X, y=Y, colorscale='Blues', name='Аналитическое решение')
])
fig.update_layout(title='Метод Лимбмана - красный, Аналитическое решение - синий', sho
wlegend=True)
fig.show()
```

Вычисление ошибки и построение графиков

```
N = [20, 30, 40, 50]

H_X1, H_Y1, ERROR_1 = get_error_array_with_h(N, x_left, x_right, y_left, y_right, a, b
, c, d=0, eps=0.0005, find_u=limbman)
h_error_plot(H_X1, ERROR_1)
h_error_plot(H_Y1, ERROR_1, s=' y')
```

Метод Зейделя

$$u_{i,j}^{k+1} = \left(\frac{u_{i-1,j}^{k+1} + u_{i+1,j}^{k}}{h_1^2} + \frac{u_{i,j-1}^{k+1} + u_{i,j+1}^{k}}{h_2^2} + cu_{i,j}^{k} + f_{i,j}\right)\left(\frac{2}{h_1^2} + \frac{2}{h_2^2}\right)^{-1}$$

```
def Seidel(x_left:float, x_right:float, y_left:float, y_right:float, a:float, b:float,
c:float, d:float=0, eps:float=0.1, n_x:float=10, n_y:float=10) -> (List[float], List[f
loat], List[List[float]]):
    hx = x_right / (n_x - 1)
    hy = y_right / (n_y - 1)
    X = get_x(x_left, x_right, hx)
    Y = get_y(y_left, y_right, hy)

    U_values = [[0.0] * len(X) for _ in range(len(Y))]
    U_prev = [[10.0] * len(X) for _ in range(len(Y))]

    border_conds(U_values, X, Y)
    interpol(U_values, X, Y, y_right)



    cnt = 0
    while (error(U_prev, U_values) > eps):
        U_prev = copy.deepcopy(U_values)
```

```
        for i in range(1, len(Y) - 1):
            for j in range(1, len(X) - 1):
                U_values[i][j] = ((U_values[i-1][j] +  U_values[i+1][j]) / hx**2 + (U_prev[i][j-1] + U_values[i][j+1]) / hy**2 + c * U_prev[i][j] - f(X[j], Y[i])) * ((hx**2 * hy**2) / (2 * (hx**2 + hy**2)))

        cnt += 1
    print('Iterations', cnt)
    return X, Y, U_values


X2, Y2, UU2 = Seidel(x_left, x_right, y_left, y_right, a, b, c, d=0, eps=0.0005, n_x=10, n_y=10)
U_true2 = real_U(X2, Y2)
```

Моя поверхность

```
fig = go.Figure(data=[go.Surface(z=UU2, x=X2, y=Y2)])
fig.show()
```

Точное решение

```
fig = go.Figure(data=[go.Surface(z=U_true2, x=X2, y=Y2)])
fig.show()
```

Две поверхности вместе

```
fig = go.Figure(data=[go.Surface(z=UU2, x=X2, y=Y2, colorscale='Reds', name='Метод Зейделя'), go.Surface(z=U_true2, x=X2, y=Y2, colorscale='Blues', name='Аналитическое решение')])
fig.update_layout(title='Метод Зейделя - красный, Аналитическое решение - синий', showlegend=True)
fig.show()
```

Вычисление ошибки и построение графиков

```
N = [20, 30, 40, 50]

H_X2, H_Y2, ERROR_2 = get_error_array_with_h(N, x_left, x_right, y_left, y_right, a, b, c, d=0, eps=0.0005, find_u=Seidel)
h_error_plot(H_X2, ERROR_2)
h_error_plot(H_Y2, ERROR_2, s=' y')
```

Метод простых итераций с верхней релаксацией

$$ u^{k+1} = \alpha u^k + \beta \\ u^{k+1} = \omega (\alpha u^k + \beta) + (1 - \omega) u^k \\ u^{k+1}_{i, j} = \omega \left(\frac{u^{k+1}_{i-1, j} + u^k_{i+1, j}}{h_1^2} + \frac{u^{k+1}_{i, j-1} + u^k_{i, j+1}}{h_2^2} + c u^k_{i, j} + f_{i, j}\right) \left(\frac{2}{h_1^2} + \frac{2}{h_2^2}\right)^{-1} + (1 - \omega) u^k_{i, j} $$

```
def relax(x_left:float, x_right:float, y_left:float, y_right:float, a:float, b:float, c:float, d:float=0, eps:float=0.1, n_x:float=10, n_y:float=10, omega:float=1.5) -> (List[float], List[float], List[List[float]]):
    if omega > 2 or omega < 1:
        print('Омега введена некорректно, она лежит в интервале (1, 2)')
    hx = x_right / (n_x - 1)
    hy = y_right / (n_y - 1)
    X = get_x(x_left, x_right, hx)
    Y = get_y(y_left, y_right, hy)


    U_values = [[0] * len(X) for _ in range(len(Y))]
    U_prev = [[10] * len(X) for _ in range(len(Y))]

    border_conds(U_values, X, Y)
```

```
    interpol(U_values, X, Y, y_right)


    cnt = 0
    while (error(U_prev, U_values) > eps):
        U_prev = copy.deepcopy(U_values)
        for i in range(1, len(Y) - 1):
            for j in range(1, len(X) - 1):
                U_values[i][j] = ((U_values[i-1][j] +  U_prev[i+1][j]) / hx**2 + (U_va
lues[i][j-1] + U_prev[i][j+1]) / hy**2 + c * U_prev[i][j] - f(X[j], Y[i])) * ((hx**2 *
hy**2) / (2 * (hx**2 + hy**2)))
                U_values[i][j] = omega * U_values[i][j] + (1 - omega) * U_prev[i][j]

        cnt += 1
    print('Iterations', cnt)
    return X, Y, U_values

X3, Y3, UU3 = relax(x_left, x_right, y_left, y_right, a, b, c, d=0, eps=0.00005, n_x=1
0, n_y=10)
U_true3 = real_U(X3, Y3)
```

Моя поверхность

```
fig = go.Figure(data=[go.Surface(z=UU3, x=X3, y=Y3)])
fig.show()
```

Точное решение

Две поверхности вместе

```
fig = go.Figure(data=[go.Surface(z=U_true3, x=X3, y=Y3)])
fig.show()

fig = go.Figure(data=[go.Surface(z=UU3, x=X3, y=Y3, colorscale='Reds', name='Метод рел
аксации'), go.Surface(z=U_true3, x=X3, y=Y3, colorscale='Blues', name='Аналитическое р
ешение')])
fig.update_layout(title='Метод релаксации - красный, Аналитическое решение - синий', s
howlegend=True)
fig.show()
```

Вычисление ошибки и построение графиков

```
N = [20, 30, 40, 50]

H_X3, H_Y3, ERROR_3 = get_error_array_with_h(N, x_left, x_right, y_left, y_right, a, b
, c, d=0, eps=0.0005, find_u=relax, omega=1.2)
h_error_plot(H_X3, ERROR_3)
h_error_plot(H_Y3, ERROR_3, s=' y')
```

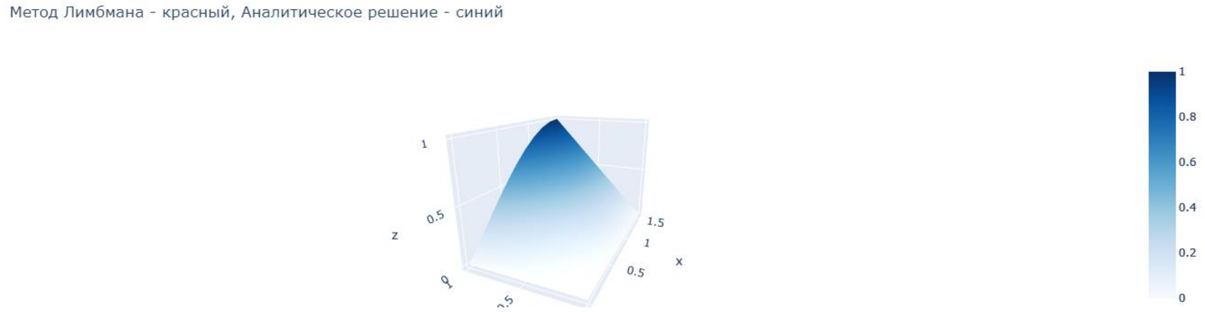
Вывод программы

Для вывода программы был взят вывод графиков, чтобы не дублировать код.

```
Метод Либмана, график численного решения
```

график аналитического решения

Две поверхности на одном графике



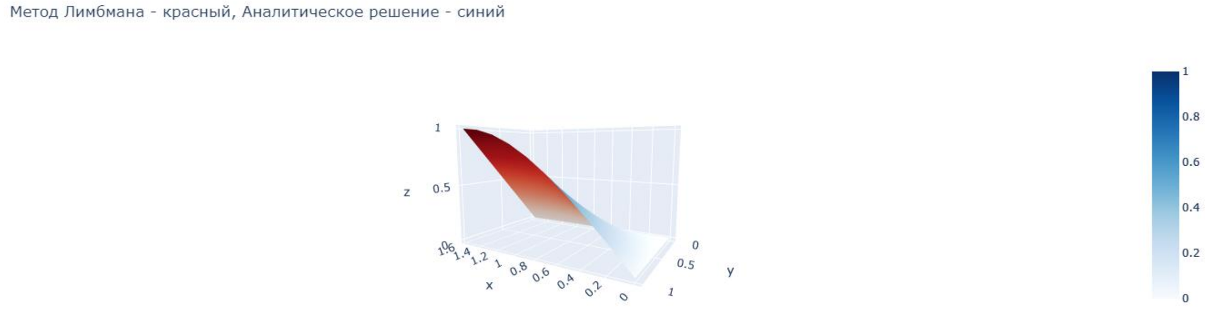
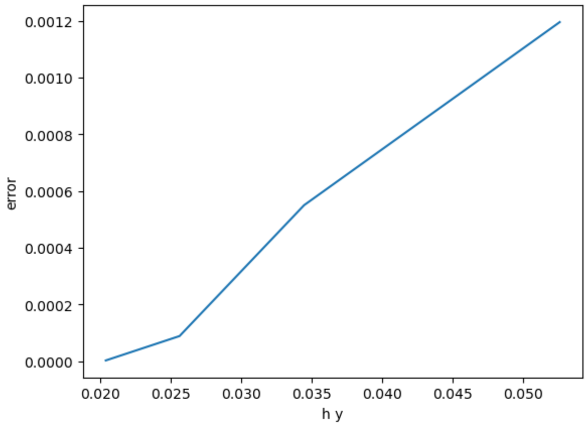
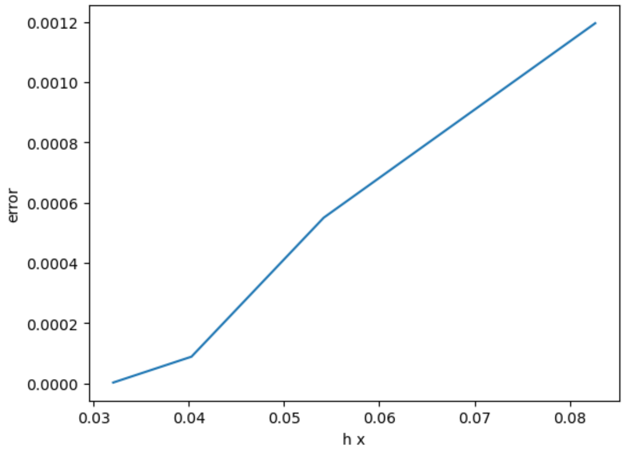


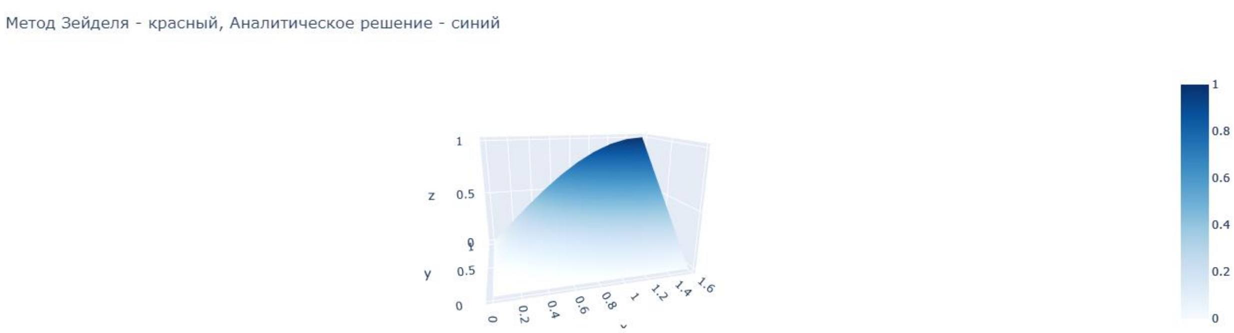
График ошибки в зависимости от шага(метод Лимбмана)





Метод Зейделя, график численного решения и график аналитического решения

Две поверхности на одном графике



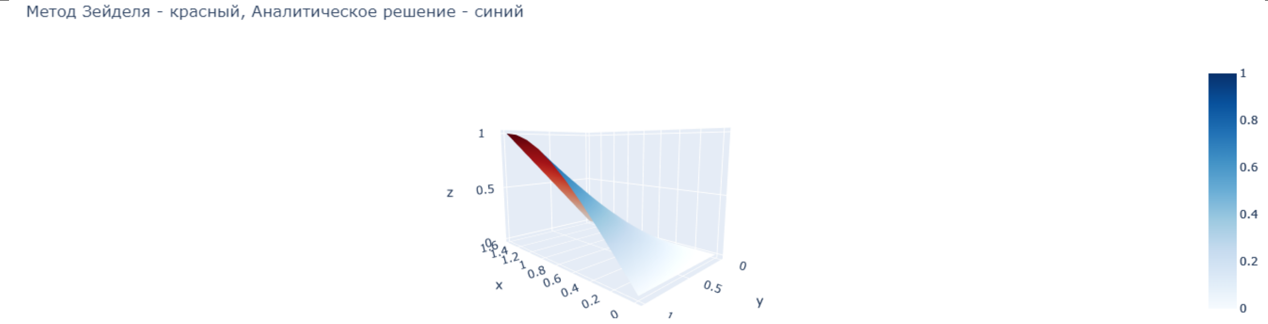
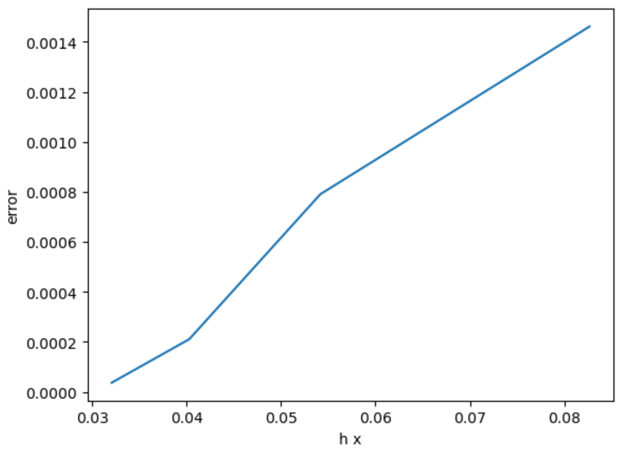
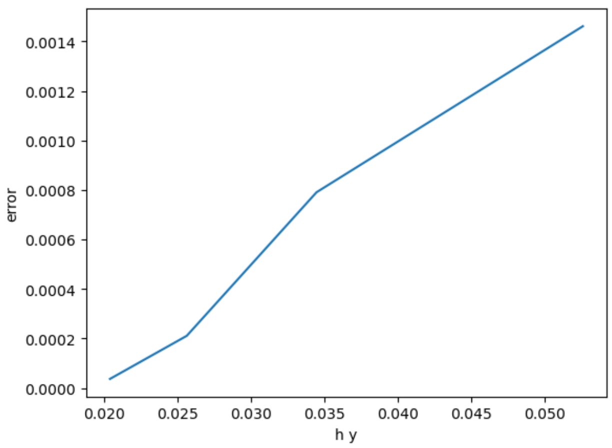


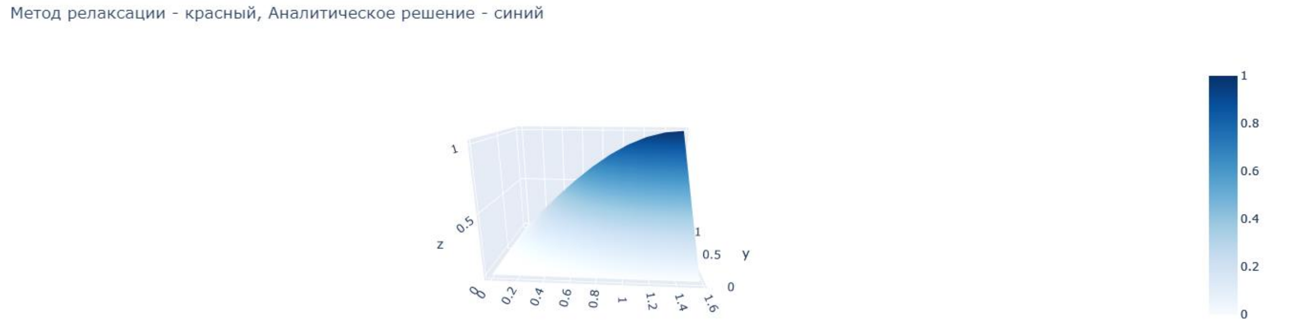
График ошибки в зависимости от шага(метод Зейделя)





Метод верхней релаксации, график численного решения и график аналитического решения

Две поверхности на одном графике



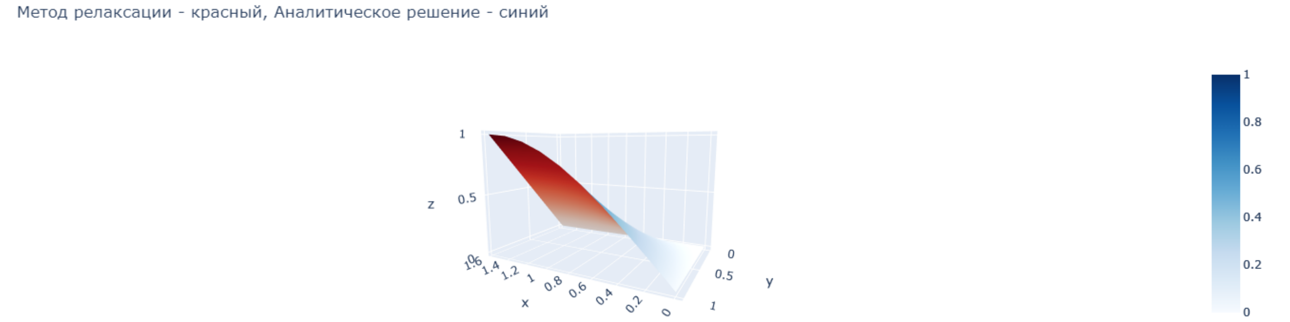
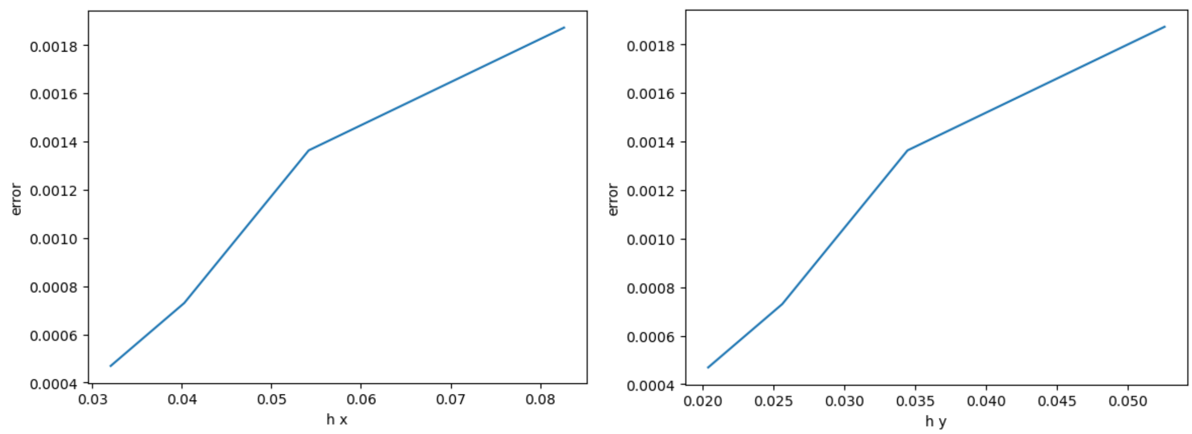


График ошибки в зависимости от шага(метод верхней релаксации)



Количество итераций и сетки

		Количество итераций		
Nx	Ny	Метод Либмана	Метод Зейделя	Метод простых итераций с верхней релаксацией
20	20	158	140	72
30	30	272	246	134
40	40	375	344	200
50	50	464	430	267

Заключение:

В данной лабораторной работе построены разностно-итерационные методы (Либмана, Зейделя, метод простых итераций с верхней релаксацией) для дифференциальных уравнений эллиптического типа с двумя пространственными переменными.
Схемы были программно реализованы на языке программирования Python в среде разработки Jupyter Notebook.
Аналитическое решение представляет собой плоскость, все методы сходятся к нему, достигая поставленной точности и завершая процесс.
Но методы отличаются количеством итераций: за наименьшее количество итераций выполняется метод простых итераций с верхней релаксацией, далее метод Зейделя, затем метод простых итераций.