**ОТЧЁТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №10**

**Уравнения эллиптического типа**

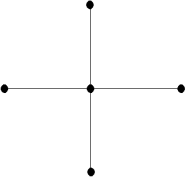
**(Вариант 9)**

Выполнил студент 3 курса МОиАИС

Сагитов Александр

**Постановка задачи:**Решить эллиптическое уравнение

методом 2-го порядка точности. Сетки по x и по y взять равномерные. Шаблон для разностной схемы:

****

Для решения разностных уравнений применить:

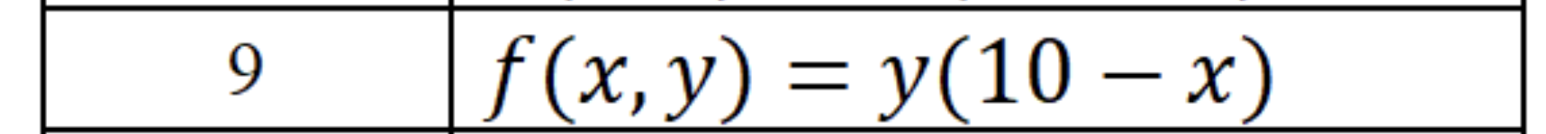
А) метод простой итерации Б) метод Зейделя

Оценивать погрешность итераций с помощью сравнения двух последовательных приближений.

Взять сетки размерами 5×5 ячеек и 10×10 ячеек и сравнить полученные решения.

Для всех вариантов [*a*, *b*] = [0; 10], [*c*, *d*] = [0; 10]. Погрешность решения 0,01.Для всех вариантов граничные условия:

𝑈(𝑥, 𝑐) = 𝑥 + 𝑐, 𝑈(𝑥, 𝑑) = 𝑥 + 𝑑, 𝑈(𝑎, 𝑦) = 𝑎 + 𝑦, 𝑈(𝑏, 𝑦) = 𝑏 + 𝑦



**Результаты расчетов**

**Метод простых итераций**

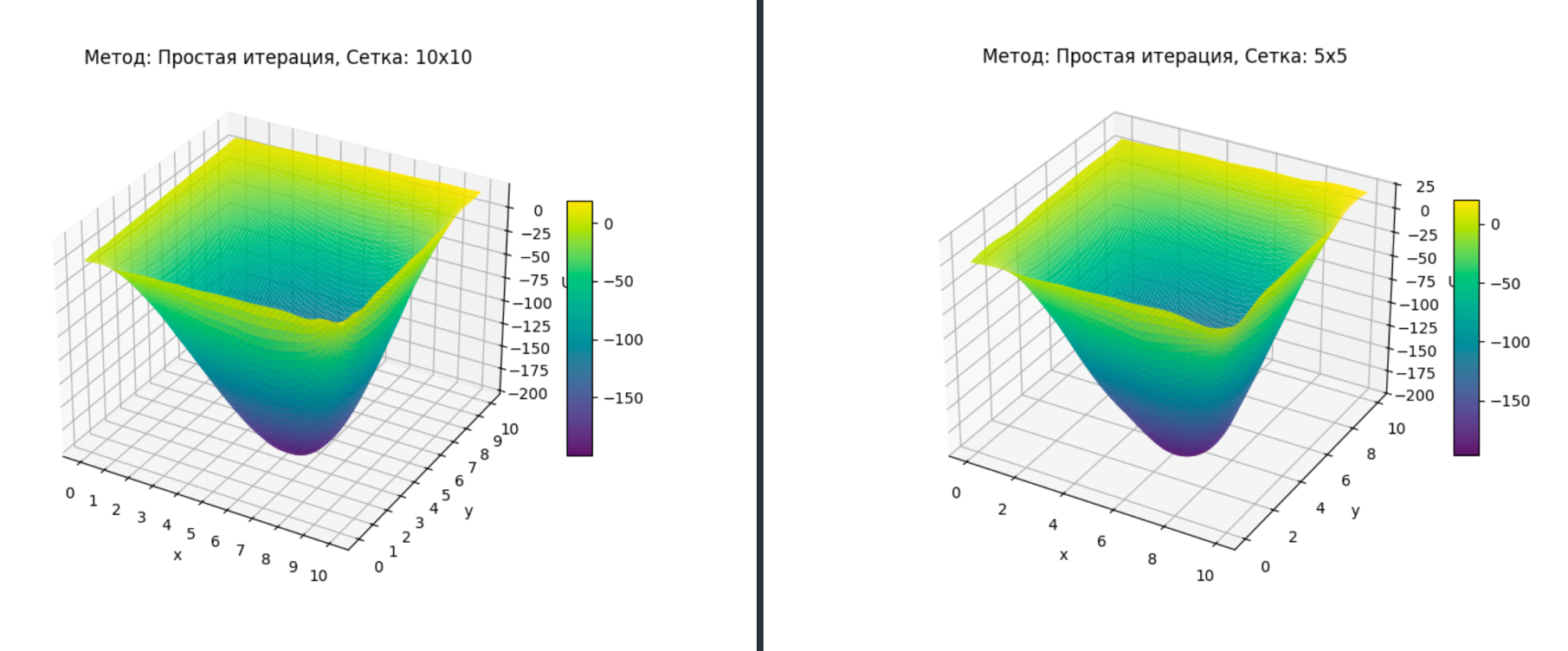
Данный метод состоит в вычислении значений узлов на итерации на основе данной итерации.

Используется формула:

.

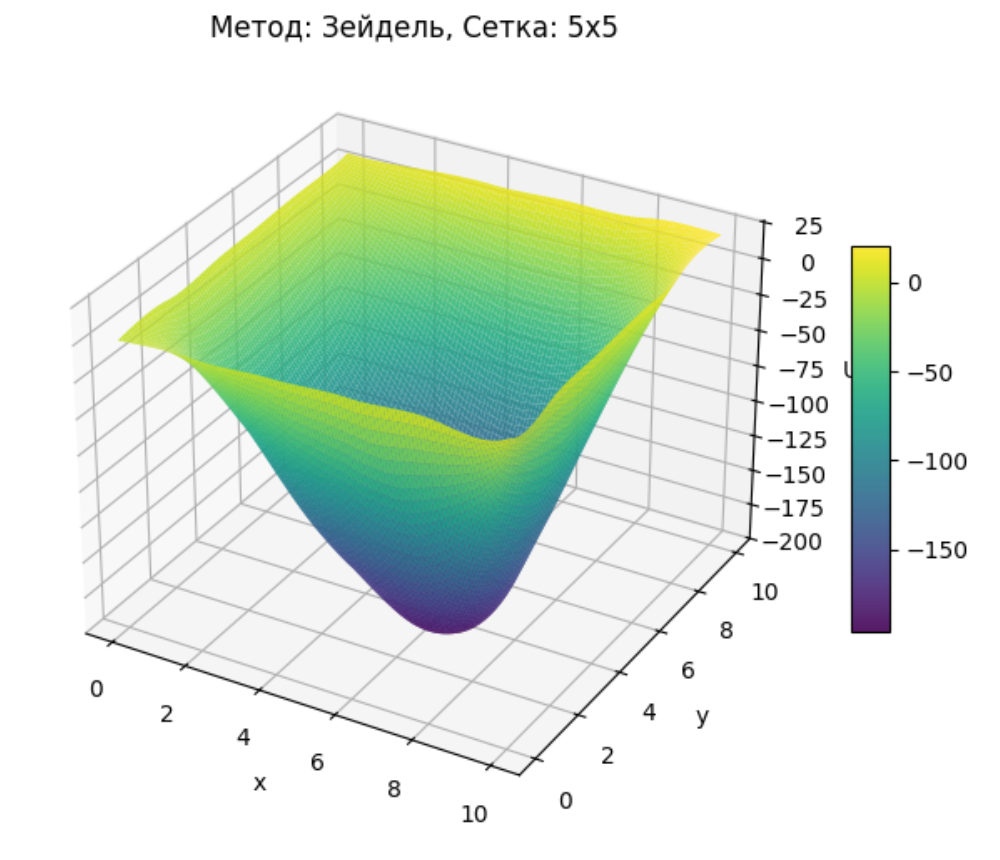
Итерационный процесс продолжается до тех пор, пока выполняется условие:

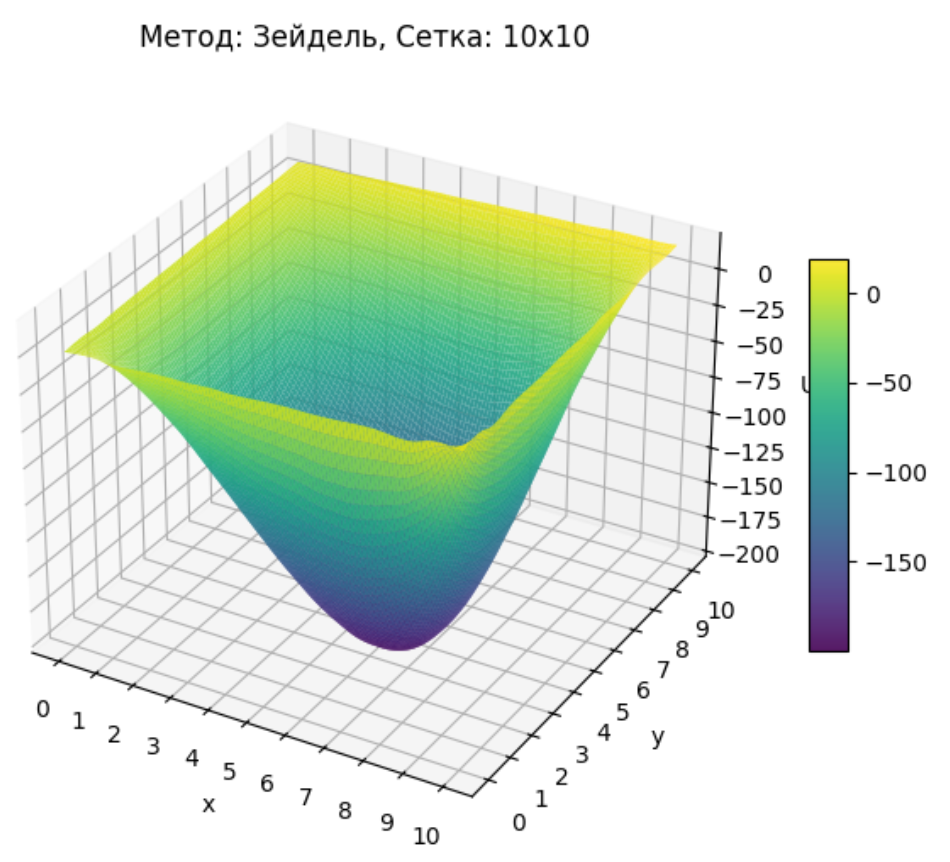
Графики:



**Метод Зейделя**

Данный метод отличается от метода простых итераций тем, что использует значения узлов, которые уже были получены на текущей итерации. Метод использует формулу:





**ПРИЛОЖЕНИЕ**

*main.py*

import numpy as np  
import matplotlib.pyplot as plt  
from scipy.interpolate import griddata  
  
  
# Граничные условия  
def Ux0(x): return x  
def Ux1(x): return x + 10  
def U0y(y): return y  
def U1y(y): return y + 10  
  
  
# Правая часть уравнения  
def f(x, y): return y \* (10 - x)  
  
  
# Общая функция для отображения результата  
def plot\_result(U, h, method\_name):  
 p = len(U) # Число узлов  
 x = np.linspace(0, 10, p) # Координаты узлов x  
 y = np.linspace(0, 10, p) # Координаты узлов y  
 x, y = np.meshgrid(x, y)  
 z = np.array(U)  
  
 fig = plt.figure(figsize=(8, 6))  
 ax = fig.add\_subplot(111, projection='3d')  
 ax.set\_title(f"Метод: {method\_name}, Сетка: {p-1}x{p-1}")  
 ax.set\_xlabel("x")  
 ax.set\_ylabel("y")  
 ax.set\_zlabel("U")  
  
 # Интерполяция для более плавной поверхности  
 grid\_x, grid\_y = np.mgrid[0:10:100j, 0:10:100j]  
 z\_interp = griddata((x.flatten(), y.flatten()), z.flatten(), (grid\_x, grid\_y), method='cubic')  
  
 # Поверхность  
 surface = ax.plot\_surface(grid\_x, grid\_y, z\_interp, cmap="viridis", rstride=1, cstride=1, edgecolor='none', alpha=0.9)  
  
 # Настраиваем деления осей  
 if h == 2.0: # Для сетки 5x5  
 ax.set\_xticks(np.linspace(0, 10, 6)) # 0, 2, 4, 6, 8, 10  
 ax.set\_yticks(np.linspace(0, 10, 6))  
 elif h == 1.0: # Для сетки 10x10  
 ax.set\_xticks(np.linspace(0, 10, 11)) # 0, 1, 2, ..., 10  
 ax.set\_yticks(np.linspace(0, 10, 11))  
  
 fig.colorbar(surface, ax=ax, shrink=0.5, aspect=10)  
 plt.show()  
  
  
# Метод простой итерации  
def simple\_iteration(h=0.1, epsilon=0.01):  
 p = int(10 / h) + 1 # Число узлов  
 U = [[0] \* p for \_ in range(p)]  
  
 # Граничные условия  
 for i in range(p):  
 x = h \* i  
 U[i][0] = Ux0(x)  
 U[i][-1] = Ux1(x)  
 for j in range(1, p):  
 y = h \* j  
 U[0][j] = U0y(y)  
 U[-1][j] = U1y(y)  
  
 # Итерации  
 while True:  
 max\_diff = 0  
 Un = [row[:] for row in U]  
 for i in range(1, p - 1):  
 for j in range(1, p - 1):  
 x, y = h \* i, h \* j  
 Un[i][j] = (U[i + 1][j] + U[i - 1][j] + U[i][j + 1] + U[i][j - 1] - h \* h \* f(x, y)) / 4  
 max\_diff = max(max\_diff, abs(Un[i][j] - U[i][j]))  
 U = Un  
 if max\_diff < epsilon:  
 break  
  
 plot\_result(U, h, "Простая итерация")  
  
  
# Метод Зейделя  
def zeidel(h=0.1, epsilon=0.01):  
 p = int(10 / h) + 1  
 U = [[0] \* p for \_ in range(p)]  
  
 # Граничные условия  
 for i in range(p):  
 x = h \* i  
 U[i][0] = Ux0(x)  
 U[i][-1] = Ux1(x)  
 for j in range(1, p):  
 y = h \* j  
 U[0][j] = U0y(y)  
 U[-1][j] = U1y(y)  
  
 # Итерации  
 while True:  
 max\_diff = 0  
 for i in range(1, p - 1):  
 for j in range(1, p - 1):  
 x, y = h \* i, h \* j  
 new\_value = (U[i + 1][j] + U[i - 1][j] + U[i][j + 1] + U[i][j - 1] - h \* h \* f(x, y)) / 4  
 max\_diff = max(max\_diff, abs(new\_value - U[i][j]))  
 U[i][j] = new\_value  
 if max\_diff < epsilon:  
 break  
  
 plot\_result(U, h, "Зейдель")  
  
  
# Вызовы функций  
simple\_iteration(h=2.0) # Сетка 5x5  
simple\_iteration(h=1.0) # Сетка 10x10  
  
zeidel(h=2.0) # Сетка 5x5  
zeidel(h=1.0) # Сетка 10x10