**ОТЧЁТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 10**

**УРАВНЕНИЯ ЭЛИПТИЧЕСКОГО ТИПА**

**(Вариант 5)**

*Выполнил студент 3 курса ПМ*

*Бут Дмитрий*

***Постановка задачи:*** усвоить методы решения ***линейного дифференциального уравнения 2-го порядка эллиптического типа***.

Численное решение дифференциального уравнения в частных производных предполагает получение двумерной числовой таблицы приближенных значений *Uij* искомой функции *U*(*x,y)* с заданной точностью для некоторых значений аргументов

*xi ∈* [*a*, *b*], *yj ∈* [*c*, *d*]

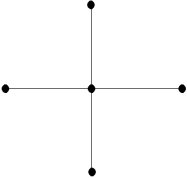
***Задание.***

Решить эллиптическое уравнение

методами 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 решения с помощью сравнения двух последовательных итераций.

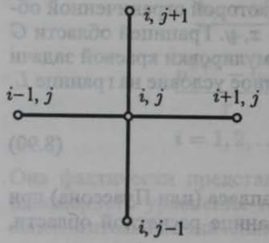
Взять сетки размерами 5×5 ячеек и 10×10 ячеек и сравнить полученные решения.

Входные данные: [*a*, *b*] = [0; 10], [*c*, *d*] = [0; 10]. Погрешность решения 0,01.

Граничные условия:

***Результаты расчетов***

Используемый шаблон:



Примем для простоты значения шагов по переменным х и y равными h.

Заменяя в исходном уравнении частные производные искомой функции с помощью отношений конечных разностей, получаем разностную схему:

С помощью данного уравнения можно записать систему линейных алгебраических уравнений относительно значений сеточной функции в узлах в виде:

Каждое из уравнений записываем в виде, разрешенном относительно значения в центральном узле:

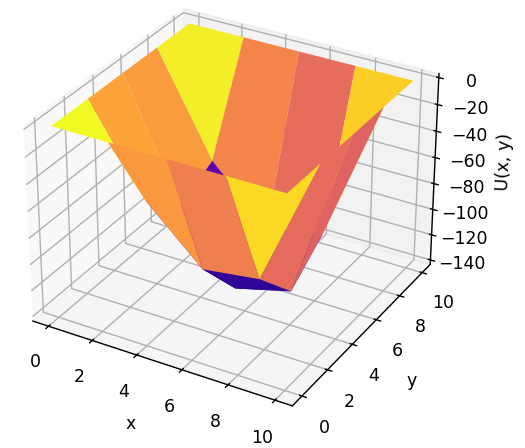
**Метод простой итерации.**

Новые значения вычисляются по формуле, содержащей предыдущее приближение. Итерации продолжаются до тех пор, пока разница между последовательными приближениями не станет меньше заданной погрешности.

Получив точки, построим трёхмерные графики:

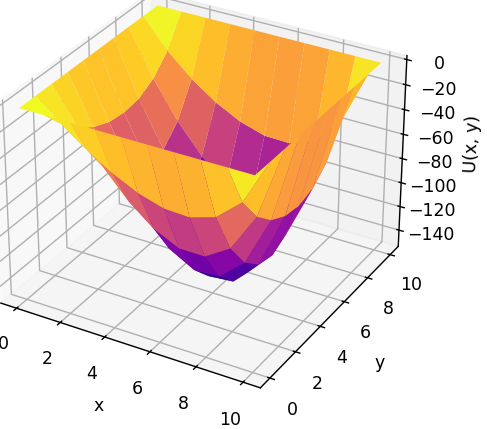
Для сетки 5×5 ячеек:

Метод потребовал 26 итераций



Для сетки 10×10 ячеек:

Метод потребовал 112 итераций



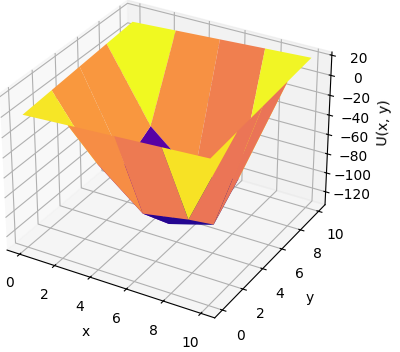
**Метод Зейделя.**

Метод Зейделя – это модификация метода простой итерации, в которой новые значения переменных используются сразу же в текущей итерации. Это ускоряет сходимость метода, так как обновлённые значения вносят коррективы на каждом шаге.

Получив точки, построим трёхмерные графики:

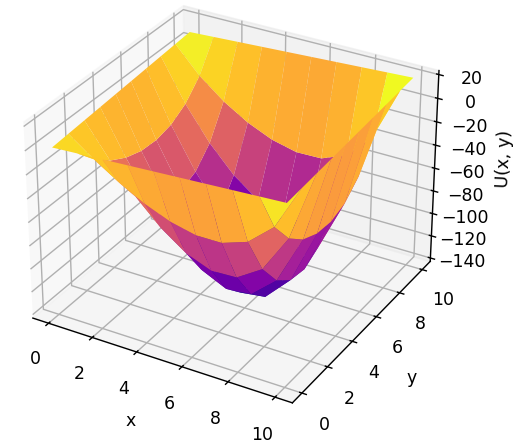
Для сетки 5×5 ячеек:

Метод потребовал 15 итераций



Для сетки 10×10 ячеек:

Метод потребовал 62 итераций



**ПРИЛОЖЕНИЕ**

***Программа реализации метода простой итерации***

import matplotlib.pyplot as plt  
import numpy as np  
  
  
def urmat(a, b, uxc, uxd, uay, uby, f, step, eps):  
 h = (b - a) / (step - 1)  
 x = np.arange(a, b + h, h)  
 y = np.arange(c, d + h, h)  
 u = np.zeros((len(x), len(y)))  
 v = np.zeros((len(x), len(y)))  
 for i in range(len(x)):  
 u[i][0] = uxc(x[i])  
 u[i][-1] = uxd(x[i])  
 for j in range(len(y)):  
 u[0][j] = uay(y[j])  
 u[-1][j] = uby(y[j])  
 M = float('inf')  
 iterations = 0  
 while M >= eps:  
 M = 0  
 for i in range(1, len(x) - 1):  
 for j in range(1, len(y) - 1):  
 v[i][j] = (u[i + 1][j] + u[i - 1][j] + u[i][j + 1] + u[i][j - 1] - h\*\*2 \* f(x[i], y[j])) / 4  
 M = max(M, np.linalg.norm(u[i][j] - v[i][j]))  
 u = v.copy()  
 iterations += 1  
 print(iterations)  
 X, Y = np.meshgrid(x, y)  
 fig = plt.figure()  
 ax = fig.add\_subplot(111, projection='3d')  
 ax.plot\_surface(X, Y, u.T, cmap='plasma')  
 ax.set\_xlabel('x')  
 ax.set\_ylabel('y')  
 ax.set\_zlabel('U(x, y)')  
 plt.show()  
  
  
a, b = 0, 10  
c, d = a, b  
uxc = lambda x: x + c  
uxd = lambda x: x + d  
uay = lambda y: a + y  
uby = lambda y: b + y  
f = lambda x, y: 3 \* x + y  
epsilon = 0.01  
urmat(a, b, uxc, uxd, uay, uby, f, 5, epsilon)  
urmat(a, b, uxc, uxd, uay, uby, f, 10, epsilon)

***Программа реализации метода Зейделя***

import matplotlib.pyplot as plt  
import numpy as np  
  
  
def urmat(a, b, uxc, uxd, uay, uby, f, step, eps):  
 h = (b - a) / (step - 1)  
 x = np.arange(a, b + h, h)  
 y = np.arange(c, d + h, h)  
 u = np.zeros((len(x), len(y)))  
 for i in range(len(x)):  
 u[i][0] = uxc(x[i])  
 u[i][-1] = uxd(x[i])  
 for j in range(len(y)):  
 u[0][j] = uay(y[j])  
 u[-1][j] = uby(y[j])  
 M = float('inf')  
 iterations = 0  
 while M >= eps:  
 M = 0  
 for i in range(1, len(x) - 1):  
 for j in range(1, len(y) - 1):  
 v = (u[i + 1][j] + u[i - 1][j] + u[i][j + 1] + u[i][j - 1] - h\*\*2 \* f(x[i], y[j])) / 4  
 M = max(M, np.linalg.norm(u[i][j] - v))  
 u[i][j] = v  
 iterations += 1  
 print(iterations)  
 X, Y = np.meshgrid(x, y)  
 fig = plt.figure()  
 ax = fig.add\_subplot(111, projection='3d')  
 ax.plot\_surface(X, Y, u.T, cmap='plasma')  
 ax.set\_xlabel('x')  
 ax.set\_ylabel('y')  
 ax.set\_zlabel('U(x, y)')  
 plt.show()  
  
  
a, b = 0, 10  
c, d = a, b  
uxc = lambda x: x + c  
uxd = lambda x: x + d  
uay = lambda y: a + y  
uby = lambda y: b + y  
f = lambda x, y: 3 \* x + y  
epsilon = 0.01  
urmat(a, b, uxc, uxd, uay, uby, f, 5, epsilon)  
urmat(a, b, uxc, uxd, uay, uby, f, 10, epsilon)