**ОТЧЁТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 10**

**Уравнения эллиптического типа**

**(Вариант 9)**

*Выполнил студент 3 курса ПМиИ*

*Кондратьев Виталий*

***Цель работы***:

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*]

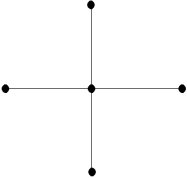
***Задание.***

Решить эллиптическое уравнение

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.

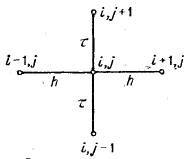
Взять сетки размерами 5×5 ячеек и 10×10 ячеек и сравнить полученные решения.

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

Для всех вариантов граничные условия

|  |  |
| --- | --- |
| № вар. | Правая часть |
| 9 |  |

Рассмотрим уравнение Лапласа:



Для решения задачи построим трехслойную разностную схему крест. Разностное уравнение, построенное по данной схеме:

Упрощая данное уравнение, получим:

Начальные условия:

Граничные условия в сеточном виде:

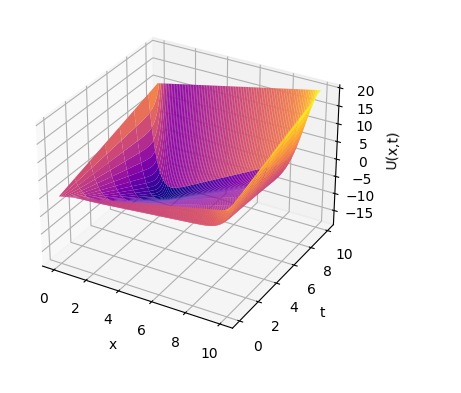
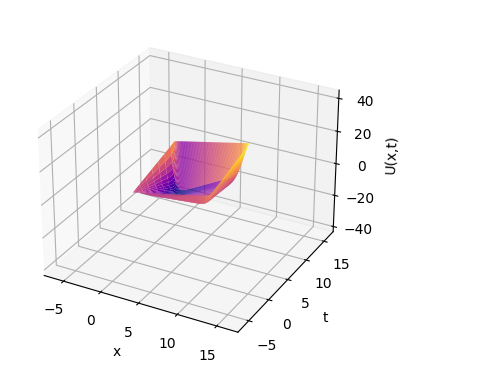
# Метод простой итерации

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

В качестве первого приближения возьмем . Итерации завершаются при выполнении условия:

**Графики:**

Сетка 5×5: Сетка 10×10:

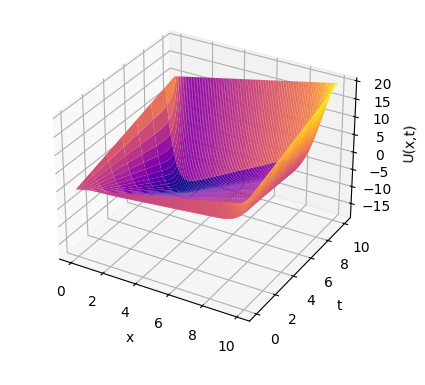
 

# Метод Зейделя

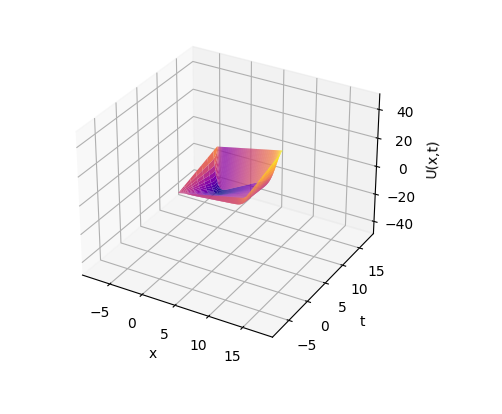
Аналогично с методом простой итерации, в качестве первого приближения возьмем . Итерации завершаются при выполнении условия:

**Графики:**

Сетка 5×5:



Сетка 10×10:



**ПРИЛОЖЕНИЕ**

*"""  
Лабораторная работа №10  
Студент ОНК «ИВТ» ВШ КНиИИ направления ПМиИ 3 курса  
Кондратьев Виталий  
Вариант 9  
"""*import numpy  
import matplotlib.pyplot as plt  
  
  
def Ux0(x):  
 return x  
  
  
def Uxl(x):  
 return x + 10  
  
  
def U0y(y):  
 return y  
  
  
def Uly(y):  
 return y + 10  
  
  
def f(x, y):  
 return y \* (10-x)  
  
  
h = 0.1  
p = int(10 / h) + 1  
U = [0] \* p  
print(p)  
  
for i in range(p):  
 U[i] = [0] \* p  
  
for i in range(0, p):  
 x = h \* i  
 U[i][0] = Ux0(x)  
 U[i][p - 1] = Uxl(x)  
  
for j in range(1, p):  
 y = h \* j  
 U[0][j] = U0y(y)  
 U[p - 1][j] = Uly(y)  
  
Un = [0] \* p  
for i in range(p):  
 Un[i] = [0] \* p  
  
  
while True:  
 for i in range(1, p - 1):  
 M = 0  
 for j in range(1, p - 1):  
 x = h \* i  
 y = 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  
 d = abs(Un[i][j] - U[i][j])  
 if (M < d):  
 M = d  
 for i in range(1, p - 1):  
 for j in range(1, p - 1):  
 U[i][j] = Un[i][j]  
 if (M < 0.01):  
 break  
  
u, v = numpy.mgrid[0:p, 0:p]  
x = h \* u  
y = h \* v  
z = x - x  
for j in range(0, p):  
 for i in range(0, p):  
 z[i][j] = U[i][j]  
  
fig = plt.figure(figsize=plt.figaspect(0.50))  
axes = fig.add\_subplot(1, 2, 1, projection='3d')  
axes.set\_xlabel("x")  
axes.set\_ylabel("t")  
axes.set\_zlabel("U(x,t)")  
suf = axes.plot\_surface(x, y, z, rstride=1, cstride=15, cmap='plasma', edgecolor='none')  
plt.show()  
  
  
for i in range(p):  
 U[i] = [0] \* p  
  
for i in range(0, p):  
 x = h \* i  
 U[i][0] = Ux0(x)  
 U[i][p - 1] = Uxl(x)  
  
for j in range(1, p):  
 y = h \* j  
 U[0][j] = U0y(y)  
 U[p - 1][j] = Uly(y)  
  
Un = [0] \* p  
for i in range(p):  
 Un[i] = [0] \* p  
  
  
while True:  
 for i in range(1, p - 1):  
 M = 0  
 for j in range(1, p - 1):  
 x = h \* i  
 y = 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  
 d = abs(Un[i][j] - U[i][j])  
 if (M < d):  
 M = d  
 for i in range(1, p - 1):  
 for j in range(1, p - 1):  
 U[i][j] = Un[i][j]  
 if (M < 0.01):  
 break  
  
u, v = numpy.mgrid[0:p, 0:p]  
x = h \* u  
y = h \* v  
z = x - x  
for j in range(0, p):  
 for i in range(0, p):  
 z[i][j] = U[i][j]  
  
fig = plt.figure(figsize=plt.figaspect(0.50))  
axes = fig.add\_subplot(1, 2, 1, projection='3d')  
axes.set\_xlabel("x")  
axes.set\_ylabel("t")  
axes.set\_zlabel("U(x,t)")  
suf = axes.plot\_surface(x, y, z, rstride=1, cstride=15, cmap='plasma', edgecolor='none')  
plt.show()