**ОТЧЁТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 8**

**Уравнения гиперболического типа**

**(Вариант 9)**

*Выполнил студент 3 курса ПМиИ*

*Кондратьев Виталий*

***Цель работы***: усвоить сущность и методы решения ***линейного дифференциального уравнения 2-го порядка гиперболического типа***.

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

*xj ∈* [*a*, *b*], *ti ∈* [*c*, *d*]

Численное решение таких дифференциальных уравнений возможно методами конечных разностей.

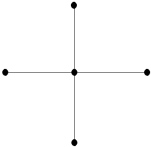
Погрешность решения, найденного этими методами, оценивается величиной O(*τp,hq*)*,* где *p*, *q* - порядок метода.

***Задание.***

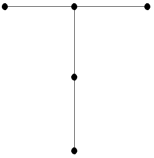
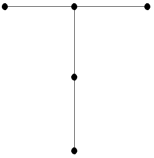
Решить волновое уравнение

явным методом и неявными методами второго порядка точности

Шаблон для явного метода:



Шаблон для неявного метода:



Вывести результаты в виде графиков U(x) для разных значений t от 0 до 10 c шагом 1

Неявные методы решать с помощью прогонки.

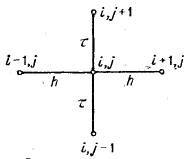
***Варианты задания (лабораторная № 8)***

Для всех вариантов [*a*, *b*] = [0; 1], [*c*, *d*] = [0; 10], *f(x,t)*=0

Погрешность решения 0,01.

|  |  |  |  |
| --- | --- | --- | --- |
| № вар. | Начальные условия | Граничные условия | D |
| 9 |  |  | 1 |

1. **ЯВНЫЙ МЕТОД**



*- схема «крест»*

Разностное уравнение, построенное по данной схеме:

Разрешая уравнение относительно :

На нулевом слое имеем:

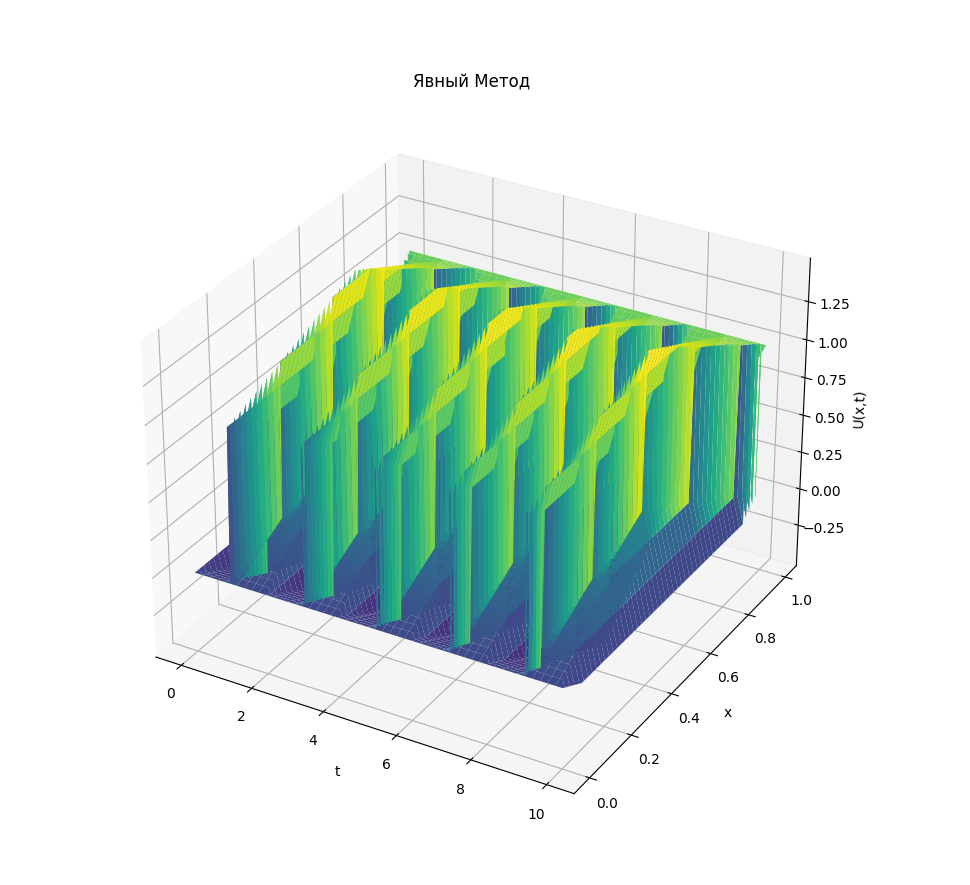
На первом слое имеем:

Граничные условия в сеточном виде:

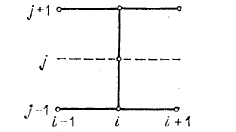
Рассмотренная разностная схема условно устойчива. Необходимое и достаточное условие устойчивости:

Возьмем в качестве первого шага разбиения: и . Итерации завершаются при выполнении условия:

**Графики:**



1. **НЕЯВНЫЙ МЕТОД**



- неявная схема

Разностное уравнение, построенное по данной схеме:

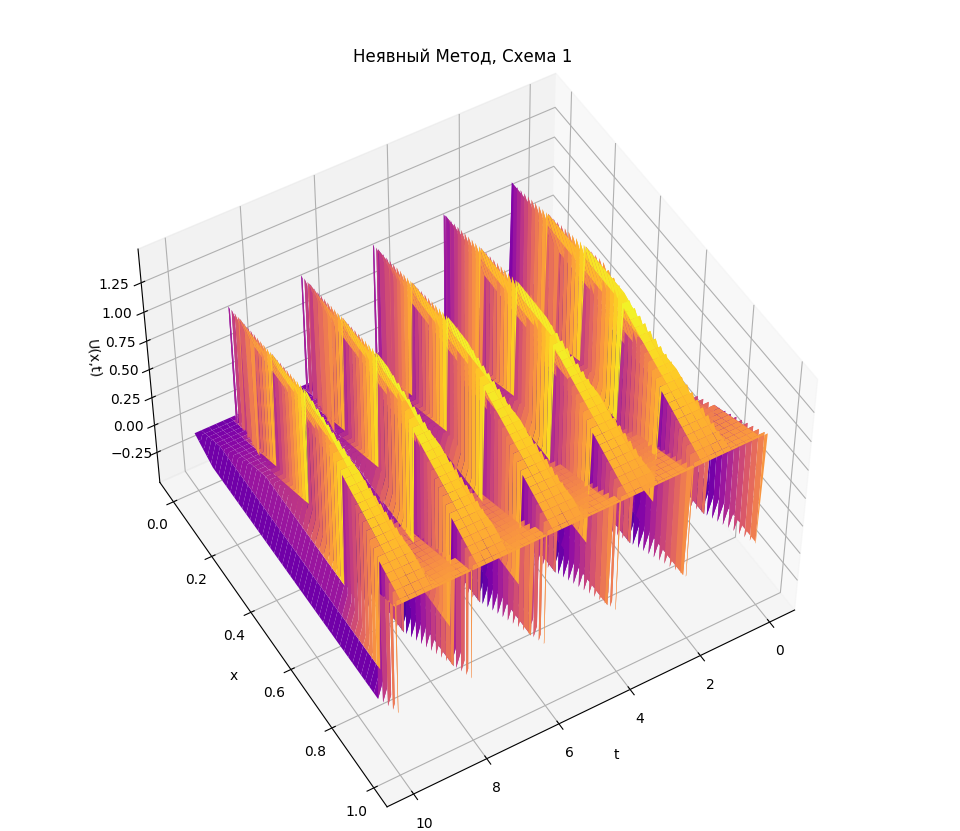
Из этого разностного соотношения можно получить систему уравнений относительно неизвестных значений сеточной функции на слое:

Полученная схема устойчива и сходится со скоростью . Граничные и начальные условия аналогичны явной схеме.

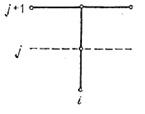
Возьмем в качестве первого шага разбиения: и . Итерации завершаются при выполнении условия:

Систему линейных уравнений решаем методом прогонки.

**Графики:**

****

1. **НЕЯВНЫЙ МЕТОД**



* неявная разностная схема

Разностное уравнение, построенное по данной схеме:

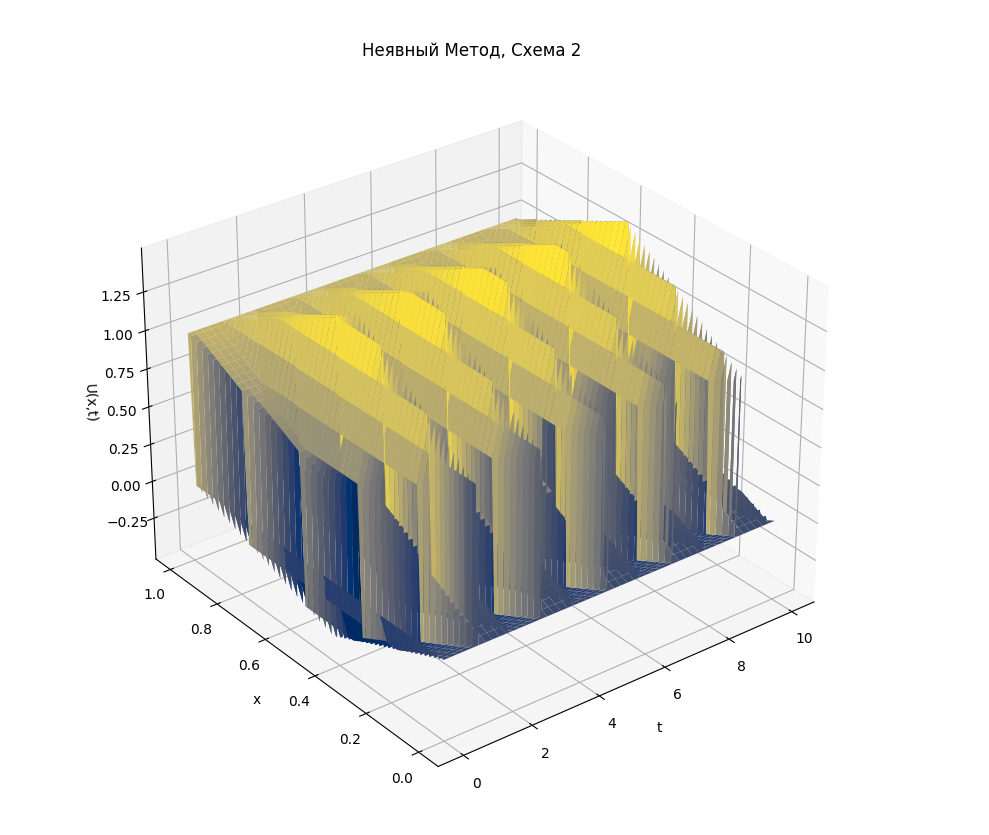
Из этого разностного соотношения можно получить систему уравнений относительно неизвестных значений сеточной функции на слое:

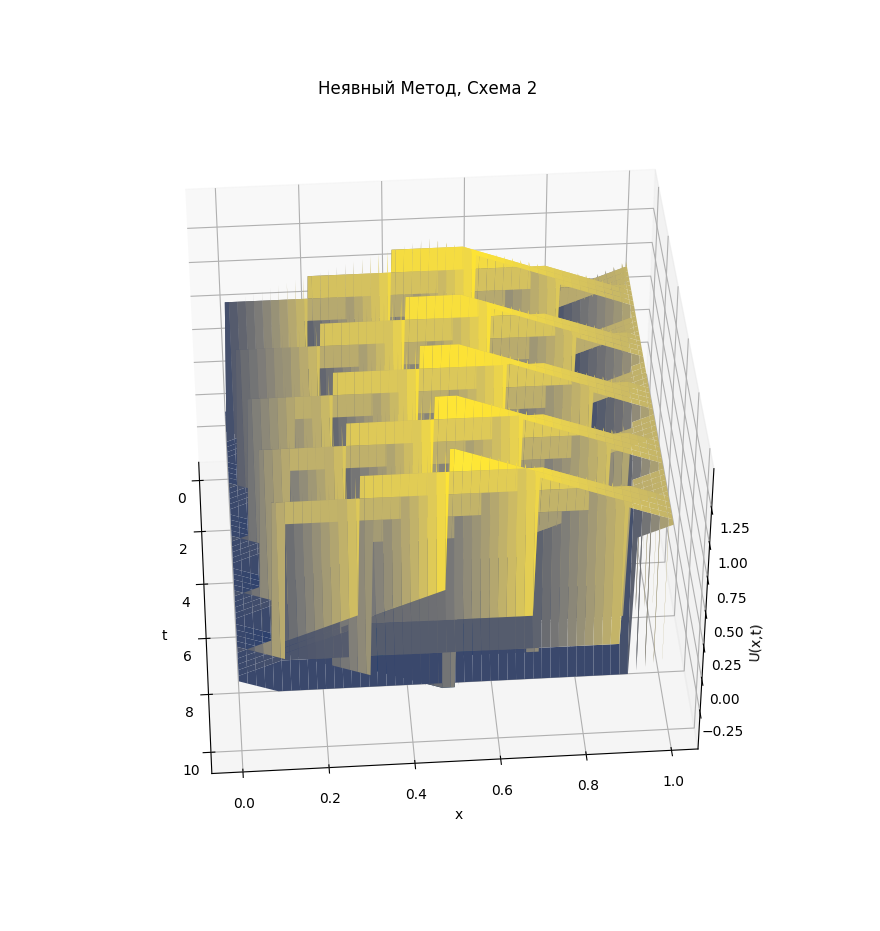
Полученная схема устойчива и сходится со скоростью . Граничные и начальные условия аналогичны явной схеме.

Возьмем в качестве первого шага разбиения: и . Итерации завершаются при выполнении условия:

Систему линейных уравнений решаем методом прогонки.

**Графики:**





**ПРИЛОЖЕНИЕ**

*"""  
Лабораторная работа №8  
Студент ОНК «ИВТ» ВШ КНиИИ направления ПМиИ 3 курса  
Кондратьев Виталий  
Вариант 9  
"""*import numpy as np  
from matplotlib import cm  
import matplotlib.pyplot as plt  
a,b = [0,1]  
c,d = [0,10]  
eps, h, tau = 0.01, 0.01, 0.01  
D = 1  
l\_ambda = (D \* tau / h) \*\* 2  
f = lambda x: 0  
U\_0 = lambda x,t: x \*\* 2  
der\_U\_t = lambda x: -2  
U\_t\_0 = lambda x,t: 0  
U\_t\_1 = lambda x,t: 1  
def Scheme1(a, b, c, d, tau, l\_ambda):  
 x\_count = int((b - a) / tau)  
 t\_count = int((d - c) / tau)  
 u = np.zeros((t\_count, x\_count))  
 for i in range(0, x\_count):  
 u[0][i] = U\_0(0, i \* h)  
 u[1][i] = u[0][i] + (-1) \* tau  
 for i in range(0, t\_count):  
 u[i][0] = U\_t\_0(i \* tau, 0)  
 u[i][x\_count - 1] = U\_t\_1(i \* tau, 1)  
 for j in range(1, t\_count - 1):  
 for i in range(1, x\_count - 1):  
 u[j + 1][i] = 2 \* (1 - l\_ambda) \* u[j][i] + l\_ambda \* (u[j][i + 1] + u[j][i - 1]) - u[j - 1][i]  
 return u  
  
def Scheme2(a, b, c, d, tau, l\_ambda):  
 x\_count = int((b - a) / tau)  
 t\_count = int((d - c) / tau)  
 A = l\_ambda  
 B = l\_ambda \* 2 + 1  
 u = np.zeros((t\_count, x\_count))  
 a = b = c = np.zeros(x\_count)  
 for i in range(0, x\_count):  
 u[0][i] = U\_0(0, i \* h)  
 u[1][i] = u[0][i] + (-1) \* tau  
 for i in range(0, t\_count):  
 u[i][0] = U\_t\_0(i \* tau, 0)  
 u[i][x\_count - 1] = U\_t\_1(i \* tau, 1)  
 for j in range(1, t\_count - 1):  
 a[x\_count - 1] = 0  
 b[x\_count - 1] = u[j + 1][x\_count - 1]  
 c[x\_count - 1] = 1 / (B - A \* a[x\_count - 1])  
 for i in range(x\_count - 1, 0, -1):  
 a[i - 1] = c[i] \* A  
 b[i - 1] = c[i] \* (A \* b[i] - (u[j - 1][i] - 2 \* u[j][i]))  
 c[i - 1] = 1 / (B - A \* a[i - 1])  
 for i in range(1, x\_count - 1):  
 u[j + 1][i + 1] = a[i] \* u[j + 1][i] + b[i]  
 return u  
  
def Scheme3(a, b, c, d, tau, l\_ambda):  
 x\_count = int((b - a) / tau)  
 t\_count = int((d - c) / tau)  
 A = l\_ambda  
 B = l\_ambda \* 2 + 1  
 u = np.zeros((t\_count, x\_count))  
 a = b = c = np.zeros(x\_count)  
 for i in range(0, x\_count):  
 u[0][i] = U\_0(0, i \* h)  
 u[1][i] = u[0][i] + (-1) \* tau  
 for i in range(0, t\_count):  
 u[i][0] = U\_t\_0(i \* tau, 0)  
 u[i][x\_count - 1] = U\_t\_1(i \* tau, 1)  
 for j in range(1, t\_count - 1):  
 a[x\_count - 1] = 0  
 b[x\_count - 1] = u[j + 1][x\_count - 1]  
 c[x\_count - 1] = 1 / (B - A \* a[x\_count - 1])  
 a[x\_count - 2] = c[x\_count - 1] \* A  
 b[x\_count - 2] = (c[x\_count - 1] \* (A \* b[x\_count - 1]  
 - (B \* u[j - 1][x\_count - 1] - A \* (u[j - 1]  
 [x\_count - 1]  
 + u[j - 1][x\_count - 2]) - 2 \* u[j][x\_count - 1])))  
 c[x\_count - 2] = 1 / (B - A \* a[x\_count - 2])  
 for i in range(x\_count - 2, 0, -1):  
 a[i - 1] = c[i] \* A  
 b[i - 1] = (c[i] \* (A \* b[i] - (B \* u[j - 1][i]  
 - A \* (u[j - 1][i + 1] + u[j - 1][i - 1]) - 2 \* u[j][i])))  
 c[i - 1] = 1 / (B - A \* a[i - 1])  
 for i in range(1, x\_count - 1):  
 u[j + 1][i + 1] = a[i] \* u[j + 1][i] + b[i]  
 return u  
  
def draw(a, b, c, d, h, tau, u, method):  
 x = np.arange(a, b, h)  
 t = np.arange(c, d, tau)  
 x, t = np.meshgrid(x, t)  
 fig, ax = plt.subplots(subplot\_kw={"projection": "3d"})  
 ax.set\_xlabel("t")  
 ax.set\_ylabel("x")  
 ax.set\_zlabel("U(x,t)")  
 ax.plot\_surface(t, x, u, cmap=cm.viridis)  
 ax.set\_title(method.title())  
 plt.show()  
  
def draw\_1(a, b, c, d, h, tau, u, method):  
 x = np.arange(a, b, h)  
 t = np.arange(c, d, tau)  
 x, t = np.meshgrid(x, t)  
 fig, ax = plt.subplots(subplot\_kw={"projection": "3d"})  
 ax.set\_xlabel("t")  
 ax.set\_ylabel("x")  
 ax.set\_zlabel("U(x,t)")  
 ax.plot\_surface(t, x, u, cmap=cm.plasma)  
 ax.set\_title(method.title())  
 plt.show()  
  
def draw\_2(a, b, c, d, h, tau, u, method):  
 x = np.arange(a, b, h)  
 t = np.arange(c, d, tau)  
 x, t = np.meshgrid(x, t)  
 fig, ax = plt.subplots(subplot\_kw={"projection": "3d"})  
 ax.set\_xlabel("t")  
 ax.set\_ylabel("x")  
 ax.set\_zlabel("U(x,t)")  
 ax.plot\_surface(t, x, u, cmap=cm.cividis)  
 ax.set\_title(method.title())  
 plt.show()  
  
u1 = Scheme1(a, b, c, d, tau, l\_ambda)  
u2 = Scheme2(a, b, c, d, tau, l\_ambda)  
u3 = Scheme3(a, b, c, d, tau, l\_ambda)  
draw(a, b, c, d, h, tau, u1, "Явный метод")  
draw\_1(a, b, c, d, h, tau, u1, "Неявный метод, схема 1")  
draw\_2(a, b, c, d, h, tau, u1, "Неявный метод, схема 2")