**ОТЧЁТ**

**ЛАБОРАТОРНАЯ РАБОТА №8**

**Уравнения гиперболического типа**

**(Вариант 9)**

*Выполнил студент 3 курса МОиАИС*

*Сагитов Александр*

**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.**

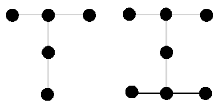
Решить волновое уравнение

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 1 до 10 c шагом 1

Неявные методы решать с помощью прогонки.

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

Погрешность решения 0,01

**Результаты расчетов**

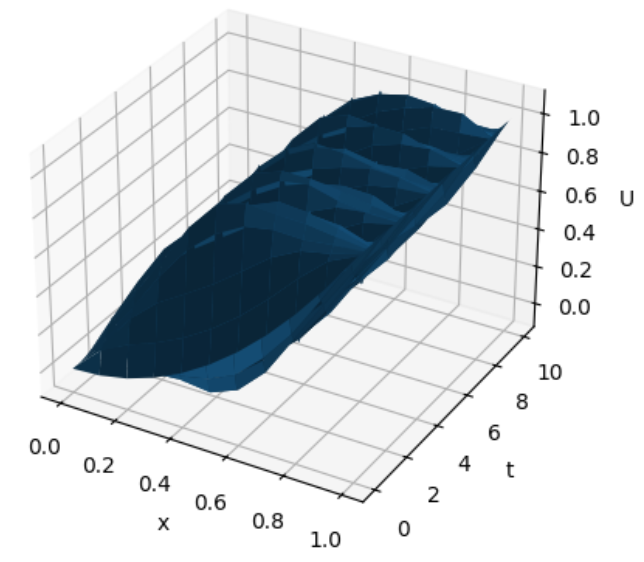
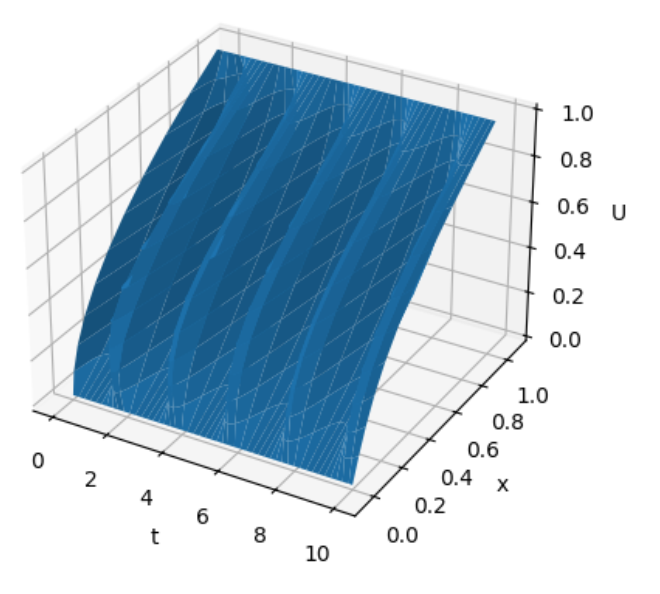
**Явная схема**

Вычисляем:

Есть условие сходимости ,

но

Поэтому будет использоваться

**Неявная схема №1**

Из этого соотношения, мы получим систему уравнений относительно j+1 слоя

Где

Аналогично берем h=0.1

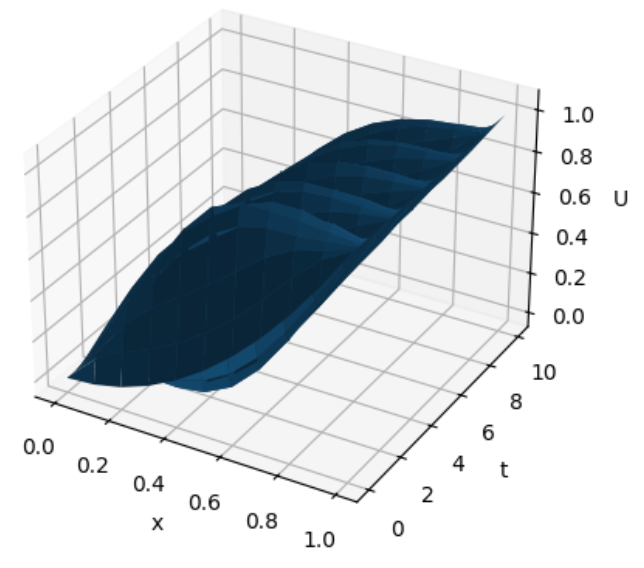
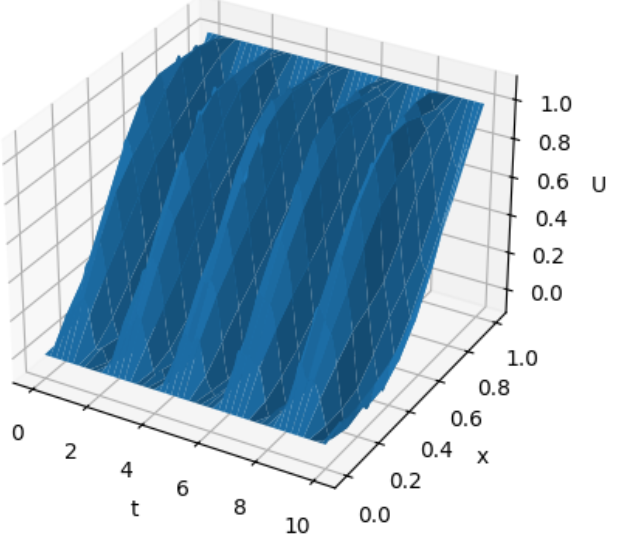
Используем метод прогонки. Метод прогонки основывается на предположении, что искомые неизвестные связаны рекуррентным соотношением:

Используя это соотношение, выразим и через и подставим в уравнение

, получим:

Это соотношение будет выполняться независимо от решения, если потребовать:

;

****

**Неявная схема №2**

Из этого соотношения мы получим систему уравнений относительно j+1 слоя

Где

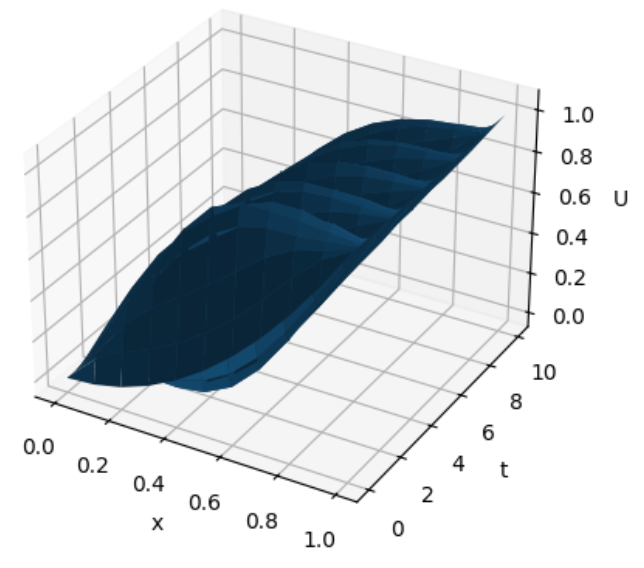
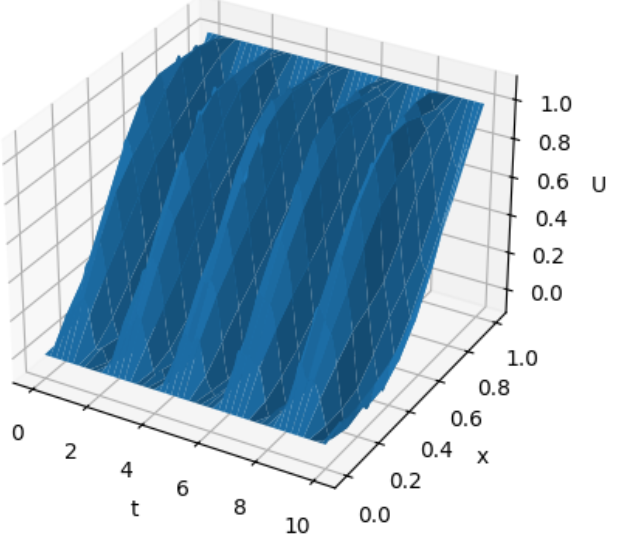
Аналогично берем h=0.1

Используем метод прогонки. Метод прогонки основывается на предположении, что искомые неизвестные связаны рекуррентным соотношением:

Используя это соотношение, выразим и через и подставим в уравнение

, получим:

Это соотношение будет выполняться независимо от решения, если потребовать: ;

****

**Приложение**

*main.py*

import numpy  
import matplotlib.pyplot as plt  
  
  
def Ux0(x\_):  
 return x\_ \*\* 2  
  
  
def Ux1():  
 return 1  
  
  
def U0t():  
 return 0  
  
  
def U1t():  
 return 1  
  
  
#======== ЯВНЫЙ МЕТОД =========#   
  
h = 0.1  
tau = 0.01  
p = int(1 / h) + 1  
q = int(10 / tau) + 1  
l = pow(tau / h, 2)  
  
U = [0] \* p  
for i in range(p):  
 U[i] = [0] \* q  
  
for i in range(0, p):  
 x = h \* i  
 U[i][0] = Ux0(x)  
 U[i][1] = U[i][0] + tau \* Ux1()  
  
for j in range(1, q):  
 t = tau \* j  
 U[0][j] = U0t()  
 U[p - 1][j] = U1t()  
  
for j in range(1, q - 1):  
 for i in range(1, p - 1):  
 x = h \* i  
 t = tau \* j  
 U[i][j + 1] = 2 \* (1 - l) \* U[i][j] + l \* (U[i + 1][j] + U[i - 1][j]) - U[i][j - 1]  
  
u, v = numpy.mgrid[0:p, 0:q]  
x = h \* u  
y = tau \* v  
z = x - x  
  
for i in range(0, p):  
 for j in range(0, q):  
 z[i][j] = U[i][j]  
  
fig = plt.figure()  
ax = fig.add\_subplot(projection='3d')  
ax.plot\_surface(y, x, z)  
ax.set\_xlabel('t')  
ax.set\_ylabel('x')  
ax.set\_zlabel('U')  
plt.show()  
  
#======== НЕЯВНЫЙ МЕТОД №1 =========#  
  
U = [0] \* p  
for i in range(p):  
 U[i] = [0] \* q  
  
for i in range(0, p):  
 x = h \* i  
 U[i][0] = Ux0(x)  
 U[i][1] = U[i][0] + tau \* Ux1()  
  
for j in range(1, q):  
 t = tau \* j  
 U[0][j] = U0t()  
 U[p - 1][j] = U1t()  
  
for j in range(1, q - 1):  
 mb = [0] \* p  
 for i in range(1, p - 2):  
 mb[i] = -l  
  
 mc = [0] \* p  
 for i in range(1, p - 1):  
 mc[i] = 1 + 2 \* l  
  
 ma = [0] \* p  
 for i in range(2, p - 1):  
 ma[i] = -l  
  
 mf = [0] \* p  
 mf[1] = 2 \* U[1][j] - U[1][j - 1] + l \* U[0][j]  
 for i in range(2, p - 2):  
 mf[i] = 2 \* U[i][j] - U[i][j - 1]  
 mf[p - 2] = 2 \* U[p - 2][j] - U[p - 2][j - 1] + l \* U[p - 1][j]  
  
 for i in range(2, p - 1):  
 m = ma[i] / mc[i - 1]  
 mc[i] -= m \* mb[i - 1]  
 mf[i] -= m \* mf[i - 1]  
  
 U[p - 2][j + 1] = mf[p - 2] / mc[p - 2]  
  
 for i in range(p -3, 0, -1):  
 U[i][j + 1] = (mf[i] - mb[i] \* U[i + 1][j + 1]) / mc[i]  
  
u, v = numpy.mgrid[0:p, 0:q]  
x = h \* u  
y = tau \* v  
z = x - x  
  
for i in range(0, p):  
 for j in range(0, q):  
 z[i][j] = U[i][j]  
  
fig = plt.figure()  
ax = fig.add\_subplot(projection='3d')  
ax.plot\_surface(y, x, z)  
ax.set\_xlabel('t')  
ax.set\_ylabel('x')  
ax.set\_zlabel('U')  
plt.show()  
  
#======== НЕЯВНЫЙ МЕТОД №2 =========#  
  
U = [0] \* p  
for i in range(p):  
 U[i] = [0] \* q  
  
for i in range(0, p):  
 x = h \* i  
 U[i][0] = Ux0(x)  
 U[i][1] = U[i][0] + tau \* Ux1()  
  
for j in range(1, q):  
 t = tau \* j  
 U[0][j] = U0t()  
 U[p - 1][j] = U1t()  
  
for j in range(1, q - 1):  
 mb = [0] \* p  
 for i in range(1, p - 2):  
 mb[i] = -l  
  
 mc = [0] \* p  
 for i in range(1, p - 1):  
 mc[i] = 1 + 2 \* l  
  
 ma = [0] \* p  
 for i in range(2, p - 1):  
 ma[i] = -l  
  
 mf = [0] \* p  
 mf[1] = 2 \* U[1][j] - U[1][j - 1] + l \* U[0][j]  
 for i in range(2, p - 2):  
 mf[i] = 2 \* U[i][j] - U[i][j - 1]  
 mf[p - 2] = 2 \* U[p - 2][j] - U[p - 2][j - 1] + l \* U[p - 1][j]  
  
 for i in range(2, p - 1):  
 m = ma[i] / mc[i - 1]  
 mc[i] -= m \* mb[i - 1]  
 mf[i] -= m \* mf[i - 1]  
  
 U[p - 2][j + 1] = mf[p - 2] / mc[p - 2]  
  
u, v = numpy.mgrid[0:p, 0:q]  
x = h \* u  
y = tau \* v  
z = x - x  
  
for i in range(0, p):  
 for j in range(0, q):  
 z[i][j] = U[i][j]  
  
fig = plt.figure()  
ax = fig.add\_subplot(projection='3d')  
ax.plot\_surface(x, y, z)  
ax.set\_xlabel('x')  
ax.set\_ylabel('t')  
ax.set\_zlabel('U')  
plt.show()