**ОТЧЁТ**

**ЛАБОРАТОРНАЯ РАБОТА №7**

**Квазилинейное уравнение переноса**

**(Вариант 9)**

*Выполнил студент 3 курса МОиАИС*

*Сагитов Александр*

**Цель работы:** усвоить сущность и методы решения квазилинейного ДУ 1-го порядка в частных производных с разрывными начальными условиями.

Численное решение ДУ 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 ДУ возможно методами конечных разностей.

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

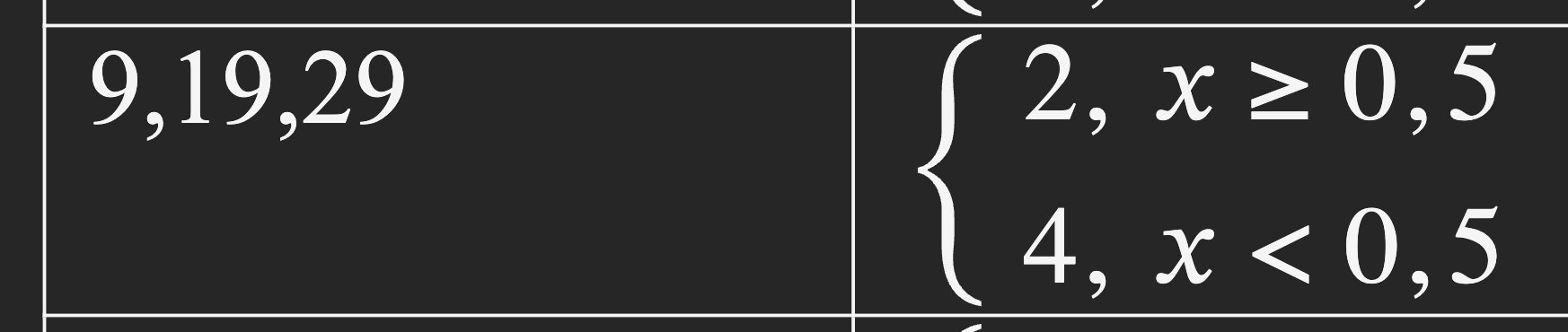
**Задание*.***

Решить уравнение переноса в прямоугольнике [*a*, *b*] = [0; 1], [*c*, *d*] = [0; 1].

(1)

методом с искусственной вязкостью и консервативной схемы.

Погрешность решения 0,01 (определяется сходимостью схемы и величиной шагов).



**Результаты расчетов**

1. *Метод искусственной вязкости*

Вместо исходного квазилинейного уравнения рассмотрим уравнение

Последний член этого уравнения - искусственная вязкость, при этом параметр мал.

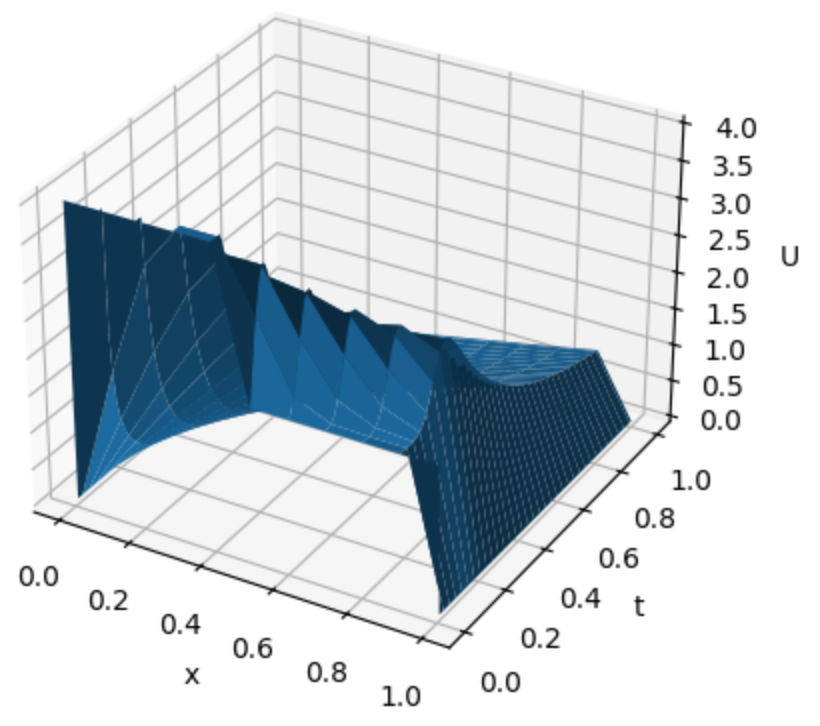
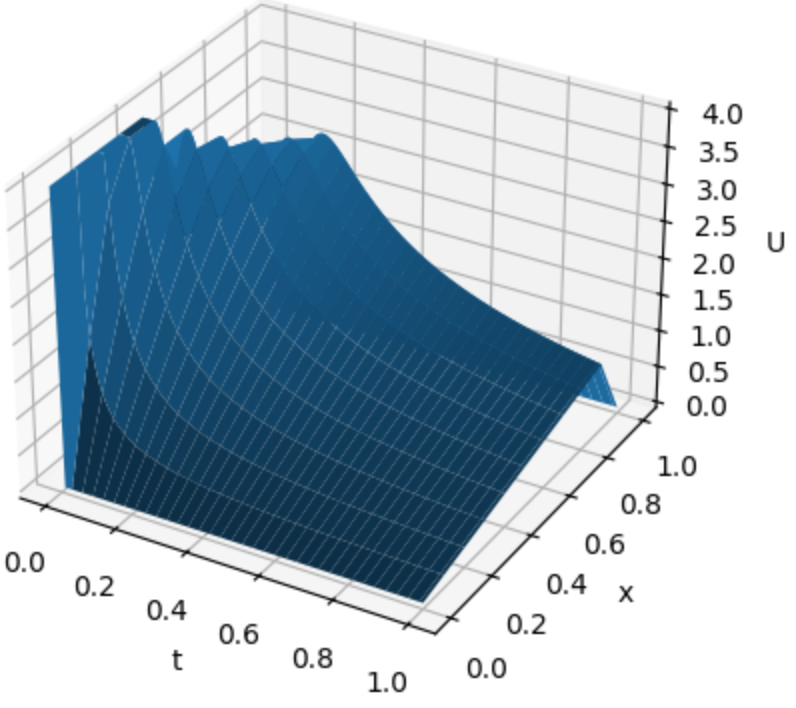
Примером разностной схемы для уравнения с искусственной вязкостью может быть следующая схема:

Упрощаем это выражение и разрешаем его относительно неизвестного значения сеточной функции на j+1 слое

Эта явная схема условно устойчива при выполнении неравенства

Пусть h = 0.1, .

***Результат:***



(приложение *main1.py*)

1. *Консервативные схемы*

Формально уравнение переноса можно записать в виде

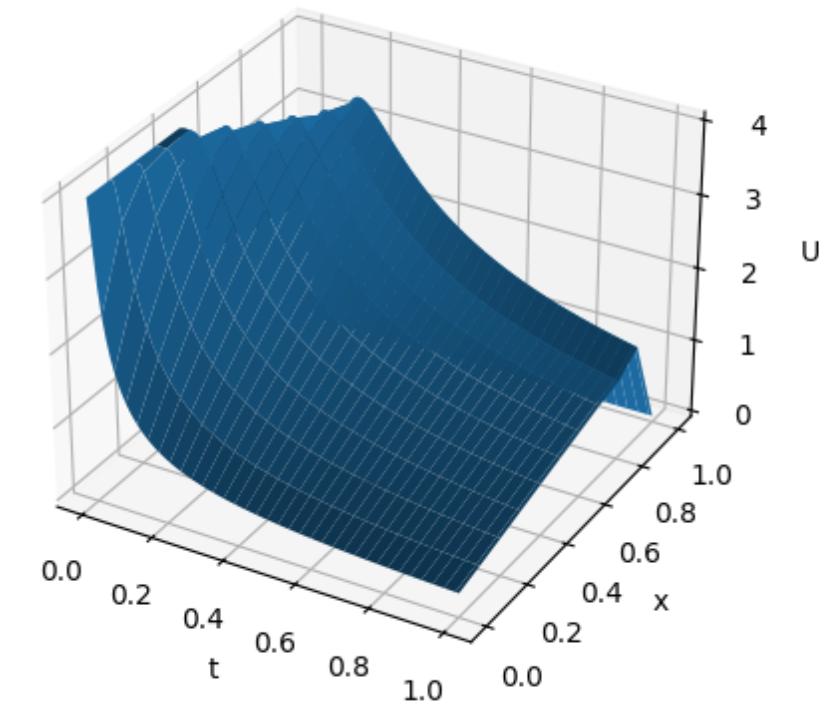
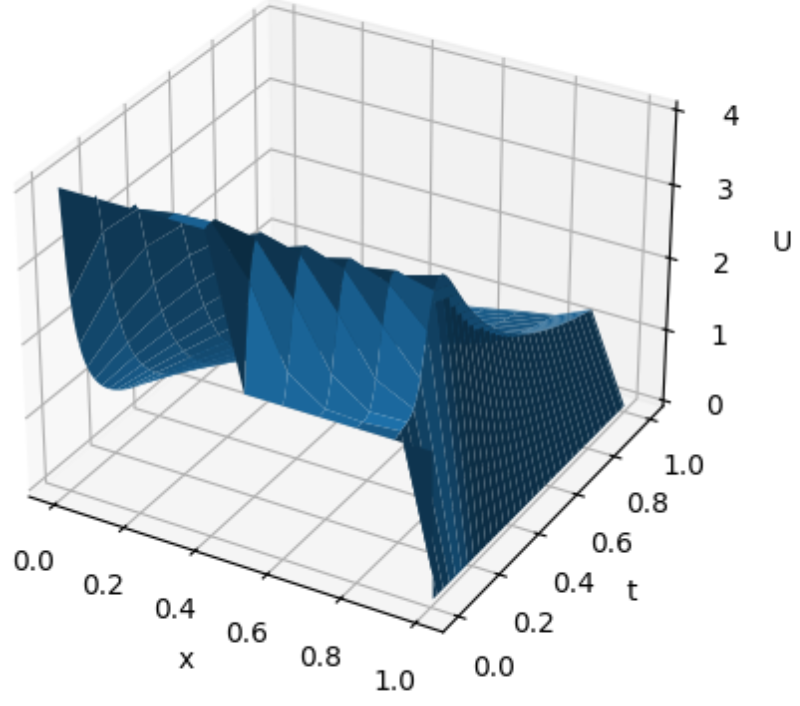
Интегрируем это уравнение по области , получаем

Используем численный метод вычисления интегралов, входящих в уравнение, а именно формулу прямоугольников, причем узлы предполагаем совпадающими с узлами рассматриваемой разностной сетки. Окончательно получим разностную схему вида

Отсюда можно найти значение искомой функции на верхнем слое с помощью решения на нижнем слое. Следовательно, это явная схема.

Упрощаем это выражение и разрешаем его относительно неизвестного значения сеточной функции на j+1 слое

***Результат:***

(приложение *main2.py*)

**Приложение**

*main1.py*

import matplotlib.pyplot as plt  
import numpy as np  
  
# Параметры  
h = 0.1  
tau = 0.01  
eps = 0.01  
a, b, c, d = 0, 1, 0, 1  
  
# Размеры сетки  
I = int((b - a) / h) + 1  
J = int((d - c) / tau) + 1  
  
# Инициализация массива  
array = np.zeros((J, I))  
  
# Начальные условия  
for i in range(I):  
 if a + i \* h >= 0.5:  
 array[0][i] = 2  
 else:  
 array[0][i] = 4  
  
# Расчёт значений по схеме  
for j in range(J - 1):  
 for i in range(1, I - 1):   
 array[j + 1][i] = (  
 array[j][i]  
 - tau / h \* array[j][i] \* (array[j][i] - array[j][i - 1])  
 - eps\*\*2 / 2 \* tau / h\*\*3 \* (array[j][i + 1] - 2 \* array[j][i] + array[j][i - 1])  
 )  
  
# Проверка условия устойчивости  
for j in range(J):  
 for i in range(I):  
 if array[j][i] > (h / tau):  
 print("Условие устойчивости не выполнилось!")  
 break  
  
# Построение сетки  
x = np.arange(a, b + h, h)  
t = np.arange(c, d + tau, tau)  
x, t = np.meshgrid(x, t)  
  
# Построение графика  
fig = plt.figure()  
ax = fig.add\_subplot(projection="3d")  
ax.set\_xlabel("t")  
ax.set\_ylabel("x")  
ax.set\_zlabel("U")  
ax.plot\_surface(t, x, array)  
plt.show()

*main2.py*

import matplotlib.pyplot as plt  
import numpy as np  
  
# Параметры  
h = 0.1  
tau = 0.01  
eps = 0.01  
a, b, c, d = 0, 1, 0, 1  
  
# Размеры сетки  
I = int((b - a) / h) + 1  
J = int((d - c) / tau) + 1  
  
# Инициализация массива  
array = np.zeros((J, I))  
  
# Начальные условия  
for i in range(I):  
 if a + i \* h >= 0.5:  
 array[0][i] = 2  
 else:  
 array[0][i] = 4  
  
# Проверка условия устойчивости  
for j in range(J - 1):  
 for i in range(I - 1):  
 array[j + 1][i] = (  
 array[j][i]  
 + tau / (2 \* h) \* (array[j][i - 1] \*\* 2 - array[j][i] \*\* 2)  
 - 0.01 \* (array[j][i + 1] - 2 \* array[j][i] + array[j][i - 1])  
 )  
  
# Построение сетки  
x = np.arange(a, b + h, h)  
t = np.arange(c, d + tau, tau)  
x, t = np.meshgrid(x, t)  
  
# Построение графика  
fig = plt.figure()  
ax = fig.add\_subplot(projection="3d")  
ax.set\_xlabel("t")  
ax.set\_ylabel("x")  
ax.set\_zlabel("U")  
ax.plot\_surface(t, x, array)  
plt.show()