**ОТЧЁТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 7**

**КВАЗИЛИНЕЙНОЕ УРАВНЕНИЕ ПЕРЕНОСА**

**(Вариант 5)**

*Выполнил студент 3 курса ПМ*

*Бут Дмитрий*

***Постановка задачи:***

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

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 уравнение переноса

методом с искусственной вязкостью и консервативной схемы.

Погрешность решения 0,01.

Входные данные: [*a*, *b*] = [0; 1], [*c*, *d*] = [0; 1], начальное условие:

***Результаты расчетов***

Во всех методах будем использовать погрешность , шаг по : , шаг по :

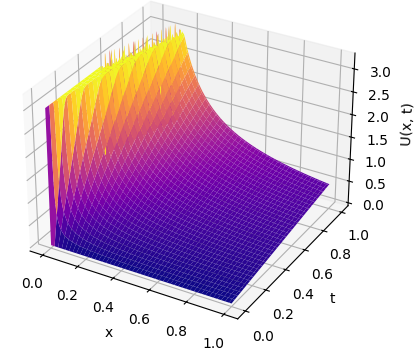
**Метод с искусственной вязкостью.** Вместо исходного квазилинейного уравнения рассмотрим уравнение

Здесь последний член в левой части описывает искусственную вязкость, при этом параметр мал. Ясно, что при малом значении решение этого уравнения при одинаковых начальных условиях будет близким к решению исходного, если эти решения достаточно гладкие (вторая производная ограничена). Примером разностной схемы для уравнения с искусственной вязкостью может быть следующая схема:

Упрощая это выражение и разрешая его относительно неизвестного значения сеточной функции на (j + 1)-м слое, получаем

Эта явная схема условно устойчива при выполнении неравенства

Произведя вычисления по схеме, получим точки, по которым можно составить график:



**Метод консервативной схемы.**

Формально квазилинейное уравнение переноса можно также за-

писать в дивергентной форме:

Проинтегрируем это уравнение по ячейке :

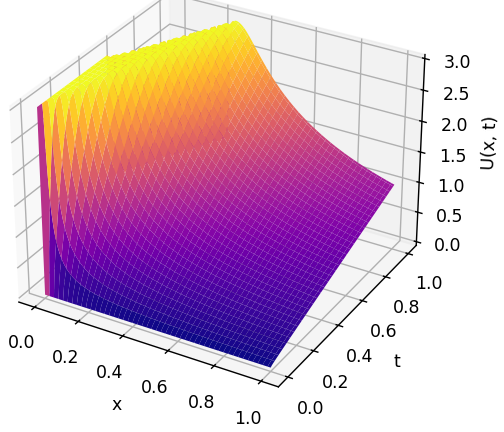
Или

В консервативных схемах дисбаланс равен нулю. Приведем пример построения такой схемы. Для этого нужно использовать некоторый численный метод вычисления интегралов, входящих в уравнение. Воспользуемся для простоты формулой прямоугольников, причем узлы интегрирования 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. Она обладает свойством консервативности.

Произведя вычисления по схеме, получим точки, по которым можно составить график:



**ПРИЛОЖЕНИЕ**

***Программа реализации метода с искусственной вязкостью***

import matplotlib.pyplot as plt  
import numpy as np  
  
  
def urmat(ux0, a, b, c, d, h, tau, eps):  
 x = np.arange(a, b + h, h)  
 t = np.arange(c, d + tau, tau)  
 u = np.zeros((len(x), len(t)))  
 for i in range(len(x)):  
 u[i][0] = ux0(x[i])  
 for j in range(len(t) - 1):  
 for i in range(1, len(x) - 1):  
 u[i][j + 1] = (u[i][j] - (tau / h) \* u[i][j] \* (u[i][j] - u[i - 1][j]) - (eps \*\* 2 \* tau / 2 / h \*\* 3) \*  
 (u[i + 1][j] - u[i - 1][j]) \* (u[i + 1][j] - u[i][j] + u[i - 1][j]))  
 u[len(x) - 1][j + 1] = u[i][j] - tau / h \* u[i][j] \* (u[i][j] - u[i - 1][j])  
 T, X = np.meshgrid(t, x)  
 fig = plt.figure()  
 ax = fig.add\_subplot(111, projection='3d')  
 ax.plot\_surface(T, X, u, cmap='plasma')  
 ax.set\_xlabel('x')  
 ax.set\_ylabel('t')  
 ax.set\_zlabel('U(x, t)')  
 plt.show()  
  
  
def ut0(x):  
 if x < 0.5:  
 return 2  
 else:  
 return 1  
  
  
a, b = 0, 1  
c, d = 0, 1  
h = 0.01  
tau = 0.001  
epsilon = 0.01  
urmat(ut0, a, b, c, d, h, tau, epsilon)

***Программа реализации консервативной схемы***

import matplotlib.pyplot as plt  
import numpy as np  
  
  
def urmat(ux0, a, b, c, d, h, tau, eps):  
 x = np.arange(a, b + h, h)  
 t = np.arange(c, d + tau, tau)  
 u = np.zeros((len(x), len(t)))  
 for i in range(len(x)):  
 u[i][0] = ux0(x[i])  
 for j in range(len(t) - 1):  
 for i in range(1, len(x)):  
 u[i][j + 1] = u[i][j] - tau / 2 / h \* (u[i][j] \*\* 2 - u[i - 1][j] \*\* 2)  
 T, X = np.meshgrid(t, x)  
 fig = plt.figure()  
 ax = fig.add\_subplot(111, projection='3d')  
 ax.plot\_surface(T, X, u, cmap='plasma')  
 ax.set\_xlabel('x')  
 ax.set\_ylabel('t')  
 ax.set\_zlabel('U(x, t)')  
 plt.show()  
  
  
def ut0(x):  
 if x < 0.5:  
 return 2  
 else:  
 return 1  
  
  
a, b = 0, 1  
c, d = 0, 1  
h = 0.01  
tau = 0.001  
epsilon = 0.01  
urmat(ut0, a, b, c, d, h, tau, epsilon)