ОТЧЁТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 7

Квазилинейное уравнение переноса

(Вариант 9)

Выполнил студент 3 курса ПМиИ
Кондратьев Виталий

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 U_{ij} искомой функции $U(t,x)$ с заданной точностью для некоторых значений аргументов

$x_j \in [a, b]$, $t_i \in [c, d]$

Численное решение таких дифференциальных уравнений возможно методами конечных разностей.

Погрешность решения, найденного этими методами, оценивается величиной $O(\tau^p, h^q)$, где p, q - порядок метода.

Задание.

Решить уравнение переноса

$$\frac{\partial U}{\partial t} + U\frac{\partial U}{\partial x} = 0$$

методом с искусственной вязкостью и консервативной схемы.

Варианты задания (лабораторная №7)

Для всех вариантов $[a, b]$ = [0; 1], $[c, d]$ = [0; 1]. Погрешность решения 0,01 (определяется сходимостью схемы и величиной шагов).

№ вариантов	Начальное условие
9,19,29	$\begin{cases} 2, x \geq 0,5 \\ 4, x < 0,5 \end{cases}$

Метод с искусственной вязкостью

Вместо исходного квазилинейного уравнения рассматривается уравнение:

$$\frac{\partial U}{\partial t} + U\frac{\partial U}{\partial x} + \frac{\varepsilon^2}{2}\frac{\partial}{\partial x}\left(\frac{\partial U}{\partial x}\right)^2 = 0 \qquad (1)$$

Примером разностной схемы для уравнения (1) с искусственной вязкостью может быть следующая схема:

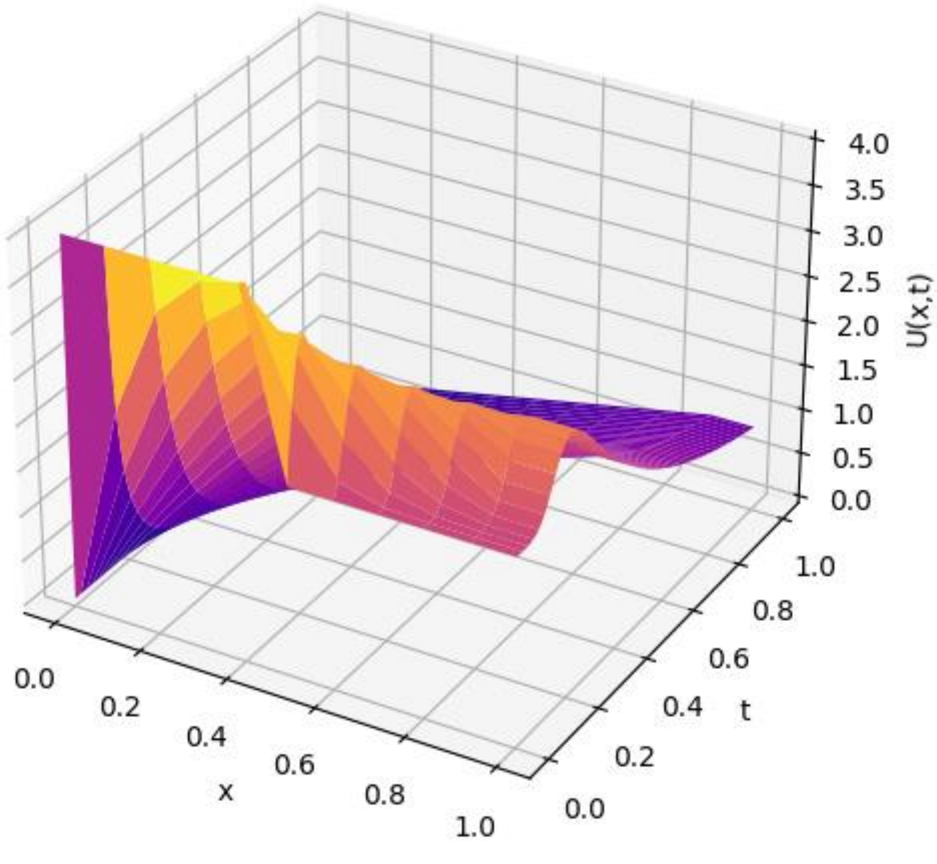
$$\frac{u_i^{j+1} - u_i^j}{\tau} + u_i^j\frac{u_i^j - u_{i-1}^j}{h} + \frac{\varepsilon^2}{2}\frac{1}{h}\left[\left(\frac{u_{i+1}^j - u_i^j}{h}\right)^2 - \left(\frac{u_i^j - u_{i-1}^j}{h}\right)^2\right] = 0$$

Упрощая это выражение и разрешая его относительно u_i^{j+1}, получаем:

$$u_i^{j+1} = u_i^j - \frac{\tau}{h}u_i^j\left(u_i^j - u_{i-1}^j\right) - \frac{\varepsilon^2\tau}{2h^3}(u_{i+1}^j - u_{i-1}^j)(u_{i+1}^j - 2u_i^j + u_{i-1}^j)$$

Эта явная схема условно устойчива при выполнении неравенства:

$$\tau \le h/U(x,t)$$



Консервативная схема

Квазилинейное уравнение переноса можно также записать в дивергентной форме:

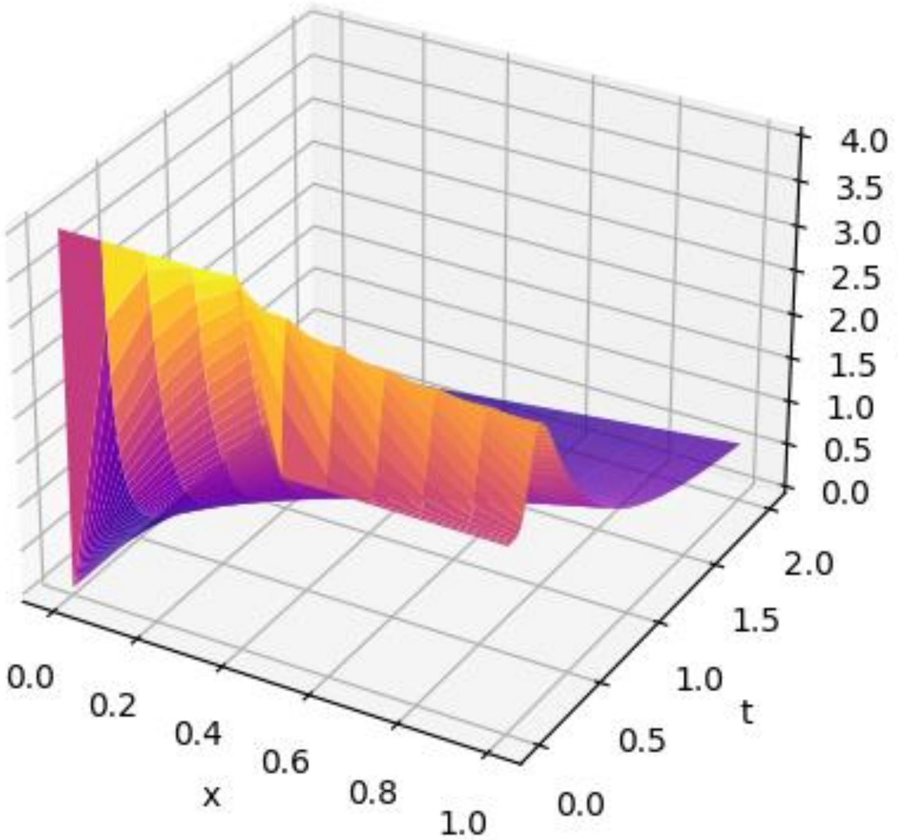
$$\frac{\partial U}{\partial t} + \frac{\partial}{\partial x}\left(\frac{U^2}{2}\right) = 0$$

Воспользуемся формулой прямоугольников, причем узлы предполагаем совпадающими с узлами рассматриваемой разностной сетки:

$$\frac{u_i^{j+1} - u_i^j}{\tau} + \frac{(u_i^j)^2 - (u_{i-1}^j)^2}{2h} = 0$$

Упрощая это выражение и разрешая его относительно u_i^{j+1}, получаем:

$$u_i^{j+1} = u_i^j - \frac{\tau}{2h}((u_i^j)^2 - (u_{i-1}^j)^2) \qquad \text{– явная схема.}$$



ПРИЛОЖЕНИЕ

Метод с искусственной вязкостью

```
"""
Лабораторная работа №7
Студент ОНК «ИВТ» ВШ КНиИИ направления ПМиИ 3 курса
Кондратьев Виталий
Вариант 9
"""

import numpy as np
import matplotlib.pyplot as plt


def Ux0(x):
    if x >= 0.5:
        return 2
    else:
        return 4


h = 0.1
T = 0.001
a = 1
d = 1
eps = 0.01
p = int(a / h) + 1
q = int(d / T) + 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)


for j in range(0, q - 1):
    for i in range(1, p - 1):
        x = h * i
        t = T * j
        U[i][j + 1] = U[i][j] - T / h * U[i][j] * (U[i][j] - U[i - 1][j]) -
eps ** 2 * T / (2 * h ** 3) * (
                    U[i + 1][j] - U[i - 1][j]) * (U[i + 1][j] - 2 * U[i][j] +
U[i - 1][j])
        U[p - 1][j + 1] = U[i][j] - T / h * U[i][j] * (U[i][j] - U[i - 1][j])

u, v = np.mgrid[0:p, 0:q]
x = h * u
y = T * v
z = x - x
for j in range(0, q):
    for i in range(0, p):
        z[i][j] = U[i][j]

fig = plt.figure(figsize=plt.figaspect(0.5))
axes = fig.add_subplot(1, 2, 1, projection='3d')
axes.set_xlabel("x")
axes.set_ylabel("t")
axes.set_zlabel("U(x,t)")
axes.plot_surface(x, y, z, cmap='plasma')
plt.show()
```

Консервативная схема

```
"""
Лабораторная работа №7
Студент ОНК «ИВТ» ВШ КНиИИ направления ПМиИ 3 курса
Кондратьев Виталий
Вариант 9
"""
import numpy as np
import matplotlib.pyplot as plt


def Ux0(x):
    if x >= 0.5:
        return 2
    else:
        return 4


h = 0.1
T = 0.001
a = 1
d = 2
p = int(a / h) + 1
q = int(d / T) + 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)


for j in range(0, q - 1):
    for i in range(1, p):
        x = h * i
        t = T * j
        U[i][j + 1] = U[i][j] - T / (2 * h) * (U[i][j] ** 2 - U[i - 1][j] **
2)

u, v = np.mgrid[0:p, 0:q]
x = h * u
y = T * v
z = x - x
for j in range(0, q):
    for i in range(0, p):
        z[i][j] = U[i][j]

fig = plt.figure(figsize=plt.figaspect(0.5))
axes = fig.add_subplot(1, 2, 1, projection='3d')
axes.set_xlabel("x")
axes.set_ylabel("t")
axes.set_zlabel("U(x,t)")
axes.plot_surface(x, y, z, rstride=1, cstride=15, cmap='plasma')
plt.show()
```