**ОТЧЁТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 7**

**ПРИБЛИЖЕННЫЕ МЕТОДЫ РЕШЕНИЯ**

**НЕЛИНЕЙНЫХ УРАВНЕНИЙ**

**(Вариант 10)**

*Выполнил студент 3 курса МОиАИС*

*Соколов Арсений*

**Задание:**

Решить уравнение переноса

методом с искусственной вязкостью и консервативной схемы.

**Дано:**

[*a*, *b*] = [0; 1],

[*c*, *d*] = [0; 1],

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

|  |  |
| --- | --- |
| № вариантов | Начальное условие |
| 10,20,30 |  |

**Решение:**

1. *Метод с искусственной вязкостью*

Для стабилизации решения уравнения переноса, необходимо ввести «искусственную вязкость».

Уравнение примет вид

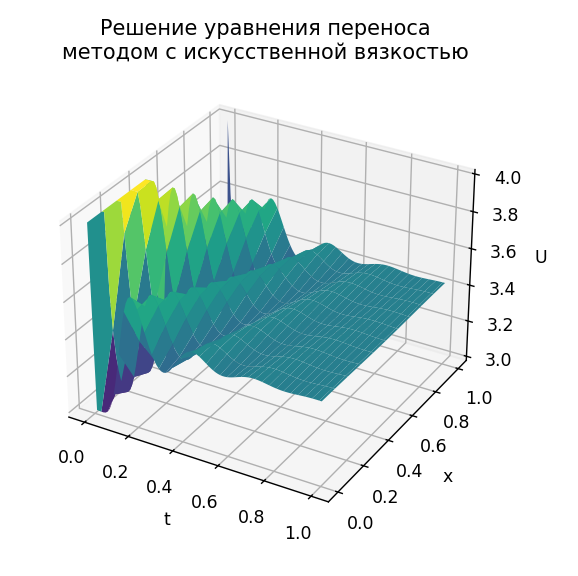
где – член искусственной вязкости, зависящий от пространственного шага.

Одна из возможных схем

Или

Это явная схема, поэтому условие устойчивости

Тогда



1. *Метод консервативной схемы*

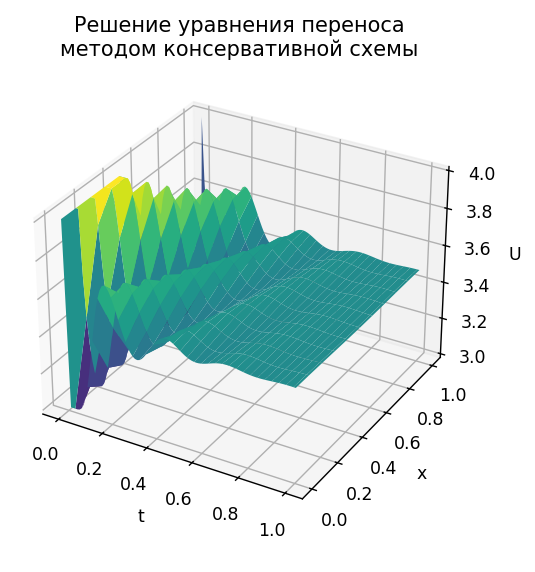
Формально уравнение переноса можно записать в виде

Интегрируем по всей области

Это явная схема, поэтому для интегрирования используем метод прямоугольников

Разрешим его относительно неизвестного значения сеточной функции на j+1 слое

Тогда

****

**ПРИЛОЖЕНИЕ 1**

***Программа вычисления уравнения переноса методом с искусственной вязкостью***

import matplotlib.pyplot as plt  
import numpy as np  
  
# Параметры  
h = 0.1 # шаг по пространству  
tau = 0.01 # шаг по времени  
eps = 0.01 # точность  
  
# Границы области  
a = c = 0  
b = d = 1  
  
# Количество точек по пространству и времени  
I = int((b - a) / h) + 1  
J = int((d - c) / tau) + 1  
u = np.zeros((J, I))  
  
# Начальные условия  
for i in range(I):  
 x\_i = a + i \* h  
 if x\_i >= 0.5:  
 u[0][i] = 3  
 else:  
 u[0][i] = 4  
  
# Вычисление значений u  
for j in range(J - 1):  
 for i in range(1, I - 1):  
 if np.abs(u[j][i]) > h / tau:  
 print("Условие устойчивости не выполнено")  
 exit(1) # Выход из программы  
 u[j + 1][i] = (  
 u[j][i] - tau / h \* u[j][i] \* (u[j][i] - u[j][i - 1]) - eps \*\* 2 / 2 \* tau / h \*\* 3 \* (u[j][i + 1] -  
 u[j][i - 1]) \* (u[j][i + 1] - u[j][i] + u[j][i - 1])  
 )  
  
 # Периодические граничные условия  
 u[j + 1][0] = u[j + 1][-2]  
 u[j + 1][-1] = u[j + 1][1]  
  
# Визуализация  
x = np.arange(a, b + h, h)  
t = np.arange(c, d + tau, tau)  
x, t = np.meshgrid(x, t)  
fig = plt.figure()  
ax = fig.add\_subplot(projection='3d')  
ax.set\_xlabel("t")  
ax.set\_ylabel("x")  
ax.set\_zlabel("U")  
ax.set\_title('Решение уравнения переноса\nметодом с искусственной вязкостью')  
ax.plot\_surface(t, x, u, cmap="viridis")  
  
plt.show()

**ПРИЛОЖЕНИЕ 2**

***Программа вычисления уравнения переноса методом консервативной схемы***

import numpy as np  
import matplotlib.pyplot as plt  
  
# Параметры  
h = 0.1 # шаг по пространству  
tau = 0.01 # шаг по времени  
eps = 0.01 # точность  
  
# Границы области  
a = c = 0  
b = d = 1  
  
# Количество точек по пространству и времени  
I = int((b - a) / h) + 1  
J = int((d - c) / tau) + 1  
u = np.zeros((J, I))  
  
# Начальные условия  
for i in range(I):  
 if a + i \* h >= 0.5: # Если x больше или равен 0.5  
 u[0][i] = 3 # Устанавливаем значение u в 3  
 else:  
 u[0][i] = 4 # Устанавливаем значение u в 4  
  
# Вычисление значений u  
for j in range(J - 1):  
 for i in range(I - 1):  
 if np.abs(u[j][i]) > h / tau:  
 print("Условие устойчивости не выполнено")  
 exit(1) # Выход из программы  
  
 u[j + 1][i] = u[j][i] + tau / (2 \* h) \* (u[j][i - 1] \*\* 2 - u[j][i] \*\* 2)  
  
 # Периодические граничные условия  
 u[j + 1][0] = u[j + 1][-2]  
 u[j + 1][-1] = u[j + 1][1]  
  
# Визуализация  
x = np.arange(a, b + h, h)  
t = np.arange(c, d + tau, tau)  
x, t = np.meshgrid(x, t)  
fig = plt.figure()  
ax = fig.add\_subplot(projection='3d')  
ax.set\_xlabel("t")  
ax.set\_ylabel("x")  
ax.set\_zlabel("U")  
ax.set\_title('Решение уравнения переноса\nметодом консервативной схемы')  
ax.plot\_surface(t, x, u, cmap="viridis")  
  
plt.show()