**ОТЧЁТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6**

**Линейное уравнение переноса**

**Вариант 9**

*Выполнил студент 3 курса МОиАИС*

*Ходосевич Данила*

Численно решить уравнение переноса

1. Для полуплоскости
2. В прямоугольнике

Во всех случаях *a* – const. Применить следующие шаблоны для явных и неявных схем.

*i,j+1 iI,j+1 i-1,j+1 i,j+1 i-1,j+1 i,j+1*

*i-1,j i,j i,j i+1,j i,j i-1,j i,j*

a>0 a<0 a>0 a>0; a<0

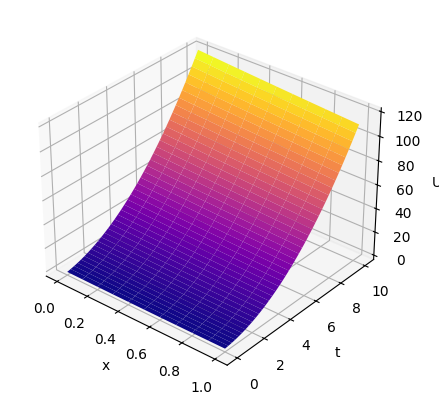
Схемы выбирать в зависимости от знака a (см. на схеме). Параметр a принимает два значения: a=2 и a= -2

Для полуплоскости и для прямоугольной области решить задачу от 0 до 1 с шагом 0.1 по х и от 0 до 10 по времени с шагом, отвечающим условиям устойчивости. Для полуплоскости применить схему 1 и 2 , для прямоугольной области применить все возможные схемы.

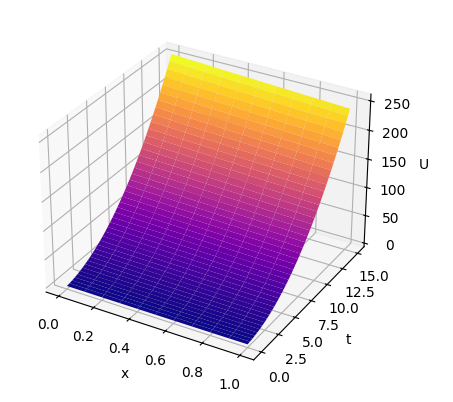
Результаты вывести в виде трехмерных рисунков.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 9 | x2-x-1 | t2-t-1 | t2-t-1 | 3x |

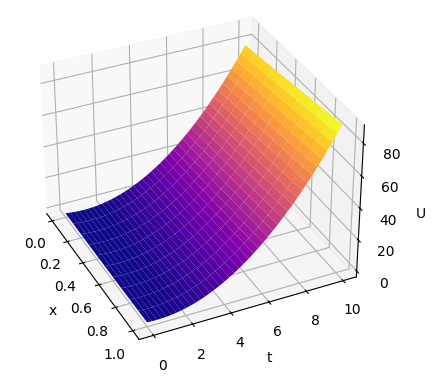
* 1. a > 0 полуплоскость



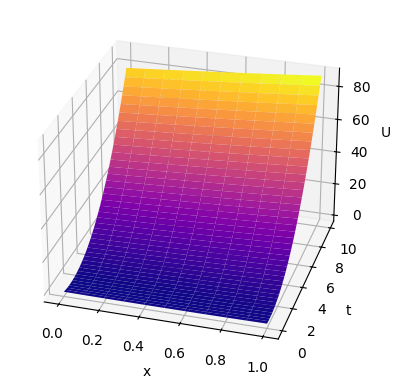
* 2. a< 0 полуплоскость



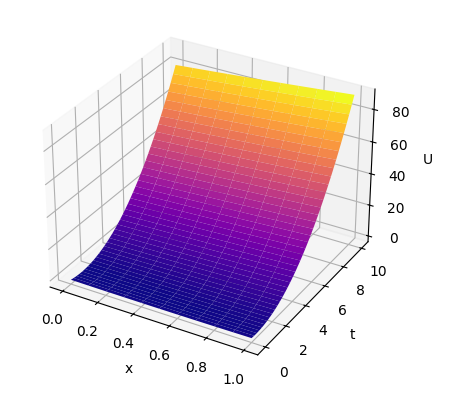
* 1. a > 0 прямоугольник



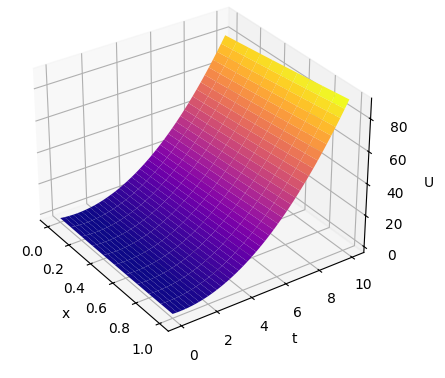
* 2. a< 0 прямоугольник



* 3. a > 0 прямоугольник



* 4. a< 0 прямоугольник



**Приложения**

import numpy as np

import matplotlib.pyplot as plt

#f(x,t)

def f(x, t):

    return 3 \* x

#U(x,0)

def Ux(x):

    return x \*\* 2 - x - 1

#U(0,t) для a>0 и прямоугольной области

def Ut\_0(t):

    return t \*\* 2 - t - 1

#U(1,t) для a<0 и прямоугольной области

def Ut\_1(t):

    return t \*\* 2 - t - 1

# 1 схема

def first\_scheme(I, J, tau, U, f, x, t):

    for i in range(1, I + 1):

        for j in range(0, J):

            U[i][j + 1] = U[i - 1][j] + tau \* f(x[i], t[j])

    return U

# 2 схема

def second\_scheme(I, J, tau, U, f, x, t):

    for i in range(I - 1, -1, -1):

        for j in range(0, J):

            U[i][j + 1] = U[i + 1][j] - tau \* f(x[i], t[j])

    return U

# 3 схема

def third\_scheme(I, J, tau, U, f, x, t, a):

    if a > 0:

        for i in range(1, I + 1):

            for j in range(0, J):

                U[i][j + 1] = (U[i][j] + U[i - 1][j + 1] + tau \* f(x[i], t[j])) / 2

    else:

        for i in range(I - 1, -1, -1):

            for j in range(0, J):

                U[i][j + 1] = (U[i][j] + U[i + 1][j + 1] - tau \* f(x[i], t[j])) / 2

    return U

# 4 схема

def fourth\_scheme(I, J, tau, U, f, x, t, a, h):

    if a > 0:

        for i in range(1, I + 1):

            for j in range(0, J):

                U[i][j + 1] = U[i - 1][j] + tau \* f(x[i] + h / 2, t[j] + tau / 2)

    else:

        for i in range(I - 1, -1, -1):

            for j in range(0, J):

                U[i][j + 1] = U[i + 1][j] - tau \* f(x[i] + h / 2, t[j] + tau / 2)

    return U

# Функция для инициализации условий задачи

def init(I, J, h, tau, a, fx, ft, rectangle):

    if rectangle:

        I1 = I

    else:

        I1 = I + J

    U = np.zeros((I1 + 1, J + 1))

    x = np.zeros(I1 + 1)

    t = np.zeros(J + 1)

    if rectangle:

        for i in range(I + 1):

            x[i] = h \* i

            U[i][0] = fx(x[i])

        if (a > 0):

            for j in range(J + 1):

                t[j] = tau \* j

                if (j != 0): U[0][j] = ft(t[j])

        else:

            for j in range(J + 1):

                t[j] = tau \* j

                if (j != 0): U[I][j] = ft(t[j])

    else:

        if (a > 0):

            for i in range(J + I + 1):

                x[i] = h \* (i - J)

                U[i][0] = fx(x[i])

        else:

            for i in range(I + J + 1):

                x[i] = i \* h

                U[i][0] = fx(x[i])

                for i in range(J + 1):

                    t[i] = i \* tau

    return x, t, U

# Функция для построения 3D графика

def draw\_plot(graph):

    for gr in graph:

        gr[0], gr[1] = np.meshgrid(gr[0], gr[1])

        gr[0], gr[1] = gr[0].T, gr[1].T

        fig = plt.figure()

        ax = fig.add\_subplot(projection='3d')

        ax.set\_xlabel("x")

        ax.set\_ylabel("t")

        ax.set\_zlabel("U")

        ax.set\_rasterization\_zorder(1)

        ax.plot\_surface(gr[0], gr[1], gr[2], cmap='plasma')

    plt.show()

# Задаем параметры задачи

rectangle = False

x\_start = 0

x\_end = 1

time\_start = 0

time\_end = 10

h = 0.1

tau = 0.05

a1 = -2

a2 = 2

I = int((x\_end - x\_start) / h)

J = int((time\_end - time\_start) / tau)

graph = []

# Код для построения графиков

x, t, U = init(I, J, h, tau, a2, Ux, Ut\_0, False)

U = first\_scheme(I + J, J, tau, U, f, x, t)

x, t = np.linspace(x\_start, x\_end, I + 1), np.linspace(time\_start, time\_end, J + 1)

U = U[J:J + I + 1, :]

graph.append([x, t, U])

print('Полуплоскость a>0 схема 1')

x, t, U = init(I, J, h, tau, a1, Ux, Ut\_1, False)

U = second\_scheme(I + J, J, tau, U, f, x, t)

x, t = np.linspace(x\_start, x\_end, I + 1), np.linspace(time\_start, time\_end, J + 1)

U = U[0:I + 1, :]

graph.append([x, t, U])

print('Полуплоскость a<0 схема 2')

x, t, U = init(I, J, h, tau, a2, Ux, Ut\_0, True)

U = first\_scheme(I, J, tau, U, f, x, t)

x, t = np.linspace(x\_start, x\_end, I + 1), np.linspace(time\_start, time\_end, J + 1)

graph.append([x, t, U])

print('Прямоугольник a>0 схема 1')

x, t, U = init(I, J, h, tau, a2, Ux, Ut\_0, True)

U = third\_scheme(I, J, tau, U, f, x, t, a2)

x, t = np.linspace(x\_start, x\_end, I + 1), np.linspace(time\_start, time\_end, J + 1)

graph.append([x, t, U])

print('Прямоугольник a>0 схема 3')

x, t, U = init(I, J, h, tau, a2, Ux, Ut\_0, True)

U = fourth\_scheme(I, J, tau, U, f, x, t, a2, h)

x, t = np.linspace(x\_start, x\_end, I + 1), np.linspace(time\_start, time\_end, J + 1)

graph.append([x, t, U])

print('Прямоугольник a>0 схема 4')

x, t, U = init(I, J, h, tau, a1, Ux, Ut\_1, True)

U = second\_scheme(I, J, tau, U, f, x, t)

x, t = np.linspace(x\_start, x\_end, I + 1), np.linspace(time\_start, time\_end, J + 1)

graph.append([x, t, U])

print('Прямоугольник a<0 схема 2')

x, t, U = init(I, J, h, tau, a1, Ux, Ut\_1, True)

U = fourth\_scheme(I, J, tau, U, f, x, t, a1, h)

x, t = np.linspace(x\_start, x\_end, I + 1), np.linspace(time\_start, time\_end, J + 1)

graph.append([x, t, U])

print('Прямоугольник a<0 схема 4')

draw\_plot(graph)