**ОТЧЁТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6**

**ЛИНЕЙНОЕ УРАВНЕНИЕ ПЕРЕНОСА**

**(Вариант 5)**

*Выполнил студент 3 курса ПМ*

*Бут Дмитрий*

***Постановка задачи:***

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.

Входные данные запишем в виде таблицы:

|  |  |  |  |
| --- | --- | --- | --- |
| U(x,0) | U(0,t) для a>0 и прямоугольной области | U(1,t) для a<0 и прямоугольной области | *f(x,t)* |
| x2-2 | t2-2 | t2-1 | 2x |

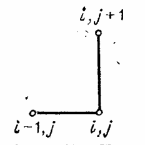
***Результаты расчетов***

Рассмотрим равномерную сетку, построенную с помощью прямых

(), ()

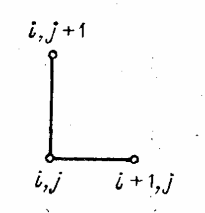
Рассматриваем , удовлетворяющее условии устойчивости задачи:

Вместо функций будем рассматривать сеточные функции значения которых в узлах () соответственно равны . Для построения разностной схемы воспользуемся шаблонами:



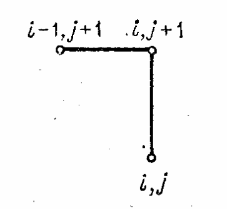
Аппроксимируем входящие в уравнение производные конечно-разностными соотношениями с использованием односторонних разностей:

Решая разностное уравнение относительно единственного значения на j+1 слое, получаем следующую разностную схему:

Где

Аппроксимируем входящие в уравнение производные конечно-разностными соотношениями с использованием односторонних разностей:

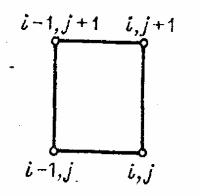
Решая разностное уравнение относительно единственного неизвестного значения на j+1 слое, получаем следующую разностную схему:

Где



Аппроксимируем входящие в уравнение производные конечно-разностными соотношениями с использованием односторонних разностей:

Решая разностное уравнение относительно единственного значения на j+1 слое, получаем следующую разностную схему:

Где

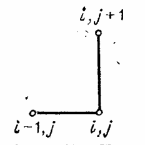


Аппроксимируем входящие в уравнение производные конечно-разностными соотношениями с использованием односторонних разностей:

Где

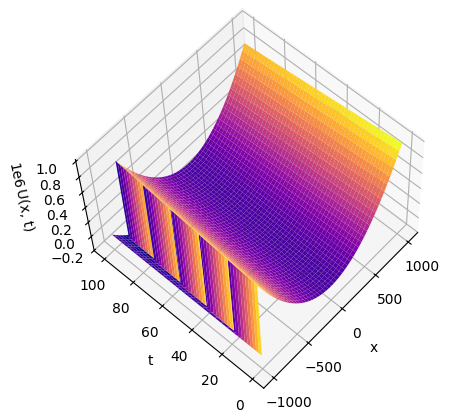
Решая разностное уравнение относительно единственного значения на j+1 слое, получаем следующую разностную схему:

Где

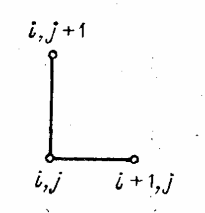


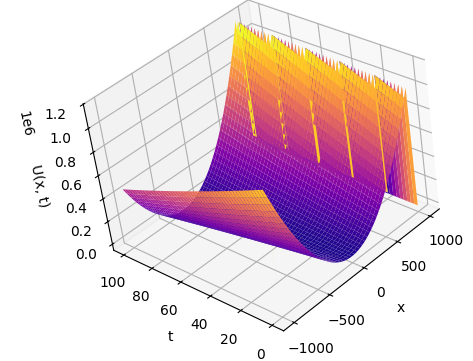
**Полуплоскость.**

При a=2 и шаблоне

****

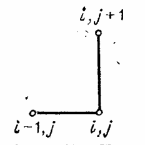
При и шаблоне

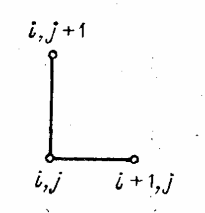
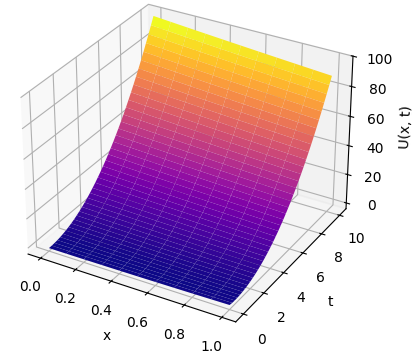




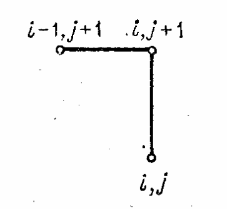
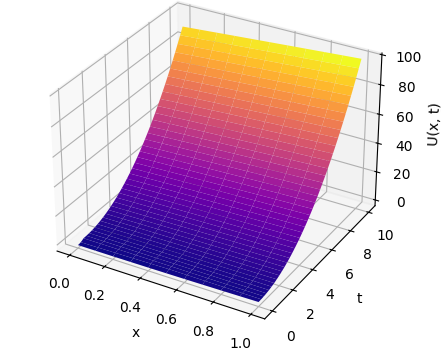
Исходные тексты программ в ПРИЛОЖЕНИИ 1 и ПРИЛОЖЕНИИ 2 соответственно

**Прямоугольник.** При:

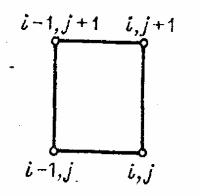
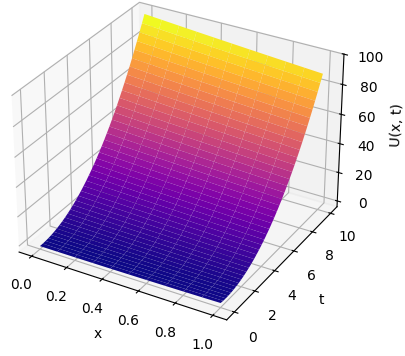
1. и шаблоне 

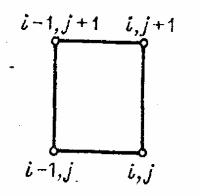
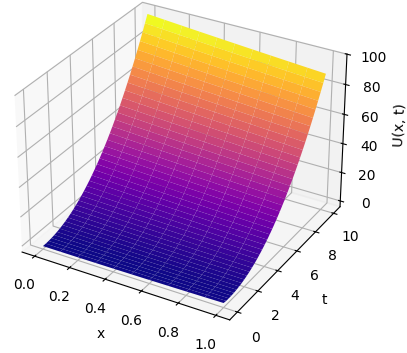
1. и шаблоне

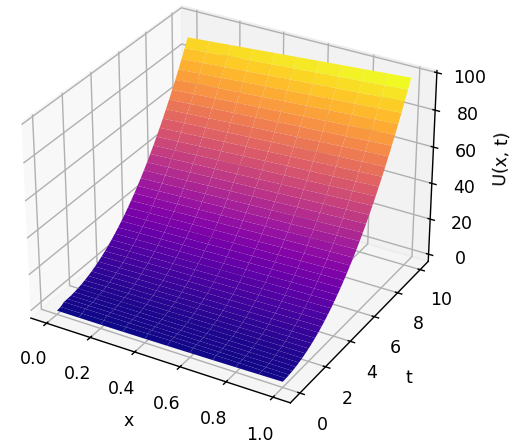
1. и шаблоне

1. и шаблоне

 ****

1. и шаблоне



**ПРИЛОЖЕНИЕ**

***ПРИЛОЖЕНИЕ 1***

import matplotlib.pyplot as plt

import numpy as np

def urmat(ux0, ut0, f, a, h):

    x = np.arange(-1000, 1000 + h, h)

    tau = h / a

    t = np.arange(0, 100 + tau, tau)

    u = np.zeros((len(x), len(t)))

    for i in range(len(x)):

        u[i][0] = ux0(x[i])

    for i in range(1, len(x)):

        for j in range(len(t)):

            u[0][j] = ut0(t[j])

        for j in range(len(t) - 1):

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

    # x = np.arange(0, 1 + h, h)

    x, T = np.meshgrid(x, t)

    fig = plt.figure()

    ax = fig.add\_subplot(111, projection='3d')

    ax.plot\_surface(x, T, u.T, cmap='plasma')

    ax.set\_xlabel('x')

    ax.set\_ylabel('t')

    ax.set\_zlabel('U(x, t)')

    plt.show()

ux0 = lambda x: x\*\*2 - 2

f = lambda x, t: 2 \* x

ut = lambda t: t\*\*2 - 2

a = 2

h = 0.1

urmat(ux0, ut, f, a, h)

***ПРИЛОЖЕНИЕ 2***

import matplotlib.pyplot as plt

import numpy as np

def urmat(ux0, ut1, f, a, h):

    x = np.arange(-1000, 1000 + h, h)

    tau = - h / a

    t = np.arange(0, 100 + tau, tau)

    u = np.zeros((len(x), len(t)))

    for i in range(len(x)):

        u[i][0] = ux0(x[i])

    for i in range(len(x) - 2, -1, -1):

        for j in range(len(t)):

            u[0][j] = ut1(t[j])

        for j in range(len(t) - 1):

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

    #x = np.arange(0, 80 + h, h)

    x, T = np.meshgrid(x, t)

    fig = plt.figure()

    ax = fig.add\_subplot(111, projection='3d')

    ax.plot\_surface(x, T, u[:].T, cmap='plasma')

    ax.set\_xlabel('x')

    ax.set\_ylabel('t')

    ax.set\_zlabel('U(x, t)')

    plt.show()

ux0 = lambda x: x\*\*2 - 2

f = lambda x, t: 2 \* x

ut = lambda t: t\*\*2 - 1

a = -2

h = 0.1

urmat(ux0, ut, f, a, h)

***ПРИЛОЖЕНИЕ 3***

import matplotlib.pyplot as plt

import numpy as np

def urmat(ux0, ut0, f, a, h):

    x = np.arange(0, 1 + h, h)

    tau = h / a

    t = np.arange(0, 10 + tau, tau)

    u = np.zeros((len(x), len(t)))

    for i in range(len(x)):

        u[i][0] = ux0(x[i])

    for i in range(1, len(x)):

        for j in range(len(t)):

            u[0][j] = ut0(t[j])

        for j in range(len(t) - 1):

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

    x, T = np.meshgrid(x, t)

    fig = plt.figure()

    ax = fig.add\_subplot(111, projection='3d')

    ax.plot\_surface(x, T, u.T, cmap='plasma')

    ax.set\_xlabel('x')

    ax.set\_ylabel('t')

    ax.set\_zlabel('U(x, t)')

    plt.show()

ux0 = lambda x: x\*\*2 - 2

f = lambda x, t: 2 \* x

ut = lambda t: t\*\*2 - 2

a = 2

h = 0.1

urmat(ux0, ut, f, a, h)

***ПРИЛОЖЕНИЕ 4***

import matplotlib.pyplot as plt

import numpy as np

def urmat(ux0, ut1, f, a, h):

    x = np.arange(0, 1 + h, h)

    tau = - h / a

    t = np.arange(0, 10 + tau, tau)

    u = np.zeros((len(x), len(t)))

    for i in range(len(x)):

        u[i][0] = ux0(x[i])

    for i in range(len(x) - 2, -1, -1):

        for j in range(len(t)):

            u[-1][j] = ut1(t[j])

        for j in range(len(t) - 1):

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

    x, T = np.meshgrid(x, t)

    fig = plt.figure()

    ax = fig.add\_subplot(111, projection='3d')

    ax.plot\_surface(x, T, u[:].T, cmap='plasma')

    ax.set\_xlabel('x')

    ax.set\_ylabel('t')

    ax.set\_zlabel('U(x, t)')

    plt.show()

ux0 = lambda x: x\*\*2 - 2

f = lambda x, t: 2 \* x

ut = lambda t: t\*\*2 - 1

a = -2

h = 0.1

urmat(ux0, ut, f, a, h)

***ПРИЛОЖЕНИЕ 5***

import matplotlib.pyplot as plt

import numpy as np

def urmat(ux0, ut0, f, a, h):

    x = np.arange(0, 1 + h, h)

    tau = h / a

    t = np.arange(0, 10 + tau, tau)

    u = np.zeros((len(x), len(t)))

    for i in range(len(x)):

        u[i][0] = ux0(x[i])

    for i in range(1, len(x)):

        for j in range(len(t)):

            u[0][j] = ut0(t[j])

        for j in range(len(t) - 1):

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

    x, T = np.meshgrid(x, t)

    fig = plt.figure()

    ax = fig.add\_subplot(111, projection='3d')

    ax.plot\_surface(x, T, u.T, cmap='plasma')

    ax.set\_xlabel('x')

    ax.set\_ylabel('t')

    ax.set\_zlabel('U(x, t)')

    plt.show()

ux0 = lambda x: x\*\*2 - 2

f = lambda x, t: 2 \* x

ut = lambda t: t\*\*2 - 2

a = 2

h = 0.1

urmat(ux0, ut, f, a, h)

***ПРИЛОЖЕНИЕ 6***

import matplotlib.pyplot as plt

import numpy as np

def urmat(ux0, ut0, f, a, h):

    x = np.arange(0, 1 + h, h)

    tau = h / a

    t = np.arange(0, 10 + tau, tau)

    u = np.zeros((len(x), len(t)))

    l = a \* tau / h

    for i in range(len(x)):

        u[i][0] = ux0(x[i])

    for i in range(1, len(x)):

        for j in range(len(t)):

            u[0][j] = ut0(t[j])

        for j in range(len(t) - 1):

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

    x, T = np.meshgrid(x, t)

    fig = plt.figure()

    ax = fig.add\_subplot(111, projection='3d')

    ax.plot\_surface(x, T, u.T, cmap='plasma')

    ax.set\_xlabel('x')

    ax.set\_ylabel('t')

    ax.set\_zlabel('U(x, t)')

    plt.show()

ux0 = lambda x: x\*\*2 - 2

f = lambda x, t: 2 \* x

ut = lambda t: t\*\*2 - 2

a = 2

h = 0.1

urmat(ux0, ut, f, a, h)

***ПРИЛОЖЕНИЕ 7***

import matplotlib.pyplot as plt

import numpy as np

def urmat(ux0, ut1, f, a, h):

    x = np.arange(0, 1 + h, h)

    tau = - h / a

    t = np.arange(0, 10 + tau, tau)

    u = np.zeros((len(x), len(t)))

    l = a \* tau / h

    for i in range(len(x)):

        u[i][0] = ux0(x[i])

    for i in range(len(x) - 2, -1, -1):

        for j in range(len(t)):

            u[-1][j] = ut1(t[j])

        for j in range(len(t) - 1):

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

    x, T = np.meshgrid(x, t)

    fig = plt.figure()

    ax = fig.add\_subplot(111, projection='3d')

    ax.plot\_surface(x, T, u[:].T, cmap='plasma')

    ax.set\_xlabel('x')

    ax.set\_ylabel('t')

    ax.set\_zlabel('U(x, t)')

    plt.show()

ux0 = lambda x: x\*\*2 - 2

f = lambda x, t: 2 \* x

ut = lambda t: t\*\*2 - 1

a = -2

h = 0.1

urmat(ux0, ut, f, a, h)