**ОТЧЁТ**

**ЛАБОРАТОРНАЯ РАБОТА №8**

**Уравнения гиперболического типа**

**Вариант 9**

*Выполнил студент 3 курса МОиАИС*

*Ходосевич Данила*

***Цель работы***: усвоить сущность и методы решения ***линейного дифференциального уравнения 2-го порядка гиперболического типа***.

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 находят методами конечных разностей.

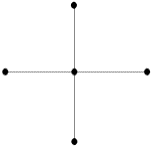
Погрешность решения, найденного этими методами, оценивается величиной O(*τp,hq*)*,* где *p*, *q* - порядок аппроксимации метода.

***Задание.***

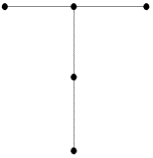
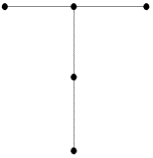
Решить волновое уравнение

явным методом и неявными методами второго порядка точности

Шаблон для явного метода:



Шаблон для неявного метода:

Вывести результаты в виде двумерных графиков U(x,t).

Неявные схемы решать с помощью прогонки.

Для всех вариантов [*a*, *b*] = [0; 1], [*c*, *d*] = [0; 10], *f(x,t)*=0

Погрешность решения 0,01. Исходя из погрешности, порядка аппроксимации и условий сходимости для явных схем определить шаги по пространству и времени

|  |  |  |  |
| --- | --- | --- | --- |
| № вар. | Начальные условия | Граничные условия | D |
| 9 |  |  | 1 |

**Явная схема**

**Волновое уравнение:**

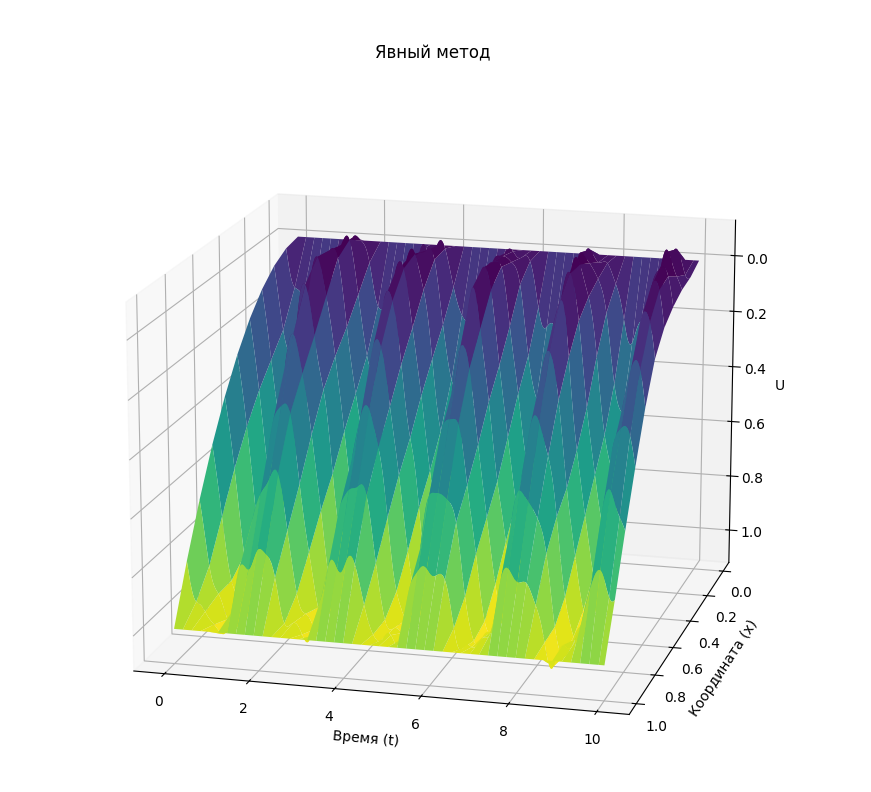
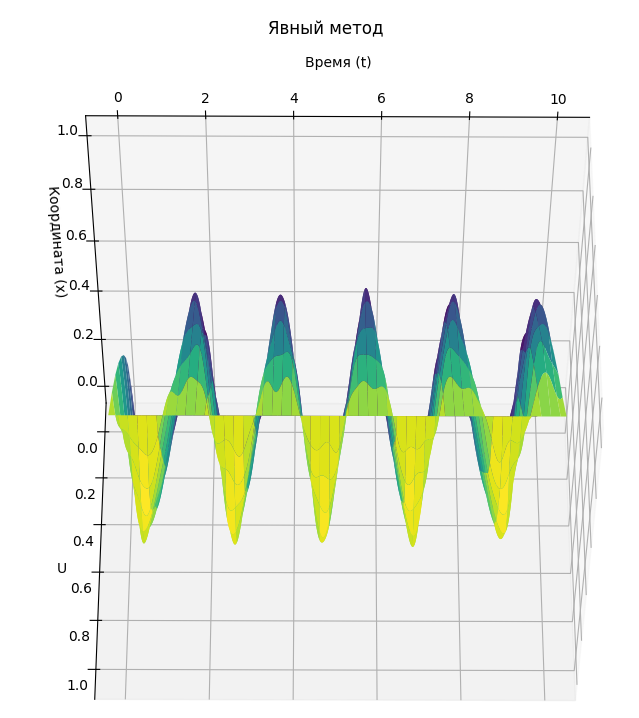
Для вычисления будем использовать:

После преобразований:

Используя производную:  получим:

Имеем условие сходимости:

При условии , h > 0, что при невозможно следовательно используем



**Неявная схема**

Обозначим λ=, перенесем все члены, содержащие , в левую часть, а остальные — в правую, получим:

После преобразований:

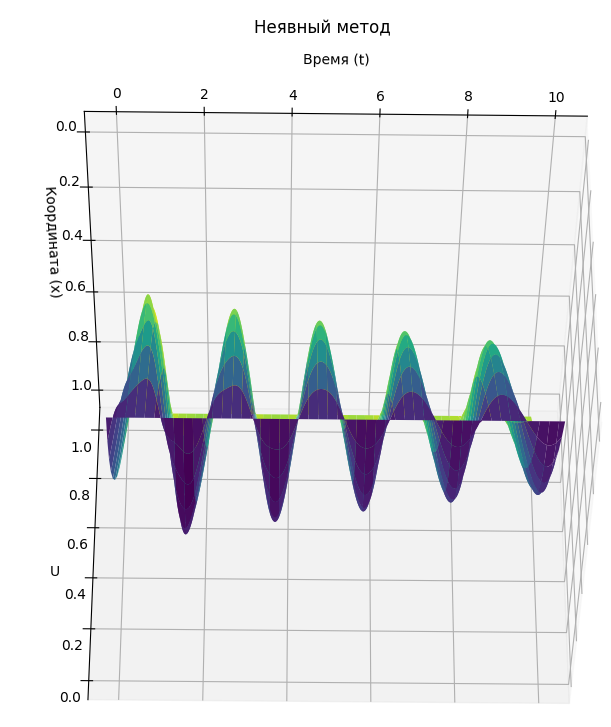
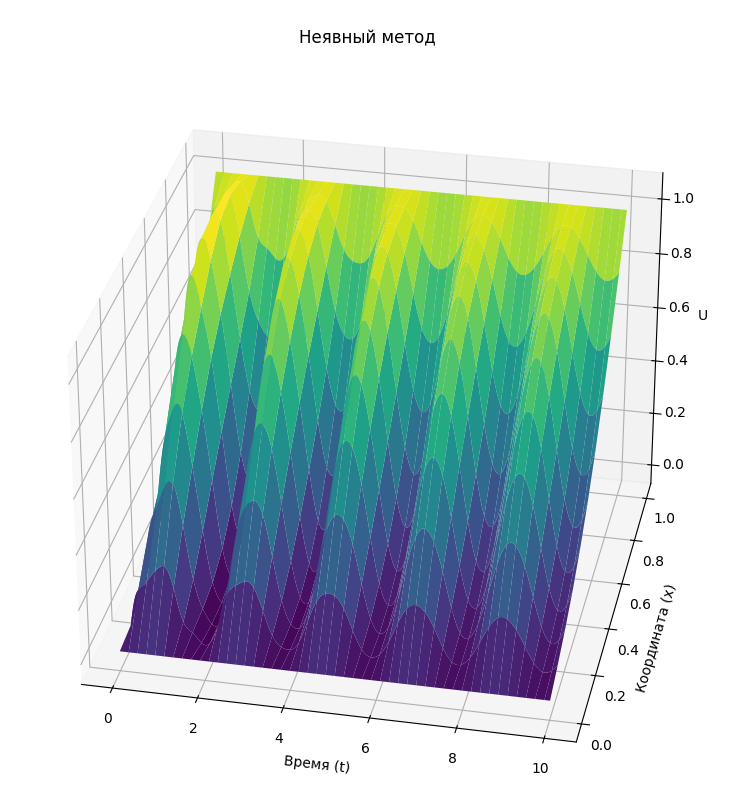
Используем метод прогонки:

Используя это соотношение, выразим и через и подставим в уравнение

, получим:

Это соотношение будет выполняться независимо от решения, при условии:

;



перенесем все члены, зависящие от , в левую часть, а остальные — в правую, получим:

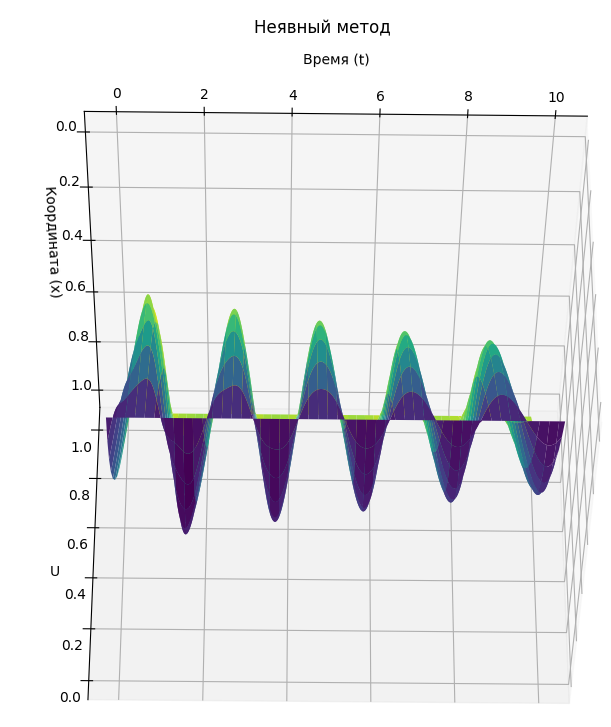
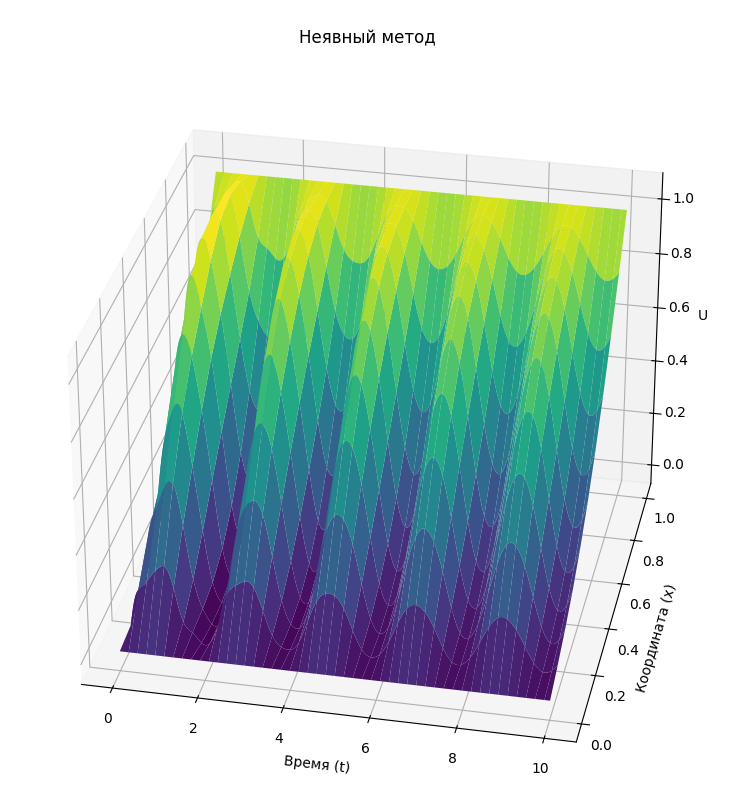
Где , Это уравнение связывает значения на новом временном слое с уже известными значениями на слоях j и j−1.

Используем метод прогонки:

Используя это соотношение, выразим и через и подставим в уравнение

, получим:

Это соотношение будет выполняться независимо от решения, если потребовать: ;



**Приложения**

import numpy as np

import matplotlib.pyplot as plt

# Определение начальных и граничных условий

def initial\_condition(x):

    return x\*\*2

def boundary\_condition\_left():

    return 0

def boundary\_condition\_right():

    return 1

def initial\_velocity():

    return 1

# Параметры сетки

h = 0.1

tau = 0.01

x\_points = int(1 / h) + 1

t\_points = int(10 / tau) + 1

l\_coef = (tau / h)\*\*2

# Функция для явного метода

def explicit\_method():

    u = np.zeros((x\_points, t\_points))

    # Задание начальных условий

    for i in range(x\_points):

        x = i \* h

        u[i, 0] = initial\_condition(x)

        u[i, 1] = u[i, 0] + tau \* initial\_velocity()

    # Граничные условия

    u[0, :] = boundary\_condition\_left()

    u[-1, :] = boundary\_condition\_right()

    # Итерации по времени

    for j in range(1, t\_points - 1):

        for i in range(1, x\_points - 1):

            u[i, j + 1] = (

                2 \* (1 - l\_coef) \* u[i, j] +

                l\_coef \* (u[i + 1, j] + u[i - 1, j]) -

                u[i, j - 1]

            )

    return u

# Функция для неявного метода

def implicit\_method():

    u = np.zeros((x\_points, t\_points))

    # Задание начальных условий

    for i in range(x\_points):

        x = i \* h

        u[i, 0] = initial\_condition(x)

        u[i, 1] = u[i, 0] + tau \* initial\_velocity()

    # Граничные условия

    u[0, :] = boundary\_condition\_left()

    u[-1, :] = boundary\_condition\_right()

    # Итерации по времени

    for j in range(1, t\_points - 1):

        a = np.full(x\_points, -l\_coef)

        b = np.full(x\_points, 1 + 2 \* l\_coef)

        c = np.full(x\_points, -l\_coef)

        d = np.zeros(x\_points)

        for i in range(1, x\_points - 1):

            d[i] = 2 \* u[i, j] - u[i, j - 1]

        d[1] += l\_coef \* u[0, j]

        d[-2] += l\_coef \* u[-1, j]

        # Прогонка (метод трехдиагональной матрицы)

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

            factor = a[i] / b[i - 1]

            b[i] -= factor \* c[i - 1]

            d[i] -= factor \* d[i - 1]

        u[-2, j + 1] = d[-2] / b[-2]

        for i in range(x\_points - 3, 0, -1):

            u[i, j + 1] = (d[i] - c[i] \* u[i + 1, j + 1]) / b[i]

    return u

# Построение графиков

for method, title in [(explicit\_method, "Явный метод"), (implicit\_method, "Неявный метод")]:

    u = method()

    x = np.linspace(0, 1, x\_points)

    t = np.linspace(0, 10, t\_points)

    x\_grid, t\_grid = np.meshgrid(x, t, indexing='ij')

    fig = plt.figure()

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

    ax.plot\_surface(t\_grid, x\_grid, u, cmap='viridis')

    ax.set\_title(title)

    ax.set\_xlabel('Время (t)')

    ax.set\_ylabel('Координата (x)')

    ax.set\_zlabel('U')

    plt.show()