Московский авиационный институт

(Национальный исследовательский университет)

Институт №8 «Компьютерные науки и прикладная математика»

**Отчет по лабораторной работе №6**

по курсу «Численные методы»

Вариант - 2

**Выполнила студентка**: Бондарева Е. Е.

**Группа:** М8О-405Б-21

# Преподаватель: Демидова О. Л.

Оценка: !

Дата: 25.10.2024\_\_\_\_\_\_ !

Москва, 2024

**Лабораторная работа №6**

**Задание:**

Используя явную схему крест и неявную схему, решить начально-краевую задачу для дифференциального уравнения гиперболического типа. Аппроксимацию второго начального условия произвести с первым и со вторым порядком. Осуществить реализацию трех вариантов аппроксимации граничных условий, содержащих производные: двухточечная аппроксимация с первым порядком, трехточечная аппроксимация со вторым порядком, двухточечная аппроксимация со вторым порядком. В различные моменты времени вычислить погрешность численного решения путем сравнения результатов с приведенным в задании аналитическим решением . Исследовать зависимость погрешности от сеточных параметров .

, ,



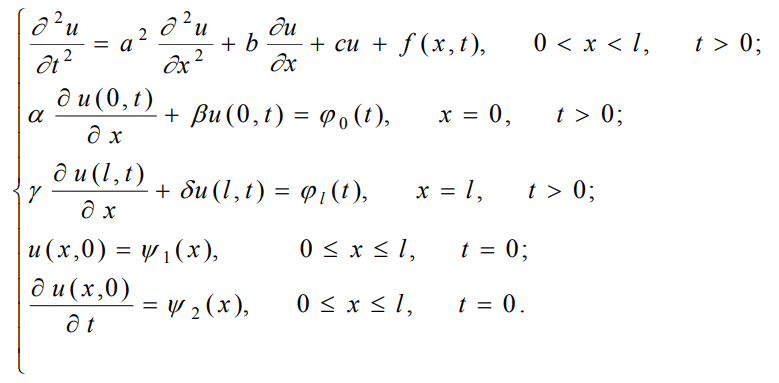
,

.

Аналитическое решение: 

**Теоретические сведения:**

Третья начально-краевая задача имеет вид:



В данной работе:

= 0

= 0

b = 0, c = 0, f(x,t) = 0

**Явная конечно-разностная схема «Крест»**

Будем решать задачу на заданном промежутке от 0 до 𝑙 по координате 𝑥 и на промежутке от 0 до заданного параметра 𝑇 по времени 𝑡. Рассмотрим конечно-разностную схему решения краевой задачи на сетке с граничными параметрами 𝑙, 𝑇 и параметрами насыщенности сетки 𝑁, 𝐾. Тогда размер шага по каждой из координат определяется:

Действуем способом, аналогичным тому, что применялся в предыдущей лабораторной работе: задаем пространственно-временную сетку и аппроксимируем производные в уравнении. Получаем явную конечно-разностную схему:

Двухточечная аппроксимация с первым порядком точности:

В результате переход на новый временной слой представляется следующим алгоритмом:

где , , .

Таким образом, получается, что:

В начальный момент времени значения определяются точно:

Воспользуемся аппроксимацией первого порядка по времени:

Явная схема условно устойчива с условием

**Неявная конечно-разностная схема**

Для неявной схемы имеем СЛАУ, которая опять же решается прогонкой, так как полученная матрица является трёхдиагональной.

В начальный момент времени значения определяются точно:

Воспользуемся аппроксимацией первого порядка по времени:

Граничные условия:

**Код программы:**

import numpy as np, matplotlib.pyplot as plt

def analyt\_func(x, t, a):

    return np.sin(x - a \* t) + np.cos(x + a \* t)

def fi1(t):

    return 0

def fi2(t):

    return 0

def psi1(x):

    return np.sin(x) + np.cos(x)

def psi2(x, a):

    return -a \*(np.sin(x) + np.cos(x))

def run\_through(a, b, c, d, s):

    P = np.zeros(s + 1)

    Q = np.zeros(s + 1)

    P[0] = -c[0] / b[0]

    Q[0] = d[0] / b[0]

    k = s - 1

    for i in range(1, s):

        P[i] = -c[i] / (b[i] + a[i] \* P[i - 1])

        Q[i] = (d[i] - a[i] \* Q[i - 1]) / (b[i] + a[i] \* P[i - 1])

    P[k] = 0

    Q[k] = (d[k] - a[k] \* Q[k - 1]) / (b[k] + a[k] \* P[k - 1])

    x = np.zeros(s)

    x[k] = Q[k]

    for i in range(s - 2, -1, -1):

        x[i] = P[i] \* x[i + 1] + Q[i]

    return x

def explicit(K, t, tau, h, x, a):

    N = len(x)

    U = np.zeros((K, N))

    t += tau

    sigma = a \* tau\*\*2 / h\*\*2

    for j in range(N):

        U[0, j] = psi1(x[j])

        U[1][j] = psi1(x[j]) + psi2(x[j], a) \* tau

    for k in range(1, K - 1):

        t += tau

        for j in range(1, N - 1):

            U[k + 1, j] = sigma \* U[k, j + 1] + 2 \* (1 - sigma) \* U[k, j] + sigma \* U[k, j - 1] - U[k - 1, j]

        U[k + 1, 0] = 1 / (h + 1) \* U[k + 1, 1] + fi1(t) \* h

        U[k + 1, N - 1] = U[k + 1, N - 2] / (1 - h)

    return U

def implicit(K, t, tau, h, x, a):

    N = len(x)

    U = np.zeros((K, N))

    t += tau

    sigma = (a\*\*2 \* tau \*\*2) / (h \*\*2)

    for j in range(N):

        U[0, j] = psi1(x[j])

        U[1][j] = psi2(x[j], a) \* tau + psi1(x[j])

    for k in range(1, K - 1):

        a = np.zeros(N)

        b = np.zeros(N)

        c = np.zeros(N)

        d = np.zeros(N)

        t += tau

        for j in range(1, N - 1):

            a[j] = - 1/ h \*\* 2

            b[j] = 2/ h \*\* 2 + 1/ tau

            c[j] = - 1/ h \*\* 2

            d[j] = 1/tau \* U[k, j]

        a[0] = 0

        b[0] = 2/h + h/tau

        c[0] = -2/h

        d[0] = h/tau \* U[k, 0] - fi1(t)\*2

        a[N - 1] = -2/h

        b[N - 1] = -2 / h + h/tau - 2

        c[N - 1] = 0

        d[N - 1] = h / tau \* U[k - 1, N - 1] + fi2(t)

        u\_new = run\_through(a, b, c, d, N)

        for i in range(N):

            U[k + 1, i] = u\_new[i]

    return U

def main(N, K, time):

    a = 1

    h = np.pi / N

    h2 = h / 2

    h3 = h / 4

    tau = time / K

    x = np.arange(0, np.pi + h / 2 - 1e-4, h)

    x2 = np.arange(0, np.pi + h2 / 2 - 1e-4, h2)

    x3 = np.arange(0, np.pi + h3 / 2 - 1e-4, h3)

    T = np.arange(0, time, tau)

    t = 0

    print("Лабораторная работа №6")

    while (1):

        print("Выберите метод:\n"

              "1) Явная конечно-разностная схема\n"

              "2) Неявная конечно-разностная схема\n"

              "0) Выход")

        method = int(input())

        if method == 0:

            break

        else:

            if method == 1:

                dt = 10

                if tau / h \*\* 2 <= 1:

                    U = explicit(K, t, tau, h, x, a)

                    U2 = explicit(K, t, tau, h2, x2, a)

                    U3 = explicit(K, t, tau, h3, x3, a)

                    U\_analytic = analyt\_func(x, T[dt], a)

                    U\_analytic2 = analyt\_func(x2, T[dt], a)

                    U\_analytic3 = analyt\_func(x3, T[dt], a)

                    plt.title("Метод явной схемы")

                    plt.plot(x, U\_analytic, label="Точное решение", color="red")

                    plt.scatter(x, U[dt, :], label="Численное решение")

                    plt.xlabel("x")

                    plt.ylabel("y")

                    plt.axis()

                    plt.grid()

                    plt.legend()

                    plt.show()

                    error = abs(U\_analytic - U[dt, :])

                    er1 = max(error)

                    error2 = abs(U\_analytic2 - U2[dt, :])

                    er2 = max(error2)

                    error3 = abs(U\_analytic3 - U3[dt, :])

                    er3 = max(error3)

                    print(h, h2, h3)

                    plt.title("График ошибки от h")

                    errors = [er1, er2, er3]

                    x\_h = [h, h2, h3]

                    plt.plot(x\_h, errors)

                    plt.scatter(x\_h, errors, label='', zorder=6, c='r')

                    plt.grid()

                    plt.show()

                else:

                    break

            elif method == 2:

                dt = 10

                U = implicit(K, t, tau, h, x, a)

                U2 = implicit(K, t, tau, h2, x2, a)

                U3 = implicit(K, t, tau, h3, x3, a)

                U\_analytic = analyt\_func(x, T[dt], a)

                U\_analytic2 = analyt\_func(x2, T[dt], a)

                U\_analytic3 = analyt\_func(x3, T[dt], a)

                plt.title("Метод неявной схемы")

                plt.plot(x, U\_analytic, label="Точное решение", color="red")

                plt.scatter(x, U[dt, :], label="Численное решение")

                plt.xlabel("x")

                plt.ylabel("y")

                plt.axis()

                plt.grid()

                plt.legend()

                plt.show()

                error = abs(U\_analytic - U[dt, :])

                er1 = max(error)

                error2 = abs(U\_analytic2 - U2[dt, :])

                er2 = max(error2)

                error3 = abs(U\_analytic3 - U3[dt, :])

                er3 = max(error3)

                print(h, h2, h3)

                plt.title("График ошибки от h")

                errors = [er1, er2, er2]

                x\_h = [h, h2, h3]

                plt.plot(x\_h, errors)

                plt.scatter(x\_h, errors, label='', zorder=6, c='r')

                plt.grid()

                plt.show()

    return 0

N = 25

K = 30000

time = 3

main(N, K, time)

**Результат работы программы:**

Лабораторная работа №6

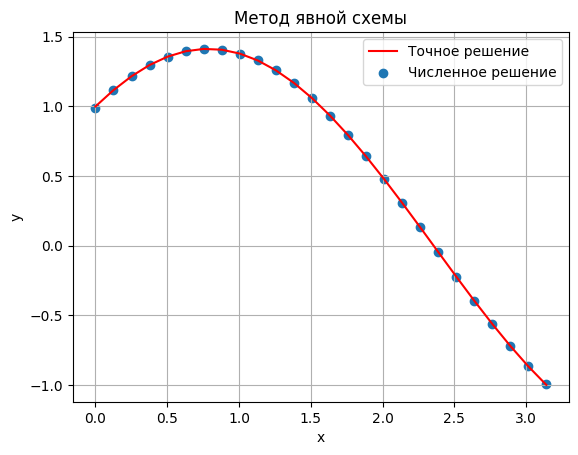
Выберите метод:

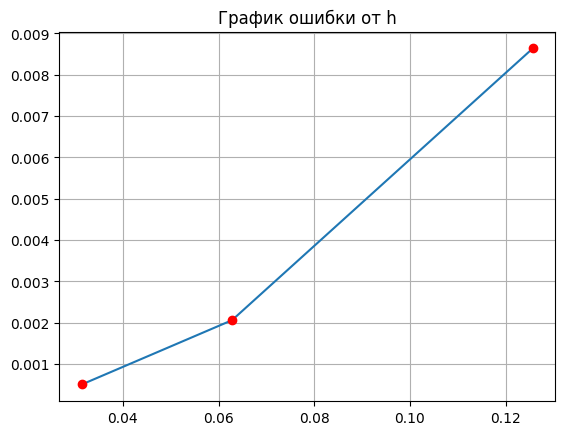
1) Явная конечно-разностная схема

2) Неявная конечно-разностная схема

0) Выход

1

****

****

0.12566370614359174 0.06283185307179587 0.031415926535897934

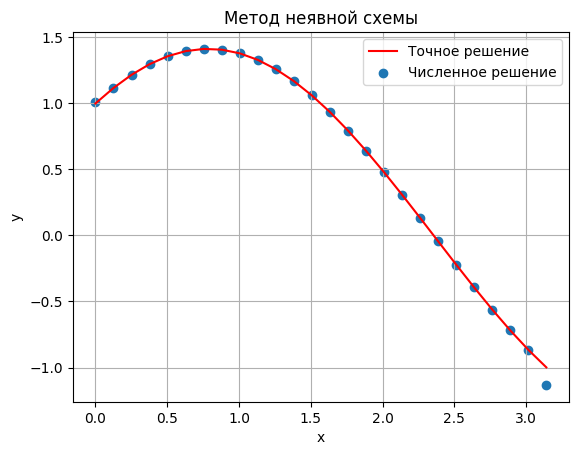
Выберите метод:

1) Явная конечно-разностная схема

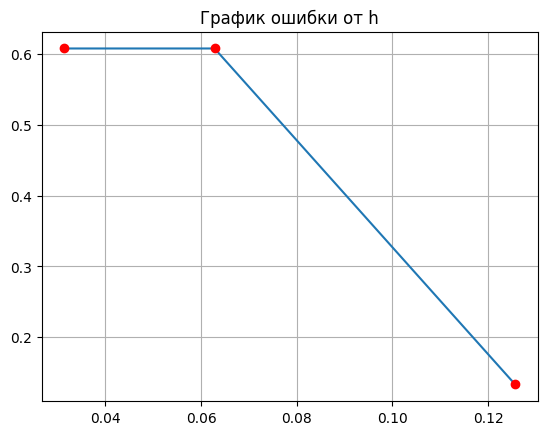
2) Неявная конечно-разностная схема

0) Выход

2

****

0.12566370614359174 0.06283185307179587 0.031415926535897934

****

Выберите метод:

1) Явная конечно-разностная схема

2) Неявная конечно-разностная схема

0) Выход

0

**Вывод:**

В результате выполнения работы была решена Начально-краевая задача для дифференциального уравнения гиперболического типа с использованием явной схемы крест и неявной конечно-разностной схемы; было проведено исследование зависимости погрешности.

Явная схема оказалась эффективнее. Исходя из графика погрешности можно сказать, что отклонение от искомого решения неявным методом в целом больше, чем у явного. При этом не стоит забывать, что явный метод применим только при выполнении критерия устойчивости, что накладывает значительные ограничения на построение численного решения.