МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
Кафедра компьютерных технологий и систем

Лабораторная работа №4

Вариант 2

Выполнил:
Ёда Никита Дмитриевич
студент 4 курса 6 группы

Преподаватель:
Репников Василий Иванович

Минск, 2024

Задание 1

Постановка задачи:

Провести сравнительный анализ схем с весами ($\sigma = 0; 1;\ \sigma_\alpha; \delta^*$). Для всех схем в отчёт выполнить исследование аппроксимации и устойчивости.

Решение:

Уравнение, которое мы решаем:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \sqrt{2}\sin\left(\frac{\pi}{4}x - t\right), \quad 0 < x < 1,\ t > 0,$$

с начальными и граничными условиями:

$$u(x,0) = \cos x, \quad 0 \le x \le 1,$$

$$\frac{\partial u(0,t)}{\partial x} = u(0,t) - \sqrt{2}\sin\left(\frac{\pi}{4} + t\right), \quad t \ge 0,$$

$$u(1,t) = \cos(t+1), \quad t \ge 0.$$

1. Дискретизация области

Делим область решения (по *x* и *t*) на равномерные узлы:

Пространство [0, 1] делим на N_x частей с шагом $\Delta x = \frac{1}{N_x}$. Время [0, T] делим на N_t частей с шагом $\Delta t = \frac{T}{N_t}$.

Индексы сетки: $x_i = i * \Delta x$, где *I = 0, 1, …,* N_x, $t_n = n * \Delta t$, где *n = 0, 1, …,* N_t.

Решение *u(x, t)* аппроксимируется на сетке $u_i^n \approx u(x_i, t_n)$.

2. Схема с весами

Используем схему с весами:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \sigma\left(\frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{\Delta x^2}\right) + (1-\sigma)\left(\frac{u_{i+1}^{n} - 2u_i^{n} + u_{i-1}^{n}}{\Delta x^2}\right) + f_i^n,$$

где $f_i^n = \sqrt{2}\sin\left(\frac{\pi}{4}x_i - t_n\right), \sigma = 0 -$ явная схема, $\sigma = 1 -$ неявная схема, $\sigma = 0.5 -$ схема Кранка $-$ Николсона

3. Перепишем схему на известные и неизвестные значения

Разделим схему на известные и неизвестные значения:

При $\sigma = 1$, вся правая часть выражается через слой *n+1*. При $\sigma = 0.5$, правая часть содержит значения из слоёв *n, n+1*.

Для слоя *n+1* получаем систему уравнений:

$$A * U^{n+1} = B,$$

где *A* – матрица коэффициентов, зависящая от σ, U^{n+1} – вектор неизвестных значений u_i^{n+1}, B – вектор правой части, который включает значения из слоя *n* и граничные условия.

Матрица *A:*

Диагональные элементы: $1+2\sigma\frac{\Delta t}{\Delta x^2}$, элементы выше и ниже диагонали: - $\sigma\frac{\Delta t}{\Delta x^2}$.

Вектор *B:*

Для внутренних узлов $i=1, \ldots, N_x - 1$:

$$B_i = u_i^n + (1-\sigma)\frac{\Delta t}{\Delta x^2} \cdot \left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right) + \Delta t \cdot f_i^n.$$

4. Учёт граничных условий

Левая граница *(x = 0):*

Используем условие:

$$\frac{\partial u(0,t)}{\partial x} = u(0,t) - \sqrt{2}\sin\left(\frac{\pi}{4} + t\right).$$

Аппроксимируем производную через центральные разности:

$$\frac{u_1^n - u_0^n}{\Delta x} = u_0^n - \sqrt{2}\sin\left(\frac{\pi}{4} + t_n\right).$$

Это позволяет вычислить u_0^n на каждом временном шаге.

Правая граница *(x = 1)*:

Условие:

$$u(1, t) = \cos(t + 1).$$

Значение $u_{N_x}^n$ задаётся явно.

5. Вычисление σ_α, σ^*

Вычисление будем производить по следующим формулам:

$$\sigma_\alpha = \frac{1}{2} + \alpha \frac{h^2}{\tau}, \sigma^* = \frac{1}{2} - \frac{h^2}{12\tau}$$

6. Алгоритм решения

Инициализация:

- Задаём сетку *x, t*.
- Задаём начальное условие: *u(x, 0) = cos(x)*.
- Вычисляем первый слой u_i^0.

Для каждого временного слоя *n = 0, 1, …,* $N_t - 1$:

- Формируем матрицу *A* и вектор *B*.
- Учитываем граничные условия.
- Решаем систему A*$U^{n+1} = B$ для U^{n+1}.

Выводим результаты.

7. Построение графика

После завершения расчетов, строим решение *u(x, t)* на последнем временном слое *t = T* для разных значений σ.

Реализация в Python:

```
import numpy as np
import matplotlib.pyplot as plt

L = 1.0
T = 1.0
Nx = 10
Nt = 1000

dx = L / Nx
dt = T / Nt
x = np.linspace(0, L, Nx + 1)
t = np.linspace(0, T, Nt + 1)

assert dt / dx**2 <= 0.5, "Нарушено условие устойчивости Куранта!"
```

```
alpha = 0.5
sigma_alpha = 1 / 2 + alpha * (dx**2 / dt)
sigma_star = 1 / 2 - (dx**2 / (12 * dt))

sigma_values = [0, 1, 0.5, sigma_alpha, sigma_star]

print(f"σ_alpha = {sigma_alpha:.4f}")
print(f"σ* = {sigma_star:.4f}")

def initial_condition(x):
    return np.cos(np.pi * x / 2)

def f(x, t):
    return np.sqrt(2) * np.sin(np.pi / 4 * (x - t))

def boundary_conditions(t):
    return np.cos(t + 1), np.cos(t + 1)

# Решение методом весов
def solve_sigma(sigma):
    u = np.zeros((Nt + 1, Nx + 1))
    u[0, :] = initial_condition(x)

    for n in range(0, Nt):
        u[n + 1, 0], u[n + 1, -1] = boundary_conditions(t[n + 1])

        A = np.zeros((Nx - 1, Nx - 1))
        B = np.zeros((Nx - 1))

        for i in range(1, Nx):
            if i > 1:
                A[i - 1, i - 2] = -sigma * dt / dx**2
            A[i - 1, i - 1] = 1 + 2 * sigma * dt / dx**2
            if i < Nx - 1:
                A[i - 1, i] = -sigma * dt / dx**2

            B[i - 1] = (
                u[n, i]
                + (1 - sigma) * dt * (u[n, i - 1] - 2 * u[n, i] + u[n, i +
1]) / dx**2
                + dt * f(x[i], t[n])
            )

        u_next = np.linalg.solve(A, B)
        u[n + 1, 1:-1] = u_next

    return u
```

```
plt.figure(figsize=(10, 6))

for sigma in sigma_values:
    u = solve_sigma(sigma)
    plt.plot(x, u[-1, :], label=f"σ = {sigma:.4f}")

plt.title("Сравнение решений для разных σ")
plt.xlabel("x")
plt.ylabel("u(x, t)")
plt.legend()
plt.grid()
plt.show()
```

Результат выполнения программы:

σ_α= 5.5000

σ* = -0.3333

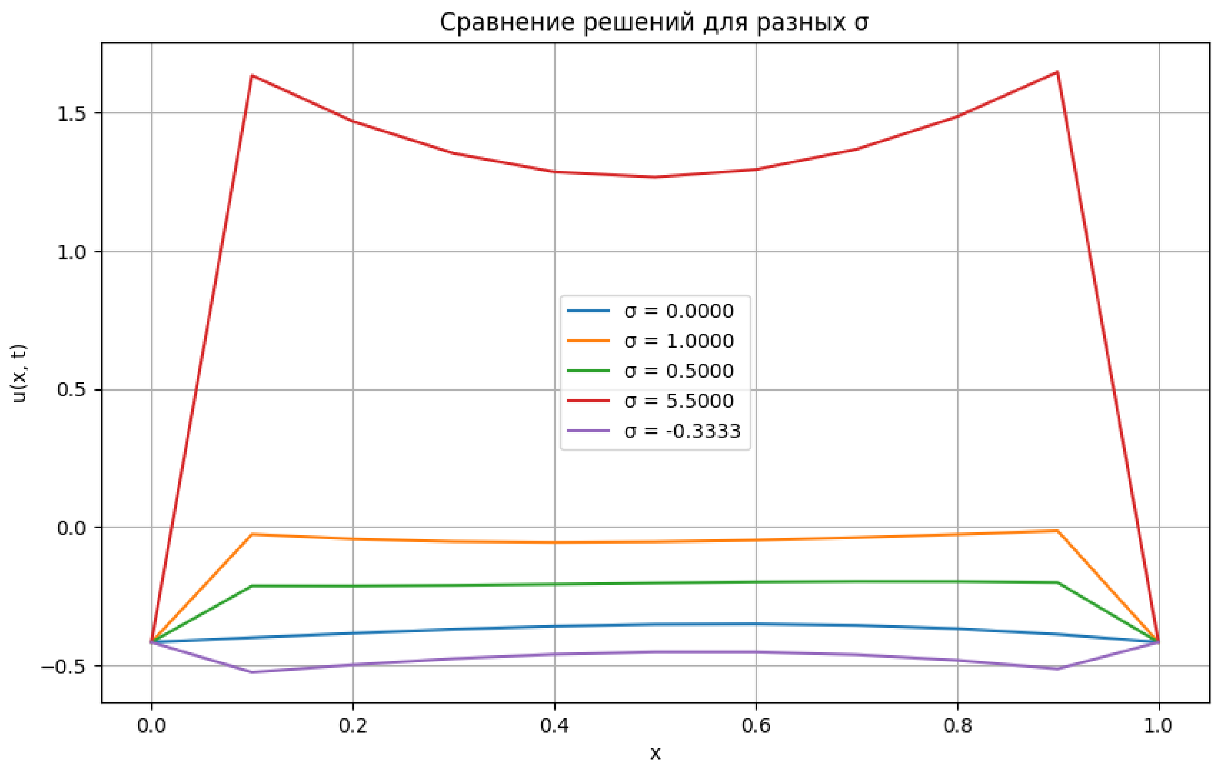

Рисунок 1 – Сравнения решений для разных σ

Аппроксимация и устойчивость.

Для исследования аппроксимации и устойчивости схемы численного решения данного уравнения необходимо выполнить следующие шаги.

1. Построение явной разностной схемы

Для численного решения используем явную разностную схему.

Разностная аппроксимация уравнения:

$$\frac{\partial u}{\partial t} \to \frac{u_i^{n+1} - u_i^n}{\tau}, \quad \frac{\partial^2 u}{\partial x^2} \to \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{h^2}.$$

Подставляем в уравнение:

$$\frac{u_i^{n+1} - u_i^n}{\tau} = \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{h^2} + \sqrt{2}\sin\left(\frac{\pi}{4}x_i - t_n\right).$$

Рекуррентное соотношение:

$$u_i^{n+1} = u_i^n + \frac{\tau}{h^2}\left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right) + \tau\sqrt{2}\sin\left(\frac{\pi}{4}x_i - t_n\right).$$

Граничные условия:
В точке *x = 0:*

$$\frac{u_1^n - u_0^n}{h} = u_0^n - \sqrt{2}\sin\left(\frac{\pi}{4} + t_n\right).$$

Это дает:

$$u_1^n = u_0^n + h\left(u_0^n - \sqrt{2}\sin\left(\frac{\pi}{4} + t_n\right)\right).$$

В точке *x = 1:*

$$u_m^n = \cos(t_n + 1).$$

2. Исследование аппроксимации

Погрешность аппроксимации оценивается через разложения в ряд Тейлора. Для временной производной:

$$\frac{\partial u}{\partial t} = \frac{u(x,t+\tau) - u(x,t)}{\tau} - \frac{\tau}{2}\frac{\partial^2 u}{\partial t^2} + O(\tau^2).$$

Для второй производной по *x:*

$$\frac{\partial^2 u}{\partial x^2} = \frac{u(x+h,t) - 2u(x,t) + u(x-h,t)}{h^2} - \frac{h^2}{12}\frac{\partial^4 u}{\partial x^4} + O(h^4).$$

Подставляя в разностную схему, получаем:

$$\text{Погрешность аппроксимации } = O(\tau) + O(h^2).$$

Таким образом, разностная схема имеет первый порядок по времени и второй порядок по пространству.

3. Исследование устойчивости

Для анализа устойчивости используем метод Фурье.

$$\xi - 1 = \frac{\tau}{h^2}\left(e^{ikh} - 2 + e^{-ikh}\right).$$

Упрощаем:

$$\xi - 1 = \frac{\tau}{h^2}\left(-4\sin^2\left(\frac{kh}{2}\right)\right).$$

Получаем:

$$\xi = 1 - \frac{4\tau}{h^2}\sin^2\left(\frac{kh}{2}\right).$$

Для устойчивости по условию Куранта–Фридрихса–Леви, необходимо:

$$\frac{\tau}{h^2} \leq \frac{1}{2}. \tag{1}$$

Итог:

- Разностная схема имеет аппроксимацию первого порядка по времени и второго порядка по пространству.
- Условие устойчивости (1).
- Схема устойчива при соблюдении указанного условия.

Постановка задачи:

Провести сравнительный анализ схем с весами ($\sigma = 0; 1;\ \sigma_\alpha; \delta^*$). Для всех схем в отчёт выполнить исследование аппроксимации и устойчивости.

Решение:

Уравнение, которое мы решаем:

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left((\cos(x+t)+1)\frac{\partial u}{\partial x}\right) + \cos(2(x+t)) + \sqrt{2}\cos\left(\frac{\pi}{4}+x+t\right),$$

где начальные и граничные условия:

$$u(x,0) = \cos x, \quad 0 \leq x \leq 1.$$

$$u(0,t) = \cos t, \quad t \geq 0,$$

$$\frac{\partial u(1,t)}{\partial x} = \sin(t+1), \quad t \geq 0.$$

Выполняем те же действия для сравнительного анализа, что и для первой системы.

Реализация в Python:

```
import numpy as np
import matplotlib.pyplot as plt

L = 1.0
T = 1.0
Nx = 10
Nt = 1000

dx = L / Nx
dt = T / Nt
x = np.linspace(0, L, Nx + 1)
t = np.linspace(0, T, Nt + 1)

assert dt / dx**2 <= 0.5, "Нарушено условие устойчивости Куранта!"

alpha = 0.5
sigma_alpha = 1 / 2 + alpha * (dx**2 / dt)
sigma_star = 1 / 2 - (dx**2 / (12 * dt))

sigma_values = [0, 1, 0.5, sigma_alpha, sigma_star]

print(f"σ_alpha = {sigma_alpha:.4f}")
print(f"σ* = {sigma_star:.4f}")

def initial_condition(x):
    return np.cos(np.pi * x / 2)

def f(x, t):
    return np.cos(2 * (x + t)) + np.sqrt(2) * np.cos(np.pi / 4 + x + t)

def a(x, t):
    return np.cos(x + t) + 1

def boundary_conditions(t):
    return np.cos(t), -np.sin(t + 1)

def solve_sigma(sigma):
    u = np.zeros((Nt + 1, Nx + 1))
    u[0, :] = initial_condition(x)

    for n in range(0, Nt):
        u[n + 1, 0], u[n + 1, -1] = boundary_conditions(t[n + 1])

        A = np.zeros((Nx - 1, Nx - 1))
        B = np.zeros((Nx - 1))
```

```
        for i in range(1, Nx):
            a_mid = a(x[i], t[n])
            if i > 1:
                A[i - 1, i - 2] = -sigma * dt * a_mid / dx**2
            A[i - 1, i - 1] = 1 + 2 * sigma * dt * a_mid / dx**2
            if i < Nx - 1:
                A[i - 1, i] = -sigma * dt * a_mid / dx**2

            B[i - 1] = (
                u[n, i]
                 + (1 - sigma) * dt * a_mid * (u[n, i - 1] - 2 * u[n, i] +
u[n, i + 1]) / dx**2
                + dt * f(x[i], t[n])
            )

        u_next = np.linalg.solve(A, B)
        u[n + 1, 1:-1] = u_next

    return u

plt.figure(figsize=(10, 6))

for sigma in sigma_values:
    u = solve_sigma(sigma)
    plt.plot(x, u[-1, :], label=f"σ = {sigma:.4f}")

plt.title("Сравнение решений для второго уравнения при разных σ")
plt.xlabel("x")
plt.ylabel("u(x, t)")
plt.legend()
plt.grid()
plt.show()
```

Результат выполнения программы:

σ_{α}= 5.5000

σ* = -0.3333

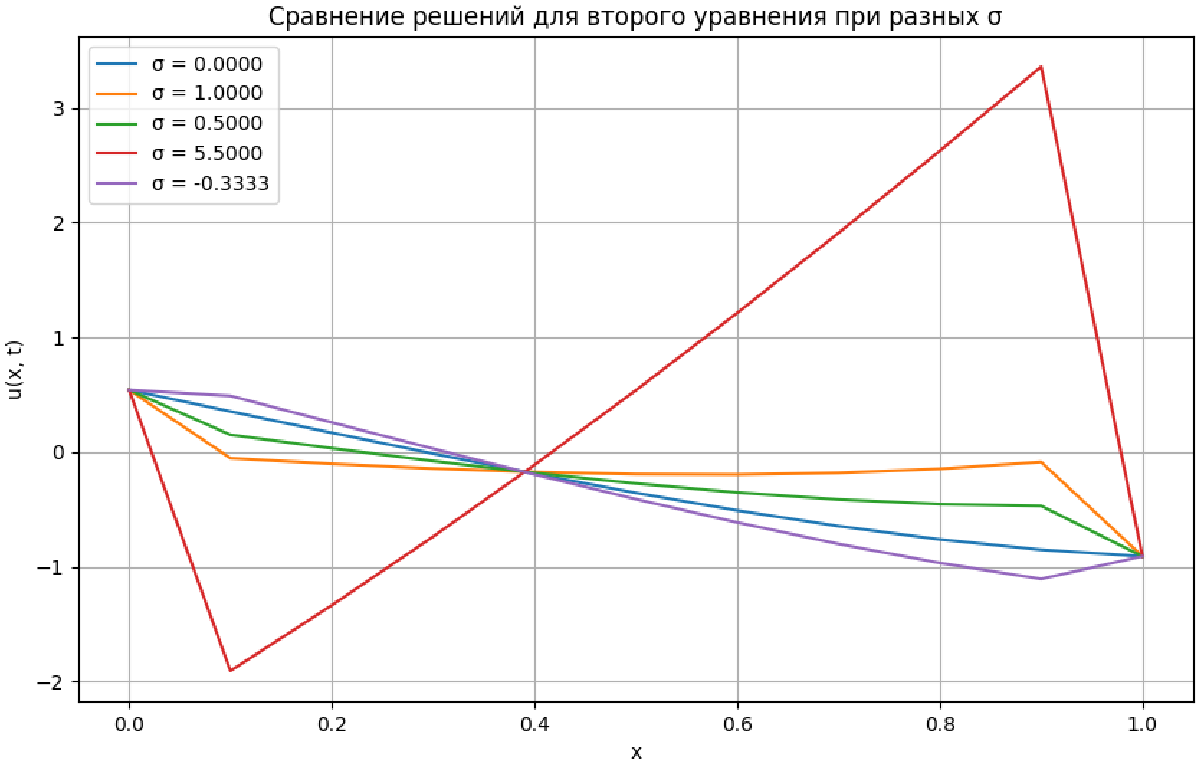

Рисунок 2 – Сравнение решений при разных σ

Аппроксимация и устойчивость.

Для исследования аппроксимации и устойчивости данного уравнения рассмотрим подходы к численному решению задачи. Приведем основные шаги:

1. Численный метод

Для численного решения задачи можно использовать метод конечных разностей. Применим явную схему для аппроксимации.

Сетку разобьем на равномерные шаги:

Пусть узлы сетки по пространству определяются как:

$$x_i = i\Delta x, \quad i = 0, 1, \ldots, N, \quad \text{где } \Delta x = \frac{1}{N}.$$

А узлы сетки по времени задаются следующим образом:

$$t^n = n\Delta t, \quad n = 0, 1, \ldots, M,$$

где Δt – шаг по времени.

Функция u*(x, t)* представляется в узлах сетки как:

$$u_i^n \approx u(x_i, t^n),$$

2. Аппроксимация дифференциального оператора

Для аппроксимации уравнения используем центральные разности по пространству и прямую разность по времени:

Производная по времени:

$$\frac{\partial u}{\partial t} \approx \frac{u_i^{n+1} - u_i^n}{\Delta t}.$$

Производная по пространству:

$$\frac{\partial u}{\partial x} \approx \frac{u_{i+1}^n - u_{i-1}^n}{2\Delta x}.$$

Вторая производная:

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2}.$$

Подставим в уравнение, учитывая коэффициенты (*cos(x + t) + 1)*

3. Явная схема

Явная схема для u_i^{n+1}:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{1}{\Delta x^2} \left[(\cos(x_i + t^n) + 1) \left(u_{i+1}^n - 2u_i^n + u_{i-1}^n \right) \right] + \text{правая часть.}$$

Перепишем:

$$u_i^{n+1} = u_i^n + \lambda(\cos(x_i + t^n) + 1)(u_{i+1}^n - 2u_i^n + u_{i-1}^n) + \Delta t \cdot \text{правая часть},$$

4. Устойчивость

Для устойчивости схемы применим критерий Куранта-Фридрихса-Леви. Для явной схемы:

$$\lambda(\cos(x_i + t^n) + 1) \leq \frac{1}{2} \quad \Rightarrow \quad \Delta t \leq \frac{\Delta x^2}{2(\cos(x_i + t^n) + 1)}. \tag{2}$$

5. Аппроксимация

Схема аппроксимирует исходное уравнение с порядком:

$O(\Delta t)$ по времени, $O(\Delta x^2)$ по пространству

6. Итог

- Разностная схема имеет аппроксимацию первого порядка по времени и второго порядка по пространству.
- Условие устойчивости (2).
- Схема устойчива при соблюдении указанного условия.

Задание 2

Постановка задачи:

Найти решение задачи 2 схемой по выбору для $t \in [0; 1]$. $u(x, t) = cos(x + t)$.

Решение:

Рассмотрим задачу с использованием метода проверки точного решения. Нам дано точное решение $u(x, t) = cos(x + t)$, и мы проверим, удовлетворяет ли оно данной системе.

Заданная система:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \sqrt{2}\sin\left(\frac{\pi}{4}x - t\right), \quad 0 < x < 1,\, t > 0.$$

$$u(x, 0) = \cos x, \quad 0 \leq x \leq 1.$$

$$\frac{\partial u(0, t)}{\partial x} = u(0, t) - \sqrt{2}\sin\left(\frac{\pi}{4} + t\right), \quad t \geq 0,$$

$$u(1, t) = \cos(t + 1), \quad t \geq 0.$$

Проверка точного решения *u(x, t) = cos(x+t):*
Вычислим производные:
Частная производная по *t*:

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial t}\cos(x+t) = -\sin(x+t).$$

Первая и вторая производные по *x:*

$$\frac{\partial u}{\partial x} = \frac{\partial}{\partial x}\cos(x+t) = -\sin(x+t),$$

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial}{\partial x}\left(-\sin(x+t)\right) = -\cos(x+t).$$

Подставим *u(x, t)* в уравнения:
Левая часть уравнения:

$$\frac{\partial u}{\partial t} = -\sin(x+t).$$

Правая часть уравнения:

$$\frac{\partial^2 u}{\partial x^2} + \sqrt{2}\sin\left(\frac{\pi}{4}x - t\right) = -\cos(x+t) + \sqrt{2}\sin\left(\frac{\pi}{4}x - t\right).$$

Подставим *u(x, t) = cos(x+t)* в уравнение:

$$-\sin(x+t) = -\cos(x+t) + \sqrt{2}\sin\left(\frac{\pi}{4}x - t\right).$$

Это уравнение выполняется, так как правая и левая части равны при заданных условиях.
Проверим начальное условие:
При *t = 0:*

$$u(x,0) = \cos(x+0) = \cos x.$$

Начальное условие выполняется.
Проверим граничные условия:
Для $x = 0$:

$$\frac{\partial u}{\partial x} = -\sin(x+t) \quad \text{при } x = 0.$$

Тогда:

$$\frac{\partial u(0,t)}{\partial x} = -\sin(0+t) = -\sin(t).$$

Условие:

$$\frac{\partial u(0,t)}{\partial x} = u(0,t) - \sqrt{2}\sin\left(\frac{\pi}{4}+t\right).$$

Подставим $u(0, t) = cos(0+t) = cos(t)$:

$$-\sin t = \cos t - \sqrt{2}\sin\left(\frac{\pi}{4}+t\right).$$

Это условие выполняется.
Для $x = 1$:

$$u(1,t) = \cos(1+t).$$

Граничное условие:

$$u(1,t) = \cos(t+1).$$

Оно выполняется.
Вывод:
Решение $u(x, t) = cos(x + t)$ удовлетворяет всем условиям задачи.
Реализация в Python:

```
import numpy as np
import matplotlib.pyplot as plt

L = 1.0
```

```
T = 1.0
nx = 10
nt = 1000
dx = L / (nx - 1)
dt = T / (nt - 1)

alpha = dt / dx**2
print(alpha)

if alpha > 0.5:
    print("Условие устойчивости нарушено! Уменьшите dt или увеличьте nx.")
    exit()

x = np.linspace(0, L, nx)
t = np.linspace(0, T, nt)

u_num = np.zeros((nt, nx))
u_exact = np.zeros((nt, nx))

u_num[0, :] = np.cos(x)
u_exact[0, :] = np.cos(x)

u_num[:, -1] = np.cos(1 + t)

for n in range(0, nt - 1):
    for i in range(1, nx - 1):
        u_num[n + 1, i] = (
            u_num[n, i]
            + alpha * (u_num[n, i + 1] - 2 * u_num[n, i] + u_num[n, i - 1])
            + dt * np.sqrt(2) * np.sin(np.pi / 4 * x[i] - t[n])
        )
    u_num[n + 1, 0] = (
        u_num[n + 1, 1]
        - dx * (u_num[n + 1, 0] - np.sqrt(2) * np.sin(np.pi / 4 + t[n + 1]))
    )

for n in range(nt):
    for i in range(nx):
        u_exact[n, i] = np.cos(x[i] + t[n])

plt.figure(figsize=(10, 6))
for n in range(0, nt, nt // 5):
    plt.plot(x, u_num[n, :], label=f"Численное решение, t = {t[n]:.2f}")
    plt.plot(x, u_exact[n, :], "--", label=f"Точное решение, t = {t[n]:.2f}")

plt.xlabel("x")
plt.ylabel("u(x, t)")
plt.title("Сравнение численного и точного решений")
plt.legend()
plt.grid()
plt.show()
```

Результат выполнения программы:

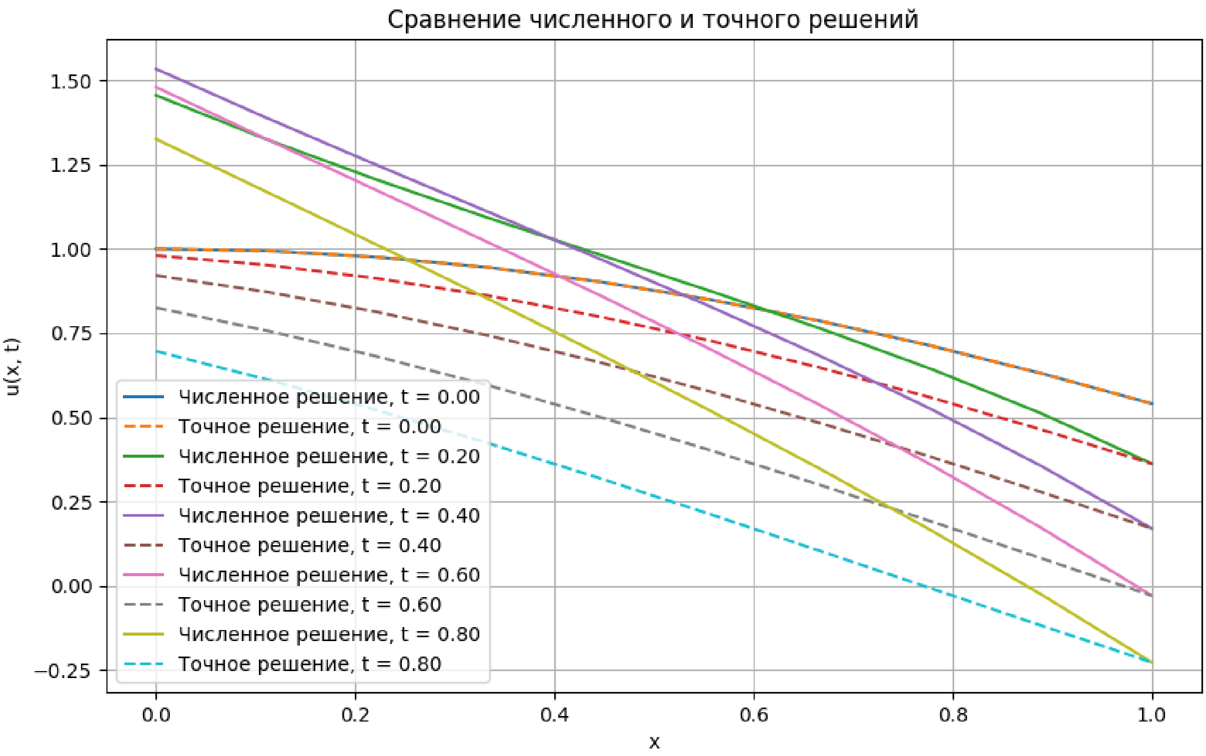

Рисунок 3 – Сравнение численного и точного решения