Московский авиационный институт
(национальный исследовательский университет)
Институт № 8 «Информационные технологии и прикладная математика»

Лабораторная работа № 5
по курсу «Численные методы»
«Численное решение уравнений с частными производными параболического типа»

Выполнил:
Гришин П. Ф.
Группа:
М8О-401Б-21
Преподаватель:
Ревизников Д. Л.

Москва, 2025

Задача

Используя явную и неявную конечно-разностные схемы, а также схему Кранка - Николсона, решить начально-краевую задачу для дифференциального уравнения параболического типа. 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 U(x, t). Исследовать зависимость погрешности от сеточных параметров τ, h.

Описание метода

Классический пример уравнения параболического типа – уравнение теплопроводности, которое в одномерном по пространству случае имеет вид:

$$\frac{\partial u}{\partial t} = a^2 \frac{\partial^2 u}{\partial x^2}, \quad 0 < x < l, \ t > 0.$$

Если на границах $x=0$ и $x=l$ заданы значения искомой функции $u(x,t)$ *в виде*

$$u(0,t) = \varphi_0(t), \quad x = 0, \quad t > 0;$$

$$u(l,t) = \varphi_l(t), \quad x = l, \quad t > 0,$$

- граничные условия первого рода.

И заданы начальные условия:

$$u(x,0) = \psi(x), \quad 0 \le x \le l, \quad t = 0$$

Также на границах могут быть заданы значения производных искомой функции по пространственной переменной, то есть граничные условия второго рода:

$$\frac{\partial u(0,t)}{\partial x} = \varphi_0(t), \quad x = 0, \quad t > 0;$$

$$\frac{\partial u(l,t)}{\partial x} = \varphi_l(t), \quad x = l, \quad t > 0,$$

или граничные условия третьего рода, то есть линейные комбинации искомой функции и ее производной по пространственной переменной:

$$\alpha \frac{\partial u(0,t)}{\partial x} + \beta u(0,t) = \varphi_0(t), \quad x = 0, \quad t > 0;$$

$$\gamma \frac{\partial u(l,t)}{\partial x} + \delta u(l,t) = \varphi_l(t), \quad x = l, \quad t > 0,$$

Для решения такой задачи применяют метод конечных разностей.

Для этого вводится понятие разностной сетки с пространственным шагом h и временным τ

$$\omega_{h\tau} = \{x_j = jh, \ j = \overline{0,N}; \ t^k = k\tau, \ k = \overline{0,K}\}$$

Также вводится понятие сеточной функции – однозначное отображение целых аргументов j, k в значения функции $u_j^k = u(x_j, t_k)$.

Вводится два временных слоя: нижний $t^k = k\tau$, на котором распределение искомой функции известно, и верхний $t^{k+1} = (k+1)\tau$, на котором распределение искомой функции подлежит определению.

Далее аппроксимируем дифференциальные операторы отношением конечных разностей.

При аппроксимации второй производной по пространству на нижнем временном слое, получаем явную конечно-разностную схему.

$$\left.\frac{\partial u}{\partial t}\right|_j^k = \frac{u_j^{k+1} - u_j^k}{\tau} + O(\tau),$$

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_j^k = \frac{u_{j+1}^k - 2u_j^k + u_{j-1}^k}{h^2} + O(h^2).$$

Явная конечно-разностная схема:

$$\frac{u_j^{k+1} - u_j^k}{\tau} = a^2 \frac{u_{j+1}^k - 2u_j^k + u_{j-1}^k}{h^2} + O(\tau + h^2), \quad j = \overline{1, N-1}, \quad k = \overline{0, K-1},$$

$$u_0^k = \varphi_0(t^k), \quad u_N^k = \varphi_l(t^k), \quad k = \overline{0, K}; \quad u_j^0 = \psi(x_j), \quad j = \overline{0, N},$$

где для каждого уравнения неизвестна только одна величина u_j^{k+1}, которую можно явно выразить:

$$u_j^{k+1} = \sigma \cdot u_{j+1}^k + (1 - 2\sigma) u_j^k + \sigma \cdot u_{j-1}^k, \quad \sigma = \frac{a^2 \tau}{h^2}, \quad j = \overline{1, N-1}, \quad k = 0, 1, 2 \ldots,$$

Данная схема будет устойчива при условии $\sigma <= ½$.

При аппроксимации второй производной по пространству на верхнем временном слое, получаем неявную конечно-разностную схему.

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_j^{k+1} = \frac{u_{j+1}^{k+1} - 2u_j^{k+1} + u_{j-1}^{k+1}}{h^2} + O(h^2),$$

Неявная конечно-разностная схема:

$$\frac{u_j^{k+1} - u_j^k}{\tau} = a^2 \frac{u_{j+1}^{k+1} - 2u_j^{k+1} + u_{j-1}^{k+1}}{h^2} + O(\tau + h^2), \quad j = \overline{1, N-1}, \quad k = \overline{0, K-1},$$

$$u_0^{k+1} = \varphi_0(t^{k+1}), \quad u_N^{k+1} = \varphi_l(t^{k+1}), \quad k = \overline{0, K-1}; \quad u_j^0 = \psi(x_j), \quad j = \overline{0, N}.$$

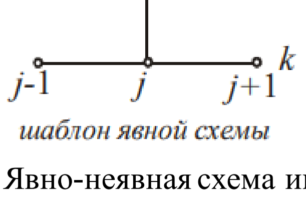
где для нахождения сеточной функции на верхнем временном слое необходимо решить СЛАУ с трехдиагональной матрицей:

$$\begin{matrix} a_1 = 0; \\ \\ c_{N-1} = 0; \end{matrix} \begin{cases} b_1 u_1^{k+1} + c_1 u_2^{k+1} = d_1, \quad j = 1 \\ a_j u_{j-1}^{k+1} + b_j u_j^{k+1} + c_j u_{j+1}^{k+1} = d_j, \quad j = \overline{2, N-2} \\ a_{N-1} u_{N-2}^{k+1} + b_{N-1} u_{N-1}^{k+1} = d_{N-1}, \quad j = N-1, \end{cases}$$

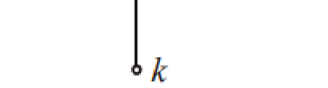
где $a_j = \sigma, j = \overline{2, N-1}; \quad b_j = -(1 + 2\sigma), \quad j = \overline{1, N-1}; \quad c_j = \sigma, \quad j = \overline{1, N-2};$

$$d_j = -u_j^k, \quad j = \overline{2, N-2}; \quad d_1 = -(u_1^k + \sigma\varphi_0(t^{k+1})); \quad d_{N-1} = -(u_{N-1}^k + \sigma\varphi_l(t^{k+1})); \quad \sigma = \frac{a^2 \tau}{h^2}.$$

Приведем шаблоны конечно-разностных схем (их геометрическую интерпретацию на конечно-разностной сетке):



шаблон явной схемы



шаблон неявной схемы

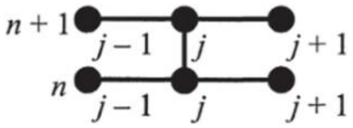
Явно-неявная схема имеет вид:

$$\frac{u_j^{k+1} - u_j^k}{\tau} = \theta a^2 \frac{u_{j+1}^{k+1} - 2u_j^{k+1} + u_{j-1}^{k+1}}{h^2} + (1-\theta) a^2 \frac{u_{j+1}^k - 2u_j^k + u_{j-1}^k}{h^2}$$

где ϴ - вес неявной части, причем 0≤ϴ≤1. При ϴ=0 получаем явную схему, при ϴ=1 – неявную схему, при ϴ=1/2 – схему Кранка-Николсона.

Как и в неявной схеме для решения явно-неявной необходимо решать СЛАУ с трехдиагональной матрицей.

Шаблон схемы Кранка-Николсона:



Также рассмотрим три варианта аппроксимации граничных условий, содержащих производные.

Двухточечная аппроксимация с первым порядком:

$$\left.\frac{\partial u}{\partial x}\right|_{j=0}^{k+1} = \frac{u_1^{k+1} - u_0^{k+1}}{h} + O(h); \qquad \left.\frac{\partial u}{\partial x}\right|_{j=N}^{k+1} = \frac{u_N^{k+1} - u_{N-1}^{k+1}}{h} + O(h),$$

Трехточечная аппроксимация со вторым порядком:

$$\frac{\partial u}{\partial x}(0, t^{k+1}) = \frac{-3u_0^{k+1} + 4u_1^{k+1} - u_2^{k+1}}{2h} + O(h^2),$$

$$\frac{\partial u}{\partial x}(l, t^{k+1}) = \frac{u_{N-2}^{k+1} - 4u_{N-1}^{k+1} + 3u_N^{k+1}}{2h} + O(h^2).$$

Двухточечная аппроксимация со вторым порядком:

Для получения формул разложим u_1^{k+1} в ряд Тейлора в окрестности x = 0 и u_N^{k+1} в окрестности x = l:

$$u_1^{k+1} = u(o + h, t^{k+1}) = u_0^{k+1} + \left.\frac{\partial u}{\partial x}\right|_0^{k+1} h + \left.\frac{\partial^2 u}{\partial x^2}\right|_0^{k+1} \frac{h^2}{2} + O(h^3)$$

$$u_{N-1}^{k+1} = u(l - h, t^{k+1}) = u_N^{k+1} - \left.\frac{\partial u}{\partial x}\right|_N^{k+1} h + \left.\frac{\partial^2 u}{\partial x^2}\right|_N^{k+1} \frac{h^2}{2} + O(h^3)$$

Далее при помощи информации из исходного уравнения, выражая оттуда вторую производную и подставляя полученные выражения, получим:

$$\left.\frac{\partial u}{\partial x}\right|_0^{k+1} = \frac{2a^2}{h(2a^2 - bh)} \cdot \left(u_1^{k+1} - u_0^{k+1}\right) - \frac{h}{2a^2 - bh} \cdot \left.\frac{\partial u}{\partial t}\right|_0^{k+1} + \frac{gh}{2a^2 - bh} \cdot u_0^{k+1} + O_1(h^2),$$

$$\left.\frac{\partial u}{\partial x}\right|_N^{k+1} = \frac{2a^2}{h(2a^2 + bh)} \cdot \left(u_N^{k+1} - u_{N-1}^{k+1}\right) + \frac{h}{2a^2 + bh} \cdot \left.\frac{\partial u}{\partial t}\right|_N^{k+1} - \frac{gh}{2a^2 + bh} \cdot u_N^{k+1} + O_2(h^2).$$

Вариант

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \cos x(\cos t + \sin t),$$
$$u(0,t) = \sin t,$$
$$u_x(\frac{\pi}{2}, t) = -\sin t,$$
$$u(x,0) = 0,$$
Аналитическое решение: $U(x,t) = \sin t \cos x$.

Решение и код

Вспомогательные функции:

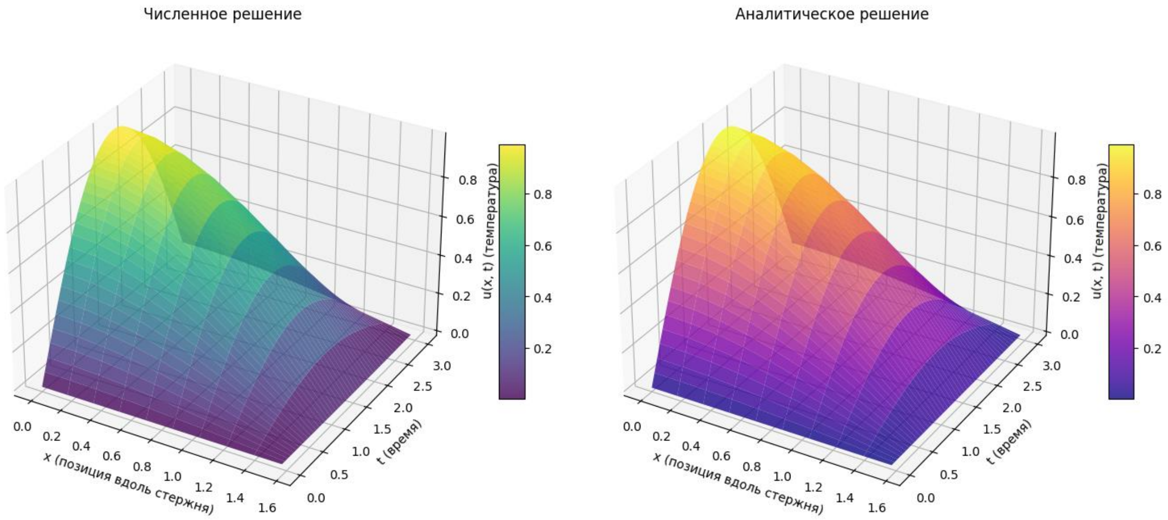
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

def f(x, t):
    return np.cos(x) * (np.cos(t) + np.sin(t))

# Функции для начальных и краевых условий
def f_t(t):
    return np.sin(t)

def f_x(x):
    return 0

def q_l(t):
    return -np.sin(t)

# Метод прогонки для решения трехдиагональной системы
def solve_tridiagonal(a, b, c, d):
    n = len(d)

    c_new = np.zeros(n - 1)
    d_new = np.zeros(n)

    c_new[0] = c[0] / b[0]
    d_new[0] = d[0] / b[0]

    for i in range(1, n):
        denom = b[i] - a[i-1] * c_new[i-1]
        if i < n - 1:
            c_new[i] = c[i] / denom
        d_new[i] = (d[i] - a[i-1] * d_new[i-1]) / denom

    x = np.zeros(n)
    x[-1] = d_new[-1]

    for i in range(n-2, -1, -1):
        x[i] = d_new[i] - c_new[i] * x[i+1]

    return x

def u_exact(x, t):
    return np.sin(t) * np.cos(x)

#///////////////////////////////////////////////////
alpha = 1
L = np.pi / 2
T = 3.0
```

```
N = 9
M = 1000
h = L / N
tau = T / M

if alpha * tau / h**2 > 0.5:
    raise ValueError("Схема неустойчива: уменьшите tau или увеличьте h.")
u = np.zeros((M + 1, N + 1))

x = np.linspace(0, L, N + 1)
t = np.linspace(0, T, M + 1)
u[0, :] = f_x(x)

for n in range(M):
    u[n + 1, 0] = f_t((n + 1) * tau)

    # Основное разностное уравнение
    for i in range(1, N):
        u[n + 1, i] = u[n, i] + (alpha * tau / h**2) * (u[n, i + 1] - 2 * u[n, i] + u[n, i - 1]) + tau * f(x[i], t[n])

    method = 1

    # Граничное условие для x = pi/2
    # 1. Двухточечная аппроксимация с первым порядком
    if method == 1:
        u[n + 1, -1] = u[n + 1, -2] + q_l((n + 1) * tau) * h

    # 2. Трехточечная аппроксимация со вторым порядком
    if method == 2:
        u[n + 1, -1] = (4 * u[n + 1, -2] - u[n + 1, -3] + 2 * q_l((n + 1) * tau) * h) / 3

    # 3. Двухточечная аппроксимация со вторым порядком
    if method == 3:
        u[n + 1, -1] = u[n + 1, -2] + h * q_l((n + 1) * tau) + (tau / 2) * (q_l((n + 1) * tau) - q_l(n * tau))

U_exact = np.array([[np.sin(ti) * np.cos(xi) for xi in x] for ti in t])

X, T = np.meshgrid(x, t)

fig = plt.figure(figsize=(14, 6))

# Левый график: Численное решение
ax1 = fig.add_subplot(121, projection='3d')
surf1 = ax1.plot_surface(X, T, u, cmap='viridis', alpha=0.8)
fig.colorbar(surf1, ax=ax1, shrink=0.5, aspect=10)
ax1.set_title("Численное решение")
ax1.set_xlabel('x (позиция вдоль стержня)')
ax1.set_ylabel('t (время)')
ax1.set_zlabel('u(x, t) (температура)')

# Правый график: Аналитическое решение
ax2 = fig.add_subplot(122, projection='3d')
surf2 = ax2.plot_surface(X, T, U_exact, cmap='plasma', alpha=0.8)
fig.colorbar(surf2, ax=ax2, shrink=0.5, aspect=10)
```

```
ax2.set_title("Аналитическое решение")
ax2.set_xlabel('x (позиция вдоль стержня)')
ax2.set_ylabel('t (время)')
ax2.set_zlabel('u(x, t) (температура)')

error = np.abs(u - U_exact)
print(f"Максимальная погрешность: {np.max(error)}\n\n")

plt.tight_layout()
plt.show()
```

#//

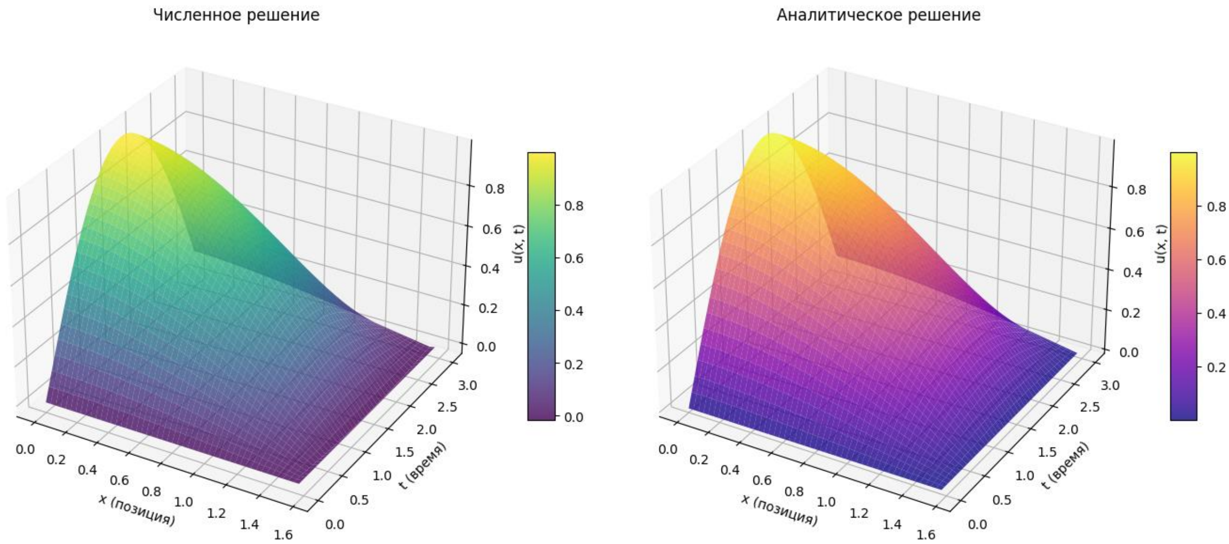
Максимальная погрешность: 0.004694693025208781



#//

```
L = np.pi / 2
T = 3.0
Nx = 50  # Количество узлов по x
Nt = 500  # Количество шагов по времени
dx = L / (Nx - 1)
dt = T / Nt
alpha = dt / dx**2  # Коэффициент схемы

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

U = np.zeros((Nt + 1, Nx))

U[0, :] = 0

# Неявная схема
for n in range(Nt):
    a = -alpha * np.ones(Nx - 2)
    b = (1 + 2 * alpha) * np.ones(Nx - 1)
    c = -alpha * np.ones(Nx - 2)

    d = U[n, 1:-1] + dt * f(x[1:-1], t[n+1])

    # Граничные условия
    d[0] += alpha * np.sin(t[n+1])
```

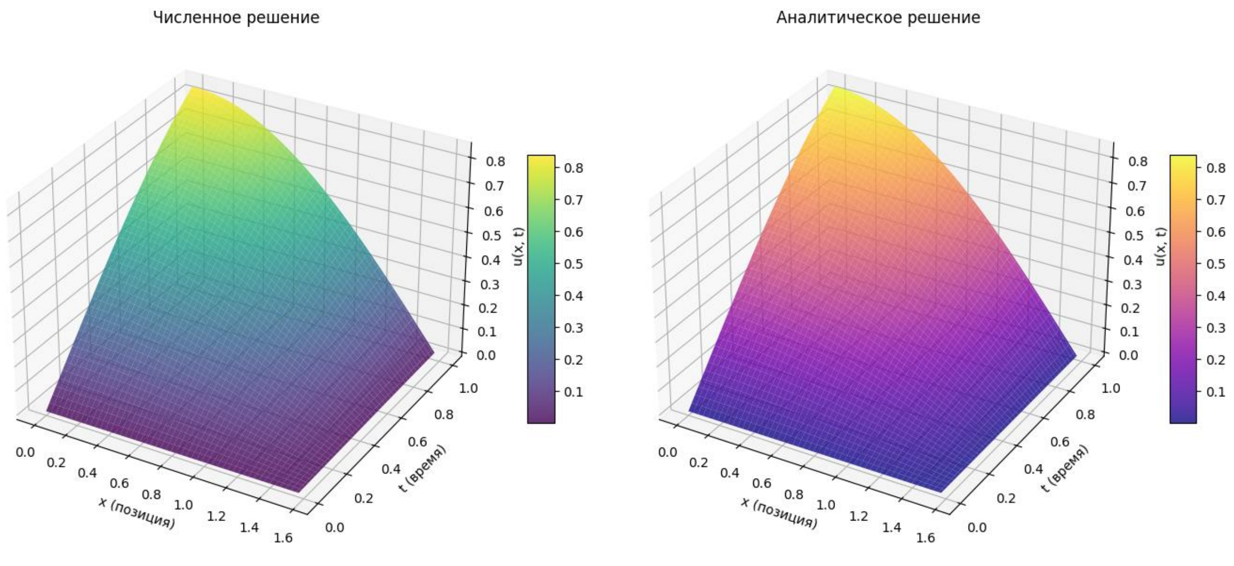
```
    g_t = -np.sin(t[n+1])  # Производная на правом краю

    method = 2
    if method == 1:  # Двухточечная аппроксимация 1-го порядка
        d[-1] -= alpha * dx * g_t
        U[n+1, -1] = U[n+1, -2] - dx * g_t
    elif method == 2:  # Трёхточечная аппроксимация 2-го порядка
        b[-1] = 1 + 2 * alpha
        d[-1] = U[n, -2] + dt * f(x[-2], t[n+1]) + 2 * dx * alpha * g_t
        U[n+1, -1] = (4 * U[n+1, -2] - U[n+1, -3]) / 3
    elif method == 3:  # Двухточечная аппроксимация 2-го порядка
        b[-1] = 1 + 2 * alpha
        d[-1] = U[n, -2] + dt * f(x[-2], t[n+1]) + dx * alpha * g_t
        U[n+1, -1] = (3 * U[n+1, -2] - U[n+1, -3]) / 2

    U[n+1, 1:-1] = solve_tridiagonal(a, b, c, d)

    # Граничные условия
    U[n+1, 0] = np.sin(t[n+1])

U_exact = np.array([[u_exact(xi, ti) for xi in x] for ti in t])
X, T = np.meshgrid(x, t)
fig = plt.figure(figsize=(14, 6))

# Левый график: Численное решение
ax1 = fig.add_subplot(121, projection='3d')
surf1 = ax1.plot_surface(X, T, U, cmap='viridis', alpha=0.8)
fig.colorbar(surf1, ax=ax1, shrink=0.5, aspect=10)
ax1.set_title("Численное решение")
ax1.set_xlabel('x (позиция)')
ax1.set_ylabel('t (время)')
ax1.set_zlabel('u(x, t)')

# Правый график: Аналитическое решение
ax2 = fig.add_subplot(122, projection='3d')
surf2 = ax2.plot_surface(X, T, U_exact, cmap='plasma', alpha=0.8)
fig.colorbar(surf2, ax=ax2, shrink=0.5, aspect=10)
ax2.set_title("Аналитическое решение")
ax2.set_xlabel('x (позиция)')
ax2.set_ylabel('t (время)')
ax2.set_zlabel('u(x, t)')

error = np.abs(U - U_exact)
print(f"Максимальная погрешность: {np.max(error)}\n\n")

plt.tight_layout()
plt.show()

#//////////////////////////////////////////////////
Максимальная погрешность: 0.0626763187881481
```




```
#////////////////////////////////////////////////////
L = np.pi / 2
T = 1.0
Nx = 100  # Количество узлов по x
Nt = 500  # Количество шагов по времени
dx = L / (Nx - 1)
dt = T / Nt
alpha = dt / (2 * dx**2)  # Коэффициент схемы Кранка-Николсона

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

U = np.zeros((Nt + 1, Nx))

U[0, :] = 0

for n in range(Nt):
    a = -alpha * np.ones(Nx - 2)
    b = (1 + 2 * alpha) * np.ones(Nx - 1)
    c = -alpha * np.ones(Nx - 2)
    d = alpha * U[n, :-2] + (1 - 2 * alpha) * U[n, 1:-1] + alpha * U[n, 2:] + \
        (dt / 2) * (f(x[1:-1], t[n+1]) + f(x[1:-1], t[n]))

    # Граничные условия
    d[0] += alpha * np.sin(t[n+1])  # Левое граничное условие
    g_t = -np.sin(t[n+1])  # Производная на правом краю

    method = 1

    # Выбор метода аппроксимации
    if method == 1:  # Двухточечная аппроксимация 1-го порядка
        d[-1] -= alpha * dx * g_t
        U[n+1, -1] = U[n+1, -2] - dx * g_t
    elif method == 2:  # Трёхточечная аппроксимация 2-го порядка
        d[-1] = U[n, -2] + dt * f(x[-2], t[n+1]) + 2 * dx * alpha * g_t
        U[n+1, -1] = (4 * U[n+1, -2] - U[n+1, -3]) / 3 - (2 * dx / 3) * g_t
    elif method == 3:  # Двухточечная аппроксимация 2-го порядка
        d[-1] = U[n, -2] + dt * f(x[-2], t[n+1]) + dx * alpha * g_t
        U[n+1, -1] = (3 * U[n+1, -2] - U[n+1, -3]) / 2 - dx * g_t
```

```
    U[n+1, 1:-1] = solve_tridiagonal(a, b, c, d)

    # Граничные условия
    U[n+1, 0] = np.sin(t[n+1])

U_exact = np.array([[u_exact(xi, ti) for xi in x] for ti in t])
X, T = np.meshgrid(x, t)

fig = plt.figure(figsize=(14, 6))

# Левый график: Численное решение
ax1 = fig.add_subplot(121, projection='3d')
surf1 = ax1.plot_surface(X, T, U, cmap='viridis', alpha=0.8)
fig.colorbar(surf1, ax=ax1, shrink=0.5, aspect=10)
ax1.set_title("Численное решение")
ax1.set_xlabel('x (позиция)')
ax1.set_ylabel('t (время)')
ax1.set_zlabel('u(x, t)')

# Правый график: Аналитическое решение
ax2 = fig.add_subplot(122, projection='3d')
surf2 = ax2.plot_surface(X, T, U_exact, cmap='plasma', alpha=0.8)
fig.colorbar(surf2, ax=ax2, shrink=0.5, aspect=10)
ax2.set_title("Аналитическое решение")
ax2.set_xlabel('x (позиция)')
ax2.set_ylabel('t (время)')
ax2.set_zlabel('u(x, t)')

error = np.abs(U - U_exact)
print(f"Максимальная погрешность: {np.max(error)}\n\n")

plt.tight_layout()
plt.show()

#/////////////////////////////////////////////////////////
```

Максимальная погрешность: 0.013351308404451747



Выводы

В ходе выполнения данной работы для решения начально-краевой задачи для дифференциального уравнения параболического типа были реализованы явная и неявная конечно-разностные схемы, схема Кранка-Николсона. Также реализованы три варианта аппроксимации начальных условий, содержащих производные: двухточечная аппроксимация с первым порядком, трехточечная со вторым и двухточечная со вторым. Путем сравнения результатов с приведенным точным решением была вычислена погрешность методов путем нахождения максимальной ошибки.