Московский авиационный институт

(национальный исследовательский университет)

Институт информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №5

по курсу “Численные методы”

Студент: Пясковский Н.Л.

Группа: М8О-409Б-19

Руководитель: Пивоваров Д.Е.

Оценка: _______________

Москва 2022

1) Задание

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.

2) Вариант

8.

$$\frac{\partial u}{\partial t} = a\frac{\partial^2 u}{\partial x^2} + cu,\ a > 0,\ c < 0.$$

$$u_x(0,t) = \exp((c-a)t),$$

$$u(\frac{\pi}{2},t) = \exp((c-a)t),$$

$$u(x,0) = \sin x,$$

Аналитическое решение: $U(x,t) = \exp((c-a)t)\sin x$.

3) Теория

Заданная задача выглядит следующим образом:

$$\begin{cases} \dfrac{\partial u}{\partial t} = a^2\dfrac{\partial^2 u}{\partial x^2} + b\dfrac{\partial u}{\partial x} + gu, \quad 0 < x < l, \quad t > 0; \\ \alpha\dfrac{\partial u(0,t)}{\partial x} + \beta u(0,t) = \varphi_0(t), \quad x = 0, \quad t > 0; \\ \gamma\dfrac{\partial u(l,t)}{\partial x} + \delta u(l,t) = \varphi_l(t), \quad x = l, \quad t > 0; \\ u(x,0) = \psi(x), \quad 0 \le x \le l, \quad t = 0. \end{cases}$$

Для её решения применяется метод конечных разностей. Нанесем на пространственно-временную область $0 \le x \le l, 0 \le t \le T$ конечно-разностную сетку:

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

с пространственным шагом h=l/N и шагом по времени τ=T/K

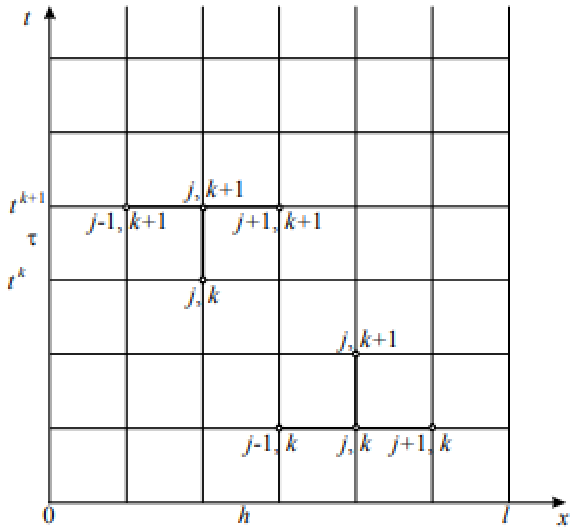


Рис. 5.1. Конечно-разностная сетка

Введем два временных слоя: нижний $t^k = k\tau$, на котором распределение искомой функции $u(x_j, t^k), j = \overline{0, N}$ известно и верхний временной слой $t^{k+1} = (k+1)\tau$, на котором распределение искомой функции $u(x_j, t^{k+1}), j = \overline{0, N}$ подлежит определению.

Сеточной функцией задачи u_j^k назовем однозначное отображение целых аргументов j, k в значения функции $u_j^k = u(x_j, t^k)$.

На введенной сетке введем сеточные функции u_j^k, u_j^{k+1}, первая из которых известна, вторая – подлежит определению. Для ее определения в задаче аппроксимируем дифференциальные операторы отношением конечных разностей, получим

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

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

Подставляя (5.13), (5.14) в задачу (5.1)-(5.4), получим *явную конечно-разностную схему* для этой задачи в форме

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

$$u_0^k = \varphi_0(t^k), \quad u_N^k = \varphi_l(t^k), \quad k = \overline{0, K}; \quad u_j^0 = \psi(x_j), \quad j = \overline{0, N}, \tag{5.15}$$

где для каждого j-го уравнения все значения сеточной функции известны, за исключением одного - u_j^{k+1}, которое может быть определено *явно* из соотношений (5.15).

Если в (5.14) дифференциальный оператор по пространственной переменной аппроксимировать отношением конечных разностей на верхнем временном слое

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_j^{k+1} = \frac{u_{j+1}^{k+1} - 2u_j^{k+1} + u_{j-1}^{k+1}}{h^2} + O(h^2), \qquad (5.16)$$

То после подстановки (5.13), (5.16) в задачу, получим ***неявную конечно-разностную схему*** для этой задачи

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

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

Явная конечно-разностная схема (5.15), записанная в форме

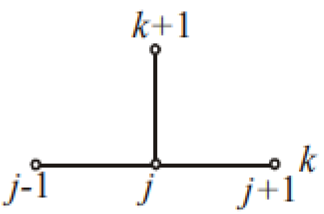
$$u_j^{k+1} = \sigma \cdot u_{j+1}^k + (1 - 2\sigma) u_j^k + \sigma \cdot u_{j-1}^k, \;\; \sigma = \frac{a^2 \tau}{h^2}, \quad j = \overline{1, N-1}, \quad k = 0, 1, 2 \ldots, \qquad (5.18)$$

обладает тем *достоинством*, что решение на верхнем временном слое t^{k+1} получается сразу (без решения СЛАУ) по значениям сеточных функций на нижнем временном слое t^k, где решение известно (при k=0 значения сеточной функции формируются из начального условия (5.4.)). Но эта же схема обладает существенным *недостатком*, поскольку она является условно устойчивой с условием $\sigma = \frac{a^2 \tau}{h^2} \le \frac{1}{2}$, накладываем на сеточные характеристики τ и h.

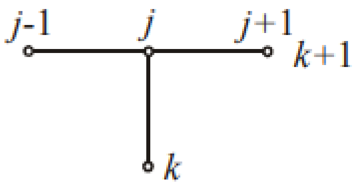
С другой стороны, неявная конечно-разностная схема (5.17), записанная форме

$$a_j u_{j-1}^{k+1} + b_j u_j^{k+1} + c_j u_{j+1}^{k+1} = d_j, \quad j = \overline{1, N-1}, \quad \text{k} = 0, 1, 2, \ldots, \qquad (5.19)$$

приводит к необходимости решать СЛАУ, но зато эта схема абсолютно устойчива.



шаблон явной схемы



шаблон неявной схемы

Рассмотрим неявно-явную схему с весами для простейшего уравнения теплопроводности

$$\frac{u_j^{k+1}-u_j^k}{\tau}=\theta a^2\frac{u_{j+1}^{k+1}-2u_j^{k+1}+u_{j-1}^{k+1}}{h^2}+(1-\theta)a^2\frac{u_{j+1}^k-2u_j^k+u_{j-1}^k}{h^2}, \qquad (5.20)$$

Где θ – вес неявной части конечно-разностной схемы, 1- θ – вес для явной части, причем $0 \le \theta \le 1$. При $\theta = 1$ имеем полностью неявную схему, при $\theta = 0$ – полностью явную схему, и при $\theta = 1/2$ – ***схему Кранка-Николсона***.

Рассмотрим виды аппроксимации граничных условий:

Двухточечная со вторым порядком:

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

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

Двухточечная с первым порядком:

$$\alpha\frac{u_1^{k+1}-u_0^{k+1}}{h}+\beta u_0^{k+1}=\varphi_0(t^{k+1})+O(h),$$

$$\gamma\frac{u_N^{k+1}-u_{N-1}^{k+1}}{h}+\delta u_N^{k+1}=\varphi_l(t^{k+1})+O(h).$$

Трехточечная со вторым порядком:

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

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

4) Программа

```
import numpy as np
from matplotlib import pyplot as plt
import warnings

T = 1  # верхняя граница времени
N = 60  # количество отрезков x
K = 1000  # количество отрезков t
l = np.pi / 2  # правая граница координат
a = 0.1  # коэффициент температуропроводности
c = -1
h = l / N  # пространственный шаг
```

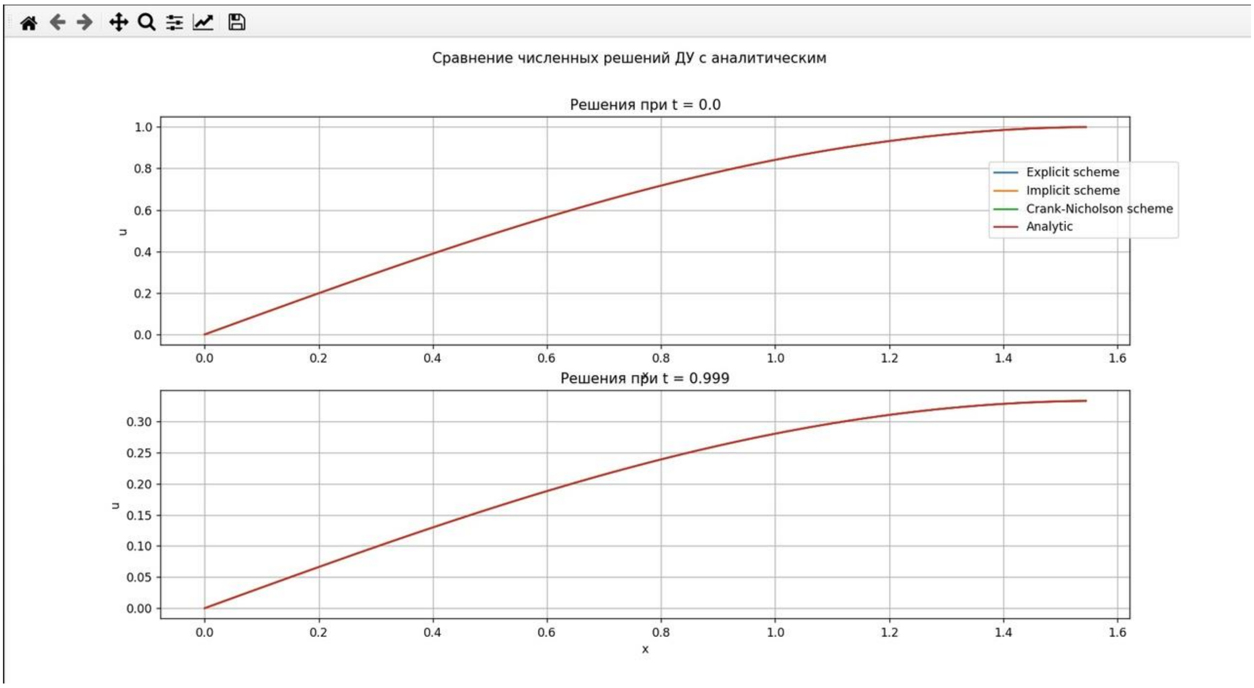
```
tau = T / K  # шаг по времени
sigma = a * tau / (h ** 2)  # число Куранта, используется для анализа устойчивости решения


# аналитическое решение
def analytic_solution(x, t):
    return np.exp((c - a) * t) * np.sin(x)


# условия
def psi(x):
    return np.sin(x)


def phi0(t):
    return np.exp((c - a) * t)


def phi1(t):
    return np.exp((c - a) * t)


# явная конечно-разностная схема
def explicit_scheme(bound_condition):
    if sigma > 0.5:
        warnings.warn("Sigma > 0.5")

    u = np.zeros((K, N))

    for i in range(1, N):
        u[0][i] = psi(i * h)

    for k in range(1, K):
        for i in range(1, N - 1):
            u[k][i] = sigma * u[k - 1][i + 1] + (1 - 2 * sigma) * u[k - 1][i] + sigma * u[k - 1][i - 1] + c * tau * \
                      u[k - 1][i]

        if bound_condition == 1:
            u[k][0] = u[k][1] - h * phi0(k * tau)
            u[k][-1] = phi1(k * tau)
        elif bound_condition == 2:
            u[k][0] = (phi0(k * tau) + u[k][2] / (2 * h) - 2 * u[k][1] / h) * 2 * h / -3
            u[k][-1] = phi1(k * tau)
        elif bound_condition == 3:
            u[k][0] = (u[k][1] - h * phi0(k * tau) + (h ** 2 / (2 * tau) * u[k - 1][0])) / (1 + h ** 2 / (2 * tau))
            u[k][-1] = phi1(k * tau)
        else:
            warnings.warn("Bound condition not found")

    return u


# неявная конечно-разностная схема
def implicit_scheme(bound_condition):
```

```
    u = np.zeros((K, N))

    ai = np.zeros(N)
    bi = np.zeros(N)
    ci = np.zeros(N)
    di = np.zeros(N)

    for i in range(1, N):
        u[0][i] = psi(i * h)

    for k in range(1, K):
        for i in range(1, N - 1):
            ai[i] = sigma
            bi[i] = -2 * sigma + c * tau - 1
            ci[i] = sigma
            di[i] = -u[k - 1][i]

        if bound_condition == 1:
            bi[0] = -(1 + 2 * sigma - c * tau)
            ci[0] = 2 * sigma
            di[0] = -(u[k - 1][0] - 2 * a * tau * phi0(k * tau) / h)
            ai[-1] = 2 * sigma
            bi[-1] = -(1 + 2 * sigma - c * tau)
            #di[-1] = -(u[k - 1][-1] + sigma * phi1(k * tau))
            di[-1] = -phi1(k * tau)
        elif bound_condition == 2:
            bi[0] = -(1 + 2 * sigma - c * tau)
            ci[0] = 2 * sigma
            di[0] = -(u[k - 1][0] - 2 * a * tau * phi0(k * tau) / h)
            ai[-1] = 2 * sigma
            bi[-1] = -(1 + 2 * sigma - c * tau)
            #di[-1] = -(u[k - 1][-1] + sigma * phi1(k * tau))
            di[-1] = -phi1(k * tau)
        elif bound_condition == 3:
            bi[0] = -(1 + 2 * sigma - c * tau)
            ci[0] = 2 * sigma
            di[0] = -((1 - sigma) * u[k - 1][1] + sigma / 2 * u[k - 1][0]) - sigma * phi0(k * tau)
            ai[-1] = 2 * sigma
            bi[-1] = -(1 + 2 * sigma - c * tau)
            di[-1] = -phi1(k * tau)
        else:
            warnings.warn("Bound condition not found")

        u[k] = thomas_algorithm(ai, bi, ci, di)

    return u


# явно-неявная схема, при omega = 0.5 - схема Кранка-Николсона, при omega = 1 - полностью неявная схема, при
# omega = 0 - полностью явная схема
def explicit_implicit_scheme(omega, bound_condition):
    u = np.zeros((K, N))
```

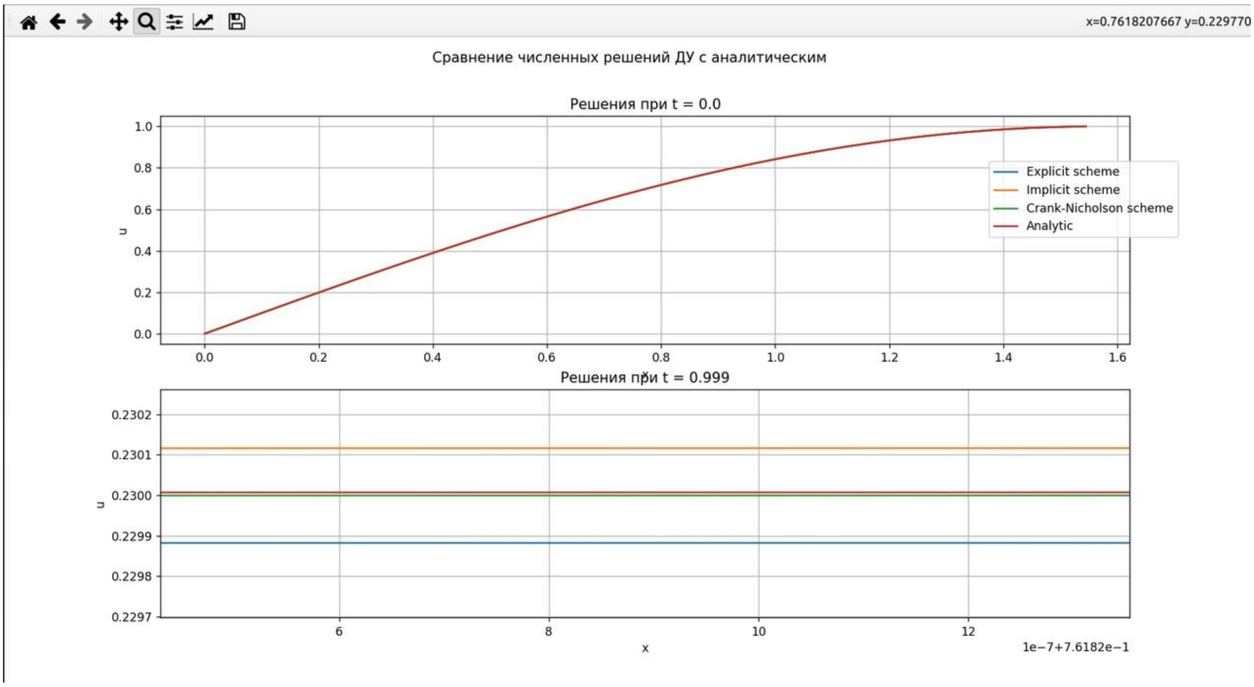
```
    imp = implicit_scheme(bound_condition)
    exp = explicit_scheme(bound_condition)

    for k in range(0, K):
        for i in range(N):
            u[k][i] = imp[k][i] * omega + exp[k][i] * (1 - omega)

    return u


# вывод решения в виде графиков
def show_result(t_axis, x_axis, u1, u2, u3):
    fig, ax = plt.subplots(2)
    fig.suptitle('Сравнение численных решений ДУ с аналитическим')
    fig.set_figheight(15)
    fig.set_figwidth(16)
    time = 0
    for i in range(2):
        ax[i].plot(x_axis, u1[time, :], label='Explicit scheme')
        ax[i].plot(x_axis, u2[time, :], label='Implicit scheme')
        ax[i].plot(x_axis, u3[time, :], label='Crank-Nicholson scheme')
        ax[i].plot(x_axis, [analytic_solution(x, t_axis[time]) for x in x_axis], label='Analytic')
        ax[i].grid(True)
        ax[i].set_xlabel('x')
        ax[i].set_ylabel('u')
        ax[i].set_title(f'Решения при t = {time / K}')
        time += K - 1

    plt.legend(bbox_to_anchor=(1.05, 2), loc='upper right', borderaxespad=0)
    plt.show()

    fig = plt.figure(num=1, figsize=(19, 12), clear=True)
    ax = fig.add_subplot(1, 1, 1, projection='3d')
    fig.suptitle('Аналитическое решение')
    xgrid, tgrid = np.meshgrid(x_axis, t_axis)
    ax.plot_surface(xgrid, tgrid, analytic_solution(xgrid, tgrid))
    ax.set(xlabel='x', ylabel='t', zlabel='u')
    fig.tight_layout()
    plt.show()


# вывод изменения ошибки со временем
def show_inaccuracy(t_axis, x_axis, u):
    inaccuracy = np.zeros(K)
    for i in range(K):
        inaccuracy[i] = np.max(np.abs(u[i] - np.array([analytic_solution(x, t_axis[i]) for x in x_axis])))

    plt.figure(figsize=(14, 8))
    plt.plot(t_axis[1:], inaccuracy[1:], 'violet', label='Ошибка')
    plt.legend(bbox_to_anchor=(1.05, 1), loc='upper right', borderaxespad=0.)
    plt.title('График изменения ошибки во времени')
    plt.xlabel('t')
    plt.ylabel('error')
    plt.grid(True)
```

```python
    plt.show()


# метод прогонки
def thomas_algorithm(a, b, c, d):
    size = len(a)
    P = np.zeros(size)
    Q = np.zeros(size)
    P[0] = -c[0] / b[0]
    Q[0] = d[0] / b[0]

    for i in range(1, size):
        s = (b[i] + a[i] * P[i - 1])
        P[i] = -c[i] / s
        Q[i] = (d[i] - a[i] * Q[i - 1]) / s

    result = np.zeros(size)
    result[-1] = Q[-1]

    for i in range(size - 2, -1, -1):
        result[i] = P[i] * result[i + 1] + Q[i]

    return result


# Основной аргумент схем - тип аппроксимации граничных условий, содержащих производные
# 1 - двухточечная аппроксимация с первым порядком
# 2 - трехточечная аппроксимация со вторым порядком
# 3 - двухточечная аппроксимация со вторым порядком
def main():
    u1 = explicit_scheme(1)
    u2 = implicit_scheme(1)
    u3 = explicit_implicit_scheme(0.5, 1)

    t_axis = np.zeros(K)
    for i in range(K):
        t_axis[i] = tau * i

    x_axis = np.zeros(N)
    for i in range(N):
        x_axis[i] = i * h

    show_result(t_axis, x_axis, u1, u2, u3)

    show_inaccuracy(t_axis, x_axis, u1)


if __name__ == '__main__':
    main()
```

5) Результат



Сравнение численных решений ДУ с аналитическим
Решения при t = 0.0
Explicit scheme
Implicit scheme
Crank-Nicholson scheme
Analytic
u
Решения при t = 0.999
u
x



x=0.7618207667 y=0.229770
Сравнение численных решений ДУ с аналитическим
Решения при t = 0.0
Explicit scheme
Implicit scheme
Crank-Nicholson scheme
Analytic
u
Решения при t = 0.999
u
x
1e−7+7.6182e−1

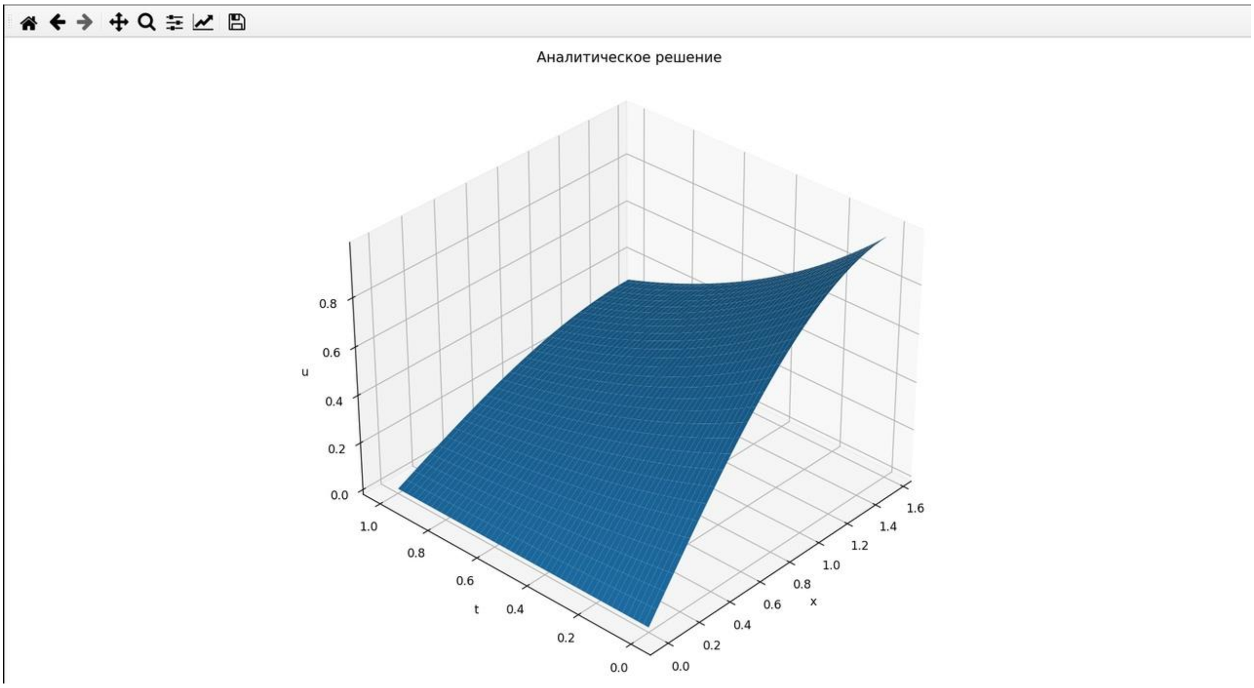


График изменения ошибки во времени для явной схемы: