Лабораторная работа №5 учебного года 2023-2024 по курсу «Численные методы»

Выполнил: Борисов Я. А
Группа: М8О-408Б-20
Преподаватель: Пивоваров Д.Е.
Вариант по списку группы: 4

Условие лабораторной работы

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.

Вариант 4

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

$$u_x(0, t) = \exp(-at),$$

$$u_x(\pi, t) = -\exp(-at),$$

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

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

Программа

main.py

```
import numpy as np
from functions import phi0, phil, psi
from draw_results import draw_results


# Метод прогонки
def tma(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):
        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])

    x = np.zeros(size)
    x[-1] = q[-1]

    for i in range(size - 2, -1, -1):
        x[i] = p[i] * x[i + 1] + q[i]

    return x


def explicit(a: float, n: int, tc: int, tau: float, x_min: float, x_max: float, al: int):

    h = (x_max - x_min) / n
    sigma = a ** 2 * tau / h ** 2

    if sigma > 0.5:
        raise Exception(f'Явная схема не устойчива sigma = {sigma}')
    u = np.zeros((tc, n))

    for j in range(1, n - 1):
        u[0][j] = psi(x_min + j * h)

    for k in range(1, tc):
        for j in range(1, n - 1):
            u[k][j] = sigma * (u[k - 1][j + 1] + u[k - 1][j - 1]) + (1 - 2 * sigma) * u[k -
1][j]
        if al == 1:
            u[k][0] = u[k][1] - h * phi0(k * tau, a)
            u[k][-1] = u[k][-2] + h * phil(k * tau, a)
        elif al == 2:
            u[k][0] = (u[k][1] - h * phi0(k * tau, a) + (h ** 2 / (2 * tau) * u[k - 1][0]))
/ (1 + h ** 2 / (2 * tau))
            u[k][-1] = (u[k][-2] + h * phil(k * tau, a) + (h ** 2 / (2 * tau) * u[k - 1][-
1])) / (1 + h ** 2 / (2 * tau))
```

```
        elif al == 3:
            u[k][0] = (phi0(k * tau, a) + u[k][2] / (2 * h) - 2 * u[k][1] / h) * 2 * h / -3
            u[k][-1] = (phil(k * tau, a) - u[k][-3] / (2 * h) + 2 * u[k][-2] / h) * 2 * h /
3
        else:
            raise Exception('Такого типа апроксимации граничных условий не существует')

    return u

def explicit_implicit(ap: float, n: int, tc: int, tau: float, x_min: float, x_max: float,
al: int, eta = 0.5):

    u = np.zeros((tc, n))
    h = (x_max - x_min) / n
    sigma = ap ** 2 * tau / h ** 2

    for i in range(1, n - 1):
        u[0][i] = psi(x_min + i * h)

    for k in range(1, tc):
        a = np.zeros(n)
        b = np.zeros(n)
        c = np.zeros(n)
        d = np.zeros(n)
        for j in range(1, n - 1):
            a[j] = sigma
            b[j] = -(1 + 2 * sigma)
            c[j] = sigma
            d[j] = -u[k - 1][j]

        if al == 1:
            b[0] = -1 / h
            c[0] = 1 / h
            d[0] = phi0((k + 1) * tau, ap)
            a[-1] = -1 / h
            a[-1] = 1 / h
            d[-1] = phil((k + 1) * tau, ap)
        elif al == 2:
            b[0] = 2 * ap ** 2 / h + h / tau
```

```
            c[0] = - 2 * ap ** 2 / h
            d[0] = (h / tau) * u[k - 1][0] - phi0((k + 1) * tau, ap) * 2 * ap ** 2
            a[-1] = -2 * ap ** 2 / h
            b[-1] = 2 * ap ** 2 / h + h / tau
            d[-1] = (h / tau) * u[k - 1][-1] + phil((k + 1) * tau, ap) * 2 * ap ** 2
        elif al == 3:
            k0 = 1 / (2 * h) / c[1]
            b[0] = (-3 / (2 * h) + a[1] * k0)
            c[0] = 2 / h + b[1] * k0
            d[0] = phi0((k + 1) * tau, ap) + d[1] * k0
            k1 = -(1 / (h * 2)) / a[-2]
            a[-1] = (-2 / h) + b[-2] * k1
            b[-1] = (3 / (h * 2)) + c[-2] * k1
            d[-1] = phil((k + 1) * tau, ap) + d[-2] * k1
        else:
            raise Exception('Такого типа апроксимации граничных условий не существует')

        u[k] = eta * tma(a, b, c, d)

        explicit_part = np.zeros(n)

        for j in range(1, n - 1):
            explicit_part[j] = (sigma * (u[k - 1][j + 1] + u[k - 1][j - 1]) + (1 - 2 *
sigma) * u[k - 1][j])
        if al == 1:
            explicit_part[0] = (explicit_part[1] - h * phi0(k * tau, ap))
            explicit_part[-1] = (explicit_part[-2] + h * phil(k * tau, ap))
        elif al == 2:
            explicit_part[0] = ((phi0(k * tau, ap) + explicit_part[2] / (2 * h) - 2 *
explicit_part[1] / h) * 2 * h / -3)
            explicit_part[-1] = ((phil(k * tau, ap) - explicit_part[-3] / (2 * h) + 2 *
explicit_part[-2] / h) * 2 * h / 3)
        elif al == 3:
            explicit_part[0] = (explicit_part[1] - h * phi0(k * tau, ap) + (h ** 2 / (2 *
tau) * u[k - 1][0])) / \
                               (1 + h ** 2 / (2 * tau))
            explicit_part[-1] = (explicit_part[-2] + h * phil(k * tau, ap) + (h ** 2 / (2 *
tau) * u[k - 1][-1])) / \
                                (1 + h ** 2 / (2 * tau))
```

```
        else:
            raise Exception('Такого типа апроксимации граничных условий не существует')

        u[k] += (1 - eta) * explicit_part

    return u



# Тесты

# Явная конечно-разностная схема
a = 1
n = 60
tc = 2000
tau = 0.001
x_min, x_max = 0, np.pi
u = explicit(a = a, n = n, tc = tc, tau = tau, x_min = x_min, x_max = x_max, al = 1)
print(u)

draw_results(tc, x_max, x_min, u, a, n, tau)

# Неявная конечно-разностная схема
a = 1
n = 80
tc = 2000
tau = 0.001
x_min, x_max = 0, np.pi
u = explicit_implicit(ap = a, n = n, tc = tc, tau = tau, x_min = x_min, x_max = x_max, al =
2, eta = 1)
print(u)

draw_results(tc, x_max, x_min, u, a, n, tau)

# Явно-неявная конечно-разностная схема
a = 1
n = 80
tc = 2000
tau = 0.001
```

```
x_min, x_max = 0, np.pi
u = explicit_implicit(ap = a, n = n, tc = tc, tau = tau, x_min = x_min, x_max = x_max, al =
3, eta = 0.5)
print(u)

draw_results(tc, x_max, x_min, u, a, n, tau)
```

functions.py

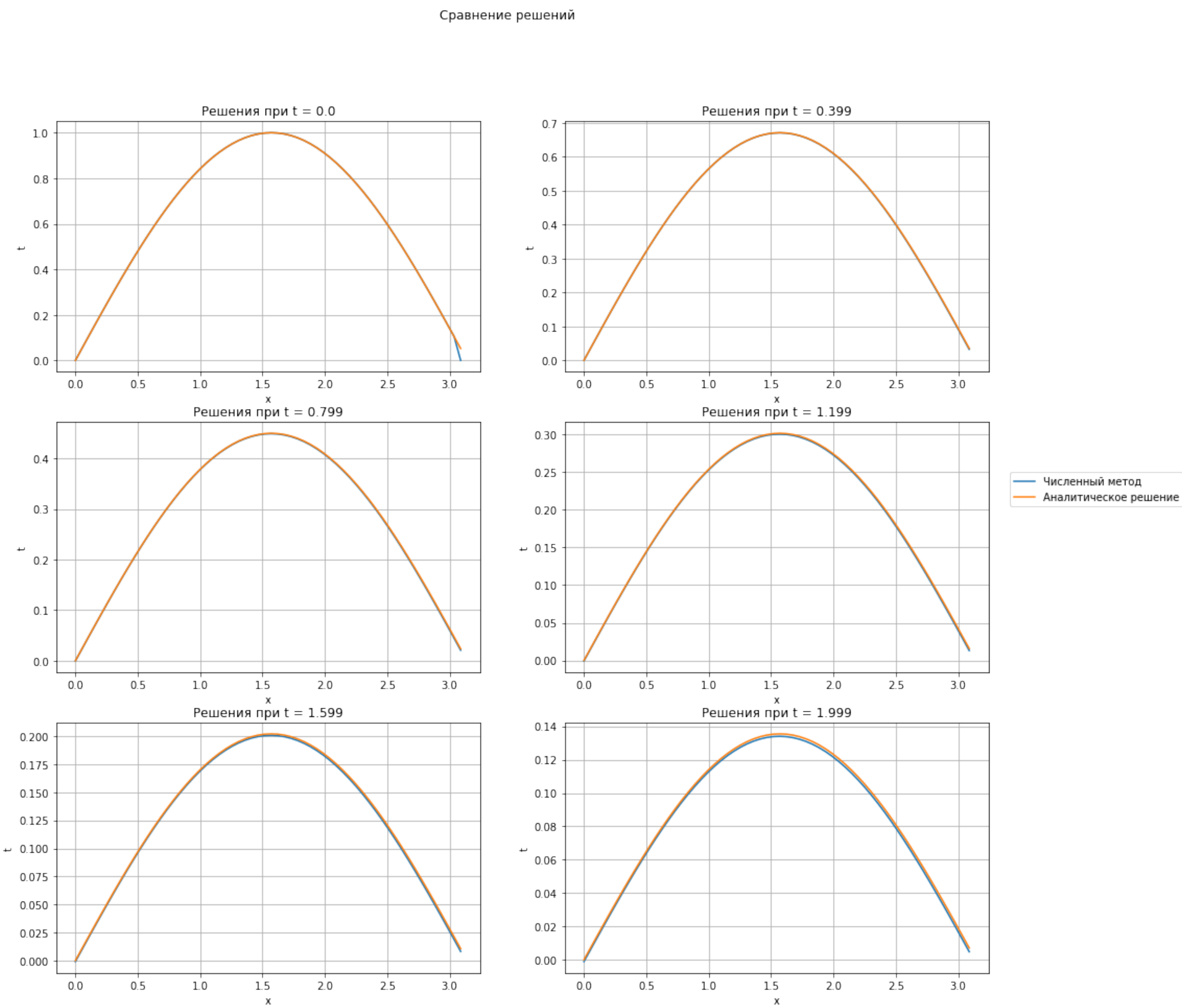
```
import numpy as np

# Граничные условия
def phi0(t: float, a: float, x = 0, ):
    return np.exp(-a * t)

def phil(t: float, a: float, x = np.pi):
    return -np.exp(-a * t)

# Начальные условия
def psi(x: float, t = 0):
    return np.sin(x)

# Аналитическое решение
def U(x: float, t: float, a: float) -> float:
    return np.exp(-a * t) * np.sin(x)
```

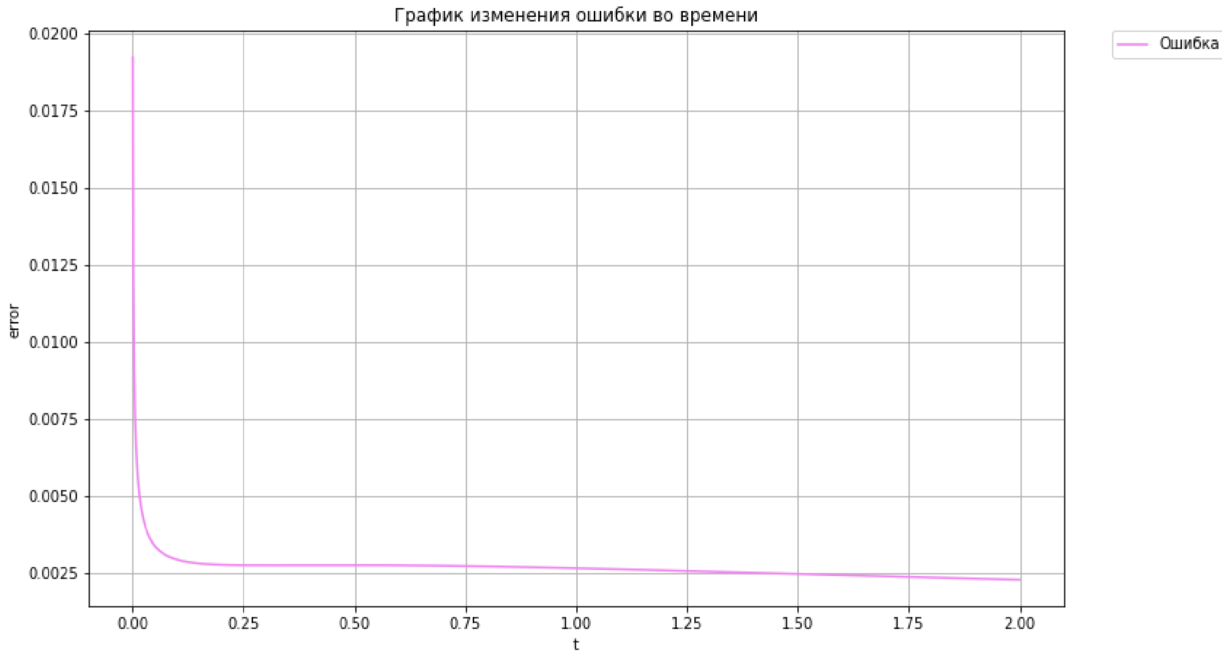
draw_results.py

```
import numpy as np
import matplotlib.pyplot as plt
from functions import U

def draw_results(tc, x_max, x_min, u, a, n, tau):

    times = np.zeros(tc)

    for i in range(tc):
        times[i] = tau * i

    space = np.zeros(n)
    step = (x_max - x_min) / n
```

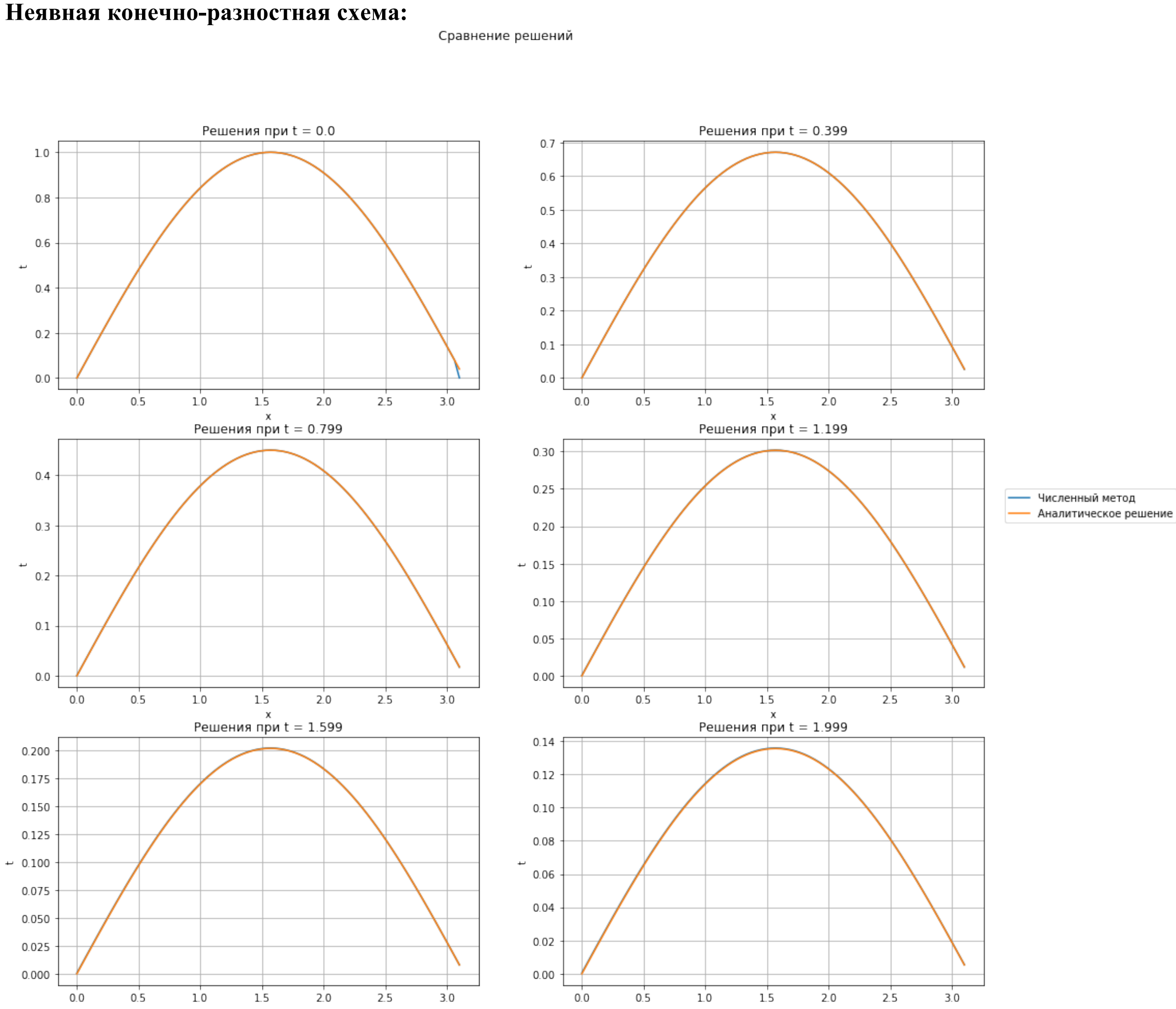
```
    for i in range(n):
        space[i] = x_min + i * step

    times_idx = np.linspace(0, times.shape[0] - 1, 6, dtype = np.int32)
    fig, ax = plt.subplots(3, 2)
    fig.suptitle('Сравнение решений')
    fig.set_figheight(15)
    fig.set_figwidth(16)
    k = 0

    for i in range(3):
        for j in range(2):
            time_idx = times_idx[k]
            ax[i][j].plot(space, u[time_idx], label = 'Численный метод')
            ax[i][j].plot(space, [U(x, times[time_idx], a) for x in space], label =
'Аналитическое решение')
            ax[i][j].grid(True)
            ax[i][j].set_xlabel('x')
            ax[i][j].set_ylabel('t')
            ax[i][j].set_title(f'Решения при t = {times[time_idx]}')
            k += 1
    plt.legend(bbox_to_anchor = (1.05, 2), loc = 'upper left', borderaxespad = 0.)
    error = np.zeros(tc)
    for i in range(tc):
        error[i] = np.max(np.abs(u[i] - np.array([U(x, times[i], a) for x in space])))
    plt.figure(figsize = (12, 7))
    plt.plot(times[1:], error[1:], 'violet', label = 'Ошибка')
    plt.legend(bbox_to_anchor = (1.05, 1), loc = 'upper left', borderaxespad = 0.)
    plt.title('График изменения ошибки во времени')
    plt.xlabel('t')
    plt.ylabel('error')
    plt.grid(True)
    plt.show()
```

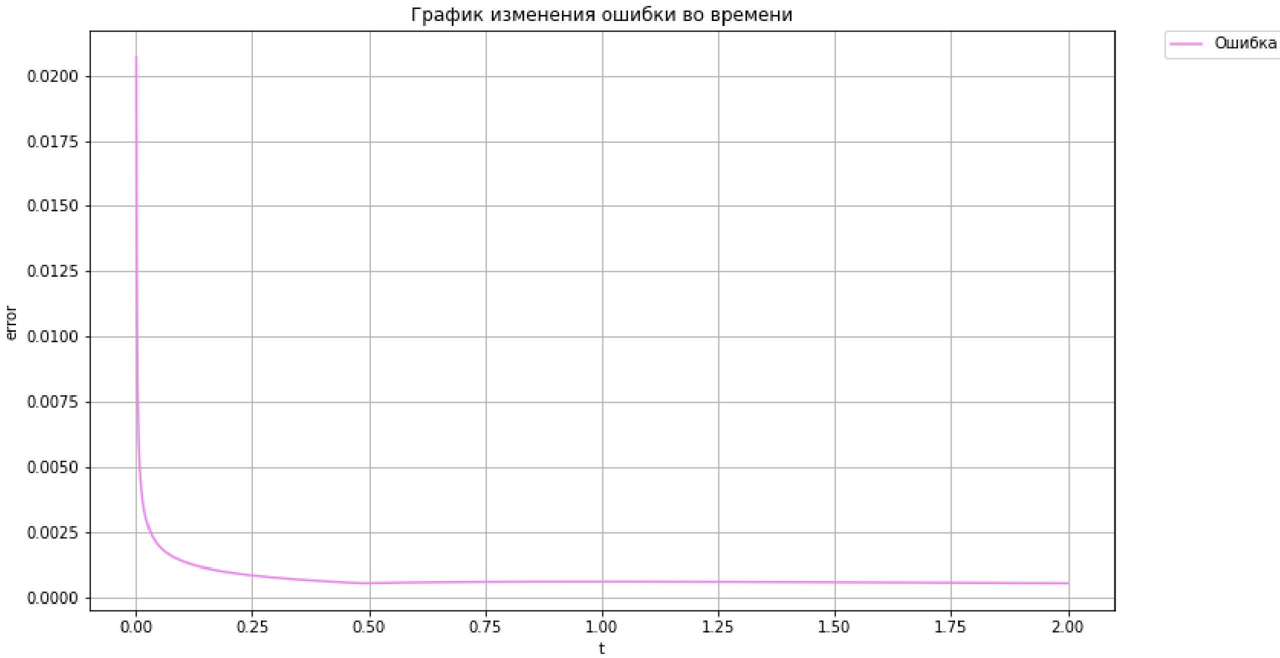
Результаты работы
Явная конечно-разностная схема:



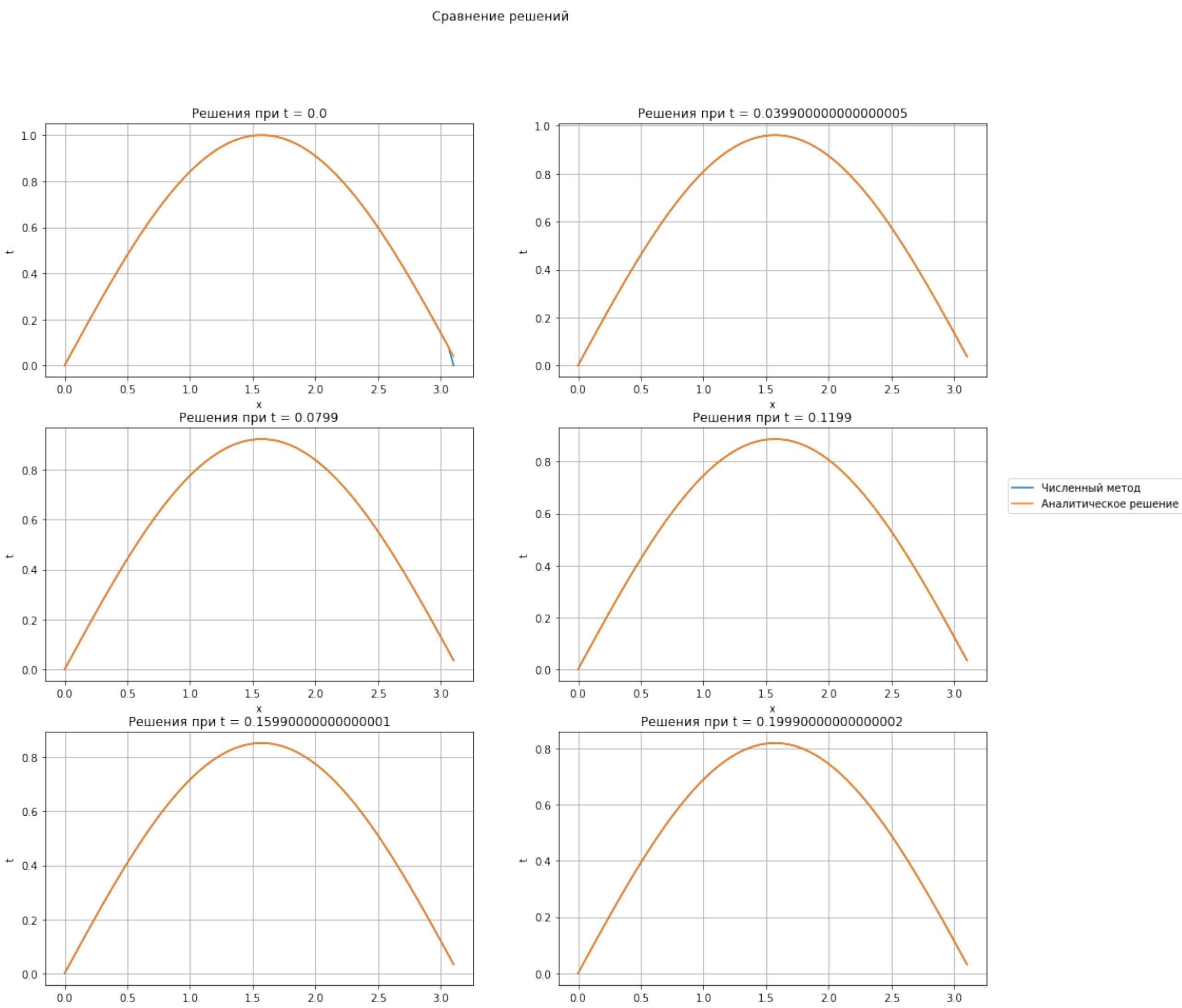


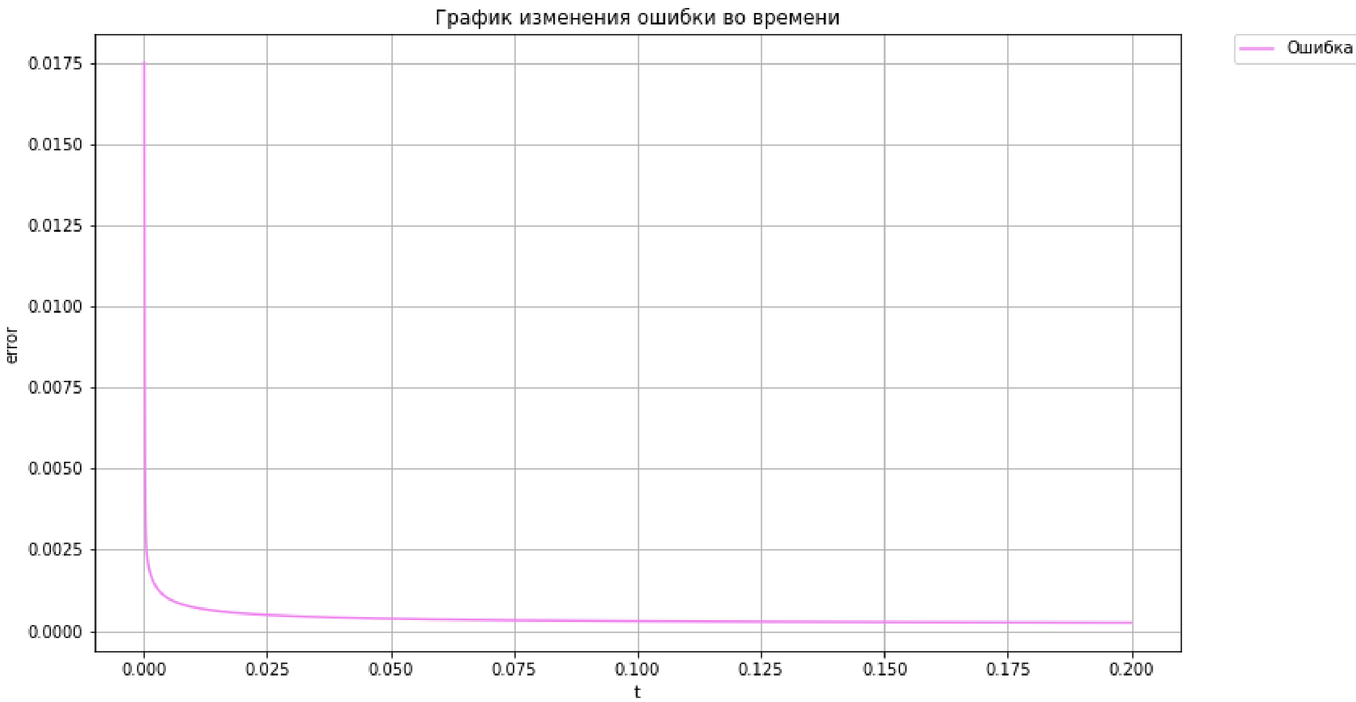
Неявная конечно-разностная схема:





Явно-неявная конечно-разностная схема





Вывод по лабораторной работе

В ходе лабораторной работы я познакомился с численным решением уравнений параболического типа, понятием о методе конечных разностей, с основными определениями и конечно-разностными схемами.

Таким образом, была решена начально-краевая задача для дифференциального уравнения параболического типа. Осуществлена реализация трех вариантов аппроксимации граничных условий, содержащих производные: двухточечная аппроксимация с первым порядком, трехточечная аппроксимация со вторым порядком, двухточечная аппроксимация со вторым порядком. Эксперименты позволили оценить точность и эффективность каждого метода.