**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное бюджетное образовательное учреждение   
высшего образования

**«Сибирский государственный университет науки и технологий   
имени академика М.Ф. Решетнева»**

Институт информатики и телекоммуникаций

Кафедра информатики и вычислительной техники

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6**

Вычислительная математика

|  |
| --- |
| Численное решение задачи Коши для дифференциального уравнения первого порядка |

Руководитель К.А.Кириллов

подпись, дата инициалы, фамилия

Обучающийся БПИ20-02, 201219047 Р.А.Сухачев

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2022 г.

# ЦЕЛЬ РАБОТЫ

Изучить численное решение задачи Коши для дифференциального уравнения первого порядка.

# постановка задачи

Вариант 1:

, u(0) = 1, T = 1,6.

1. Найдите точное решение *U*(*t*) задачи Коши

. (1)

1. Найдите шаг интегрирования для приближенного решения задачи Коши (1) методом Эйлера

с точностью *ε* = 0,001:

,

(2)

*un* и *u͂n* – приближенные решения задачи Коши (1) в точке *tn*, найденные с шагами *τ* и 2*τ* соответственно.

Найдите приближенное решение задачи Коши (1) на отрезке [0, *T*] методом Эйлера с точностью *ε* = 0,001.

Постройте приближенную интегральную кривую – график приближенного решения (*u*0, *u*1,…,*uN*), найденного методом Эйлера.

Сравните точное решение U(t) с приближенным (u0, u1,…,uN), полученным методом Эйлера – найдите величину

.

1. Найдите шаг интегрирования для приближенного решения задачи Коши (1) методом Рунге – Кутты

*k*1 = *τf*(*tn,un*), *k*2 = *τf*(*tn* + 0.5*τ, un* + 0.5*k*1)

*k*3 = *τf*(*tn* + 0.5*τ*, *un* + 0.5*k*2), *k*4 = *τf*(*tn + τ, un* + *k*3),

с точностью *ε* = 0,001:

,

,

un и u͂n имеют тот же смысл, что и в формуле (2).

Найдите приближенное решение задачи Коши (1) на отрезке [0, *T*] методом Рунге – Кутты с точностью *ε* = 0,001.

Постройте приближенную интегральную кривую – график приближенного решения (*u*0, *u*1,…,*uN*), найденного методом Рунге – Кутты.

Сравните точное решение *U*(*t*) с приближенным (*u*0, *u*1,…,*uN*), полученным методом Рунге – Кутты – найдите величину

.

# ХОД РАБОТЫ

**Общие теоретические сведения:**

Было найдено точное решение *U*(*t*) задачи Коши

Методом Эйлера были найдены приближенные решения (u0, u1,…,uN) задачи Коши и значение шага интегрирования, они представлены на рисунке 1.

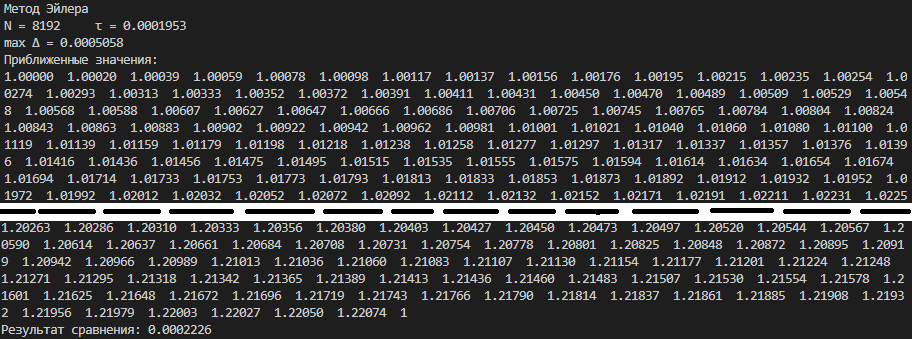


Рисунок 1 – приближенные решения задачи Коши

После была построена приближенная интегральная кривая – график приближенного решения (u0, u1,…,uN), она показана на рисунке 2.

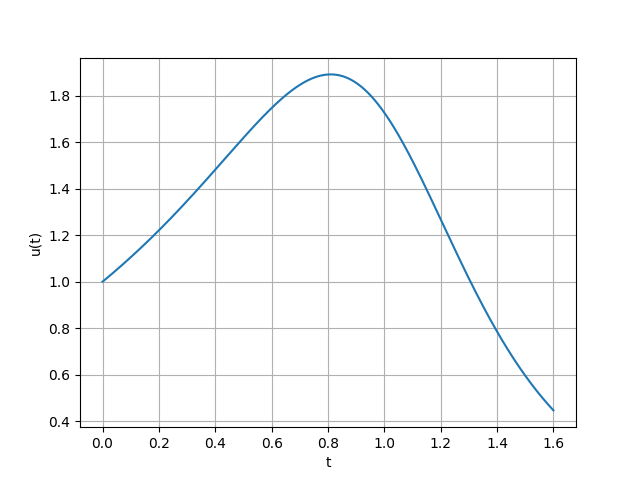


Рисунок 2 – приближенная интегральная кривая

В конце результаты точного решения U(t) и приближенных решений (u0, u1,…,uN) были сравнены

.

С помощью метода Рунге – Кутты были найдены приближенные решения (u0, u1,…,uN) задачи Коши и значение шага интегрирования, они представлены на рисунке 3.

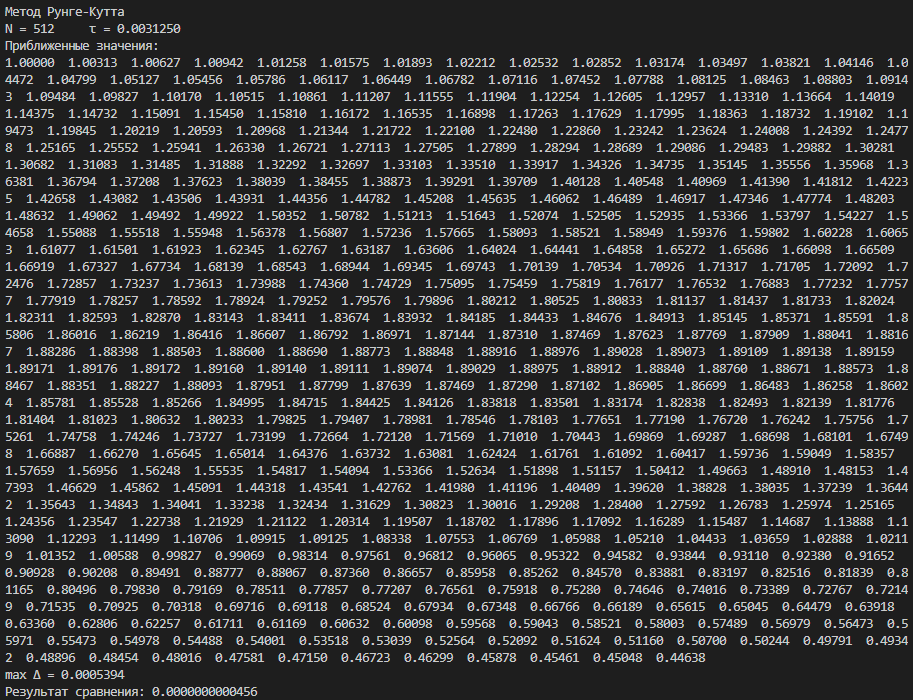


Рисунок 3 – приближенные решения задачи Коши

Также как и с методом Эйлера, была построена приближенная интегральная кривая, представленная на рисунке 4.

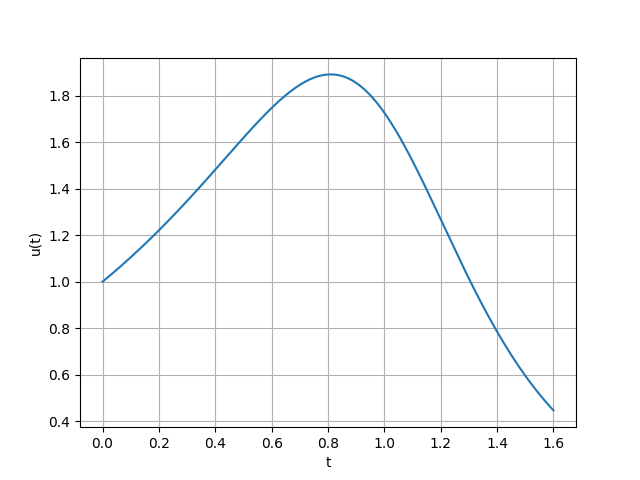


Рисунок 4 – приближенная интегральная кривая

В конце были сравнены результаты точного решения U(t) с приближенными (u0, u1,…,uN)

0000456.

Построенный общий график кривых:

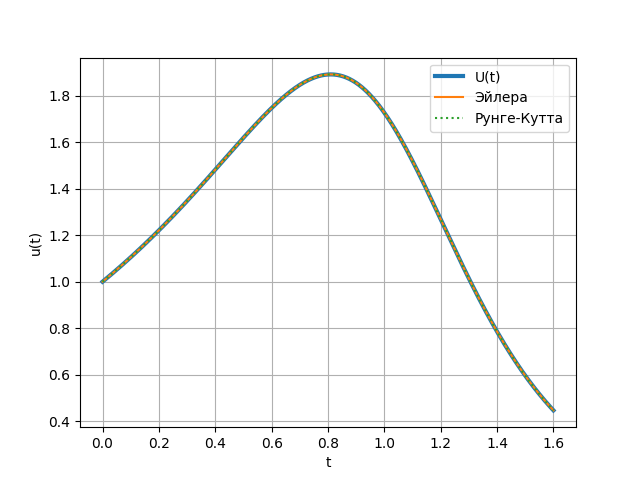


Рисунок 5 – общий график кривых

**Текст программы**

import numpy as np

import matplotlib.pyplot as plt

EPSILON = 0.001

def str\_arr(arr):

    s = ''

    for el in arr:

        s += '{:.5f}'.format(el).ljust(9)

    return s + '\n'

def tau(t, a, n):

    return (t - a) / n

def delta(a, b):

    return [abs(d) for d in [x - y for x, y in zip(a, b)]]

def ti(i, t, a,  n):

    return i \* tau(t, a, n) + a

def diff\_eq(x):

    return -(1 / (np.exp(x) - 2 \* np.exp((x\*x) / 2)))

def func(x, u):

    return (((1 - x) \* np.exp(x) \* np.power(u, 2)) - x \* u)

def euler(x, t, a, n):

    u = [diff\_eq(x[0])]

    for i in range(1, n + 1):

        u.append(u[i - 1] + tau(t, a,n) \* func(x[i - 1], u[i - 1]))

    return u

def euler\_sup(x, u,  t, a, n):

    u\_sup = [diff\_eq(x[0])]

    for i in range(1, n + 1):

        u\_sup.append(u[i - 1] + 2 \* tau(t, a, n) \* func(x[i - 1], u[i - 1]))

    return u\_sup

def rk(x, t, a, n):

    u = [diff\_eq(x[0])]

    for i in range(1, n+1):

        k1 = tau(t, a, n) \* func(x[i - 1], u[i - 1])

        k2 = tau(t, a, n) \* func(x[i - 1] + tau(t, a, n) / 2, u[i-1] + k1 / 2)

        k3 = tau(t, a, n) \* func(x[i - 1] + tau(t, a, n) / 2, u[i-1] + k2 / 2)

        k4 = tau(t, a, n) \* func(x[i - 1] + tau(t, a, n), u[i-1] + k3)

        u.append(u[i-1] + (k1 + 2 \* k2 + 2 \* k3 + k4) / 6)

    return u

def rk\_sup(x, u, t, a, n):

    u\_sup = [diff\_eq(x[0])]

    for i in range(1, n+1):

        k1 = tau(t, a, n) \* func(x[i - 1], u[i - 1])

        k2 = tau(t, a, n) \* func(x[i - 1] + tau(t, a, n) / 2, u[i-1] + k1 / 2)

        k3 = tau(t, a, n) \* func(x[i - 1] + tau(t, a, n) / 2, u[i-1] + k2 / 2)

        k4 = tau(t, a, n) \* func(x[i - 1] + tau(t, a, n), u[i-1] + k3)

        u\_sup.append(u[i-1] + (k1 + 2 \* k2 + 2 \* k3 + k4) / 3)

    return u\_sup

def main():

    a = 0

    n = 2

    t = 1.6

    u = []

    u\_sup = []

    x = np.linspace(a, t, 1000)

    figs = []

    ax = []

    figs.append(plt.figure('Все графики'))

    figs.append(plt.figure("Коши"))

    figs.append(plt.figure('Эйлера'))

    figs.append(plt.figure('Рунге-Кутта'))

    for i in range(4):

        ax.append(figs[i].add\_subplot())

        ax[i].grid()

        ax[i].set(xlabel='t', ylabel='u(t)')

    ax[1].plot(x, diff\_eq(x), label='U(t)')

    ax[0].plot(x, diff\_eq(x), linewidth=3, label='U(t)')

    print('Метод Эйлера')

    with open('euler.txt', 'w', encoding='utf8') as f:

        while True:

            x = [ti(i, t, a, n) for i in range(n + 1)]

            u = euler(x, t, a, n)

            u\_sup = euler\_sup(x, u, t, a, n)

            f.write(f'N = {n}\n')

            print(f'\rN = {n}     \u03C4 = {tau(t, a, n):.7f}', end='\r')

            f.write(f'\u03C4 = {tau(t, a, n):.7f}\n')

            f.write(f'max \u0394 = {max(delta(u, u\_sup)) / 15:.7f}\n')

            f.write('Приближенные значения:\n')

            f.write(str\_arr(u)+'\n')

            if max(delta(u, u\_sup)) < EPSILON:

                break

            n \*=2

    print(f'\rN = {n}     \u03C4 = {tau(t, a, n):.7f}')

    print(f'max \u0394 = {max(delta(u, u\_sup)):.7f}')

    print('Приближенные значения: ')

    print(str\_arr(u), end='')

    print()

    print(f'Результат сравнения: {max(delta(u, diff\_eq(np.array(x)))):.7f}')

    i = delta(u, diff\_eq(np.array(x))).index(max(delta(u, diff\_eq(np.array(x)))))

    print(f'tn = {x[i]:.1f} n = {i}')

    print(f'U({x[i]:.1f}) = {diff\_eq(np.array(x))[i]:.7f}')

    print(f'u{i} = {u[i]:.7f}')

    ax[2].plot(x, u, label='Эйлера')

    ax[0].plot(x, u, '-', label='Эйлера')

    print('\nМетод Рунге-Кутта')

    n = 1

    with open('runge\_kutta.txt', 'w', encoding='utf8') as f:

        while True:

            x = [ti(i, t, a, n) for i in range(n + 1)]

            u = rk(x, t, a, n)

            u\_sup = rk\_sup(x, u, t, a, n)

            f.write(f'N = {n}\n')

            print(f'\rN = {n}     \u03C4 = {tau(t, a, n):.7f}', end='\r')

            f.write(f'\u03C4 = {tau(t, a, n):.7f}\n')

            f.write(f'max \u0394 = {max(delta(u, u\_sup)) / 15:.7f}\n')

            f.write('Приближенные значения:\n')

            f.write(str\_arr(u)+'\n')

            if max(delta(u, u\_sup)) / 15 < EPSILON:

                break

            n \*= 2

    print(f'\rN = {n}     \u03C4 = {tau(t, a, n):.7f}')

    print('Приближенные значения: ')

    print(str\_arr(u), end='')

    print(f'max \u0394 = {max(delta(u, u\_sup)) / 15:.7f}')

    print(f'Результат сравнения: {max(delta(u, diff\_eq(np.array(x)))):.13f}')

    i = delta(u, diff\_eq(np.array(x))).index(max(delta(u, diff\_eq(np.array(x)))))

    print(f'tn = {x[i]:.1f}    n = {i}')

    print(f'U({x[i]:.1f}) = {diff\_eq(np.array(x))[i]:.7f}')

    print(f'u{i} = {u[i]:.7f}')

    ax[3].plot(x, u, label='Рунге-Кутта')

    ax[0].plot(x, u, ':', label='Рунге-Кутта')

    ax[0].legend()

    figs[0].savefig('all.png')

    figs[1].savefig('couchy.png')

    figs[2].savefig('euler.png')

    figs[3].savefig('rk.png')

if \_\_name\_\_ == "\_\_main\_\_":

    main()

# ВЫВОДЫ

В ходе работы мы изучили численное решение задачи Коши для дифференциального уравнения первого порядка.