	Министерство науки и высшего образования Российской Федерации **Федеральное государственное бюджетное образовательное учреждение высшего образования** **«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»** **(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ ____«Информатика и системы управления»________________

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 2

Дисциплина: Моделирование

Тема: Решение системы дифференциальных уравнений с помощью метода Рунге-Кутта

Студент Тимонин А. С.

Группа ИУ7-62Б

Оценка (баллы) _______________

Преподаватель Градов В.М.

Москва.
2020 г.

Тема: Программно- алгоритмическая реализация методов Рунге-Кутта 2-го и 4-го порядков точности при решении системы ОДУ в задаче Коши.

Цель работы: Получение навыков разработки алгоритмов решения задачи Коши при реализации моделей, построенных на системе ОДУ, с использованием методов Рунге-Кутта 2-го и 4-го порядков точности.

Задача: Дан колебательный контур с газоразрядной трубкой:

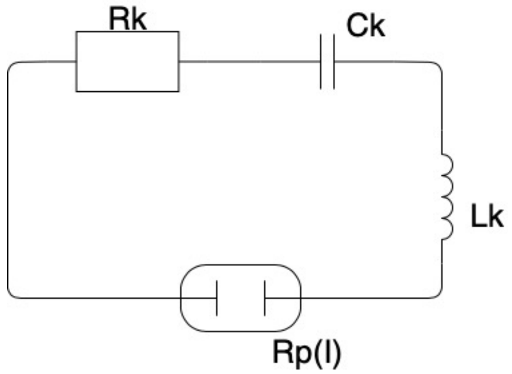


Рисунок 1. Схема колебательного контура с газовой трубкой

Данный контур можно описать с помощью системы уравнений:

$$\begin{cases} L_k \frac{dI}{dt} + \left(R_k + R_p(IRp)\right)I - U_c = 0 \\ \frac{dU_c}{dt} = -I/C_k \end{cases}$$

Необходимо решить численными методами данную систему и построить графики.

Сопротивление $\mathbf{R_p(I)}$ рассчитать по формуле:

$$Rp = \frac{Le}{2\pi R^2 \int_0^1 \sigma(T(z)) z dz}$$

Для функции **T(z)** применить выражение: $T(z) = T_0 + \left(T_w - T_0\right) z^m$

Параметры $\mathbf{T_0}$, **m** находятся интерполяцией из таблицы (1) при известном токе **I**.

Коэффициент электропроводности **σ(T)** зависит от **T** и рассчитывается интерполяцией из таблицы (2).

T, K	σ, 1/Ом см
4000	0.031
5000	0.27
6000	2.05
7000	6.06
8000	12.0
9000	19.9
10000	29.6
11000	41.1
12000	54.1
13000	67.7
14000	81.5

I, A	T_0, K	m
0.5	6400	0.40
1	6790	0.55
5	7150	1.70
10	7270	3.0
50	8010	11.0
200	9185	32.0
400	10010	40.0

800	11140	41.0
1200	12010	39.0

Параметры разрядного контура:

R = 0,35 см (радиус трубки лампы)

$\mathbf{l_e}$ = 12 см (расстояние между электродами лампы)

$\mathbf{L_k}$ = 187 10^{-6} Гн (индуктивность катушки)

$\mathbf{C_k}$ = 268 10^{-6} Ф (емкость конденсатора)

$\mathbf{R_k}$ = 0,25 Ом (сопротивление резистора)

$\mathbf{U_{co}}$ = 1400 В (напряжение на конденсаторе в начальный момент времени)

$\mathbf{I_o}$ = 0..3 А (сила тока в цепи в начальный момент времени)

$\mathbf{T_w}$ = 2000 К (температура в конечный момент времени)

Метод Рунге-Кутта 4-го порядка для системы ОДУ

$$I_{n+1} = I_n + \Delta t \frac{k_1 + 2k_2 + 2k_3 + k_4}{6}$$

$$U_{n+1} = U_n + \Delta t \frac{m_1 + 2m_2 + 2m_3 + m_4}{6}$$

где

$$k_1 = f(I_n, U_{c_n}), m_1 = g(I_n)$$
$$k_2 = f(I_n + \Delta t \frac{k_1}{2}, U_{c_n} + \Delta t \frac{m_1}{2}), m_2 = g(I_n + \Delta t \frac{k_1}{2})$$
$$k_3 = f(I_n + \Delta t \frac{k_2}{2}, U_{c_n} + \Delta t \frac{m_2}{2}), m_3 = g(I_n + \Delta t \frac{k_2}{2})$$
$$k_4 = f(I_n + \Delta t k_3, U_{c_n} + \Delta t m_3), m_4 = g(I_n + \Delta t k_3)$$

Метод Рунге-Кутта 2-го порядка для системы ОДУ

$$\begin{cases} I_{n+1} = I_n + h_n \cdot [(1-\alpha)\cdot f(\Delta t, I_n, U_n) + \alpha \cdot f(x_n + \frac{h_n}{2\alpha}, I_n + \frac{h_n}{2\alpha}\cdot f(\Delta t, I_n, U_n), U_n + \frac{h_n}{2\alpha}\cdot \varphi(\Delta t, I_n, U_n))] \\ U_{n+1} = U_n + h_n \cdot [(1-\alpha)\cdot \varphi(\Delta t, I_n, U_n) + \alpha \cdot \varphi(x_n + \frac{h_n}{2\alpha}, I_n + \frac{h_n}{2\alpha}\cdot f(\Delta t, I_n, U_n), U_n + \frac{h_n}{2\alpha}\cdot \varphi(\Delta t, I_n, U_n))] \end{cases}$$

где **α** задается равной 1 или 0.5

Неявный метод трапеций

Возьмем уравнения:

$$\frac{dI}{dt} = \frac{U_c - (R_k + R_p(I))I}{L_k} \equiv f(I, Uc)$$
$$\frac{dU_c}{dt} = -\frac{1}{C_k} I \equiv g(I)$$

Запишем уравнения для метода:

$$I_{n+1} = I_n + \Delta t \frac{f(I_n, U_{c_n}) + f(I_{n+1}, U_{cn+1})}{2}$$
$$U_{cn+1} = U_{c_n} - \Delta t \frac{I_n + I_{n+1}}{2C_k}$$

Подставим выражения **f** и **g**:

$$I_{n+1} = I_n + \Delta t \frac{U_{c_n} - (R_k + R_p(I_n))I_n + U_{cn+1} - (R_k + R_p(I_{n+1}))I_{n+1}}{2L_k}$$

$$U_{cn+1} = U_{c_n} - \Delta t \frac{I_n + I_{n+1}}{2C_k}$$

Имеем систему с двумя неизвестными $\mathbf{I_{n+1}}$ и $\mathbf{U_{cn+1}}$. Теперь нужно выразить одно из неизвестных. Выражаем $\mathbf{I_{n+1}}$ через $\mathbf{U_{cn+1}}$.

Получаем следующее:

$$I_{n+1} = \frac{-2C_k R_p(I_n) I_n \Delta t + 4C_k L_k I_n - 2C_k I_n R_k \Delta t + 4C_k U_{c_n} \Delta t - I_n \Delta t^2}{4C_k L_k + 2C_k R_k \Delta t + 2C_k R_p(I_{n+1}) \Delta t + \Delta t^2}$$

Данное уравнение можно решить методом простой итерации:

$$x^{(s)} = f(x^{(s-1)})$$

В правую часть подставляем $\mathbf{I_n}$ до тех пор пока не будет достигнута необходимая точность

Листинг 1. Реализации 2-х методов по решению системы ОДУ для колебательного контура.

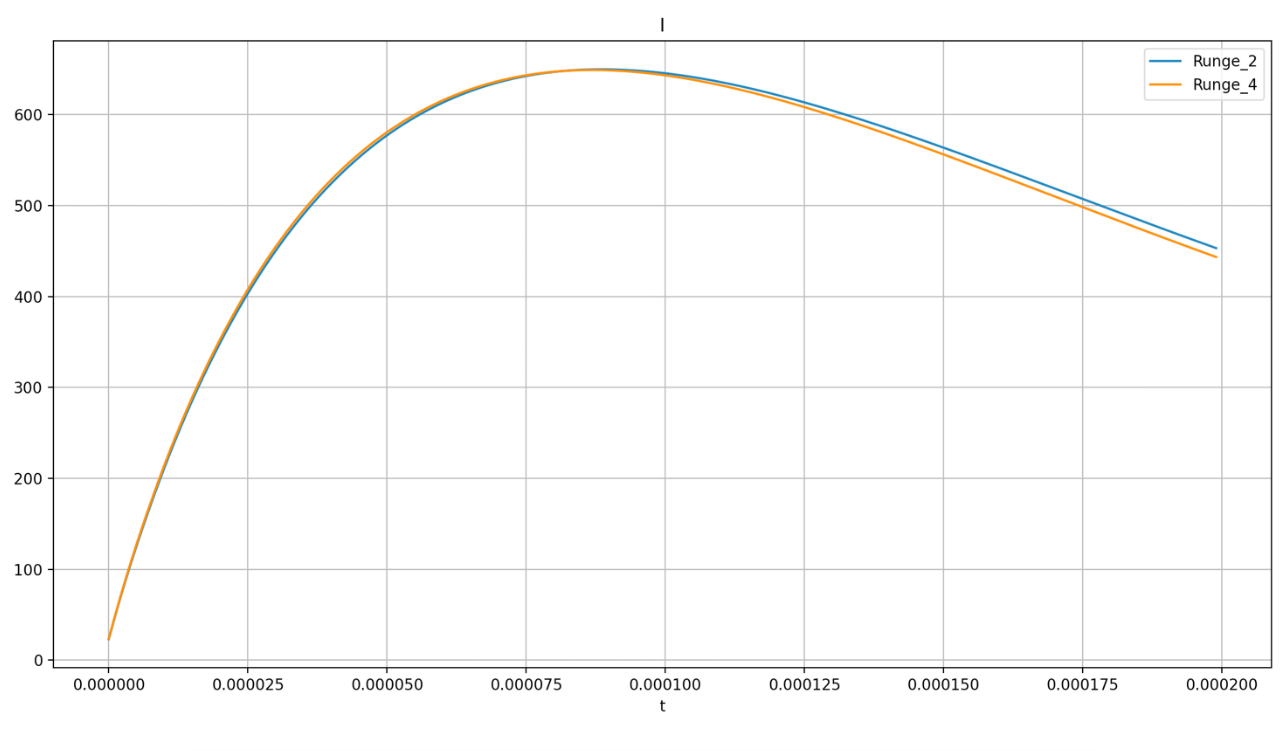
```
from scipy import integrate
from numpy import linspace
from math import pi
import matplotlib.pyplot as plt

def table(number):
    if number == 1:
        return[[0.5,   6400,  0.4],
               [1,     6790,   0.55],
               [5,     7150,   1.7],
               [10,    7270,   3],
               [50,    8010,   11],
               [200,   9185,   32],
               [400,   10010,  40],
               [800,   11140,  41],
               [1200,  12010,  39]]
    elif number == 2:
        return [[4000,   0.031],
                [5000,   0.27],
```

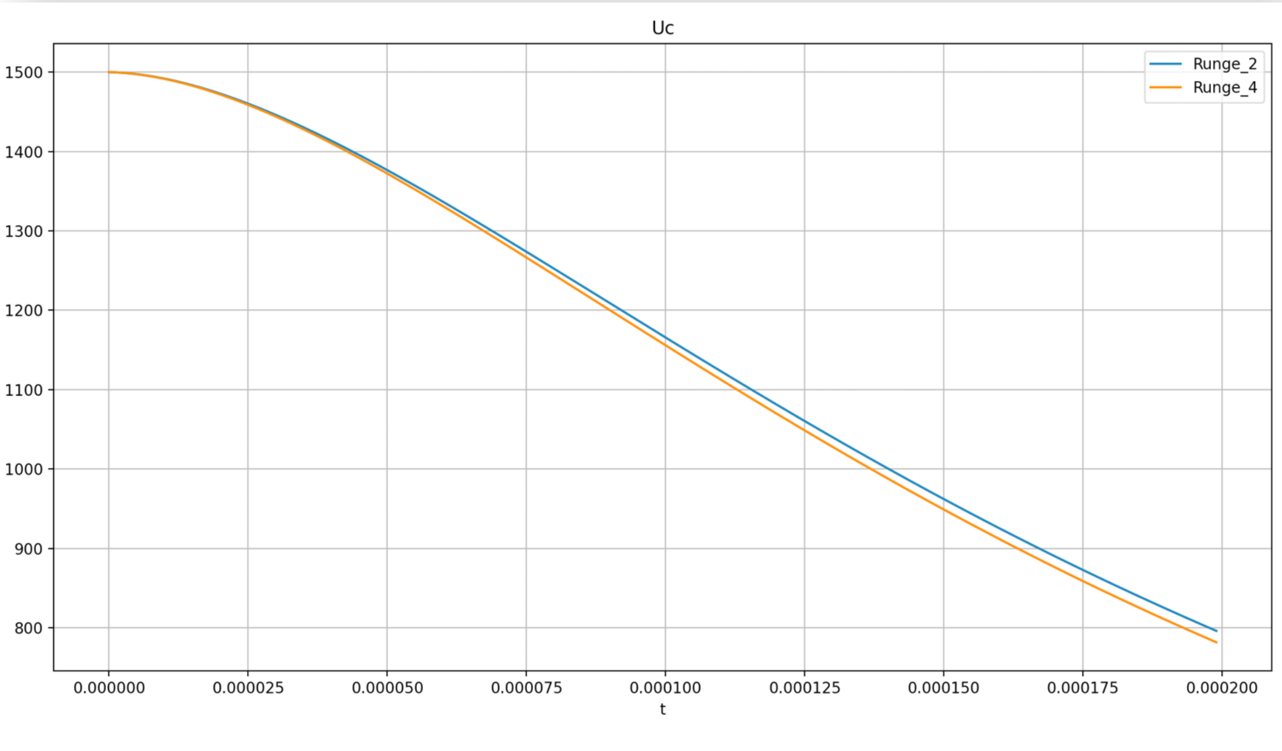
```
                  [6000,  2.05],
                  [7000,  6.06],
                  [8000,  12.0],
                  [9000,  19.9],
                  [10000, 29.6],
                  [11000, 41.1],
                  [12000, 54.1],
                  [13000, 67.7],
                  [14000, 81.5]]


def f(xn, yn, zn):
    return (zn - (Rk + Rp) * yn) / Lk


def g(xn, yn, zn):
    return -yn / Ck

def getRp(I, Rps):
    z = linspace(0.0, 1.0, 20)
    s = [z * sigma(get_T(z, I)) for z in z]
    global Rp
    Rp = le / (2 * pi * r ** 2 * integrate.simps(s, z))
    Rps.append(Rp)

    return Rp


def sigma(T):
    return interpolate(T, table(2), 1, 0, T)


def runge4(xn, yn, zn):
    Rp = getRp(yn, Rps4)
    k1 = h * f(xn, yn, zn); q1 = h * g(xn, yn, zn)
    k2 = h * f(xn + h / 2, yn + k1 / 2, zn + q1 / 2)
    q2 = h * g(xn + h / 2, yn + k1 / 2, zn + q1 / 2)
    k3 = h * f(xn + h / 2, yn + k2 / 2, zn + q2 / 2)
    q3 = h * g(xn + h / 2, yn + k2 / 2, zn + q2 / 2)
    k4 = h * f(xn + h, yn + k3, zn + q3)
    q4 = h * g(xn + h, yn + k3, zn + q3)

    yn1 = yn + (k1 + 2 * k2 + 2 * k3 + k4) / 6
    zn1 = zn + (q1 + 2 * q2 + 2 * q3 + q4) / 6

    return yn1, zn1

def runge2(xn, yn, zn):
    Rp = getRp(yn, Rps2)
    alpha = 0.5
    yn1 = yn + h * ((1 - alpha) * f(xn, yn, zn) + alpha *
                f(xn + h / (2 * alpha),
                  yn + h / (2 * alpha) * f(xn, yn, zn),
                  zn + h / (2 * alpha) * g(xn, yn, zn)))

    zn1 = zn + h * ((1 - alpha) * g(xn, yn, zn) + alpha *
                g(xn + h / (2 * alpha),
                  yn + h / (2 * alpha) * f(xn, yn, zn),
```

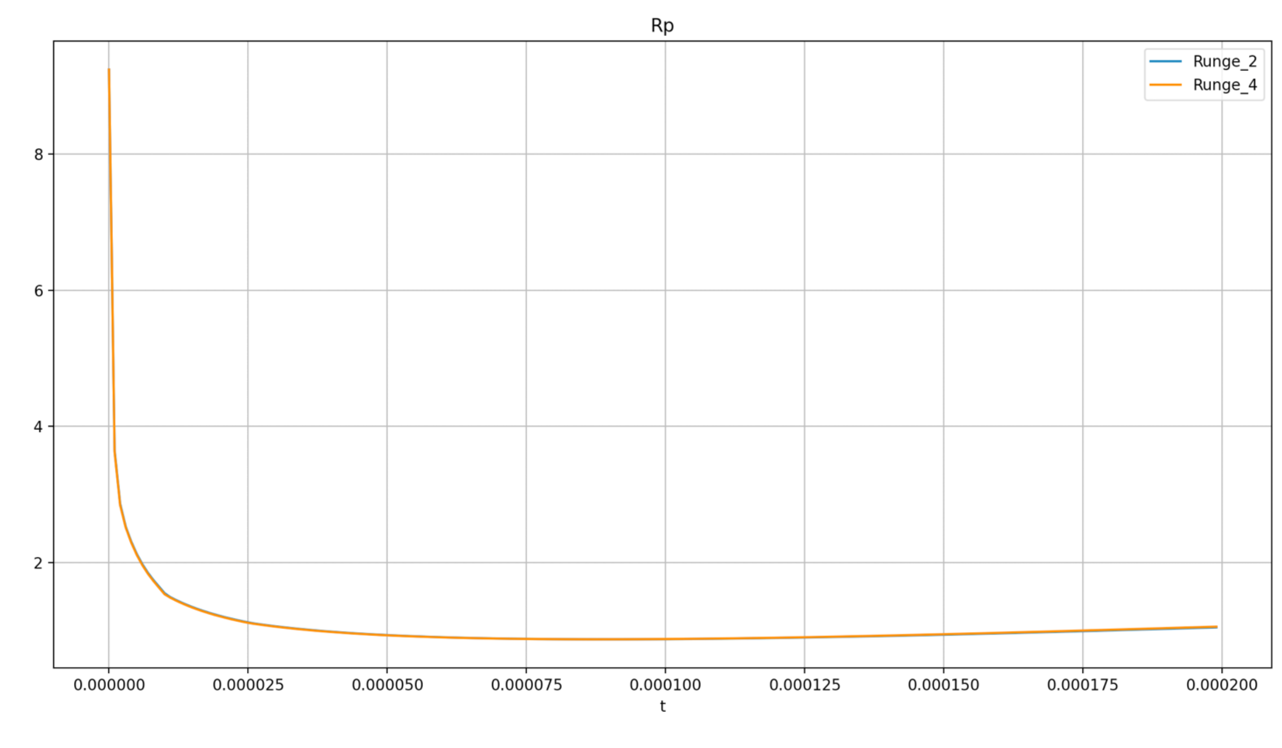
```
                 zn + h / (2 * alpha) * g(xn, yn, zn)))

    return yn1, zn1

def get_T0(I):
    return interpolate(I, table(1), 1, 0, I)

def get_T(z, I):
    T0 = get_T0(I)
    n = get_n(T0)
    T = T0 + (Tw - T0) * z ** n
    return T

def get_n(I):
    return interpolate(I, table(1), 2, 0, I)

def interpolate(purpose, table, first, second, value):
    i = 0;
    while purpose > table[i][0] and i < len(table) - 2:
        i += 1
    if i == 0:
        a, b = table[0], table[1]
    elif i == len(table) - 1:
        a, b = table[-2], table[-1]
    else:
        a, b = table[i - 1], table[i]
    return a[first] + (b[first] - a[first]) * \
           (value - a[second]) / (b[second] - a[second])


steps = 200
Tw = 2000.0
le = 12.0
r = 0.35
h = 1e-6
Ck = 150e-6
Lk = 60e-6
Rk = 1.0
I0 = 0.5
U0 = 1500.0
Rps2 = []
Rps4 = []

def plot(timings, first, second, xlabel, title, legend1, legend2):
    plt.plot(timings, first, timings, second)
    plt.xlabel(xlabel)
    plt.title(title)
    plt.legend((legend1, legend2))
    plt.grid(True)
    plt.show()

if __name__ == '__main__':
    In2 = I0; Un2 = U0
    In4 = I0; Un4 = U0

    I2 = []; Uc2 = []
    I4 = []; Uc4 = []

    for i in range(steps):
```

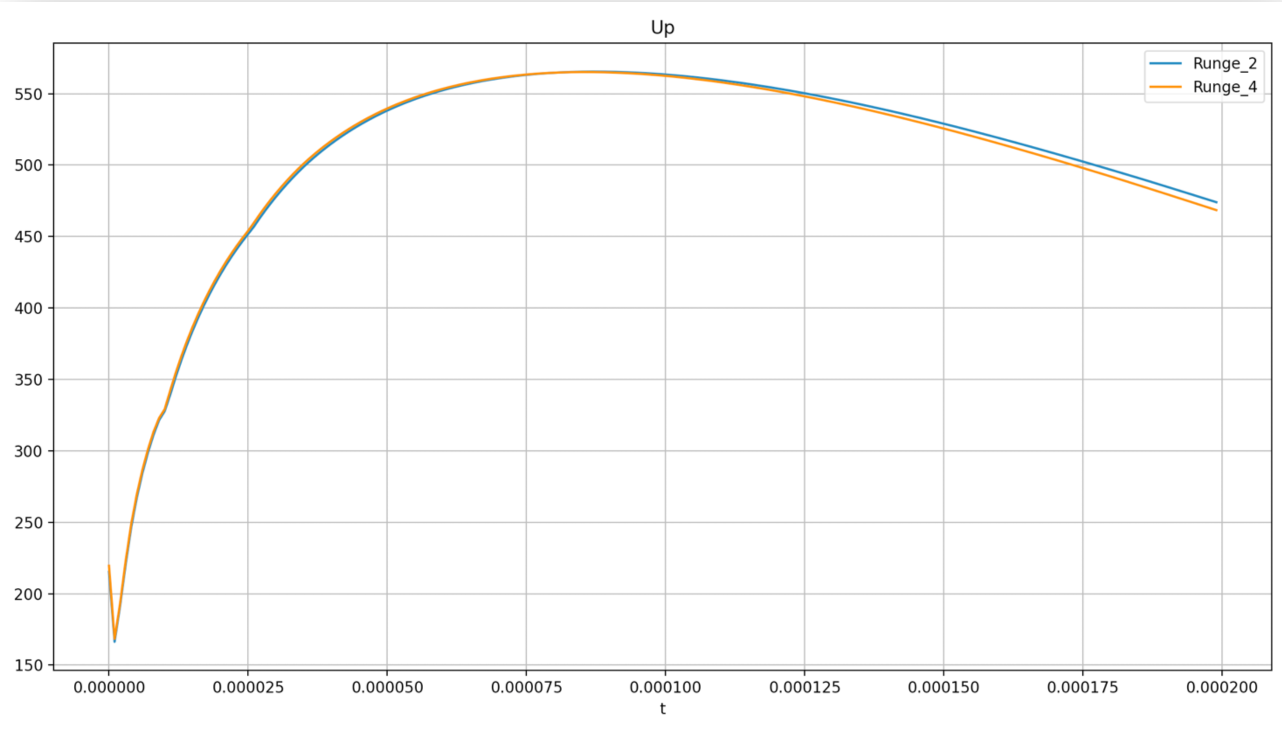
```
    In2, Un2 = runge2(h, In2, Un2)
    In4, Un4 = runge4(h, In4, Un4)

    I2.append(In2)
    Uc2.append(Un2)
    I4.append(In4)
    Uc4.append(Un4)

Up2 = [I2[i] * Rps2[i] for i in range(steps)]
Up4 = [I4[i] * Rps4[i] for i in range(steps)]

timings= [h * i for i in range(steps)]

plot(timings, I2, I4, 't', 'I', "Runge_2", "Runge_4")
plot(timings, Uc2, Uc4, 't', 'Uc', "Runge_2", "Runge_4")
plot(timings, Rps2, Rps4, 't', 'Rp', "Runge_2", "Runge_4")
plot(timings, Up2, Up4, 't', 'Up', "Runge_2", "Runge_4")
```

Рисунки с результатом работы программы






Rp
Runge_2
Runge_4
8
6
4
2
0.000000
0.000025
0.000050
0.000075
0.000100
0.000125
0.000150
0.000175
0.000200
t


Up
Runge_2
Runge_4
550
500
450
400
350
300
250
200
150
0.000000
0.000025
0.000050
0.000075
0.000100
0.000125
0.000150
0.000175
0.000200
t

Заключение

В ходе лабораторной работы были получены навыки по применению численного метода Рунге-Кутта для решения системы дифференциальных уравнений. Можно заметить, что метод Рунге-Кутте 4-ого порядка точнее метода Рунге-Кутте 2-ого порядка. Точность вычисления можно протестировать с помощью использования неявного метода трапеций, так как он приближает вычисляемые значения к истинным в пределах возможности вычислительной машины.