Цель работы: изучение метода Рунге-Кутта 2-го и 4-го порядка для решения системы дифференциальных уравнений.

1 Теоретическая часть

В данном разделе рассматривается метод Рунге-Кутта 2-го и 4-го порядка.

1.1 Метод Рунге-Кутта 2-го порядка для системы ОДУ

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

Обычно α задается равным 1 или 0.5.

1.2 Метод Рунге-Кутта 4-го порядка для системы ОДУ

$$I_{n+1} = I_n + \Delta t \cdot \frac{k_1 + 2 \cdot k_2 + 2 \cdot k_3 + k_4}{6}$$
$$U_{n+1} = U_n + \Delta t \cdot \frac{q_1 + 2 \cdot q_2 + 2 \cdot q_3 + q_4}{6}$$

$$\begin{cases} k_1 = h_n \cdot f(\Delta t, I_n, U_n) \\ q_1 = h_n \cdot \varphi(\Delta t, I_n, U_n) \\ k_2 = h_n \cdot f(\Delta t + \frac{h_n}{2}, I_n + \frac{k_1}{2}, U_n + \frac{q_1}{2}) \\ q_2 = h_n \cdot \varphi(\Delta t + \frac{h_n}{2}, I_n + \frac{k_1}{2}, U_n + \frac{q_1}{2}) \\ k_3 = h_n \cdot f(\Delta t + \frac{h_n}{2}, I_n + \frac{k_2}{2}, U_n + \frac{q_2}{2}) \\ q_3 = h_n \cdot \varphi(\Delta t + \frac{h_n}{2}, I_n + \frac{k_2}{2}, U_n + \frac{q_2}{2}) \\ k_4 = h_n \cdot f(\Delta t + h_n, I_n + k_3, U_n + q_3) \\ q_4 = h_n \cdot \varphi(\Delta t + h_n, I_n + k_3, U_n + q_3) \end{cases}$$

2 Практическая часть

Опишем колебательный контур с помощью системы уравнений:

$$\begin{cases} L_k \frac{dI}{dt} + (R_k + R_p(I)) \cdot I - U_C = 0 \\ \frac{dU_c}{dt} = -\frac{I}{C_k} \end{cases}$$

Значение Rp(I) можно вычислить по формуле:

$$Rp = \frac{l_e}{2\pi \cdot \int_0^R \sigma(T(r)) r dr} = \frac{l_e}{2\pi R^2 \cdot \int_0^1 \sigma(T(z)) dz}$$

Т.к. $z = r/R$

Значение T(z) вычисляется по формуле:

$T(z) = T_0 + (T_w - T_0) \cdot Z^m$

Заданы начальные параметры:

R = 0.35 см (Радиус трубки)
Le = 12 см (Расстояние между электродами лампы)
Lk = 187e-6 Гн (Индуктивность)
Ck = 268e-6 Ф (Емкость конденсатора)
Rk = 0.25 Ом (Сопротивление)
Uc0 = 1400 В (Напряжение на конденсаторе в начальный момент времени)
I0 = 0 .. 3 А (Сила тока в цепи в начальный момент времени t = 0)

Параметры можно менять из интерфейса.

В листинге 1 представлена реализация расмотренных методов.

Листинг 1: Листинг программы

```
from matplotlib import pyplot as plt
from numpy import arange
from math import pi
from scipy import integrate
from scipy.interpolate import InterpolatedUnivariateSpline


table1 = [[0.5, 6400, 0.4],
          [1.0, 6790, 0.55],
          [5.0, 7150, 1.7],
          [10.0, 7270, 3],
          [50.0, 8010, 11],
          [200.0, 9185, 32],
          [400.0, 10010, 40],
          [800.0, 11140, 41],
          [1200.0, 12010, 39]]


table2 = [[4000,  0.031],
          [5000, 0.27],
          [6000, 2.05],
          [7000, 6.06],
          [8000, 12],
          [9000, 19.9],
          [10000, 29.6],
          [11000, 41.1],
          [12000, 54.1],
          [13000, 67.7],
          [14000, 81.5]]


def f(x, y, z, Rp):
    return -((Rk + Rp) * y - z) / Lk

```

```
def phi(x, y, z):
    return -y / Ck


def interpolate(x, x_mas, y_mas):
    order = 1
    s = InterpolatedUnivariateSpline(x_mas, y_mas, k=order)
    return float(s(x))


def T(z):
    return T0 + (Tw - T0) * z ** m


def sigma(T):
    T_from_table = []
    for i in range (len(table2)):
        T_from_table.append(table2[i][0])

    sigm_from_table = []
    for j in range(len(table2)):
        sigm_from_table.append(table2[j][1])

    return interpolate(T, T_from_table, sigm_from_table)


def Rp(I):
    global m
    global T0

    I_from_table = []
    for i in range(len(table1)):
        I_from_table.append(table1[i][0])

    T0_from_table = []
    for j in range(len(table1)):
        T0_from_table.append(table1[j][1])

    m_from_table = []
    for z in range(len(table1)):
        m_from_table.append(table1[z][2])

    m = interpolate(I, I_from_table, m_from_table)
    T0 = interpolate(I, I_from_table, T0_from_table)

    func = lambda z: sigma(T(z)) * z
    integral = integrate.quad(func, 0, 1)
    Rp = Le / (2 * pi * R ** 2 * integral[0])

    return Rp


def runge_kutta_second_order(xn, yn, zn, hn, Rp):
```

```
    alpha = 0.5
    y_next = yn + hn * ((1 - alpha) * f(xn, yn, zn, Rp) +
        alpha * f(xn + hn / (2 * alpha),
                yn + hn / (2 * alpha) * f(xn, yn, zn, Rp),
                zn + hn / (2 * alpha) * phi(xn, yn, zn), Rp))

    z_next = zn + hn * ((1 - alpha) * phi(xn, yn, zn) + alpha
        * phi(xn + hn / (2 * alpha),
                yn + hn / (2 * alpha) * f(xn, yn, zn, Rp),
                zn + hn / (2 * alpha) * phi(xn, yn, zn)))

    return y_next, z_next


def runge_kutta_fourth_order(xn, yn, zn, hn, Rp):
    k1 = hn * f(xn, yn, zn, Rp)
    q1 = hn * phi(xn, yn, zn)

    k2 = hn * f(xn + hn / 2, yn + k1 / 2, zn + q1 / 2, Rp)
    q2 = hn * phi(xn + hn / 2, yn + k1 / 2, zn + q1 / 2)

    k3 = hn * f(xn + hn / 2, yn + k2 / 2, zn + q2 / 2, Rp)
    q3 = hn * phi(xn + hn / 2, yn + k2 / 2, zn + q2 / 2)

    k4 = hn * f(xn + hn, yn + k3, zn + q3, Rp)
    q4 = hn * phi(xn + hn, yn + k3, zn + q3)

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

    return y_next, z_next

if __name__ == "__main__":
    R = 0.35
    Le = 12
    T0 = 0.0
    Tw = 2000.0
    m = 0.0

    Ck = 150e-6
    Lk = 60e-6
    Rk = 1
    Uc0 = 1500.0
    I0 = 0.5

    I_2order = I_4order = I0
    Uc_2order = Uc_4order = Uc0

    t_plot = []

    I_2order_plot = []
    U_2order_plot = []
    Rp_2order_plot = []
    T_2order_plot = []
```

```

    I_4order_plot = []
    U_4order_plot = []
    Rp_4order_plot = []
    T_4order_plot = []

    h = 1e-6
    for t in arange(0, 0.0003, h):
        Rp_2order = Rp(I_2order)
        Rp_4order = Rp(I_4order)
        if t > h:
            t_plot.append(t)
            I_2order_plot.append(I_2order)
            T_2order_plot.append(T0)
            U_2order_plot.append(Uc_2order)
            Rp_2order_plot.append(Rp_2order)
            I_4order_plot.append(I_4order)
            T_4order_plot.append(T0)
            U_4order_plot.append(Uc_4order)
            Rp_4order_plot.append(Rp_4order)
        I_2order, Uc_2order = runge_kutta_second_order(t,
            I_2order, Uc_2order, h, Rp_2order)
        I_4order, Uc_4order = runge_kutta_fourth_order(t,
            I_4order, Uc_4order, h, Rp_4order)
```

3 Результат работы программы

На рис. 1 и 2 представлен результат работы программы:

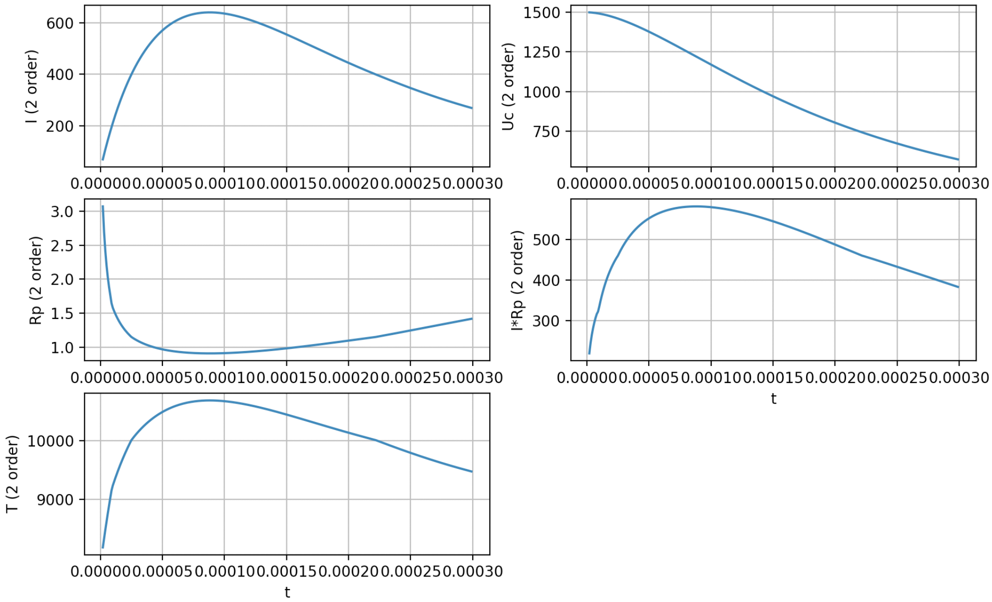

Рис. 1: Метод Рунге-Кутта 2-го порядка

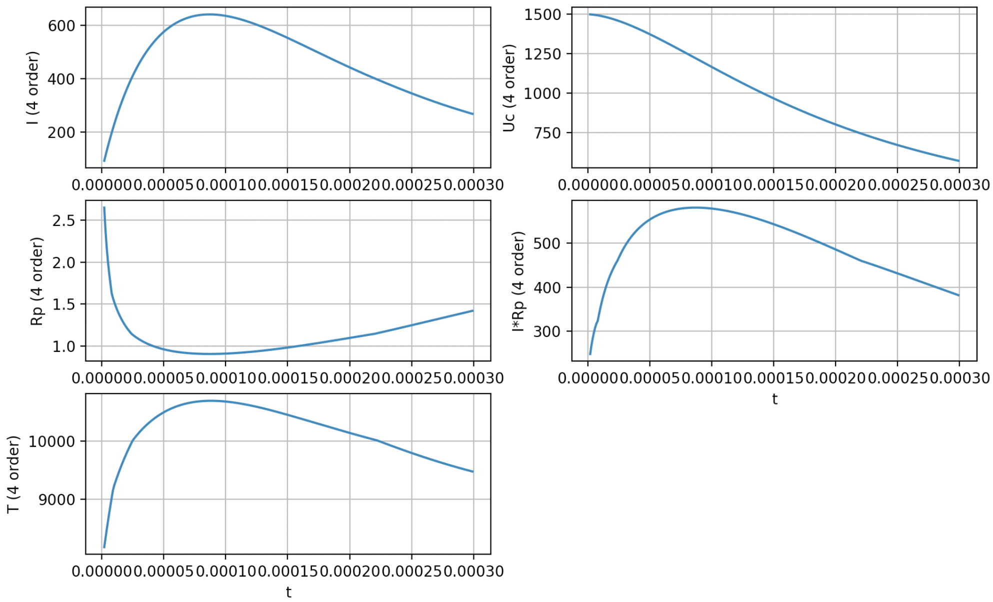

Рис. 2: Метод Рунге-Кутта 4-го порядка

Вывод

В ходе лабораторной работы были получены навыки по применению численного метода Рунге-Кутта для решения системы обыкновенных дифференциальных уравнений..