	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: Гасанзаде М.А.

Группа ИУ7-66Б

Оценка (баллы) _______________

Преподаватель : Градов В.М.

Москва.
2020 г.

СОДЕРЖАНИЕ

I. АНАЛИТИЧЕСКАЯ ЧАСТЬ..3

Метод Рунге-Кутта 4-го порядка для системы ОДУ..5

Метод Рунге-Кутта 2-го порядка для системы ОДУ..5

II. ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ...6

III. ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ..9

ЗАКЛЮЧЕНИЕ...10

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ...11

I. АНАЛИТИЧЕСКАЯ ЧАСТЬ.

Введение: Дан колебательный контур с газоразрядной трубкой.

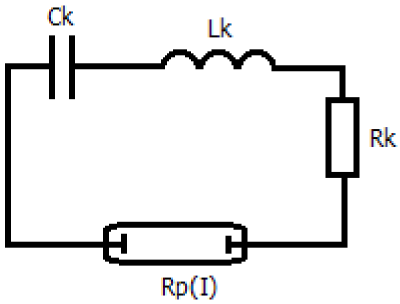

Рисунок 1. Колебательный контур.

Цель работы: получение навыков решения системы дифференциальных уравнений с помощью численного метода Рунге-Кутта 2-го и 4-го порядка

Данный контур можно описать при помощи системы уравнений:

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

Ее необходимо решить (численно) и построить графики:

- `I(t) - сила тока в цепи`
- `Uc(t) - напряжение на конденсаторе`
- `Ucp(t) - напряжение на газоразрядной трубке`
- `Rp(t) - сопротивление лампы`

Значение Rp(I) можно вычислить по формуле:

$$Rp = \frac{L_e}{2\pi \int_0^R \sigma(T(r)) r dr} = \left| z = \frac{r}{R} = \frac{L_e}{2\pi R^2 \cdot \int_0^1 \sigma(T(z)) dz} \right.$$

Значение T(z) вычисляется по формуле:

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

Параметры могут меняться из интерфейса:

- `Rk = 0.2 Ом (Сопротивление)`

- Lk = 60e-6 Гн (Индуктивность)
- Ck = 150e-6 Ф (Емкость конденсатора)
- R = 0.35 см (Радиус трубки)
- T0 = 300 К
- Tw = 2000К
- le = 12 см (Расстояние между электродами лампы)
- Uc0 = 3000 В (Напряжение на конденсаторе в начальный момент времени t = 0)
- I0 = 0 .. 2 А (Сила тока в цепи в начальный момент времени t = 0)

Даны таблицы (1-2), по которым, зная силу тока, можно найти значения T_0, m а также σ :

Таблица №1.

I	T_0	m
0.5	6400	0.40
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

Таблица №2.

T	σ
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

Для решения системы дифференциальных уравнений воспользуемся методом Рунге-Кутта разных порядков:

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

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

$$U_{n+1} = U_n + \Delta t \cdot \frac{q_1 + 2 \cdot q_2 + 2 \cdot q3 + q_4}{6}$$

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

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

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

где α задается как 0.5 либо 1.

II. ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

ЯП был выбран `Python3` из-за простоты работы с графиками и библиотеки `matplotlib`. Ниже на листингах будет представлена реализация программы:

```
import numpy
import matplotlib.pyplot as plt
from decimal import Decimal
from scipy import integrate
from scipy.interpolate import InterpolatedUnivariateSpline
# одномерная кусочно-линейная интерполяция функции, которая задана точками
(xp, fp).

# Table1
masI = [0.5,    1,     5,    10,    50,   200,    400,    800,   1200]
masT0 = [6400, 6790, 7150, 7270, 8010, 9185, 10010, 11140, 12010]
masm = [0.4, 0.55,  1.7,     3,    11,    32,     40,     41,     39]

# Table2
masT = [4000, 5000, 6000, 7000, 8000, 9000, 10000, 11000, 12000, 13000,
14000]
masSigm = [0.031, 0.27, 2.05, 6.06, 12.0,
           19.9,  29.6,  41.1,  54.1,  67.7,  81.5]


def interpolate(x, masX, masY):
    order = 1
    s = InterpolatedUnivariateSpline(masX, masY, k=order)
    return float(s(x))


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


def sigma(T):
    return interpolate(T, masT, masSigm)


def Rp(I):
    global m
    global T0
    m = interpolate(I, masI, masm)
    T0 = interpolate(I, masI, masT0)

    def func(z): return sigma(T(z)) * z
    integral = integrate.quad(func, 0, 1)
    Rp = le/(2 * numpy.pi * R**2 * integral[0])

    return Rp


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


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

```python
def second_order(xn, yn, zn, hn, m_Rp):
    global m_Rp_global

    m_Rp_global = m_Rp

    alpha = 0.5
    yn_1 = yn + hn * ((1 - alpha) * f(xn, yn, zn) + alpha
                      * f(xn + hn/(2*alpha),
                          yn + hn/(2*alpha) * f(xn, yn, zn),
                          zn + hn/(2*alpha) * phi(xn, yn, zn)))

    zn_1 = zn + hn * ((1 - alpha) * phi(xn, yn, zn) + alpha
                      * phi(xn + hn/(2*alpha),
                            yn + hn/(2*alpha) * f(xn, yn, zn),
                            zn + hn/(2*alpha) * phi(xn, yn, zn)))

    return yn_1, zn_1


def fourth_order(xn, yn, zn, hn, m_Rp):
    global m_Rp_global
    m_Rp_global = m_Rp

    k1 = hn * f(xn, yn, zn)
    q1 = hn * phi(xn, yn, zn)

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

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

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

    yn_1 = yn + (k1 + 2*k2 + 2*k3 + k4)/6
    zn_1 = zn + (q1 + 2*q2 + 2*q3 + q4)/6

    return yn_1, zn_1


def do_plot(pltMasT, mas1, mas2, xlabel, ylabel, name1, name2):
    plt.plot(pltMasT, mas1)
    plt.plot(pltMasT, mas2)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.legend((name1, name2))
    plt.grid(True)
   plt.show()


if __name__ == "__main__":
    R = 0.35
    Tw = 2000.0
    Ck = 150e-6
    Lk = 60e-6
    Rk = 1  # от 0.5 до 200
    Uc0 = 1500.0
    I0 = 0.5  # от 0.5 до 3
    le = 12.0
```

```
    I4 = I0
    Uc4 = Uc0
    I2 = I0
    Uc2 = Uc0

    T0 = 0.0
    m = 0.0

    pltMasT = []
    pltMasI4 = []
    pltMasU4 = []
    pltMasRp4 = []
    pltMasI2 = []
    pltMasU2 = []
    pltMasRp2 = []

    h = 1e-6
    for t in numpy.arange(0, 0.0003, h):
        try:
            m_Rp4 = Rp(I4)
            m_Rp2 = Rp(I2)

            if t > h:
                pltMasT.append(t)
                pltMasI4.append(I4)
                pltMasU4.append(Uc4)
                pltMasRp4.append(m_Rp4)
                pltMasI2.append(I2)
                pltMasU2.append(Uc2)
                pltMasRp2.append(m_Rp2)

            I4, Uc4 = fourth_order(t, I4, Uc4, h, m_Rp4)
            I2, Uc2 = second_order(t, I2, Uc2, h, m_Rp2)

        except:
            break

    do_plot(pltMasT, pltMasI4,  pltMasI2, 't',  'I', '4th order', '2nd order')
    do_plot(pltMasT, pltMasU4,  pltMasU2, 't', 'Uc', '4th order', '2nd order')
    do_plot(pltMasT, pltMasRp4, pltMasRp2, 't', 'Rp', '4th order', '2nd order')

    for i in range(len(pltMasI4)):
        pltMasI4[i] *= pltMasRp4[i]
        pltMasI2[i] *= pltMasRp2[i]

    do_plot(pltMasT, pltMasI4, pltMasI2, 't', 'Up', '4th order', '2nd order')
```

III. ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ

В данном разделе будет рассмотрен вывод программы и представлены графики зависимостей.

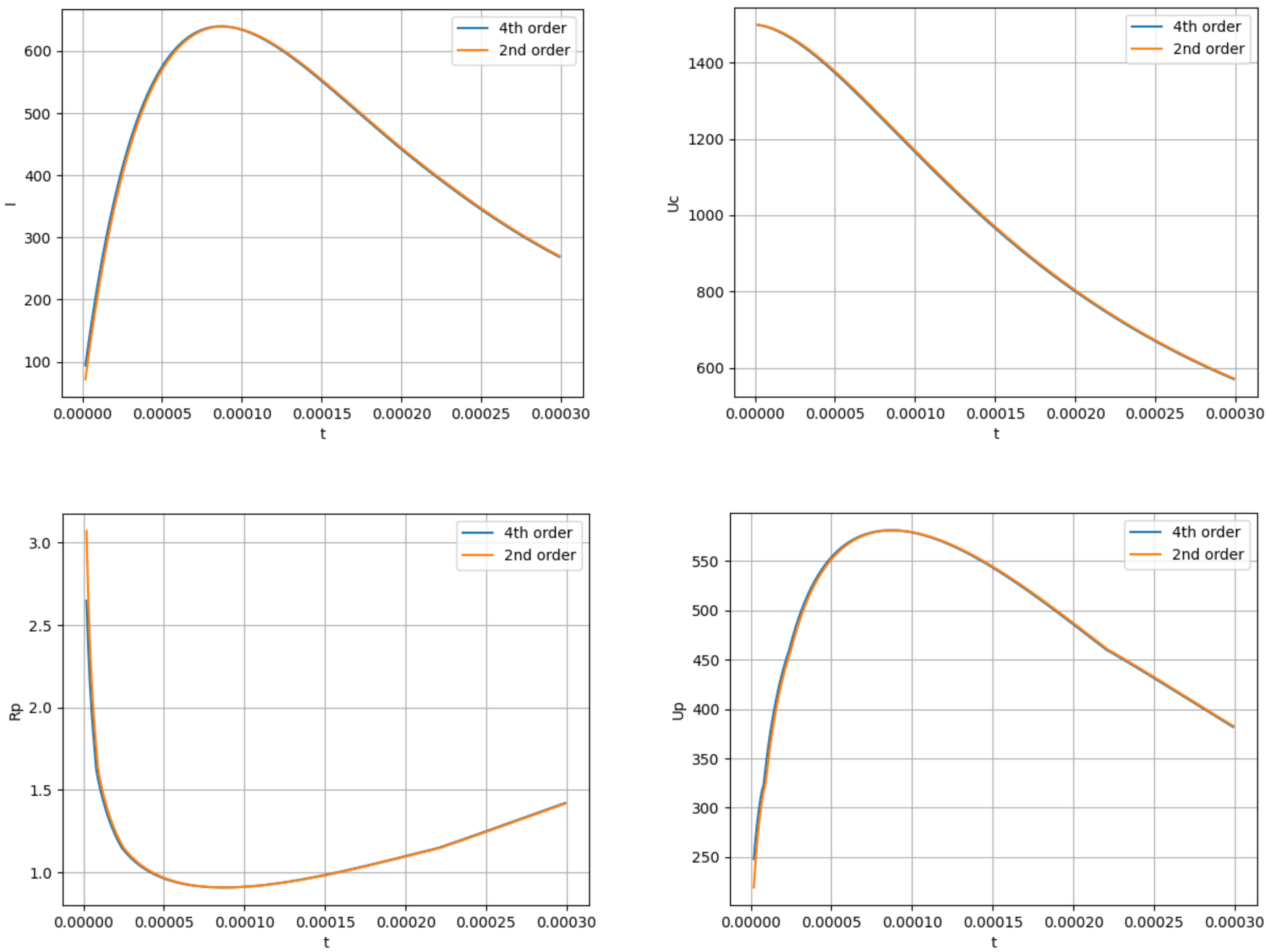

Рисунок 1. Результат работы программы — Графики зависимости.

ЗАКЛЮЧЕНИЕ

В ходе лабораторной работы были получены навыки по применению численного метода Рунге-Кутта для решения системы дифференциальных уравнений, проанализированы разные ситуации со входными данными.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Schema URL: https://draw.io (дата обращения 10.03.2020)
2. Градов В.М. Курс лекций по Моделированию — 2020
3. Matplotlib URL: https://matplotlib.org (дата обращения 10.03.2020)