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 Юмаев А.Р.

Группа ИУ7-65Б

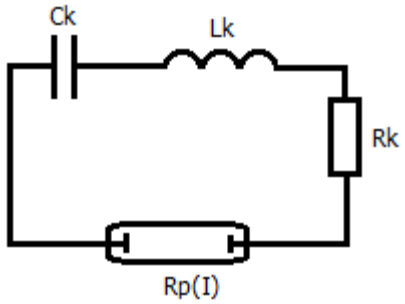
Оценка (баллы) _____

Преподаватель Градов В.М.

Москва.
2020 г.

Условие

Дан колебательный контур с газоразрядной трубкой



Из некоторых физических соображений получена система дифференциальных уравнений:

$$L_k \frac{dI}{dt} + (R_k + R_p(I))I - U_c = 0$$

$$\frac{dUc}{dt} = -\frac{I}{Ck}$$

Ее необходимо решить (численно) и построить графики:

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

Rp, оно же сопротивление газоразрядной трубки, находится в зависимости от силы тока:

$$R_p = \frac{l_e}{2\pi \int_0^R \sigma(T(r), p) r\, dr}$$

Давление p в формуле находится из следующего уравнения:

$$\frac{2}{R^2} \int_0^R n(T(r), p) r\, dr - \frac{p_{0[atm]} \cdot 7242}{T_{start}} = 0$$

Функция T(r):

$$T(r) = (T_w - T_0)(r/R)^m + T_0$$

Электропроводность σ(T, p), концентрация тяжелых частиц n(T, p), а также T0(I) и m(I) заданы таблично.

Входные параметры:

- Rk – Сопротивление резистора, Ом
- Lk = Индуктивность катушки, Гн
- Ck = Емкость конденсатора, Ф
- R = Радиус трубки лампы, см
- T0 = температура в начальный момент времени, К
- Tw = температура в конечный момент времени, К
- le = Расстояние между электродами лампы, см
- Uc0 = Напряжение на конденсаторе в начальный момент времени, В
- I0 = Сила тока в цепи в начальный момент времени, А

Даны таблицы, по которым, зная силу тока, можно найти значения T_0, m и σ:

I	T_0	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

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

Метод Рунге-Кутта 4-го порядка для системы ОДУ

Для начала выразим значения производных dI/dt и dUc/dt и для удобства обозначим их соответсвенно f и g.

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

$$\frac{dU_c}{dt} = -\frac{1}{C_k}I \equiv g(I)$$

Решение системы заключается в применении формул:

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

Коэффициенты k, m считаются следующим образом:

$$k_1 = f(I_n, U_{c_n}), m_1 = g(I_n)$$

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

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

$$k_4 = f(I_n + \Delta tk_3, U_{c_n} + \Delta tm_3), m_4 = g(I_n + \Delta tk_3)$$

Помимо I(t) и Uc(t), которые тут вычисляются, надо также вывести:

- Rp(t) - по сути, это просто значение сопротивления Rp для соответствующего значения силы тока I
- Ucp(t) - это напряжение на трубке, считается как Ucp = I * Rp

Метод Рунге-Кутта 2-го порядка для системы ОДУ

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

Листинг программы

```
import numpy
import matplotlib.pyplot as plt
from decimal import Decimal
from scipy import integrateimport numpy
import matplotlib.pyplot as plt
from decimal import Decimal
from scipy import integrate
from scipy.interpolate import InterpolatedUnivariateSpline

import matplotlib.style
import matplotlib as mpl
mpl.style.use('grayscale')

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]

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)

    func = lambda z: 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
##    return zn/Lk

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

def Runge2(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 Runge4(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 plot_data(pltMasT, mas1, mas2, xlabel, ylabel, name1, name2):
    plt.plot(pltMasT, mas1, 'r')
    plt.plot(pltMasT, mas2, 'b')
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.legend((name1, name2))
    plt.grid(True)
    plt.show()


print("1.Автоматичекий ввод констант\n2.Ручной ввод")
R = 0
Tw = 0
Ck = 0
Lk = 0
Rk = 0
Uc0 = 0
I0 = 0
le = 0


while True:
    x = input('>>')
    if x =='1':
        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
        break
    elif x == '2':
        R = float(input('Радиус (см): '))
        Tw = float(input('Температура (К): '))
        Ck = float(input('Ёмкость конденсатора (Ф): '))
        Lk = float(input('Индуктивность катушки (Гн): '))
        Rk = float(input('Сопротивление резистора(Ом): '))
        Uc0 = float(input('Напряжение при t=0 сек (В): '))
        I0 = float(input('Сила тока при t=0 сек (А): '))
        le = float(input('Расстояние между пластинами (см): '))
        break
    else:
        print('Неправильный ввод')
```

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

T0 = 0.0
m = 0.0

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

print("1.Автоматичекий ввод шага\n2.Ручной ввод")
x = input('>>')
h = 0
while True:
    if x =='1':
        h = 1e-6
    elif x == '2':
        h = float(input("Шаг: "))
        break
    else:
        print('Неправильный ввод')

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 = Runge4(t, I4, Uc4, h, m_Rp4)
        I2, Uc2 = Runge2(t, I2, Uc2, h, m_Rp2)

    except:
        break

plot_data(pltMasT, pltMasI4,  pltMasI2, 't, сек',  'I, A', '4-й порядок', '2-й
порядок')
plot_data(pltMasT, pltMasU4,  pltMasU2, 't, сек', 'Uc, B', '4-й порядок', '2-й
порядок')
plot_data(pltMasT, pltMasRp4, pltMasRp2,'t, сек', 'Rp, Ом', '4-й порядок', '2-й
порядок')

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

plot_data(pltMasT, pltMasI4, pltMasI2, 't, сек', 'Up, B', '4-й порядок', '2-й порядок')

from scipy.interpolate import InterpolatedUnivariateSpline

import matplotlib.style
import matplotlib as mpl
mpl.style.use('grayscale')

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]

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)

    func = lambda z: 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
##    return zn/Lk

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

def Runge2(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 Runge4(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 plot_data(pltMasT, mas1, mas2, xlabel, ylabel, name1, name2):
    plt.plot(pltMasT, mas1, 'r')
    plt.plot(pltMasT, mas2, 'b')
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.legend((name1, name2))
    plt.grid(True)
    plt.show()


print("1.Автоматичекий ввод констант\n2.Ручной ввод")
R = 0
Tw = 0
Ck = 0
Lk = 0
Rk = 0
Uc0 = 0
I0 = 0
le = 0


while True:
    x = input('>>')
    if x =='1':
        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
        break
    elif x == '2':
        R = float(input('Радиус (см): '))
        Tw = float(input('Температура (К): '))
        Ck = float(input('Ёмкость конденсатора (Ф): '))
        Lk = float(input('Индуктивность катушки (Гн): '))
        Rk = float(input('Сопротивление резистора(Ом): '))
        Uc0 = float(input('Напряжение при t=0 сек (В): '))
        I0 = float(input('Сила тока при t=0 сек (А): '))
        le = float(input('Расстояние между пластинами (см): '))
        break
    else:
        print('Неправильный ввод')


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

T0 = 0.0
m = 0.0

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

print("1.Автоматичекий ввод шага\n2.Ручной ввод")
x = input('>>')
```

```
h = 0
while True:
    if x =='1':
        h = 1e-6
    elif x == '2':
        h = float(input("Шаг: "))
        break
    else:
        print('Неправильный ввод')

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 = Runge4(t, I4, Uc4, h, m_Rp4)
        I2, Uc2 = Runge2(t, I2, Uc2, h, m_Rp2)

    except:
        break

plot_data(pltMasT, pltMasI4,  pltMasI2, 't, сек',  'I, A', '4-й порядок', '2-й
порядок')
plot_data(pltMasT, pltMasU4,  pltMasU2, 't, сек', 'Uc, B', '4-й порядок', '2-й
порядок')
plot_data(pltMasT, pltMasRp4, pltMasRp2,'t, сек', 'Rp, Ом', '4-й порядок', '2-й
порядок')

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

plot_data(pltMasT, pltMasI4, pltMasI2, 't, сек', 'Up, B', '4-й порядок', '2-й порядок')
```

Результат работы программы

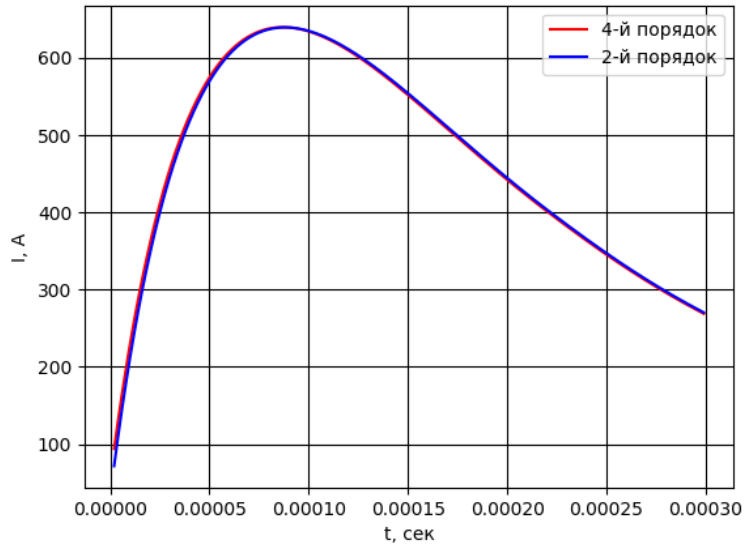


Рисунок 1. Зависимость силы тока в цепи

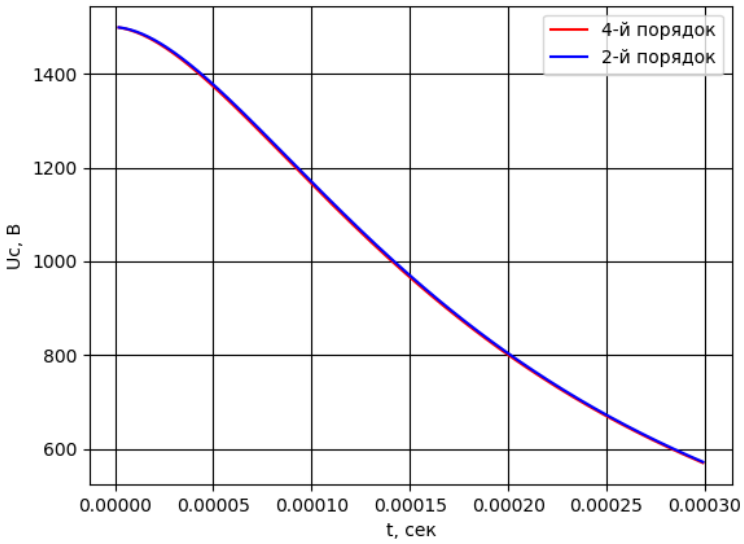


Рисунок 2. Зависимость напряжения на конденсаторе

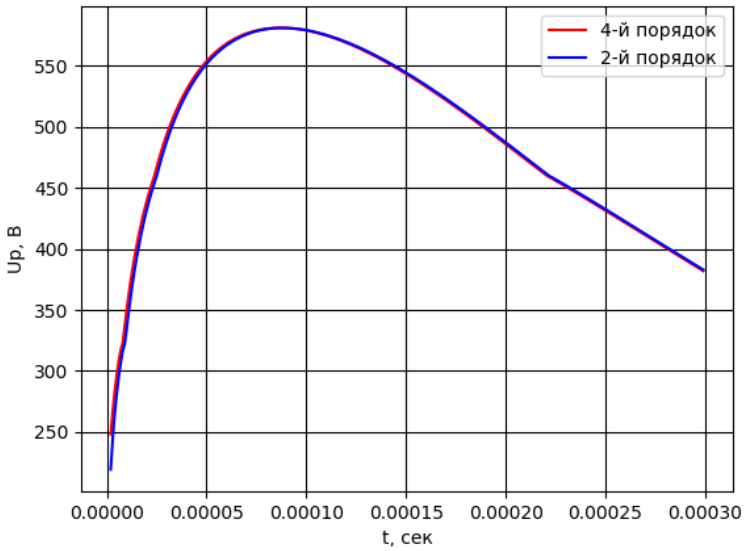


Рисунок 3. Зависимость напряжения на газоразрядной трубке

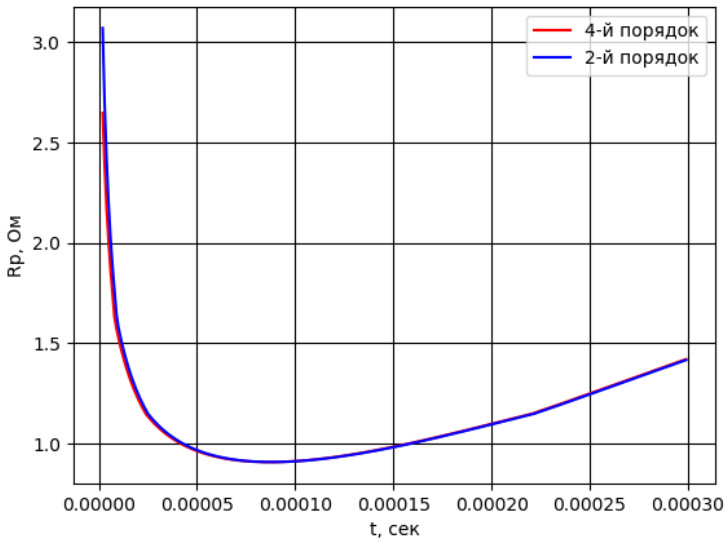


Рисунок 4. Зависимость сопротивления лампы

Заключение

В ходе лабораторной работы были получены навыки по применению численного метода Рунге-Кутта для решения системы дифференциальных уравнений. Можно заметить, что при повышении порядка метода Рунге-Кутта и уменьшении количества шагов, повышается точность вычисления.