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

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

Студент Сушина А.Д.

Группа ИУ7-61б

Оценка (баллы) _____

Преподаватель Градов В.М.

Москва.
2020 г

Оглавление

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.

I. Исходные данные.

Задана система электротехнических уравнений, описывающих разрядный контур, включающий постоянное активное сопротивление R_k, нелинейное сопротивление $R_p(I)$, зависящее от тока I, индуктивность L_k и емкость C_k.

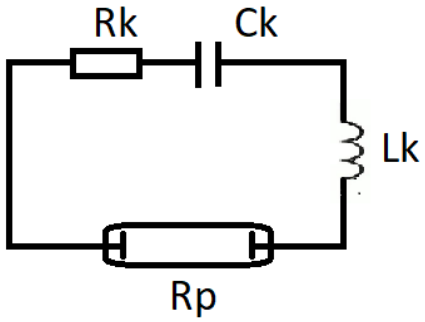

Рис 1. Разрядный контур

$$\begin{cases} \dfrac{dI}{dT} = \dfrac{U-(R_k+R_p(I))I}{L_k} \\ \dfrac{dU}{dt} = -\dfrac{I}{C_k} \end{cases}$$

Начальные условия: $t=0$, $I=I_0$, $U=U_0$.

Здесь I, U - ток и напряжение на конденсаторе.

Сопротивление R_p рассчитать по формуле

$$R_p = \frac{l_p}{2\pi R^2 \int_o^1 \sigma(T(z))\, z\, dz}.$$

Для функции $T(z)$ применить выражение $T(z)=T_0+(T_w-T_0)\,z^m$.

Параметры T_0, m находятся интерполяцией из табл.1 при известном токе I.

Коэффициент электропроводности $\sigma(T)$ зависит от T и рассчитывается интерполяцией из табл.2.

I, A	To, K	m
0.5	6730	0.50
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 1

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

Таблица 2

Параметры разрядного контура:

R=0.35 см
$l_э$=12 см
L_k=187 10^{-6} Гн
C_k=268 10^{-6} Ф
R_k=0.25 Ом
U_{co}=1400 В
I_o=0..3 А
T_w=2000 К

II. Аналитическая часть

II.i. Метод Рунге-Кутта второго порядка точности.

Для системы уравнений вида

$$\begin{cases} u'(x) = f(x, u(x)) \\ u(x_0) = y_0 \end{cases}$$

метод Рунге-Кутта второго порядка описывается следующим уравнением:

$$y_{n+1} = y_n + h_n * ((1-\alpha) * f(x_n, y_n) + \alpha * f(x_n + \frac{h_n}{2*\alpha}, y_n + \frac{h_n}{2*\alpha} * f(x_n, y_n)))$$

Где α – произвольный параметр, α ∈ [0, 1] (обычно α = 0.5 или α = 1)
При α = 0.5 получается неявный метод трапеций, при α = 1 – метод средних

II.ii. Метод Рунге-Кутта четвертого порядка точности.

Дана система уравнений вида

$$\begin{cases} u'(x) = f(x, u, v) \\ v'(x) = \varphi(x, u, v) \\ u(\xi) = \eta_1 \\ v(\xi) = \eta_2 \end{cases}$$

Тогда

$$y_{n+1} = y_n + \frac{k_1 + 2k_2 + 2k_3 + k_4}{6}$$

$$z_{n+1} = z_n + \frac{g_1 + 2g_2 + 2g_3 + g_4}{6}$$

, где

$$k_1 = h_n f(x_n, y_n, z_n) \qquad g_1 = h_n \varphi(x_n, y_n, z_n)$$

$$k_2 = h_n f\left(x_n + \frac{h_n}{2}, y_n + \frac{k_1}{2}, z_n + \frac{g_1}{2}\right) \qquad g_2 = h_n \varphi\left(x_n + \frac{h_n}{2}, y_n + \frac{k_1}{2}, z_n + \frac{g_1}{2}\right)$$

$$k_3 = h_n f\left(x_n + \frac{h_n}{2}, y_n + \frac{k_2}{2}, z_n + \frac{g_2}{2}\right) \qquad g_3 = h_n \varphi\left(x_n + \frac{h_n}{2}, y_n + \frac{k_2}{2}, z_n + \frac{g_2}{2}\right)$$

$$k_4 = h_n f(x_n + h_n, y_n + k_3, z_n + g_3) \qquad g_4 = h_n \varphi(x_n + h_n, y_n + k_3, z_n + g_3)$$

III. Технологическая часть

На листингах 1-3 приведен код программы.

Листинг 1. Метод Рунге-Кутта второго порядка точности.

```
def RungeKutta2(x0, y0, z0, h):
    alpha = 0.5
    nh = h / (2 * alpha)
    k1 = functionF_2(x0, y0, z0)
    q1 = functionPHI(x0, y0, z0)
    k2 = functionF_2(x0 + nh, y0 + nh * k1, z0 + nh * q1)
    q2 = functionPHI(x0 + nh, y0 + nh * k1, z0 + nh * q1)
    y1 = y0 + h * ((1 - alpha) * k1 + alpha * k2)
    z1 = z0 + h * ((1 - alpha) * q1 + alpha * q2)
    return y1, z1
```

Листинг 2. Метод Рунге-Кутта четвертого порядка точности.

```
def RungeKutta4(xn, yn, zn, hn):
    hn2 = hn / 2
    k1 = hn * functionF_4(xn, yn, zn)
    q1 = hn * functionPHI(xn, yn, zn)
    k2 = hn * functionF_4(xn + hn2, yn + k1 / 2, zn + q1 / 2)
    q2 = hn * functionPHI(xn + hn2, yn + k1 / 2, zn + q1 / 2)
    k3 = hn * functionF_4(xn + hn2, yn + k2 / 2, zn + q2 / 2)
    q3 = hn * functionPHI(xn + hn2, yn + k2 / 2, zn + q2 / 2)
    k4 = hn * functionF_4(xn + hn, yn + k3, zn + q3)
    q4 = hn * functionPHI(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
```

Листинг 3. Функции F и PHI

```
def functionF_2(t, I, U):
    Rp = calculateRp(I)
    return ((U - (data['Rk'] + Rp) * I) / data['Lk'])

def functionPHI(t, I, U):
    return -1 / data['Ck'] * I
```

Листинг 4. Вычисление Rp, метод Симпсона и интерполяция

```
def calculateRp(I):
    R = data['R']
    integral = integrateSimpson(I)
    return data['Le'] / (2 * math.pi * R * R * integral)

def integrateSimpson(I):
    n = 40
```

```
    begin = 0
    end = 1
    width = (end - begin) / n
    result = 0
    for step in range(n):
        x1 = begin + step * width
        x2 = begin + (step + 1) * width
        result += (x2 - x1) / 6.0 * (siqmaFunc(I, x1) + 4.0 * siqmaFunc(I, 0.5 * (x1 + x2)) + siqmaFunc(I, x2))
    return result

def siqmaFunc(I, z):
    m = interpolate(ItK, I, 0, 2)
    T0 = interpolate(ItK, I, 0, 1)
    Tz = getTz(T0, m, z)
    siqma = interpolate(Tsigma, Tz, 0, 1)
    return siqma

def getTz(T0, m, r):
    z = r / data['R']
    return (data['Tw'] - T0) * math.pow(z, m) + T0
```

Листинг 5. Интерполяция

```
def interpolate(table, xValue, xIndex, yIndex):
    interpolateIndexFound = False
    x1 = 0
    x2 = 0
    y1 = 0
    y2 = 0
    yResult = 0
    for i in range(len(table) - 1):
        if (table[i][xIndex] <= xValue and table[i + 1][xIndex] >= xValue):
            y1 = table[i][yIndex]
            y2 = table[i + 1][yIndex]
            x1 = table[i][xIndex]
            x2 = table[i + 1][xIndex]
            interpolateIndexFound = True
        if (interpolateIndexFound):
            yResult = y1 + ((xValue - x1) / (x2 - x1)) * (y2 - y1)
        else:
            if (xValue < table[0][xIndex]):
                yResult = table[0][yIndex]
            if (xValue > table[len(table) - 1][xIndex]):
                yResult = table[len(table) - 1][yIndex]
    return yResult
```

IV. Экспериментальная часть

IV.i. Результаты работы программы.

1. Графики зависимости от времени импульса t : $I(t)$, $U(t)$, $R_p(t)$, $I(t)\cdot R_p(t)$, $T_0(t)$ при заданных выше параметрах. На одном из графиков привести результаты вычислений двумя методов разных порядков точности. Показать, как влияет выбор метода на шаг сетки.
Параметры разрядного контура:
R=0.35 см
$l_э$=12 см
L_k=187 10^{-6} Гн
C_k=268 10^{-6} Ф
R_k=0.25 Ом
U_{co}=1400 В
I_o=0..3 А
T_w=2000 К
На рисунках 2 и 3 представлены рузультаты работы программы для методов Рунге-Кутта 2ого и 4ого порядков точности для заданных параметров.

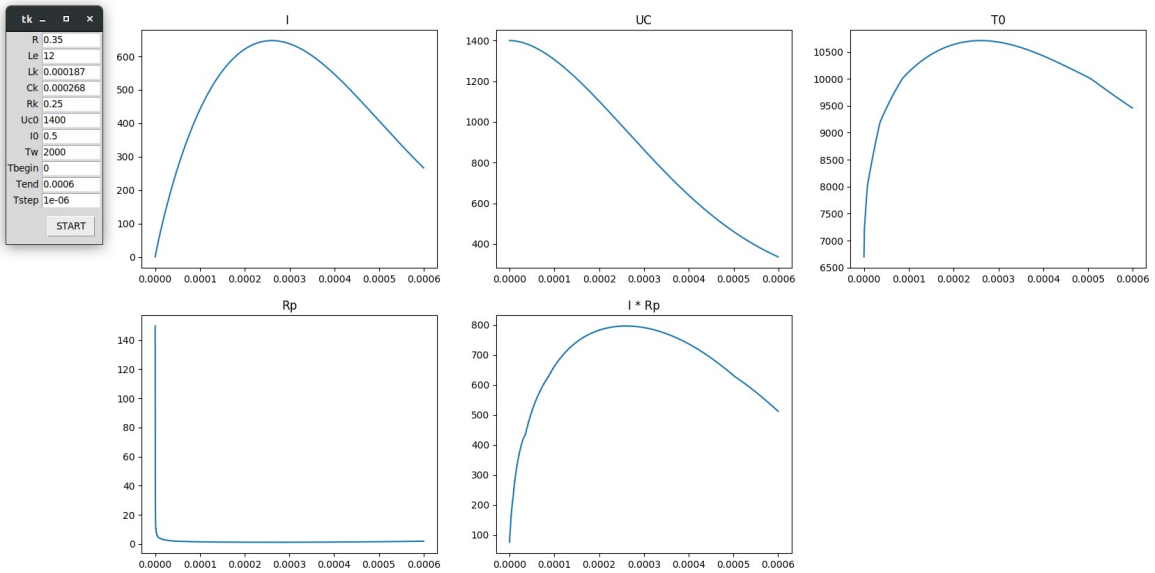

Рис 2. Результат работы программы для метода Рунге-Кутта 4ого порядка.

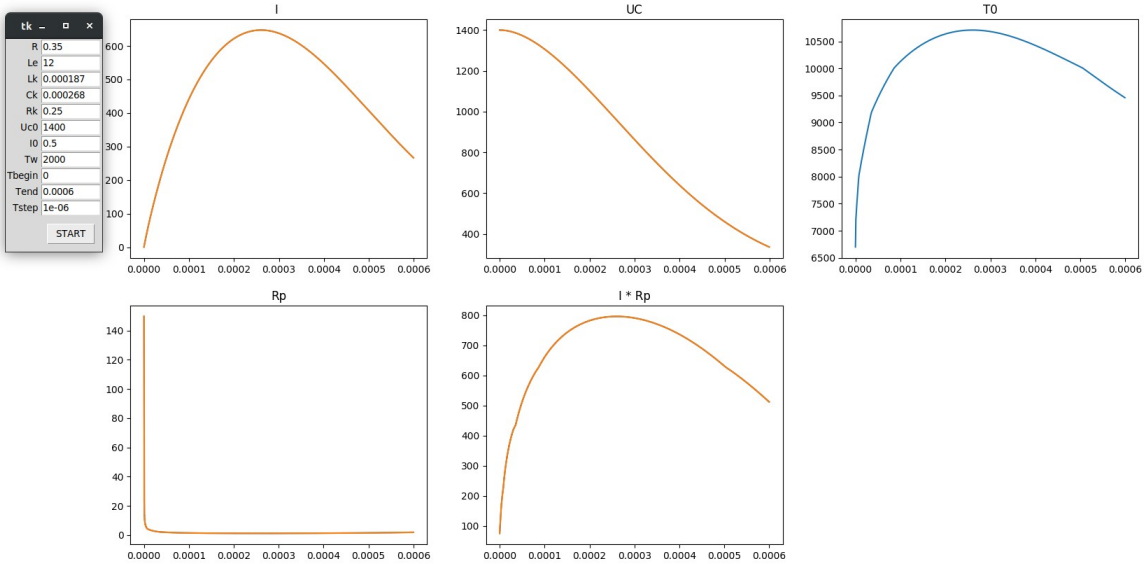

Рис 3. Результат работы программы для метода Рунге-Кутта 2ого порядка.

Сравним методы второго(желтый) и четвертого(синий) порядка при различных значениях шага.

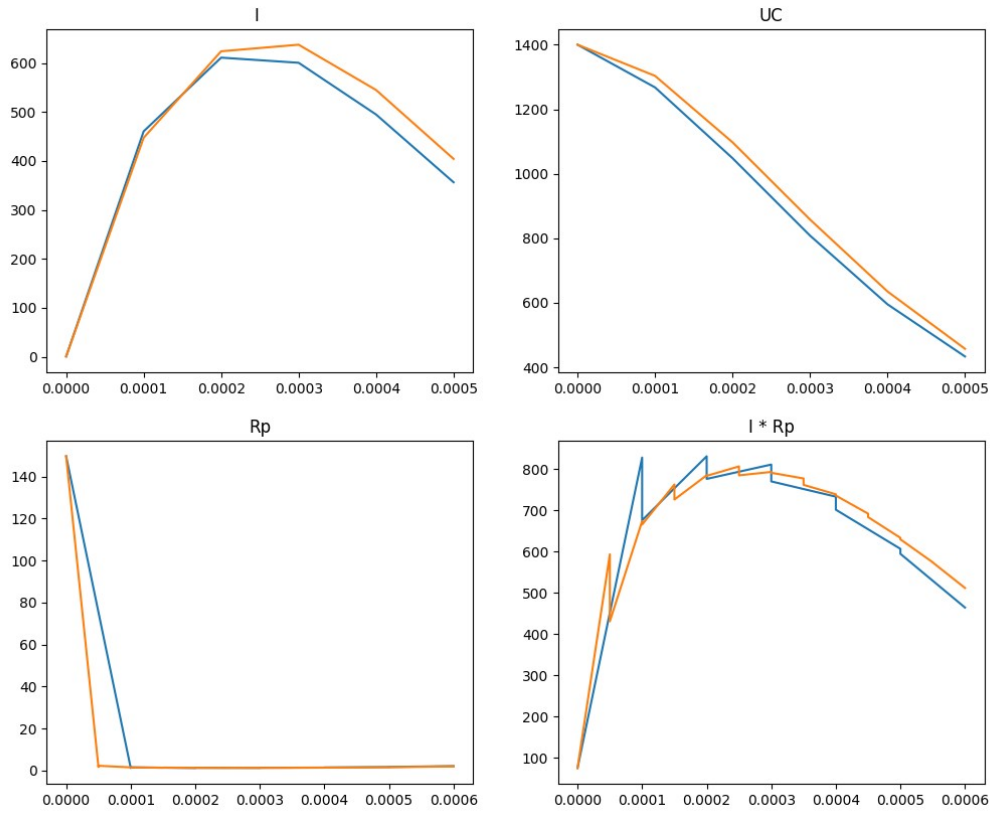

Рис 4. Результат работы программы с шагом 1e-4

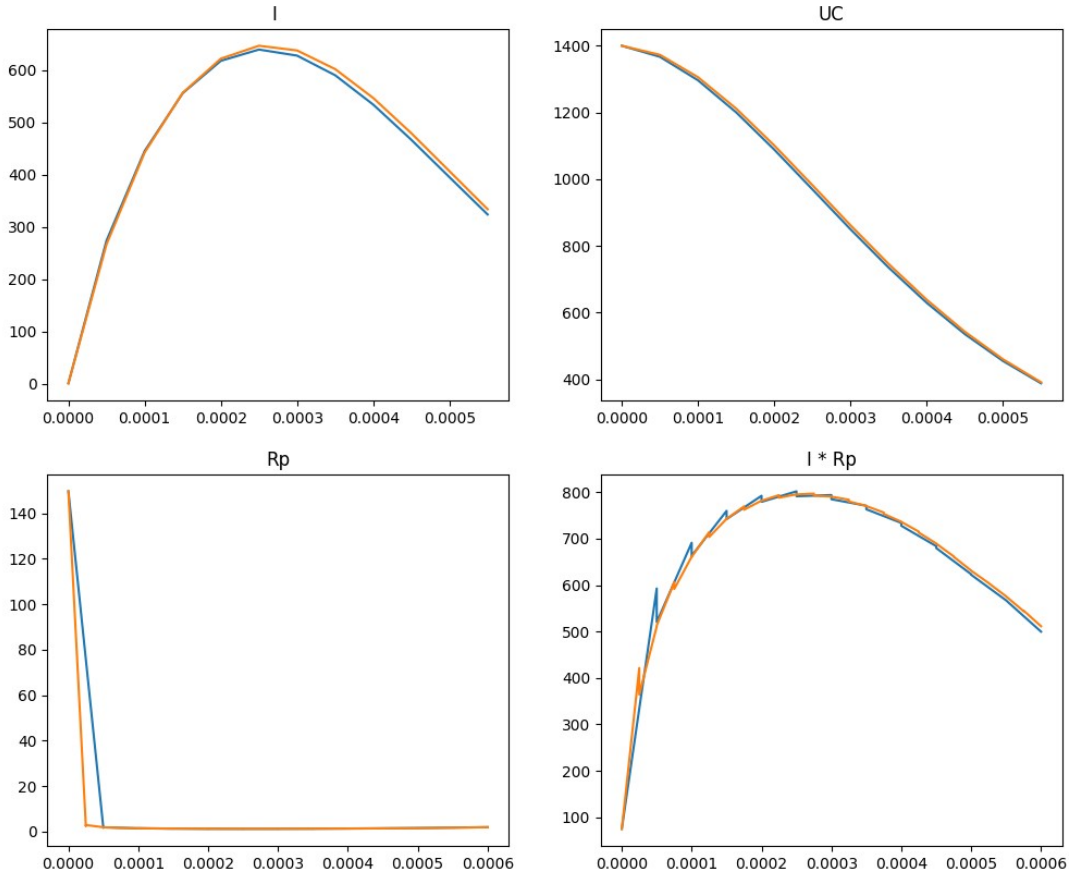

Рис 5. Результат работы с шагом 5e-5

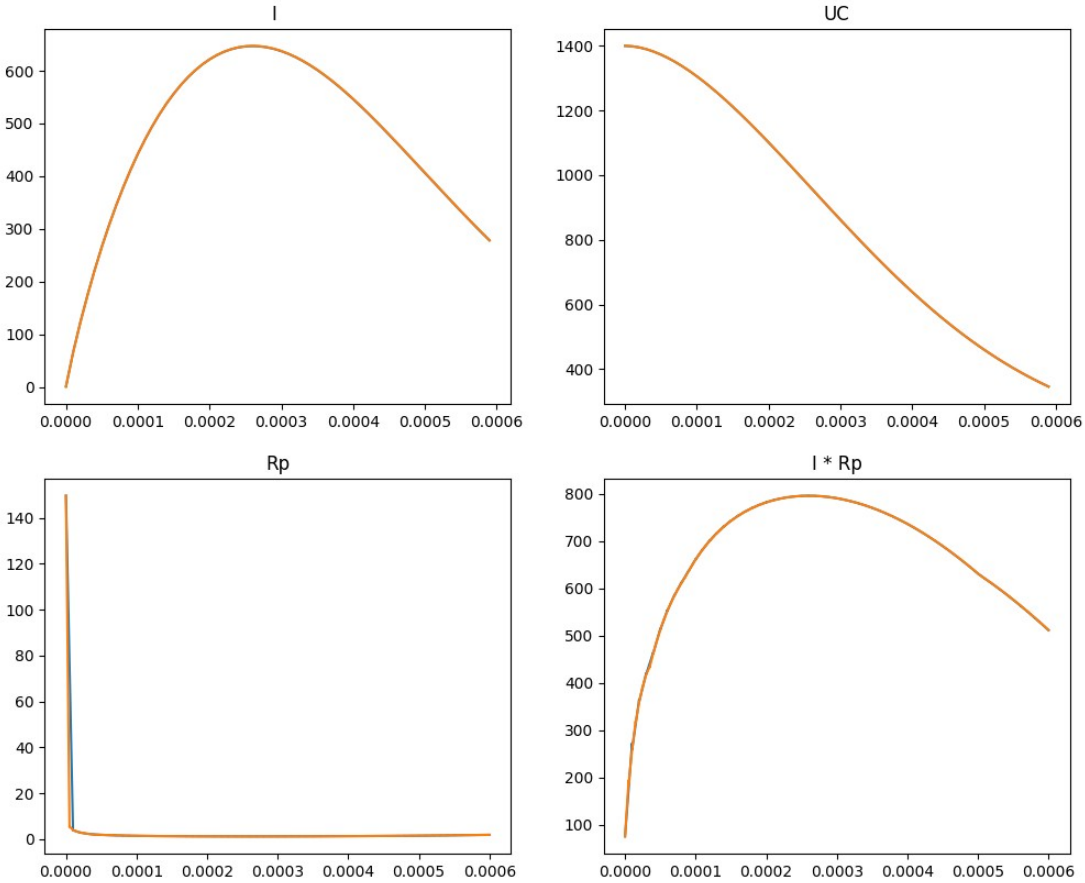

Рис 6. Результат работы программы с шагом 1e-5

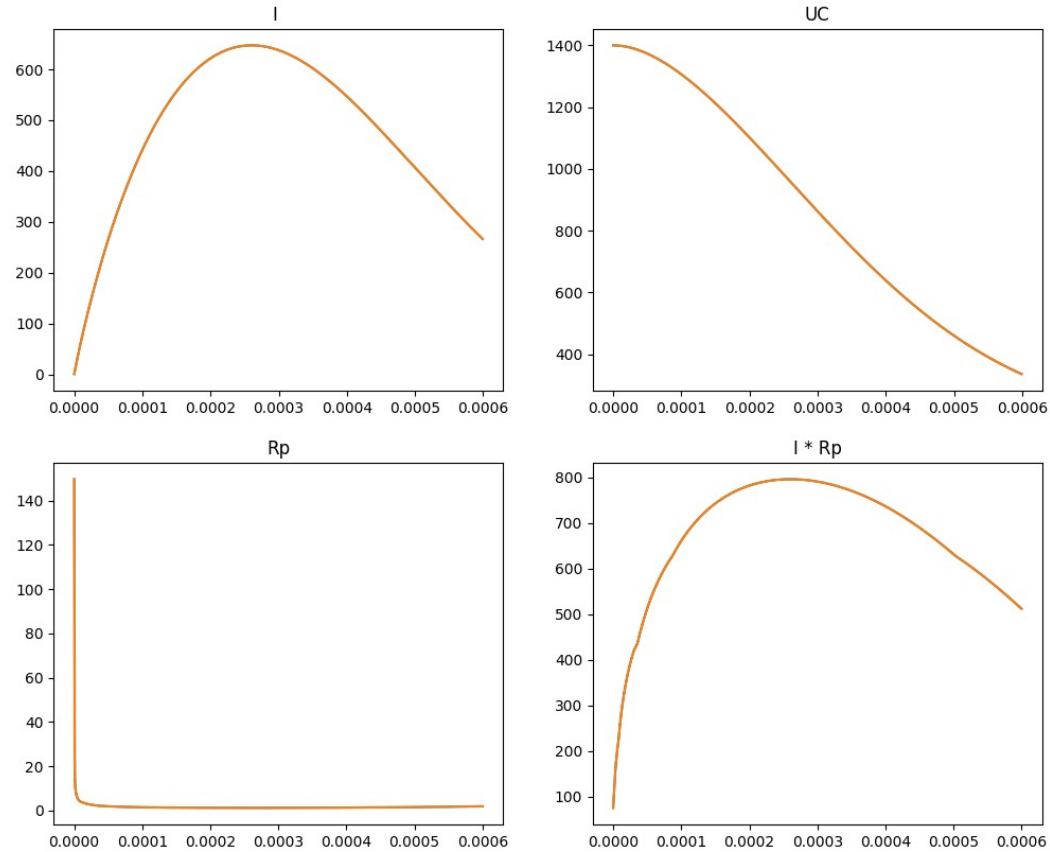

Рис 7. Результат работы программы с шагом 1e-6

На рисунках 4-7 представлены результаты работы программы с разным шагом. Можно заметить, что при маленьком значении шага результаты обоих методов совпадают. При увеличении шага, разниц между методами увеличивается. Однако метод Рунге-Кутта 4 порядка более точен и его график даже при больших значениях шага более приближен к итоговому результату, это можно увидеть на рисунке 8. Для большего значения шага метод 4ого порядка точности дает практически идентичный результат, что и метод 2ого порядка.

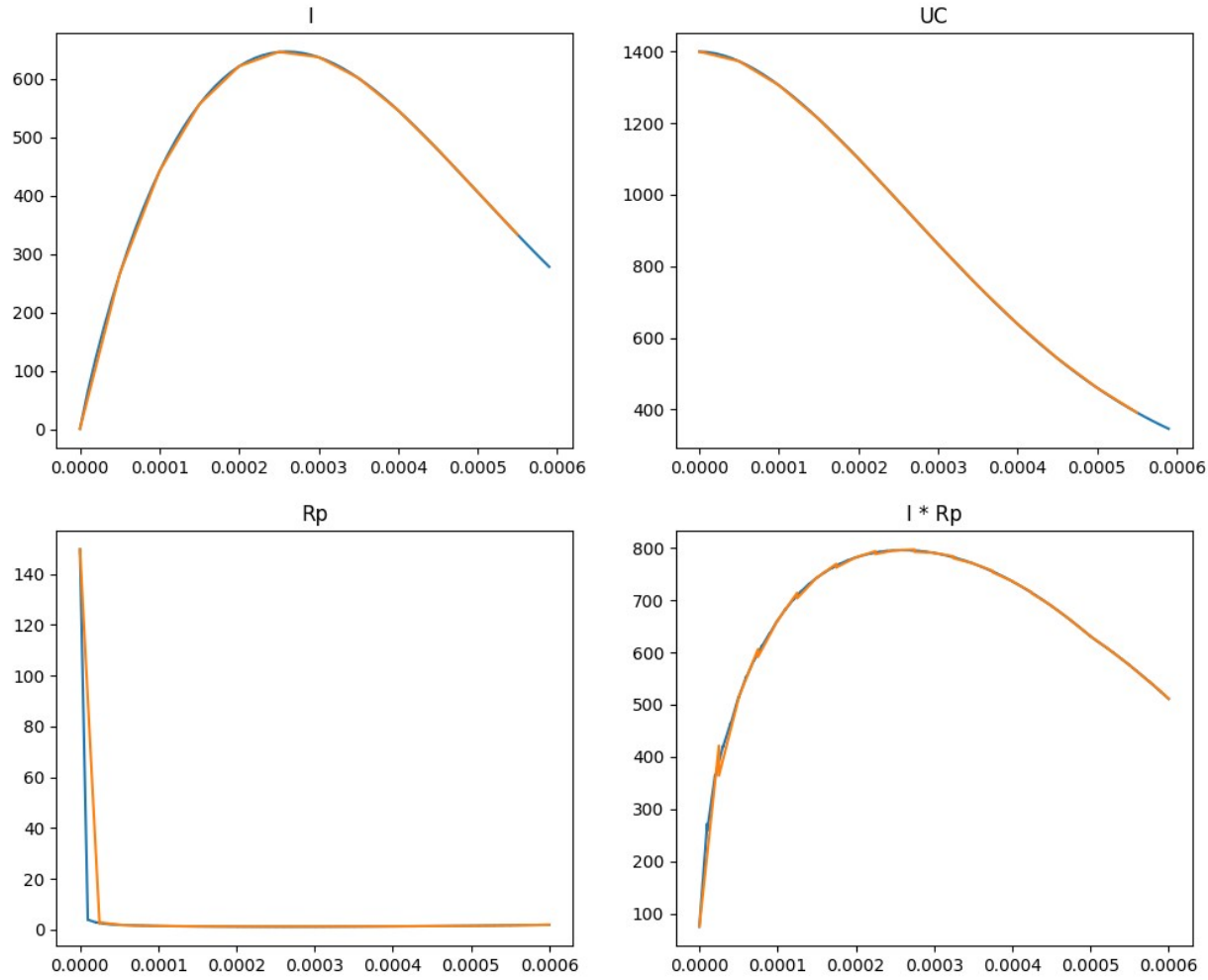

Рис 8. Результат работы программы для метода Рунге-Кутта 2ого порядка с шагом 1e-5 и метода 4ого порядка с шагом 5e-5

2. График зависимости $I(t)$ при $R_k + R_p = 0$. Обратить внимание на то, что в этом случае колебания тока будут не затухающими.

Для достижения условия установим следующие параметры:
R=0.35 см
$l_э$=0
L_k=187 10^{-6} Гн
C_k=268 10^{-6} Ф
R_k=0
U_{co}=1400 В
I_o=0..3 А
T_w=2000 К

На рисунке 9 представлен результат работы программы для этих параметров.

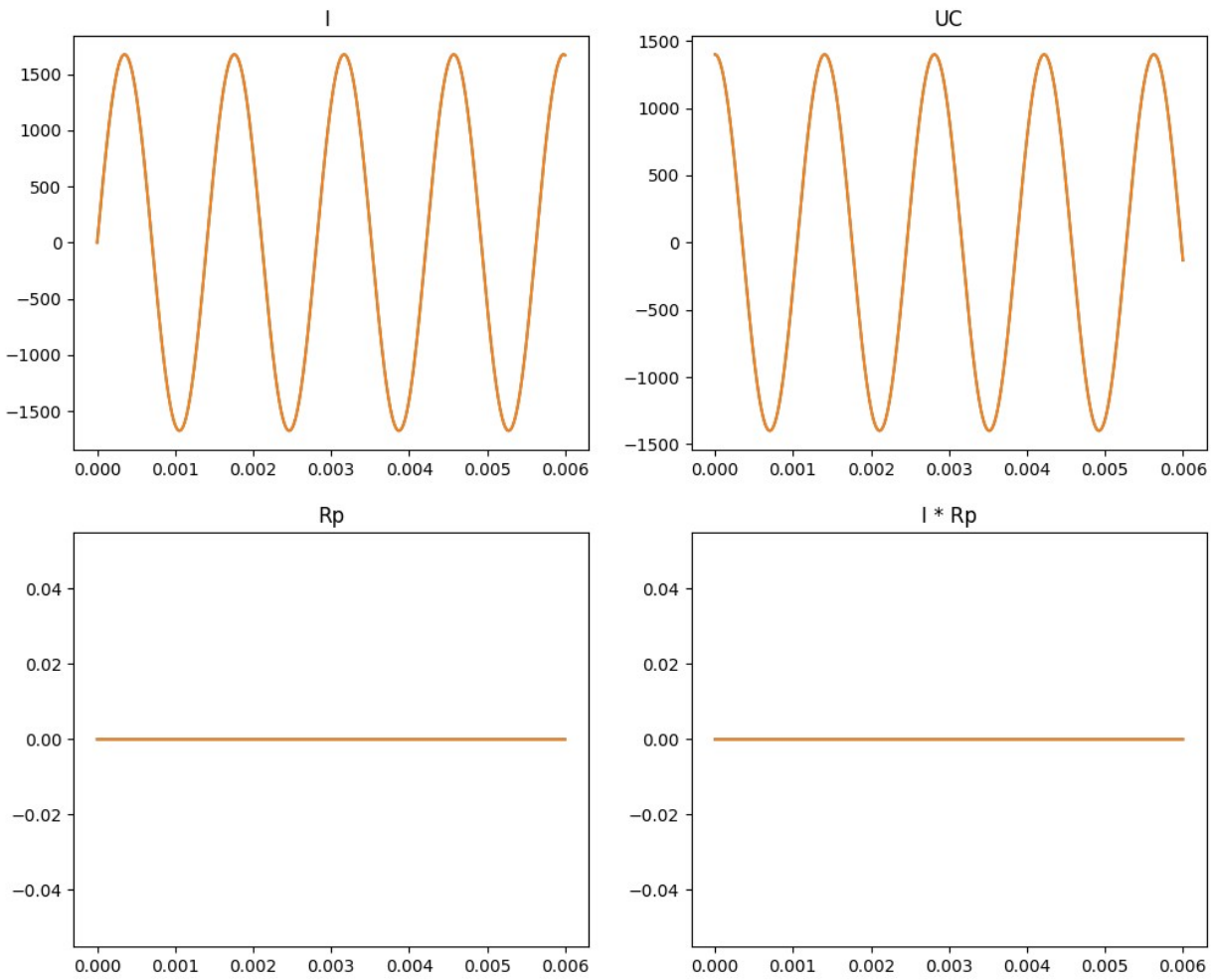

Рис 9. Результат работы программы для Rp+Rk = 0

При отсутсвии сопротивлений контур превращается в колебательный, поэтому колебания незатухающие.

3. График зависимости $I(t)$ при $R_k = 200$ Ом в интервале значений t 0-20 мкс.

На рисунке 10 представлен результат работы программы при Rk=200Ом.

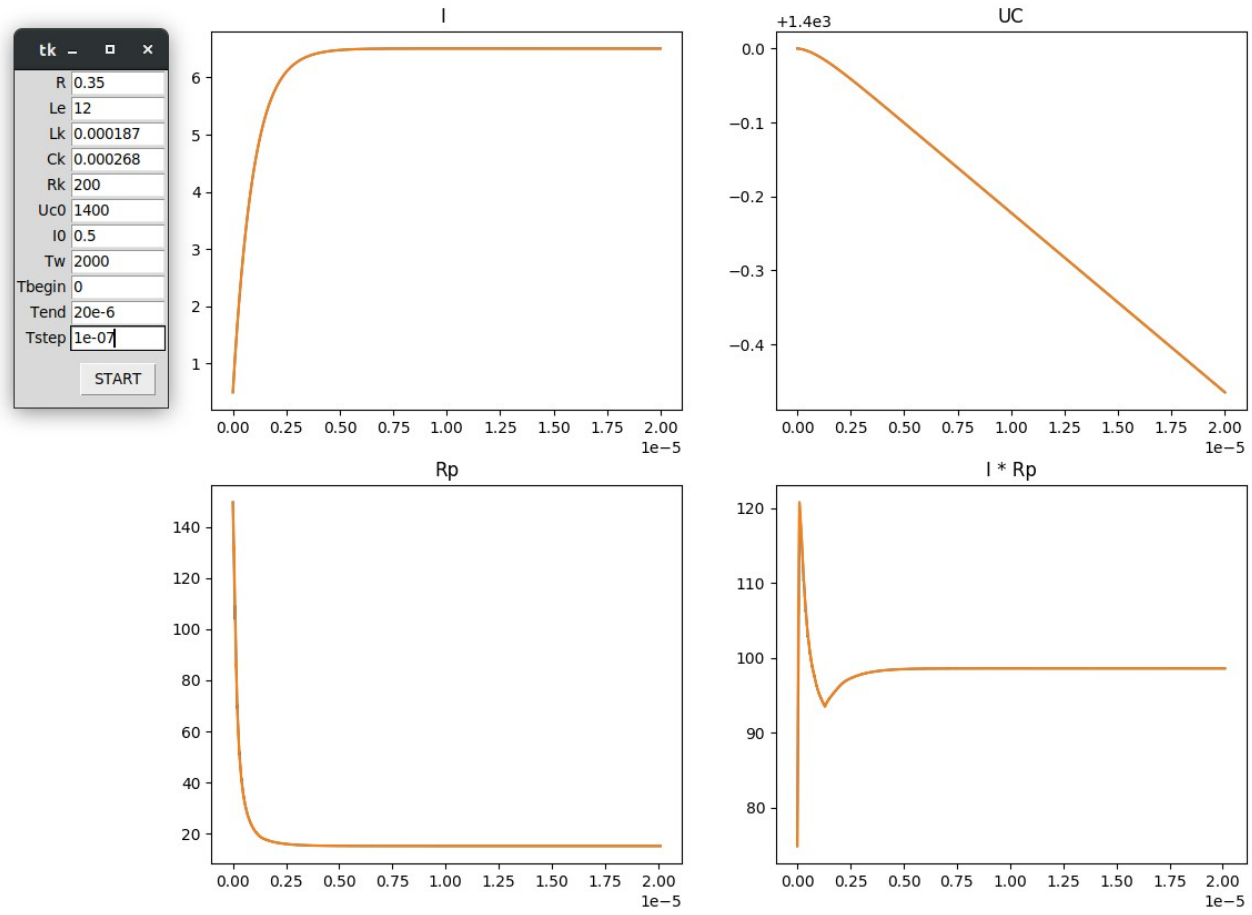

Рис 10. Результат работы программы при Rk=200Ом

V. Вопросы при защите лабораторной работы.

1. Какие способы тестирования программы можно предложить?

Программа должна выводить результаты, соответствующие законам физики. Это можно определить по виду графиков при определенных значениях входных параметров. Также, можно проверить, что программа правильно ведет себя при вводе большого значения сопротивления или в случае, когда сумма сопротивлений контура равна нулю (контур обращается в колебательный, колебания не затухают).

Также программу можно тестировать при разных значениях шага. При определенном введем значении шага, дальнейшее уменьшение шага не меняет результат. Это означает, что найден точный результат.

Помимо этого можно сравнить результаты работы двух методов разной точности. При маленьком значении шага они должны совпадать. Это объясняется тем, что при маленьком шаге результат перестает меняться, а результаты двух методов решения одной и той же системы уравнений должны совпадать.

2. Получите систему разностных уравнений для решения сформулированной задачи неявным методом трапеций. Опишите алгоритм реализации полученных уравнений.

Неявный метод трапеций – это метод Рунге-Кутта 2-го порядка точности с α=0.5.

Для системы уравнений вида

$$\begin{Bmatrix} u'(x)=f(x,u(x)) \\ u(x_0)=y_0 \end{Bmatrix}$$

метод Рунге-Кутта второго порядка описывается следующим уравнением:

$$y_{n+1}=y_n+h_n*((1-\alpha)*f(x_n,y_n)+\alpha*f(x_n+\frac{h_n}{2*\alpha},y_n+\frac{h_n}{2*\alpha}*f(x_n,y_n)))$$

$$y_{n+1}=y_n+h_n*(\frac{f(x_n,y_n)+f(x_n+h_n,y_n+h_n*f(x_n,y_n))}{2})$$

Наша система:

$$\begin{Bmatrix} u'(x)=f(x,u,v) \\ v'(x)=\varphi(x,u,v) \\ u(\xi)=\eta_1 \\ v(\xi)=\eta_2 \end{Bmatrix} => \begin{Bmatrix} \frac{dI}{dT}=\frac{U-(R_k+R_p(I))I}{L_k}=f(t,I,Uc) \\ \frac{dU}{dt}=-\frac{I}{C_k}=\phi(t,I) \end{Bmatrix}$$

Подставим наши уравнения

$$I_{n+1} = I_n + h_n * \left(\frac{f(t_n, I_n, Uc_n) + f(t_n + h_n, I_n + h_n * f(t_n, I_n, Uc_n), Uc_n + h_n * \phi(t_n, I_n))}{2} \right)$$

$$Uc_{n+1} = Uc + h_n * \left(\frac{\phi(t_n, I_n) + \phi(t_n + h_n,)}{2} \right)$$

Таким образом, получаем уравнения для метода трапеций. С помощью этих уравнений можно итерационно найти решения. Нам известны I0, t0,U0, а также шаг. Отталкиваясь от этих значений, подставляя их в полученные формулы, можно получить решение системы уравнений. От шага будет зависеть точность полученных результатов.

3. Из каких соображений проводится выбор того или иного метода, учитывая, что чем выше порядок точности метода, тем он более сложен?

Чтобы добиться точного результата на методах низкого уровня точности, нужно использовать очень маленький шаг, а следовательно производить очень много итераций. Методы более высокого порядка сложнее, однако требуют меньше итераций для поиска точного значения. Выбор метода должен основываться на возможностях системы, а также точности, с которой необходимо получить удовлетворительный результат. Для получения результата более высокой точности следует использовать методы более высокого порядка точности. Для менее точного результата целесообразнее использовать методы меньшего порядка точности, так как удовлетворительный результат можно получить быстро и просто и не требуется производить сложные вычисления.