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 Лучина Е.Д

Группа ИУ7-61Б

Преподаватель Градов В.М.

Москва.
2020 г.

Содержание

Цель работы

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.

Постановка задачи

Задан разрядный контур (рисунок 1), включающий катушку индуктивностью L_k, конденсатор емкостью C_k, постоянное активное сопротивление R_k и разрядную трубку с нелинейным сопротивлением $R_p(I)$. Трубка имеет длину l и радиус R. Данные контур описывается следующей системой электротехнических уравнений (1).

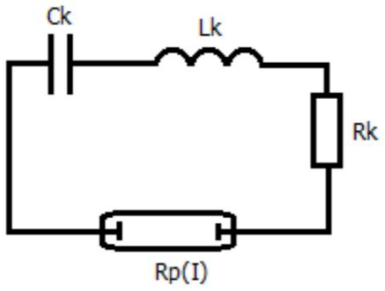


Рисунок 1 - Схема контура

$$\begin{aligned} L_k \frac{dI}{dt} + (R_k + R_p) \times I - U = 0 \\ C_k \frac{dU}{dt} = -I \end{aligned} \qquad (1)$$

Здесь I, U - ток и напряжение на конденсаторе.

Сопротивление трубки $R_p(I)$ находится в зависимости от тока и рассчитывается по формуле (2).

$$R_P = \frac{l}{2\pi R^2 \int_0^1 \sigma(T(z)) z dz} \qquad (2), \text{ где}$$

$$T(z) = (T_w - T_0)\, z^m + T_0 \qquad (3)$$

Необходимо с применением формул Рунге-Кутта второго и четвертого порядков решить систему. Решение продемонстрировать в виде графиков.

Исходные данные

Начальные значения: $t = 0;\ I = I_0;\ U = U_0$.

R = 0.35 см l = 12 см	$L_k = 60 \times 10^{-6}$ Гн $C_k = 150 \times 10^{-6}$ Ф	U_0 = 1500 В I_0 = 0.5 ... 3 А	R_K = 0.5 ... 200 Ом T_w = 2000 К

* все размерности согласованы

Даны связующие таблицы (таблицы 1, 2), с помощью интерполяции которых можно получить значение R_p. Параметры T_0, m находятся интерполяцией из табл.1 при известном токе I. Коэффициент электропроводности σ находится интерполяцией из табл.2 при определенной температуре T.

Таблица 1

I, А	T_0, К	m
0.5	6700	0.5
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, К	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

Методы Рунге-Кутта

- Пусть даны уравнения

$$y'(x) = f(x,\ y,\ z) \qquad (3)$$

$$z'(x) = \varphi(x,\ y,\ z) \qquad (4)$$

а также начальные условия

$$y_0 = y(x_0)\ ,\ z_0 = z(x_0)\ . \qquad (5)$$

- Рекуррентные формулы Рунге-Кутта второго порядка точности для решения этой системы уравнений:

$$y_{n+1} = y_n + h_n[(1-\alpha) \times k_1 + \alpha \times k_2] \qquad (6)$$

$$z_{n+1} = z_n + h_n[(1-\alpha) \times q_1 + \alpha \times q_2] \qquad (7)$$

где

$$k_1 = f(x_n,\ y_n,\ z_n);\ q_1 = \varphi(x_n,\ y_n,\ z_n)$$
$$k_2 = f(x_n + \frac{h_n}{2\alpha},\ y_n + \frac{h_n}{2\alpha}k_1,\ z_n + \frac{h_n}{2\alpha}q_1)$$
$$q_2 = \varphi(x_n + \frac{h_n}{2\alpha},\ y_n + \frac{h_n}{2\alpha}k_1,\ z_n + \frac{h_n}{2\alpha}q_1)$$
$$\alpha = 1 \text{ или } \alpha = \frac{1}{2}$$

- Рекуррентные формулы Рунге-Кутта четвертого порядка точности для решения этой системы:

$$y_{n+1} = y_n + \frac{k_1+2k_2+2k_3+k_4}{6} \quad (8)$$
$$z_{n+1} = z_n + \frac{q_1+2q_2+2q_3+q_4}{6} \quad (9)$$

где

$$k_1 = h_n \times f(x_n,\ y_n,\ z_n)$$
$$q_1 = h_n \times \varphi(x_n,\ y_n,\ z_n)$$
$$k_2 = h_n \times f(x_n + \frac{h_n}{2},\ y_n + \frac{k_1}{2},\ z_n + \frac{q_1}{2})$$
$$q_2 = h_n \times \varphi(x_n + \frac{h_n}{2},\ y_n + \frac{k_1}{2},\ z_n + \frac{q_1}{2})$$
$$k_3 = h_n \times f(x_n + \frac{h_n}{2},\ y_n + \frac{k_2}{2},\ z_n + \frac{q_2}{2})$$
$$q_3 = h_n \times \varphi(x_n + \frac{h_n}{2},\ y_n + \frac{k_2}{2},\ z_n + \frac{q_2}{2})$$
$$k_4 = h_n \times f(x_n + h_n,\ y_n + k_3,\ z_n + q_3)$$
$$q_4 = h_n \times \varphi(x_n + h_n,\ y_n + k_3,\ z_n + q_3)$$

Решение заданной системы (1)

Приведем уравнения системы (1) к виду (3), (4), выразим производные и при соотношении определим функции f и φ.

$$\frac{dI}{dt} = \frac{1}{L_k} \times [U - (R_k + R_p) \times I] = f(t,\ I,\ U) \quad (10)$$
$$\frac{dU}{dt} = -\frac{I}{C_k} = \varphi(t,\ I,\ U) \quad (11)$$

Теперь, представив уравнения в нужном виде, можем применять к ним методы по формулам (6), (7) для второго порядка точности и (8), (9) для четвертого.

Отметим методы нахождения $R_p(I)$.

- Интерполировать таблицы будем полиномом Ньютона первой степени по формуле

$$y(x) = y_0 + (x - x_0)\frac{y_1 - y_0}{x_1 - x_0}, \quad (12)$$

где $(x_0,\ y_0)$ и $(x_1,\ y_1)$ - ближайшие к x точки в таблице.

- Интеграл вычислим методом Симпсона

$$\int_a^b f(x)dx \approx \frac{h}{3}(f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_4) + ... + 4f(x_{N-1}) + f(x_N)) \quad (13)$$

Где N - количество отрезков (обязательно четное), $h = \frac{b-a}{N}$ - шаг. Для решения задачи N было выбрано равным 40.

Таким образом алгоритм решения заданной системы следующий:

Для каждого значения t_{i+1} и найденных I_i, U_i

Вычислить $R_p(I)$

Найти T_0 и m из таблицы 1 для I_i по (12)

Вычислить интеграл по (13)

Для каждого z от 0 до 1 с шагом 1/40

Найти T(z) по формуле (3)

Найти σ из таблицы 2 для найденного T по (12)

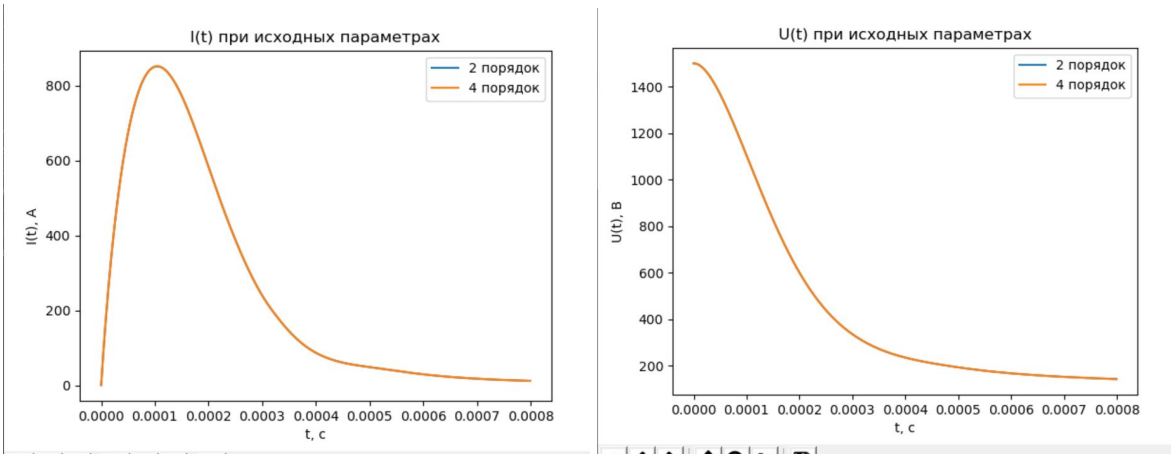
Вычислить $R_p(I)$ по формуле (2)

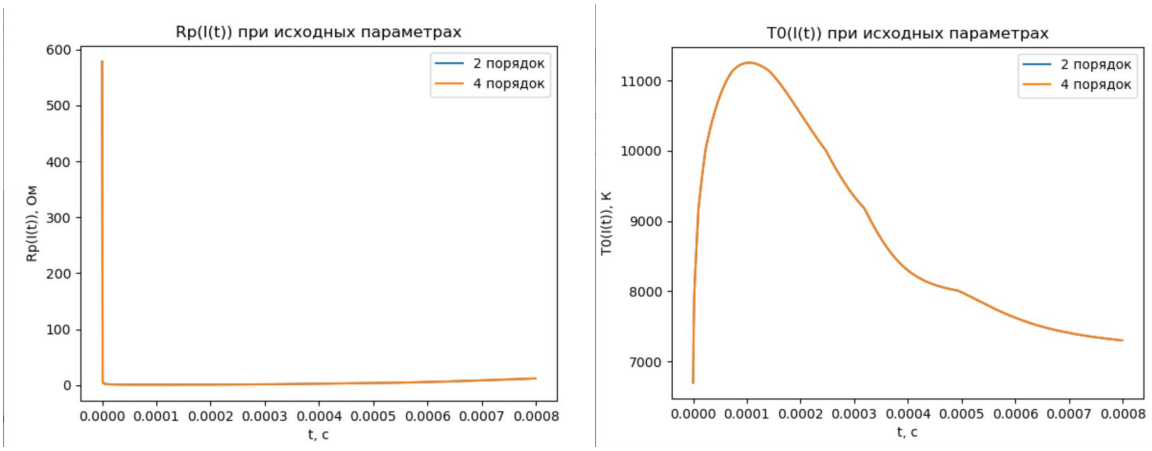
Найти I_{i+1}, U_{i+1} применив метод Рунгу-Кутта (6), (7) или (8), (9)

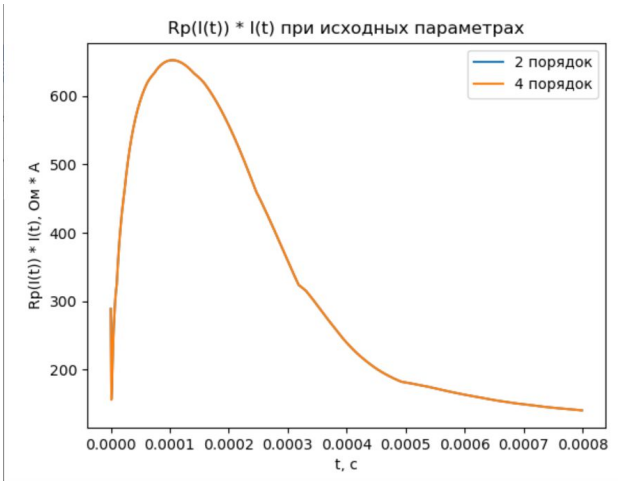
Результаты

- графики зависимости от времени импульса t:
  $I(t)$, $U(t)$, $R_p(t)$, $R_p(t) \times I(t)$, $T_0(t)$ при заданных выше параметрах. На одном из графиков привести результаты вычислений двумя методами разных порядков точности. Показать, как влияет выбор метода на шаг сетки.

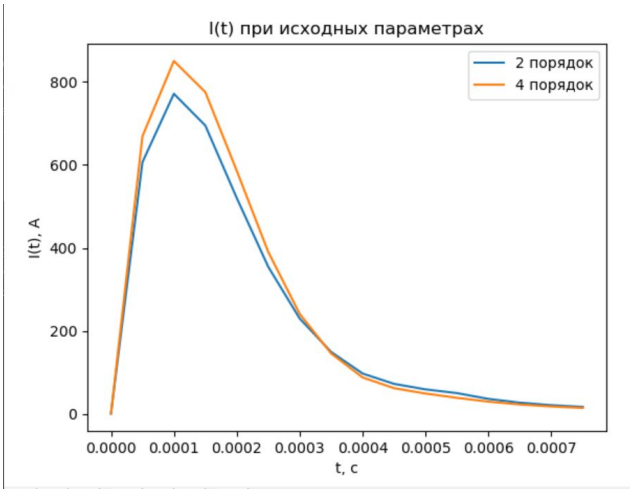
h = 1e-6

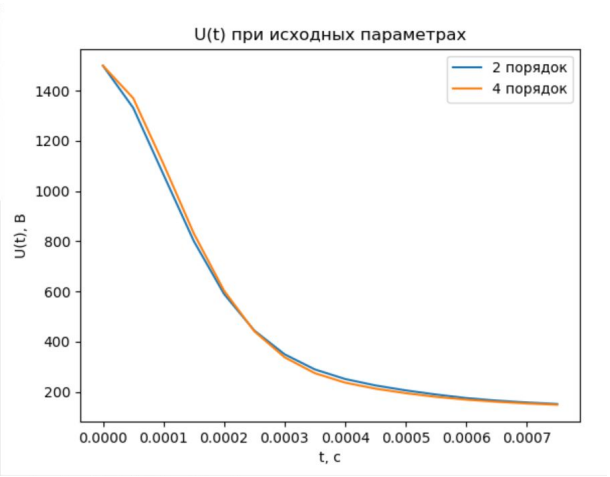


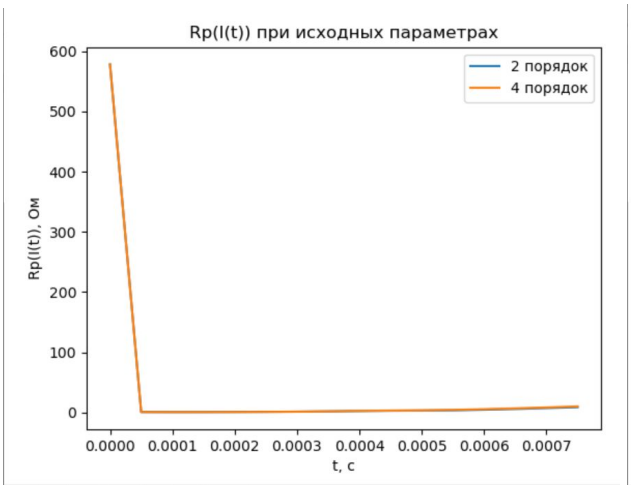


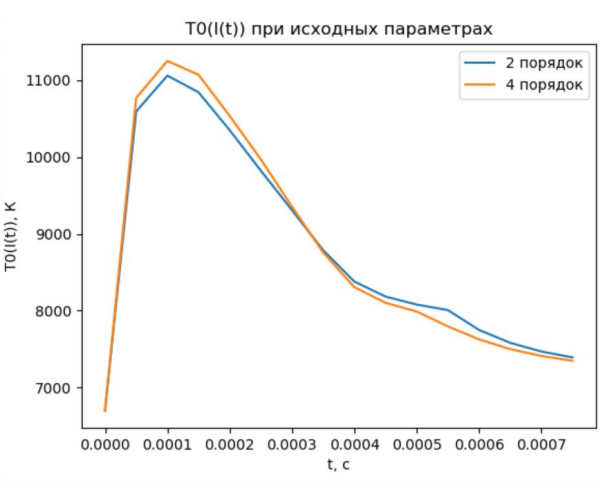

Rp(I(t)) * I(t) при исходных параметрах
2 порядок
4 порядок
Rp(I(t)) * I(t), Ом * А
600
500
400
300
200
0.0000 0.0001 0.0002 0.0003 0.0004 0.0005 0.0006 0.0007 0.0008
t, c

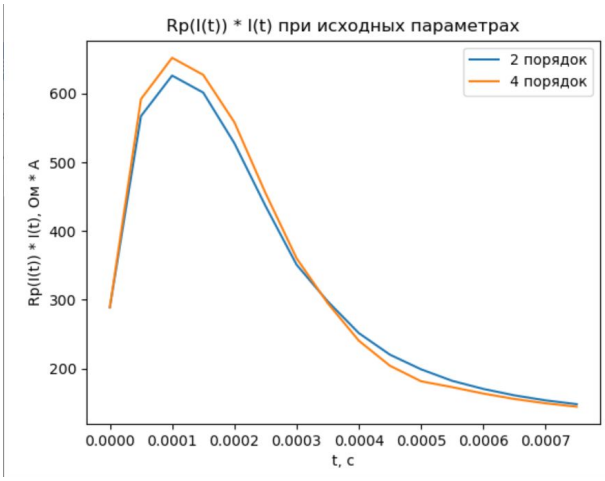
h = 5e-5


I(t) при исходных параметрах
2 порядок
4 порядок
I(t), A
800
600
400
200
0
0.0000 0.0001 0.0002 0.0003 0.0004 0.0005 0.0006 0.0007
t, c


U(t) при исходных параметрах
2 порядок
4 порядок
U(t), B
1400
1200
1000
800
600
400
200
0.0000 0.0001 0.0002 0.0003 0.0004 0.0005 0.0006 0.0007
t, c

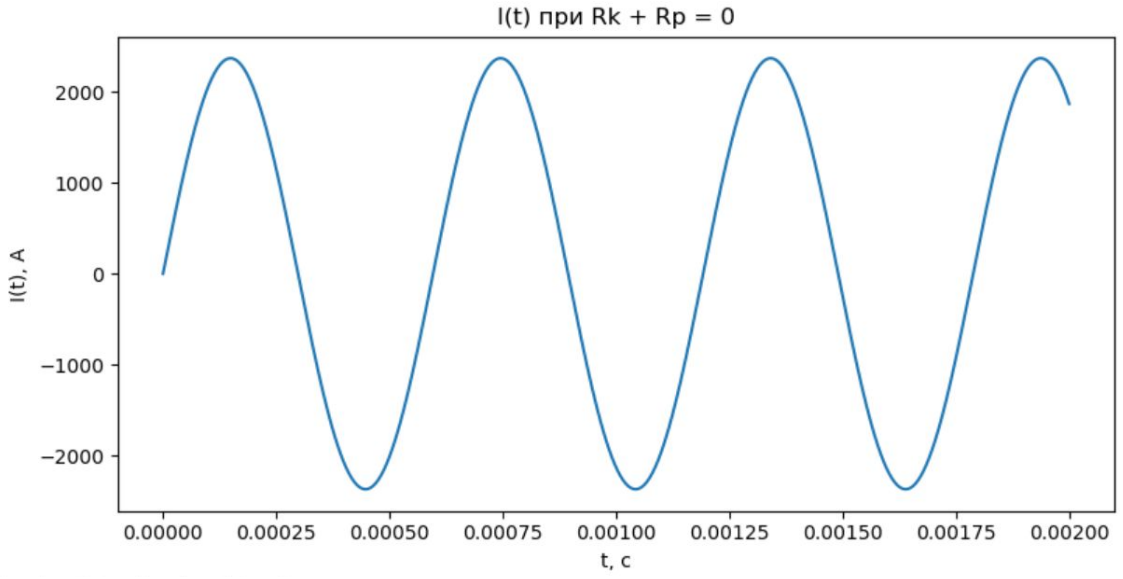

Rp(I(t)) при исходных параметрах
2 порядок
4 порядок
Rp(I(t)), Ом
600
500
400
300
200
100
0
0.0000 0.0001 0.0002 0.0003 0.0004 0.0005 0.0006 0.0007
t, c


T0(I(t)) при исходных параметрах
2 порядок
4 порядок
T0(I(t)), K
11000
10000
9000
8000
7000
0.0000 0.0001 0.0002 0.0003 0.0004 0.0005 0.0006 0.0007
t, c

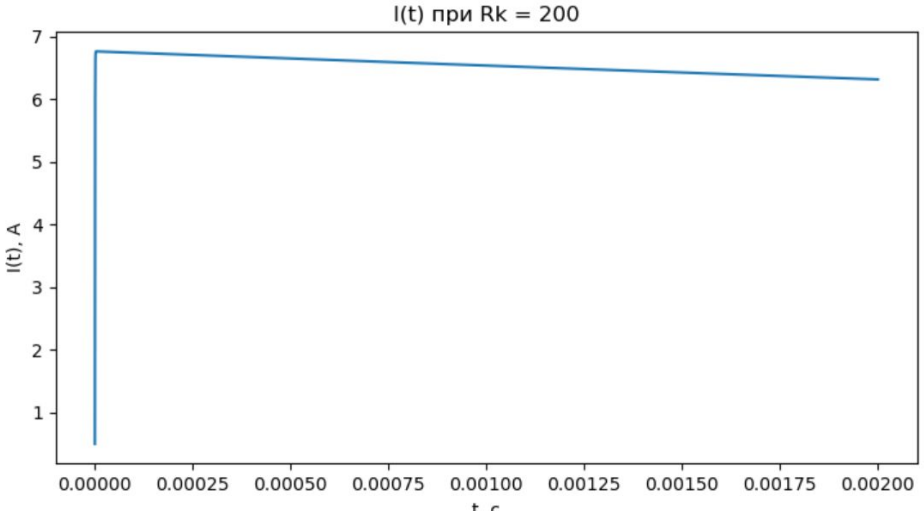


Видим, что при уменьшении шага точность методов возрастает. ОТметим, что при шаге 0.000001 результаты обоих методов идентичны, следовательно следует использовать метод второго порядка точности так как он требует меньше вычислительных ресурсов.

- график зависимости $I(t)$ при $R_k + R_p = 0$. Пронаблюдать незатухающие колебания.



- график зависимоти $I(t)$ при $R_k = 200$ Ом в интервале значений t 0-20 мкс.



Листинг программы

Программа реализована на языке Python с использованием модулей math и matplotlib.

Интерполяции полиномом Ньютона первой степени

```
def interpolate(x, xValues, yValues, zValues = None):
    lenV = len(xValues)
    if (x <= xValues[0]):
        if (zValues):
            return yValues[0], zValues[0]
        return yValues[0]
    if (x >= xValues[lenV - 1]):
        if (zValues):
            return yValues[lenV - 1], zValues[lenV - 1]
        return yValues[lenV - 1]
    for i in range (lenV - 1):
        if (xValues[i] <= x <= xValues[i + 1]):
            x0 = xValues[i]
            x1 = xValues[i + 1]
            y0 = yValues[i]
            y1 = yValues[i + 1]
            if (zValues):
                z0 = zValues[i]
                z1 = zValues[i + 1]
            break
    y = y0 + (x - x0) * (y1 - y0) / (x1 - x0)
    if (zValues):
        z = z0 + (x - x0) * (z1 - z0) / (x1 - x0)
        return y, z
    return y
```

Интеграл методом Симпсона по формуле (13)

```
def integral(a, b, N, func):
    h = (b - a) / N
    m_sum = 0
    cur = a + h
    for i in range (1, N):
```

```
        if (i % 2):
            m_sum += 4 * func(cur)
        else:
            m_sum += 2 * func(cur)
        cur += h
    I = h / 3(func(a) + m_sum + func(b))
    return I
```

Метод Рунге-Кутта второго порядка точности

```
def rungekutta2(x0, y0, z0, f, fi, h, alpha = 1):
    k1 = f(x0, y0, z0)
    q1 = fi(x0, y0, z0)
    k2 = f(x0 + h/2/alpha, y0 + h/2/alpha*k1, z0 + h/2/alpha*q1)
    q2 = fi(x0 + h/2/alpha, y0 + h/2/alpha*k1, z0 + h/2/alpha*q1)

    y1 = y0 + h * ((1 - alpha) * k1 + alpha * k2)
    z1 = z0 + h * ((1 - alpha) * q1 + alpha * q2)
    return y1, z1
```

Метод Ругне-Кутта четвертого порядка точности

```
def rungekutta4(x0, y0, z0, f, fi, h):
    k1 = h * f(x0, y0, z0)
    q1 = h * fi(x0, y0, z0)
    k2 = h * f(x0 + h/2, y0 + k1/2, z0 + q1/2)
    q2 = h * fi(x0 + h/2, y0 + k1/2, z0 + q1/2)
    k3 = h * f(x0 + h/2, y0 + k2/2, z0 + q2/2)
    q3 = h * fi(x0 + h/2, y0 + k2/2, z0 + q2/2)
    k4 = h * f(x0 + h, y0 + k3, z0 + q3)
    q4 = h * fi(x0 + h, y0 + k3, z0 + q3)
```

```
    y1 = y0 + (k1 + 2 * k2 + 2 * k3 + k4) / 6
    z1 = z0 + (q1 + 2 * q2 + 2 * q3 + q4) / 6
    return y1, z1
```

Функции $f(t, I, U)$ и $\varphi(t, I, U)$ из уравнений (10), (11)

```
def f(t, I, U):
    return (U - (Rk + Rp(I)) * I) / Lk
    # return U / Lk

def fi(t, I, U):
    return - I / Ck
```

Функция, считающая нелинейное сопротивление трубки Rp(I)

```
def Rp(I):
    found_T0, found_m = interpolate(I, Itable, T0table, mtable)
    def underintegral(z, T0 = found_T0, m = found_m):
        T = (Tw - T0) * math.pow(z, m) + T0
        sigma = interpolate(T, Ttable, sigmatable)
        return sigma * z
    found_integral = integral(0, 1, 40, underintegral)
    return l / (2 * math.pi * R * R * found_integral)
```

Цикл, осуществляющий итерационное вычисление значений

```
t = tmin
while (t < tmax):
    t_arr.append(t)
    I_arr.append(I0)
    U_arr.append(U0)
    Rp_arr.append(Rp(I0))
    Rp_I_arr.append(Rp(I0) * I0)
    T0_arr.append(interpolate(I0, Itable, T0table))
    I1, U1 = rungekutta2(t, I0, U0, f, fi, tstep)
    # I1, U1 = rungekutta2(t, I0, U0, f, fi, tstep, 0.5)
    # I1, U1 = rungekutta4(t, I0, U0, f, fi, tstep)
    I0 = I1
    U0 = U1
    t += tstep
```

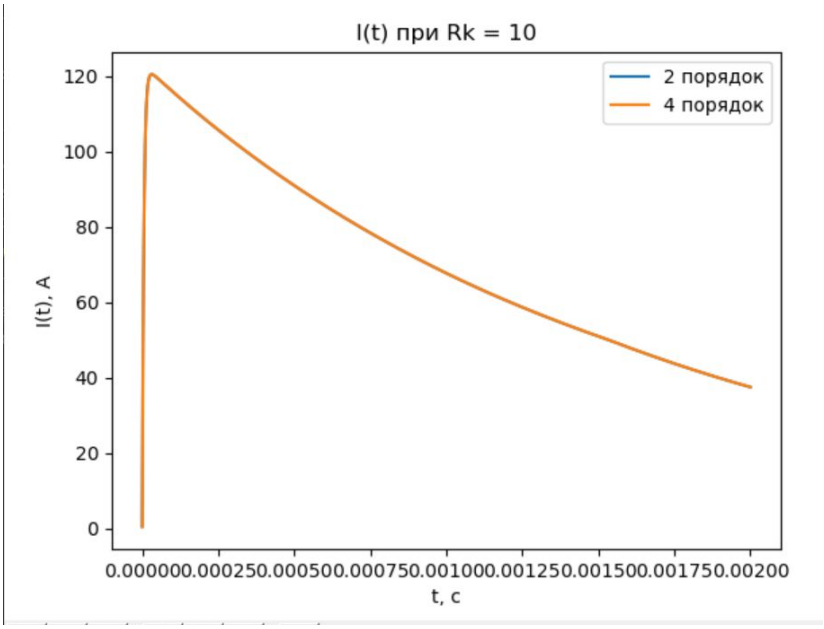
Функция вывода графика

```
def draw_plot(t_arr, I_arr2, I_arr4, labely, title):
    plt.plot(t_arr, I_arr2, label = "2 порядок")
    plt.plot(t_arr, I_arr4, label = "4 порядок")
    plt.xlabel('t, c')
    plt.ylabel(labely)
    plt.title(title)
    plt.legend()
    plt.show()
```

Вопросы

1. Какие способы тестирования программы можно предложить?

При нулевом сопротивлении контур становится колебательным. Увеличивая сопротивление Rk будем наблюдать менее пологое уменьшение силы тока.



Для тестирования методов Рунге-Кутта: уменьшая шаг наблюдаем бОльшую точность результата. Наступит момент, когда очередное уменьшение шага не изменит результат, тогда можно говорить, что полученный результат точный.

Соответственно для тестирования вычисления интеграла, можно запустить функцию для простого известного берущегося интеграла. А результат интерполяции можно сверить с таблицей.

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

Представим ОДУ как интегральное уравнение. Полученный интеграл вычислим с помощью метода трапеций.

$$u'(x) = f(x) \text{ - ДУ}$$

$$\int_{x_n}^{x_{n+1}} \frac{du}{dx} dx = \int_{x_n}^{x_{n+1}} f(x)dx$$

$$u_{n+1} = u_n + \int_{x_n}^{x_{n+1}} f(x)dx$$

$$y_{n+1} = y_n + \frac{h}{2}[f(x_n) + f(x_n + h)], \text{ где } h = (x_{n+1} - x_n) \qquad (14)$$

Метод Ругне-Кутта второго порядка точности при $\alpha = 0.5$ и представляет собой неявный метод трапеций. И уравнения (6), (7) будут выглядеть так:

$$y_{n+1} = y_n + 0.5h_n[k_1 + k_2] \quad (15)$$

$$z_{n+1} = z_n + 0.5h_n[q_1 + q_2] \quad (16)$$

где

$$k_1 = f(x_n,\ y_n,\ z_n);\ q_1 = \varphi(x_n,\ y_n,\ z_n)$$

$$k_2 = f(x_n + h_n,\ y_n + h_n k_1,\ z_n + h_n q_1)$$

$$q_2 = \varphi(x_n + h_n,\ y_n + h_n k_1,\ z_n + h_n q_1)$$

$$\frac{dI}{dt} = \frac{1}{L_k} \times [U - (R_k + R_p) \times I] = f(t,\ I,\ U) \quad (10)$$

$$\frac{dU}{dt} = -\frac{I}{C_k} = \varphi(t,\ I,\ U) \quad (11)$$

Решение ищется итерационно подстановкой (10), (11) в (15), (16).

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

При малом шаге производится большое количество менее сложных операций - метод меньшей точности. Результат остается приемлемым ввиду малого шага. (отметим про применение метода более высокой точности в таком случае не улучшит результат, а лишь использует больше ресурсов ввиду своей сложности вычисления)

При большем шаге, чтобы достичь приемлемой точности результата, нужно использовать метод большей точности. А его сложность будет компенсирована меньшим количеством итераций.

Также значение может иметь сама функция и шаг можно менять во время решения. Так, для более изменчивой функции, нужно выбрать меньший шаг. Если при очередной итерации результат не сильно изменился, шаг можно попробовать увеличить.