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

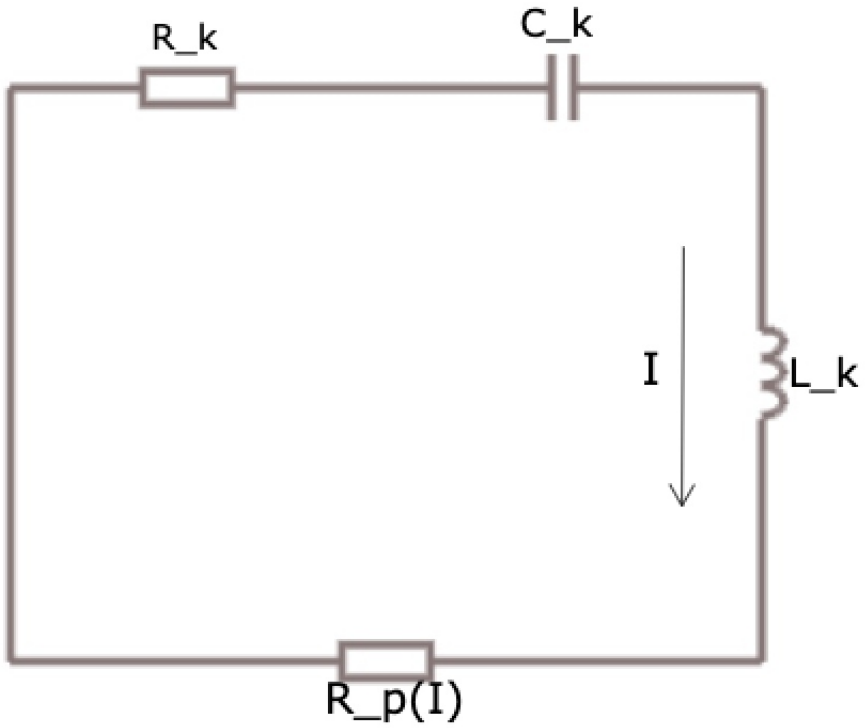
Тема: «Задача Коши для системы из двух уравнений ОДУ»

Студент: Левушкин И.К.
Группа: ИУ7-62Б
Оценка (баллы):
Преподаватель: Градов В.М.

Москва, 2020 г.

1 Условие задачи

Рассмотрим следующую электрическую схему:



Начальные условия:

- $t = 0$
- $I = I0$
- $U_c = U_c0$

Обход цепи по второму правилу Кирхгофа:

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

Где

$$R_p(I) = \frac{L_e}{2\pi R^2 \int_0^1 \sigma(T(z))zdz}$$

По определению конденсатора:

$$C_k \frac{dU_c}{dt} = -I$$

Итого, имеем систему из двух уравнений ОДУ:

$$\begin{cases} U'(t) = -\dfrac{I}{C_k} \\ I'(t) = \dfrac{U_c - (R_k + R_p(I))I}{L_k} \end{cases}$$

2 Метод решения задачи

В качестве решения будем использовать численный явный метод - Рунге-Кутта, имеющий 4-ый порядок точности.

2.1 Преимущества схем Рунге-Кутта

- схемы явные
- формулы достаточно точные
- позволяют вести расчеты с n-ым шагом

2.2 Рунге-Кутт 4-ого порядка точности

Ниже приведены формулы обобщенной схемы Рунге-Кутта для двух переменных

$$\begin{cases} U'(x) = f(x, U, V) -> y \\ V'(x) = \phi(x, U, V) -> z \\ V(\eta) = V_0 \\ U(\eta) = U_0 \end{cases}$$

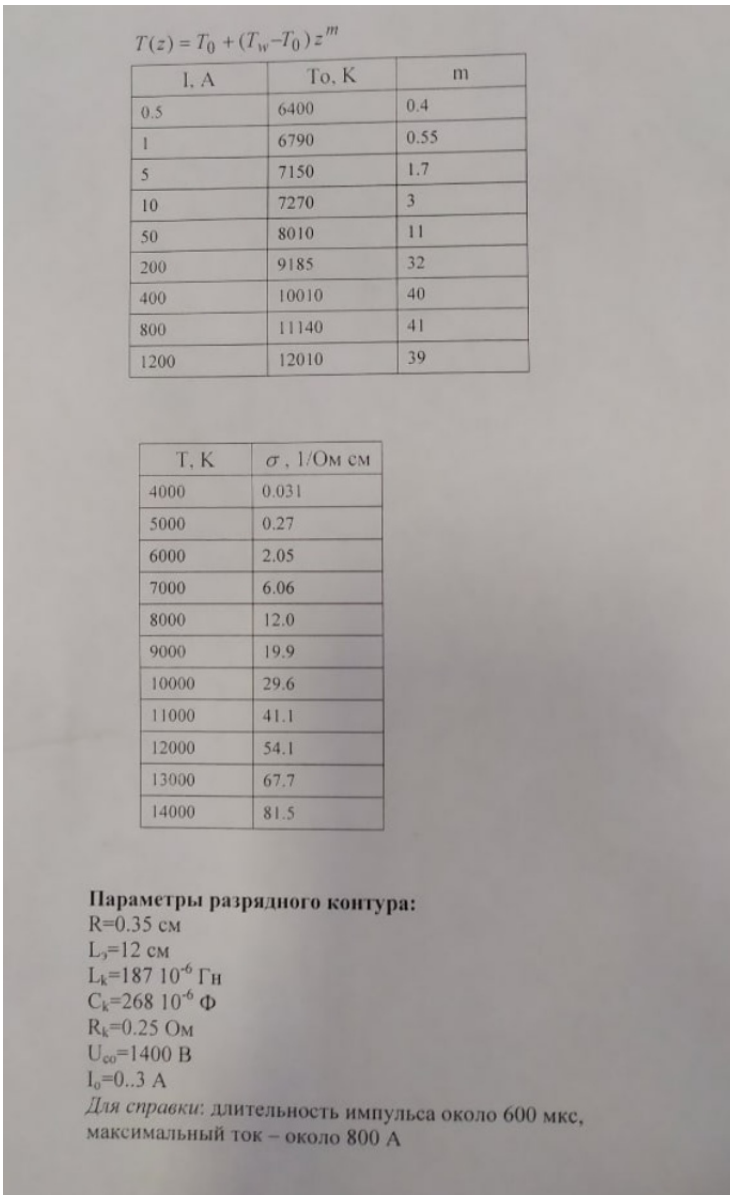
$$y_{n+1} = y_n + \frac{k_1 + 2k_2 + 2k_3 + k_4}{6}$$

$$z_{n+1} = z_n + \frac{q_1 + 2q_2 + 2q_3 + q_4}{6}$$

Где

- $k_1 = h_n f(x_n, y_n, z_n),\ q_1 = h_n \phi(x_n, y_n, z_n)$
- $k_2 = h_n f(x_n + \frac{h_n}{2}, y_n + \frac{k_1}{2}, z_n + \frac{q_1}{2}),\ q_2 = h_n \phi(x_n + \frac{h_n}{2}, y_n + \frac{k_1}{2}, z_n + \frac{q_1}{2})$
- $k_3 = h_n f(x_n + \frac{h_n}{2}, y_n + \frac{k_2}{2}, z_n + \frac{q_2}{2}),\ q_3 = h_n \phi(x_n + \frac{h_n}{2}, y_n + \frac{k_2}{2}, z_n + \frac{q_2}{2})$
- $k_1 = h_n f(x_n + h_n, y_n + k_3, z_n + q_3),\ q_1 = h_n \phi(x_n + h_n, y_n + k_3, z_n + q_3)$

3 Использованные константы



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

I, A	To, K	m
0.5	6400	0.4
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

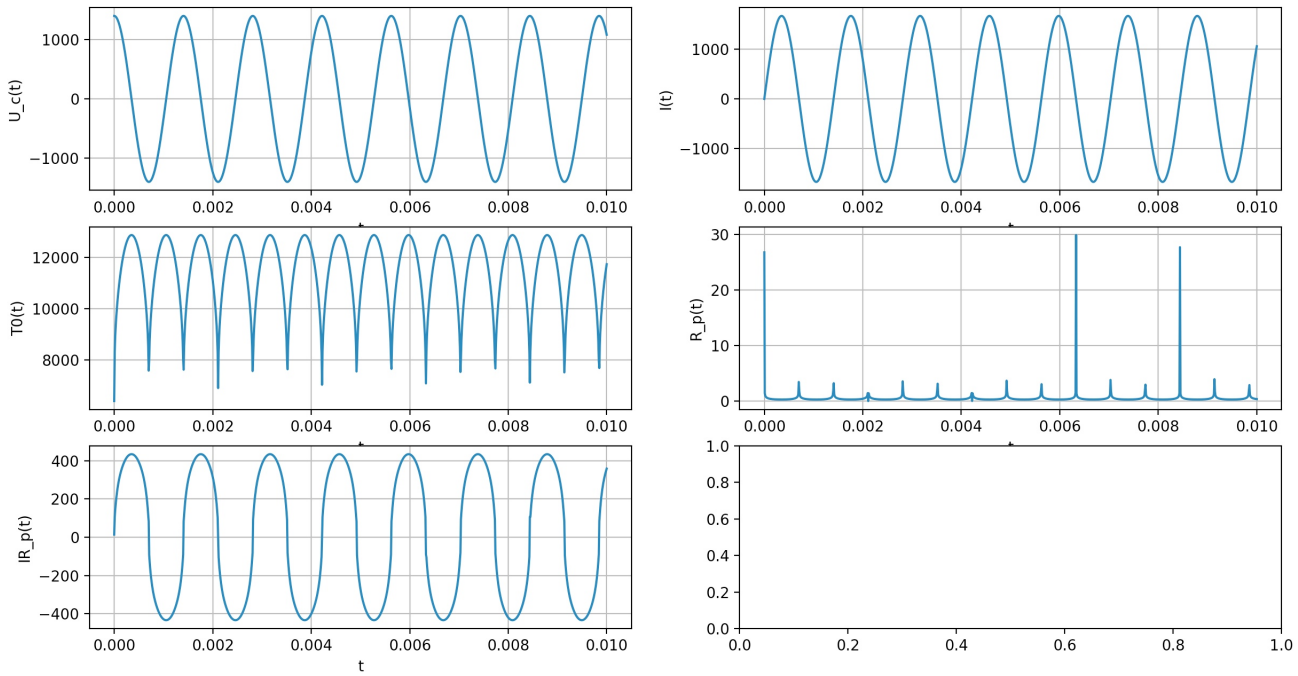
T, K	σ, 1/Ом см
4000	0.031
5000	0.27
6000	2.05
7000	6.06
8000	12.0
9000	19.9
10000	29.6
11000	41.1
12000	54.1
13000	67.7
14000	81.5

Параметры разрядного контура:
R=0.35 см
L_y=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
Для справки: длительность импульса около 600 мкс, максимальный ток – около 800 А

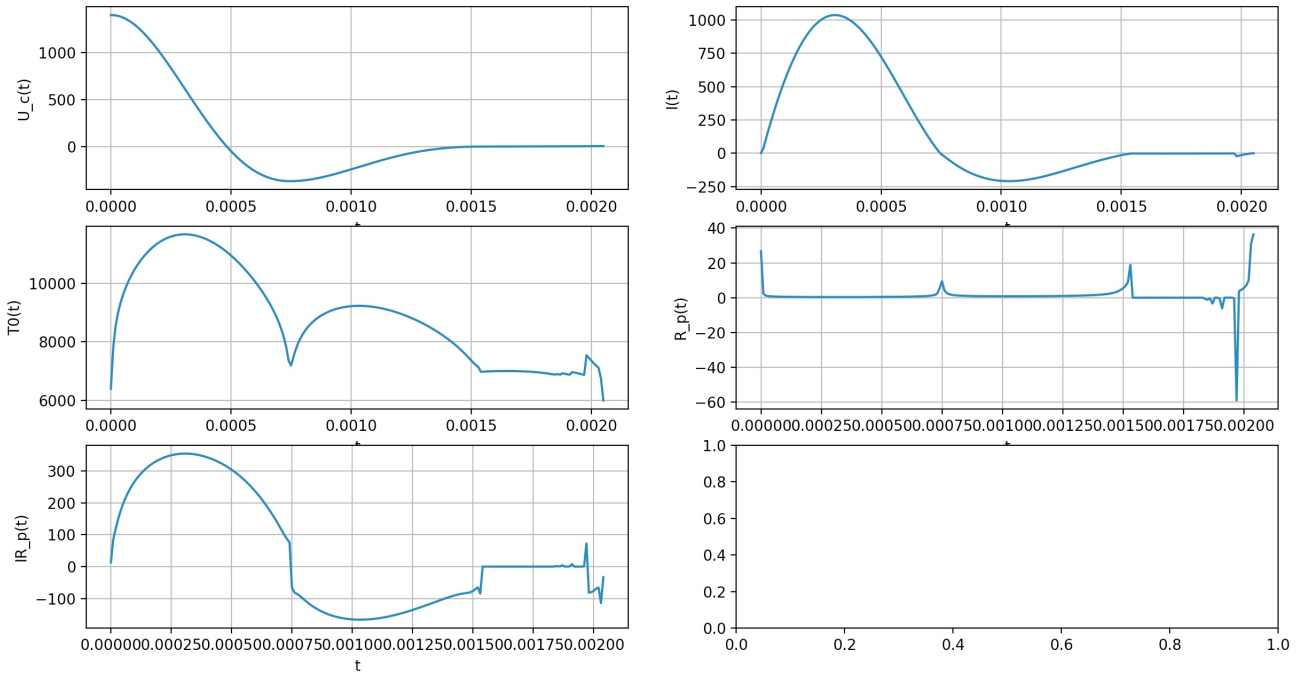
$T_w = 2000$

4 Пример работы программы

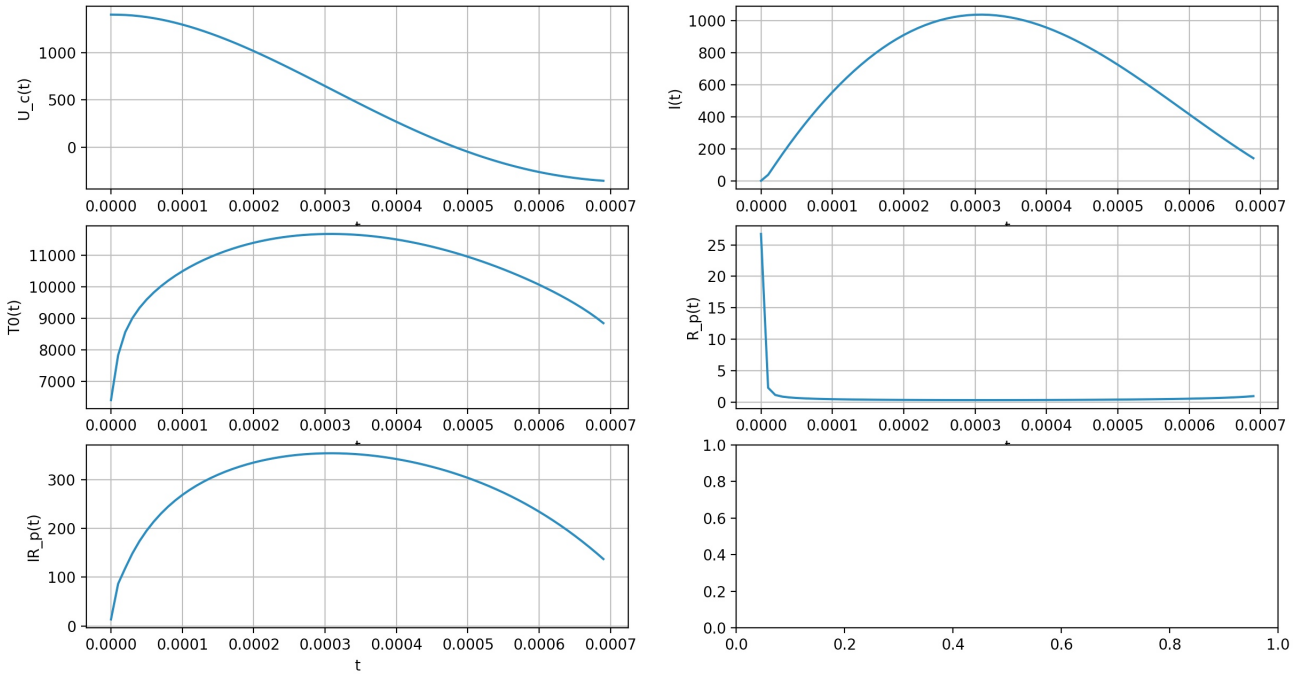
Ниже приведен пример работы программы при $R_k + R_p = 0$ (идеальные условия, без потерь); при шаге $h = 10$ мкс, $t_{max} = 10000$ мкс, $t_{min} = 0$ мкс



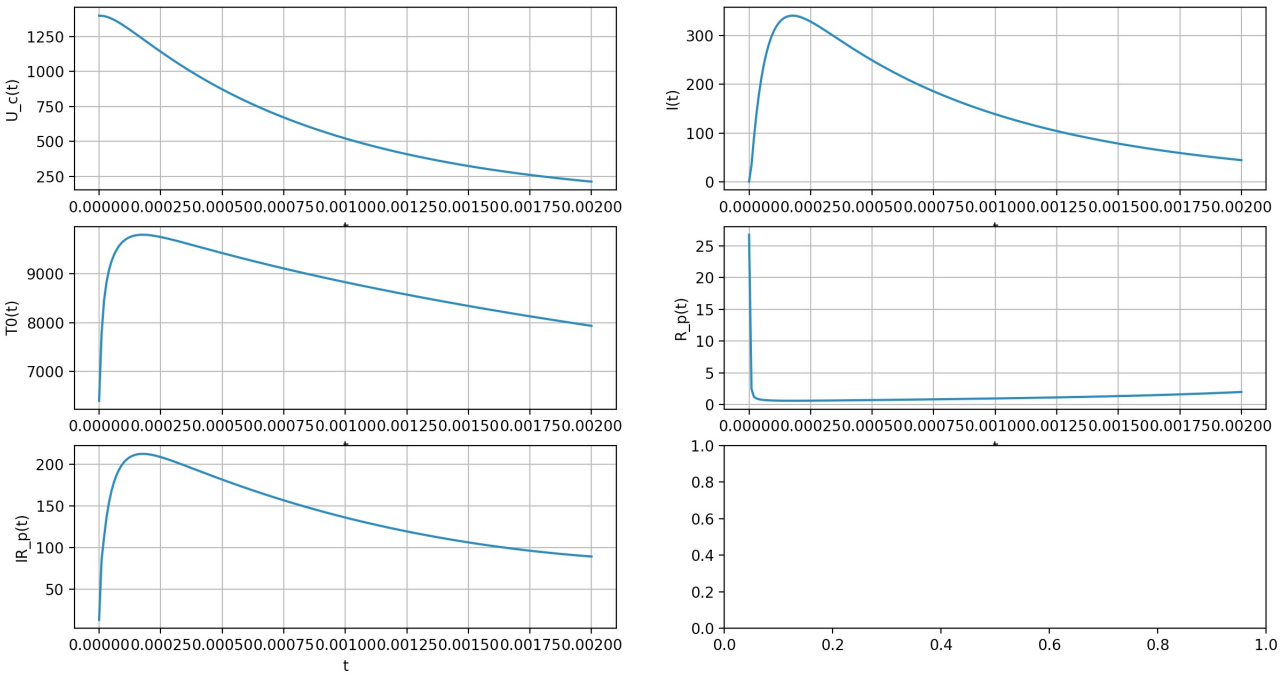
Ниже приведен пример работы программы при шаге $h = 10$ мкс, $t_{max} = 10000$ мкс, $t_{min} = 0$ мкс, $R_k = 0.35$ Ом.



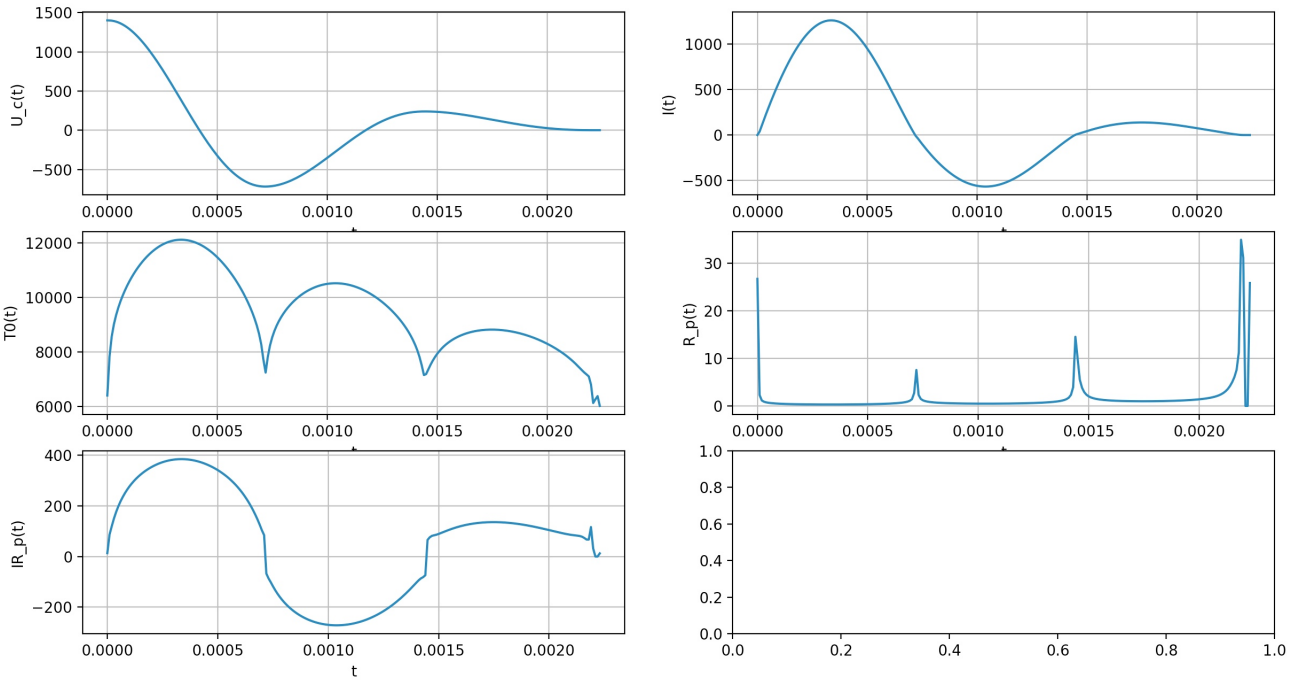
Ниже приведен пример работы программы при шаге $h = 10$ мкс, $t_{max} = 700$ мкс, $t_{min} = 0$ мкс, $R_k = 0.35$ Ом.



Ниже приведен пример работы программы при шаге $h = 10$ мкс, $t_{max} = 2000$ мкс, $t_{min} = 0$ мкс, $R_k = 3.0$ Ом.



Ниже приведен пример работы программы при шаге $h = 10$ мкс, $t_{max} = 10000$ мкс, $t_{min} = 0$ мкс, $R_k = 0.0$ Ом.



5 Листинг кода

Листинг 1: Реализация метода Рунге-Кутта 4-ого порядка точности

```
from interpolation import *

from math import *


def calc_temperature(t0, t_w, m, z):
  return t0 + (t_w - t0) * (z ** m)



def calc_integral_func(z, sigma_from_temperature, t_w, t0, m):
  return z * calculate_value(sigma_from_temperature, calc_temperature(t0,
     t_w, m, z))



def calc_integral_by_trap_method(sigma_from_temperature, t_w, t0, m):
  n, b, a = 1000, 1, 0
  h = (b - a) / n

  result = calc_integral_func(b, sigma_from_temperature, t_w, t0, m)

  a += h
  while (a < b):
    result += 2 * calc_integral_func(a, sigma_from_temperature, t_w, t0, m)
    a += h

  return result * h
```

```
28
29
30
31
32 def calc_r_p(l_e, r, sigma_from_temperature, t_w, t0, m):
33   result = l_e / (2 * pi * r * r)
34   result /= calc_integral_by_trap_method(sigma_from_temperature, t_w, t0, m)
35   return result
36
37
38
39 def calc_f(i, u_c, r_k, l_k, r_p):
40   return (u_c - (r_k + r_p) * i) / l_k
41   #return u_c / l_k
42
43
44 def calc_phi(c_k, i):
45   return -i / c_k
46
47 def calc_k_q_values(h_t, i, u_c, r_k, l_k, c_k, r_p):
48   k, q = [], []
49   k1 = h_t * calc_f(i, u_c, r_k, l_k, r_p)
50   q1 = h_t * calc_phi(c_k, i)
51
52   k2 = h_t * calc_f(i + k1 / 2, u_c + q1 / 2, r_k, l_k, r_p)
53   q2 = h_t * calc_phi(c_k, i + k1 / 2)
54
55   k3 = h_t * calc_f(i + k2 / 2, u_c + q2 / 2, r_k, l_k, r_p)
56   q3 = h_t * calc_phi(c_k, i + k2 / 2)
57
58   k4 = h_t * calc_f(i + k3, u_c + q3, r_k, l_k, r_p)
59   q4 = h_t * calc_phi(c_k, i + k3)
60
61   k.append(k1)
62   k.append(k2)
63   k.append(k3)
64   k.append(k4)
65
66   q.append(q1)
67   q.append(q2)
68   q.append(q3)
69   q.append(q4)
70
71   return k, q
72
73
74 def runge_kutta_part(k, y_n):
75   return y_n + (k[0] + 2 * k[1] + 2 * k[2] + k[3]) / 6
76
77
78 def calc_runge_kutta_fourth(k, q, i, u_c):
79   return runge_kutta_part(k, i), runge_kutta_part(q, u_c)
80
81
82 def calc_values(d):
```

```
83    result = []
84    u_c = d.get("U_c")
85    i = d.get("I")
86    time = d.get("Time")
87    l_e = d.get("L_e")
88    r = d.get("R")
89    t_w = d.get("T_w")
90    r_k = d.get("R_k")
91    l_k = d.get("L_k")
92    c_k = d.get("C_k")
93
94
95    t0_from_i = interp(d.get("I_T_0_m"), 0, 1)
96    m_from_i = interp(d.get("I_T_0_m"), 0, 2)
97
98
99    sigma_from_temperature = interp(d.get("T_sigma"), 0, 1)
100
101   t0 = calculate_value(t0_from_i, log2(i))
102   m = calculate_value(m_from_i, log2(i))
103
104   r_p = calc_r_p(l_e, r, sigma_from_temperature, t_w, t0, m)
105
106
107   result.append([u_c, i, t0, r_p, time[0]])
108
109
110   for j in range(len(time) - 1):
111     h_t = (time[j + 1] - time[j])
112
113     k, q = calc_k_q_values(h_t, i, u_c, r_k, l_k, c_k, r_p)
114
115     i, u_c = calc_runge_kutta_fourth(k, q, i, u_c)
116
117
118     t0 = calculate_value(t0_from_i, log2(fabs(i)))
119     m = calculate_value(m_from_i, log2(fabs(i)))
120     r_p = calc_r_p(l_e, r, sigma_from_temperature, t_w, t0, m)
121
122     result.append([u_c, i, t0, r_p, time[j + 1]])
123
124   return result
```