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»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

По курсу: "Моделирование"

Тема Программно–алгоритмическая реализация метода Рунге-Кутта 4-го порядка точности при решении системы ОДУ в задаче Коши.

Группа ИУ7-63Б

Студент Сукочева А.

Преподаватель Градов В.М.

Москва — 2021 г.

0.1 Постановка задачи

Цель работы. Получение навыков разработки алгоритмов решения задачи Коши при реализации моделей, построенных на системе ОДУ, с использованием метода Рунге-Кутта 4-го порядка точности.

0.1.1 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

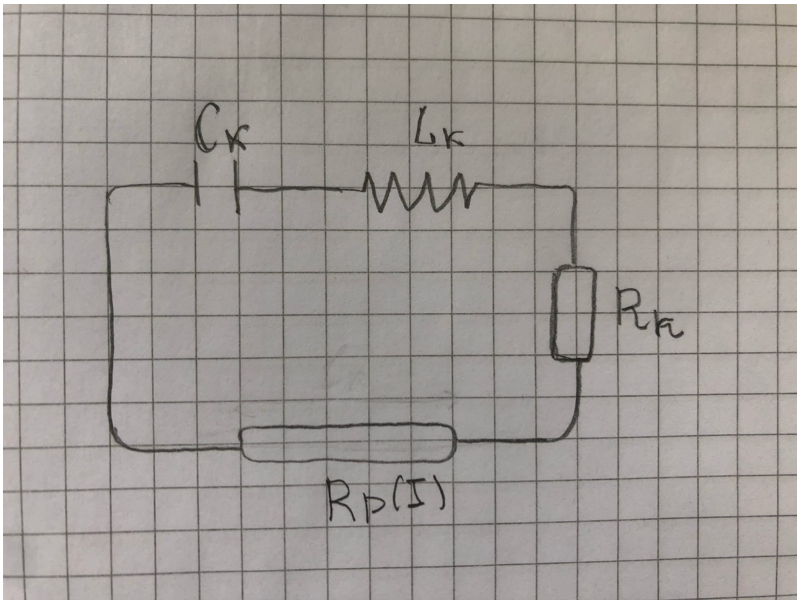


Рис. 1: Разрядный контур

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

Начальные условия:

$$t = 0, I = I_0, U = U_0$$

Здесь I, U - ток и напряжение на конденсаторе.

Сопротивление R_p рассчитать по формуле:

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

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.

Таблица 1

I, A	T_0, 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 2

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
$l_p = 12$ см
$L_k = 187\ 10^{-6}$ Гн
$C_k = 268\ 10^{-6}$ Ф
$R_k = 0.25$ Ом
$U_{co} = 1400$ В
$I_0 = 0...3$ А
$T_w = 2000$ К

0.2 Теоретическая часть

0.2.1 ОДУ

Дано ОДУ (Обыкновенное Дифференциальное уравнение) n-ого порядка (1).

$$F(x, u', u'', ..., u^{(n)} = 0) \tag{1}$$

ОДУ любого порядка может быть сведено к системе ОДУ 1-ого порядка.

0.2.2 Задача Коши

Задача Коши состоит в нахождении решения дифференциального уравнения, удовлетворяющего начальным условиям. Это одна из основных задач теории дифференциальных уравнений.

Имеется задача Коши (3).

$$\begin{cases} u'(x) = f(x, u) \\ u(\xi) = \eta \end{cases} \tag{2}$$

Методы решения ОДУ в задачи Коши:

1. аналитические;
2. приближенно аналитические;
3. численные.

0.2.3 Метод Рунге-Кутта четвертого порядка точности

Преимущества схем Р-К.

1. Достаточно точные.
2. Легко изменить шаг.
3. Методы не требуют перехода к другим методам.
4. Явные.

Дана система уравнений вида:

$$\begin{cases} u'(x) = f(x, u, v) \\ v'(x) = \phi(x, u, v) \\ u(\xi) = \eta_1 \\ v(\xi) = \eta_2 \end{cases} \tag{3}$$

Тогда

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

$$z_{n+1} = z_n + \frac{p_1 + 2p_2 + 2p_3 + p_4}{6}$$

, где

$$k_1 = hf(x_n, y_n, z_n)$$

$$k_2 = hf(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}, z_n + \frac{p_1}{2})$$

$$k_3 = hf(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}, z_n + \frac{p_2}{2})$$

$$k_4 = hf(x_n + h, y_n + k_3, z_n + p_3)$$

$$p_1 = h\phi(x_n, y_n, z_n)$$

$$p_2 = h\phi(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}, z_n + \frac{p_1}{2})$$

$$p_3 = h\phi(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}, z_n + \frac{p_2}{2})$$

$$p_4 = h\phi(x_n + h, y_n + k_3, z_n + p_3)$$

0.3 Реализация

Метод Рунге-Кутта:

```
class Runge
{
    public static Tuple<double, double> Runge_Kutta(
                                        Func<double, double, double, double> f,
                                        Func<double, double, double, double> g,
                                        double xn, double yn, double zn,
                                        double h)
    {
        double k1, k2, k3, k4;
        double p1, p2, p3, p4;

        k1 = h * f(xn, yn, zn);
        p1 = h * g(xn, yn, zn);

        k2 = h * f(xn + h / 2, yn + k1 / 2, zn + p1 / 2);
        p2 = h * g(xn + h / 2, yn + k1 / 2, zn + p1 / 2);

        k3 = h * f(xn + h / 2, yn + k2 / 2, zn + p2 / 2);
        p3 = h * g(xn + h / 2, yn + k2 / 2, zn + p2 / 2);

        k4 = h * f(xn + h, yn + k3, zn + p3);
        p4 = h * g(xn + h, yn + k3, zn + p3);

        double y_next = yn + (k1 + 2 * k2 + 2 * k3 + k4) / 6;
        double z_next = zn + (p1 + 2 * p2 + 2 * p3 + p4) / 6;

        return Tuple.Create(y_next, z_next);
    }
}
```

Интерполяция:

```
class Interpolation
{
    public static double LinearInterpolation(double[] xs, double[] ys, double
        x)
    {
        int left, right;

        if (x <= xs[0])
        {
            left = 0;
            right = 1;
        }
        else if (x >= xs[xs.Length - 1])
        {
            left = xs.Length - 2;
            right = xs.Length - 1;
        }
        else
        {
            // Find the border.
            int i = 0;
            while (x > xs[i])
                i++;

            left = i - 1;
            right = i;
        }
```

```
        return ys[left] + (ys[right] - ys[left]) * (x - xs[left]) / (xs[right]
            - xs[left]);
    }
}
```

Вспомогательные функции:

```
class Functions
{
    static double T_0;
    static double m;

    static public double Rp(double I)
    {
        T_0 = Interpolation.LinearInterpolation(Constants.I, Constants.T_0, I);
        m = Interpolation.LinearInterpolation(Constants.I, Constants.m, I);
        double integral_value = Integral.Trapezoidal(f_integral, 0, 1);
        double denominator = 2 * Math.PI * Constants.R_squared *
            integral_value;
        return Constants.l_e / denominator;
    }

    static double f_integral(double x)
    {
        double sigma_arg = (Constants.T_w - T_0) * Math.Pow(x, m) + T_0;
        double[] new_sigma = new double[Constants.Sigma.Length - 1];
        for (int i = 0; i < new_sigma.Length; i++)
            new_sigma[i] = Math.Log(Constants.Sigma[i]);
        return Math.Exp(Interpolation.LinearInterpolation(Constants.T,
            new_sigma, sigma_arg)) * x;
    }

    static double dI(double I, double U)
    {
        return (U - (Constants.R_k + Rp(I)) * I) / Constants.L_k;
        // return U / Constants.L_k; // for (Rp + Rk) = 0
        // return (U - 200 * I) / Constants.L_k; // for (Rp + Rk) = 200
    }

    static double dU(double I)
    {
        return -I / Constants.C_k;
    }

    public static double f(double x, double I, double U) => dI(I, U);
    public static double g(double x, double I, double z) => dU(I);
}

class Integral
{
    public static double Trapezoidal(Func<double, double> f, double a, double
        b, double step = 0.05)
    {
        double result = f(a) + f(b);
        while (a + step < b)
        {
            a += step;
            result += 2 * f(a);
        }
        return result * (step / 2);
    }
}
```

0.4 Экспериментальная часть

0.4.1 Задание 1

Графики зависимости от времени импульса t: I(t), U(t), $R_p(t)$, произведения I(t) * $R_p(t)$, $T_0(t)$ при заданных выше параметрах. Шаг сетки: 1e-6

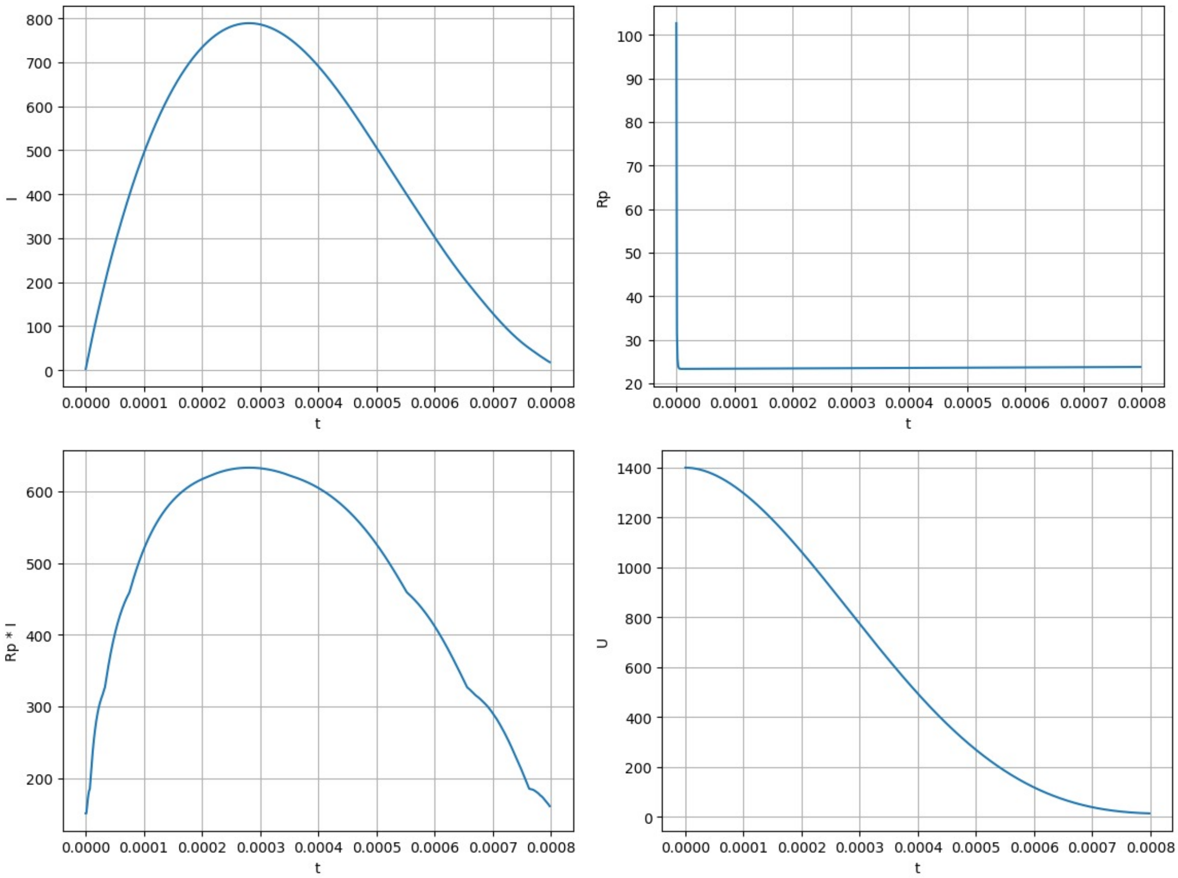


Рис. 2: Графики зависимостей

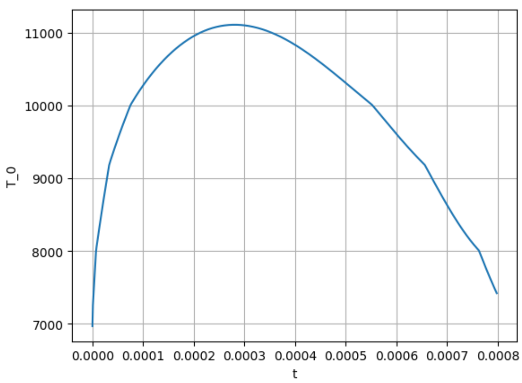


Рис. 3: График зависимости $T_0(t)$

0.4.2 Задание 2

График зависимости I(t) при $R_k + R_p = 0$. Колебания тока являются незатухающими.

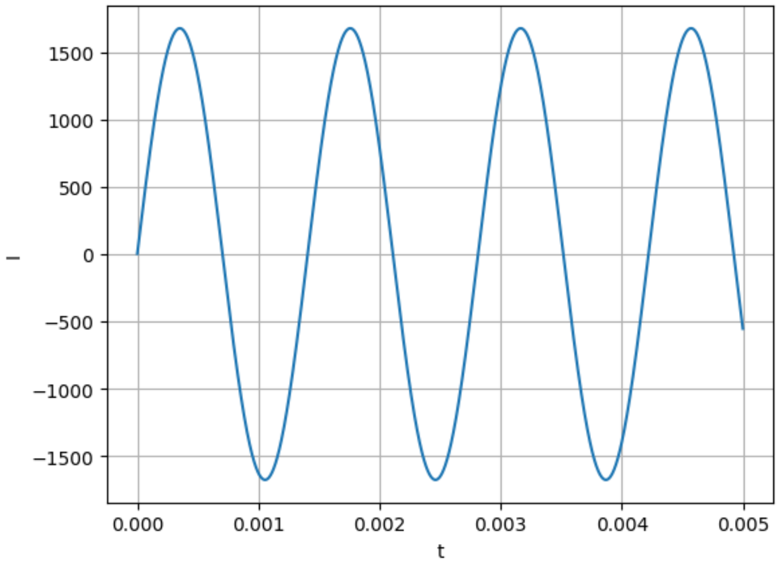


Рис. 4: График зависимости I(t) $(R_k + R_p = 0)$

0.4.3 Задание 3

График зависимости I(t) при $R_k + R_p = const = 200$ Ом.

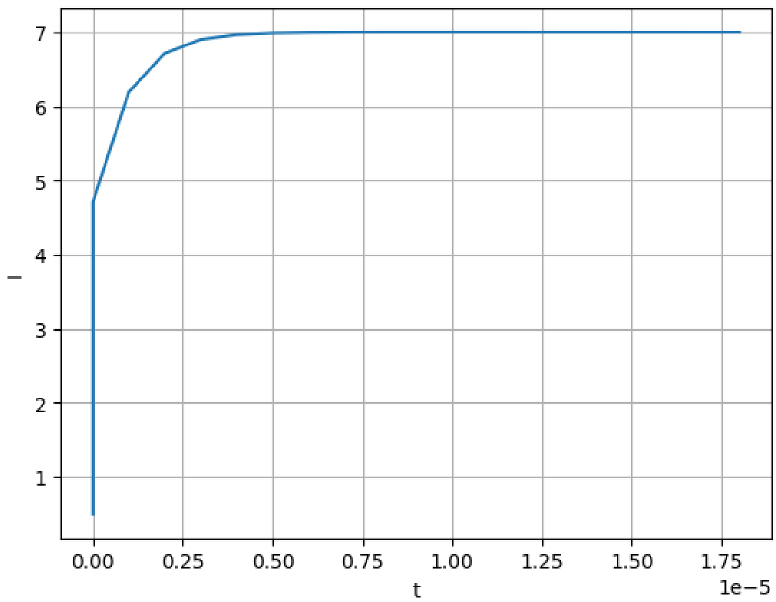


Рис. 5: График зависимости I(t) $(R_k + R_p = 200)$

0.4.4 Задание 4

Результаты исследования влияния параметров контура C_k, L_k, R_k на длительность импульса tимп. апериодической формы. Длительность импульса определяется по кривой зависимости тока от времени на высоте 0.35 * I_{max} , I_{max} - значение тока в максимуме.

Для исследования влияния параметров контура на длительность импульса tимп будем использовать приведенный ниже код.

```
double Imax = arr_I.Max();
double pulseDuration = arr_I.Count(I => I > Imax * 0.35);
```

0.4.5 Исследование влияния C_k на длительность импульса.

Из приведенного исследования видно, что при увеличении C_k длительность импульса увеличивается, а при уменьшении C_k длительность импульса уменьшается.

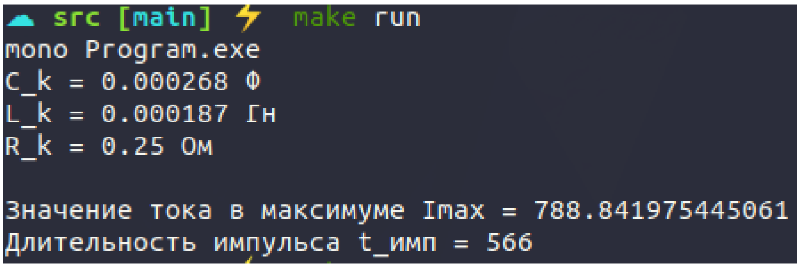


Рис. 6: Исследование влияния C_k на длительность импульса tимп.

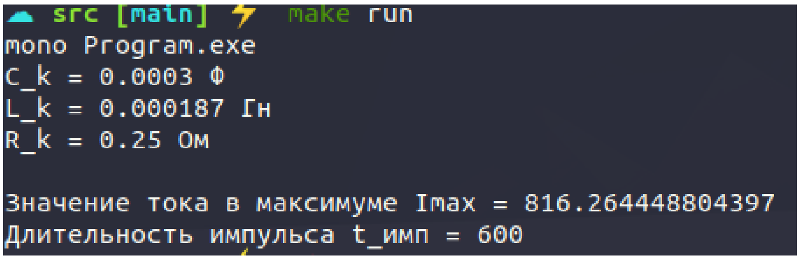


Рис. 7: Исследование влияния C_k на длительность импульса tимп.

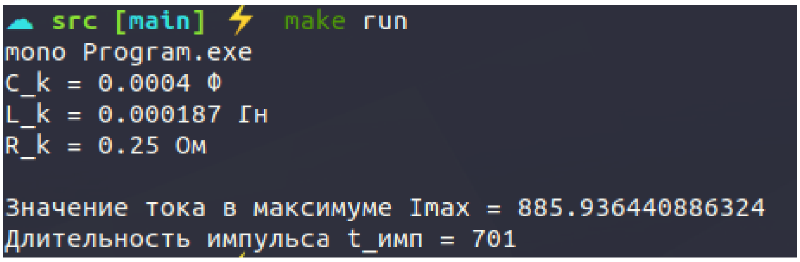


Рис. 8: Исследование влияния C_k на длительность импульса tимп.

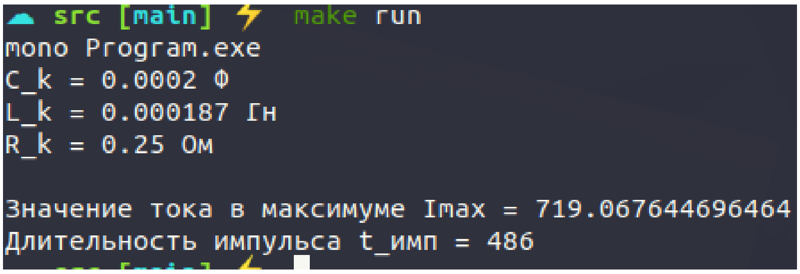


Рис. 9: Исследование влияния C_k на длительность импульса tимп.

0.4.6 Исследование влияния L_k на длительность импульса.

Из приведенного исследования видно, что при увеличении L_k длительность импульса увеличивается, а при уменьшении L_k длительность импульса уменьшается.

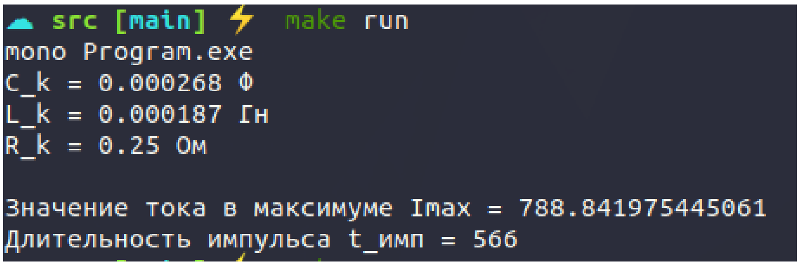


Рис. 10: Исследование влияния L_k на длительность импульса tимп.

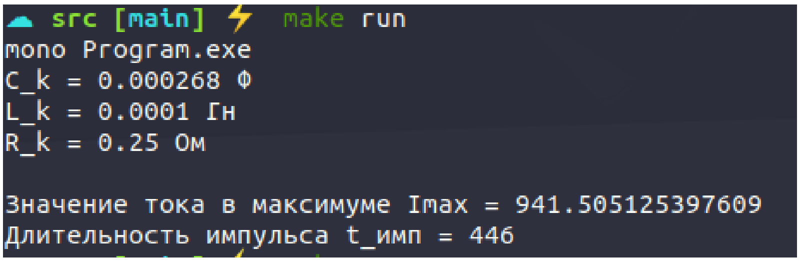


Рис. 11: Исследование влияния L_k на длительность импульса tимп.

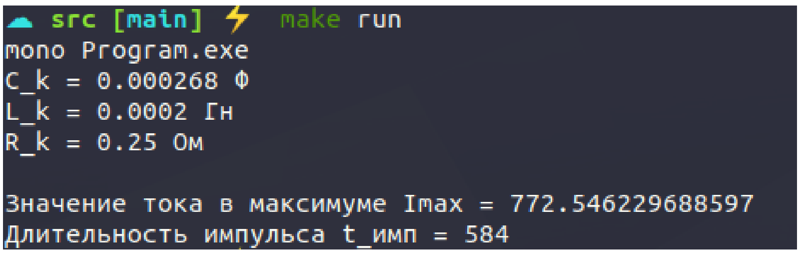
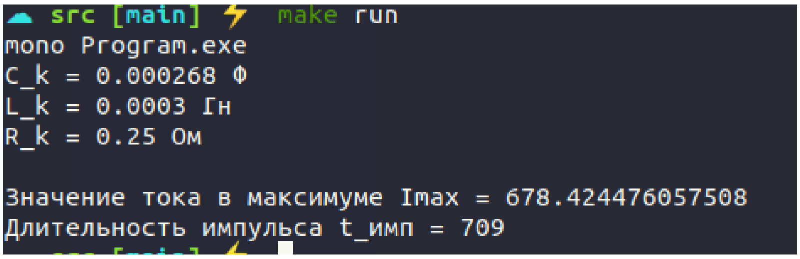


Рис. 12: Исследование влияния L_k на длительность импульса tимп.



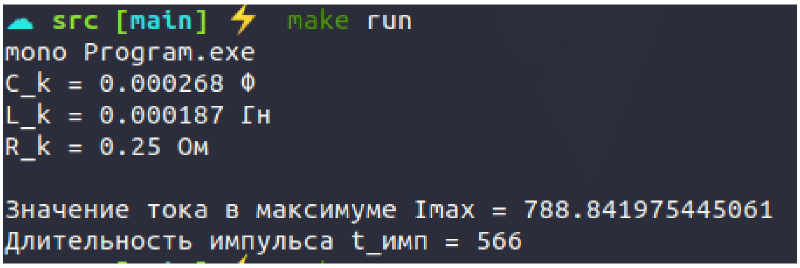
```
src [main] ⚡ make run
mono Program.exe
C_k = 0.000268 Ф
L_k = 0.0003 Гн
R_k = 0.25 Ом

Значение тока в максимуме Imax = 678.424476057508
Длительность импульса t_имп = 709
```

Рис. 13: Исследование влияния L_k на длительность импульса тимп.

0.4.7 Исследование влияния R_k на длительность импульса.

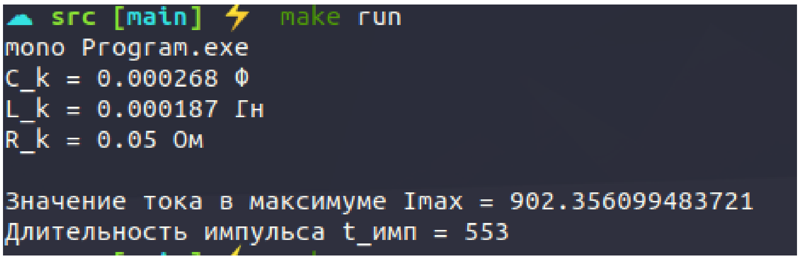
Из приведенного исследования видно, что при увеличении R_k длительность импульса увеличивается, а при уменьшении R_k длительность импульса уменьшается.



```
src [main] ⚡ make run
mono Program.exe
C_k = 0.000268 Ф
L_k = 0.000187 Гн
R_k = 0.25 Ом

Значение тока в максимуме Imax = 788.841975445061
Длительность импульса t_имп = 566
```

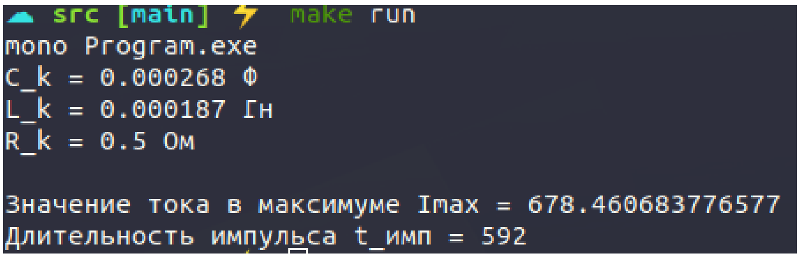
Рис. 14: Исследование влияния R_k на длительность импульса тимп.



```
src [main] ⚡ make run
mono Program.exe
C_k = 0.000268 Ф
L_k = 0.000187 Гн
R_k = 0.05 Ом

Значение тока в максимуме Imax = 902.356099483721
Длительность импульса t_имп = 553
```

Рис. 15: Исследование влияния R_k на длительность импульса тимп.



```
src [main] ⚡ make run
mono Program.exe
C_k = 0.000268 Ф
L_k = 0.000187 Гн
R_k = 0.5 Ом

Значение тока в максимуме Imax = 678.460683776577
Длительность импульса t_имп = 592
```

Рис. 16: Исследование влияния R_k на длительность импульса тимп.

0.5 Ответы на вопросы

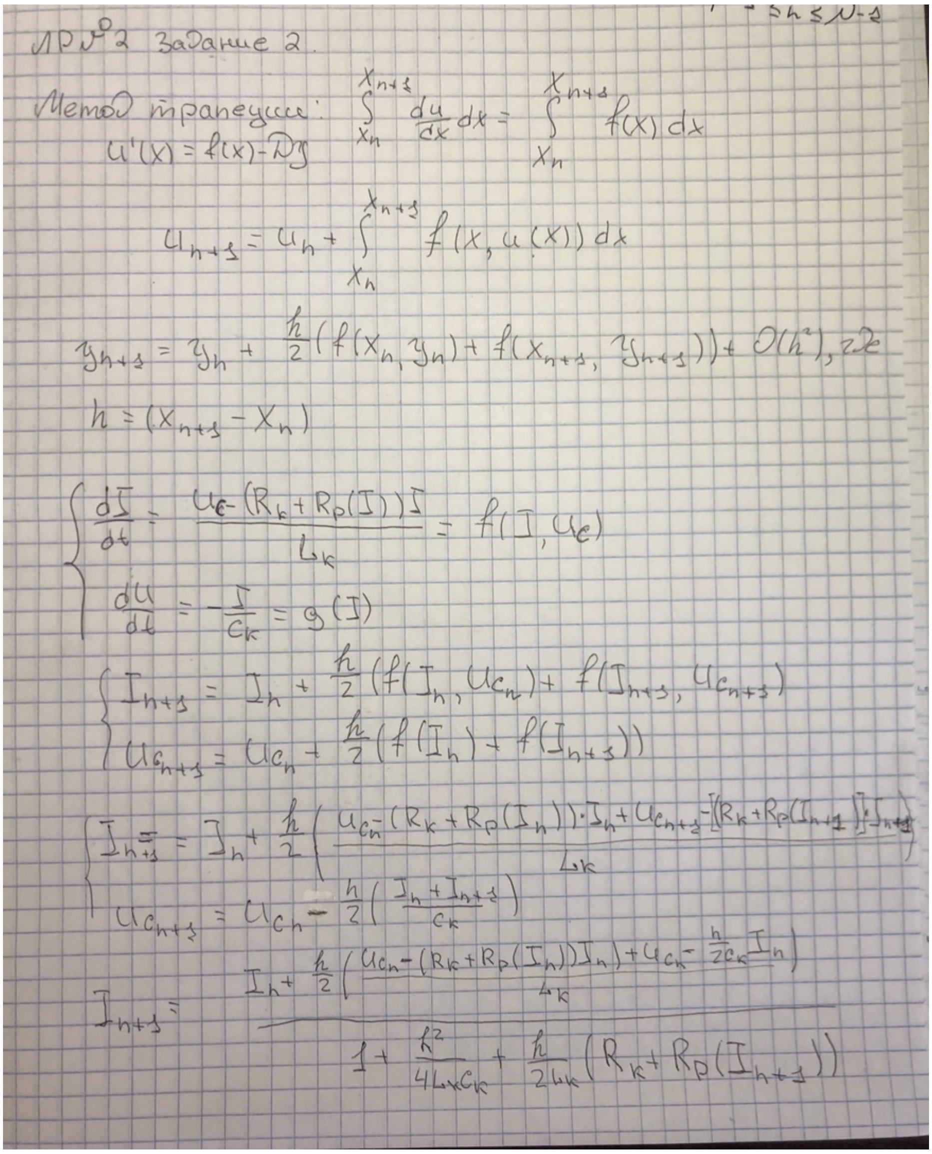
1. Какие способы тестирования программы, кроме указанного в п.2, можете предложить ещё?

Наша программа должна выводить результаты, соответствующие законам физики, поэтому можно изучить вид получаемых графиков при стандартных и измененных параметрах и сравнить с теоретическим ожидаемым видом. Также можно протестировать в случае, когда сумма сопротивлений контура равна нулю, контур обращается в колебательный, колебания незатухающие.

Также можно тестировать программу при разных значениях шага. Малое изменение шага должно приносить малое изменение выходного значения.

Помимо этого можно сравнивать результаты работы нескольких методов разной точности (сравнивать тот метод, который мы использовали с каким-то другим методом).

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



ЛР№2 Задание 2.

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

$u'(x) = f(x) - Dy$

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

$$y_{n+1} = y_n + \frac{h}{2}\left(f(x_n, y_n) + f(x_{n+1}, y_{n+1})\right) + O(h^2), \text{ где}$$

$$h = (x_{n+1} - x_n)$$

$$\begin{cases} \frac{dI}{dt} = \frac{U_c - (R_k + R_p(I))I}{L_k} = f(I, U_c) \\ \frac{dU}{dt} = -\frac{I}{C_k} = g(I) \end{cases}$$

$$\begin{cases} I_{n+1} = I_n + \frac{h}{2}\left(f(I_n, U_{c_n}) + f(I_{n+1}, U_{c_{n+1}})\right) \\ U_{c_{n+1}} = U_{c_n} + \frac{h}{2}\left(f(I_n) + f(I_{n+1})\right) \end{cases}$$

$$\begin{cases} I_{n+1} = I_n + \frac{h}{2}\left(\frac{U_{c_n} - (R_k + R_p(I_n)) \cdot I_n + U_{c_{n+1}} - (R_k + R_p(I_{n+1})) I_{n+1}}{L_k}\right) \\ U_{c_{n+1}} = U_{c_n} - \frac{h}{2}\left(\frac{I_n + I_{n+1}}{C_k}\right) \end{cases}$$

$$I_{n+1} = \frac{I_n + \frac{h}{2}\left(\frac{U_{c_n} - (R_k + R_p(I_n)) I_n + U_{c_n} - \frac{h}{2C_k} I_n}{L_k}\right)}{1 + \frac{h^2}{4 L_k C_k} + \frac{h}{2L_k}\left(R_k + R_p(I_{n+1})\right)}$$

Рис. 17: Вопрос 2

После нахождения I_{n+1}, используя полученное значение, находится Uc_{n+1}.

3. Из каких соображений проводится выбор численного метода того или иного порядка точности, учитывая, что чем выше порядок точности метода, тем он более сложен и требует, как правило, больших ресурсов вычислительной системы?

На выбор численного метода того или иного порядка влияют требуемая точность результата и время, которое необходимо для получения результата. Также на выбор влияет и сама функция. Если функция сильно изменчива, то стоит выбрать малый шаг вычислений. Выбор порядка точности метода Рунге-Кутта зависит от порядка производной в правой части - стоит использовать метод той же точности.

4. Можно ли метод Рунге-Кутта применить для решения задачи, в которой часть условий задана на одной границе, а часть на другой? Например, напряжение по-прежнему задано при t=0, т.е. t=0, U=U_0, а ток задан в другой момент времени, к примеру, в конце импульса, т.е. при t = T, $I = I_T$. Какой можете предложить алгоритм вычислений?

Можно воспользоваться методом стрельбы и свести краевую задачу к некоторой задаче Коши для этой же системы дифференциальных уравнений. Это позволит применить метод Рунге-Кутта. В данном случае можно вычислить значение тока в начальный момент времени или значение напряжения в конечный момент времени.