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

Студент: Платонова О. С.

Группа: ИУ7-65Б

Оценка(баллы) _____

Преподаватель: Градов В. М.

Москва, 2021 г.

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.

Входные данные:

Система электротехнических уравнений, описывающих разрядный контур.

$$\begin{cases} \dfrac{dI}{dt} = \dfrac{U - \left(R_k + R_p(I)\right) I}{L_k} \\ \dfrac{dU}{dt} = -\dfrac{I}{C_k} \end{cases} \qquad (1)$$

где $R_p = \frac{L_p}{2\pi R^2 \int_0^1 \sigma(T(z)) z dz}$

Начальные условия:

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

Выходные данные:

1. Графики зависимости от времени импульса t: *I(t), U(t),* R_p*(t), I(t) ** R_p *(t),* T_0 *(t).*

2. График зависимости *I(t)* при $R_k + R_p = 0$.

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

4. Результаты исследования влияния параметров контура C_k, L_k, R_k на длительность импульса $t_{имп}$ апериодической формы.

<u>Метод Рунге-Кутта 4-го порядка точности.</u>

Численный метод.

$$\begin{cases} I_{n+1} = I_n + \Delta t \dfrac{k_1 + 2k_2 + 2k_3 + k_4}{6} \\ U_{n+1} = U_n + \Delta t \dfrac{q_1 + 2q_2 + 2q_3 + q_4}{6} \end{cases} \qquad (2)$$

где $$\begin{cases} k_1 = h_n f(\Delta t, I_n, U_n) \\ q_1 = h_n g(\Delta t, I_n, U_n) \\ k_2 = h_n f\left(\Delta t + \frac{h_n}{2}, I_n + \frac{k_1}{2}, U_n + \frac{q_1}{2}\right) \\ q_2 = h_n g\left(\Delta t + \frac{h_n}{2}, I_n + \frac{k_1}{2}, U_n + \frac{q_1}{2}\right) \\ k_3 = h_n f\left(\Delta t + \frac{h_n}{2}, I_n + \frac{k_2}{2}, U_n + \frac{q_2}{2}\right) \\ q_3 = h_n g\left(\Delta t + \frac{h_n}{2}, I_n + \frac{k_2}{2}, U_n + \frac{q_2}{2}\right) \\ k_4 = h_n f(\Delta t + h_n, I_n + k_3, U_n + q_3) \\ q_4 = h_n g(\Delta t + h_n, I_n + k_3, U_n + q_3) \end{cases}$$

Параметры T_0, m находятся интерполяцией из таблицы 1:

$$T(I) = T(I_0) + \frac{T(I_1) - T(I_0)}{(I_1 - I_0)} (I - I_0)$$

$$m(I) = m(I_0) + \frac{m(I_1) - m(I_0)}{(I_1 - I_0)} (I - I_0)$$

Погрешности схемы Рунге-Кутта определяются максимальными значениями соответствующих производных.

Листинг

1. Считывание таблиц и параметров из файла.

```
void getTable1(vector<double> &ITable, vector<double> &ToTable, vector<double> &mTable)
{
    ifstream fin(pathTable1);

    if (fin.is_open()) {
        double I = 0, To = 0, m = 0;

        while (!fin.eof()) {
            fin >> I >> To >> m;

            ITable.push_back(I);
            ToTable.push_back(To);
            mTable.push_back(m);
        }
    }

    fin.close();

    ITable.pop_back();
    ToTable.pop_back();
    mTable.pop_back();
}
```

```
void getTable2(vector<double> &TTable, vector<double> &sigTable)
{
    ifstream fin(pathTable2);

    if (fin.is_open()) {
        double T = 0, sig = 0;

        while (!fin.eof()) {
            fin >> T >> sig;

            TTable.push_back(T);
            sigTable.push_back(sig);
        }
    }

    fin.close();

    TTable.pop_back();
    sigTable.pop_back();
}

void getParams(double &R, double &l, double &Lk, double &Ck, double &Rk,
               double &Uco, double &I0, double &Tw)
{
    ifstream fin(pathParams);

    if (fin.is_open()) {
        if (!fin.eof()) {
            fin >> R >> l >> Lk >> Ck >> Rk >> Uco >> I0 >> Tw;
        }
    }

    fin.close();
}
```

2. Метод Рунге-Кутта 4го порядка.

```
int main()
{
    double R = 0, Le = 0, Lk = 0, Ck = 0, Rk = 0,
                  Uco = 0, I0 = 0, Tw = 0;

    double Icur = 0, t = 0;
    double h = 1e-6;

    getParams(R, Le, Lk, Ck, Rk, Uco, I0, Tw);

    getTable1(ITable, ToTable, mTable);
    getTable2(TTable, sigTable);

    vector<double> IPlot, UPlot;
    vector<double> tPlot;

    for (int i = 0; i < 12; i++) {
        Icur = RungeKutta4(Icur, Uco, Le, R, Lk, h, Rk, Ck, Tw, 1);
        Uco = RungeKutta4(Icur, Uco, Le, R, Lk, h, Rk, Ck, Tw, 2);
        t += h;

        tPlot.push_back(t);
        IPlot.push_back(Icur);
        UPlot.push_back(Uco);
    }

    return 0;
}
```

```
double RungeKutta4(double x, double y, double z, double R, double Lk, double hn, double Rk, double Ck, double Tw, int var)
{
    double k1 = hn * f(x, y, z, R, Lk, Rk, Tw);
    double q1 = hn * g(x, Ck);

    double k2 = hn * f(x + k1 / 2, y + q1 / 2, z, R, Lk, Rk, Tw);
    double q2 = hn * g(x + k1 / 2, Ck);

    double k3 = hn * f(x + k2 / 2, y + q2 / 2, z, R, Lk, Rk, Tw);
    double q3 = hn * g(x + k2 / 2, Ck);

    double k4 = hn * f(x + k3, y + q3, z, R, Lk, Rk, Tw);
    double q4 = hn * g(x + k3, Ck);

    double res = 0;
    if (var == 1) {
        res = x + (k1 + 2 * k2 + 2 * k3 + k4) / 6;
    }
    else {
        res = y + (q1 + 2 * q2 + 2 * q3 + q4) / 6;
    }

    return res;
}
```

3. Вспомогательные вычисления и интерполирование.

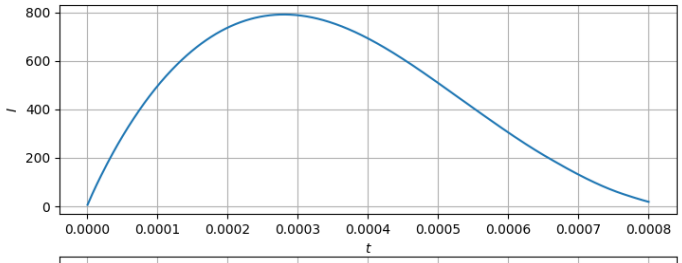
```
double integrate(double I, double Tw)
{
    double a = 0, b = 1;
    double n = 100;
    double h = (b - a) / n;

    double res = (T(a, Tw, I) + T(b, Tw, I)) / 2;
    double x = a;

    for (int i = 0; i < n - 1; i++) {
        x += h;
        res += T(x, Tw, I);
    }
    res *= h;

    return res;
}

double Rp(double Le, double R, double I, double Tw)
{
    return Le / (2 * M_PI * pow(R, 2) * integrate(I, Tw));
}

double f(double y, double z, double Le, double R, double Lk, double Rk, double Tw)
{
    double resRp = Rp(Le, R, fabs(y), Tw);
    RpPlot.push_back(resRp);

    return (z - (Rk + resRp) * y) / Lk;
}

double g(double I, double Ck)
{
    return -I / Ck;
}
```

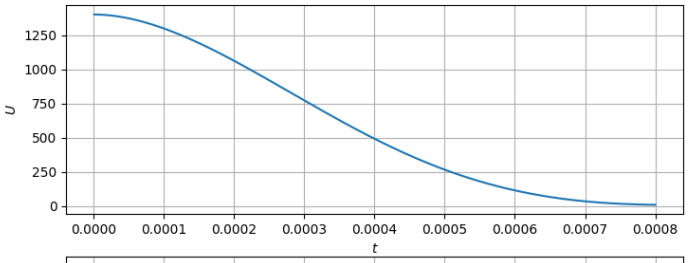
```
double interpolate(double x, vector<double> Xtable, vector<double> f)
{
    int x0 = 0, x1 = 1;

    for (int i = 0; i <= Xtable.size(); i++) {     ⚠ comparison of inte
        x1 = i;

        if (x <= Xtable[i]) {
            break;
        }
    }

    x0 = x1 - 1;

    double res = f[x0] + (f[x1] - f[x0]) / (Xtable[x1] - Xtable[x0])
                 * (x - Xtable[x0]);
    return res;
}
```

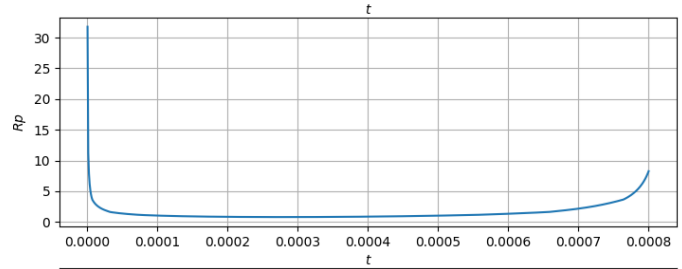
```
static vector<double> ITable, ToTable, mTable;
static vector<double> TTable, sigTable;

static vector<double> RpPlot, T0Plot;

double T(double z, double Tw, double I)
{
    double T0 = interpolate(I, ITable, ToTable);
    T0Plot.push_back(T0);

    double m = interpolate(I, ITable, mTable);

    double T = T0 + (Tw - T0) * pow(z, m);
    double sigma = interpolate(T, TTable, sigTable);

    return z * sigma;
}
```

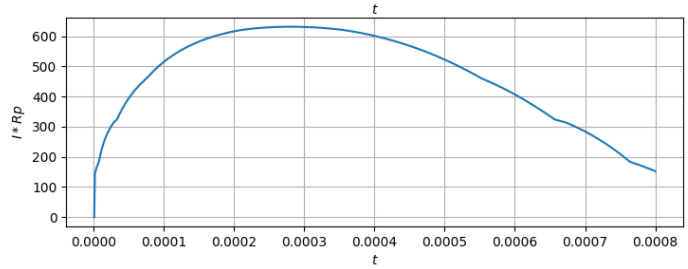
Результаты работы

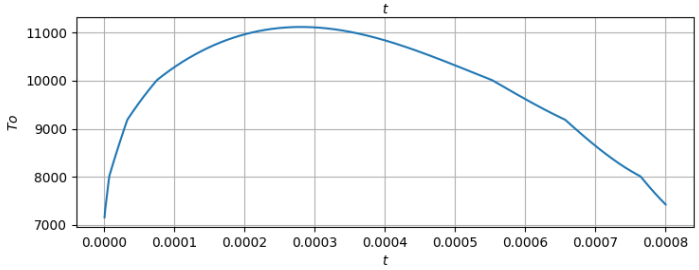
1. Графики зависимости от времени импульса t, при шаге h = 1e-6: *I(t)*, *U(t)*, $R_p(t)$, $I(t) * R_p\ (t)$, $T_0\ (t)$.



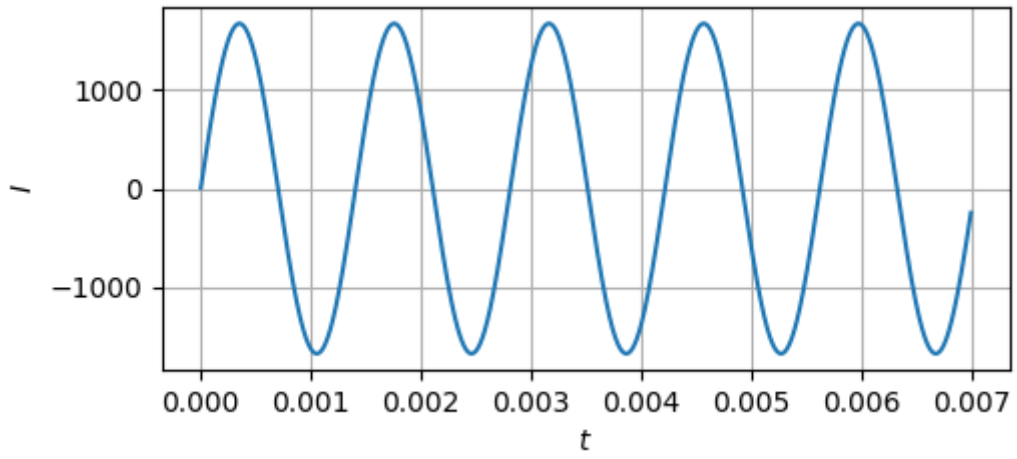






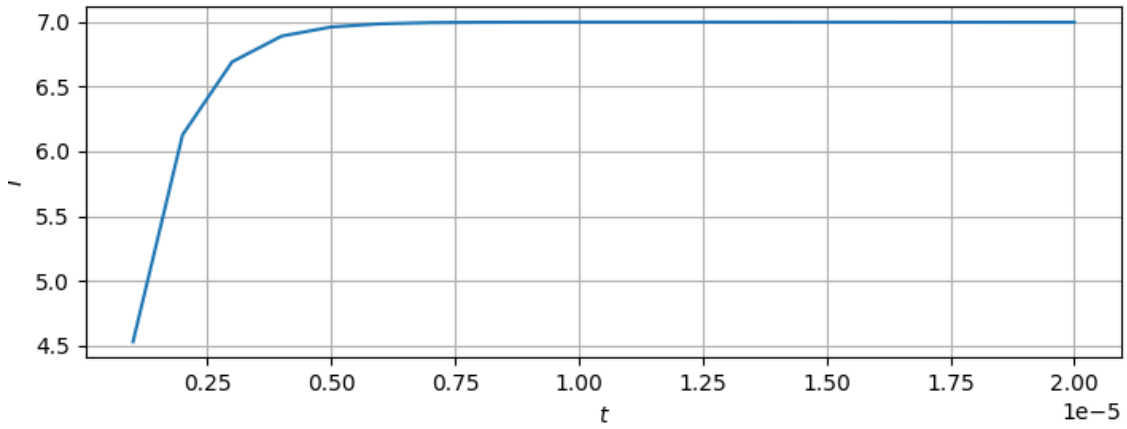


2. График зависимости *I(t)* при $R_k + R_p = 0$ и шаге h = 1e-6.



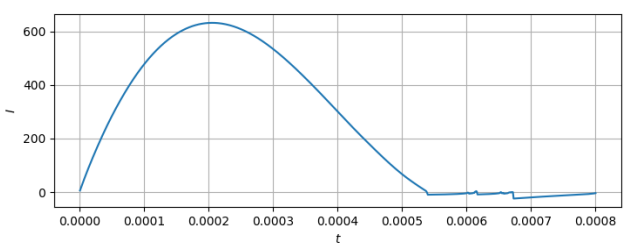
Отметим, что колебания являются незатухающими.

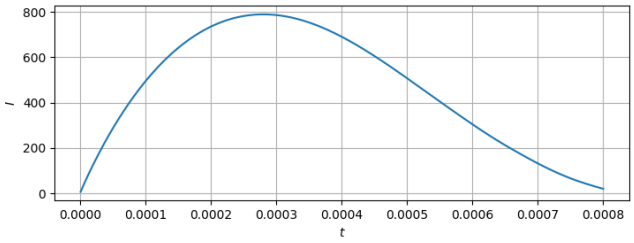
3. График зависимости *I(t)* при $R_k + R_p = const$ = 200 Ом в интервале значений t 0-20 мкс и шаге h = 1e-6.

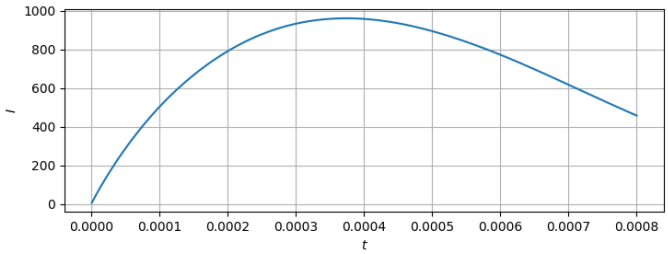


4. Результаты исследования влияния параметров контура C_k, L_k, R_k на длительность импульса $t_{имп}$ апериодической формы.

a) Изменение C_k:



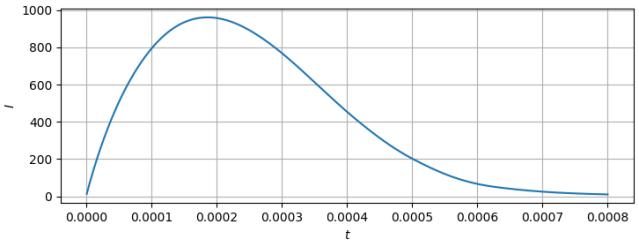


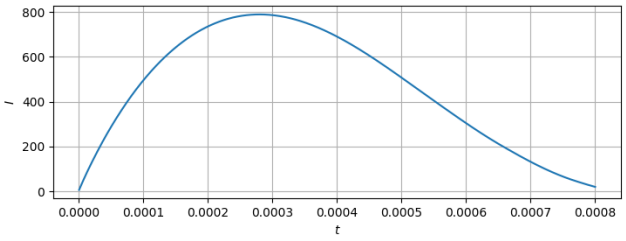


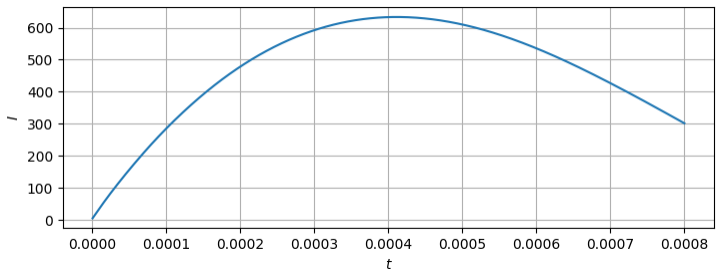
Графики соответствуют значениям C_k = 134e-6, 268e-6, 536e-6.

Следует отметить зависимость $t_{\text{имп}}$ от значения параметра C_k: с увеличением значения, $t_{\text{имп}}$ увеличивается, с уменьшением - уменьшается.

b) Изменение L_k:



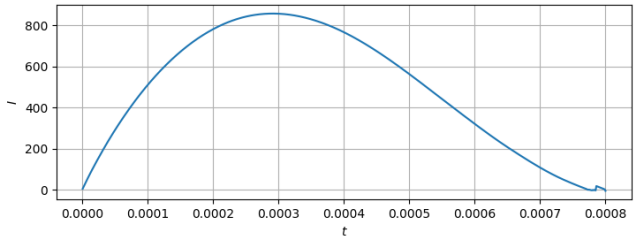


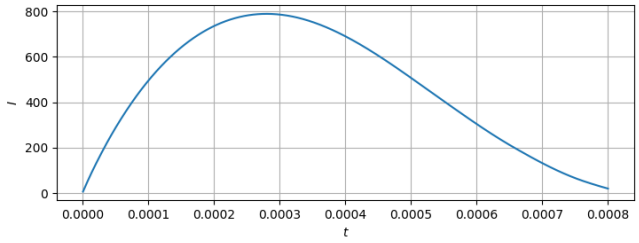


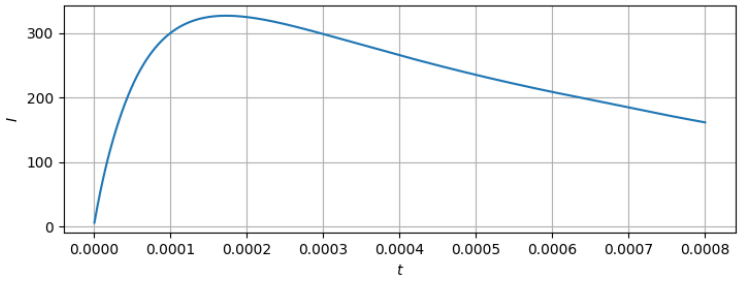
Графики соответствуют значениям L_k = 93e-6, 187e-6, 374e-6.

В данном случае также наблюдается зависимость $t_{\text{имп}}$ от значения параметра L_k: с увеличением значения, $t_{\text{имп}}$ увеличивается, с уменьшением - уменьшаются.

c) Изменение R_k:







Графики соответствуют значениям R_k = 0.125, 0.5, 2.5.

Аналогично предыдущим пунктам, с увеличением значения параметра R_k, наблюдается рост $t_{\text{имп}}$, с уменьшением – уменьшение.

Вопросы.

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

- Тестирование с изменением шага: уменьшение его до тех пор, пока изменение значений не станет пренебрежимо малым.
- Тестирование с изменением значений параметров I_0, Uc_0, R_k. Так, при малых значениях $R_k + R_p$, колебания должны быть затухающими.

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

Метод трапеций:

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

$$u_{n+1} = u_n + \frac{h}{2}\,[f(x_n, y_n) + f(x_{n+1}, y_{n+1})] + O(h^2)$$

В условиях поставленной задачи:

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

$$I_{n+1} = I_n + \frac{h}{2}\,[f(I_n, Uc_n) + f(I_{n+1}, Uc_{n+1})]$$

$$Uc_{n+1} = Uc_n + \frac{h}{2}\,[g(I_n) + g(I_{n+1})]$$

$$I_{n+1} = I_n + \frac{h}{2}\left[\frac{Uc_n - \left(R_k + R_p(I_n)\right)I_n + Uc_{n+1} - \left(R_k + R_p(I_{n+1})\right)I_{n+1}}{L_k}\right]$$

$$Uc_{n+1} = Uc_n - \frac{h}{2}\left[\frac{I_n + I_{n+1}}{C_k}\right]$$

Подставим Uc_{n+1} в выражение для I_{n+1} и выразим I_{n+1}:

$$I_{n+1} = \frac{I_n\,(4L_kC_k - 2C_kh(R_k + R_p(I_n)) - h^2) + 4C_khUc_n}{4L_kC_k + h^2 + 2C_kh\,(R_k + R_p(I_{n+1}))}$$

Полученное уравнение можно решить методом простых итераций или методом Ньютона.

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?

Выбор метода прежде всего зависит от условия задачи. Если функция φ(x, v) непрерывна, ограничена и имеет четвертые производные, которые также непрерывны и ограничены, то допускается применение метода Рунге-Кутта 4-го порядка. В случае невыполнения поставленных выше условий, или отсутствия необходимости решения высокого порядка точности, следует использовать метод Рунге-Кутта 2-го порядка 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 = U0, а ток задан в другой момент времени, к примеру, в конце импульса, т.е. при t = T, I = It. Какой можете предложить алгоритм вычислений?

Краевая задача может быть сведена к задаче Коши для той же системы ДУ методом стрельбы (метод пристрелки). В результате применения метода, допускается применение Рунге-Кутта.