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»

ОТЧЁТ

По лабораторной работе №2

По курсу: «Моделирование»

Тема: «Задача Коши для системы из 2-х ОДУ»

Студент: Керимов А. Ш.

Группа: ИУ7-64Б

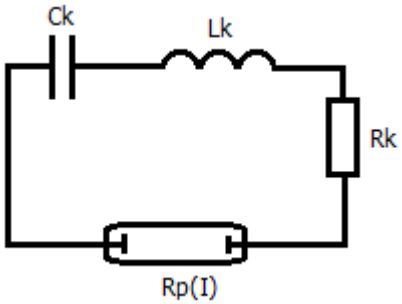
Оценка (баллы): _____

Преподаватель: Градов В.М.

Москва

2020

Дан колебательный контур с газоразрядной трубкой



Получена система дифференциальных уравнений:

$$\begin{cases} L_k \dfrac{\mathrm{d}I}{\mathrm{d}t} + (R_k + R_p)I - U_c = 0, \\ C_k \dfrac{\mathrm{d}U_c}{\mathrm{d}t} = -I. \end{cases}$$

Даны таблицы зависимостей T_0 и m от I, σ от T.
Требуется построить графики зависимостей $I(t)$, $R_p(t)$, $U_c(t)$, $U_p(t)$.
Сопротивление газоразрядной трубки находится в зависимости от силы тока:

$$R_p(I) = \frac{l_э}{2\pi R^2 \int_0^1 \sigma(T(z))z\,\mathrm{d}z}.$$

Система уравнений решается методом Рунге-Кутта 4-го порядка:

$$y_{n+1} = y_n + \frac{k_1 + 2k_2 + 2k_3 + k_4}{6},$$
$$z_{n+1} = z_n + \frac{q_1 + 2q_2 + 2q_3 + q_4}{6},$$

где

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

Листинг 1: Интерполяция

```
template <typename Array>
static double linear_interpolation(double x, const Array& xs, const Array& ys) {
        const auto lower = std::lower_bound(xs.begin(), xs.end(), x);
        const auto i = static_cast<size_t>(std::distance(xs.begin(), lower)) -
            static_cast<size_t>(lower != xs.begin());
        return ys[i] + (ys[i + 1] - ys[i]) * (x - xs[i]) / (xs[i + 1] - xs[i]);
}

static double T0(double I) {
        return linear_interpolation(I, I_TABLE_, T0_TABLE_);
}

static double m(double I) {
        return linear_interpolation(I, I_TABLE_, m_TABLE_);
}

double sigma(double t) const {
        return std::exp(linear_interpolation(std::log(t), log_T_table_, log_sigma_table_));
}

static double T(double T0, double Tw, double z, double m) {
        return T0 + (Tw - T0) * std::pow(z, m);
}
```

Листинг 2: Интегрирование

```
template <typename Function>
static double trapezoidal(const Function& f, double x0, double xn, size_t n) {
        assert(x0 < xn && n > 1);

        const double h = (xn - x0) / static_cast<double>(n);
        double s = (f(x0) + f(xn)) / 2.0;
        for (size_t i = 1; i < n; ++i) {
                s += f(x0 + static_cast<double>(i)*h);
        }
        s *= h;

        return s;
}

double Rp_enable(double I) const {
        const auto integral = trapezoidal([&](double z) { return z * sigma(T(T0(I), Tw_, z,
            m(I))); }, 0, 1, 128);
        return Le_ / (2 * M_PI * R_ * R_ * integral);
}
```

Листинг 3: Решение системы ОДУ

```
static double Ucp(double I, double _Rp) {
        return I * _Rp;
}

double dI(double I, double Uc) const {
        return (Uc - (Rk_ + Rp(std::abs(I))) * I) / Lk_;
}

double dUc(double I) const {
        return -I / Ck_;
}

static std::pair<double, double> Runge_Kutta(
                const std::function<double (double, double, double)>& f,
                const std::function<double (double, double, double)>& g,
                double xn, double yn, double zn,
                double hn) {
        const double k1 = hn * f(xn, yn, zn);
        const double q1 = hn * g(xn, yn, zn);

        const double k2 = hn * f(xn + hn / 2, yn + k1 / 2, zn + q1 / 2);
        const double q2 = hn * g(xn + hn / 2, yn + k1 / 2, zn + q1 / 2);

        const double k3 = hn * f(xn + hn / 2, yn + k2 / 2, zn + q2 / 2);
        const double q3 = hn * g(xn + hn / 2, yn + k2 / 2, zn + q2 / 2);

        const double k4 = hn * f(xn + hn, yn + k3, zn + q3);
        const double q4 = hn * g(xn + hn, yn + k3, zn + q3);

        const double ynp1 = yn + (k1 + 2 * k2 + 2 * k3 + k4) / 6;
        const double znp1 = zn + (q1 + 2 * q2 + 2 * q3 + q4) / 6;

        return {ynp1, znp1};
}

std::pair<double, double> find_next_IUc(double In, double Ucn, double dt) const {
        const auto f = [&](double, double I, double Uc) { return dI(I, Uc); };
        const auto g = [&](double, double I, double   ) { return dUc(I);    };

        return Runge_Kutta(f, g, 0, In, Ucn, dt);
}
```