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
по курсу Моделирование

Студент: Нечитайло Д.В.
Группа: ИУ7-66Б
Преподаватель: Градов В.М.

Москва, 2021 г.

1 Условия и задачи лабораторной работы

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

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

Описание алгоритма: 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.

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

Начальные условия:

$t = 0, I = I_0, U = U_0$. Здесь I, U - ток и напряжение на конденсаторе. Сопротивление R_p рассчитать по формуле:

$$R_p = \frac{l_p}{2\pi R^2 \int\limits_0^1 \sigma(T(z)) z dz}$$

Для функции $T(z)$ применить выражения $T(z) = T_0 + (T_w - T_0) z^m$. Параметры T_0, m находятся интерполяцией из таблицы 1 при известном токе I. Коэффициент электропроводности $\sigma(T)$ зависит от T и рассчитывается интерполяцией из таблицы 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

$l_e = 12$ 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

Метод Рунге-Кутта 4-го порядка точности:

$$y_n + 1 = y_n + \frac{k1 + 2 * k_2 + 2 * k_3 + k_4}{6}$$

$$z_n + 1 = z_n + \frac{p1 + 2 * p_2 + 2 * p_3 + p_4}{6}$$

$$k_1 = h_n f(x_n, y_n, z_n)$$
$$k_2 = h_n f(x_n + \frac{h_n}{2}, y_n + \frac{k_1}{2}, z_n + \frac{p_1}{2})$$
$$k_3 = h_n f(x_n + \frac{h_n}{2}, y_n + \frac{k_2}{2}, z_n + \frac{p_2}{2})$$
$$k_4 = h_n f(x_n + h_n, y_n + k_3, z_n + p_3)$$

$$p_1 = h_n \varphi(x_n, y_n, z_n)$$
$$p_2 = h_n \varphi(x_n + \frac{h_n}{2}, y_n + \frac{k_1}{2}, z_n + \frac{p_1}{2})$$
$$p_3 = h_n \varphi(x_n + \frac{h_n}{2}, y_n + \frac{k_2}{2}, z_n + \frac{p_2}{2})$$
$$p_4 = h_n \varphi(x_n + h_n, y_n + k_3, z_n + p_3)$$

2 Листинг программы

На листинге ?? показан код программы, выводящий агрумент с результатами работы методов, которые необходимо было реализовать по условию лабораторной.

Листинг 2.1: код программы

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace lab2_modeling
{
    class Methods
    {
        public Methods()
        {
            fillData();
        }

        private double[,] ItK = {{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} };

        private double[,] Tsigma = {{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}};


        private static List<double>[] createGraph(int n = 2)
        {
            var graph = new List<double>[n];
            for (int i = 0; i < n; i++)
            {
                graph[i] = new List<double>();
            }
            return graph;
        }

        public List<double>[] graph1 = createGraph(2);
        public List<double>[] graph2 = createGraph(2);
        public List<double>[] graph3 = createGraph(2);
        public List<double>[] graph4 = createGraph(2);
        public List<double>[] graph5 = createGraph(2);

        private Dictionary<string, double> data = new Dictionary<string,
            double>();

        private void fillData()
        {
            data.Add("R", 0.35);
            data.Add("Le", 12);
            data.Add("Lk", 0.000187);
            data.Add("Ck", 0.000268);
            data.Add("Rk", 0.25);
            data.Add("Uc0", 1400);
            data.Add("I0", 0.5);
            data.Add("Tw", 2000);
            data.Add("Tbegin", 0);
            data.Add("Tend", 0.0006);
            data.Add("Tstep", 1e-6);
        }

        private double interpolate(double[,] table, double xValue, int
            xIndex, int yIndex)
        {
            bool interpolateIndexFound = false;
```

```
            double x1 = 0;
            double x2 = 0;
            double y1 = 0;
            double y2 = 0;
            double yResult = 0;
            int len = table.Length;
            if (len == 22)
                len = 11;
            else
                len = 9;

            for (int i = 0; i < len - 1; i++)
            {
                if (table[i,xIndex] <= xValue && xValue <= table[i + 1,
                    xIndex])
                {
                    y1 = table[i, yIndex];
                    y2 = table[i + 1, yIndex];
                    x1 = table[i, xIndex];
                    x2 = table[i + 1, xIndex];
                    interpolateIndexFound = true;
                }
            }
            if (interpolateIndexFound)
            {
                yResult = y1 + ((xValue - x1) / (x2 - x1)) * (y2 - y1);
            }
            else
            {
                if (xValue < table[0, xIndex])
                    yResult = table[0, yIndex];
                if (xValue > table[len - 1, xIndex])
                    yResult = table[len - 1, yIndex];
            }

            return yResult;
        }


        private double integrateSimpson(double I)
        {
            double n = 40;
            double begin = 0;
            double end = 1;
            double width = (end - begin) / n;
            double result = 0;
```

```
            double x1, x2;
            for (double step = 0; step < n; step++)
            {
                x1 = begin + step * width;
                x2 = begin + (step + 1) * width;
                result += (x2 - x1) / 6.0 * (sigmaFunc(I, x1) + 4.0 *
                   sigmaFunc(I, 0.5 * (x1 + x2)) + sigmaFunc(I, x2));
            }
            return result;
        }

        private double calculateRp(double I)
        {
            double R = data["R"];
            double integral = integrateSimpson(I);
            return data["Le"] / (2 * Math.PI * R * R * integral);
        }

        private double f_PHI(double t, double I, double U)
        {
            return -1 / data["Ck"] * I;
        }

        private double getTz(double T0, double m, double r)
        {
            double z = r;
            return (data["Tw"] - T0) * Math.Pow(z, m) + T0;
        }

        private double sigmaFunc(double I, double z)
        {
            double m = interpolate(ItK, I, 0, 2);
            double T0 = interpolate(ItK, I, 0, 1);
            double Tz = getTz(T0, m, z);
            double sigma = interpolate(Tsigma, Tz, 0, 1);
            return z * sigma;
        }

        private double functionF_4(double t, double I, double U)
        {
            double Rp = calculateRp(I);
            graph3[0].Add(t);
            graph3[1].Add(Rp);
            graph4[0].Add(t);
            graph4[1].Add(I * Rp);
            return (U - (data["Rk"] + Rp) * I) / data["Lk"];
```

```
        }

        private double[] Runge4(double xn, double yn, double zn, double hn)
        {
            double[] result = new double[2];

            double hn2 = hn / 2;
            double k1 = hn * functionF_4(xn, yn, zn);
            double q1 = hn * f_PHI(xn, yn, zn);

            double k2 = hn * functionF_4(xn + hn2, yn + k1 / 2, zn + q1 / 2);
            double q2 = hn * f_PHI(xn + hn2, yn + k1 / 2, zn + q1 / 2);

            double k3 = hn * functionF_4(xn + hn2, yn + k2 / 2, zn + q2 / 2);
            double q3 = hn * f_PHI(xn + hn2, yn + k2 / 2, zn + q2 / 2);

            double k4 = hn * functionF_4(xn + hn, yn + k3, zn + q3);
            double q4 = hn * f_PHI(xn + hn, yn + k3, zn + q3);

            double yn_1 = yn + (k1 + 2 * k2 + 2 * k3 + k4) / 6;
            double zn_1 = zn + (q1 + 2 * q2 + 2 * q3 + q4) / 6;

            result[0] = yn_1;
            result[1] = zn_1;
            return result;
        }

        public void beginCalculations()
        {
            double t = data["Tbegin"];
            double tmax = data["Tend"];
            double I = data["I0"];
            double Uc = data["Uc0"];
            double hn = data["Tstep"];
            for (double i = t; i < tmax + hn; i += hn)
            {
                graph1[0].Add(i);
                graph1[1].Add(I);
                graph2[0].Add(i);
                graph2[1].Add(Uc);
                graph5[0].Add(i);
                graph5[1].Add(interpolate(ItK, I, 0, 1));
                var result = Runge4(i, I, Uc, hn);
                I = result[0];
```

```
                    Uc =result[1];
                }
            }
        }

}
```

Результат работы программы:

На рисунках 2.1 - 2.5 графики зависимости от времени импульса $t : I(t), U(t), R_p(t), I(t) * R_p(t), T_0(t)$ при заданных в условиях параметрах разрядного контура.

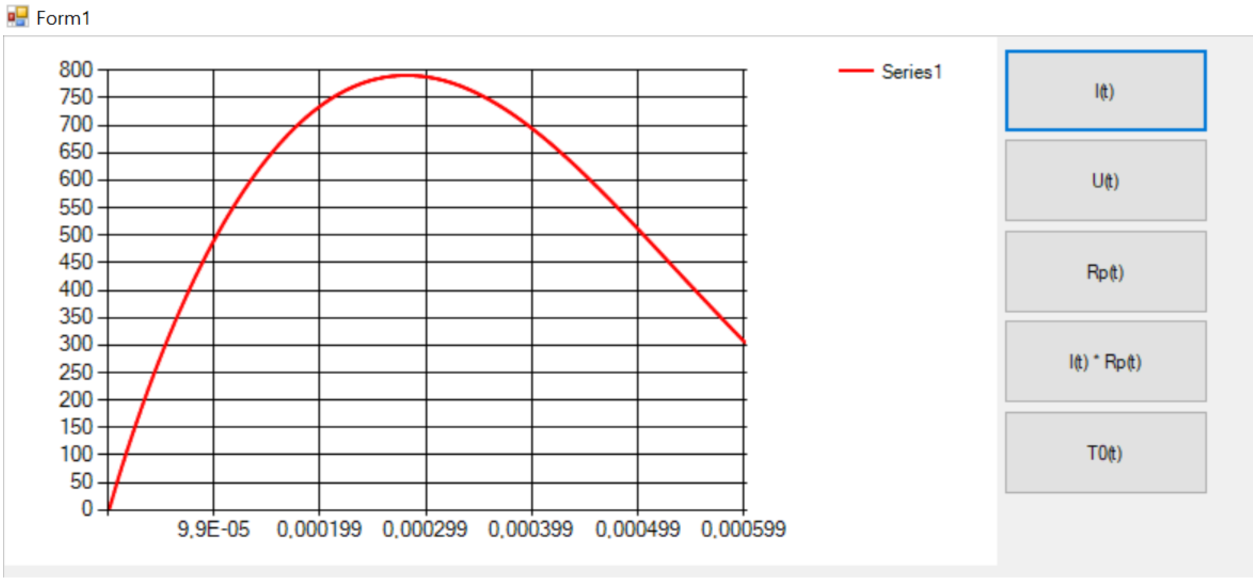


Рисунок 2.1: график $I(t)$

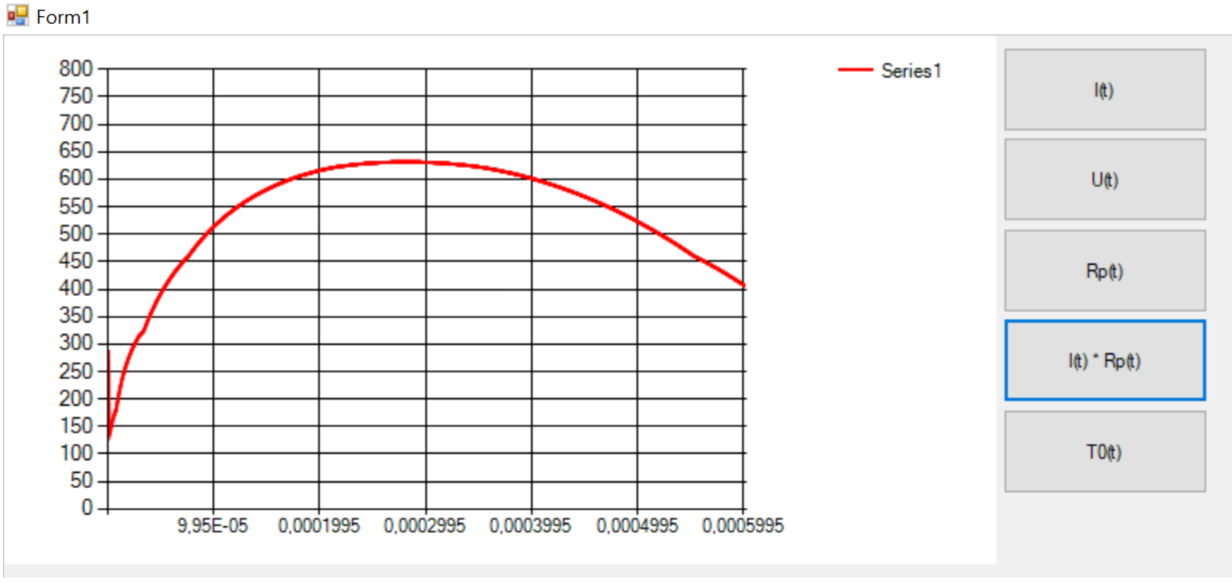


Рисунок 2.2: график $I(t) * R_p(t)$

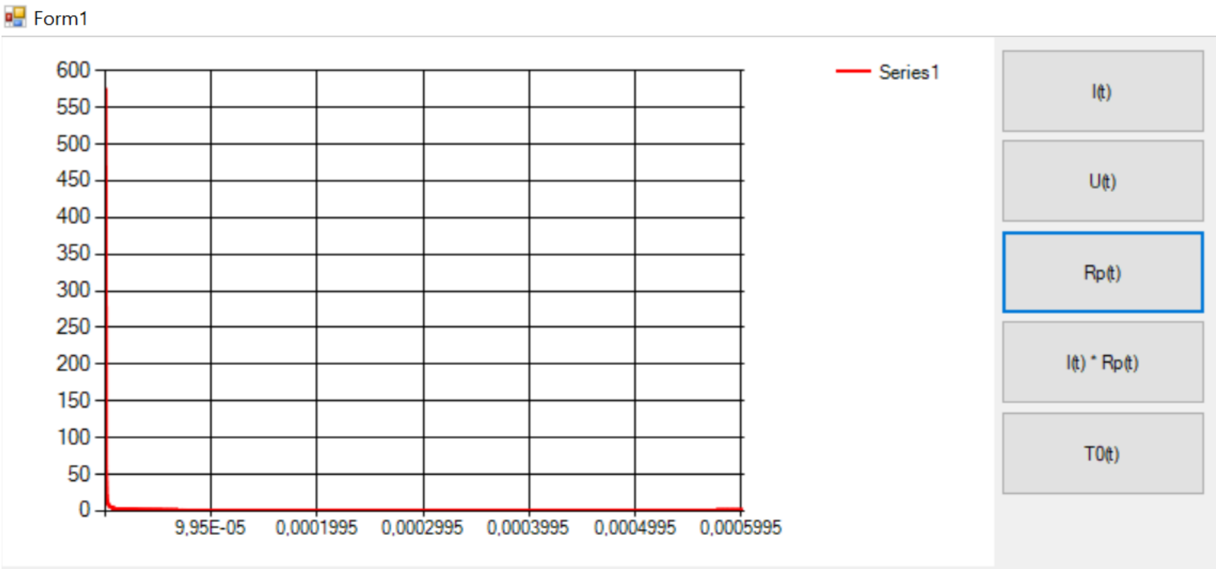


Рисунок 2.3: график $R_p(t)$

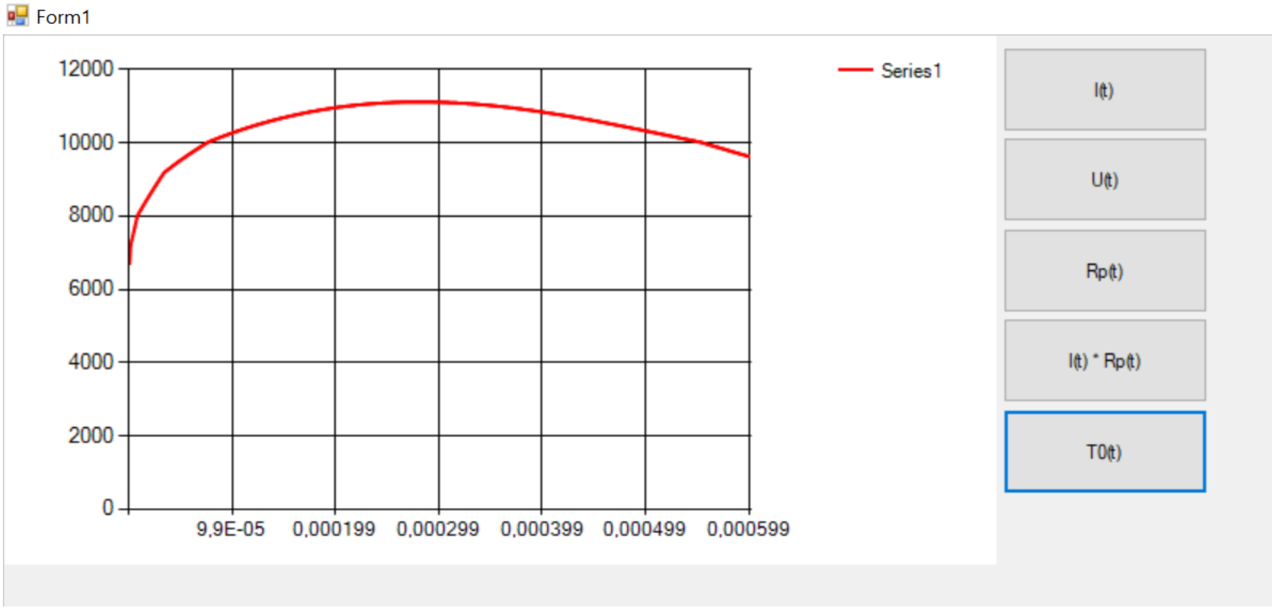


Рисунок 2.4: график $T_0(t)$

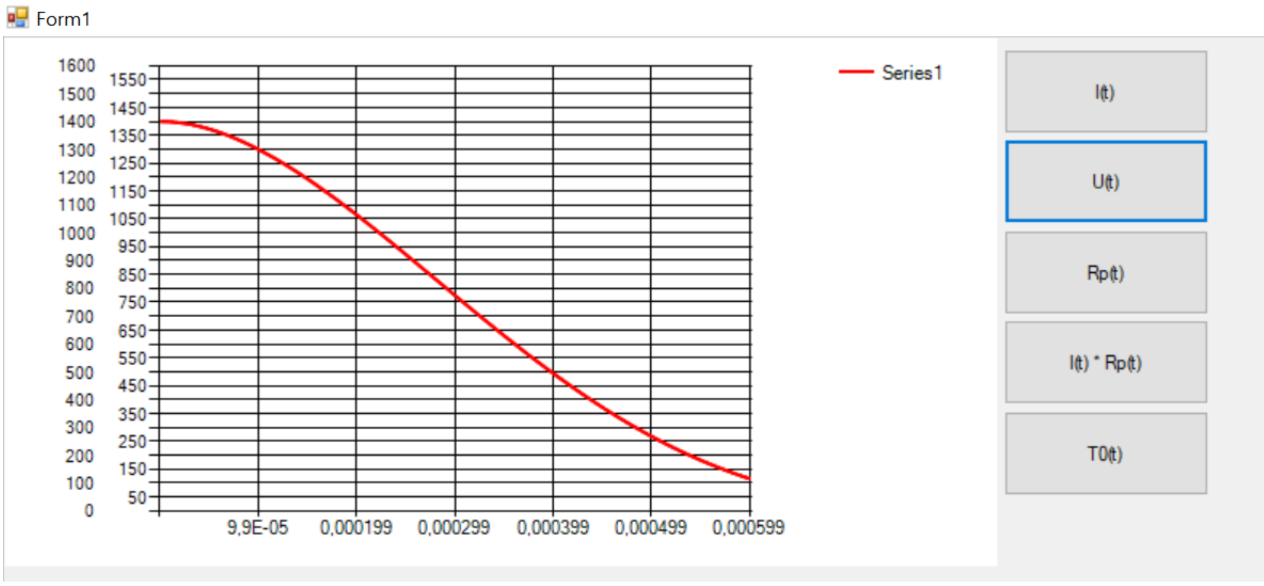
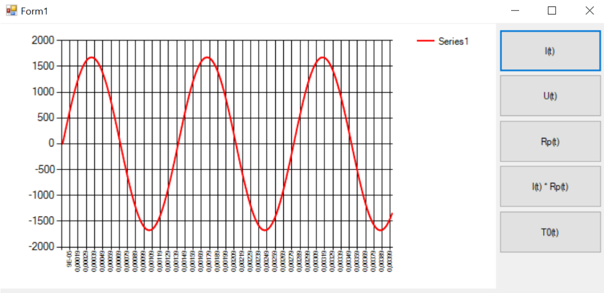
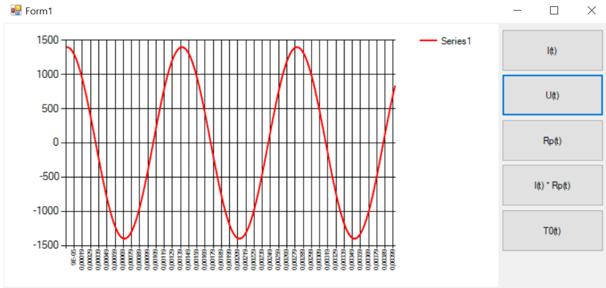


Рисунок 2.5: график $U(t)$

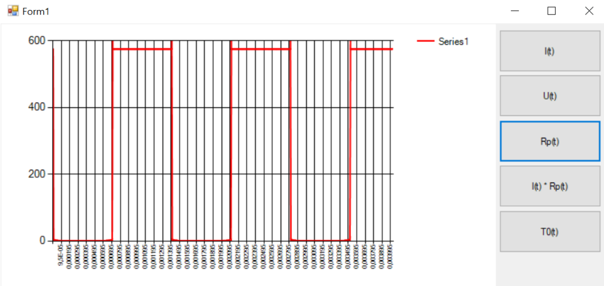
Графики зависимостей $I(t), U(t), R_p(t), I(t) * R_p(t)$ при $R_k + R_p = 0$ приведены на рисунке 2.6:



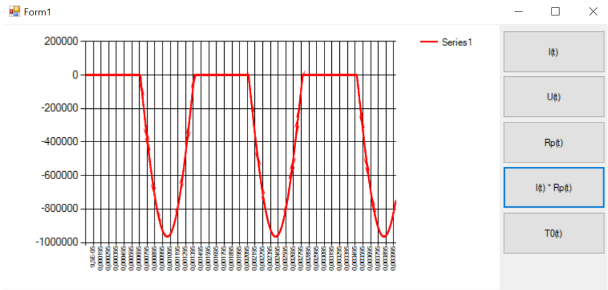
a)



b)



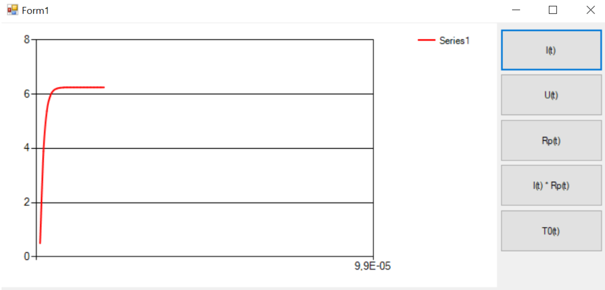
c)

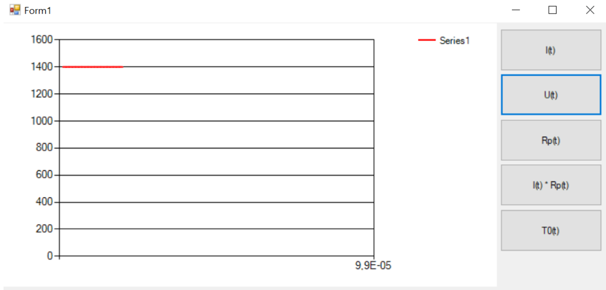


d)

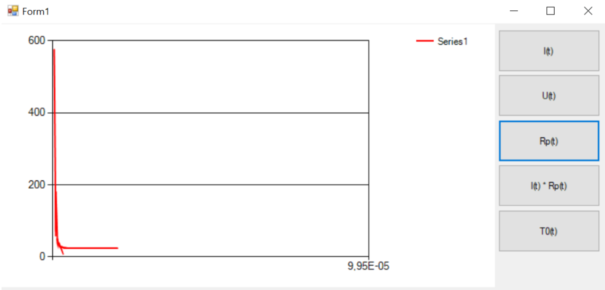
Рисунок 2.6: Графики: a) I(t), b) U(t), c) $R_p(t)$, d) $I(t) * R_p(t)$.

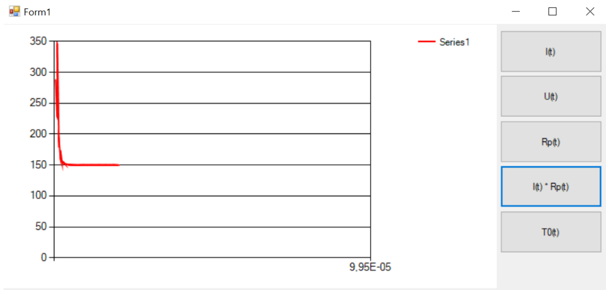
Графики зависимости $I(t), U(t), R_p(t), I(t) * R_p(t)$ при $R_k = 200$ Ом в интервале значений t 0-20 мкс приведены на рисунке 2.7





a) b)

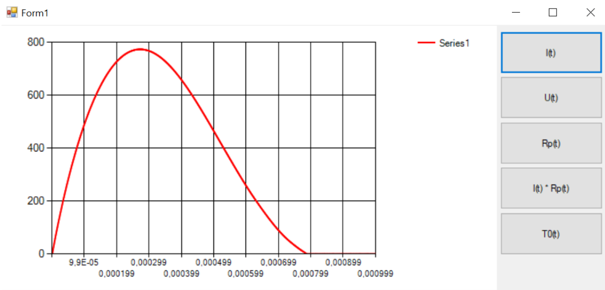


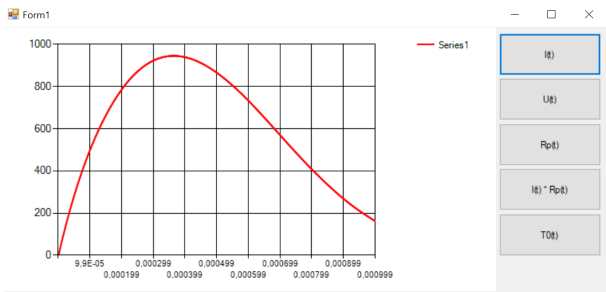


c) d)

Рисунок 2.7: Графики: a) I(t), b) U(t), c) $R_p(t)$, d) $I(t) * R_p(t)$.

Результаты исследования влияния параметров контура C_k, L_k, R_k на длительность импульса t имп. апериодической формы.





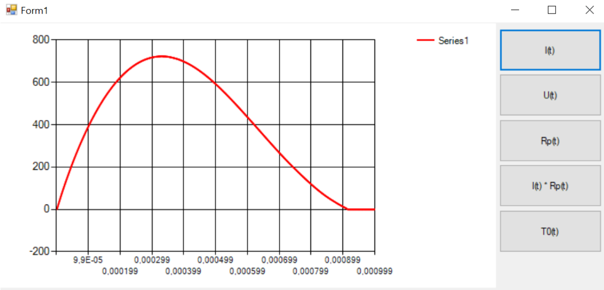
a) b)

Рисунок 2.8: Графики: a) C_k = 0,00025, b) C_k = 0,0005

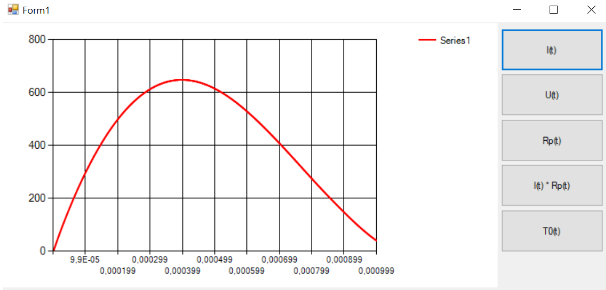
a) $C_k = 0.000250$ $\Delta t = 0.000536$

b) $C_k = 0.000500$ $\Delta t = 0.000804$

При увеличении $_k$ в 2 раза, Δt увеличилось в 1.5 раза



a)



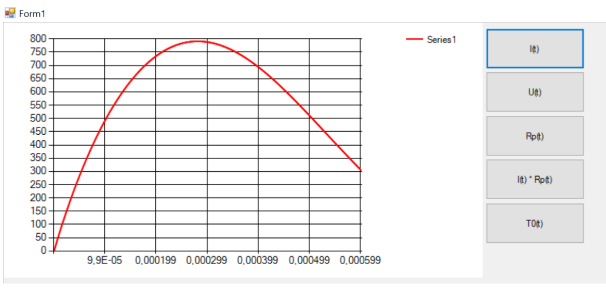
b)

Рисунок 2.9: Графики: a) $L_k = 0.000250$, b) $L_k = 0.000350$

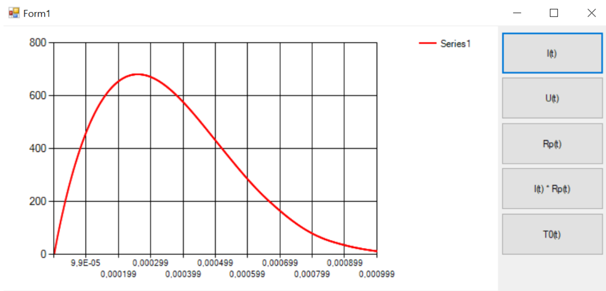
a) $L_k = 0.000250$ $\Delta t = 0.000536$

b) $L_k = 0.000350$ $\Delta t = 0.000777$

При увеличении L_k в 1.4 раза, Δt увеличилось в 1.45 раза



a)



b)

Рисунок 2.10: Графики: a) $R_k = 0.25$, b) $R_k = 0.5$

$R_k = 0.25$ $\Delta t = 0.000575$

$R_k = 0.5$ $\Delta t = 0.000715$

При увеличении R_k в 2 раза, Δt увеличилось в 1.24 раза

3 Ответы на вопросы

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

Для тестирования данной программы можно сравнить результаты работы методов разной точности, например, усовершенствованный метод Эйлера, который имеет 2-й порядок точности и меньшую погрешность, чем метод Эйлера 1-го порядка.

Так же для решения поставленной задачи можно воспользоваться приближенным аналитическим методом, например, методом Пикара.

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

Рассмотрим выражение $u'(x) = f(x)$

Проинтегрируем обе части от x_n до x_{n+1}:

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

Взяв интеграл, получим:

$$u_n + 1 - u_n = \int\limits_{x_n}^{x_{n+1}} f(x)dx$$

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

Вычислим интеграл методом трапеций:

$$y_n + 1 = y_n + h * (\frac{f(x_n) + f(x_{n+1})}{2})$$

Применим данный метод к нашей задаче. По условию задана система уравнений:

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

Начальные условия: $t = 0, I = I_0, U_c = U_0$.

Введём обозначения: $\frac{dI}{dT} \equiv f_1(I, U)$; $\frac{dU}{dt} \equiv f_2(I)$.

Воспользуемся методом трапеций, получим:

$$
\begin{cases}
I_{n+1} = I_n + h(\frac{f_1(I_n, U_n) + f_1(I_{n+1}, U_{n+1})}{2}) \\
U_{n+1} = U_n + h(\frac{f_2(I_n) + f_2(I_{n+1})}{2}) \\
t = 0 \\
I = I_0 \\
U = U_0
\end{cases}
$$

Подставим выражения $f_1(I, U)$ и $f_2(I)$:

$$
I_{n+1} = I_n + h(\frac{U_n - (R_k + R_p(I_n))I_n + U_{n+1} - (R_k + R_p(I_{n+1}))I_{n+1}}{2L_k})
$$

$$
U_{n+1} = U_n - h(\frac{I_n + I_{n+1}}{2C_k})
$$

Подставим выражение для U_{n+1} из второго уравнения в первое и решим относительно I_{n+1}. Получится выражение, которое решается методом простой итерации. Зная начальные условия и подставив их, мы можем получить I_1, которое затем можно подставить в (2), чтобы получить U_1. Последующие приближения можно получить таким же образом. Этот процесс продолжаем, пока не будет достигнута необходимая точность, то есть пока $|I_{n+1} - I_n| > \epsilon$.

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?

Для метод четвёртого порядка точности должны быть четвёртые производные ограниченные, то есть правая часть должна быть непрерывна и ограничена вместе со своими четвёртыми производными. Если это не так, то метод четвёртого порядка точности не обеспечивает этот порядок. Для метода второго порядка правая часть должна быть непрерывна и ограничена вместе со своими производными до второго порядка. Если это не так, то метод второго порядка точности не обеспечивают этот порядок и следует использовать метод Эйлера.

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?

Можно, только сначала нужно методом стрельбы свести эту задачу к задаче Коши для заданной системы уравнений, затем применить метод Рунге-Кутта. В заданном примере нужно найти значение напряжения в момент времени $t = T$, такое чтобы выполнялось краевое условие для тока $I = IT$.