Министерство образования и науки 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 Робототехники и комплексной автоматизации

КАФЕДРА Системы автоматизированного проектирования (РК-6)

ОТЧЕТ О ВЫПОЛНЕНИИ ДОМАШНЕЙ РАБОТЫ

Студент	Никифорова Ирина Андреевна
Группа	РК6-71б
Тип задания	домашняя работа
Тема работы	расчет потенциала в электрической схеме
Вариант	72

Студент _____ **Никифорова И. А.**
подпись, дата *фамилия, и.о.*

Преподаватель _____ **Трудоношин В. А.**
подпись, дата *фамилия, и.о.*

Оценка _____

Москва, 2019 г.

Оглавление

Задание на домашнюю работу

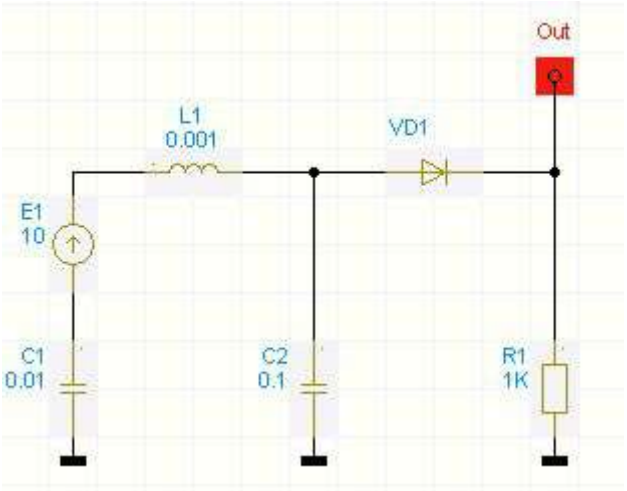

Рис. 1. Заданная схема №**2**.

Необходимо рассчитать потенциал в заданной точке на схеме (рис. 1) в течение времени T = 1e-3c. Для расчета необходимо использовать **расширенный узловой метод** формирования математической модели.

Теоретическое решение

Для использования выбранного метода диод необходимо разложить в следующую эквивалентную схему:

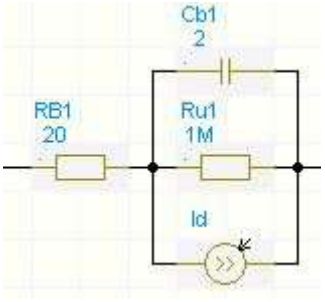

Рис. 2. Схема диода.

На рис.2 ток через диод выражается по формуле: $I_d = I_t(e^{\frac{Ucb}{m\varphi T}} - 1)$.

Данные о диоде были взяты в программе ПА9:

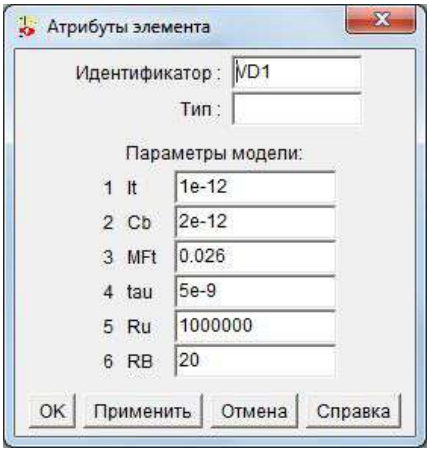

Рис. 3. Параметры диода в ПА9.

Схема с замененным диодом выглядит следующим образом:

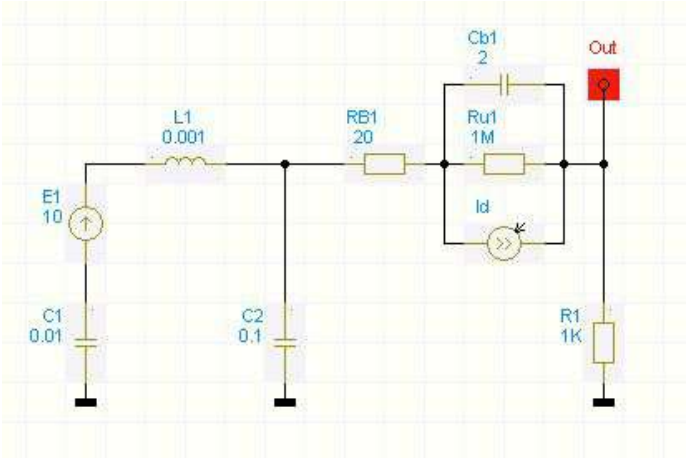

Рис. 4. Схема с заменой диода.

После того, как итоговая схема (рис. 4) получена, необходимо применить расширенный узловой метод формирования математической модели. В базис этого метода входят:

1) Производные переменных состояния: $\mathring{U}_{C1}$, $\mathring{U}_{C2}$, $\mathring{U}_{Cb}$, $\dot{I}_{L1}$.
2) Переменные состояния: U_{C1} ,U_{C2} ,U_{Cb} , I_{L1} .
3) Узловые потенциалы: φ_1 - φ_5 . Нумерация узлов и направления токов в схеме имеют следующий вид (искомый потенциал - φ_5):

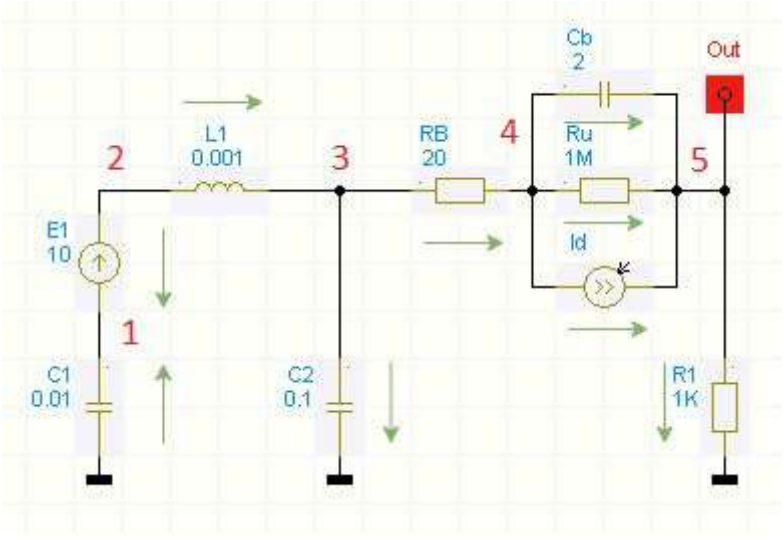

Рис. 5. Нумерация узлов и направления токов в схеме.

4) Токи идеальных источников ЭДС: I_{E1} .

Исходя из этого, вектор неизвестных состоит из 14 элементов и имеет следующий вид (транспонирован):

1	2	3	4	5	6	7	8	9	10	11	12	13	14
$\Delta\mathring{U}_{C1}$	$\Delta\mathring{U}_{C2}$	$\Delta\mathring{U}_{Cb}$	$\Delta\dot{I}_{L1}$	ΔU_{C1}	ΔU_{C2}	ΔU_{Cb}	ΔI_{L1}	$\Delta\varphi_1$	$\Delta\varphi_2$	$\Delta\varphi_3$	$\Delta\varphi_4$	$\Delta\varphi_5$	ΔI_{E1}

Вектор невязок составляется, исходя из компонентных уравнений элементов цепи, метода интегрирования и балансовых уравнений для токов в узлах. Он имеет следующий вид:

1	$\mathring{U}_{C1}$ - $\frac{U_{C1} - U_{C1}^{n-1}}{\Delta t}$	Из метода Эйлера
2	$\mathring{U}_{C2}$ - $\frac{U_{C2} - U_{C2}^{n-1}}{\Delta t}$	аналогично
3	$\mathring{U}_{Cb}$ - $\frac{U_{Cb} - U_{Cb}^{n-1}}{\Delta t}$	аналогично
4	$\dot{I}_{L1}$ - $\frac{I_{L1} - I_{L1}^{n-1}}{\Delta t}$	аналогично
5	$U_{C1} + \varphi_1$	т.к. $U_{C1} = 0 - \varphi_1$
6	$U_{C2} - \varphi_3$	т.к. $U_{C2} = \varphi_3 - 0$
7	$U_{Cb} - \varphi_4 + \varphi_5$	т.к. $U_{Cb} = \varphi_4 - \varphi_5$
8	$L_1 \cdot \dot{I}_{L1} - \varphi_2 + \varphi_3$	т.к. $U_{L1} = \varphi_2 - \varphi_3$ и из компонентного уравнения индуктивности: $U_L = L\dot{I}_L$
9	$-C_1\mathring{U}_{C1} - I_{E1}$	из I закона Кирхгофа и компонентного уравнения емкости: $I_C = C\mathring{U}_C$
10	$I_{E1} + I_{L1}$	аналогично
11	$-I_{L1} + C_2\mathring{U}_{C2} + \frac{(\varphi_3 - \varphi_4)}{R_B}$	аналогично и $I_R = U_R / R$
12	$-\frac{(\varphi_3 - \varphi_4)}{R_B} + C_b\mathring{U}_{Cb} + \frac{(\varphi_4 - \varphi_5)}{Ru} + I_t(e^{\frac{Ucb}{m\varphi T}} - 1)$	аналогично и $I_d = I_t(e^{\frac{Ucb}{m\varphi T}} - 1)$
13	$-C_b\mathring{U}_{Cb} - \frac{(\varphi_4 - \varphi_5)}{Ru} - I_t(e^{\frac{Ucb}{m\varphi T}} - 1) + \frac{\varphi_5}{R_1}$	аналогично
14	$E - \varphi_2 + \varphi_1$	т.к. $E = \varphi_2 - \varphi_1$

Матрица Якоби для метода Ньютона составляется посредством взятия частных производных. Она будет выглядеть так (нули опущены):

1				-1/Δt									
	1				-1/Δt								
		1				-1/Δt							
			1				-1/Δt						
				1				1					
					1					-1			
						1					-1	1	
			L_1						-1	1			
$-C_1$													-1
							1						1
	C_2						-1			$1/R_B$	$-1/R_B$		
		C_b								$-1/R_B$	$-1/R_B+1/R_u$	$-1/R_u$	
		$-C_b$				$\frac{I_t}{m\varphi T} e^{\frac{Ucb}{m\varphi T}}$					$-1/R_u$	$1/R_u+1/R_1$	
						$\frac{-I_t}{m\varphi T} e^{\frac{Ucb}{m\varphi T}}$		1	-1				

Алгоритм решения и динамический расчет шага

Алгоритм решения представлен на рисунке может быть описан с помощью следующей блок-схемы:

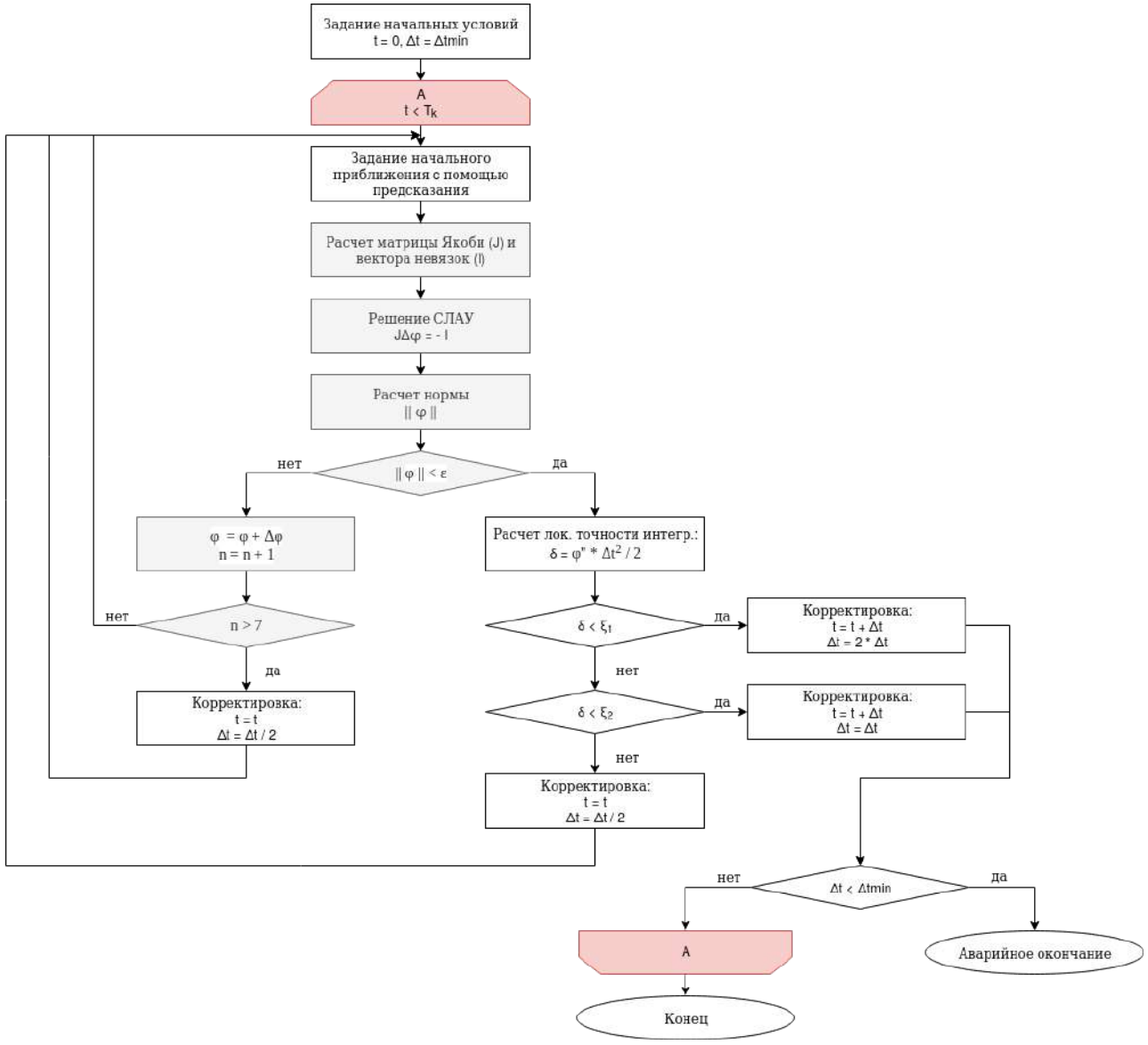

Рис. 6. Блок-схема программы.

На рисунке 6 светло-серым фоном отмечены блоки, относящиеся к итерациям метода Ньютона, красным - начало и конец общего цикла интегрирования.

Задание начального приближения для метода Ньютона выполняется в конце вычислений для текущего шага по времени с помощью линейной

аппроксимации на основании значений для предыдущего и текущего шага по следующей формуле:

$$\varphi_{\text{предск}} = [\varphi_i + (\varphi_i - \varphi_{i-1})] \frac{\Delta t}{\Delta t_{\text{прошл}}}$$

Локальная точность интегрирования определяется как:

$$\delta = \frac{d^2\varphi}{dt^2} \frac{\Delta t^2}{2}$$

Аппроксимируя эту формулу с помощью разностной схемы для второй производной, можно получить:

$$\delta = [\frac{\varphi_i - \varphi_{i-1}}{\Delta t} + \frac{\varphi_{i-1} - \varphi_{i-2}}{\Delta t_{\text{прошл}}}] \frac{\Delta t}{2}$$

На рисунке 6 под ξ_1 - подразумевается размер минимально допустимого шага, а под ξ_2 - максимально допустимого.

Описание и текст программы

Идея программной реализации алгоритма заключается в том, чтобы заранее задав константы, начальные условия и первое приближение, с помощью метода Ньютона найти решение на каждом временном шаге. Алгоритм программы почти полностью описан блок-схемой на рисунке 6. Кроме того, добавляются разгоночные шаги в начале программы.

В программе были использованы написанные ранее функция csv_parser и класс Matrix (включающий реализацию метода Гаусса).

Листинг файла main.cpp:

```
#include "csv_parser.hpp"
#include "matrix.hpp"
#include <cmath>

#define RESULT_FILENAME "res/result.dat"

#define NUMBER_CAUSE 0
#define NORM_CAUSE 1

#define GO_NEWTON   10
#define GO_FORWARD  11

// сложение для словарей
map<string, double> operator+ (map<string, double> a, map<string, double> b) {
    map<string, double> c = a;

    for (auto e : c) {
        c[e.first] += b[e.first];
    }

    return c;
}

// вычитание для словарей
map<string, double> operator- (map<string, double> a, map<string, double> b) {
    map<string, double> c = a;

    for (auto e : c) {
        c[e.first] -= b[e.first];
    }

    return c;
}

// деление словаря на число
map<string, double> operator/ (map<string, double> a, double b) {
```

```
        map<string, double> c = a;

        for (auto e : c) {
            c[e.first] /= b;
        }

        return c;
    }

    // создание вектора невязок
    Matrix<double>     create_residual_vector(map<string,     double>&     C,
map<string, double>& S, map<string, double>& pS, double dt) {
        Matrix<double> V(14, 1);

        V.set(0, 0, S["dUc1_dt"] - (S["Uc1"] - pS["Uc1"]) / dt);
        V.set(1, 0, S["dUc2_dt"] - (S["Uc2"] - pS["Uc2"]) / dt);
        V.set(2, 0, S["dUcb_dt"] - (S["Ucb"] - pS["Ucb"]) / dt);
        V.set(3, 0, S["dIl1_dt"] - (S["Il1"] - pS["Il1"]) / dt);
        V.set(4, 0, S["Uc1"] + S["phi_1"]);
        V.set(5, 0, S["Uc2"] - S["phi_3"]);
        V.set(6, 0, S["Ucb"] - S["phi_4"] + S["phi_5"]);
        V.set(7, 0, C["L1"] * S["dIl1_dt"] - S["phi_2"] + S["phi_3"]);
        V.set(8, 0, -C["C1"] * S["dUc1_dt"] - S["Ie1"]);
        V.set(9, 0, S["Ie1"] + S["Il1"]);
          V.set(10,  0,  -S["Il1"]  +  C["C2"]  *  S["dUc2_dt"]  +  (S["phi_3"]  -
S["phi_4"]) / C["Rb"]);
        V.set(11, 0, -(S["phi_3"] - S["phi_4"]) / C["Rb"] +
                     C["Cb"] * S["dUcb_dt"] +
                     (S["phi_4"] - S["phi_5"]) / C["Ru"] +
                     C["It"] * (exp(S["Ucb"] / C["MFt"]) - 1)
        );
        V.set(12, 0, -C["Cb"] * S["dUcb_dt"] +
                     -(S["phi_4"] - S["phi_5"]) / C["Ru"] +
                     -C["It"] * (exp(S["Ucb"] / C["MFt"]) - 1) +
                     S["phi_5"] / C["R1"]
        );
        V.set(13, 0, C["E"] - S["phi_2"] + S["phi_1"]);

        return V;
    }

    // создание матрицы Якоби
    Matrix<double> create_Jacobi_matrix(map<string, double>& C, map<string,
double>& S, double dt) {
        Matrix<double> J(14, 14);

        // единицы
        for (int i = 0; i < 7; i++) {
            J.set(i, i, 1);
        }
        J.set(4, 8, 1);
        J.set(6, 12, 1);
        J.set(7, 10, 1);
        J.set(9, 7, 1);
        J.set(9, 13, 1);
        J.set(13, 8, 1);
```

```
        // - единицы
        J.set(5, 10, -1);
        J.set(6, 11, -1);
        J.set(7, 9, -1);
        J.set(8, 13, -1);
        J.set(10, 7, -1);
        J.set(13, 9, -1);


        // -1 / dt
        for (int i = 0; i < 4; i++) {
            J.set(i, i + 4, -1 / dt);
        }


        // R
        J.set(10, 10, 1 / C["Rb"]);
        J.set(10, 11, -1 / C["Rb"]);
        J.set(11, 10, -1 / C["Rb"]);
        J.set(11, 11, 1 / C["Rb"] + 1 / C["Ru"]);
        J.set(11, 12, -1 / C["Ru"]);
        J.set(12, 11, -1 / C["Ru"]);
        J.set(12, 12, 1 / C["Ru"] + 1 / C["R1"]);


        // Diod
        const double dId_dt = C["It"] * exp(S["Ucb"] / C["MFt"]) / C["MFt"];
        J.set(11, 6, dId_dt);
        J.set(12, 6, -dId_dt);


        // L & C
        J.set(7, 3, C["L1"]);
        J.set(8, 0, -C["C1"]);
        J.set(10, 1, C["C2"]);
        J.set(11, 2, C["Cb"]);
        J.set(12, 2, -C["Cb"]);

        return J;
    }

    // расчет равномерной нормы вектора
    double Inf_norm(Matrix<double>& v) {
        if (v.get_cols() != 1) {
             cout << "ОШИБКА: вектор имеет более одного столбца, а именно: "
<< v.get_cols() << endl;
            return -1;
        }

        double max_el = 0;
        for (int i = 0; i < v.get_rows(); i++) {
            if(v.get(i, 0) > max_el) {
                max_el = v.get(i, 0);
            }
        }
```

```
        return max_el;
    }

    // расчет равномерной нормы для словаря состояния
    double Inf_norm(map<string, double>& v) {
        double max_el = 0;
        for (auto el : v) {
            if (el.first[0] != 'd') {
                if (el.second > max_el) {
                    max_el = el.second;
                };
            }
        }


        return max_el;
    }

    // добавляет рассчитанные приращения к состоянию
    map<string, double> append_d_to_state(map<string, double>& S,
Matrix<double>& d) {
        map<string, double> S_new = S;

        S_new["dUc1_dt"] += d.get(0, 0);
        S_new["dUc2_dt"] += d.get(1, 0);
        S_new["dUcb_dt"] += d.get(2, 0);
        S_new["dIl1_dt"] += d.get(3, 0);
        S_new["Uc1"] += d.get(4, 0);
        S_new["Uc2"] += d.get(5, 0);
        S_new["Ucb"] += d.get(6, 0);
        S_new["Il1"] += d.get(7, 0);
        S_new["phi_1"] += d.get(8, 0);
        S_new["phi_2"] += d.get(9, 0);
        S_new["phi_3"] += d.get(10, 0);
        S_new["phi_4"] += d.get(11, 0);
        S_new["phi_5"] += d.get(12, 0);
        S_new["Ie1"] += d.get(13, 0);

        return S_new;
    }

    // расчет локальной точности
    double calc_local_acc(
        map<string, double>& ppS,
        map<string, double>& pS,
        map<string, double>& S,
        double pdt,
        double dt
    ) {
        // создание матрицы с теми же полями,
        // обнуляем значения; погрешности для каждого элемента
        map<string, double> acr = S;
        for (auto e : acr) {
            acr[e.first] = 0;
        }

        // расчет погрешностей
```

```
    for (auto e : S) {
        if (e.first[0] == 'd') {
            continue;
        }

        string key = e.first;

        acr[key] = (S[key] - pS[key]) / dt;
        acr[key] += (pS[key] - ppS[key]) / pdt;
        acr[key] /= dt;

        acr[key] *= dt * dt / 2;
    }


    // выбор максимальной погрешности
    return Inf_norm(acr);
}

// реализация метода Ньютона
map<string, double> Newton(
    map<string, double>& C,
    map<string, double>& pS,
    map<string, double>& S,
    double dt,
    bool* stop_cause
) {
    double norm = C["eps"] + 1;
    map<string, double> S_new = S;
    int n = 0;      // счетчик на случай зацикливания метода

    while (norm > C["eps"] && n <= C["max_steps"]) {
        n++;

        Matrix<double> J = create_Jacobi_matrix(C, S_new, dt);
        Matrix<double> R = create_residual_vector(C, S_new, pS, dt);
        R.mul_num(-1);

        J.Gauss(&R);

        norm = Inf_norm(R);

        S_new = append_d_to_state(S_new, R);
    };

    if (norm <= C["eps"]) {
        *stop_cause = NORM_CAUSE;
    } else {
        *stop_cause = NUMBER_CAUSE;
    }

    return S_new;
}


int main() {
    // задание начального состояния
```

```
        map<string, double> C = create_map_from_file("input/constants.dat");
                                map<string,         double>         S         =
create_map_from_file("input/initial_state.dat");
          map<string, double> ppS = S, pS = S, nS; // предпред-, пред- и
следующий шаги

        double t = 0, dt = C["dt_start"], pdt = dt;

        bool stop_cause = NORM_CAUSE;


        // открытие файла и запись в этот файл начального состояния
        std::ofstream result_file;
        result_file.open(RESULT_FILENAME);
        if (!result_file) {
             cout << "ОШИБКА: Невозможно открыть файл для записи результатов:
" << RESULT_FILENAME << endl;
            return -1;
        }


        // расчет на каждом временном шаге
        int step_num = 0;   // заход в цикл
        int where_to_go = GO_NEWTON;
        while (t <= C["T"]) {
            // записываем в файл только на нулевой итерации
            // или когда переходим к следующему шагу по времени
            result_file << t << "\t" << nS["phi_5"] << endl;


            // на разгонных шагах цикла выполняем только
            // метод Ньютона
            if (step_num < C["racing_steps"]) {

                do {
                      // метод Ньютона, пока итераций не слишком много, либо
до нормы
                    nS = Newton(C, pS, S, dt, &stop_cause);

                     // выясняем, почему метод Ньютона остановился и изменяем
шаг и время в зависимости от этого
                    if (stop_cause == NUMBER_CAUSE) {
                        pdt = dt;
                        dt /= 2;
                        // предсказание метода Ньютона
                        S = nS + (nS - pS) / (pdt / dt);
                    }
                } while (stop_cause != NORM_CAUSE);

                // предсказание метода Ньютона
                S = nS + (nS - pS) / (pdt / dt);

                // переходим к следующему шагу по времени
                ppS = pS;
                pS = nS;
                t += dt;
                step_num++;
```

```
            } else {

                // когда разгонные шаги прекращаются, начинаем анализировать
                // длину шага по времени и менять ее
                do {
                      // метод Ньютона, пока итераций не слишком много, либо
до нормы
                    nS = Newton(C, pS, S, dt, &stop_cause);

                    pdt = dt;
                     // выясняем, почему метод Ньютона остановился и изменяем
шаг и время в зависимости от этого
                    if (stop_cause == NORM_CAUSE) {
                        double acr = calc_local_acc(ppS, pS, nS, pdt, dt);

                        if (acr < C["xi_1"]) {
                            t += dt;
                            dt *= 2;
                            where_to_go = GO_FORWARD;
                        } else if (acr < C["xi_2"]) {
                            t += dt;
                            where_to_go = GO_FORWARD;
                        } else {
                            dt /= 2;
                            // предсказание метода Ньютона
                            S = nS + (nS - pS) / (pdt / dt);
                        }
                    } else if (stop_cause == NUMBER_CAUSE) {
                        dt /= 2;
                    }
                } while (where_to_go != GO_FORWARD);


                // Проверка: не стал ли шаг слишком маленьким
                if (dt < C["dt_min"]) {
                        cout << "ОШИБКА: Шаг сократился слишком сильно: " <<
endl;
                    cout << "dt = " << dt << endl;
                    cout << "dt_min = " << C["dt_min"] << endl;
                    return -1;
                }

                // предсказание метода Ньютона
                S = nS + (nS - pS) / (pdt / dt);

                // переход к следующему шагу
                ppS = pS;
                pS = nS;
                step_num++;
            }
        }

        result_file.close();

        return 0;
    }
```

Листинг скрипта для отрисовки графика результата plot.gnu:

```
set terminal png
set output 'res/result.png'

# Set linestyle 1 to blue (#0060ad)
set style line 1 \
    linecolor rgb '#0060ad' \
    linetype 1 linewidth 2 \
    pointtype 7 pointsize 0.1

plot 'res/result.dat' with linespoints linestyle 1
```

Исходные данные программы в файле constants.dat:

```
C1 = 0.01e-6
C2 = 0.1e-6
E = 10
L1 = 0.001
R1 = 1000
It = 1e-12
Cb = 2e-12
MFt = 0.026
Ru = 1e6
Rb = 20

T = 1e-3
dt_start = 1e-15
dt_min = 1e-15

eps = 1e-6
max_steps = 7

xi_1 = 1e-2
xi_2 = 2e-2

racing_steps = 3
```

Результаты работы программы

В результате работы программы был получен файл *result.dat*, который был визуализирован с помощью *gnuplot.* В результате был получен следующий график:

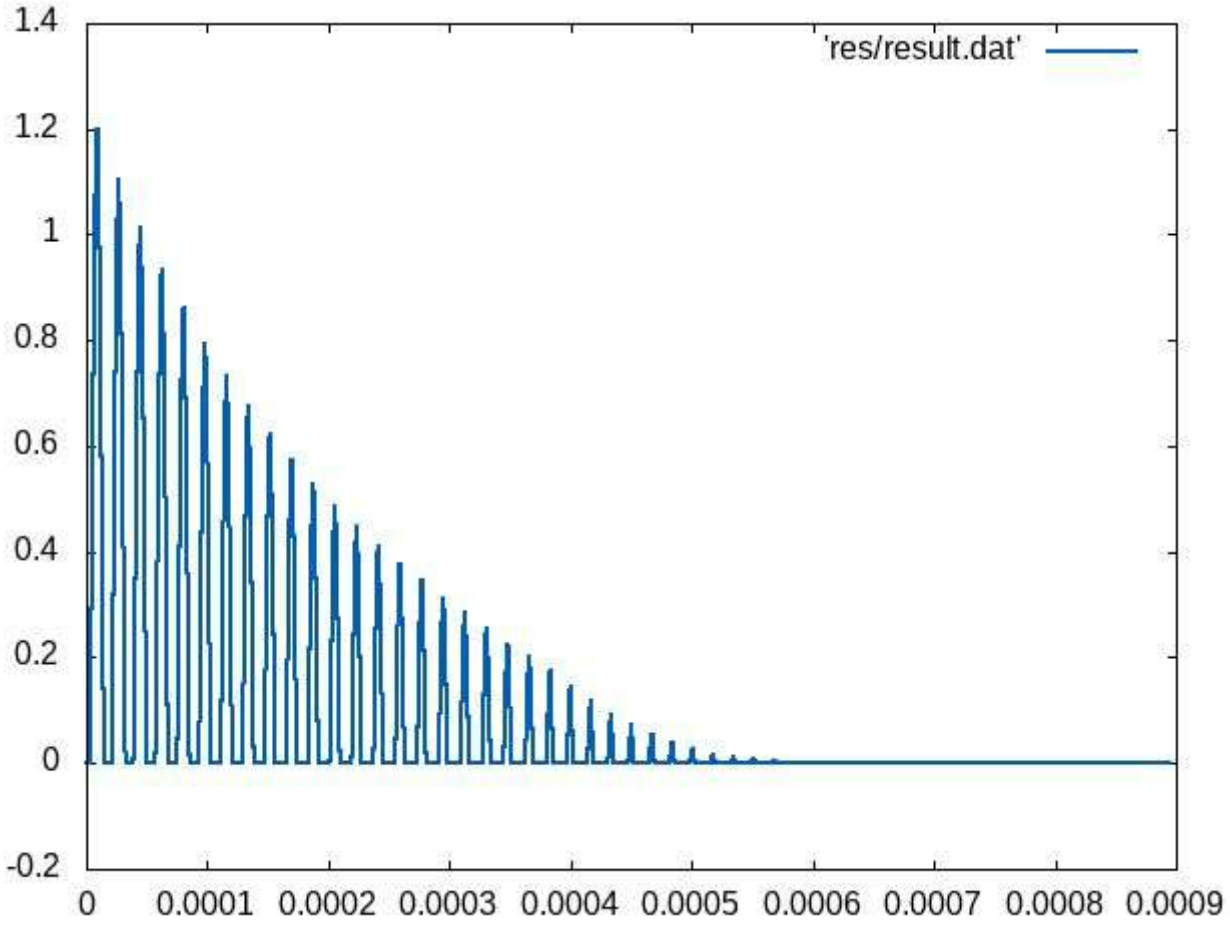

Рис.7. График потенциала в точке 5 в зависимости от времени.

Сравнение решения с решением в программе ПА9

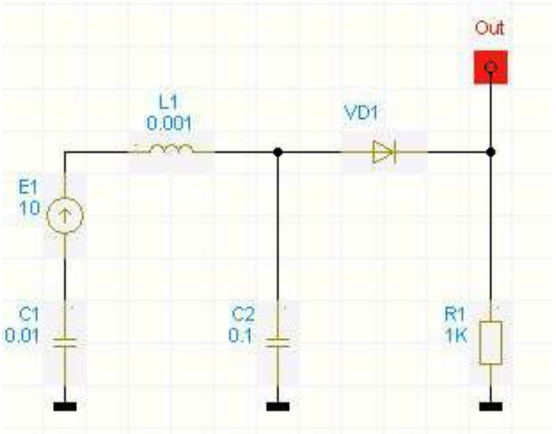

Рис.8. Схема, реализованная в ПА9.

В программе ПА9 была реализована схема, показанная на рисунке 8.

В результате динамического анализа был получен график потенциала φ_5 во времени до момента T = 1e-3 с, аналогичный полученному в своей программе:

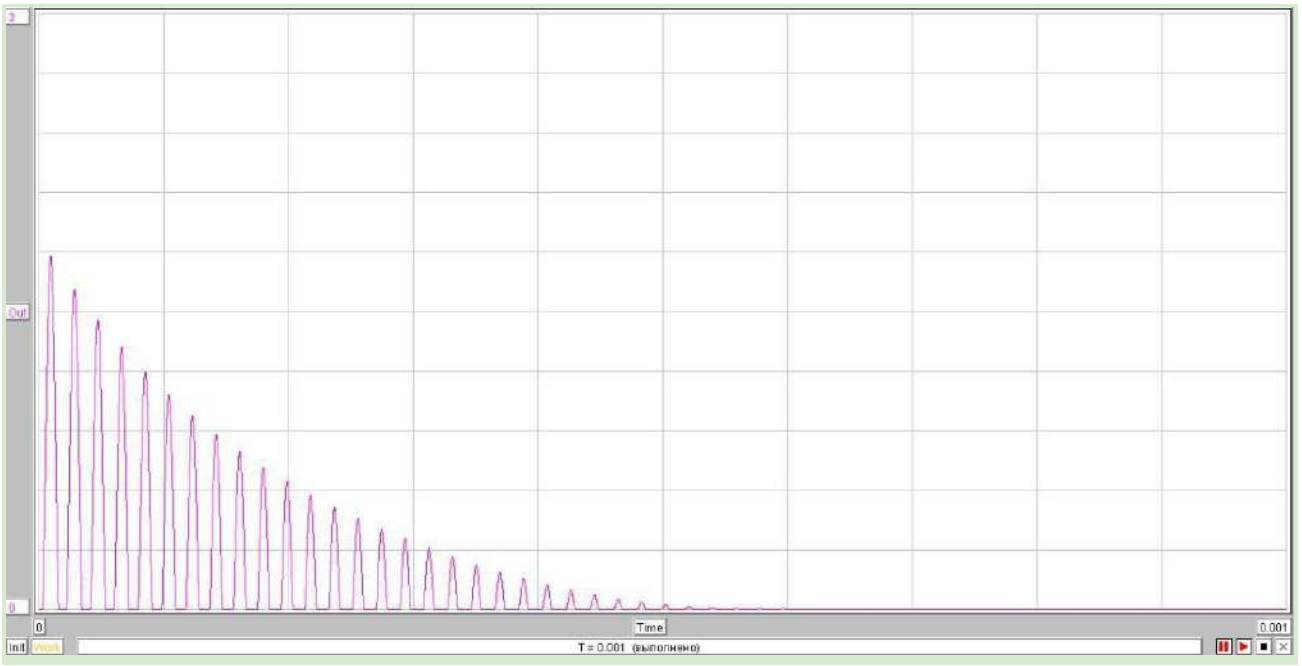

Рис.8. График потенциала в точке 5 в зависимости от времени, полученный в ПА9.

Список использованных источников

1. Трудоношин В.А. Лекции по курсу “Модели и методы проектных решений”.
2. А.О. Ильницкий, В.Б.Маничев, М.Ю.Уваров, В.А. Трудоношин “Моделирование динамики технических систем с использованием программного комплекса ПА9 ” - Методические указания к циклу лабораторных работ по курсу “Основы автоматизированного проектирования”, Москва, 2004