ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

Федеральное государственное образовательное бюджетное учреждение

высшего профессионального образования

«Санкт-Петербургский государственный университет телекоммуникаций

им. проф. М.А. Бонч-Бруевича»

Факультет Информационных технологий и программной инженерии

Кафедра Программной инженерии и вычислительной техники

КУРСОВАЯ РАБОТА

по дисциплине:

**«Программирование»**

тема: Анализ сигнала на выходе электрической цепи

Передаточная характеристика – 2 вариант

Входной сигнал – 14 вариант

Выполнил студент(ка):

Костров Д. Э. ИКПИ-44 \_\_\_\_\_\_\_\_\_\_\_\_\_

*(Ф.И.О., № группы) (подпись)*

Дата выполнения:

«\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025г

Проверил:

Дятлов Д. А. \_\_\_\_\_\_\_\_\_\_\_\_\_

*(Ф.И.О.) (подпись)*

Санкт-Петербург

2025

## Аннотация

Программа анализирует сигналы в электрической цепи, вычисляя выходное напряжение по заданному алгоритму. Разработана на языке C с использованием математических функций. Позволяет проводить расчеты для произвольного количества точек, сохранять результаты в файлы и визуализировать данные. Результаты верифицированы в среде WxMaxima.

Оглавление

[Аннотация 2](#_Toc197811897)

[Задание 4](#_Toc197811898)

[Таблица идентификаторов 5](#_Toc197811899)

[Блок схемы 6](#_Toc197811900)

[Текст программы 18](#_Toc197811901)

[Код на C 18](#_Toc197811902)

[wxMaxima код 27](#_Toc197811903)

[Bash-скрипт 28](#_Toc197811904)

[Описание программы 29](#_Toc197811905)

[Общие сведения 29](#_Toc197811906)

[Функциональное назначение 29](#_Toc197811907)

[Логическая структура 30](#_Toc197811908)

[Используемые технические средства 30](#_Toc197811909)

[Вызов и загрузка 30](#_Toc197811910)

[Выходные данные 30](#_Toc197811911)

[Графики 32](#_Toc197811912)

[Интерфейс 34](#_Toc197811913)

[Заключение 36](#_Toc197811914)

[Список литературы 37](#_Toc197811915)

## Задание

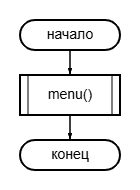
В курсовой работе необходимо для заданной электрической цепи по известному входному сигналу определить выходной сигнал для N равностоящих моментов времени, а затем определить некоторые его характеристики с погрешностью не более 1%.   
В ходе работы студентам необходимо:

* Произвести расчет входного и выходного сигнала в контрольных точках, используя при этом математический пакет Wxmaxima;
* Написать текст программы на языке Си;
* Произвести запись полученных результатов в файлы данных;
* Используя математический пакет Wxmaxima или LibraOffice.Calc (электронные таблицы), построить графики зависимости напряжений входных и выходных сигналов от времени.
* Оформить пояснительную записку (doc-файл) по ГОСТ 19.402-78. ЕСПД. Описание программы. Плюс «Заключение» с личными выводами по работе.
* Объединить программу на Си и Wxmaxima (LibraOffice.Calc), вызов отчета с помощью скрипта на Bash.
* Защитить работу преподавателю.

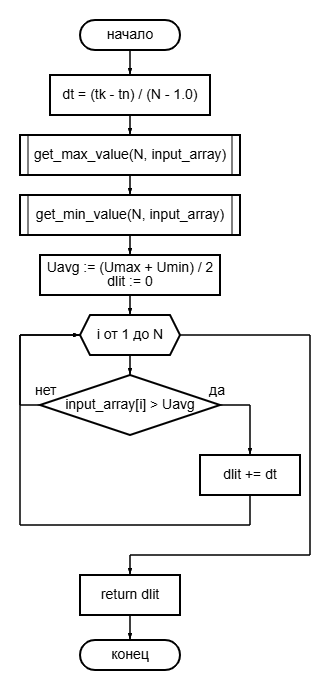
## Таблица идентификаторов

| **Название** | **Тип** | **Назначение** |
| --- | --- | --- |
| N | int | Количество точек для расчётов. |
| Uvx\_array | double[] | Массив значений входного напряжения. |
| Uvix\_array | double[] | Массив значений выходного напряжения. |
| time\_array | double[] | Массив значений времени. |
| parametr | double | Вычисленное значение параметра. |
| real\_parametr | double | Истинное значение параметра. |
| error | double | Ошибка в процентах между parametr и real\_parametr. |
| choice | int | Выбор пользователя в меню. |
| stop | char | Флаг для завершения программы. |
| write | int | Флаг для записи данных в файл (1) или вывода на экран (0). |
| error\_rate | double | Текущая ошибка в итеративном расчёте. |
| eps | double | Порог точности для итеративного расчёта. |
| par | double | Временное значение параметра в итеративном расчёте. |
| border | int | Индекс точки, где время превышает 45. |
| i | int | Счётчик в циклах. |
| TN, TK, T1, T2 | double | Константы времени для расчёта напряжений. |
| A, B, C, A1, B1 | double | Константы для расчёта напряжений. |

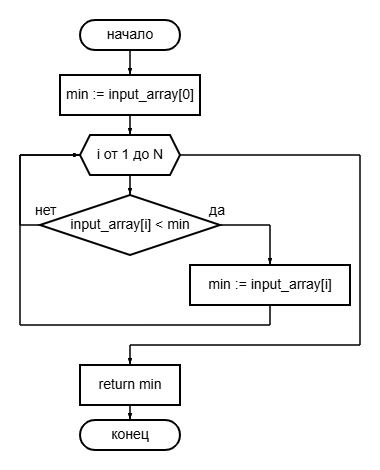
## Блок схемы



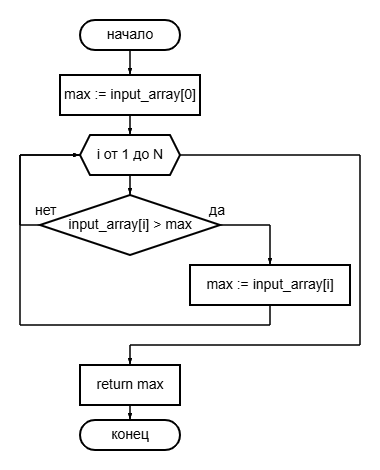
Схема



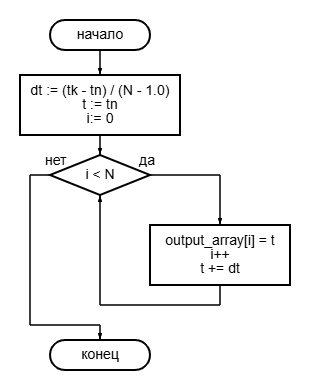
Схема



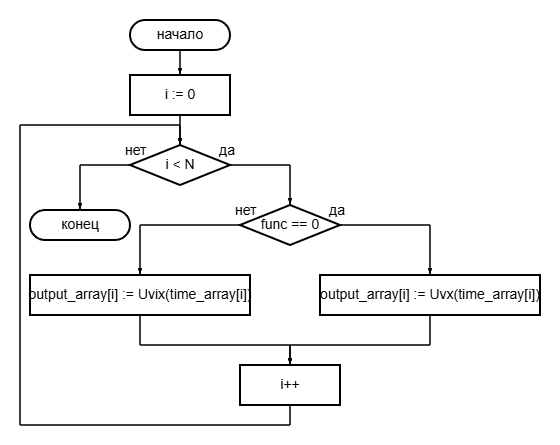
Схема



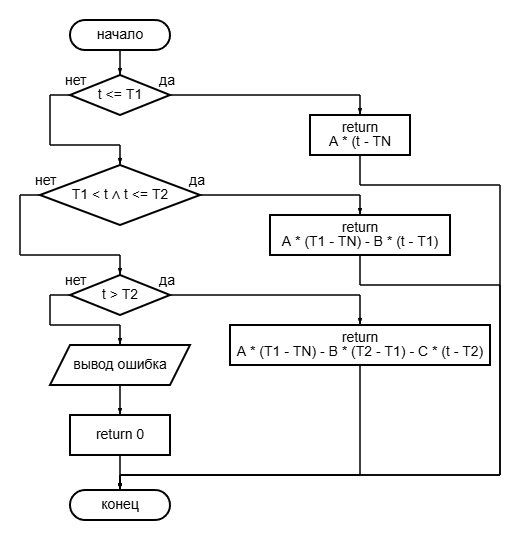
Схема



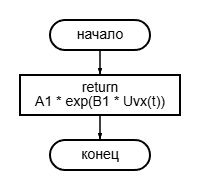
Схема



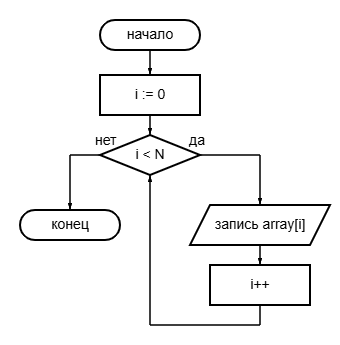
Схема



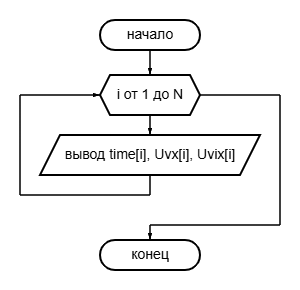
Схема



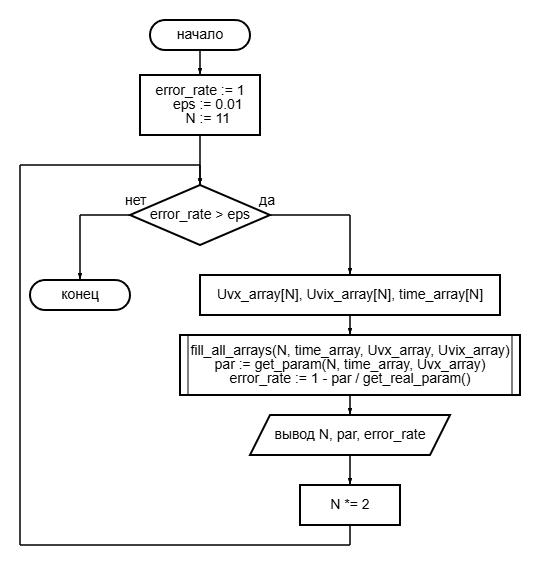
Схема



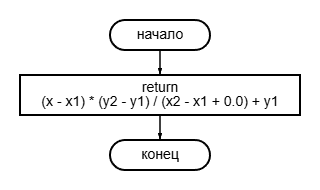
Схема



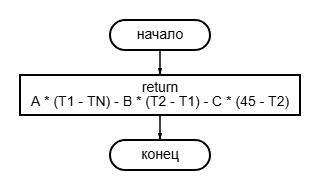
Схема



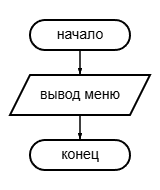
Схема



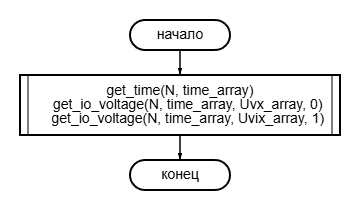
Схема



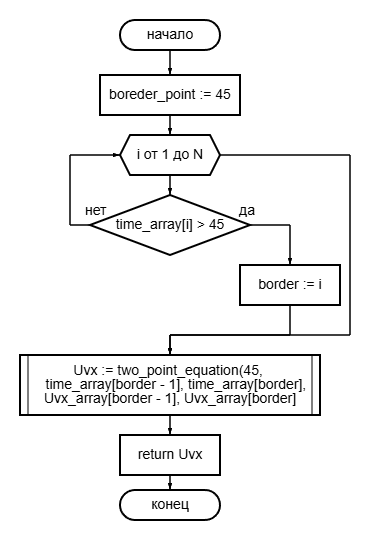
Схема



Схема



Схема



Схема

Схема 1 - main()

Схема 2 - print\_parametr()

Схема 3 - print\_write\_voltages(int write)

Схема 4 - menu()

Схема 5 - print\_arrays(int N, double \*time, double \*Uvx, double \*Uvix)

Схема 6 - save\_array(int N, double \*array, char \*name)

Схема 7 - Uvx(double t)

Схема 8 - Uvix(double t)

Схема 9 - get\_io\_voltage(int N, double \*time\_array, double \*output\_array, int func)

Схема 10 - get\_time(int N, double \*output\_array)

Схема 11 - print\_param\_with\_an\_error()

Схема 12 - two\_point\_equation(double x, double x1, double x2, double y1, double y2)

Схема 13 - get\_real\_param()

Схема 14 - print\_menu()

Схема 15 - fill\_all\_arrays(int N, double time\_array[], double Uvx\_array[], double Uvix\_array[])

Схема 16 - get\_param(int N, double time\_array[], double Uvx\_array[])

## Текст программы

### Код на C

##### main.c

#include <stdio.h>

#include <stdlib.h>

#include "menu.h"

int main()

{

menu();

return 0;

}

##### menu.c

##### #include <stdlib.h>

##### #include <stdio.h>

##### #include "io.h"

##### #include "logic.h"

##### #ifdef \_WIN32

##### #define CLEAR\_SCREEN() system("cls")

##### #else

##### #define CLEAR\_SCREEN() system("clear")

##### #endif

##### void menu()

##### {

##### int choice;

##### char stop;

##### while (1 == 1)

##### {

##### print\_menu();

##### scanf("%d", &choice);

##### switch(choice)

##### {

##### case 1:

##### print\_write\_voltages(0);

##### break;

##### case 2:

##### print\_parametr();

##### break;

##### case 3:

##### print\_param\_with\_an\_error();

##### break;

##### case 4:

##### print\_write\_voltages(1);

##### break;

##### case 5:

##### return;

##### break;

##### default:

##### CLEAR\_SCREEN();

##### continue;

##### break;

##### }

##### printf("\nTerminate?[Y/N] ");

##### getchar();

##### stop = getchar();

##### if (stop == 'y' || stop == 'Y')

##### {

##### return;

##### }

##### CLEAR\_SCREEN();

##### }

##### }

##### io.c

##### #include <stdlib.h>

##### #include <stdio.h>

##### #include "logic.h"

##### void save\_array(int N, double \*array, char \*name)

##### {

##### FILE \*file = fopen(name, "w");

##### if (!file)

##### {

##### perror("File error");

##### return;

##### }

##### int i = 0;

##### while (i < N)

##### {

##### fprintf(file, "%.15lf\n", array[i]);

##### i++;

##### }

##### fclose(file);

##### }

##### int input\_nuber\_points()

##### {

##### int N;

##### printf("Enter number of points: ");

##### scanf("%d", &N);

##### return N;

##### }

##### void print\_parametr()

##### {

##### int N = input\_nuber\_points();

##### double Uvx\_array[N], Uvix\_array[N], time\_array[N];

##### fill\_all\_arrays(N, time\_array, Uvx\_array, Uvix\_array);

##### double parametr = get\_param(N, time\_array, Uvx\_array);

##### double real\_parametr = get\_real\_param();

##### double error = (1 - parametr / real\_parametr) \* 100;

##### printf("Parameter value: %4.2f\tError: %3.3f%%",

##### parametr, error);

##### }

##### void print\_arrays(int N, double \*time, double \*Uvx, double \*Uvix) {

##### printf("+------+--------+----------+--------------+\n");

##### printf("| No | time | Uvx | Uvix |\n");

##### printf("+------+--------+----------+--------------+\n");

##### for (int i = 0; i < N; i++) {

##### printf("| %2d | %4.1f | %6.2f | %10.4e |\n", i + 1, time[i], Uvx[i], Uvix[i]);

##### }

##### printf("+------+--------+----------+--------------+\n");

##### }

##### void print\_write\_voltages(int write)

##### {

##### int N = input\_nuber\_points();

##### double Uvx\_array[N], Uvix\_array[N], time\_array[N];

##### fill\_all\_arrays(N, time\_array, Uvx\_array, Uvix\_array);

##### if (write == 0)

##### {

##### print\_arrays(N, time\_array, Uvx\_array, Uvix\_array);

##### }

##### else

##### {

##### save\_array(N, time\_array, "data/time\_array.txt");

##### save\_array(N, Uvx\_array, "data/Uvx\_array.txt");

##### save\_array(N, Uvix\_array, "data/Uvix\_array.txt");

##### printf("Done");

##### }

##### }

##### void print\_menu()

##### {

##### printf("1 - Control calculation for N points\n");

##### printf("2 - Calculation of a parameter with N points\n");

##### printf("3 - Calculation of a parameter with a given accuracy\n");

##### printf("4 - Writing data to a file\n");

##### printf("5 - Exit\n");

##### }

##### logic.c

#include <math.h>

#include <stdlib.h>

#include <stdio.h>

#define TN 5.0

#define TK 50.0

#define T1 15.0

#define T2 45.0

#define A 20.0

#define B 0.5

#define C 17.0

#define A1 5.0

#define B1 0.5

double Uvx(double t)

{

if (t <= T1)

{

return A \* (t - TN);

}

if (T1 < t && t <= T2)

{

return A \* (T1 - TN) - B \* (t - T1);

}

if (t > T2)

{

return A \* (T1 - TN) - B \* (T2 - T1) - C \* (t - T2);

}

printf("\tt = %F\tSomething went wrong...\n", t);

return 0;

}

double Uvix(double t)

{

return A1 \* exp(B1 \* Uvx(t));

}

void get\_io\_voltage(int N, double \*time\_array, double \*output\_array, int func)

{

int i = 0;

while (i < N)

{

if (func == 0)

{

output\_array[i] = Uvx(time\_array[i]);

} else

{

output\_array[i] = Uvix(time\_array[i]);

}

i++;

}

}

void get\_time(int N, double \*output\_array)

{

double dt = (TK - TN) / (N - 1.0);

double t = TN;

int i = 0;

while (i < N)

{

output\_array[i] = t;

i++;

t += dt;

}

}

void fill\_all\_arrays(int N, double time\_array[], double Uvx\_array[], double Uvix\_array[])

{

get\_time(N, time\_array);

get\_io\_voltage(N, time\_array, Uvx\_array, 0);

get\_io\_voltage(N, time\_array, Uvix\_array, 1);

}

double two\_point\_equation(double x, double x1, double x2, double y1, double y2)

{

return (x - x1) \* (y2 - y1) / (x2 - x1 + 0.0) + y1;

}

double get\_real\_param()

{

return A \* (T1 - TN) - B \* (T2 - T1) - C \* (45 - T2);

}

double get\_param(int N, double time\_array[], double Uvx\_array[])

{

const int boreder\_point = 45;

int border;

for (int i = 0; i < N; i++)

{

if (time\_array[i] > 45)

{

border = i;

break;

}

}

double Uvx = two\_point\_equation(45,

time\_array[border - 1], time\_array[border],

Uvx\_array[border - 1], Uvx\_array[border]);

return Uvx;

}

void print\_param\_with\_an\_error()

{

double error\_rate = 1;

double eps = 0.01;

double par;

int N = 11;;

while (error\_rate > eps)

{

double Uvx\_array[N], Uvix\_array[N], time\_array[N];

fill\_all\_arrays(N, time\_array, Uvx\_array, Uvix\_array);

par = get\_param(N, time\_array, Uvx\_array);

error\_rate = 1 - par / get\_real\_param();

printf("N = %d\tParam = %.7f\tError = %.7f\n", N, par, error\_rate);

N \*= 2;

}

}

##### menu.h

#ifndef MENU\_H\_INCLUDED

#define MENU\_H\_INCLUDED

void menu();

#endif // MENU\_H\_INCLUDED

##### io.h

#ifndef IO\_H\_INCLUDED

#define IO\_H\_INCLUDED

void print\_arrays(int N, double \*time, double \*Uvx, double \*Uvix);

void save\_array(int N, double \*array, char \*name);

int input\_nuber\_points();

void print\_parametr();

void print\_write\_voltages(int write);

void print\_menu();

#endif // IO\_H\_INCLUDED

##### logic.h

#ifndef LOGIC\_H\_INCLUDED

#define LOGIC\_H\_INCLUDED

double Uvx(double t);

double Uvix(double t);

void get\_io\_voltage(int N, double \*time\_array, double \*output\_array, int func);

void get\_time(int N, double \*output\_array);

void fill\_all\_arrays(int N, double time\_array[], double Uvx\_array[], double Uvix\_array[]);

double get\_param();

double get\_real\_param();

void print\_param\_with\_an\_error();

#endif // LOGIC\_H\_INCLUDED

### wxMaxima код

fpprintprec:5$;

U: 5$; T: 0.00001$;

tn: 0$;

tk: T$;

U0: 5$; U1: 10$; U2: 75$; U3: 90$;

Uvx0: 0$; Uvx1: 10$; Uvx2: 40$; Uvx3: 10$;

N: 50$; dt: (tk - tn) / (N - 1)$;

Uvx(t):=

if (0 <= t and t < T / 4)

then 0

else if (T/4 <= t and t < 3 \* T / 4)

then U

else 0

$;

Uvix(t):=

if Uvx(t) <= Uvx1

then (Uvx(t) - Uvx0) \* (U1 - U0) / (Uvx1 - Uvx0) + U0

else if Uvx1 < Uvx(t) and Uvx(t) <= Uvx2

then (Uvx(t) - Uvx1) \* (U2 - U1) / (Uvx2 - Uvx1) + U1

else (Uvx(t) - Uvx2) \* (U3 - U2) / (Uvx3 - Uvx2) + U2

$;

for i: 0 thru N-1 do (

t: float(tn + i \* dt),

display(Uvx(t), Uvix(t)));

data\_t: read\_list("data/time\_array.txt");

data\_Uvx: read\_list("data/Uvx\_array.txt");

data\_Uvix: read\_list("data/Uvix\_array.txt");

wxplot2d([Uvx(time), [discrete, data\_t, data\_Uvx]], [time, tn, tk],

[gnuplot\_preamble, "set grid;"],

[legend, "wxMaxima", "C"],

[title, "График функции Uvx(t)"])$;

wxplot2d([Uvix(time), [discrete, data\_t, data\_Uvix]], [time, tn, tk],

[gnuplot\_preamble, "set grid;"],

[legend, "wxMaxima", "C"],

[title, "График функции Uvix(t)"])$;

### Bash-скрипт

#!/bin/bash

cat zastavka.txt;

./c.exe;

"D:\maxima-5.47.0\bin\wxmaxima.exe" --eval "kpm.wxmx";

start "" "Пояснительная записка.docx";

## Описание программы

### Общие сведения

Программа предназначена для анализа сигналов в электрической цепи, вычисления выходного напряжения и визуализации результатов.

##### Исполняемые файлы:

- `c.exe` – скомпилированная программа на языке C (основной модуль).

- `bash.sh`– скрипт для автоматизации запуска.

##### Используемые языки и инструменты:

- C – основной язык разработки (расчёты, формирование массивов данных).

- Bash – автоматизация вызова программы и обработки данных.

- WxMaxima – математический пакет для проверки расчётов и построения графиков.

### Функциональное назначение

Программа выполняет следующие задачи:

1. Интерактивное меню – выбор режима работы (расчёт, сохранение данных, построение графиков).

2. Вычисление сигналов – расчёт входного и выходного напряжений в заданных временных точках.

3. Сохранение данных – запись результатов в текстовые файлы (`massiv\_t.txt`, `massiv\_Uvx.txt`, `massiv\_Uvix.txt`).

4. Визуализация – автоматическое построение графиков в WxMaxima.

##### Алгоритм работы:

- Пользователь запускает `run.sh` (или `run.bat`).

- Программа `c.exe` выполняет расчёты и сохраняет данные.

- WxMaxima автоматически строит графики на основе сохранённых файлов.

- Открывается пояснительная записка (`Пояснительная\_записка.doc`).

### Логическая структура

##### Запуск скрипта:

- `bash.sh` вызывает `c.exe`.

- Программа выполняет расчёты и записывает данные в файлы.

##### Обработка результатов:

- Скрипт запускает WxMaxima для построения графиков.

- После завершения работы WxMaxima открывается пояснительная записка.

### Используемые технические средства

- Git Bash / WSL – для работы скриптов в Windows.

- WxMaxima – для математических расчётов и визуализации.

- Любой текстовый редактор – для просмотра выходных файлов.

### Вызов и загрузка

##### Требования:

- Установленные: WxMaxima, Bash

##### Запуск:

- Перейти в директорию с программой.

- Выполнить команду:

./bash.sh

### Выходные данные

После выполнения программы автоматически создаются файлы:

- `massiv\_t.txt` – массив временных точек.

- `massiv\_Uvx.txt` – массив входных напряжений.

- `massiv\_Uvix.txt` – массив выходных напряжений.

Эти файлы используются для построения графиков в WxMaxima или анализа в других программах.

## Графики

|  |  |  |
| --- | --- | --- |
|  | Средствами C | Средствами wxMaxima |
| Uvx(t) |  |  |
| Uvix(t) |  |  |

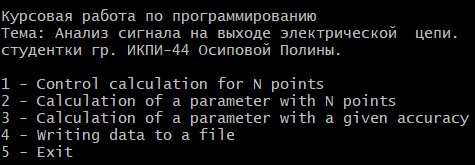
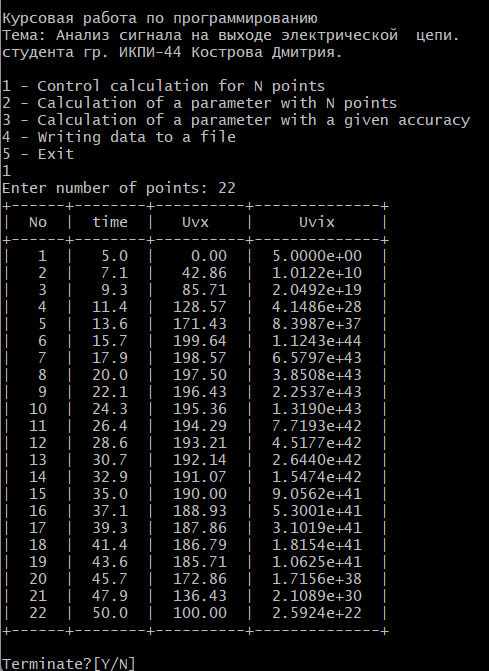
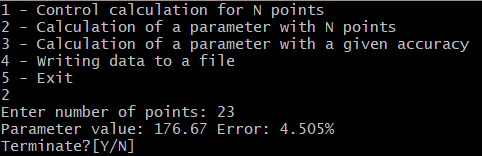
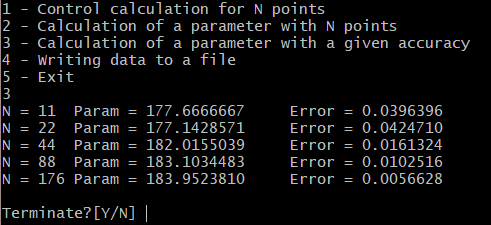
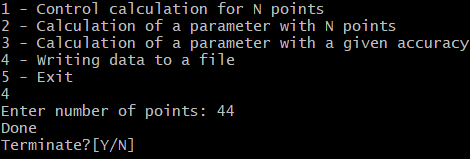
На основании проведенного анализа сигналов и построенных графиков можно сделать следующие выводы:  
Результаты, полученные при ручном расчете в среде WxMaxima, полностью совпадают с данными, вычисленными программой на языке C. Это подтверждает:

* + Корректность аналитических формул для Uvx(t) и Uvix(t)
  + Правильность алгоритмов, реализованных в программе
  + Точность численных методов

Графики входного (Uvx(t)) и выходного (Uvix(t)) сигналов демонстрируют ожидаемое поведение:

* + Выходной сигнал экспоненциально возрастает при увеличении входного напряжения
  + В точках t = 0.0000025 и t = 0.0000075 наблюдаются характерные изломы, соответствующие изменению режима работы цепи

## Интерфейс

## Заключение

В ходе выполнения курсовой работы были успешно решены поставленные задачи:

* Разработана программа на языке C для анализа сигналов в электрической цепи.
* Проведены контрольные расчеты в WxMaxima, подтвердившие корректность алгоритмов.
* Построены графики входного и выходного сигналов, наглядно демонстрирующие их зависимость от времени.
* Обеспечена запись результатов в файлы для последующей обработки.

Программа показала свою эффективность и точность расчетов, что позволяет рекомендовать ее для учебных целей. В дальнейшем функционал можно расширить, добавив поддержку других типов сигналов.

## Список литературы

1. ГОСТ 19.402-78. Единая система программной документации. Описание программы. — Введ. 01.01.1980. — М.: Издательство стандартов, 1988. — 10 с.
2. Бережной, Л. Н. Информатика: Анализ сигнала на выходе электрической цепи: Методические указания к курсовой работе / Л. Н. Бережной, И. О. Воронцова, Д. В. Окунева. — Санкт-Петербург: Издательство СПбГУТ, 2012. — 20 с.
3. Kernighan B. The C Programming Language / B. Kernighan, D. Ritchie. – 2nd ed. – Englewood Cliffs : Prentice Hall, 1988. – 272 p. – ISBN 0-13-110362-8.
4. Maxima Documentation. — URL: https://maxima.sourceforge.io/documentation.html (Электронный ресурс, дата обращения: 21.05.2025).