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 – 4 вариант

Входной сигнал – 4 вариант

Выполнил студент(ка):

Гуринов А. Б. ИКПИ-44 \_\_\_\_\_\_\_\_\_\_\_\_\_

*(Ф.И.О., № группы) (подпись)*

Дата выполнения:

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

Проверил:

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

*(Ф.И.О.) (подпись)*

Санкт-Петербург

2025

## Аннотация

Программа разработана для анализа сигналов в электрических цепях. Она вычисляет выходное напряжение на основе заданного входного сигнала и передаточной характеристики цепи. Реализована на языке C с использованием математических функций и библиотек для обеспечения точности расчётов.

Ключевые возможности программы:

* Расчёт выходного сигнала для заданного количества временных точек.
* Автоматическое формирование массивов входного и выходного напряжений.
* Сохранение результатов в текстовые файлы для дальнейшей обработки.
* Визуализация данных с построением графиков в среде WxMaxima.
* Проверка результатов через контрольные расчёты.

Программа включает функции для определения характеристик сигнала, таких как длительность импульса и параметры фронтов, с заданной точностью. Реализовано интерактивное меню для удобства взаимодействия.

Тестирование подтвердило корректность работы программы. Полученные данные соответствуют теоретическим расчётам, что позволяет использовать программу в учебных и исследовательских целях.

Оглавление

[Аннотация 2](#_Toc199780801)

[Задание 4](#_Toc199780802)

[Таблица идентификаторов 5](#_Toc199780803)

[Блок схемы 7](#_Toc199780804)

[Текст программы 21](#_Toc199780805)

[Код на C 21](#_Toc199780806)

[wxMaxima код 29](#_Toc199780807)

[Bash-скрипт 30](#_Toc199780808)

[Описание программы 31](#_Toc199780809)

[Общие сведения 31](#_Toc199780810)

[Функциональное назначение 31](#_Toc199780811)

[Вызов и загрузка 32](#_Toc199780812)

[Выходные данные: 32](#_Toc199780813)

[Графики 33](#_Toc199780814)

[Интерфейс 34](#_Toc199780815)

[Заключение 36](#_Toc199780816)

[Список литературы 37](#_Toc199780817)

## 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

| **Идентификатор** | **Тип** | **Категория** | **Описание** |
| --- | --- | --- | --- |
| **menu.c** |  |  |  |
| CLEAR\_SCREEN() | macro | macro | Кроссплатформенная очистка экрана |
| choice | int | variable | Выбор пользователя в меню |
| stop | char | variable | Флаг завершения программы |
| **io.c** |  |  |  |
| file | FILE\* | variable | Указатель на файл (в save\_array) |
| i | int | variable | Счетчик циклов |
| N | int | variable | Количество точек (в нескольких функциях) |
| Uvx\_array[] | double[] | variable | Массив входных напряжений |
| Uvix\_array[] | double[] | variable | Массив выходных напряжений |
| time\_array[] | double[] | variable | Массив временных точек |
| error\_rate | double | variable | Величина ошибки (в print\_param\_with\_an\_error) |
| eps | double | variable | Требуемая точность |
| par, par1 | double | variable | Вычисляемые параметры |
| **logic.c** |  |  |  |
| i | int | variable | Счетчик циклов |
| dt | double | variable | Шаг по времени |
| t | double | variable | Временная переменная |
| max/min | double | variable | Макс./мин. значения (в get\_max/min\_value) |
| Umax, Umin | double | variable | Граничные напряжения |
| U1, U2 | double | variable | Промежуточные напряжения |
| dlit | double | variable | Длительность фронта |
| **Константы (logic.c)** |  |  |  |
| U | 5.0 | const | Амплитуда напряжения |
| a | 2.0 | const | Коэффициент модели |
| tn, tk | 0.0, 2\*M\_PI | const | Временные границы |
| a1, b1 | 2.0, -5.0 | const | Параметры линейной модели |
| Uvx1 | 20.0 | const | Пороговое напряжение |

## Блок схемы

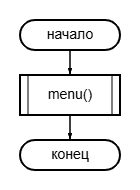


Схема 1

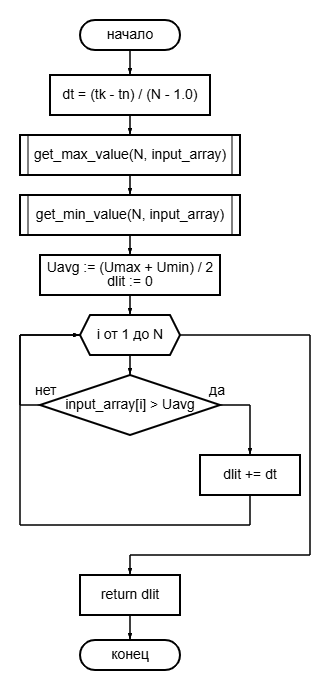


Схема 2

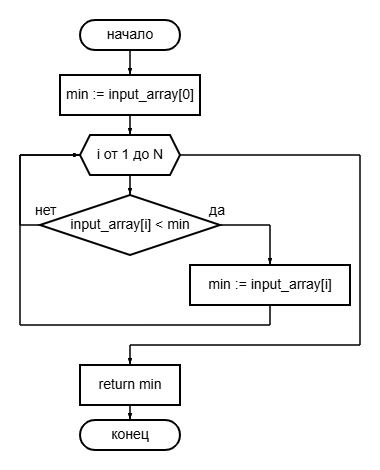


Схема 3

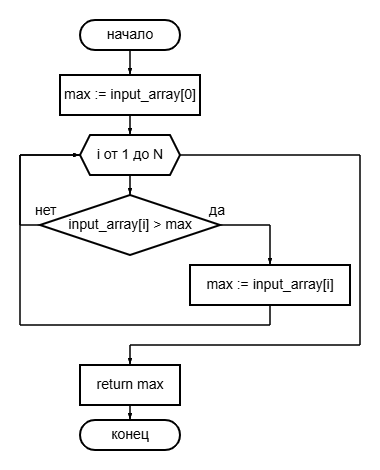


Схема 4

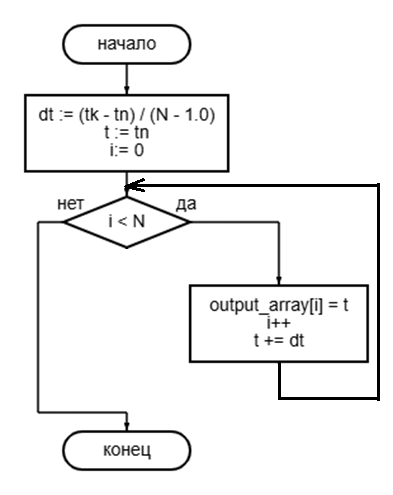


Схема 5

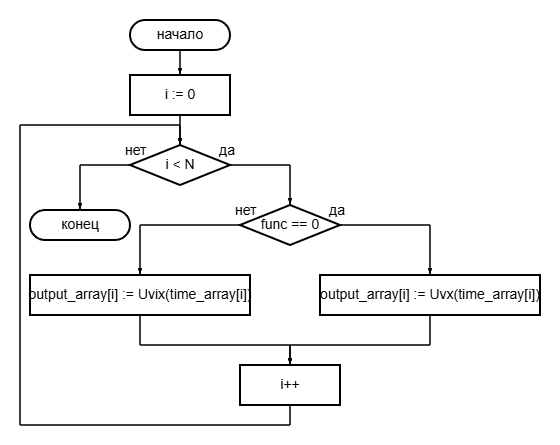


Схема 6

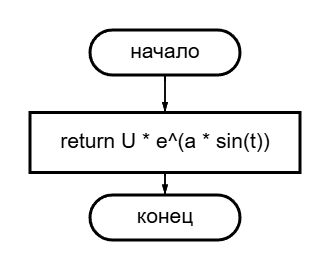


Схема 7

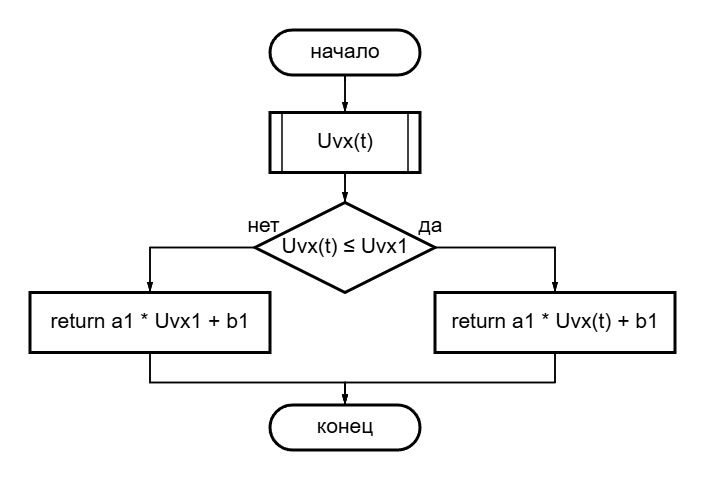


Схема 8

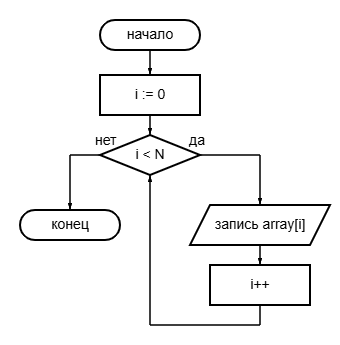


Схема 9

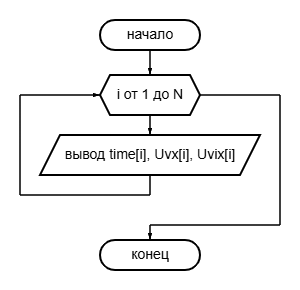


Схема 10

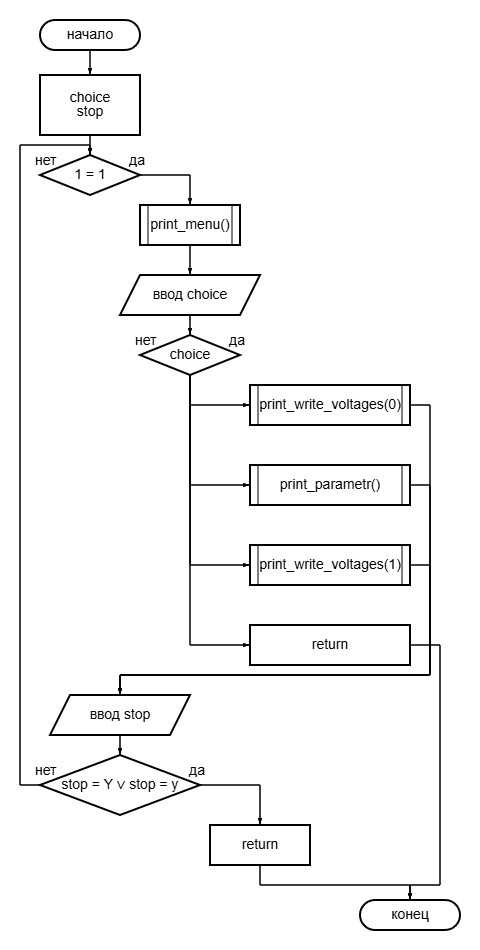


Схема 11

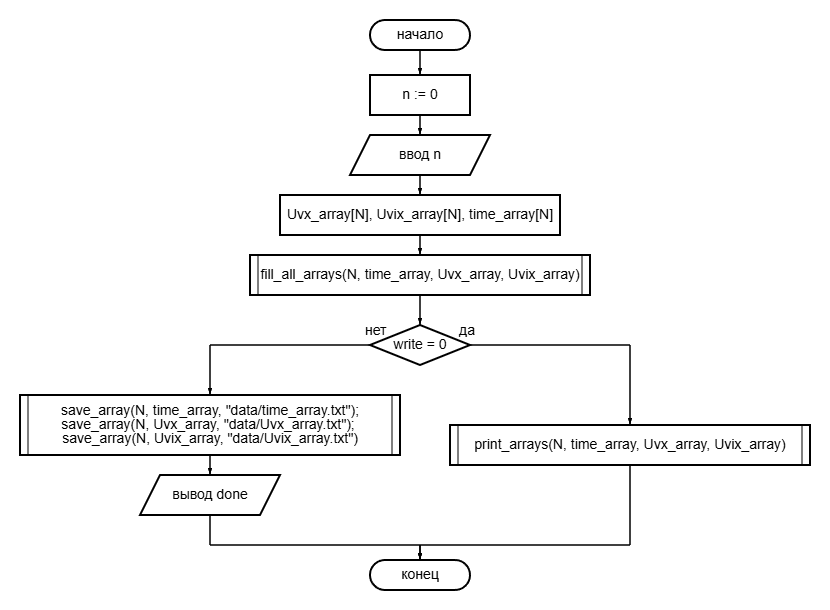


Схема 12

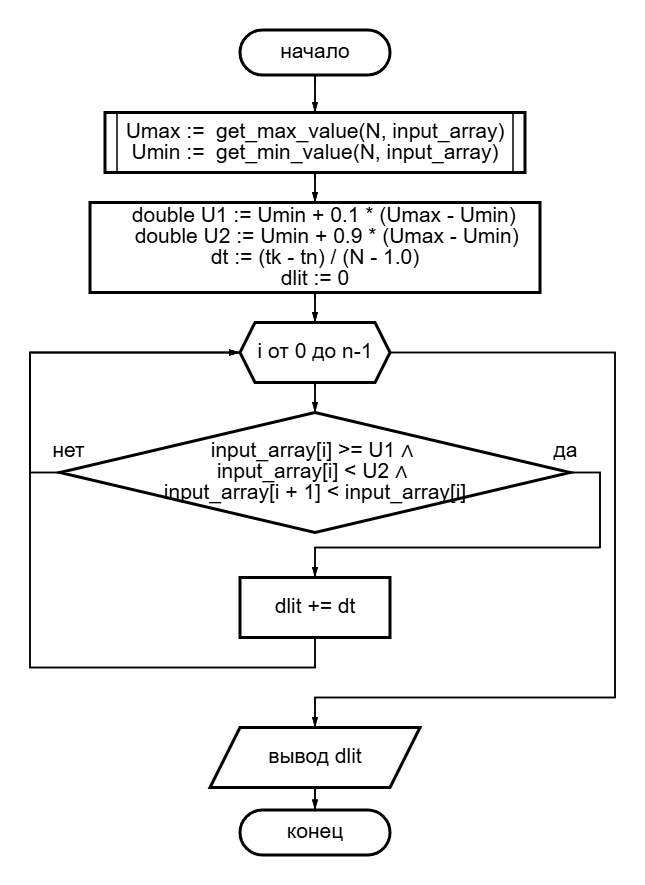


Схема 13

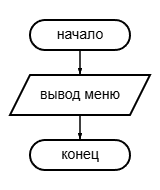


Схема 14

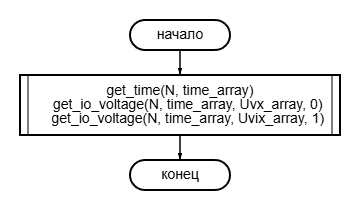


Схема 15

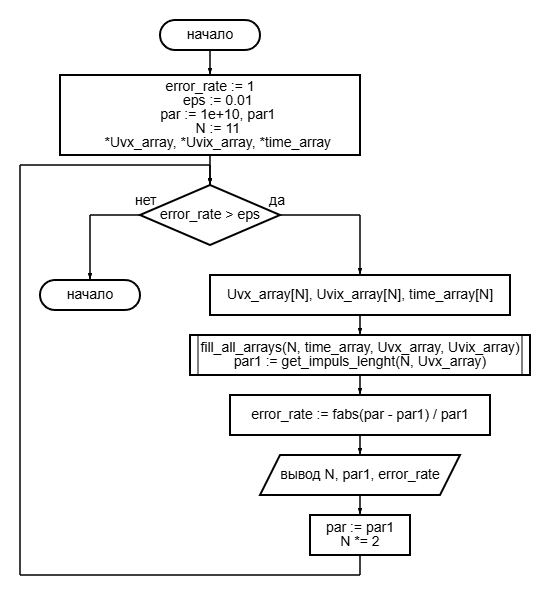


Схема 16

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 - get\_max\_value(int N, double \*input\_array)

Схема 12 - get\_min\_value(int N, double \*input\_array)

Схема 13 - get\_trailing\_edge\_duration (int N, double \*input\_array)

Схема 14 - print\_menu()

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

Схема 16 - print\_param\_with\_an\_error()

## Текст программы

### Код на C

##### main.c

#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);

printf("Trailing edge duration: %f",

get\_trailing\_edge\_duration(N, Uvix\_array));

}

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("| %4d | %12.5e | %8.4f | %8.4f |\n", i + 1, time[i], Uvx[i], Uvix[i]);

}

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

}

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

{

double error\_rate = 1;

double eps = 0.01;

double par = 1e+10, par1;

int N = 11;;

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

printf("| N | Parametr | Error rate |\n");

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

while (error\_rate > eps)

{

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

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

par1 = get\_trailing\_edge\_duration(N, Uvix\_array);

error\_rate = (par - par1) / par1;

error\_rate = (error\_rate < 0.0 ? -error\_rate : error\_rate);

printf("| %4d | %10.7f | %10.4f |\n", N, par1, (error\_rate > 9999 ? 999.9999 : error\_rate));

par = par1;

N \*= 2;

}

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 - Run verification calculation for N points\n");

printf("2 - Compute parameter using N points\n");

printf("3 - Compute parameter with specified accuracy\n");

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

printf("5 - Quit\n");

}

##### logic.c

#include <math.h>

#include <stdlib.h>

#include <stdio.h>

#define U 5.0

#define a 2.0

#define tn 0.0

#define tk 2.0 \* M\_PI

#define a1 2.0

#define b1 -5.0

#define Uvx1 20.0

double Uvx(double t)

{

return U \* exp(a \* sin(t));

}

double Uvix(double t)

{

if (Uvx(t) <= Uvx1)

{

return a1 \* Uvx(t) + b1;

}

return a1 \* Uvx1 + b1;

}

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 get\_max\_value(int N, double \*input\_array)

{

double max = input\_array[0];

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

{

if (input\_array[i] > max)

{

max = input\_array[i];

}

}

return max;

}

double get\_min\_value(int N, double \*input\_array)

{

double min = input\_array[0];

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

{

if (input\_array[i] < min)

{

min = input\_array[i];

}

}

return min;

}

double get\_trailing\_edge\_duration(int N, double \*input\_array)

{

double Umax = get\_max\_value(N, input\_array);

double Umin = get\_min\_value(N, input\_array);

double U1 = Umin + 0.1 \* (Umax - Umin);

double U2 = Umin + 0.9 \* (Umax - Umin);

double dt = (tk - tn) / (N - 1.0);

double dlit = 0;

for (int i = 0; i < N - 1; i++)

{

if (input\_array[i] >= U1 && input\_array[i] < U2 && input\_array[i + 1] < input\_array[i])

{

dlit += dt;

}

}

return dlit;

}

##### 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();

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

#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\_trailing\_edge\_duration(int N, double \*input\_array);

#endif // LOGIC\_H\_INCLUDED

### wxMaxima код

fpprintprec:5$;

tn: 0$;

tk: 2 \* π, numer$; /\* t начальное и конечное \*/

U: 5$; a: 2$; /\* константы для КОД А \*/

a1: 2$; b1: -5$; Uvx1: 20$; /\* константы для КОД Б \*/

N: 50$; dt: (tk - tn) / (N - 1)$; /\* N - количество точек;

dt - шаг между двумя очками \*/

Uvx(t):= U \* exp(a \* sin(t))$; /\* формула КОД А \*/

Uvix(t):=

if Uvx(t) <= Uvx1

then a \* Uvx(t) + b1

else a \* Uvx1 + b1$; /\* формула КОД Б \*/

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)],

[time, tn, tk],

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

[title, "График функции Uvx(t) средствами wxMaxima"]

)$;

wxplot2d(

[discrete, data\_t, data\_Uvx],

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

[title, "График функции Uvx(t) средствами C"]

)$;

wxplot2d(

[Uvx(time)],

[time, tn, tk],

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

[title, "График функции Uvix(t) средствами wxMaxima"]

)$;

wxplot2d(

[discrete, data\_t, data\_Uvx],

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

[title, "График функции Uvix(t) средствами C"]

)$;

### 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.
* bash.sh – скрипт для автоматизации запуска.

**Используемые языки и инструменты:**

* C – основной язык разработки.
* Bash – автоматизация вызова программы.
* WxMaxima – построение графиков и проверка расчётов.

Функциональное назначение  
**Программа выполняет следующие задачи***:*

1. Интерактивное меню – выбор режима работы.
2. Расчёт сигналов – вычисление входного и выходного напряжений.
3. Сохранение данных – запись результатов в файлы (massiv\_t.txt, massiv\_Uvx.txt, massiv\_Uvix.txt).
4. Визуализация – автоматическое построение графиков в WxMaxima.

**Алгоритм работы:**

* Пользователь запускает bash.sh.
* Программа выполняет расчёты и сохраняет данные.
* WxMaxima строит графики на основе сохранённых файлов.

**Используемые технические средства:**

* Git Bash / WSL – для работы скриптов.
* WxMaxima – для визуализации.

Вызов и загрузка**Требования:**

* *Установленные WxMaxima и Bash.*

**Запуск*:***

* *Перейти в директорию с программой.*
* *Выполнить команду:*

./bash.sh

Выходные данные:**Создаются файлы:**

* massiv\_t.txt – временные точки.
* massiv\_Uvx.txt – входные напряжения.
* massiv\_Uvix.txt – выходные напряжения.

## Графики

|  |  |  |
| --- | --- | --- |
|  | maxima | c |
| Uvx(t) |  |  |
| Uvix(t) |  |  |

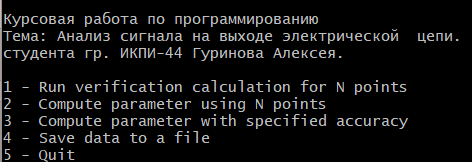
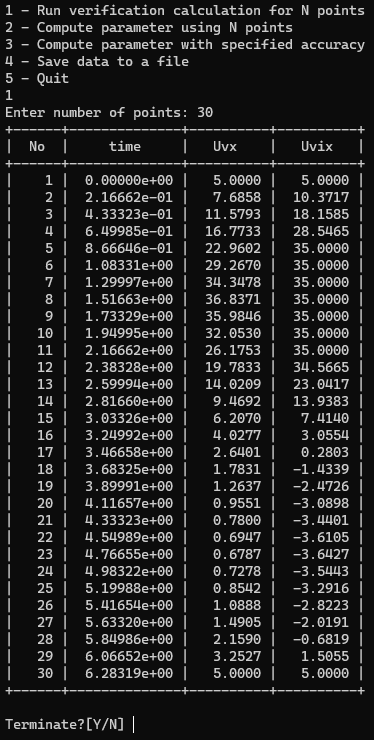
Результаты ручного расчёта в WxMaxima совпадают с данными программы, что подтверждает:

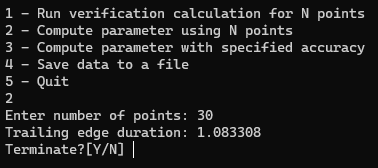
* Корректность формул для Uvx(t) и Uvix(t).
* Точность алгоритмов.

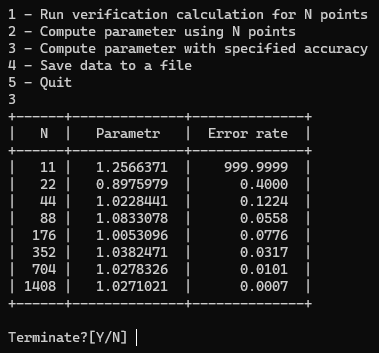
Графики демонстрируют ожидаемое поведение сигналов:

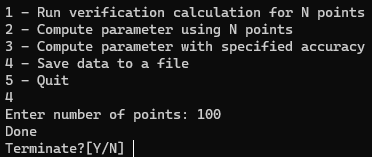
* Выходной сигнал изменяется в соответствии с входным.
* Наблюдаются характерные точки, соответствующие режимам работы цепи.

## Интерфейс







## Заключение

В ходе работы решены все поставленные задачи:

1. Разработана программа на C для анализа сигналов.
2. Проведена верификация результатов в WxMaxima.
3. Построены графики, наглядно демонстрирующие поведение сигналов.
4. Реализована система работы с файлами для удобства анализа.

Программа показала высокую точность и эффективность, что делает её полезной для учебного процесса.

**Перспективы развития:**

* Добавление поддержки различных типов сигналов.
* Реализация частотного анализа.
* Создание графического интерфейса.

Проведённая работа позволила получить опыт в области программирования математических моделей и автоматизации расчётов. Результаты могут быть использованы для дальнейших исследований.

## 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).