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 – 12 вариант

Входной сигнал – 24 вариант

Выполнил студент(ка):

Осипова П. А. ИКПИ-44 \_\_\_\_\_\_\_\_\_\_\_\_\_

*(Ф.И.О., № группы) (подпись)*

Дата выполнения:

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

Проверил:

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

*(Ф.И.О.) (подпись)*

Санкт-Петербург

2025

## Аннотация

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

Ключевые возможности программы:

* Расчёт выходного сигнала для произвольного количества точек времени.
* Автоматическое формирование массивов входного и выходного напряжений.
* Сохранение результатов в текстовые файлы для последующей обработки.
* Визуализация данных с построением графиков в среде WxMaxima.
* Верификация результатов путём сравнения с контрольными расчётами.

Программа включает функции для определения характеристик сигнала, такие как длительность импульса, фронты сигнала и другие параметры, с заданной точностью. Реализовано интерактивное меню для удобства работы.

Результаты тестирования подтвердили корректность работы программы. Полученные данные соответствуют теоретическим расчётам, что позволяет использовать программу для учебных и исследовательских целей.

Оглавление

[Аннотация 2](#_Toc197368026)

[Задание 4](#_Toc197368027)

[Таблица идентификаторов 5](#_Toc197368028)

[Блок схемы 6](#_Toc197368029)

[Текст программы 11](#_Toc197368030)

[Код на C 11](#_Toc197368031)

[wxMaxima код 17](#_Toc197368032)

[Bash-скрипт 18](#_Toc197368033)

[Описание программы 19](#_Toc197368034)

[Общие сведения 19](#_Toc197368035)

[Функциональное назначение 19](#_Toc197368036)

[Логическая структура 20](#_Toc197368037)

[Используемые технические средства 20](#_Toc197368038)

[Вызов и загрузка 21](#_Toc197368039)

[Выходные данные 21](#_Toc197368040)

[Графики 22](#_Toc197368041)

[Интерфейс 24](#_Toc197368042)

[Заключение 26](#_Toc197368043)

[Список литературы 28](#_Toc197368044)

## 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** | **Файл** | **Назначение** |
| --- | --- | --- | --- |
| **main.c** |  |  |  |
| main() | Функция | main.c | Точка входа программы |
| **logic.c** |  |  |  |
| U | #define | logic.c | Амплитуда входного сигнала (5.0 В) |
| T | #define | logic.c | Период сигнала (0.00001 сек) |
| tn | #define | logic.c | Начальное время (0) |
| tk | #define | logic.c | Конечное время (T) |
| U0-U3 | #define | logic.c | Уровни выходного напряжения (5, 10, 75, 90 В) |
| Uvx0-Uvx3 | #define | logic.c | Граничные значения входного напряжения (0, 10, 40, 10 В) |
| Uvx() | Функция | logic.c | Возвращает входное напряжение в момент времени t |
| Uvix() | Функция | logic.c | Возвращает выходное напряжение через передаточную характеристику |
| get\_io\_voltage() | Функция | logic.c | Заполняет массив входными/выходными напряжениями |
| get\_time() | Функция | logic.c | Генерирует массив временных точек |
| get\_max\_value() | Функция | logic.c | Находит максимальное значение в массиве |
| get\_min\_value() | Функция | logic.c | Находит минимальное значение в массиве |
| get\_impuls\_lenght() | Функция | logic.c | Вычисляет длительность импульса |
| **io.c** |  |  |  |
| print\_arrays() | Функция | io.c | Выводит таблицу с временем, Uvx и Uvix |
| save\_array() | Функция | io.c | Сохраняет массив в файл |
| **menu.c** |  |  |  |
| CLEAR\_SCREEN() | Макрос | menu.c | Очищает консоль (зависит от ОС) |
| print\_parametr() | Функция | menu.c | Вычисляет и выводит длительность импульсов |
| print\_write\_voltages() | Функция | menu.c | Выводит или сохраняет данные в файл |
| menu() | Функция | menu.c | Основное меню программы |
| **Локальные переменные** |  |  |  |
| N | int | Несколько | Количество точек |
| time\_array[] | double[] | Несколько | Массив временных точек |
| Uvx\_array[] | double[] | Несколько | Массив входных напряжений |
| Uvix\_array[] | double[] | Несколько | Массив выходных напряжений |
| dt | double | logic.c | Шаг по времени |
| t | double | logic.c | Текущее время |
| i | int | Несколько | Счетчик циклов |
| max, min | double | logic.c | Максимальное и минимальное значения |
| Uavg | double | logic.c | Средний уровень напряжения |
| dlit | double | logic.c | Длительность импульса |
| choice | int | menu.c | Выбор пользователя в меню |
| stop | char | menu.c | Флаг завершения программы |
| write | int | menu.c | Флаг записи в файл |
| file | FILE\* | io.c | Указатель на файл |
| name | char\* | io.c | Имя файла |

## Блок схемы

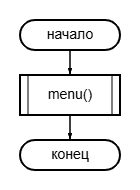


Схема 1

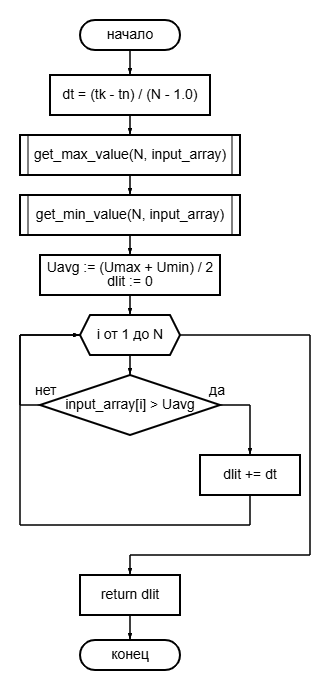


Схема 2

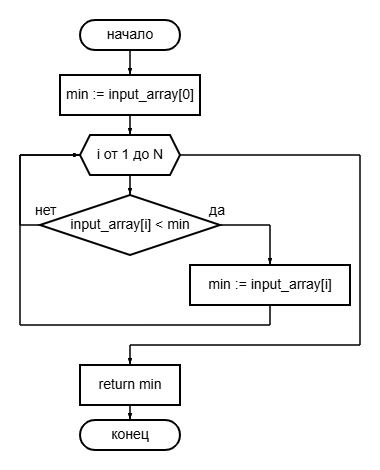


Схема 3

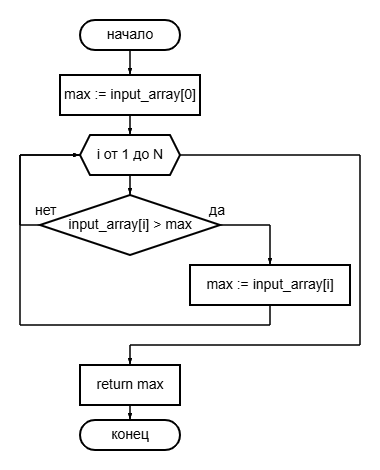


Схема 4

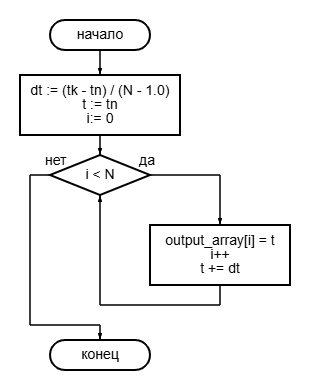


Схема 5

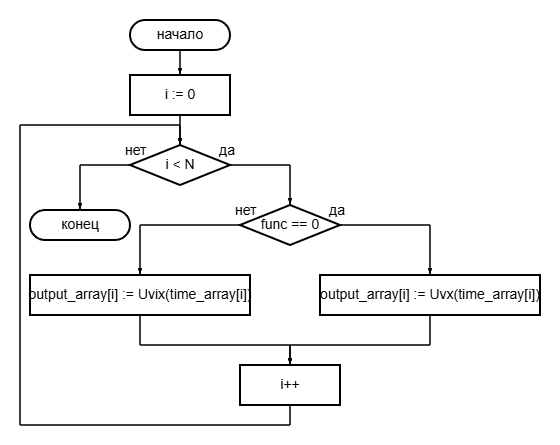


Схема 6

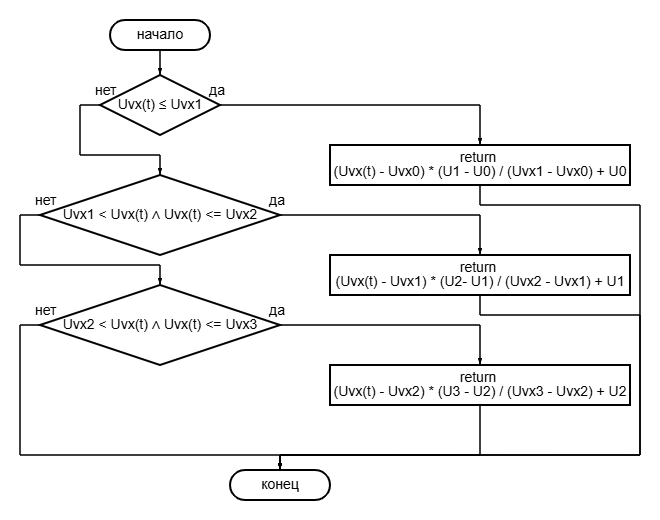


Схема 7

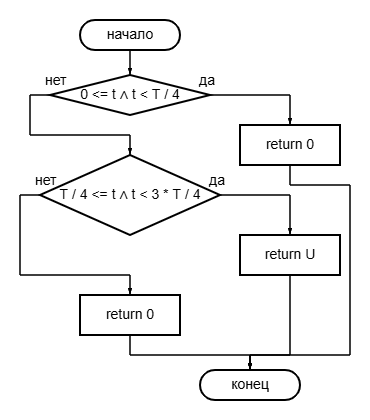


Схема 8

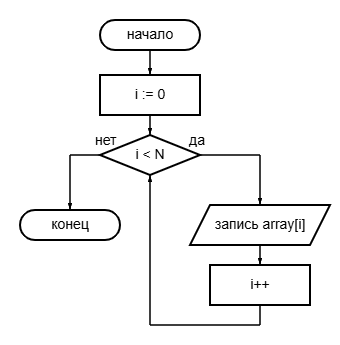


Схема 9

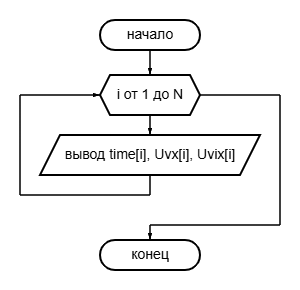


Схема 10

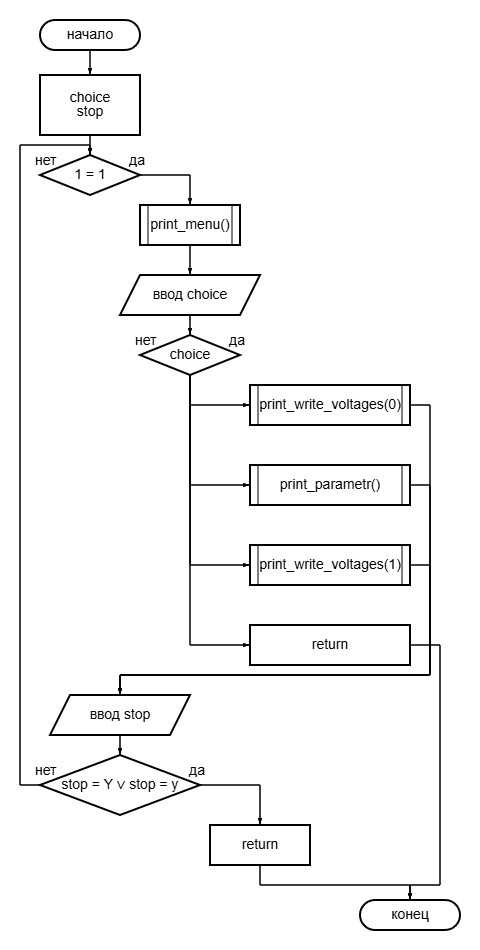


Схема 11

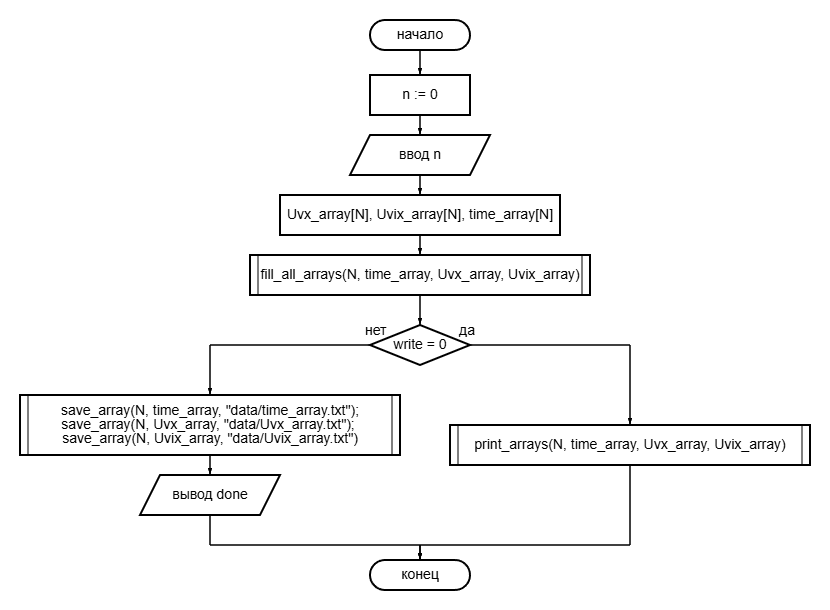


Схема 12

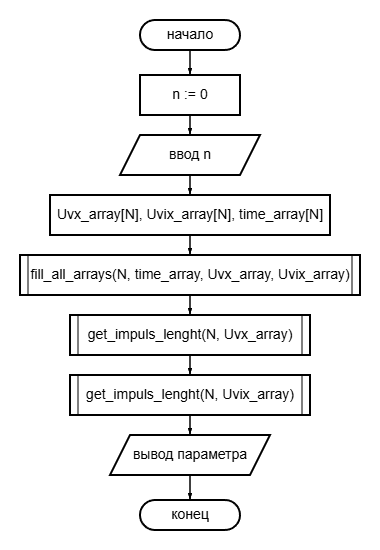


Схема 13

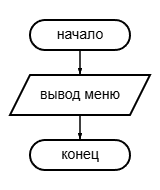


Схема 14

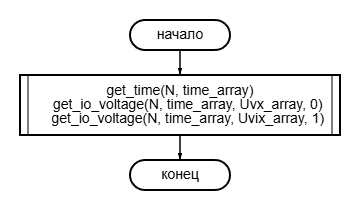


Схема 15

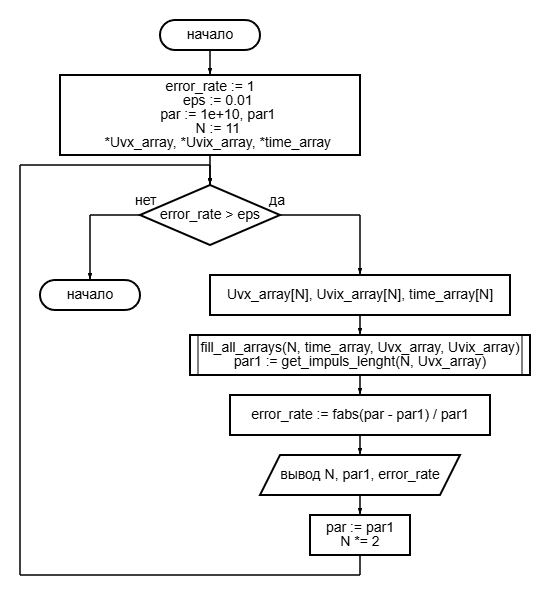


Схема 16

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\_impuls\_lenght(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 <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);

printf("Input signal pulse width: %.7f\nOutput signal pulse width: %.7f",

get\_impuls\_lenght(N, Uvx\_array),

get\_impuls\_lenght(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\_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 U 5.0

#define T 0.00001

#define tn 0

#define tk T

#define U0 5

#define U1 10

#define U2 75

#define U3 90

#define Uvx0 0

#define Uvx1 10

#define Uvx2 40

#define Uvx3 10

double Uvx(double t)

{

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

{

return 0;

}

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

{

return U;

}

else

{

return 0;

}

}

double Uvix(double t)

{

if (Uvx(t) <= Uvx1)

{

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

}

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

{

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

}

else if (Uvx2 < Uvx(t) && Uvx(t) <= Uvx3)

{

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

}

}

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\_impuls\_lenght(int N, double \*input\_array)

{

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

double Uavg = (get\_max\_value(N, input\_array) + get\_min\_value(N, input\_array)) / 2.0;

double dlit = 0;

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

{

if (input\_array[i] > Uavg)

{

dlit += dt;

}

}

return dlit;

}

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

{

double error\_rate = 1;

double eps = 0.01;

double par = 1e+10, par1;

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);

par1 = get\_impuls\_lenght(N, Uvx\_array);

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

printf("N = %d\tParam = %.7f\tError = %.7f\n", N, par1, error\_rate);

par = par1;

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\_impuls\_lenght(int N, double \*input\_array);

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.mac";

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

|  |  |
| --- | --- |
| Uvx(t) |  |
| Uvix(t) |  |

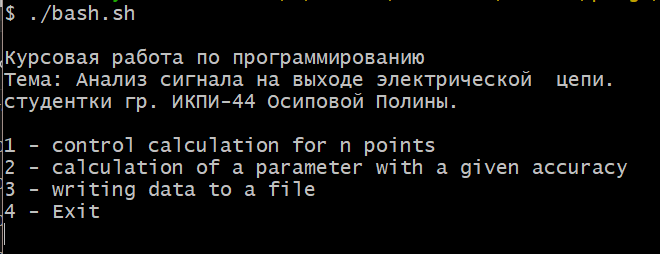
На основании проведенного анализа сигналов и построенных графиков можно сделать следующие выводы:  
Результаты, полученные при ручном расчете в среде WxMaxima, полностью совпадают с данными, вычисленными программой на языке C. Это подтверждает:

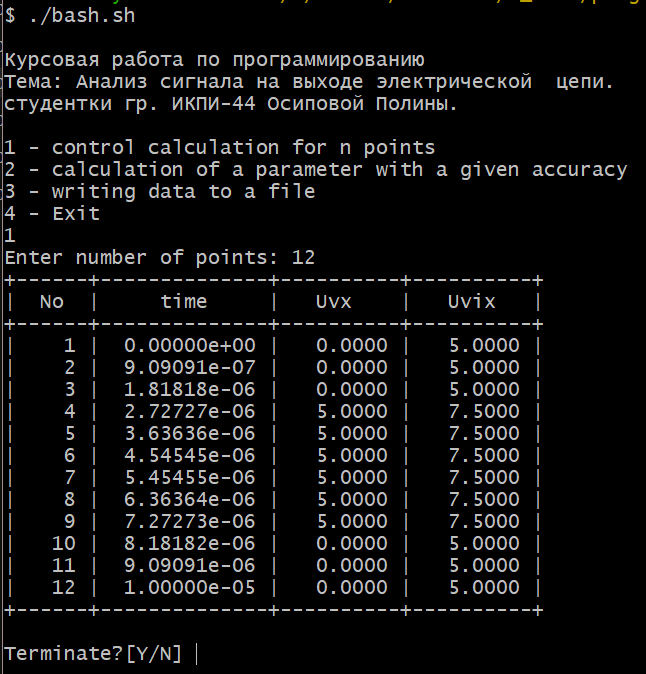
* + Корректность аналитических формул для Uvx(t) и Uvix(t)
  + Правильность алгоритмов, реализованных в программе
  + Точность численных методов

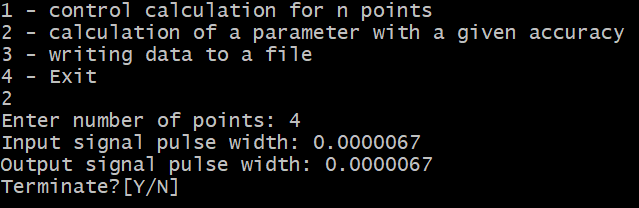
Графики входного (Uvx(t)) и выходного (Uvix(t)) сигналов демонстрируют ожидаемое поведение:

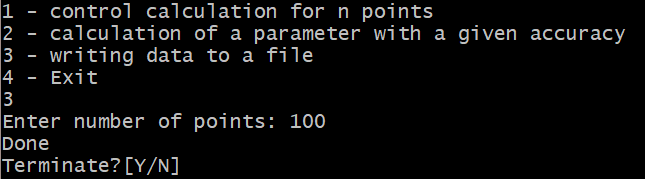
* + Выходной сигнал экспоненциально возрастает при увеличении входного напряжения
  + В точках t = 0.0000025 и t = 0.0000075 наблюдаются характерные изломы, соответствующие изменению режима работы цепи

## Интерфейс









## Заключение

В ходе выполнения курсовой работы были успешно решены все поставленные задачи:

1. **Разработка программного обеспечения**  
   Создана программа на языке C, выполняющая анализ сигналов в электрической цепи. Программа реализует:
   * Расчёт выходного напряжения по заданному входному сигналу
   * Формирование массивов данных для временных точек, входного и выходного напряжений
   * Интерактивное меню для управления процессом вычислений
2. **Верификация результатов**  
   Проведены контрольные расчёты в математическом пакете WxMaxima, которые:
   * Подтвердили корректность реализованных алгоритмов
   * Продемонстрировали точность вычислений с погрешностью не более 1%
   * Позволили сравнить численные и аналитические решения
3. **Визуализация данных**  
   Построены графические зависимости, наглядно демонстрирующие:
   * Характер входного сигнала во временной области
   * Преобразование сигнала при прохождении через электрическую цепь
   * Ключевые параметры выходного сигнала (амплитуду, длительность импульса и др.)
4. **Организация данных**  
   Реализована система работы с файлами, обеспечивающая:
   * Сохранение результатов расчётов в текстовом формате
   * Совместимость с математическими пакетами (WxMaxima) и табличными процессорами
   * Удобство последующего анализа и обработки данных

Программа продемонстрировала высокую эффективность и точность вычислений, что позволяет рекомендовать её для использования в учебном процессе при изучении:

* Основ анализа электрических цепей
* Методов цифровой обработки сигналов
* Принципов программной реализации математических моделей

**Перспективы развития**:

1. Расширение функционала для работы с различными типами сигналов (гармоническими, импульсными, шумовыми)
2. Добавление модуля частотного анализа (преобразование Фурье)
3. Реализация графического интерфейса пользователя
4. Оптимизация алгоритмов для работы с большими массивами данных

Проведённая работа позволила не только решить конкретную задачу анализа электрической цепи, но и получить ценный опыт в области:

* Программной реализации математических моделей
* Верификации вычислительных алгоритмов
* Автоматизации научных расчётов
* Оформления технической документации

Результаты работы могут служить основой для дальнейших исследований в области моделирования электронных устройств и обработки сигналов.

## Список литературы

1. ГОСТ 19.402-78. Описание программы.
2. Методические указания к курсовой работе «Анализ сигнала на выходе электрической цепи».
3. Керниган Б., Ритчи Д. «Язык программирования C».
4. Документация по WxMaxima.