**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

Факультет Информационных технологий и программной инженерии

Кафедра Программной инженерии и вычислительной техники

КУРСОВАЯ РАБОТА

по дисциплине:

**«Программирование»**

тема: Анализ сигнала на выходе электрической цепи

8 вариант

Выполнил студент(ка):

Золотарев И. 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](#_Toc200070604)

[Задание 4](#_Toc200070605)

[Таблица идентификаторов 5](#_Toc200070606)

[Блок схемы 8](#_Toc200070607)

[Текст программы 20](#_Toc200070608)

[Код на C 20](#_Toc200070609)

[wxMaxima код 26](#_Toc200070610)

[Bash-скрипт 27](#_Toc200070611)

[Описание программы 28](#_Toc200070612)

[Общие сведения 28](#_Toc200070613)

[Функциональное назначение 28](#_Toc200070614)

[Логическая структура 29](#_Toc200070615)

[Используемые технические средства 29](#_Toc200070616)

[Вызов и загрузка 29](#_Toc200070617)

[Выходные данные 29](#_Toc200070618)

[Графики 31](#_Toc200070619)

[Интерфейс 32](#_Toc200070620)

[Заключение 32](#_Toc200070621)

[Список литературы 36](#_Toc200070622)

## Задание

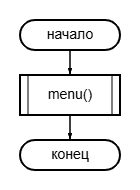
В курсовой работе необходимо для заданной электрической цепи по известному входному сигналу определить выходной сигнал для N равностоящих моментов времени, а затем определить некоторые его характеристики с погрешностью не более 1%.   
В ходе работы студентам необходимо:

* 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.

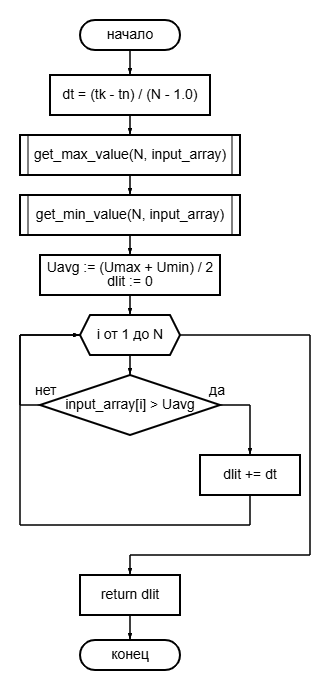
## Таблица идентификаторов

| **Название** | **Тип** | **Файл** | **Назначение** |
| --- | --- | --- | --- |
| **Переменные** |  |  |  |
| N | int | io.c | Количество точек для расчёта/вывода. |
| Uvx\_array | double[] | io.c | Массив значений входного напряжения Uvx. |
| Uvix\_array | double[] | io.c | Массив значений выходного напряжения Uvix. |
| time\_array | double[] | io.c | Массив временных точек. |
| error\_rate | double | io.c | Относительная погрешность вычислений. |
| eps | double | io.c | Требуемая точность (порог остановки расчёта). |
| par, par1 | double | io.c | Параметры для вычисления погрешности. |
| choice | int | menu.c | Выбор пользователя в меню. |
| stop | char | menu.c | Флаг завершения программы (Y/N). |
| t | double | logic.c | Временная переменная для генерации временного массива. |
| dt | double | logic.c | Шаг времени между точками. |
| min | double | logic.c | Минимальное значение в массиве. |
| **defines** |  |  |  |
| U0 | double (макрос) | logic.c | Постоянная составляющая входного напряжения. |
| U1, U2 | double (макрос) | logic.c | Амплитуды гармоник входного напряжения. |
| T | double (макрос) | logic.c | Период сигнала. |
| tn, tk | double (макрос) | logic.c | Начальное и конечное время расчёта. |
| Uvx1, Uvx2 | double (макрос) | logic.c | Границы зон входного напряжения для кусочно-линейной модели Uvix. |
| Uvix1, Uvix2 | double (макрос) | logic.c | Значения выходного напряжения в соответствующих зонах. |
| CLEAR\_SCREEN() | макрос | menu.c | Очистка консоли (зависит от ОС: cls для Windows, clear для Linux/macOS). |
| **Функции** |  |  |  |
| save\_array | void | io.c | Сохраняет массив в файл. |
| input\_nuber\_points | int | io.c | Запрашивает у пользователя количество точек (N). |
| print\_parametr | void | io.c | Выводит минимальное значение Uvx и его время. |
| print\_arrays | void | io.c | Печатает таблицу с массивами time, Uvx, Uvix. |
| print\_param\_with\_an\_error | void | io.c | Вычисляет параметр с заданной точностью, выводя промежуточные результаты. |
| print\_write\_voltages | void | io.c | Сохраняет или выводит данные в зависимости от флага write. |
| print\_menu | void | io.c | Выводит меню программы. |
| Uvx | double | logic.c | Вычисляет входное напряжение в момент времени t. |
| Uvix | double | logic.c | Вычисляет выходное напряжение на основе Uvx(t) (кусочно-линейная модель). |
| get\_io\_voltage | void | logic.c | Заполняет массив Uvx или Uvix в зависимости от флага func. |
| get\_time | void | logic.c | Генерирует массив времени time\_array. |
| fill\_all\_arrays | void | logic.c | Заполняет все массивы: time, Uvx, Uvix. |
| get\_param | double | logic.c | Находит параметр |
| menu | void | menu.c | Основной цикл меню программы. |

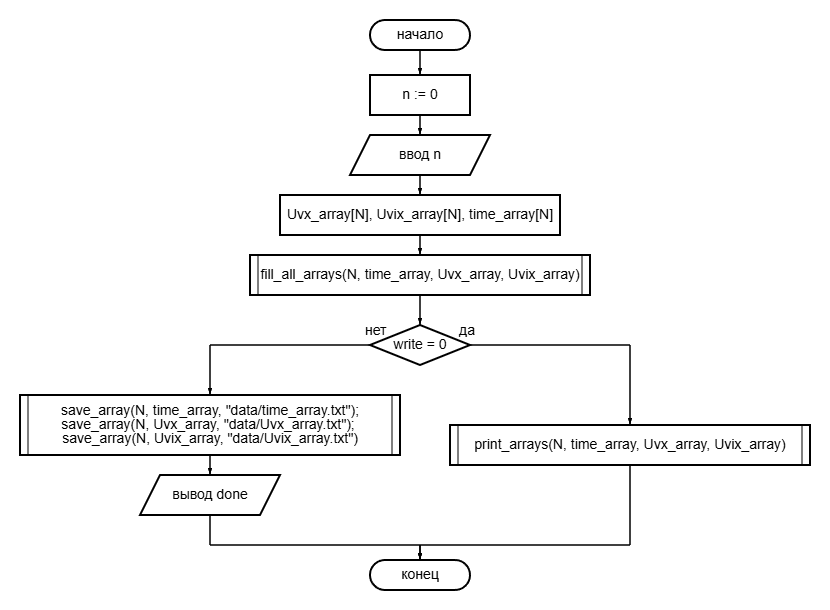
## Блок схемы



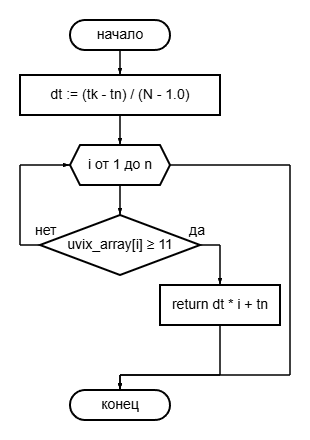
Схема



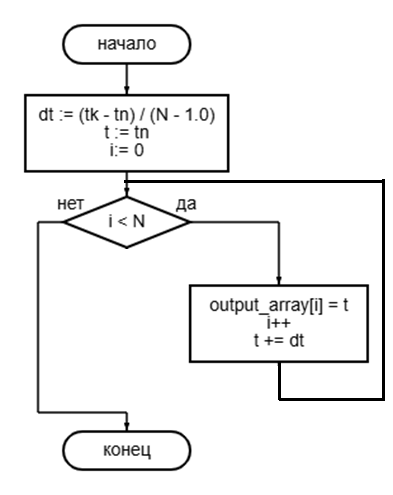
Схема



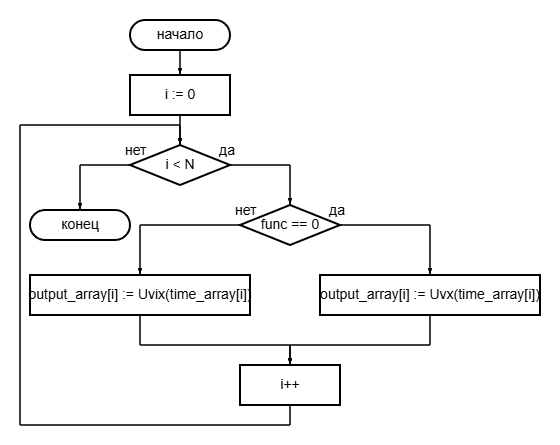
Схема



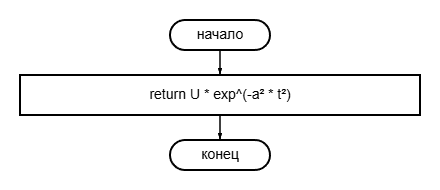
Схема



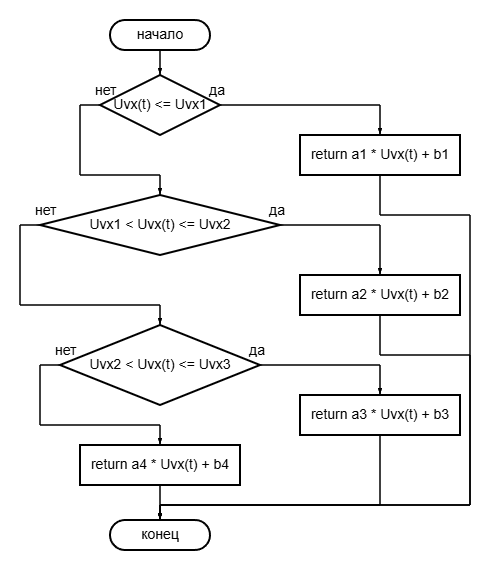
Схема



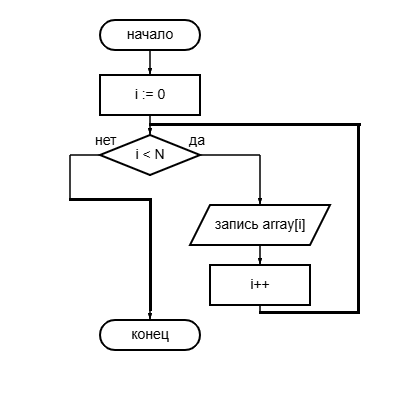
Схема



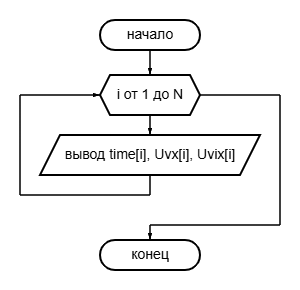
Схема



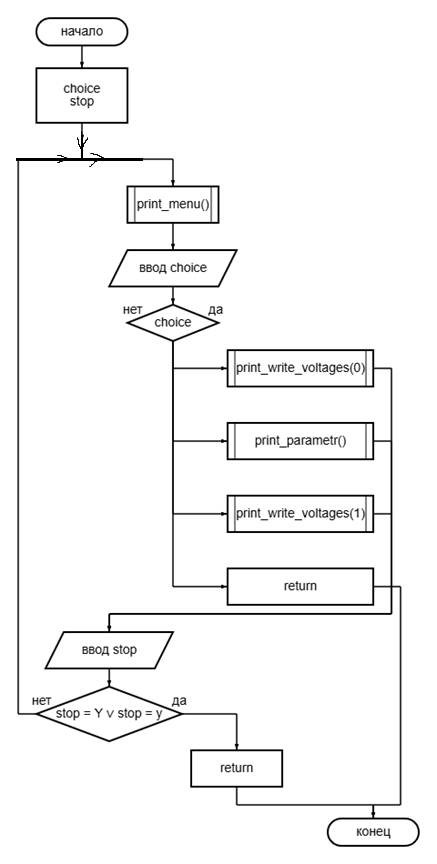
Схема



Схема



Схема



Схема

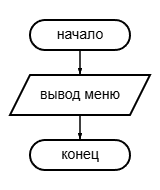


Схема 12

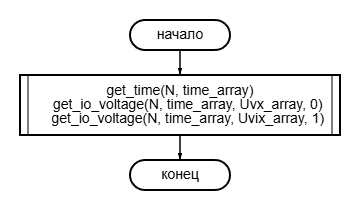


Схема 13

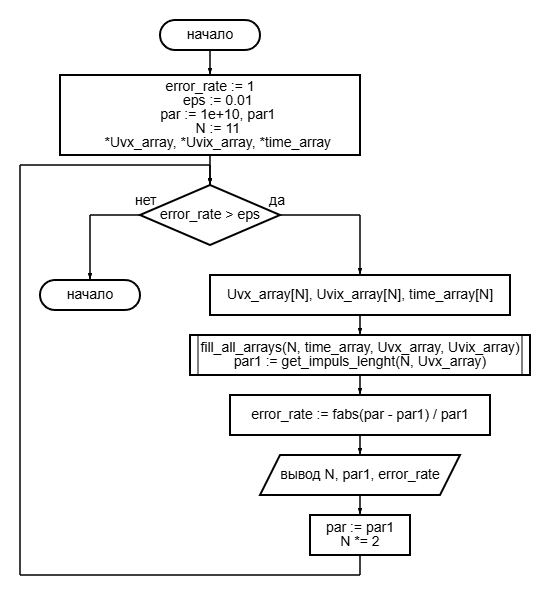


Схема 14

Схема 1 - main()

Схема 2 - print\_parametr()

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

Схема 4 - get\_param(int N, double \*uvix\_array);

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 - menu()

Схема 12 - print\_menu()

Схема 13 - fill\_all\_arrays(int N, double time\_array[], double Uvx\_array[], double Uvix\_array[])

Схема 14 - 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("Time when Uvix -> 11.11: %.4f", get\_param(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 | %8.4f | %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\_param(N, Uvix\_array);

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

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

printf("| %3d | %10.6f | %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>

// constanti dlya KOD A

#define U 10.0

#define a 1.5

// NACHALNAYA i KONECHNAYA TOCHKI

#define tn -1.5

#define tk 1.5

// constanti dlya KOD B

#define a1 0.25

#define a2 0.5

#define a3 1.1

#define a4 25.0

#define b1 9.0

#define b2 7.75

#define b3 0.25

#define b4 -29.75

#define Uvx1 5.0

#define Uvx2 15.0

#define Uvx3 30.0

double Uvx(double t)

{

return U \* exp(-pow(a, 2) \* pow(t, 2));

}

double Uvix(double t)

{

if (Uvx(t) <= Uvx1)

{

return a1 \* Uvx(t) + b1;

}

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

{

return a2 \* Uvx(t) + b2;

}

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

{

return a3 \* Uvx(t) + b3;

}

else

{

return a4 \* Uvx(t) + b4;

}

}

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\_param(int N, double \*uvix\_array)

{

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

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

{

if (uvix\_array[i] >= 11.11)

{

return dt \* i + tn;

}

}

return 0;

}

##### 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 (int N, double \*uvix\_array);

#endif // LOGIC\_H\_INCLUDED

### wxMaxima код

fpprintprec:5$;

U: 10$; a: 1.5$; /\* константы для КОД А \*/

tn: -1.5$;

tk: 1.5$; /\* t начальное и конечное \*/

Uvx1: 5$; Uvx2: 15$; Uvx3: 30$;

a1: 0.25$; a2: 0.5$; a3: 1.1$; a4: 25$;

b1: 9$; b2: 7.75$; b3: 0.25$; b4: -29.75$; /\* константы для КОД Б \*/

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

dt - шаг между двумя очками \*/

Uvx(t):= U \* exp((-a^2) \* t^2)$; /\* формула КОД А \*/

Uvix(t):=

if (Uvx(t) <= Uvx1) then

a1 \* Uvx(t) + b1

else if (Uvx(t) <= Uvx2) then

a2 \* Uvx(t) + b2

else if Uvx(t) <= Uvx3 then

a3 \* Uvx(t) + b3

else

a4 \* Uvx(t) + b4$; /\* 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([[discrete, data\_t, data\_Uvx]],

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

[xlabel, ""],

[ylabel, ""],

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

wxplot2d([Uvx(time)], [time, tn, tk],

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

[xlabel, ""],

[ylabel, ""],

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

wxplot2d([[discrete, data\_t, data\_Uvix]],

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

[xlabel, ""],

[ylabel, ""],

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

wxplot2d([Uvix(time)], [time, tn, tk],

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

[xlabel, ""],

[ylabel, ""],

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

### 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

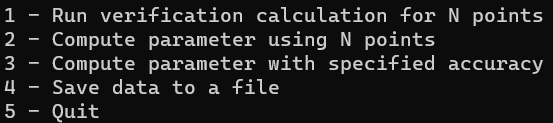
|  |  |  |
| --- | --- | --- |
|  | Средствами C | Средствами wxMaxima |
| Uvx(t) |  |  |
| Uvix(t) |  |  |

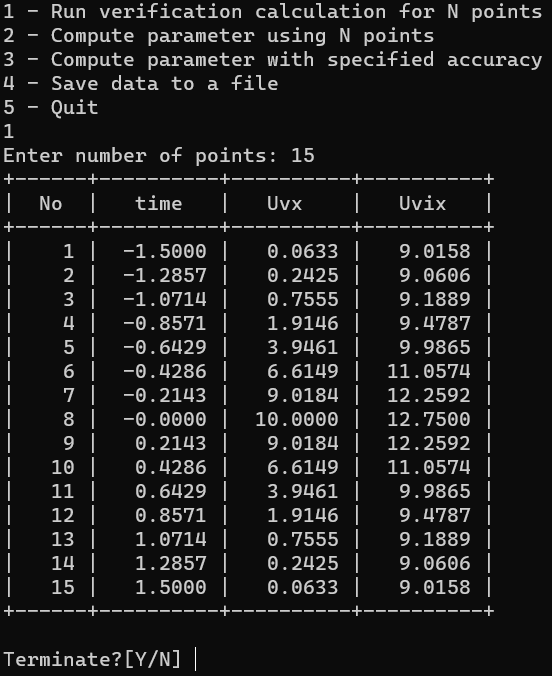
На основании проведенного анализа сигналов и построенных графиков можно сделать следующие выводы:  
Результаты, полученные при ручном расчете в среде WxMaxima, полностью совпадают с данными, вычисленными программой на языке C. Это подтверждает:

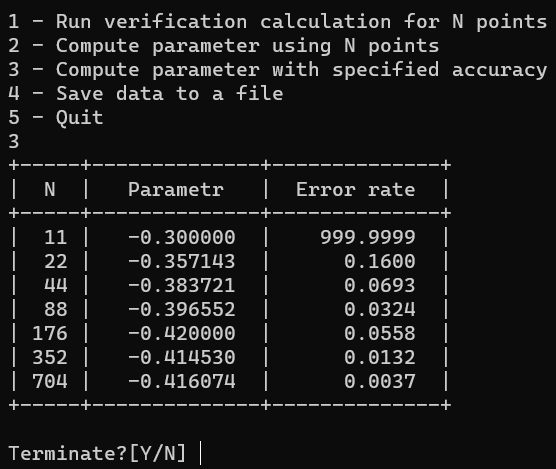
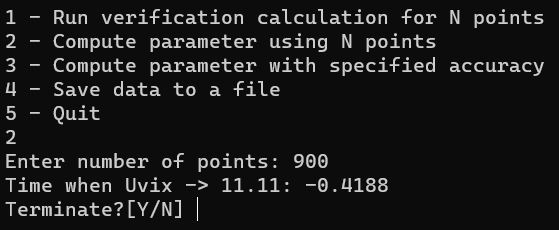
* + Корректность аналитических формул для Uvx(t) и Uvix(t)
  + Правильность алгоритмов, реализованных в программе
  + Точность численных методов

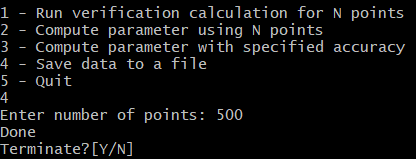
Графики входного (Uvx(t)) и выходного (Uvix(t)) сигналов демонстрируют ожидаемое поведение.

## 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

## 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).