**Оглавление**

[Введение 3](#_Toc135975194)

[I. Задание к курсовой работе 4](#_Toc135975195)

[II. Распечатка контрольного расчета 5](#_Toc135975196)

[III. Таблица идентификаторов 7](#_Toc135975197)

[IV. Блок-схема алгоритма (функции main) 8](#_Toc135975198)

[V. Текст программы 9](#_Toc135975199)

[VI. Графики (обработка результатов) 15](#_Toc135975200)

[Вывод 17](#_Toc135975201)

[Список литературы 18](#_Toc135975202)

# Введение

Курсовая работа посвящена решению задач машинного анализа электрических цепей.

Электрическая цепь задается передаточной характеристикой, которая описывает зависимость напряжения на выходе цепи Uвых(выходного сигнала) от напряжения на входе  Uвх(входного сигнала). Напряжение на входе цепи Uвх (t) и передаточная характеристика могут задаваться либо в виде формул, т.е. аналитически, либо графически. Конкретный вид формул и графиков определяются кодами варианта.

Результатом курсовой работы являются программа на языке С++ с пояснительной запиской. Для заданного варианта электрической цепи и входного сигнала программа должна позволять:

А) рассчитать значения входного сигнала,

Б) рассчитать значения выходного сигнала,

В) рассчитать с заданной точностью значение характеристики выходного сигнала.

**Цель работы:** получение навыков в разработке программ на языке C++, а также анализ и решение поставленной задачи.

# Задание к курсовой работе

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 0.001%. На рисунках ниже представлены варианты параметра входного (рис.1) и выходного сигналов (рис. 2).

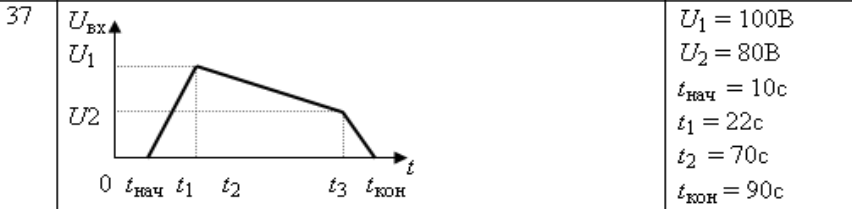


Рис. 1 График зависимости напряжения входного сигнала от времени



Рис. 2 Параметр выходного сигнала в виде функции

# Распечатка контрольного расчета

Передаточная характеристика описана графиком (рис. 1), который необходимо преобразовать в аналитическое выражение. Результатом преобразования является следующая система выражений (рис. 3):

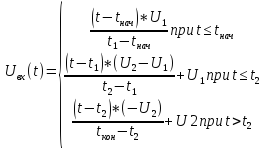


Рис. 3 Окончательный вид передаточной характеристики после преобразований

Воспользуемся программой wxMaxima для выполнения контрольного расчета к курсовой работе.

wxMaxima - свободная система компьютерной алгебры, написанная на языке Common Lisp. Произошла от системы Macsyma, разрабатывавшейся в Массачусетском технологическом институте с 1968 по 1982 годы в рамках проекта Project MAC, финансируемого Министерством энергетики США и другими государственными организациями.

**Расчет в среде wxMaxima:**

fpprintprec:5; /\*количество знаков после запятой\*/

U1:100;U2:80;t1:22;t2:70;

tn:10;tk:90;N:11;dt:(tk-tn)/(N-1),numer; /\*определение начальных условий для расчета Uvx(t)\*/

Uvx(t):=if (t<=t1) then (((t-tn)\*(U1-0))/(t1-tn))

elseif (t<=t2) then ((((t-t1)\*(U2-U1))/(t2-t1))+U1)

elseif (t>t2) then ((((t-t2)\*(0-U2))/(tk-t2))+U2); /\*определение значений Uvx(t)\*/

for t:tn thru tk step dt do display(Uvx(t)),numer; /\*вывод значений на экран морнитора\*/

a: 2;b:4;c:-1;

Uvix(t):=a\*Uvx(t)\*Uvx(t)+b\*Uvx(t)+c;

for t:tn thru tk step dt do display(Uvix(t)),numer;

wxplot2d([Uvx(t)], [t,tn,tk],[gnuplot\_preamble, "set grid;"]);

wxplot2d([Uvix(t)], [t,tn,tk],[y,0,25000], [gnuplot\_preamble, "set grid;"])

**Результаты расчета:**

Полученный график для Uvx в среде wxMaxima представлен на рис. 4:

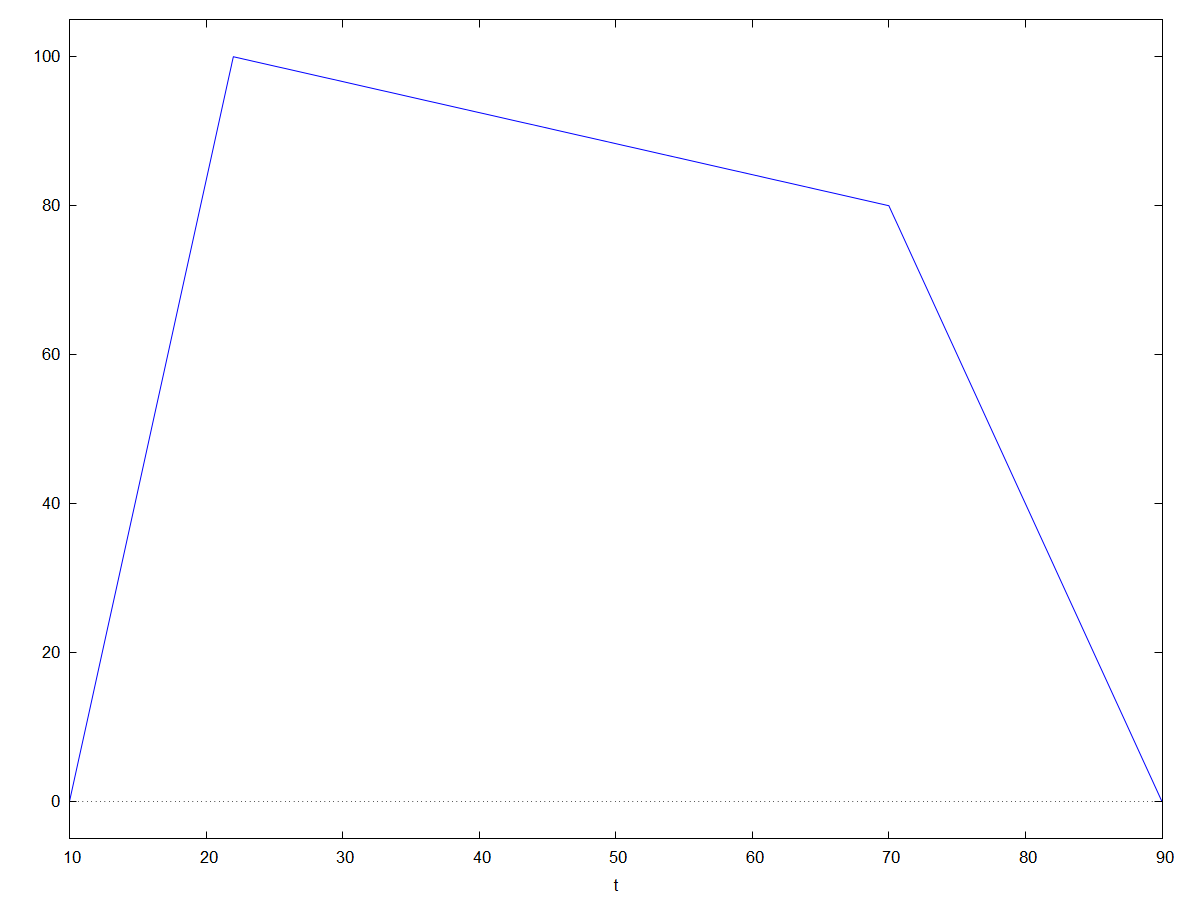


Рис. 4 График для Uvx

Полученный график для Uvix в среде wxMaxima представлен на рис. 5:

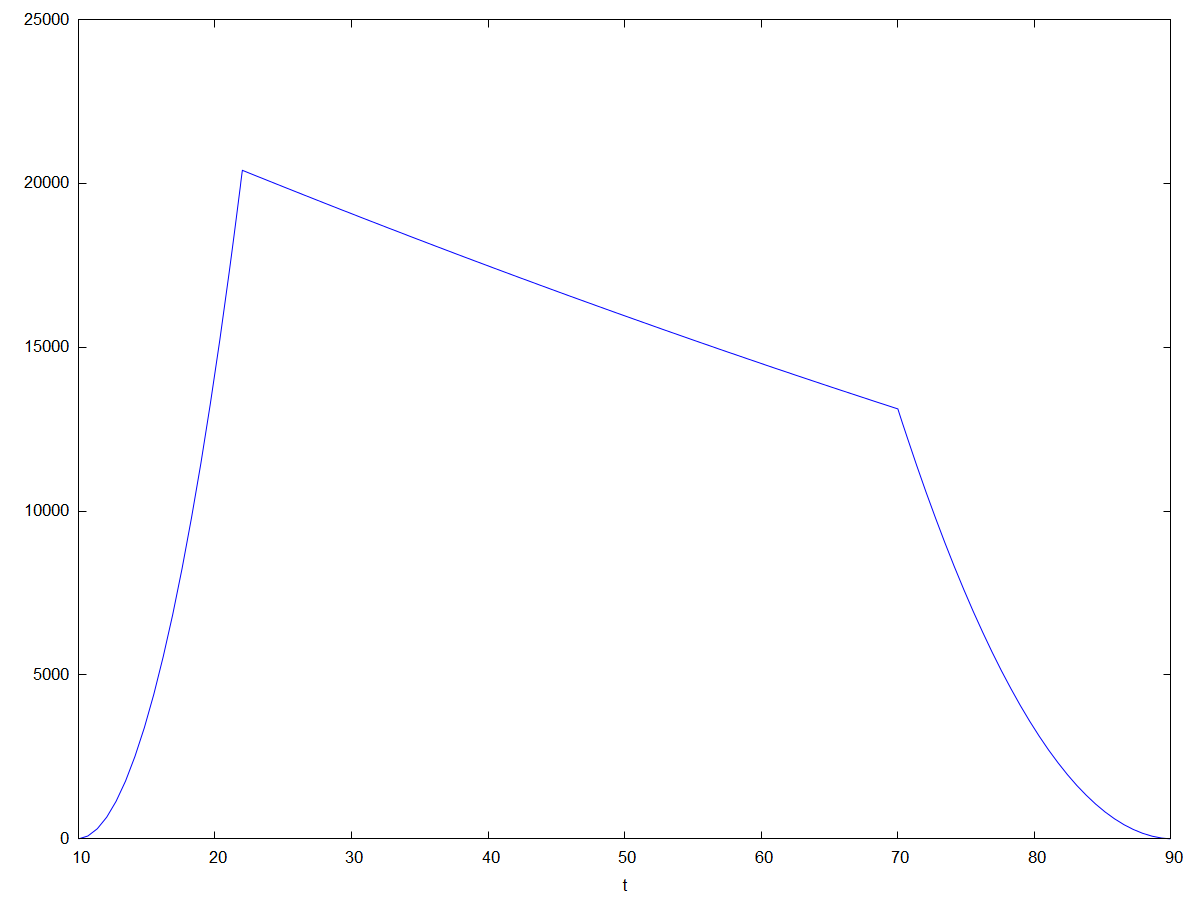


Рис. 5 График для Uvix

# Таблица идентификаторов

В таблице 1 описаны идентификаторы, используемые в программе.

Таблица 1. Таблица идентификаторов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Обозначение в задаче** | **Идентификатор** | **Назначение** | **Тип данных** |
| 1 | U1, U2, t1, t2, tнач, tкон | U1, U2, t1, t2, tn, tk | Параметры входного напряжения | Входные данные |
| 2 | a, b, c | a, b, c, | Параметры передаточной характеристики |
| 3 | - | n | Кол-во контрольных точек для расчета |
| 4 | - | ch | Выбор для меню |
| 5 | Uvx | uvx | Массив напряжений входного сигнала после обработки | Выходные данные |
| 6 | Uvix | uvix | Массив напряжений выходного сигнала после обработки |
| 7 | t | t | Массив временных промежутков для напряжений сигналов на входе и выходе |
| 8 | - | dlit | Длительность переднего импульса |
| 9 | - | i | Счётчик | Промежуточные данные |
| 10 | - | dt | Шаг изменения времени |
| 11 | - | vvod | Выбор для записи в файл |
| 12 | - | exit | Условие завершения  цикла |
| 13 | - | f | Указатель на файл  для записи результатов |
| 14 | - | N | Размерность массивов |
| 15 | - | isUvx | Проверка на вид массива для функций |
| 16 | - | Ui | Массив напряжений, который может быть Uvx или Uvix |
| 17 | - | Umax, Umin | Максимальное и минимальное значение входного напряжения |
| 18 | - | p | Погрешность |  |

# Блок-схема алгоритма (функции main)

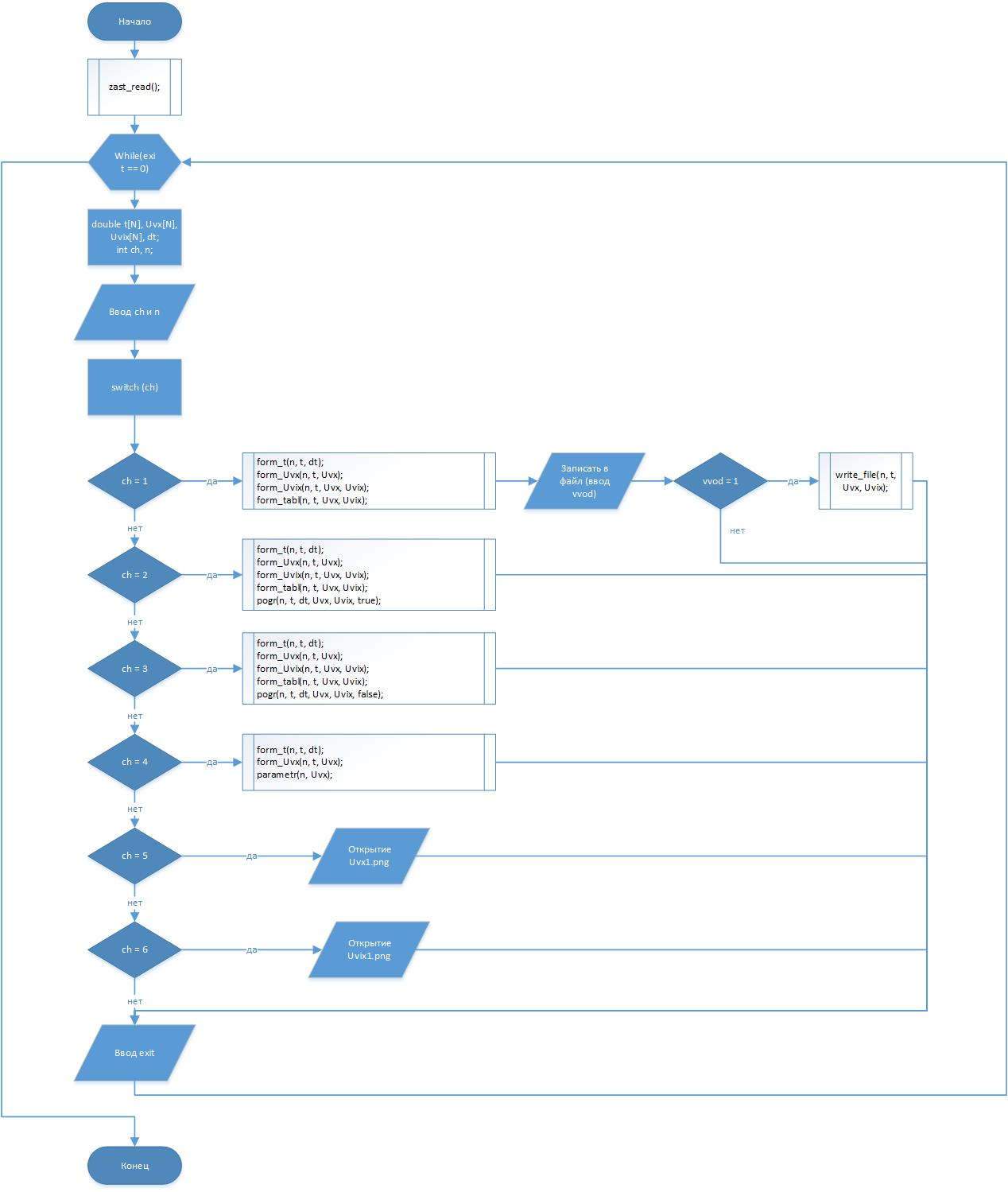


Рис. 6 Блок-схема

# Текст программы

**ElectricalCircuit-Signal-Analysis.cpp**

#include <iostream>

#include "Functions.h"

#include <Windows.h>

using namespace std;

# define N 10000

#define \_CRT\_SECURE\_NO\_DEPRECATE

int main()

{

setlocale(LC\_ALL, "RUSSIAN");

int exit = 0;

zast\_read();

while (exit == 0) {

double t[N], Uvx[N], Uvix[N], dt;

int ch;

cout << "\nВыберите задание: \n1 - контрольный расчет для n точек \n2 - расчет погрешности для Uvx \n3 - расчет погрешности для Uvix \n4 – расчет параметра с заданной точностью \n5 - открыть график Uvx \n6 - открыть график Uvix \n=> ";

cin >> ch;

int n;

if (ch < 5) {

cout << "\nВведите кол - во точек для контрольного расчета: ";

cin >> n;

}

switch (ch) {

case 1: {

form\_t(n, t, dt);

form\_Uvx(n, t, Uvx);

form\_Uvix(n, t, Uvx, Uvix);

form\_tabl(n, t, Uvx, Uvix);

int vvod;

cout << "\n\nЗаписать в файл " << n << " точек " << "(1-Да,2-Нет): ";

cin >> vvod;

if (vvod == 1)

{

write\_file(n, t, Uvx, Uvix);

}

break;

}

case 2: {

form\_t(n, t, dt);

form\_Uvx(n, t, Uvx);

form\_Uvix(n, t, Uvx, Uvix);

form\_tabl(n, t, Uvx, Uvix);

cout << endl << endl;

pogr(n, t, dt, Uvx, Uvix, true);

break;

}

case 3: {

form\_t(n, t, dt);

form\_Uvx(n, t, Uvx);

form\_Uvix(n, t, Uvx, Uvix);

form\_tabl(n, t, Uvx, Uvix);

cout << endl << endl;

pogr(n, t, dt, Uvx, Uvix, false);

break;

}

case 4: {

form\_t(n, t, dt);

form\_Uvx(n, t, Uvx);

parametr(n, Uvx);

break;

}

case 5: {

ShellExecuteA(NULL, "open", "Uvx1.png", NULL, NULL, SW\_SHOWNORMAL);

break;

}

case 6: {

ShellExecuteA(NULL, "open", "Uvix1.png", NULL, NULL, SW\_SHOWNORMAL);

break;

}

default: {

cout << "Такого задания нет! ";

break;

}

}

cout << "\nВыйти? 1-Да 0-Нет ";

cin >> exit;

}

return 0;

}

**Functions.h**

#pragma once

void form\_t(int n, double\* t, double& dt); //Заголовок функции со списком формальных параметров

void form\_Uvx(int n, double\* t, double\* Uvx); //Заголовок функции со списком формальных параметров

void form\_Uvix(int n, double\* t, double\* Uvx, double\* Uvix); //Заголовок функции со списком формальных параметров

void form\_tabl(int n, double\* t, double\* Uvx, double\* Uvix); //Заголовок функции со списком формальных параметров

void pogr(int n, double\* t, double& dt, double\* Uvx, double\* Uvix, bool isUvx);

void write\_file(int n, double\* t, double\* Uvx, double\* Uvix);

void zast\_read();

double Umax(int n, double\* Ui);

double Umin(int n, double\* Ui);

void parametr(int n, double\* Ui);

**Functions.cpp**

#include <iostream>

#include <fstream>

#include <locale>

#define \_CRT\_SECURE\_NO\_DEPRECATE

#pragma warning (disable : 4996)

using namespace std;

const int tn = 10, tk = 90;

void form\_t(int n, double\* t, double& dt)

{

dt = (tk - tn) / (n - 1);

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

t[i] = tn + i \* dt;

}

void form\_Uvx(int n, double\* t, double\* Uvx) {

int U1 = 100, U2 = 80, t1 = 22, t2 = 70;

for (int i = 0; i < n; i++) // Формирование массива Uvx

if (t[i] <= t1) Uvx[i] = ((t[i] - tn) \* (U1 - 0)) / (t1 - tn);

else if (t[i] <= t2) Uvx[i] = (((t[i] - t1) \* (U2 - U1)) / (t2 - t1)) + U1;

else if (t[i] > t2) Uvx[i] = (((t[i] - t2) \* (0 - U2)) / (tk - t2)) + U2;

}

void form\_Uvix(int n, double\* t, double\* Uvx, double\* Uvix) {

int a = 2, b = 4, c = -1;

for (int i = 0; i < n; i++) // Формирование массива Uvix

Uvix[i] = a \* Uvx[i] \* Uvx[i] + b \* Uvx[i] + c;

}

void form\_tabl(int n, double\* t, double\* Uvx, double\* Uvix) {

cout << " № t Uvx Uvix " << endl;

for (int i = 0; i < n; i++) // Вывод данных в виде таблицы

printf("\n % 3d % 6.3f % 6.3f % 7.3f", i, t[i], Uvx[i], Uvix[i]);

}

void write\_file(int n, double\* t, double\* Uvx, double\* Uvix) {

std::ofstream file1("massiv\_t.txt");

std::ofstream file2("massiv\_Uvx.txt");

std::ofstream file3("massiv\_Uvix.txt");

std::locale loc(std::locale::classic(), new std::numpunct<char>('.'));

file1.imbue(loc); // установка локали для файла

file2.imbue(loc); // установка локали для файла

file3.imbue(loc); // установка локали для файла

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

{

file1 << t[i] << std::endl;

file2 << Uvx[i] << std::endl;

file3 << Uvix[i] << std::endl;

}

file1.close();

file2.close();

file3.close();

printf("Запись прошла успешно!\n");

}

void zast\_read() {

FILE\* f = fopen("zast.txt", "r");// Открытие файла для чтения

if (f == NULL)

{

printf("Ошибка при открытии файла!\n");

return;

}

char ch;

while (!feof(f)) // Считывание данных из файла до тех пор,

{

fscanf\_s(f, "%c", &ch); // пока не найдено значение конца файла

printf("%c", ch);

}

fclose(f);

}

double Umin(int n, double\* Ui) {

double min = Ui[0];

int min\_index = 0;

int vvod = 0;

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

{

if (min > Ui[i])

{

min = Ui[i];

min\_index = i;

}

}

return Ui[min\_index];

}

double Umax(int n, double\* Ui) {

double max = Ui[0];

int max\_index = 0;

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

{

if (max < Ui[i])

{

max = Ui[i];

max\_index = i;

}

}

return Ui[max\_index];

}

void parametr(int n, double\* Ui) {

cout << "\n\nРасчет длительности переднего импульса...";

double umax = Umax(n, Ui);

double umin = Umin(n, Ui);

double dt = (tk - tn) / (n - 1);

double dlit = 0;

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

if (Ui[i] >= umin + 0.5 \* (umax - umin)) dlit += dt;

double U1 = umin + 0.9f \* (umax - umin);

double U2 = umin + 0.1f \* (umax - umin);

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

if (Ui[i] > U1 && Ui[i] < U2 && Ui[i + 1]>Ui[i]) dlit += dt;

cout << "\n\nДлительность переднего импульса равна: " << dlit;

cout << endl;

}

void pogr(int n, double\* t, double& dt, double\* Uvx, double\* Uvix, bool isUvx) {

double p = 1; // текущая погрешность

float eps = 0.001; // заданная погрешность

double par = 1000; // начальное значение параметра (очень большое число)

double par1 = 0;

string text = "";

while (p > eps) {

form\_t(n, t, dt);

form\_Uvx(n, t, Uvx);

form\_Uvix(n, t, Uvx, Uvix);

if (isUvx) {

par1 = Umax(n, Uvx);

text = " Uvx[max] = ";

}

else {

par1 = Umax(n, Uvix);

text = " Uvix[max] = ";

}

p = fabs(par - par1) / par1;

cout << "n = " << n << text << par1 << " Погрешность = " << p << endl;

par = par1;

n = 2 \* n;

}

cout << endl;

int vvod;

cout << "Записать в файл " << n / 2 << " точек " << "(1-Да,2-Нет): ";

cin >> vvod;

if (vvod == 1)

{

write\_file(n / 2, t, Uvx, Uvix);

}

}

# Графики (обработка результатов)

С помощью программы wxMaxima нарисуем графики по точкам, вычисленных программой.

**Расчет в среде wxMaxima:**

t:read\_list("D:/GitHub/ElectricalCircuit-Signal-Analysis/ElectricalCircuit-Signal-Analysis/massiv\_t.txt"); /\*команда чтения данных из файла \*/

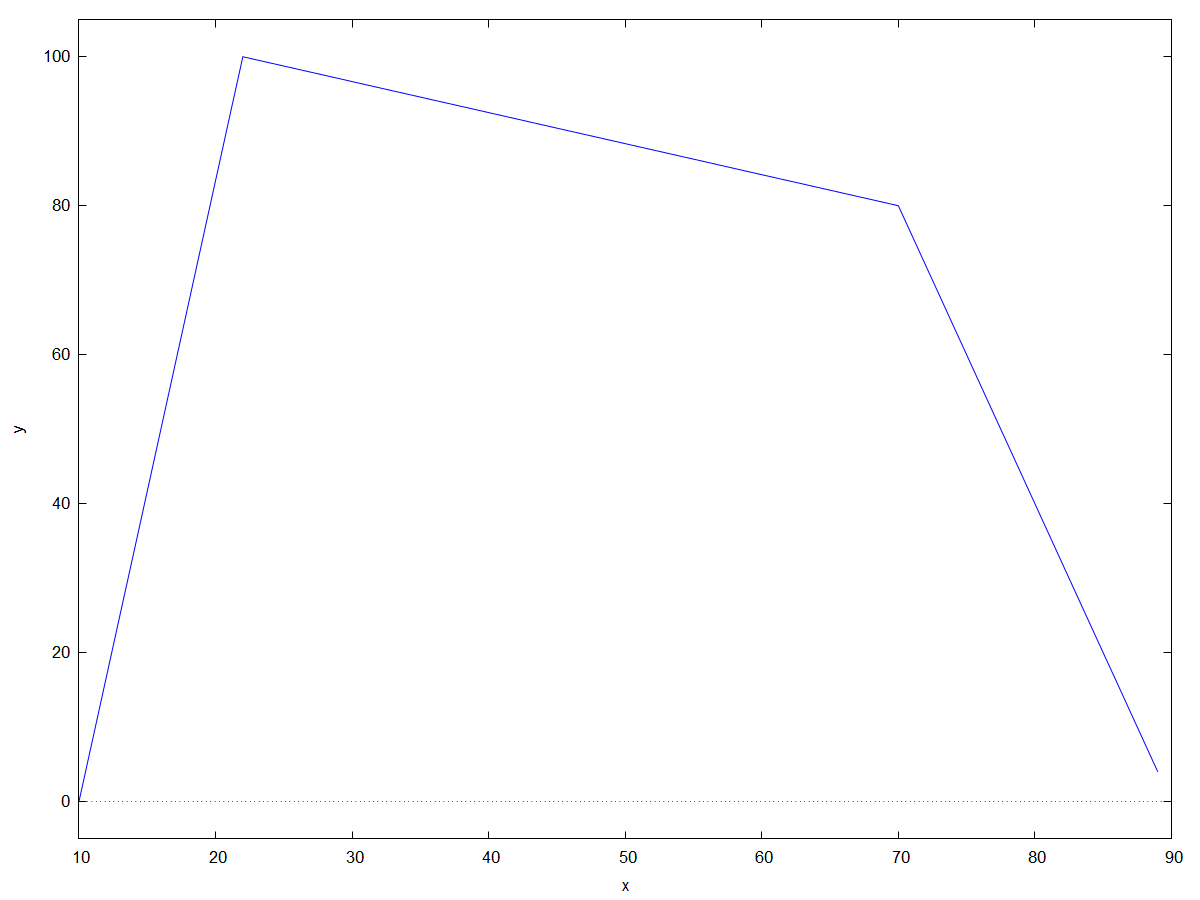
Uvx:read\_list("D:/GitHub/ElectricalCircuit-Signal-Analysis/ElectricalCircuit-Signal-Analysis/massiv\_Uvx.txt");

Uvix:read\_list("D:/GitHub/ElectricalCircuit-Signal-Analysis/ElectricalCircuit-Signal-Analysis/massiv\_Uvix.txt");

wxplot2d([['discrete, t, Uvx]],[gnuplot\_preamble, "set grid;"]); /\*построение графика\*/

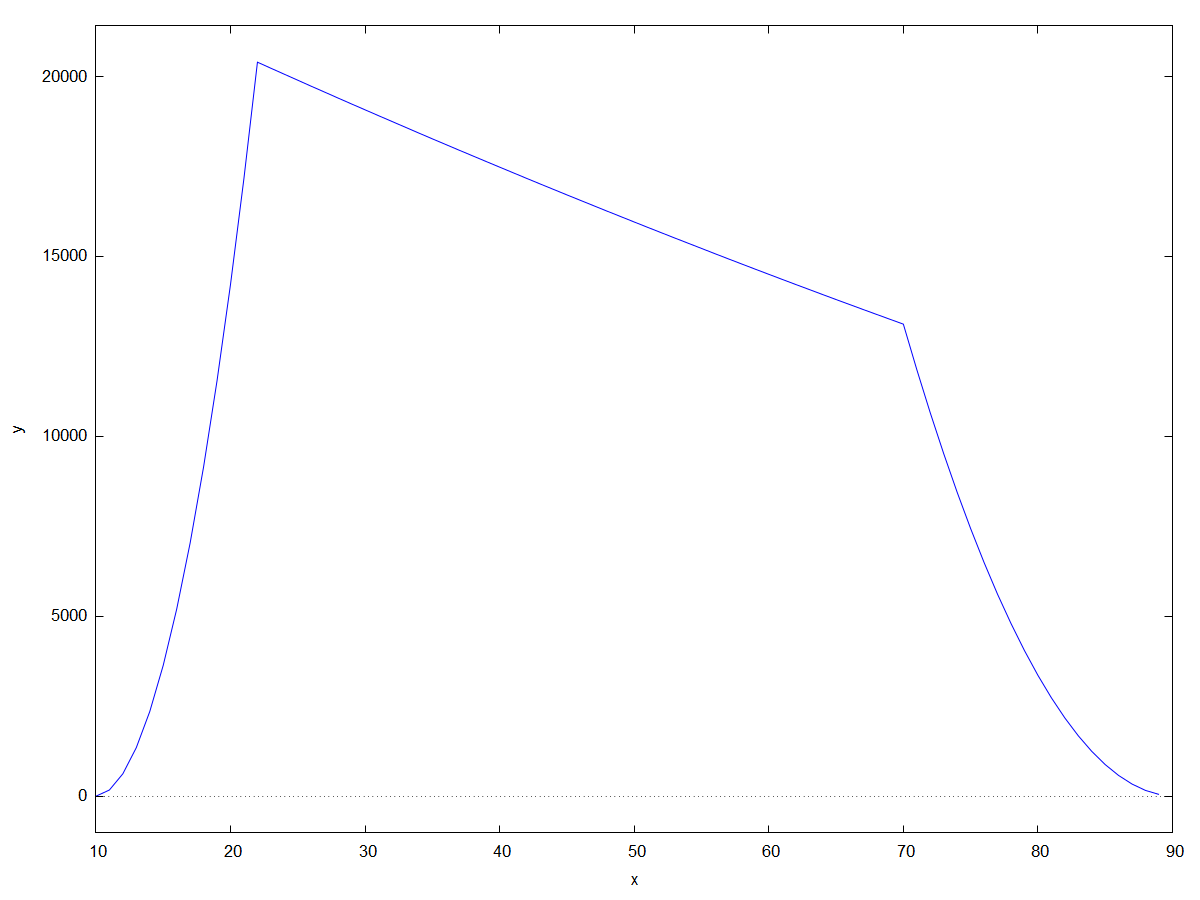
wxplot2d([['discrete, t, Uvix]],[gnuplot\_preamble, "set grid;"]);

График для входного напряжения, полученный из точек, рассчитанных программой, представлен на рис. 7:



*Рис. 7*

График для выходного напряжения, полученный из точек, рассчитанных программой, представлен на рис. 8:



*Рис. 8*

# Вывод

В ходе выполнения курсовой работы были произведены расчеты в среде wxMaxima и построены графики для входного и выходного сигнала согласно варианту. Результатом работы является программа на языке C++, которая рассчитывает значения входного и выходного сигнала. Графики, построенные по точкам, рассчитанных программой и расчитанных в среде wxMaxima совпадают.

Программа может быть использована в научных и инженерных исследованиях, а также в производственной деятельности для контроля качества электронных устройств и оборудования.

# Список литературы

1. Лекции по дисциплине “Программирование”;
2. Л.Н. Бережной И.О. Воронцова Д.В. Окунева. Программирование. Методические указания к курсовой работе. Санкт-Петербург: СПбГУТ, 2023