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 – 7 вариант

Входной сигнал – 7 вариант

Выполнил студент(ка):

Дымнич К.А. \_\_\_\_\_\_\_\_

*(Ф.И.О., № группы) (подпись)*

Дата выполнения:

«25» Апрель 2025г

Проверил:

Хазиев Н.Н. \_\_\_\_\_\_\_\_

*(Ф.И.О.) (подпись)*

Санкт-Петербург

2025

Оглавление

[Задание к курсовой работе 3](#_Toc197952309)

[Использование технических средств 4](#_Toc197952310)

[Вызов и загрузка: 4](#_Toc197952311)

[Входные данные: 4](#_Toc197952312)

[Выходные данные: 4](#_Toc197952313)

[Контрольный расчёт 5](#_Toc197952314)

[Таблица идентификаторов 6](#_Toc197952315)

[Описание функций 7](#_Toc197952316)

[Описание логической структуры 9](#_Toc197952317)

[Код программы 13](#_Toc197952318)

[Си: 13](#_Toc197952319)

[Bash: 19](#_Toc197952320)

[Графики (обработка полученных результатов, выводы) 33](#_Toc197952321)

[Заключение 35](#_Toc197952322)

[Список используемой литературы 36](#_Toc197952323)

# Задание к курсовой работе

Работа посвящена решению задач машинного анализа электрических цепей.

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%. Варианты параметров входного сигнала (код А) и передаточной характеристики (код Б) электрической цепи приведены в приложении. Номер варианта определяется преподавателем индивидуально для каждого студента.

|  |  |
| --- | --- |
| Входной сигнал | Рабочий набор |
|  |  |

|  |  |
| --- | --- |
| Выходной сигнал | Рабочий набор |
|  |  |

В ходе работы 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 (электронные таблицы), построить графики зависимости напряжений входных и выходных сигналов от времени.
* Объединить программу на Си и Wxmaxima (LibraOffice.Calc), вызов отчета с помощью скрипта на Bash.

# Аннотация

Данная работа посвящена разработке программного обеспечения для вычисления и анализа параметров функции на основе численных методов. Основной целью является создание программы, способной выполнять прямой расчет значений функции, а также приближенный расчет с заданной точностью. Программа реализована на языке программирования C и запускается через команду make run, при этом все необходимые данные запрашиваются изнутри программы.

В работе рассматриваются два режима работы программы. В первом режиме осуществляется контрольный расчет параметров функции на основе заданных коэффициентов и диапазона времени. Во втором режиме выполняется приближенный расчет параметра с учетом заданной пользователем погрешности. Результаты вычислений выводятся в табличной форме.

Программное обеспечение также включает функции для формирования массивов времени и значений функции, а также для автоматического вывода и сохранения данных. Данная работа может быть полезна в учебных целях, а также для решения прикладных задач, связанных с анализом временных характеристик функций.

# ****Использование технических средств****

Для выполнения программы применяются технические средства, обеспечивающие ее разработку, тестирование и эксплуатацию. В процессе работы программы используется компьютерная система с операционной системой на базе Linux, необходимая для запуска Bash-скриптов и выполнения компилируемого на языке Си приложения. Компиляция программы осуществляется с помощью компилятора GCC, а визуализация данных выполняется посредством программного обеспечения Maxima для построения графиков. Управление программой и взаимодействие с пользователем осуществляется через командную строку, обеспечивая ввод параметров и отображение результатов работы программы.

# Вызов и загрузка:

Программа запускается командой make run. После вызова приложение автоматически загружает необходимые модули и конфигурации, выполняя начальную инициализацию без необходимости дополнительных действий со стороны пользователя.

# Входные данные:

Программа самостоятельно запрашивает у пользователя параметры расчета через интерфейс, предоставляя инструкции по вводу значений, таких как количество точек разбиения, погрешность вычислений и другие параметры, необходимые для выполнения поставленной задачи.

# Выходные данные:

После выполнения расчетов программа выводит результаты в виде таблицы значений, содержащей временные точки и рассчитанные значения переменных. Дополнительно результаты могут быть сохранены в файлы или визуализированы в виде графиков.

# Контрольный расчёт

|  |  |
| --- | --- |
| Контрольный расчет для n точек | Параметры |
|  | При кол-во контрольных точек n=25 |

|  |  |
| --- | --- |
| Расчёт параметра с заданной точностью | Параметры |
|  | n = 10, eps = 1% |

# Таблица идентификаторов

|  |  |
| --- | --- |
| a0 | Постоянная составляющая напряжения |
| a1 | Коэффициент линейного изменения напряжения во времени |
| a2 | Коэффициент квадратичного изменения напряжения во времени |
| tn | Время, до которого рассматривается изменение напряжения (начальное время) |
| tk | Время, до которого рассматривается изменение напряжения (конечное время) |
| a | Постоянное значение выходного напряжения при малых входных напряжениях |
| b | Коэффициент квадратичной зависимости выходного напряжения |
| Uvx1 | Пороговое значение входного напряжения, при котором меняется зависимость |
| n | Количество элементов в массиве |
| eps | Погрешность |
| p | Параметр |
| t | Массив времени |
| Uvx | Массив входного напряжение |
| Uvix | Массив выходного напряжение |

# Описание функций

|  |  |
| --- | --- |
| Функция | Описание |
| main | Основная точка входа в программу на языке C. Запускает функцию run\_app. |
| run\_app | Управляющая функция приложения на C. Выбирает действие (прямой или приближённый расчёт). |
| form\_time | Формирует массив времён t по шагу времени. |
| form\_Uvx | Вычисляет массив напряжений Uvx на основе временных точек. |
| form\_Uvix | Вычисляет массив напряжений Uvix по кусочной линейной аппроксимации. |
| parametr | Вычисляет среднее значение сигнала U за промежуток времени. |
| form\_tabl1 | Выводит таблицу значений времён и напряжений (t, Uvx, Uvix). |
| control\_calc | Основная функция расчёта (вариант 1). Формирует массивы и выводит таблицу. |
| approx\_value | Основная функция приближённого расчёта (вариант 2) с заданной точностью. |
| menu.sh | Скрипт bash. Отображает меню программы и управляет пользователем. |
| out\_zast | Выводит заставку программы. |
| out\_menu | Основное меню программы. Обрабатывает пользовательский ввод. |
| start | Запускает программу, очищает массивы данных. |
| out\_file | Записывает массивы t, Uvx, Uvix в текстовые файлы и запускает построение графиков. |
| style | Выводит текст с заданным стилем (цветом). |
| clear\_line | Очищает текущую строку терминала. |
| is\_number | Проверяет, является ли введённое значение числом. |
| out\_info\_pr | Выводит текстовые файлы с пояснениями к параметрам программы. |
| pg1 | Запускает бинарное приложение (вариант 1) и выводит результаты в таблице. |
| pg2 | Запускает бинарное приложение (вариант 2) и выводит таблицу с погрешностями. |

# Описание логической структуры

Программа состоит из двух основных частей:

1. **С++ приложение (src)**, которое выполняет математические расчеты и выводит таблицы значений.
2. **Bash-скрипты (scripts)**, которые управляют запуском приложения, меню и записью результатов.

**1. Приложение на C (src)**

**Основные модули:**

* main.c — Точка входа. В зависимости от аргументов командной строки вызывает одну из двух функций:
  + control\_calc() — Прямой расчет.
  + approx\_value() — Приближенный расчет с заданной точностью.
* globals.h — Заголовочный файл с глобальной структурой AppParams, содержащей параметры работы программы.
* app.h — Заголовочный файл с прототипом функции run\_app().
* funct.h — Заголовочный файл с прототипами функций, выполняющих математические операции:
  + form\_time() — Формирование массива времени.
  + form\_Uvx() — Расчет промежуточных значений.
  + form\_Uvix() — Расчет конечных значений по заданному закону.
  + form\_tabl1() — Вывод таблицы результатов.
  + parametr() — Расчет среднего значения.

**Логика работы приложения:**

1. **Обработка аргументов командной строки:**
   * Первый аргумент (arg[1]) — режим работы (1 — прямой расчет, 2 — приближенный).
   * Второй аргумент (arg[2]) — количество точек (n).
   * Третий аргумент (arg[3]) — точность (eps) (только для приближенного метода).
2. **Инициализация структуры параметров (AppParams), которая содержит:**
   * Параметры уравнения (a0, a1, a2).
   * Начальное (tn) и конечное (tk) время.
   * Пороговое значение напряжения (Uvx1).
   * Количество точек (n) и точность (eps).
3. **В зависимости от режима работы вызывается одна из функций:**
   * control\_calc():
     + Вычисляет значения времени, промежуточных (Uvx) и конечных (Uvix) значений.
     + Выводит результаты в виде таблицы.
   * approx\_value():
     + Вычисляет параметр (parametr) до достижения заданной точности (eps).
     + Выводит таблицу с шагами расчета и точностью.

**2. Bash-скрипты (scripts)**

**Основные компоненты:**

* start.sh — Главный скрипт запуска программы.
* functions.sh — Функции для форматирования текста и проверки ввода.
* variables.sh — Глобальные переменные (цвета текста, меню).
* file\_output.sh — Запись результатов в файлы.
* output\_data.sh — Построение графиков с помощью Maxima.
* p1.sh и p2.sh — Обработчики для двух режимов работы программы.

**Логика работы скриптов:**

1. **Старт программы (start()):**
   * Выводит меню с вариантами действий:
     + 1 — Контрольный расчет для n точек.
     + 2 — Расчет параметра с заданной точностью.
     + 3 — Запись данных в файлы.
     + p — Пояснение параметров.
     + q — Выход из программы.
2. **Выбор действия пользователем:**
   * Ввод количества точек (n) и точности (eps).
   * Вызов соответствующей функции:
     + pg1() — Запуск приложения с режимом 1 (прямой расчет).
     + pg2() — Запуск приложения с режимом 2 (приближенный расчет).
3. **Обработка вывода приложения:**
   * В режиме 1 (pg1) программа считывает три строки:
     + Временные точки (t).
     + Промежуточные значения (Uvx).
     + Конечные значения (Uvix).
   * В режиме 2 (pg2) программа считывает строки с шагами расчета и погрешностями.
4. **Запись результатов в файлы (out\_file()):**
   * Три файла данных:
     + massiv\_t.txt — Временные точки.
     + massiv\_Uvx.txt — Промежуточные значения.
     + massiv\_Uvix.txt — Конечные значения.
   * Запуск Maxima для построения графиков.

**3. Общая схема работы**

1. Пользователь запускает программу (start()).
2. Программа выводит меню.
3. Пользователь выбирает один из двух режимов:
   * Прямой расчет (pg1()).
   * Приближенный расчет (pg2()).
4. Введенные параметры передаются приложению на C.
5. Приложение выполняет расчеты и возвращает результаты:
   * Временные точки (t).
   * Промежуточные значения (Uvx).
   * Конечные значения (Uvix).
6. Результаты сохраняются в текстовые файлы.
7. Пользователь может построить графики или выйти из программы.

# Код программы

## Си:

#include "app.h"

int main(int count, char\* arg[]) {

run\_app(count, arg);

return 0;

}

#include <stdlib.h> // Подключение стандартной библиотеки (atoi, atof)

#include "globals.h" // Заголовочный файл с глобальными переменными или структурами

#include "rpzt.h" // Заголовочный файл, предположительно с функцией control\_calc()

#include "krnt.h" // Заголовочный файл, предположительно с функцией approx\_value()

// Главная управляющая функция приложения

void run\_app(int count, char\* arg[]) {

// Инициализация структуры параметров приложения

struct AppParams ap\_pr = {

.a0 = 2, // значение a0

.a1 = 5, // значение a1

.a2 = 0.3, // значение a2

.tn = 1, // Начальное время tn

.tk = 10, // Конечное время tk

.Uvx1 = 10, // Значение напряжения в первой точке

.n = atoi(arg[2]) // Количество точек, переданное через аргументы

};

// Выбор действия по номеру варианта, переданного в аргументах

switch (atoi(arg[1])) {

case 1:

// Вариант 1: выполнить прямой расчёт

control\_calc(ap\_pr);

break;

case 2:

// Вариант 2: установить погрешность и выполнить приближённый расчёт

ap\_pr.eps = atof(arg[3]) / 100; // Перевод из процентов в дробное значение

approx\_value(ap\_pr);

break;

}

}

#include <stdio.h> // Для функций ввода-вывода (printf)

#include <math.h> // Для математических операций

#include "globals.h" // Заголовочный файл с глобальной структурой AppParams

#include "funct.h" // Заголовочный файл, содержащий прототипы текущих функций

// Формирование массива времён t по шагу dt

void form\_time(struct AppParams ap\_pr, float\* t) {

float dt = (ap\_pr.tk - ap\_pr.tn) / (ap\_pr.n - 1); // Шаг между точками времени

for (int i = 0; i < ap\_pr.n; i++) {

t[i] = ap\_pr.tn + i \* dt; // t[i] = начальное + шаг \* номер

}

}

// Формирование массива значений Uvx по заданному закону

void form\_Uvx(struct AppParams ap\_pr, float\* t, float\* Uvx) {

for (int i = 0; i < ap\_pr.n; i++) {

Uvx[i] = ap\_pr.a0 + ap\_pr.a1\*t[i] + ap\_pr.a2\*pow(t[i],2);

}

}

// Формирование массива значений Uvix на основе Uvx по кусочной линейной аппроксимации

void form\_Uvix(struct AppParams ap\_pr, float\* Uvx, float\* Uvix) {

for (int i = 0; i < ap\_pr.n; i++) {

if (Uvx[i] <= ap\_pr.Uvx1) Uvix[i] = 5;

else Uvix[i] = 0.05\*pow(Uvx[i], 2);

}

}

// Функция вычисляет продолжительность (в секундах), когда сигнал превышает порог

float parametr(int n, float sum, float \*U, float \*t) {

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

sum += U[i];

}

return sum / n;

}

// Вывод таблицы значений t, Uvx, Uvix в три строки

void form\_tabl1(int n, float\* t, float\* Uvx, float\* Uvix) {

for (int i = 0; i < n \* 3; i++) {

if (i < n) {

if (i < (n - 1)) printf("%.3g ", t[i]);

else printf("%.3g\n", t[i]);

} else if (i < n \* 2) {

if (i < (n \* 2 - 1)) printf("%.3g ", Uvx[i - n]);

else printf("%.3g\n", Uvx[i - n]);

} else {

if (i < (n \* 3 - 1)) printf("%.3g ", Uvix[i - n \* 2]);

else printf("%.3g\n", Uvix[i - n \* 2]);

}

}

}

#include "globals.h" // Заголовочный файл с определением структуры AppParams и констант

#include "krnt.h" // Заголовок с прототипом control\_calc

#include "funct.h" // Прототипы функций для расчёта и формирования таблиц

// Основная управляющая функция для варианта 1 — расчёт и вывод таблицы

void control\_calc(struct AppParams ap\_pr) {

float t[N], Uvx[N], Uvix[N]; // Массивы для времён, промежуточного и результирующего напряжения

form\_time(ap\_pr, t); // Заполнение массива времени

form\_Uvx(ap\_pr, t, Uvx); // Расчёт промежуточного напряжения Uvx

form\_Uvix(ap\_pr, Uvx, Uvix); // Расчёт результирующего напряжения Uvix

form\_tabl1(ap\_pr.n, t, Uvx, Uvix); // Вывод таблицы значений

}

#include "stdio.h" // Для printf

#include "globals.h" // Структура AppParams и глобальные константы

#include "rpzt.h" // Прототип approx\_value

#include "funct.h" // Прототипы функций расчёта

#include "math.h" // Для fabs

// Функция приближённого расчёта значения параметра с заданной точностью

void approx\_value(struct AppParams ap\_pr) {

float t[N], Uvx[N], Uvix[N];

float p = 1;

float par = 1e10;

float par1 = 0;

printf("n parametr pogrechnost\n");

while (p > ap\_pr.eps && ap\_pr.n < N) {

form\_time(ap\_pr, t);

form\_Uvx(ap\_pr, t, Uvx);

form\_Uvix(ap\_pr, Uvx, Uvix);

par1 = parametr(ap\_pr.n, 0, Uvix, t);

p = fabs(par - par1) / fabs(par1);

if (p > 1) p = 1;

printf("%d %.5f %.5f\n", ap\_pr.n, par1, p);

par = par1;

ap\_pr.n = 2 \* ap\_pr.n;

}

}

#include "app.h"

int main(int count, char\* arg[]) {

run\_app(count, arg);

return 0;

}

ghost@DigmaEVEP4851:~/unvr/pr\_xz/kursach/program$ cat src/include/\*

#ifndef APP\_H

#define APP\_H

void run\_app(int count, char\* arg[]);

#endif

#ifndef FUNCT\_H

#define FUNCT\_H

void form\_time(struct AppParams ap\_pr, float\* t);

void form\_Uvx(struct AppParams ap\_pr, float\* Uvx, float\* t);

void form\_Uvix(struct AppParams ap\_pr, float\* Uvx, float\* Uvix);

void form\_tabl1(int n, float \*t, float \*Uvx, float \*Uvix);

float parametr(int n, float sum, float \*U, float \*t);

#endif

#ifndef GLOBALS\_H

#define GLOBALS\_H

// Максимальное количество точек (для массивов)

#define N 10000

// Структура параметров приложения

struct AppParams {

int n; // Количество точек разбиения

float eps; // Допустимая погрешность (для приближённого метода)

float a0, a1, a2; // Коэффициенты линейных участков функции Uvx

float tn, tk, t1; // Начальное, конечное время и момент излома

float Uvx1; // Пороговые значения для функции Uvix

};

#endif

#ifndef KRNT\_H

#define KRNT\_H

void control\_calc(struct AppParams ap\_pr);

#endif

#ifndef RPZT\_H

#define RPZT\_H

void approx\_value(struct AppParams ap\_pr);

#endif

## Bash:

#!/bin/bash

clear # Очистка экрана

# Подключение вспомогательных скриптов с переменными и функциями

. ./scripts/helpers/variables.sh --source-only

. ./scripts/helpers/functions.sh --source-only

# Подключение вспомогательных частей программы

. ./scripts/helpers/p1.sh --source-only

. ./scripts/helpers/p2.sh --source-only

. ./scripts/helpers/file\_output.sh --source-only

. ./scripts/helpers/output\_data.sh --source-only

export LC\_NUMERIC=C # Установка десятичного разделителя как точка

N=10000 # Максимальное количество точек

# Функция вывода заставки

out\_zast(){

while read -r line; do

style "$line" $yellow # Цветной вывод строки

done < $file\_name\_zast # Чтение строк из файла

printf "\n\n"

}

# Функция отображения основного меню

out\_menu() {

while true; do

style "Меню программы:" $green

for indx in "${!variant\_menu[@]}"; do

if [ "$indx" != "2" ]; then

style "${variant\_menu[${indx}]}" $yellow

elif [[ "$indx" == "2" && "${#t[@]}" -gt "0" ]]; then

style "${variant\_menu[${indx}]} <-----" $i\_yellow

fi

done

echo

while true; do

# Определение доступных пунктов меню

if [ "${#t[@]}" -gt "0" ]; then con\_vr=3

else con\_vr=2; fi

style "Выберите действие 1-${con\_vr} и p или q для выхода " $blue n

read -rsn1 key # Чтение одного символа

printf "\n"

case $key in

1|2)

clear

style "Ведите n точек:" $yellow

style "Диапазон n: [2;${N}]" $yellow

while true; do

is\_number " Ведите n: " '^[0-9]+$' # Проверка ввода целого числа

if [ "$num" -gt "1" ]; then

if [ "$num" -le "$N" ]; then break

else

clear\_line

style " Error: Число ($num) > $N" $red

fi

else

clear\_line

style " Error: Число ($num) < 2" $red

fi

done

n=$num # Сохранение введённого значения

;;&

2)

style "Ведите погрешность eps:" $yellow

style "Диапазон eps: [0.001; 99.99]%" $yellow

while true; do

is\_number " Введите eps: " '^[0-9]\*\.?[0-9]+$' # Проверка вещественного числа

# Проверка: num > 0.0009

valid\_min=$(echo "$num > 0.0009" | bc -l)

# Проверка: num < 99.99

valid\_max=$(echo "$num < 99.99" | bc -l)

if [[ "$valid\_min" -eq 1 && "$valid\_max" -eq 1 ]]; then

break

elif [[ "$valid\_max" -ne 1 ]]; then

clear\_line

style " Ошибка: число ($num) > 99.99" $red

else

clear\_line

style " Ошибка: число ($num) < 0.0009" $red

fi

done

eps=$num # Сохранение значения

;;&

[1-2])

clear

style "Данне успешно переданны в программу!" $green

style "Данные из программы успешно считанны!" $green

pg${key} $key # Вызов функции pg1 или pg2 в зависимости от выбора

;;&

3)

if [ "$con\_vr" == "3" ];then

out\_file # Запись результатов в файл

else

clear\_line

style "Erorr: Не верное значение ($key) не входит в промежуток [1;$con\_vr]!" $red

break

fi

;;&

p)

out\_info\_pr

;;&

[1-$con\_vr]|p)

clear

out\_zast # Повторный вывод заставки

break

;;

q)

return # Завершение работы

;;

\*)

clear\_line

style "Erorr: Не верное значение ($key) не входит в промежуток [1;$con\_vr] и p!" $red

;;

esac

done

done

}

# Функция запуска программы

start() {

clear

inp\_data=() # Массив входных данных

out\_data=() # Массив выходных данных

out\_zast # Отображение заставки

out\_menu # Запуск главного меню

}

start # Старт программы

clear

exit # Завершение

#!/bin/sh

# Функция out\_file — записывает массивы t, Uvx и Uvix в отдельные текстовые файлы

# и запускает скрипт для построения графиков с помощью Maxima

out\_file() {

clear

style "Происходит запись в файл!" $green

var\_file=( # Массив с путями к выходным файлам

"./data/massiv\_t.txt"

"./data/massiv\_Uvx.txt"

"./data/massiv\_Uvix.txt"

)

{

for i in "${!t[@]}"; do

echo "${t[$i]}"

done

} > "${var\_file[0]}" &

{

for i in "${!Uvx[@]}"; do

echo "${Uvx[$i]}"

done

} > "${var\_file[1]}" &

{

for i in "${!Uvix[@]}"; do

echo "${Uvix[$i]}"

done

} > "${var\_file[2]}" &

clear

style "Данные успешно записанны в файл!" $green

style "Происходит генерация графиков пожалуйста подождите!" $green

# Запуск Maxima-скрипта для построения графиков

maxima -b scripts/Wxmax\_scr/make\_graphs.mac > /dev/null 2>&1 &

clear

style "Графики успешно нарисованы!" $yellow

style "Вывести открыть графики ? (y/n)" $blue n

read -rsn1 nn

if [ "$nn" == "y" ]; then

style "\nЗакройте окно с графиками для продолжения!" $yellow

eog data/graphs/graph\_Uvx.png > /dev/null 2>&1 # Открытие изображения через eog

fi

}

#!/bash/sh

style() {

if [ "$2" == "void" ];then

style\_text=""

local nc=""

else

style\_text=""

fi

if [ -z "$3" ];then

echo -e "${style\_text}${1}${nc}"

else

echo -ne "${style\_text}${1}${nc}"

fi

}

clear\_line() {

echo -ne '\e[A\e[K'

echo -ne "\007"

}

is\_number() {

re="$2"

num=0

style "$1" $blue n

while true

do

read num

if [[ $num =~ $re ]]; then break;fi

clear\_line

style " ОШИБКА: '$num'-не является целым числом" $red

style "$1" $blue n

done

}

#!/bin/bash

. ./scripts/helpers/functions.sh --source-only

. ./scripts/helpers/variables.sh --source-only

out\_info\_pr() {

out\_file\_name=(

"./config/explantion\_krnt.txt"

"./config/explantion\_rpzt.txt"

)

clear

for file\_name in "${out\_file\_name[@]}";do

while IFS= read -r line;do

style "$line" $yellow

done < "$file\_name"

if [ "$file\_name" != "${out\_file\_name[-1]}" ];then

style "Нажмите enter чтоб перелестнуть страницу!" $blue n

else

style "Нажмите enter чтоб закончить просмотр!" $blue n

fi

read

clear

done

}

#!/bin/sh

# Функция pg1 — вызывает бинарный файл, считывает и обрабатывает его вывод

pg1() {

out\_data=() # Очистка массива выходных данных

inp\_data=("$1 $n") # Формирование аргументов для вызова бинарного приложения

t=() # Массив временных точек

Uvx=() # Массив значений Uvx

Uvix=() # Массив значений Uvix

i=0 # Счётчик строк

n\_n=$n

# Чтение вывода программы построчно

while read -r line; do

case $i in

[0-2])

read -a lin <<<"$line" # Разбивает строку в массив

;;& # Продолжает выполнение следующего условия case

0)

t=("${lin[@]}") # Первая строка — массив t

;;

1)

Uvx=("${lin[@]}") # Вторая строка — массив Uvx

;;

2)

Uvix=("${lin[@]}") # Третья строка — массив Uvix

;;

esac

let "i+=1" # Увеличение счётчика

done <<< "$(./bin/myapp ${inp\_data[@]})" # Вызов внешней программы и обработка её вывода

style "\nРезультат программы: " $yellow

read -a header <<< "${out\_data[0]}" # Чтение первой строки как заголовок (не используется далее)

# Печать заголовка таблицы в консоль

printf "\n %-7s %8s %10s %9s\n" " №" "t" "Uvx" "Uvix"

# Запись заголовка таблицы в файл

printf "%-4s %7s %9s %8s\n" "№" "t" "Uvx" "Uvix" > "data/tabls/table\_p1.txt"

# Печать и запись каждой строки таблицы

for i in "${!t[@]}"; do

printf " %5d %9.1f %9.1f %9.1f\n" \

"$((i+1))" "${t[$i]}" "${Uvx[$i]}" "${Uvix[$i]}"

printf "%5d %9.1f %9.1f %9.1f\n" \

"$((i+1))" "${t[$i]}" "${Uvx[$i]}" "${Uvix[$i]}" >> "data/tabls/table\_p1.txt"

done

style "\n-> enter для окончания просмотра" $yellow n

read

clear # Очистка экрана

}

#!/bin/sh

# Функция pg2 — вызывает бинарный файл, считывает вывод и выводит табличные данные с погрешностью

pg2() {

inp\_data=("$1 $n $eps") # Формирование аргументов: номер варианта, количество точек, погрешность

out\_data=() # Очистка массива выходных данных

# Запуск бинарного приложения и построчное считывание вывода

while read -r line; do

out\_data+=("$line") # Добавление каждой строки в массив

done <<< "$(./bin/myapp ${inp\_data[@]})"

style "Результат программы: " $yellow # Заголовок результата

# Чтение заголовка таблицы

read -a header <<< "${out\_data[0]}"

printf "\n %7s %12s %14s\n" " ${header[0]}" "${header[1]}" "${header[2]}"

printf "%7s %12s %14s\n" "${header[0]}" "${header[1]}" "${header[2]}" > "data/tabls/table\_p2.txt"

# Построчная обработка данных таблицы (начиная со второй строки)

while read -a arr; do

num=${arr[2]} # Извлечение значения погрешности

num=$(awk "BEGIN { print ${arr[2]} \* 100 }")

# num=$(echo "${arr[2]} \* 100" | bc -l) # Преобразование в проценты

# Печать строки в консоль

printf " %6d %10.3f %12f%%\n" \

"${arr[0]}" "${arr[1]}" "${num}"

# Запись строки в файл

printf "%7d %10.3f %12f%%\n" \

"${arr[0]}" "${arr[1]}" "${num}" >> "data/tabls/table\_p2.txt"

# Прекращение при достижении половины массива

if [ "${arr[0]}" -gt "$((N/2))" ]; then

style " Достигнут предел массива (${N} элементов). Остановка" $red

break

fi

done < <(printf "%s\n" "${out\_data[@]:1}") # Передача строк начиная со второй (без заголовка)

style "-> enter для окончания просмотра" $yellow n

read

clear # Очистка экрана

}

#!/bash/sh

p\_blue="$(tput setaf 6)"

p\_res="$(tput sgr0)"

red="\033[0;31m"

blue="\033[0;34m"

green="\033[0;32m"

yellow="\033[0;33m"

i\_yellow='\033[93m'

bold="\033[1m"

nc="\033[0m"

alfv="abcdefghijklmnopqrstuvwxyz"

variant\_menu=(

"1 — Контрольный расчет для n точек"

"2 — Расчёт параметра с заданной точностью"

"3 — Запись данных в файлы"

"p - Вывод пояснений к параметрам"

"q — Выход из программы"

)

file\_name\_zast="./config/zast.txt"

# Графики (обработка полученных результатов, выводы)

|  |  |
| --- | --- |
| График Uvx | Параметры |
|  | a0 = 2B; a1 = 5B/c; a2 = 0,3B/c^2  tn=1c; tk = 10c;  n = 25 |

|  |  |
| --- | --- |
| График Uvix | Параметры |
|  | a = 5B; b = 0,05(1/B); Uvx1 = 10B  n = 25 |

**Вывод по графикам**

На основании представленных графиков зависимости напряжения от времени можно сделать следующие выводы:

1. **График UEX(t)*U*EX​(*t*)** (первое изображение):
   * Напряжение UEX*U*EX​ демонстрирует **нелинейный рост** с увеличением времени t*t*.
   * В начальный момент (t=1 c*t*=1c) значение напряжения составляет около 150 В150В, а к t=10 c*t*=10c достигает 350 В350В.
   * Такое поведение соответствует **квадратичной зависимости** UEX(t)=a0+a1t+a2t2*U*EX​(*t*)=*a*0​+*a*1​*t*+*a*2​*t*2, где доминирует квадратичный член (a2t2*a*2​*t*2), обеспечивающий ускоренный рост.
2. **График Uвх(t)*U*вх​(*t*)** (второе изображение):
   * Напряжение Uвх*U*вх​ изменяется **линейно** или **кусочно-линейно** в зависимости от времени.
   * В интервале от t=1 c*t*=1c до t=10 c*t*=10c напряжение снижается с 80 В80В до 10 В10В.
   * Такая динамика может описывать, например, **разряд конденсатора** или работу системы с обратной связью.

**Сравнительный анализ**

* Оба графика отражают временную динамику, но характер изменений различен:
  + UEX(t)*U*EX​(*t*) — **возрастает** с ускорением (квадратичная зависимость).
  + Uвх(t)*U*вх​(*t*) — **убывает** почти линейно.
* Разница в поведении указывает на **разные физические процессы**, лежащие в основе этих зависимостей:
  + Рост UEX*U*EX​ может быть связан с зарядкой цепи или воздействием внешнего источника.
  + Спад Uвх*U*вх​ характерен для процессов рассеивания энергии (например, в RC-цепях).

**Рекомендации**

1. Для UEX(t)*U*EX​(*t*):
   * Исследовать влияние коэффициентов a1*a*1​ и a2*a*2​ на форму кривой.
   * Проверить, соответствует ли модель экспериментальным данным.
2. Для Uвх(t)*U*вх​(*t*):
   * Уточнить, есть ли участки с нелинейностью (например, при малых t*t*).
   * Сравнить с теоретической моделью разряда (например, U(t)=U0e−t/RC*U*(*t*)=*U*0​*e*−*t*/*RC*).

**Заключение:**  
Графики наглядно иллюстрируют два различных режима работы электронных систем. Дальнейший анализ требует уточнения параметров моделей и сравнения с экспериментальными данными.

# Заключение

В ходе работы были проанализированы две математические модели, описывающие зависимость выходного напряжения от входного сигнала и времени.

1. **Кусочно-линейная модель** (Uвых=f(Uвх)*U*вых​=*f*(*U*вх​)):
   * При входном напряжении Uвх≤10 В*U*вх​≤10В выходное напряжение постоянно и равно 5 В5В.
   * При Uвх>10 В*U*вх​>10В зависимость становится квадратичной: Uвых=0,05⋅Uвх2*U*вых​=0,05⋅*U*вх2​.
   * Данная модель может описывать, например, работу ограничителя сигнала или нелинейного усилителя.
2. **Квадратичная временная модель** (UEX(t)=f(t)*U*EX​(*t*)=*f*(*t*)):
   * Выходное напряжение изменяется во времени по квадратичному закону, включая постоянную (a0*a*0​), линейную (a1*a*1​) и квадратичную (a2*a*2​) составляющие.
   * Анализ проводился в интервале времени от 1 с1с до 10 с10с.
   * Такая модель может использоваться для описания процессов с ускоренным изменением напряжения, например, в цепях с заряжающимся конденсатором или в системах автоматического регулирования.

**Итог:**  
Обе модели демонстрируют нелинейные зависимости, но в разных аспектах — первая от входного напряжения, вторая от времени. Полученные результаты могут быть применены для анализа и проектирования электронных схем, систем управления и других технических устройств, где важна нелинейная динамика сигналов.

# Список используемой литературы

1. ГОСТ 19.402-78. Единая система программной документации. Пояснительная записка.  
2. ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ, данных и систем.  
3. Брауде Э.Я. Основы программирования на языке C. — М.: Финансы и статистика.  
4. Документация GNU Bash. URL: https://www.gnu.org/software/bash/  
5. Документация wxMaxima. URL: https://wxmaxima-developers.github.io/wxmaxima/