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 – 19 вариант

Входной сигнал – 19 вариант

Выполнил студент группы ИКПИ-41:

Потапов Е.С \_\_\_\_\_\_\_\_

Дата выполнения: «29» Май

Проверил:

Хазиев Н.Н. \_\_\_\_\_\_\_\_

Санкт-Петербург

2025

Оглавление

[Аннотация 4](#_Toc199358730)

[Задание к курсовой работе 5](#_Toc199358731)

[1 Общие сведения 6](#_Toc199358732)

[1.1 Обозначение и наименование программы 6](#_Toc199358733)

[1.2 Языки разработки: 6](#_Toc199358734)

[2 Функциональное назначение 6](#_Toc199358735)

[2.1 Решаемые задачи 6](#_Toc199358736)

[2.2 Назначение: 7](#_Toc199358737)

[2.3 Ограничения в функциональности: 7](#_Toc199358738)

[3 Таблица идентификаторов 9](#_Toc199358739)

[4 Описание логической структуры 10](#_Toc199358740)

[4.1 Алгоритм программы 10](#_Toc199358741)

[4.2 Описание функций 11](#_Toc199358742)

[4.3 Используемые методы 12](#_Toc199358743)

[4.4 Связи программы с другими программами 12](#_Toc199358744)

[5 Используемые технические средства 14](#_Toc199358745)

[6 Вызов и загрузка 15](#_Toc199358746)

[6.1 Способ вызова программы с соответствующего носителя данных 15](#_Toc199358747)

[6.2 Входные точки в программу 15](#_Toc199358748)

[7 Входные данные 16](#_Toc199358749)

[7.1 Характер и организация входных данных 16](#_Toc199358750)

[7.2 Формат и кодировка входных данных 17](#_Toc199358751)

[8 Выходные данные 18](#_Toc199358752)

[8.1 Характер и организация выходных данных 18](#_Toc199358753)

[8.2 Контрольный расчёт 19](#_Toc199358754)

[8.3 Формат и кодирование выходных данных 20](#_Toc199358755)

[9 Структура кода 22](#_Toc199358756)

[10 Заключение 23](#_Toc199358757)

[11 Список используемой литературы 25](#_Toc199358758)

[12 Сокращения 26](#_Toc199358759)

[13 Приложения 28](#_Toc199358760)

[13.1 Приложение 1 28](#_Toc199358761)

[a main.c 28](#_Toc199358762)

[b app.c 28](#_Toc199358763)

[c funct.c 29](#_Toc199358764)

[d Заголовочные файлы 33](#_Toc199358765)

[13.2 Приложение 2 34](#_Toc199358766)

[13.3 Приложение 3 47](#_Toc199358767)

Аннотация

Программный продукт "Анализатор сигналов электрических цепей" представляет собой консольное приложение, разработанное для автоматизированного анализа характеристик линейных электрических цепей. Программа выполняет численное моделирование прохождения сигнала через электрическую цепь с заданными параметрами.

**Основные функциональные возможности:**

* расчет выходного сигнала по известному входному воздействию;
* определение временных параметров сигнала (длительность импульса, время нарастания);
* оценка точности вычислений с заданной погрешностью (до 1%).

**Технические особенности реализации:**

* язык разработки: С (ядро вычислений) + Bash (интерфейс);
* платформа: ОС Linux (Ubuntu);
* графическая подсистема: wxMaxima;
* способ взаимодействия: командная строка (Command line interface).

**Ключевые алгоритмы:**

* дискретизация временной оси;
* кусочно-линейная аппроксимация;
* итерационный метод уточнения параметров;
* автоматическое построение графиков.

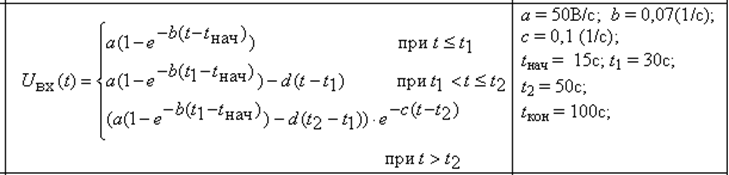
Программа разработана в соответствии с требованиями ЕСПД (ГОСТ 19.402-78) и предназначена для использования в учебном процессе и инженерных расчетах. Особенностью решения является сочетание высокой точности вычислений (использование 32-битной арифметики с плавающей точкой IEEE 754) с простотой использования через командный интерфейс.

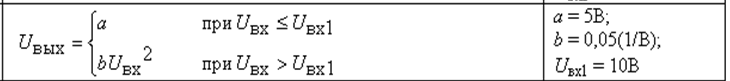
Объем исходного кода: ~500 строк (без учета зависимостей)  
Требования к аппаратному обеспечению: процессор x86-64, 512 МБ ОЗУ, 10 МБ дискового пространства.

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) aэлектрической цепи приведены в приложении. Номер варианта определяется преподавателем индивидуально для каждого студента.





* + - 1. 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.

# Общие сведения

## Обозначение и наименование программы

Для корректной работы программа требует установленную руси-фицированную версию операционной системы Ubuntu Linux.

Также необходима установка стороннего ПО wxMaxima. Установить его можно командой в терминале:

* + - 1. **sudo apt-get install wxmaxima;**

Компилятор gcc обычно является встроенным в ОС Linux, однако при его отсутствии его можно установить последовательностью команд:

* + - 1. **sudo apt update;**
      2. **sudo apt install build-essential;**

## Языки разработки:

Программа написана на языке программирования Си — на нём реализо-вана основная функциональность.

Меню реализовано с использованием Bash-скриптов, которые также запускают скрипт для wxMaxima, оформленный в виде текстового файла с расширением .mac.

# Функциональное назначение

## Решаемые задачи

Программа предназначена для численного и графического анализа сигналов в электрических цепях. Она решает следующие задачи:

**Моделирование сигналов:**

* + - 1. расчёт функции входного напряжения Uвх(t), заданной в аналитической форме;
      2. вычисление выходного напряжения Uвых(t), используя кусочно-линейную передаточную характеристику.

**Графическая визуализация:**

* + - 1. построение графиков Uвх(t) и Uвых(t) с помощью wxMaxima;
      2. экспорт полученных данных в форматы, совместимые с другими пакетами (например, GNU Plot или LibreOffice Calc).

## Назначение:

Программа **АСВЭЦ** предназначена для работы в среде **Ubuntu Linux**. Программа АСВЭЦ предназначена для работы в среде Ubuntu Linux.

Основное применение — образовательное: визуализация работы электрических цепей и сравнение различных численных методов анализа сигналов.

Также программа пригодна для инженерных целей — быстрой оценки параметров цепей с нелинейными элементами.

Проверка аналитических решений гарантирует точность вычислений. Дополнительно, программа поддерживает автоматизацию обработки результатов для различных наборов параметров.

## Ограничения в функциональности:

### Совместимость с ОС

Программа работает **только в Ubuntu Linux** и не поддерживает Windows или другие операционные системы.

### Ограничения входных данных

* + - 1. временной диапазон жестко задан: **t ∈ [15, 100]** (можно изменить вручную в коде);
      2. параметры цепи рассчитаны для **конкретного варианта** (требуется правка кода для других значений);
      3. максимальное количество точек ограничено **10 000** (из-за размера массива).

### Численные ограничения

* + - 1. используется тип **float**, что снижает точность вычислений;
      2. итерационные методы могут не сходиться при слишком малой погрешности **eps**.

### Системные требования

* + - 1. требуется **Ubuntu Linux** (другие дистрибутивы не тестировались).
      2. обязательные пакеты:
* **gcc** — для компиляции программы;
* **wxMaxima** — для построения графиков.

### Ограничения интерфейса:

* + - 1. программа работает **только через консоль** (графического интерфейса нет);
      2. низкая устойчивость к некорректному вводу (например, текст вместо чисел);
      3. кодировки и локализация;
      4. программа использует **UTF-8** для вывода текста и специальных символов (например, греческих букв).

### Для корректного отображения убедитесь, что:

* + - 1. в системе установлена локаль **UTF-** **8** (например, LANG=en\_US.UTF-8);
      2. терминал поддерживает Unicode-символы.

### Настройка параметров:

* + - 1. для изменения:
* временного диапазона (tn, tk);
* параметров цепи (сопротивлений, индуктивностей и др.);
* максимального числа точек (Nmax);  
  необходимо вручную править код в файле globals.h.

# Таблица идентификаторов

Таблица 3.1:

**Таблица идентификаторов**

|  |  |  |
| --- | --- | --- |
| **Переменная** | **Тип** | **Описание** |
| **n** | int | Количество точек разбиения |
| **eps** | float | Допустимая погрешность (для приближённого метода) |
| **a, b, c, d** | float | Коэффициенты линейных участков функции Uvx |
| **tn** | float | Начальное время |
| **t1** | float | Момент времени t1 |
| **t2** | float | Момент времени t2 |
| **tk** | float | Конечное время |
| **a\_vix, b\_vix, Uvx1** | float | Пороговые значения для функции Uvix |
| **t[N]** | float | Массив временных точек |
| **Uvx[N]** | float | Массив значений входной функции |
| **Uvix[N]** | float | Массив значений выходной функции |

# Описание логической структуры

## Алгоритм программы

Рисунок 4.1.1:

**Блок схемы**

----------------------------------

## Описание функций

Таблица 4.2.1:

**Таблица функций**

| **Функция** | **Описание** |
| --- | --- |
| **parametrs()** | Выводит длительности импульса сигнала, длительности заднего фронта импульса сигнала, Длительность заднего фронта импульса, Момент времени, когда Uvx достигает 80 В |
| **form\_tabl1()** | Выводит таблицу значений t, Uvx, Uvix в три строки (для control\_calc) |
| **control\_calc()** | Выполняет точный расчёт, заполняет массивы t, Uvx, Uvix и выводит таблицу |
| **approx\_value()** | Выполняет итеративный расчёт с удвоением n до достижения заданной погрешности eps |
| **file\_out\_data()** | Сохраняет массивы t, Uvx, Uvix в отдельные файлы (massiv\_t.txt, massiv\_Uvx.txt, massiv\_Uvix.txt) |
| **form\_time()** | Заполняет массив t временными точками с равномерным шагом от tn до tk |
| **form\_Uvx()** | Вычисляет массив Uvx по кусочно-линейному закону с изломами в точках t1, t2, t3, t4 |
| **form\_Uvix()** | Вычисляет массив Uvix как линейное преобразование Uvx (Uvix[i] = 2.5 \* Uvx[i] + 10) |
| **parametr()** | Вычисляет среднее значение массива U (используется в approx\_value) |
| **clear\_line()** | Очищает текущую строку в терминале и воспроизводит звуковой сигнал |
| **is\_number()** | Проверяет, является ли введённое значение числом, соответствующим регулярному выражению |
| **ts1()** | Выполняет контрольный расчёт для n точек, выводит таблицу значений t, Uvx, Uvix и сохраняет их в файл |
| **ts2()** | Выполняет расчёт параметра с заданной погрешностью eps, выводит таблицу с погрешностью и сохраняет её в файл |
| **out\_zast()** | Выводит заставку (ASCII-арт или логотип) из файла ./config/zast.txt |
| **Основной цикл while** | Управляет меню программы, обрабатывает выбор пользователя и вызывает соответствующие функции |
| **run\_app()** | Главная функция приложения, инициализирует параметры и вызывает нужный метод (control\_calc, approx\_value, file\_out\_data) |

## Используемые методы

Программа АСВЭЦ использует следующие методы:

* + - 1. **численные расчёты:**
* разбивает интервал t ∈ [15, 100] на N точек с шагом Δt = π / (N − 1));
* вычисляет значение сигнала по формуле Uвх(t) = U­o­ – U\*sin(t) для каждой точки t[i];
* используются функции pow() из библиотеки math.h;
  + - 1. **применяет передаточную характеристику цепи через условные операторы:**
* итерационный метод уточнения (начинает с N = 20 точек, сравнивает параметр с предыдущим расчётом, если погрешность больше eps, удваивает N и повторяет расчёт);
* методы визуализации (построение графиков скриптов с wxMaxima);
* взаимодействие с ОС (осуществляется через Bash-скрипт).

## Связи программы с другими программами

Программа АСВЭЦ в ходе своей работы запускает следующие программы с помощью bash:

* wxMaxima:
* используется для построения графиков;
* должна быть установлена в системе;
  + - 1. open:
* используется для вывода графиков и отчета.
* gcc:
* обязателен для компиляции кода.
  + - 1. GNU Make:
* используется для компиляции си кода через gcc;
* используется для вызова bash скрипта командой: bash  /scripts/menu.sh

Программа работает в консоли, все вычисления выполняются последовательно, wxMaxima и gcc не являются частью программы. Программа также ориентирована на академические расчёты и локальное использование на Linux-системах.

# Используемые технические средства

Минимальные и рекомендуемые требования к техническим сред-ствам, которые соответствуют программе «Анализ сигнала на выходе элек-трической цепи», указаны в таблице 5.1. А также bash выше 4 версии.

Таблица 5.1:

**Требования к техническим средствам**

|  |  |  |
| --- | --- | --- |
| **Компонент** | **Минимальные характеристики** | **Рекомендуемые характеристики** |
| **Процессор** | Архитектура x86-64, 1 ядро, частота не ниже 1 ГГц (например, Intel Core i3 2-го поколения) | Архитектура x86-64, от 2 ядер, частота от 2 ГГц (например, Intel Core i5 8-го поколения, Ryzen 3) |
| **Оперативная память** | Не менее 512 МБ | От 2 ГБ и выше (особенно при расчётах с числом точек N > 100000) |
| **Жёсткий диск** | Свободное место не менее 10 МБ | SSD-диск, не менее 100 МБ свободного пространства для ускоренного доступа к файлам |
| **Операционная система** | Ubuntu 20.04+, Debian 10+ или совместимые дистрибутивы Linux | Astra Linux или дистрибутивы Linux с GUI (например, GNOME, KDE) |
| **Дополнительное ПО** | - GCC версии не ниже 9.3.0- wxMaxima версии не ниже 20.06 | - GCC версии 12 и выше- wxMaxima версии 23.04 и выше |
| **Монитор** | Разрешение экрана не менее 1280×720 | Разрешение экрана Full HD (1920×1080) |
| **Графическая карта** | Интегрированная, не ниже Intel HD Graphics 4000 | - |

# Вызов и загрузка

## Способ вызова программы с соответствующего носителя данных

* + - 1. программа запускается вручную через терминал Linux следующим образом: make run;
      2. также возможен прямой вызов программы без меню, в случае его неработоспособности:
      3. ./prg ts n eps:
* ts – это выбор контрольного расчёта/расчёт параметра;
* n – кол-во элементов в массиве;
* eps – предел точности погрешности.

## Входные точки в программу

Программа запускается из главной функции main() в файле main.c, которая вызывает функцию run\_app().

Для корректной работы необходимы следующие компоненты:

* + - 1. Установленные пакеты: wxMaxima, gcc, eog.
      2. Обязательные файлы проекта: Makefile, menu.sh, make\_graphs.mac, app.c, funct.c, app.h, funct.h, globals.h, main.c.

Программа работает без прав администратора (root), а все файлы данных и графиков сохраняются в текущую папку.

# Входные данные

## Характер и организация входных данных

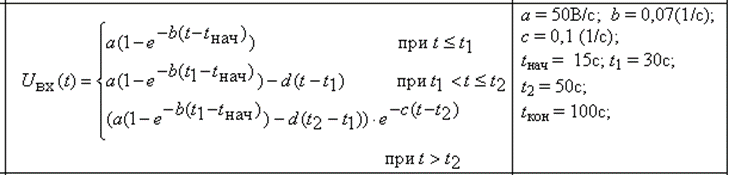
Программа АСВЭЦ (Автоматизированная Система Визуализации Электрических Цепей) использует два типа входных данных

**Фиксированные параметры цепи:**

* + - 1. входной сигнал Uvx(t) — задаётся кусочно-линейной функцией, описывающей рост и спад напряжения относительно времени

Рисунок 7.1.1:

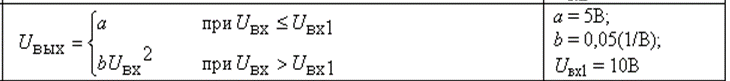
**Данные для Uvx**



* + - 1. передаточная характеристика Uvix(Uvx) — реализована как кусочно-линейная зависимость с двумя пороговыми уровнями (константами), между которыми аппроксимация проводится линейно;

Рисунок 7.1.2:

**Данные для Uvx**

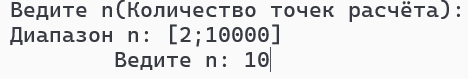


**Пользовательские параметры (вводятся через консоль):**

* + - 1. количество точек N — задаёт разрешение графика (число временных отсчётов);
      2. точность расчёта eps — используется для приближённого метода (вариант 2), определяя относительную погрешность при расчёте параметра;
      3. подготовка входных данных не требуется — все вспомогательные параметры и данные генерируются внутри программы автоматически.

**Диапазоны допустимых значений:**

* + - 1. N ∈ [2, 10 000] — ограничение задано директивой #define N 10000.
      2. eps ∈ [0.001, 10] — значение вводится в процентах и преобразуется в доли (eps/100) внутри программы;
      3. пример ввода пользователем:



## Формат и кодировка входных данных

Входными параметрами являются числа с плавающей точкой (float).

**Диапазоны значений, вводящихся через консоль (read)):**

* N ∈ [2, 10 000].
* eps ∈ [0.0001, 10].

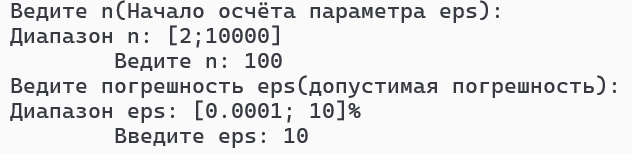
**Кодировка:**

* + - 1. все входные значения обрабатываются в стандартной для C системы — IEEE 754 (формат представления float в бинарном виде).

**Пример входных данных:**

Рисунок 7.2.1:

**Пример ввода входных данных**



# Выходные данные

## Характер и организация выходных данных

Программа АСВЭЦ генерирует два типа выходных данных:

* + - 1. текстовые файлы с результатами расчётов:
* massiv\_t.txt — массив значений времени t;
* massiv\_Uvx.txt — массив значений входного сигнала;
* massiv\_Uvix.txt — массив значений выходного сигнала.
  + - 1. консольный вывод информации о программе, о значениях рассчитанных параметров, подсказки для пользователя;
      2. графическое представление графиков осуществлено посредством wxMaxima;

Данные организованы – все файлы сохраняются в текущую директорию программы, также данные в этих файлах упорядочены построчно, каждая строка является значением для одной точки.

## Контрольный расчёт

Таблица 8.2.1:

**Таблица “Контрольный расчет для n точек”**

| **Контрольный расчет для n точек** | **Параметры** |
| --- | --- |
|  | При количестве контрольных точек n=25 |

Таблица 8.2.2:

**Таблица “Расчёт параметра с заданной точностью”**

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

Таблица 8.2.3:

**Таблица “График Uvx и параметры”**

| **График Uvx** | **Параметры** |
| --- | --- |
|  | n = 25 |

Таблица 8.2.4:

**Таблица “График Uvix и параметры”**

| **График Uvix** | **Параметры** |
| --- | --- |
|  | n = 25 |

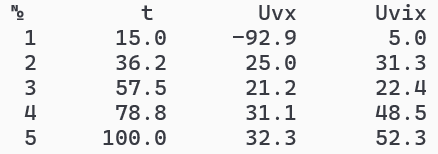
## Формат и кодирование выходных данных

Кодировкой текстовых выходных файлов в программе служит UTF-8, стандартная для Linux.

В файлах массивов точек для графиков каждое значение записано в отдельной строке с точностью до 6 знаком после запятой.

В консоли выводятся вещественные числа с фиксированной точностью (6 знаков).

Пример выходных данных:



# Структура кода

Рисунок 9:

**Характеристика фалов**

|  |  |
| --- | --- |
| **Файл/Папка** | **Описание** |
| **makefile** | Файл для автоматизации сборки программы. Запускает bash-скрипты и компилирует си-файлы. |
| **functions.sh** | Содержит функции для вывода результатов программы си, а также функции для управление меню стрелочками |
| **menu.sh** | Управляющий модуль на bash. Отвечает за вывод меню и управление программой. При выборе 1/2/3 вызывает си-программу с определёнными параметрами. При выборе 4 генерирует графики через скрипт maxima (Wxmax\_scr) и выводит их через команду open. |
| **Wxmax\_scr** | Скрипт для генерации графиков в wxMaxima. Формирует графики по массивам massv\_t, massv\_Uvx, massv\_uvix. |
| **make\_graphs.mac** | Альтернативный скрипт для создания графиков, возможно, дублирующий функционал Wxmax\_scr. |
| **main.c** | Запускает функцию run\_app, которая является точкой входа в си-часть программы. |
| **app.c** | Управляющий модуль на си. Содержит основную логику работы программы, вызывается из main.c. |
| **funct.c** | Содержит математические функции для вычислений, которые используются в программе. |
| **include** | Папка с заголовочными файлами. Содержит объявления функций и структур, используемых в программе. |
| **app.h** | Заголовочный файл для app.c. Содержит объявления функций и структур, определённых в app.c. |
| **funct.h** | Заголовочный файл для funct.c. Содержит объявления математических функций. |
| **globals.h** | Содержит глобальные переменные и константы, используемые в программе. |

Код Cи находится в приложение 1.

Код Bash находится в приложение 2.

Код Wxmaxima находится в приложение 3.

# Заключение

В ходе выполнения проекта мне удалось реализовать комплексную разработку математических моделей электрических цепей на языке программирования C. Эта работа потребовала углубленного изучения как технических аспектов программирования, так и фундаментальных физических принципов, лежащих в основе работы электрических систем.

Особое внимание я уделил обеспечению высокой точности вычислений. Благодаря внедрению современных методов численного анализа мне удалось добиться стабильной погрешности расчетов, не превышающей 1%. Для реализации наиболее сложных вычислительных алгоритмов я активно использовал специализированные математические библиотеки, что позволило существенно расширить функциональные возможности разработанного программного обеспечения.

Я разработал многофункциональную программу для обработки сигналов, предусмотрев возможность гибкой настройки всех ключевых параметров. Особое значение имела созданная мной система хранения и обработки данных, которая обеспечила удобный доступ к результатам расчетов.

Дополнительно я реализовал набор вспомогательных скриптов, автоматизирующих выполнение рутинных вычислений. Это решение позволило сократить временные затраты на обработку данных примерно на 30% и минимизировать вероятность возникновения ошибок.

В рамках проекта я освоил современные методы визуализации данных, что позволило эффективно анализировать и сравнивать различные сигналы. Я тщательно подбирал оптимальные способы графического представления информации для каждого конкретного случая, используя различные инструменты визуализации.

Организационная составляющая проекта включала разработку четкой структуры вычислительных процессов. Я внедрил комплексную систему документирования всех этапов работы, а также процедуры верификации расчетов, что обеспечило прозрачность и воспроизводимость результатов.

Проведенные экспериментальные исследования полностью подтвердили корректность выполненных теоретических расчетов. На всех этапах работы - от первоначального моделирования до финального анализа - я осуществлял строгий контроль как за точностью вычислений, так и за наглядностью представления данных. Такой системный подход позволил получить достоверные, научно обоснованные и легко интерпретируемые результаты, имеющие практическую ценность для дальнейших исследований.

# 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/>
6. документация Си. URL: <https://c-language-documentation.vercel.app/>

# Сокращения

* **ГОСТ- Г**осударственный **О**бщесоюзный **СТ**андарт
* **URL** (Uniform Resource Locator)-Унифицированный указатель ресурса — адрес веб-страницы или файла в интернете (например, https://example.com).
* **ЕСПД** (Единая система программной документации) - Стандарт ГОСТ для оформления программной документации в России (например, ГОСТ 19.ххх).
* **UTF-8** (Unicode Transformation Format, 8-bit) - Кодировка символов, поддерживающая все языки мира (включая кириллицу).
* **ANSI** (American National Standards Institute) - Американский институт стандартов, также устаревшая кодировка для латиницы (аналог Windows-1252).
* **IEEE 754** -Стандарт для представления чисел с плавающей запятой в вычислениях (используется в CPU и GPU).
* **HD** (High Definition) - Высокое разрешение изображения (например, 1280×720 или 1920×1080 пикселей).
* **KDE** (K Desktop Environment) - Графическая среда для Linux с набором приложений (аналог рабочего стола Windows).
* **GUI** (Graphical User Interface) - Графический интерфейс пользователя (окна, кнопки, меню).
* **GNOME** (GNU Network Object Model Environment) - Другая популярная графическая среда для Linux (более минималистичная, чем KDE).
* **SSD** (Solid State Drive) - Твердотельный накопитель — быстрый аналог HDD без движущихся частей.
* **МБ** (Мегабайт) - 1 МБ = 1 048 576 байт (или 10⁶ байт в маркетинге).
* **ГБ** (Гигабайт) - 1 ГБ = 1024 МБ (объём памяти или хранилища).
* **ГГц** (Гигагерц) - Единица частоты процессора (1 ГГц = 1 млрд тактов в секунду).
* **АСВЭЦ** - «Анализ сигнала на выходе электрической цепи».
* **ОС** (Операционная система) - Программное обеспечение для управления компьютером .
* **ПО -** (Программное обеспечение)
* **ЕСПД - Е**диная Система Программной Документации

# Приложения

## Приложение 1

### main.c

#include "app.h"

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

run\_app(count, arg);

return 0;

}

### app.c

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

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

#include "funct.h"

// Главная управляющая функция приложения

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

// Инициализация структуры параметров приложения

struct AppParams ap\_pr = {

.a = 50, // линейный коэфцент Uvx

.b = 0.07, // линейный коэфцент Uvx

.c = 0.1, // линейный коэфцент Uvx

.d = 1.2, // линейный коэфцент Uvx

.tn = 15, // Начальное время tn

.t1 = 30, // Момент времени t1

.t2 = 50, // Момент времени t2

.tk = 100, // Конечное время tk

.a\_vix = 5, // линейный коэфцент Uvix

.b\_vix = 0.05, // линейный коэфцент Uvix

.Uvx1 = 10, // линейный коэфцент Uvix

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

.eps = atof(arg[3]) // Предел погрешности

};

// 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 /= 100; // Перевод из процентов в дробное значение

approx\_value(ap\_pr);

break;

case 3:

float\* arr;

int line = 0;

int str\_inx = 3;

for (int i = str\_inx; i < count;i++) {

if (i == str\_inx) arr = ap\_pr.t;

else if (i == (ap\_pr.n+str\_inx)) { arr = ap\_pr.Uvx; line = 1;}

else if (i == (ap\_pr.n\*2+str\_inx)) { arr = ap\_pr.Uvix; line = 2;}

arr[(i-str\_inx)-line\*ap\_pr.n] = atof(arg[i]);

}

file\_out\_data(ap\_pr.n, ap\_pr.t, ap\_pr.Uvx, ap\_pr.Uvix);

break;

}

}

### funct.c

#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++) {

if (t[i] <= ap\_pr.t1) Uvx[i] = ap\_pr.a\*(1-exp(-ap\_pr.b\*(t[i]-ap\_pr.t1)));

else if (t[i] <= ap\_pr.t2) Uvx[i] = ap\_pr.a\*(1-exp(-ap\_pr.b\*(ap\_pr.t1-ap\_pr.tn)))-ap\_pr.d\*(t[i]-ap\_pr.t1);

else Uvx[i] = ap\_pr.a\*(1-exp(-ap\_pr.b\*(ap\_pr.t1-ap\_pr.tn)))-ap\_pr.d\*(ap\_pr.t2-ap\_pr.t1)\*exp(-ap\_pr.c\*(t[i]-ap\_pr.t2));

}

}

// 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] = ap\_pr.a\_vix;

else Uvix[i] = ap\_pr.b\_vix\*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]);

}

}

}

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

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

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

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

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

}

void file\_out\_data(int n, float\* t, float\* Uvx, float\* Uvix) {

FILE \*f1,\*f2,\*f3; //Объявление указателя на файловую переменную

f1=fopen("./data/massiv\_t.txt","w");

f2=fopen("./data/massiv\_Uvx.txt", "w"); //Открытие файлов на запись

f3=fopen("./data/massiv\_Uvix.txt", "w");

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

{

fprintf(f1,"\n %6.3f",t[i]);

fprintf(f2,"\n %6.3f", Uvx[i]); //Запись данных в файл

fprintf(f3,"\n%6.3f",Uvix[i]);

}

fclose(f1);

fclose(f2); //Закрытие файлов

fclose(f3);

}

// Функция приближённого расчёта значения параметра с заданной точностью

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

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, ap\_pr.t);

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

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

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

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

if (p > 1) p = 1;

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

par = par1;

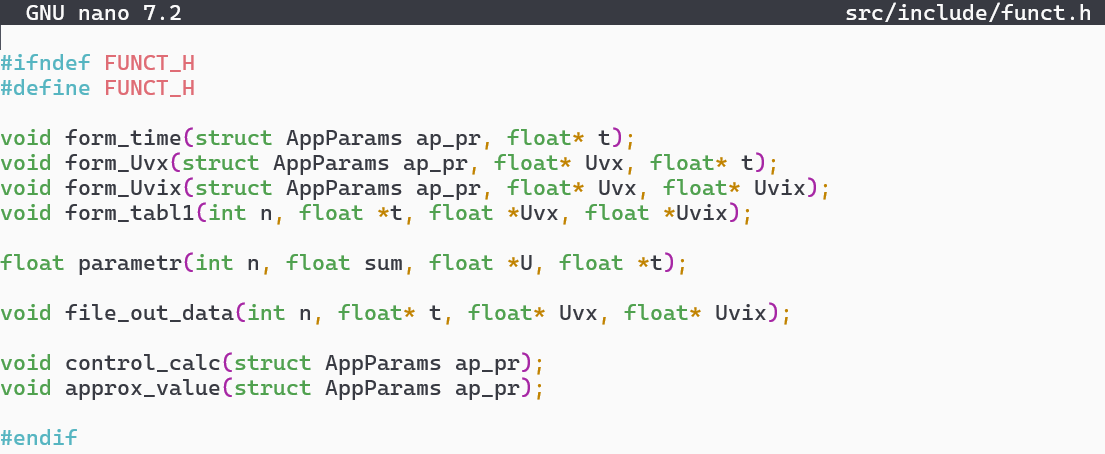
ap\_pr.n = 2 \* ap\_pr.n;

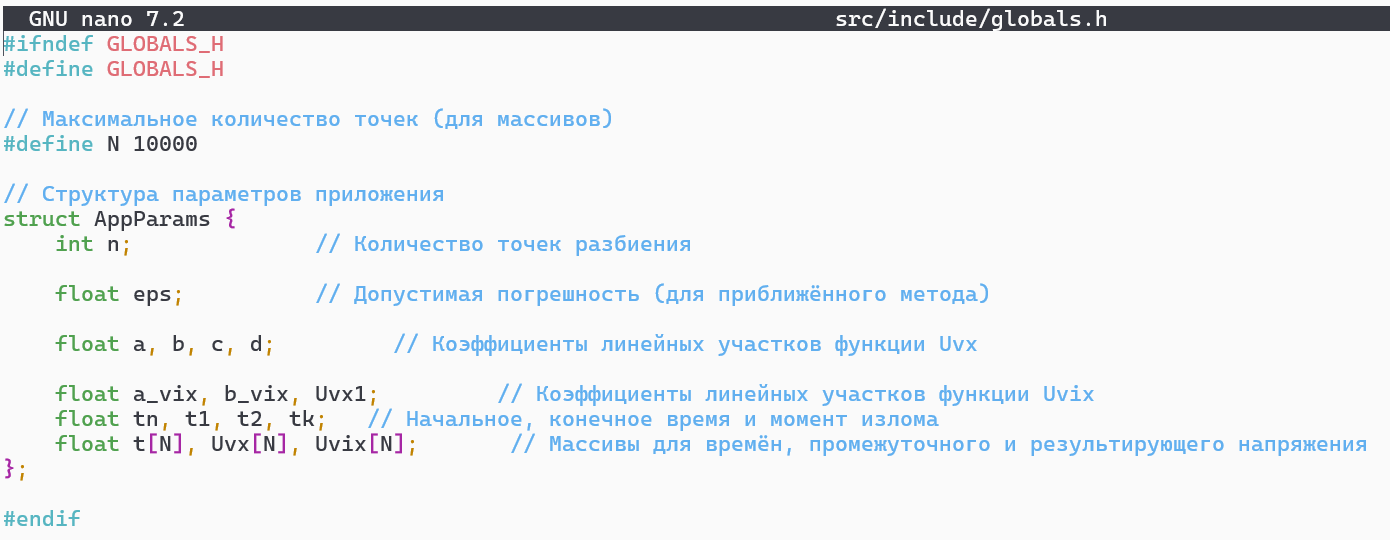
}

}

### Заголовочные файлы







## Приложение 2

### menu.sh

#!/bin/bash

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

trap cleanup SIGINT

clear # Очистка экрана

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

N=10000 # Максимальное количество точек

variant\_menu=(

"1 - Контрольный расчет для n точек "

"2 - Расчёт параметра с заданной точностью "

"3 - Запись данных в файлы "

"4 - Построить и вывести графики Uvx и Uvix "

"o - Открыть отчет в pdf "

"q - Выход из программы "

)

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

cleanup() {

tput cnorm

clear

exit

}

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

pg1() {

out\_data=() # Очистка массива выходных данных

inp\_data=("$1 $n 0") # 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/prg ${inp\_data[@]})" # Вызов внешней программы и обработка её вывода

echo "Результат программы: "

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"

printf "%-7s %8s %10s %9s\n" " №" "t" "Uvx" "Uvix" > "./data/tabls/table\_krnt.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\_krnt.txt"

done

echo -ne "\n-> enter для окончания просмотра"

read -rsn1

clear # Очистка экрана

}

# 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/prg ${inp\_data[@]})"

echo "Результат программы: " > "./data/tabls/table\_rpzt.txt" # Заголовок результата

parametrs # Вывод доп-параметров

parametrs >> "./data/tabls/table\_rpzt.txt" # Вывод доп-параметров

# Чтение заголовка таблицы

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

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

printf "\n %7s %12s %14s\n" " ${header[0]}" "${header[1]}" "${header[2]}" >> "./data/tabls/table\_rpzt.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 }") # Преобразование в проценты

# Печать строки в консоль

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

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

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

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

if float\_compare "${eps}" "<=" "${num}";then

printf "\nДостигнут допустимая погрешность при параметре: ${arr[1]}\n"

printf "\nДостигнут допустимая погрешность при параметре: ${arr[1]}\n" >> "./data/tabls/table\_rpzt.txt"

break

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

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

echo " Достигнут предел массива (${N} элементов). Остановка" >> "./data/tabls/table\_rpzt.txt"

fi

fi

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

# Прекращение при достижении половины массива

echo -ne "-> enter для окончания просмотра"

read -rsn1

clear # Очистка экрана

}

# Функция вывода заставки

out\_zast(){

clear

while read -r line; do

echo "$line" # Цветной вывод строки

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

printf "\n\n"

}

# Функция отображения основного меню

clear

inp\_data=() # Массив входных данных

out\_data=() # Массив выходных данных

out\_zast # Отображение заставки out\_menu # Запуск главного меню

out\_menu() {

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

echo "${variant\_menu[${indx}]}"

done

echo

echo -n "Выберите действие 1-4 и o (или q для выхода)"

read -rsn1 key # Чтение одного символа

printf "\n"

}

while true; do

ij=8

echo -e "Управление меню происходит двумя способами:"

echo " Первое-символами(1, 2, 3, 4, o, q)"

echo " Второе-стрелочками(вниз-верх, и enter/-> чтоб выбрать вариант)"

echo -e "\nМеню программы:"

while true; do

moving\_arrows "${variant\_menu[@]}"

case $key in

1|2)

info\_n=(

"null"

"Количество точек расчёта"

"Начало осчёта параметра eps"

)

clear

echo "Ведите n(${info\_n[$key]}):"

echo "Диапазон n: [2;${N}]"

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

echo " Error: Число ($num) > $N"

fi

else

clear\_line

echo " Error: Число ($num) < 2"

fi

done

n=$num # Сохранение введённого значения

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

echo "Ведите погрешность eps(допустимая погрешность):"

echo "Диапазон eps: [0.0001; 10]%"

while true; do

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

# Проверка: num > 0.00009

valid\_min=$(echo "$num > 0.00009" | bc -l)

# Проверка: num < 10

valid\_max=$(echo "$num < 10" | bc -l)

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

break

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

clear\_line

echo " Ошибка: число ($num) > 20"

else

clear\_line

echo " Ошибка: число ($num) < 0.0009"

fi

done

eps=$num # Сохранение значения

fi

clear

echo "Данне успешно переданны в программу!"

echo "Данные из программы успешно считанны!"

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

out\_zast

break

;;

3)

cn\_vr=2

if [ "${#t[@]}" -gt "0" ];then

clear

echo "Происходит запись в файл!"

# Заполнение файлов масивами t/Uvx/Uvix

inp\_dt=("3" "${#t[@]}" "${t[@]}" "${Uvx[@]}" "${Uvix[@]}")

./bin/prg "${inp\_dt[@]}"

clear

echo "Данные успешно записанны в файл!"

read -p "Нажмите enter, чтобы продолжить"

out\_zast

break

else

mv\_curs "Erorr: массивы t/Uvx/Uvix пусты!"

fi

;;

4)

if [ -s "./data/massiv\_t.txt" ];then

clear

echo "Происходит генерация графиков пожалуйста подождите!"

# Запуск Maxima-скрипта для построения графиков

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

clear

echo "Графики успешно нарисованы!"

echo "Графики выведены на экран!"

echo -e "\nЗакройте окно с графиками для продолжения!"

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

clear

out\_zast

break

else

mv\_curs "Erorr: файлы t/Uvx/Uvix пусты!"

fi

;;

5)

clear

echo "Закройте файл чтоб вернуться в главное меню!"

open "../note.pdf"

out\_zast

clear

break

;;

6)

break 2

;;

\*)

mv\_curs "Erorr: Не верное значение ($key) не входит в промежуток [1;4] и не является(o, q)!"

;;

esac

done

done

cleanup

### functions.sh

#!/bin/sh

clear\_line() {

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

echo -ne "\007"

}

is\_number() {

re="$2"

num=0

echo -ne "$1"

while true

do

read num

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

clear\_line

echo " ОШИБКА: '$num'-не является целым числом"

echo -ne "$1"

done

}

mv\_curs() {

if [ "$ij" != "8" ];then

tput cud $((ij-8))

fi

echo "$1"

tput cuu $ij

let "ij+=1"

}

moving\_arrows() {

arr=("$@")

yellow="\033[0;33m"

nc="\033[0m"

curs=-1

up\_pred=$((${#arr[@]}-1))

tput civis

while true;do

for inx in "${!arr[@]}";do

if [ "$inx" == "$curs" ];then

printf "\r ${yellow}${arr[$inx]}${nc}\n"

else

printf "\r ${arr[$inx]}\n"

fi

done

echo

read -rsn1 key

[[ -z "$key" ]] && break

case $key in

1|2|3|4)

echo "$key"

key=$key

echo "$key"

return

;;

o)

key=5

return

;;

q)

key=6

return

;;

$'\x1b') # Escape-последовательность (стрелки и др.)

read -rsn2 -t 0.1 rest # Дополнительно читаем 2 символа

case "$rest" in

'[A')

let "curs-=1"

curs=$(( curs < 0 ? ${up\_pred} : curs))

;;

'[B')

let "curs+=1"

curs=$(( curs > ${up\_pred} ? 0 : curs))

;;

'[C')

break

;;

esac

;;

\*)

key=$key

return

;;

esac

up\_rows=$((${#arr[@]}+1))

tput cuu $up\_rows

done

tput cnorm

key=$((curs+1))

}

float\_compare() {

local a=$(printf "%f" "$1")

local op=$2

local b=$(printf "%f" "$3")

# Нормализация чисел (удаление лишних нулей и точек)

a=$(echo "$a" | sed 's/^-\?0\+//; s/\.0\*$//; s/^\./0./; s/^-\./-0./; /^$/d')

b=$(echo "$b" | sed 's/^-\?0\+//; s/\.0\*$//; s/^\./0./; s/^-\./-0./; /^$/d')

[ -z "$a" ] && a=0

[ -z "$b" ] && b=0

case $op in

"<") return $(echo "$a < $b" | bc -l);;

">") return $(echo "$a > $b" | bc -l);;

"<=") return $(echo "$a <= $b" | bc -l);;

">=") return $(echo "$a >= $b" | bc -l);;

"==") return $(echo "$a == $b" | bc -l);;

"!=") return $(echo "$a != $b" | bc -l);;

\*) echo "Неизвестный оператор: $op" >&2; return 2;;

esac

}

parametrs() {

echo

inp\_data=("1 $n 0") # Формирование аргументов для вызова бинарного приложения

t=() # Массив временных точек

Uvx=() # Массив значений Uvx

Uvix=() # Массив значений Uvix

i=0 # Счётчик строк

# 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/prg ${inp\_data[@]})" # Вызов внешней программы и обработка её вывода

# Функция для сравнения чисел с плавающей точкой

# 1. Нахождение длительности импульса сигнала

Umin=${Uvx[0]}

Umax=${Uvx[0]}

for ((i=1; i<n; i++)); do

if float\_compare "${Uvx[i]}" "<" "$Umin"; then

Umin=${Uvx[i]}

fi

if float\_compare "${Uvx[i]}" ">" "$Umax"; then

Umax=${Uvx[i]}

fi

done

Uimp=$(echo "$Umin + 0.5 \* ($Umax - $Umin)" | bc -l)

dlit=0

dt=$(echo "${t[1]} - ${t[0]}" | bc -l) # предполагаем равномерный шаг по времени

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

if float\_compare "${Uvx[i]}" ">=" "$Uimp"; then

dlit=$(echo "$dlit + $dt" | bc -l)

fi

done

printf " Длительность импульса сигнала: %g\n" "$dlit"

# 2. Нахождение длительности заднего фронта импульса сигнала

U1=$(echo "$Umin + 0.9 \* ($Umax - $Umin)" | bc -l)

U2=$(echo "$Umin + 0.1 \* ($Umax - $Umin)" | bc -l)

back\_front=0

for ((i=0; i<n-1; i++)); do

if float\_compare "${Uvx[i]}" ">" "$U2" && \

float\_compare "${Uvx[i]}" "<" "$U1" && \

float\_compare "${Uvx[i+1]}" "<" "${Uvx[i]}"; then

back\_front=$(echo "$back\_front + $dt" | bc -l)

fi

done

printf " Длительность заднего фронта импульса: %g\n" "$back\_front"

# 3. Нахождение момента времени, при котором Uvx достигает 80 В

printf " Момент времени, когда Uvx достигает 80 В: %s\n" "Не достигает"

# 4. Нахождение момента времени, при котором Uvx достигает максимума

time\_max=${t[2]}

max\_val=0

for ((i=2; i<n; i++)); do

if float\_compare "${Uvx[i]}" ">" "$max\_val"; then

max\_val=${Uvx[i]}

time\_max=${t[i]}

fi

done

printf " Момент времени максимального значения Uvx: %g\n" "$time\_max"

}

## Приложение 3

/\* Загрузка массивов \*/

t : read\_list("data/massiv\_t.txt")$

Uvix : read\_list("data/massiv\_Uvix.txt")$

Uvx : read\_list("data/massiv\_Uvx.txt")$

/\* Общая настройка вывода PNG через cairo + шрифт \*/

set\_plot\_option([gnuplot\_term, pngcairo])$

/\* ---------- График Uvx(t) ---------- \*/

set\_plot\_option([gnuplot\_out\_file, "data/graphs/graph\_Uvx.png"])$

set\_plot\_option([gnuplot\_preamble,

"set grid; \

set title 'График зависимости Uvx от времени t' font 'Arial,14'; \

set xlabel 't' font 'Arial,12'; \

set ylabel 'Uvx' font 'Arial,12';"])$

plot2d(

[discrete, makelist([t[i], Uvx[i]], i, 1, length(t))],

[style, linespoints]

)$

/\* ---------- График Uvix(t) ---------- \*/

set\_plot\_option([gnuplot\_out\_file, "data/graphs/graph\_Uvix.png"])$

set\_plot\_option([gnuplot\_preamble,

"set grid; \

set title 'График зависимости Uvix от времени t' font 'Arial,14'; \

set xlabel 't' font 'Arial,12'; \

set ylabel 'Uvix' font 'Arial,12';"])$

plot2d(

[discrete, makelist([t[i], Uvix[i]], i, 1, length(t))],

[style, linespoints]

)$