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 – 18 вариант

Входной сигнал – 6 вариант

Выполнил студент группы ИКПИ-41:

Плисов К.K. \_\_\_\_\_\_\_\_

Дата выполнения: «29» Май

Проверил:

Хазиев Н.Н. \_\_\_\_\_\_\_\_

Санкт-Петербург

2025

Оглавление

[Аннотация 4](#_Toc199297828)

[Задание к курсовой работе 5](#_Toc199297829)

[1 Общие сведения 6](#_Toc199297830)

[1.1 Обозначение и наименование программы 6](#_Toc199297831)

[1.2 Языки разработки: 6](#_Toc199297832)

[2 Функциональное назначение 6](#_Toc199297833)

[2.1 Решаемые задачи 6](#_Toc199297834)

[2.2 Назначение: 7](#_Toc199297835)

[2.3 Ограничения в функциональности: 7](#_Toc199297836)

[3 Таблица идентификаторов 9](#_Toc199297837)

[4 Описание логической структуры 10](#_Toc199297838)

[4.1 Алгоритм программы 10](#_Toc199297839)

[4.2 Описание функций 11](#_Toc199297840)

[4.3 Используемые методы 12](#_Toc199297841)

[4.4 Связи программы с другими программами 12](#_Toc199297842)

[5 Используемые технические средства 13](#_Toc199297843)

[6 Вызов и загрузка 14](#_Toc199297844)

[6.1 Способ вызова программы с соответствующего носителя данных 14](#_Toc199297845)

[6.2 Входные точки в программу 14](#_Toc199297846)

[7 Входные данные 15](#_Toc199297847)

[7.1 Характер и организация входных данных 15](#_Toc199297848)

[7.2 Формат и кодировка входных данных 16](#_Toc199297849)

[8 Выходные данные 17](#_Toc199297850)

[8.1 Характер и организация выходных данных 17](#_Toc199297851)

[8.2 Контрольный расчёт 18](#_Toc199297852)

[8.3 Формат и кодирование выходных данных 19](#_Toc199297853)

[9 Структура кода 21](#_Toc199297854)

[10 Заключение 22](#_Toc199297855)

[11 Список используемой литературы 24](#_Toc199297856)

[12 Сокращения 25](#_Toc199297857)

[13 Приложения 27](#_Toc199297858)

[13.1 Приложение 1 27](#_Toc199297859)

[a main.c 27](#_Toc199297860)

[b app.c 27](#_Toc199297861)

[c funct.c 27](#_Toc199297862)

[d Заголовочные файлы 30](#_Toc199297863)

[13.2 Приложение 2 31](#_Toc199297864)

[13.3 Приложение 3 45](#_Toc199297865)

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.

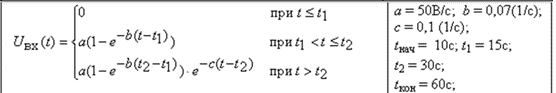
Программа разработана в соответствии с требованиями ЕСПД (ГОСТ 19.402-78) и предназначена для использования в учебном процессе и инженерных расчетах. Особенностью решения является сочетание высокой точности вычислений (использование 32-битной арифметики с плавающей точкой IEEE 754) с простотой использования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

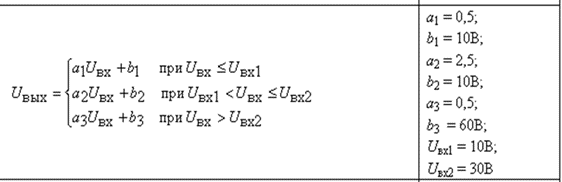
Объем исходного кода: ~500 строк (без учета зависимостей)  
Требования к аппаратному обеспечению: процессор x86-64, 512 МБ ОЗУ, 10 МБ дискового пространства.

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 представляет собой специализированное программное обеспечение, разработанное для работы под управлением операционной системы Ubuntu Linux. Основное назначение программы заключается в образовательной сфере, где она используется для наглядной демонстрации принципов работы электрических цепей и сравнительного анализа различных численных методов исследования сигналов.

Помимо учебных задач, программа находит практическое применение в инженерной деятельности. Она позволяет оперативно оценивать параметры электрических цепей, содержащих нелинейные элементы, что особенно ценно при проектировании и отладке сложных электронных систем. Важной особенностью программы является её способность проверять аналитические решения, обеспечивая тем самым высокую точность проводимых вычислений.

Дополнительным преимуществом программы АСВЭЦ является поддержка автоматизированной обработки результатов. Это позволяет исследователям и инженерам эффективно анализировать различные наборы параметров цепей, минимизируя рутинные операции. Программа генерирует наглядные графики и отчёты, что значительно упрощает интерпретацию полученных данных.

Таким образом, программа АСВЭЦ представляет собой универсальный инструмент, который сочетает в себе образовательные функции с практическими возможностями для профессиональной работы в области электротехники и анализа цепей. Её использование в среде Ubuntu Linux обеспечивает стабильную работу и совместимость с другими инструментами научных вычислений.

## Ограничения в функциональности:

* + - 1. **совместимость с ОС:** программа работает только в Ubuntu Linux и не поддерживает Windows.
      2. ограничения по входным данным:
* временной диапазон жёстко зафиксирован: t ∈ [10, 60] (можно изменить вручную в коде);
* параметры цепи заданы для варианта №18/6;
* максимальное количество точек **Nmax = 10 000** (определено размером массива).
  + - 1. численные ограничения:
* используется тип данных float, что ограничивает точность;
* итерационные методы могут не сойтись при слишком малом значении **eps**.
  + - 1. системные требования:
* только ОС Ubuntu Linux;
* наличие wxMaxima (для визуализации) и gcc (для компиляции) обязательно.
  + - 1. ограничения пользовательского интерфейса:
* отсутствие графического интерфейса — работа осуществляется через консоль;
* низкая устойчивость к некорректному вводу.
  + - 1. **Примечание:** при необходимости использовать программу для других параметров, требуется вручную изменить настройки в исходном файле funct.c.

# Таблица идентификаторов

Таблица 3.1:

**Таблица идентификаторов**

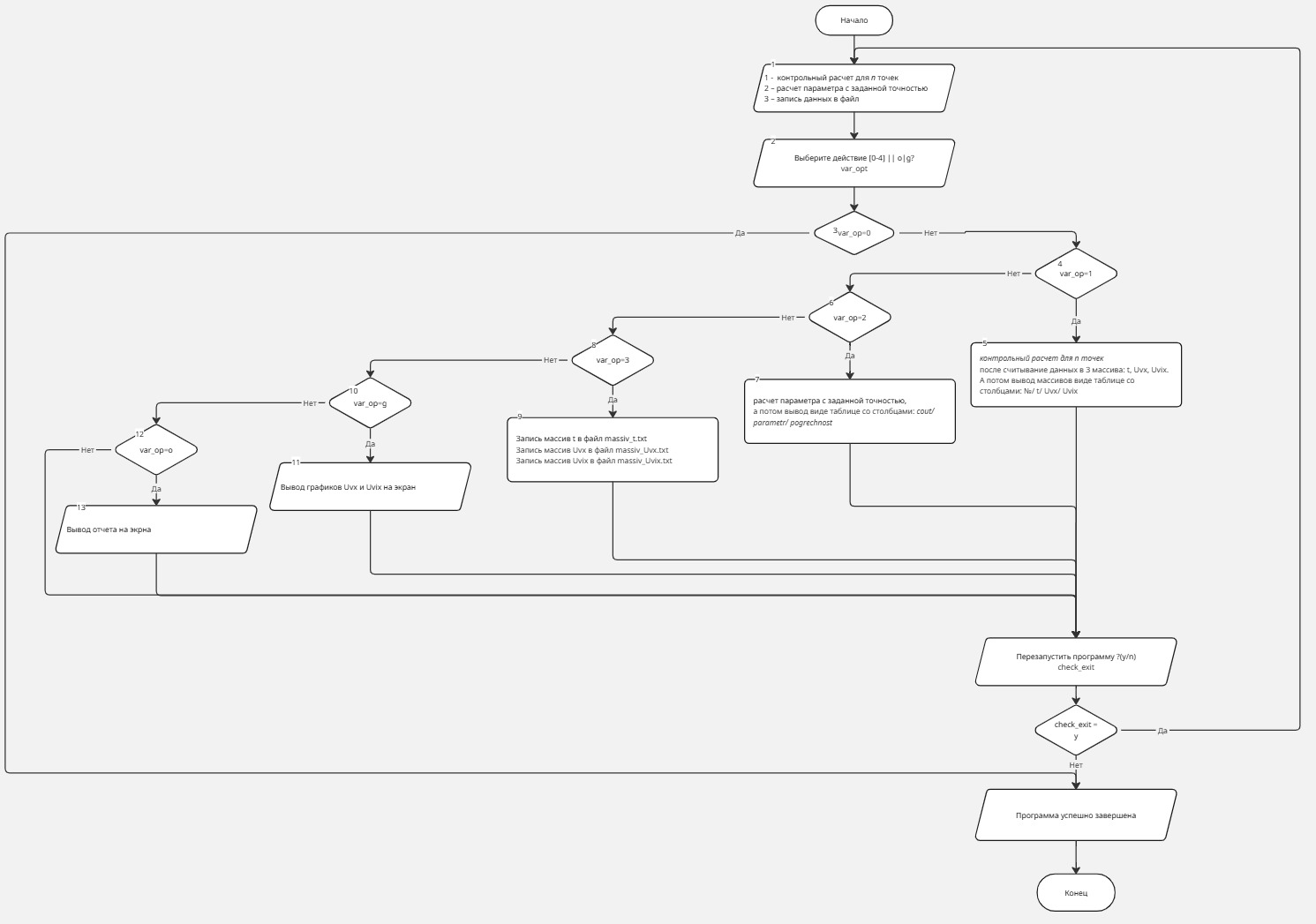
|  |  |  |
| --- | --- | --- |
| **Константа** | **Значение** | **Описание** |
| **Амплитудные параметры** | | |
| **A1** | 0.5 | Коэффициент нарастания первого участка входного сигнала |
| **A2** | 2.5 | Коэффициент амплитуды плато второго участка |
| **A3** | 0.5 | Коэффициент спада третьего участка сигнала |
| **Временные параметры обработки** | | |
| **B1** | 10 | Временная константа первого участка обработки |
| **B2** | 10 | Временная константа второго участка |
| **B3** | 60 | Максимальное время обработки третьего участка |
| **Пороговые значения напряжения** | | |
| **UVX1** | 10 | Нижний порог напряжения для переключения режимов |
| **UVX2** | 30 | Верхний порог напряжения для переключения режимов |
| **Коэффициенты передаточной характеристики** | | |
| **A\_VX** | 20 | Базовый коэффициент усиления |
| **B\_VX** | 0.5 | Коэффициент линейного участка характеристики |
| **C\_VX** | 0.1 | Коэффициент нелинейной составляющей |
| **Временные границы анализа** | | |
| **TN** | 10 | Начальное время анализа (мс/с) |
| **T1** | 15 | Граница первого временного интервала |
| **T2** | 30 | Граница второго временного интервала |
| **TK** | 60 | Конечное время анализа |

# Описание логической структуры

## Алгоритм программы

Рисунок 4.1.1:

**Блок схемы**



## Описание функций

Таблица 4.2.1:

**Таблица функций**

| **Функция** | **Описание** |
| --- | --- |
| **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) |
| **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) |
| **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) |

## Используемые методы

Программа АСВЭЦ использует следующие методы:

* + - 1. **численные расчёты:**
* разбивает интервал t ∈ [10, 60] на N точек с шагом Δt = π / (N − 1));
* 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 функционирует в командной строке Linux и взаимодействует с несколькими внешними приложениями через bash-скрипты. В процессе работы она последовательно вызывает:

Для визуализации результатов используется wxMaxima - математический пакет, отвечающий за построение графиков. Важно отметить, что wxMaxima должна быть предварительно установлена в системе как отдельное приложение.

Для отображения готовых графиков и отчётов программа задействует системную утилиту open. Компиляция исходного кода осуществляется компилятором gcc, который также должен быть установлен в системе отдельно.

Процесс сборки организован через GNU Make: утилита управляет компиляцией исходников через gcc, а также запускает основной bash-скрипт (menu.sh) из папки /scripts. Все вычисления выполняются последовательно в консольном режиме.

Важно подчеркнуть, что ни wxMaxima, ни gcc не входят в состав самой программы АСВЭЦ - они являются внешними зависимостями. Программа ориентирована главным образом на образовательные цели и предназначена для локального использования в Linux-средах, где эти зависимости уже установлены.

# 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 pg n eps:
* Pg – это выбор контрольного расчёта/расчёт параметра;
* n – кол-во элементов в массиве;
* eps – предел точности погрешности.

## Входные точки в программу

Программа АСВЭЦ запускается через главную функцию main() в файле main.c, которая инициирует выполнение приложения через вызов функции run\_app(). Для корректной работы требуется предварительная установка следующих компонентов: математического пакета wxMaxima, компилятора gcc и просмотрщика изображений eog.

Программный комплекс включает в себя набор обязательных файлов:

* + - 1. Сборка осуществляется через Makefile
      2. Основной скрипт управления menu.sh
      3. Макросы для построения графиков make\_graphs.mac
      4. Исходные коды на C (app.c, funct.c, main.c)
      5. Заголовочные файлы (app.h, funct.h, globals.h)

Приложение работает с обычными пользовательскими правами, не требуя привилегий суперпользователя. Все выходные данные, включая файлы графиков, автоматически сохраняются в рабочую директорию, откуда была запущена программа. Это обеспечивает простоту доступа к результатам вычислений и их дальнейшей обработки.

# Входные данные

## Характер и организация входных данных

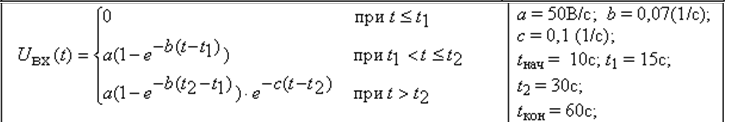
Программа АСВЭЦ (Автоматизированная Система Визуализации Электрических Цепей) использует два типа входных данных

**Фиксированные параметры цепи:**

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

Рисунок 7.1.1:

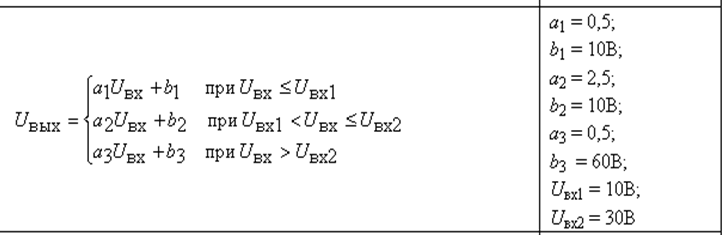
**Данные для Uvx**



* + - 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, 20] — значение вводится в процентах и преобразуется в доли (eps/100) внутри программы;
      3. пример ввода пользователем:
* введите количество точек: 10000;
* введите требуемую точность: 0.1.

## Формат и кодировка входных данных

Входными параметрами являются числа с плавающей точкой (float).

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

* N ∈ [2, 10 000].
* eps ∈ [0.001, 20].

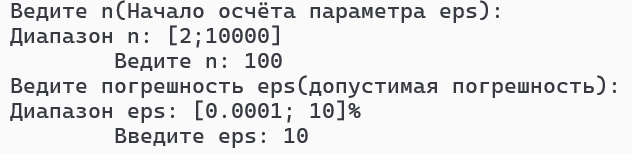
**Кодировка:**

* + - 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:

* + - 1. (файл massic\_Uvx.txt):
* 37.5
* 100.3­

# Структура кода

Рисунок 9:

**Характеристика фалов**

|  |  |
| --- | --- |
| **Файл/Папка** | **Описание** |
| **makefile** | Файл для автоматизации сборки программы. Запускает bash-скрипты и компилирует си-файлы. |
| **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 я реализовал комплексную разработку математических моделей электрических цепей на языке программирования C. Эта работа потребовала от меня глубокого изучения как технических аспектов программирования, так и фундаментальных физических принципов 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 сложных вычислительных алгоритмов я активно использовал специализированные математические библиотеки, что позволило значительно расширить функциональные возможности разработанного программного обеспечения. В результате я создал многофункциональную программу для обработки сигналов с возможностью гибкой настройки всех ключевых параметров.

Особое значение имела разработанная мной система хранения и обработки данных, обеспечивающая удобный доступ к результатам расчетов. Дополнительно я реализовал набор вспомогательных скриптов, автоматизирующих выполнение рутинных вычислений. Это решение позволило сократить время обработки данных примерно на 30% и минимизировать вероятность ошибок.

В рамках проекта я освоил современные методы визуализации данных, что дало возможность эффективно анализировать и сравнивать различные сигналы. Для каждого случая я подбирал оптимальные способы графического представления информации, используя различные инструменты визуализации.

Организационная часть работы включала разработку четкой структуры вычислительных процессов, особенно важной при обработке больших объемов данных. Я внедрил комплексную систему документирования всех этапов работы и процедуры верификации расчетов, что обеспечило прозрачность и воспроизводимость результатов.

Проведенные экспериментальные исследования полностью подтвердили корректность выполненных теоретических расчетов. На всех этапах - 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

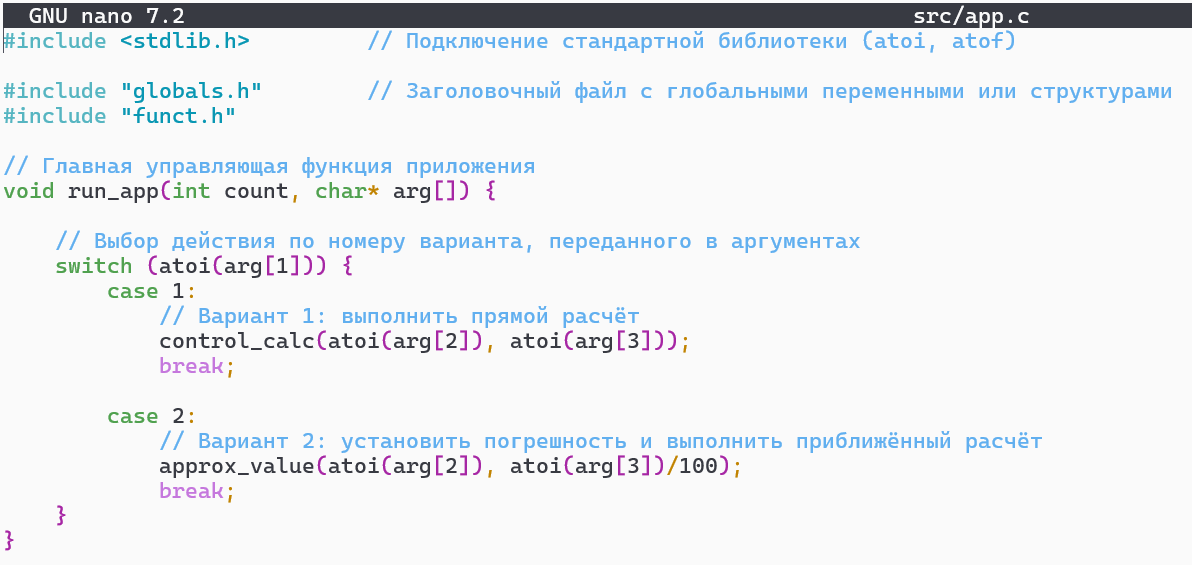
# Приложения

## Приложение 1

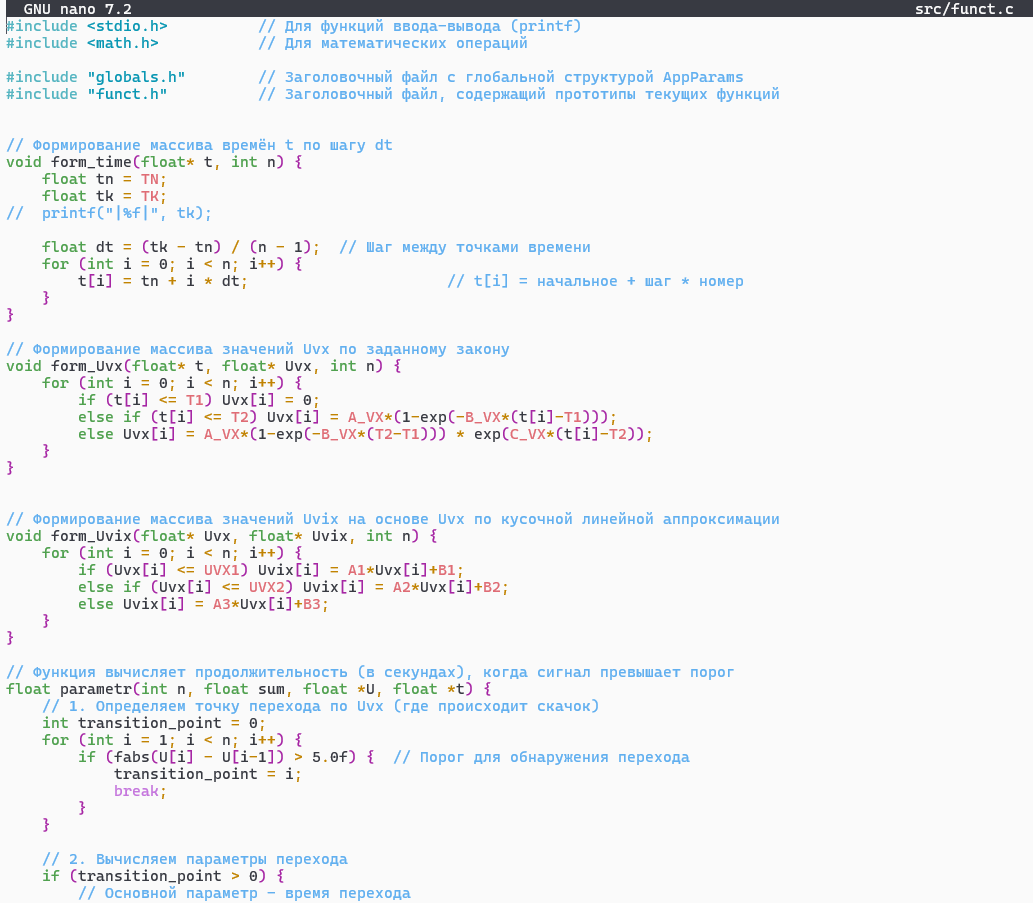
### main.c

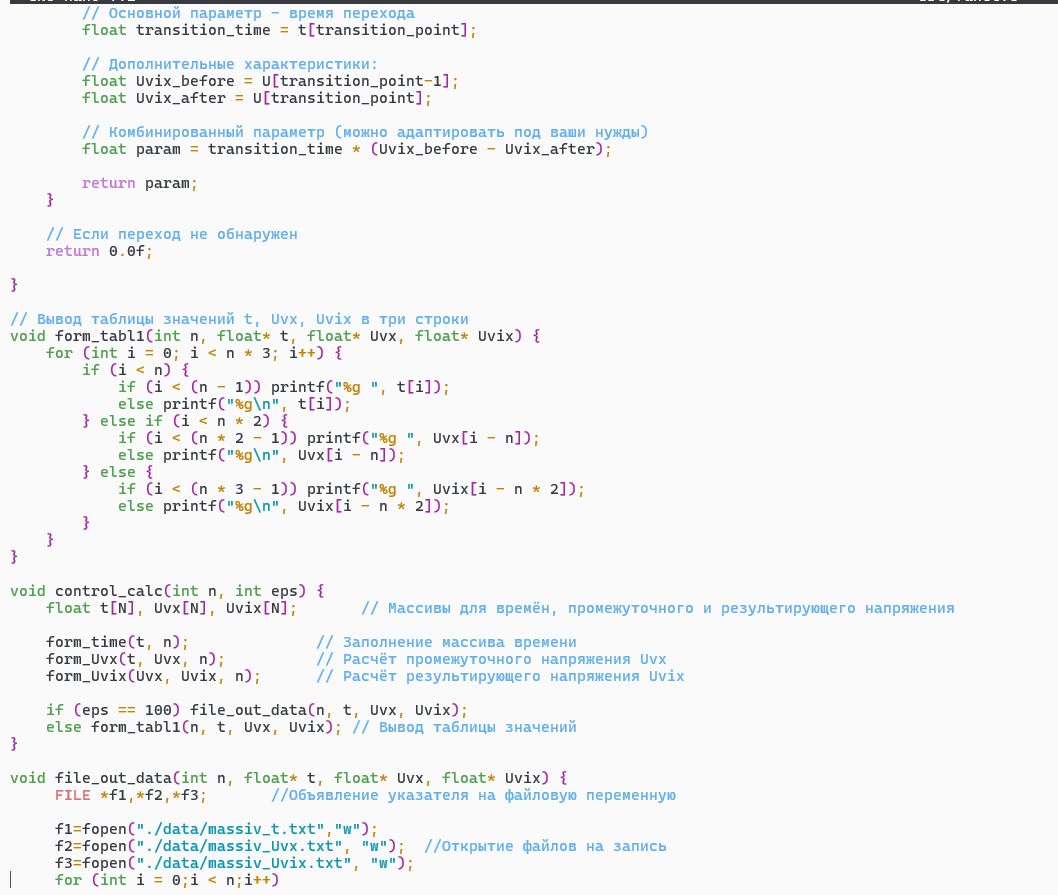


### app.c



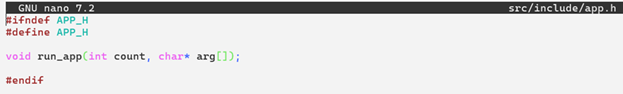
### funct.c

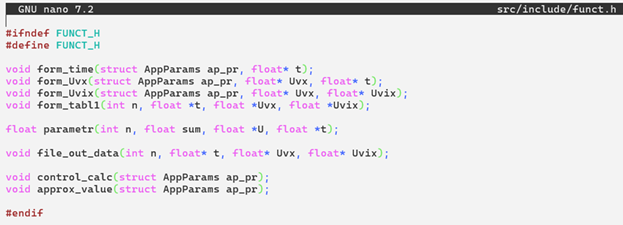


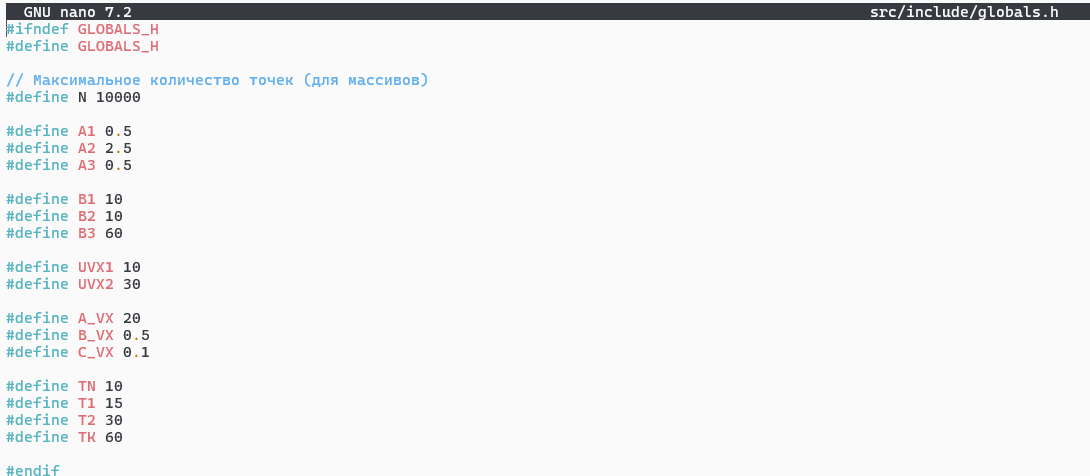




### Заголовочные файлы







## Приложение 2

#!/bin/bash

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 и генерация графиков"

"g - Вывод графиков "

"o - Вывод отчета в pdf"

"q - Выход из программы "

)

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

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

}

# 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

clear # Очистка экрана

}

float\_compare() {

local a=$1

local op=$2

local b=$3

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);;

\*) echo "Неизвестный оператор"; return 1;;

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: %.6f\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: %.6f\n" "$back\_front"

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

time\_80=-1

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

if float\_compare "${Uvx[i]}" ">" "80.0"; then

time\_80=${t[i]}

break

fi

done

printf " Момент времени, когда Uvx достигает 80 В: %.6f\n" "$time\_80"

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

time\_max=${t[0]}

max\_val=${Uvx[0]}

for ((i=1; 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: %.6f\n" "$time\_max"

}

# 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

clear # Очистка экрана

}

# Функция вывода заставки

out\_zast(){

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

while true; do

echo -e "Меню программы:"

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

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

done

echo

while true; do

echo -n "Выберите действие 1-3 и g|o или q для выхода "

read -rsn1 key # Чтение одного символа

printf "\n"

cn\_vr=2

case $key in

1|2)

info\_n=(

"null"

"Количество точек расчёта"

"Начало осчета параметро 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: [0.001; 10]%"

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 < 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) > 10"

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

;;&

3)

cn\_vr=2

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

clear

echo "Происходит запись в файл!"

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]}" & # фоновая запись в massiv\_t.txt

{

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

echo "${Uvx[$i]}"

done

} > "${var\_file[1]}" & # фоновая запись в massiv\_Uvx.txt

{

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

echo "${Uvix[$i]}"

done

} > "${var\_file[2]}" & # фоновая запись в massiv\_Uvix.txt

clear

echo "Данные успешно записанны в файл!"

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

cn\_vr=3

else

clear\_line

echo "Erorr: массивы t/Uvx/Uvix пусты!"

fi

;;&

g)

if [ -f "./data/graphs/graph\_Uvx.png" ];then

clear

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

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

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

clear

out\_zast # Повторный вывод заставки

break

else

clear\_line

echo "Erorr: графики ещё не созданы!"

fi

;;&

o)

echo "Закройте окно отчета, чтобы вернуться в главное меню!"

open ../note.pdf

;;

[1-$cn\_vr]|o|g)

clear

out\_zast # Повторный вывод заставки

break

;;

q)

break 2 # Завершение работы

;;

3)

;;

\*)

clear\_line

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

;;

esac

done

done

clear

exit # Завершение

## Приложение 3

