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.К. _____

Дата выполнения: «29» Май
Проверил:
Хазиев Н.Н. _____

Санкт-Петербург
2025

Оглавление

Плисов Кирилл Константинович ИКПИ - 41

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: C (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.

Плисов Кирилл Константинович ИКПИ - 41

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

$U_{вх}(t) = \begin{cases} 0 & \text{при } t \le t_1 \\ a(1-e^{-b(t-t_1)}) & \text{при } t_1 < t \le t_2 \\ a(1-e^{-b(t_2-t_1)}) \cdot e^{-c(t-t_2)} & \text{при } t > t_2 \end{cases}$	$a = 50$В/с; $b = 0{,}07$(1/с); $c = 0{,}1$ (1/с); $t_{нач} = 10$с; $t_1 = 15$с; $t_2 = 30$с; $t_{кон} = 60$с;
$U_{вых} = \begin{cases} a_1 U_{вх} + b_1 & \text{при } U_{вх} \le U_{вх1} \\ a_2 U_{вх} + b_2 & \text{при } U_{вх1} < U_{вх} \le U_{вх2} \\ a_3 U_{вх} + b_3 & \text{при } U_{вх} > U_{вх2} \end{cases}$	$a_1 = 0{,}5$; $b_1 = 10$В; $a_2 = 2{,}5$; $b_2 = 10$В; $a_3 = 0{,}5$; $b_3 = 60$В; $U_{вх1} = 10$В; $U_{вх2} = 30$В

_ 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.

1 Общие сведения

1.1 Обозначение и наименование программы

Для корректной работы программа требует установленную русифицированную версию операционной системы Ubuntu Linux.

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

_ **sudo apt-get install wxmaxima;**

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

_ **sudo apt update;**

_ **sudo apt install build-essential;**

1.2 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.

2 Функциональное назначение

2.1 Решаемые задачи

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

Моделирование сигналов:

_ расчёт функции входного напряжения Uвх(t), заданной в аналитической форме;

_ вычисление выходного напряжения Uвых(t), используя кусочно-линейную передаточную характеристику.

Графическая визуализация:

_ построение графиков Uвх(t) и Uвых(t) с помощью wxMaxima;

_ экспорт полученных данных в форматы, совместимые с другими пакетами (например, GNU Plot или LibreOffice Calc).

2.2 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.

2.3 Ограничения в функциональности:

_ **совместимость с ОС:** программа работает только в Ubuntu Linux и не поддерживает Windows.

_ ограничения по входным данным:

- временной диапазон жёстко зафиксирован: $t \in [10, 60]$ (можно изменить вручную в коде);
- параметры цепи заданы для варианта №18/6;

- максимальное количество точек **Nmax = 10 000** (определено размером массива).

_ численные ограничения:

- используется тип данных float, что ограничивает точность;
- итерационные методы могут не сойтись при слишком малом значении **eps**.

_ системные требования:

- только ОС Ubuntu Linux;
- наличие wxMaxima (для визуализации) и gcc (для компиляции) обязательно.

_ ограничения пользовательского интерфейса:

- отсутствие графического интерфейса — работа осуществляется через консоль;
- низкая устойчивость к некорректному вводу.

_ **Примечание:** при необходимости использовать программу для других параметров, требуется вручную изменить настройки в исходном файле funct.c.

3 Таблица идентификаторов

Таблица 3.1:

Таблица идентификаторов

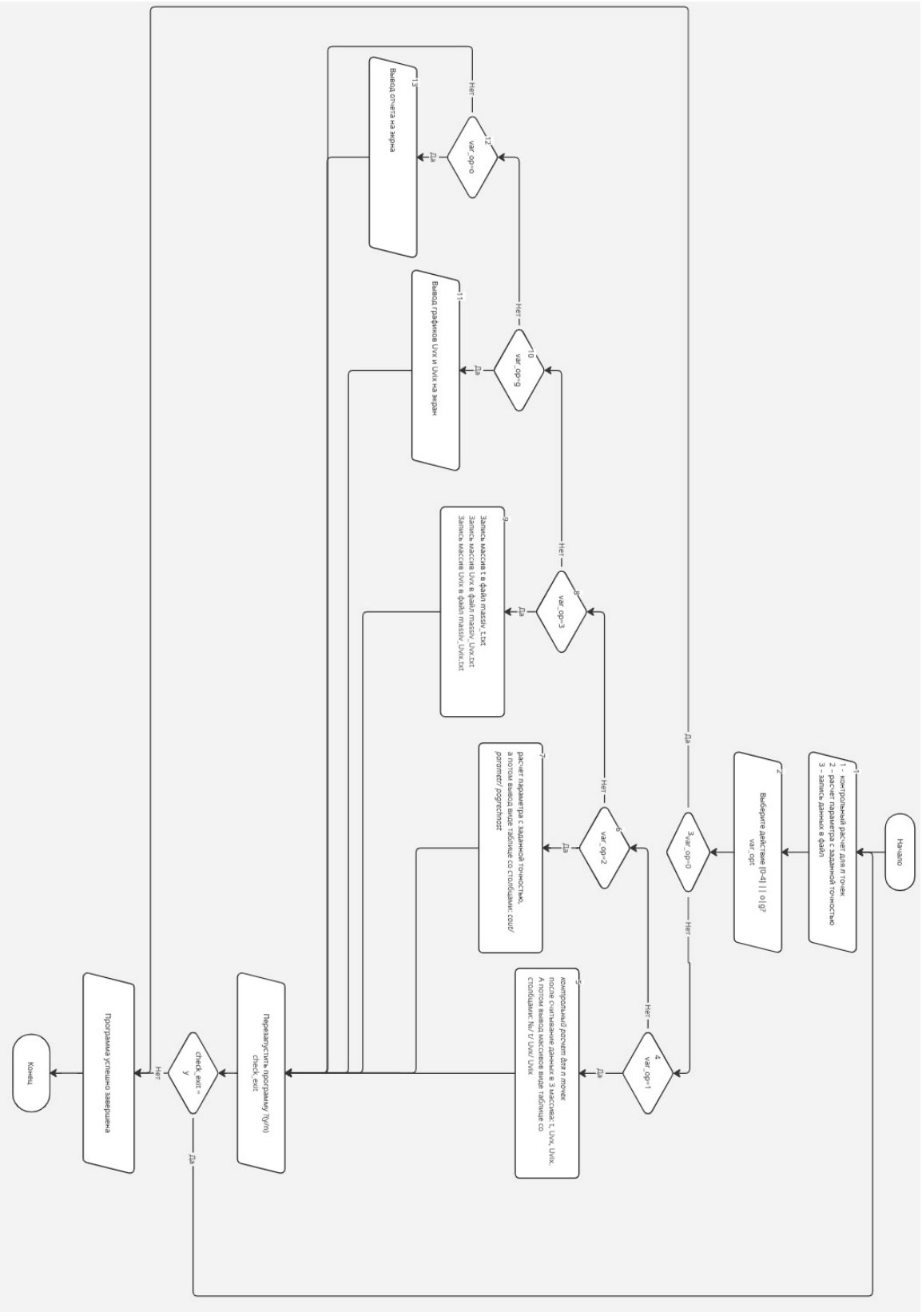
Константа	Значение	Описание
Амплитудные параметры		
A1	0.5	Коэффициент нарастания первого участка входного сигнала
A2	2.5	Коэффициент амплитуды плато второго участка
A3	0.5	Коэффициент спада третьего участка сигнала
Временные параметры обработки		
B1	10	Временная константа первого участка обработки
B2	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

4 Описание логической структуры

4.1 Алгоритм программы

Рисунок 4.1.1:

Блок схемы



4.2 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)

4.3 Используемые методы

Программа АСВЭЦ использует следующие методы:

_ численные расчёты:

- разбивает интервал $t \in [10, 60]$ на N точек с шагом $\Delta t = \pi / (N - 1)$);
- используются функции pow() из библиотеки math.h;

_ 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).

4.4 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.

5 Используемые технические средства

Минимальные и рекомендуемые требования к техническим средствам, которые соответствуют программе «Анализ сигнала на выходе электрической 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	-

6 Вызов и загрузка

6.1 Способ вызова программы с соответствующего носителя данных

_ программа запускается вручную через терминал Linux следующим образом: make run;

_ также возможен прямой вызов программы без меню, в случае его неработоспособности:

_ ./prg pg n eps:

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

6.2 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:

_ Сборка осуществляется через Makefile

_ Основной скрипт управления menu.sh

_ Макросы для построения графиков make_graphs.mac

_ Исходные коды на C (app.c, funct.c, main.c)

_ 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.

7 Входные данные

7.1 Характер и организация входных данных

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

Фиксированные параметры цепи:

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

Рисунок 7.1.1:

Данные для Uvx

$U_{вх}(t) = \begin{cases} 0 & \text{при } t \le t_1 \\ a(1-e^{-b(t-t_1)}) & \text{при } t_1 < t \le t_2 \\ a(1-e^{-b(t_2-t_1)}) \cdot e^{-c(t-t_2)} & \text{при } t > t_2 \end{cases}$	$a = 50В/с;\ b = 0{,}07(1/с);$ $c = 0{,}1\ (1/с);$ $t_{нач} = 10с;\ t_1 = 15с;$ $t_2 = 30с;$ $t_{кон} = 60с;$

_ 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

$U_{вых} = \begin{cases} a_1 U_{вх} + b_1 & \text{при } U_{вх} \le U_{вх1} \\ a_2 U_{вх} + b_2 & \text{при } U_{вх1} < U_{вх} \le U_{вх2} \\ a_3 U_{вх} + b_3 & \text{при } U_{вх} > U_{вх2} \end{cases}$	$a_1 = 0{,}5;$ $b_1 = 10В;$ $a_2 = 2{,}5;$ $b_2 = 10В;$ $a_3 = 0{,}5;$ $b_3 = 60В;$ $U_{вх1} = 10В;$ $U_{вх2} = 30В$

Пользовательские параметры (вводятся через консоль):

_ количество точек N — задаёт разрешение графика (число временных отсчётов);

_ точность расчёта eps — используется для приближённого метода (вариант 2), определяя относительную погрешность при расчёте параметра;

_ подготовка входных данных не требуется — все вспомогательные параметры и данные генерируются внутри программы автоматически.

Диапазоны допустимых значений:

_ N ∈ [2, 10 000] — ограничение задано директивой #define N 10000.

_ eps ∈ [0.001, 20] — значение вводится в процентах и преобразуется в доли (eps/100) внутри программы;

_ пример ввода пользователем:

- введите количество точек: 10000;
- введите требуемую точность: 0.1.

7.2 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:

_ 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

```
Ведите n(Начало осчёта параметра eps):
Диапазон n: [2;10000]
        Ведите n: 100
Ведите погрешность eps(допустимая погрешность):
Диапазон eps: [0.0001; 10]%
        Введите eps: 10
```

8 Выходные данные

8.1 Характер и организация выходных данных

Программа АСВЭЦ генерирует два типа выходных данных:

_ 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.

_ консольный вывод информации о программе, о значениях рассчитанных параметров, подсказки для пользователя;

_ 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.

8.2 Контрольный расчёт

Таблица 8.2.1:

Таблица “Контрольный расчет для n точек”

Контрольный расчет для n точек				Параметры
№	t	Uvx	Uvix	При количестве контрольных точек n=25
1	10.0	0.0	10.0	
2	12.1	0.0	10.0	
3	14.2	0.0	10.0	
4	16.2	9.3	14.6	
5	18.3	16.2	50.6	
6	20.4	18.7	56.7	
7	22.5	19.5	58.8	
8	24.6	19.8	59.6	
9	26.7	19.9	59.9	
10	28.8	20.0	59.9	
11	30.8	21.7	64.3	
12	32.9	26.8	76.9	
13	35.0	33.0	76.5	
14	37.1	40.6	80.3	
15	39.2	50.0	85.0	
16	41.2	61.6	90.8	
17	43.3	75.8	97.9	
18	45.4	93.4	106.7	
19	47.5	115.0	117.5	
20	49.6	141.7	130.8	
21	51.7	174.5	147.2	
22	53.8	214.9	167.5	
23	55.8	264.7	192.3	
24	57.9	326.0	223.0	
25	60.0	401.5	260.7	

Таблица 8.2.2:

Таблица “Расчёт параметра с заданной точностью”

Расчёт параметра с заданной точностью	Параметры
Длительность импульса сигнала: 44.444800 Длительность заднего фронта импульса: 22.222400 Момент времени, когда Uvx достигает 80 В: 10.000000 Момент времени максимального значения Uvx: 10.000000 n parametr pogrechnost 10 -954.641 100.000000% 20 -662.243 44.153000% 40 -404.953 63.536000% 80 -447.771 9.563000% 160 -378.970 18.155000% 320 -341.601 10.939000% 640 -336.437 1.535000% 1280 -333.896 0.761000%	n = 10, eps = 1%

Таблица 8.2.3:

Таблица “График Uvx и параметры”

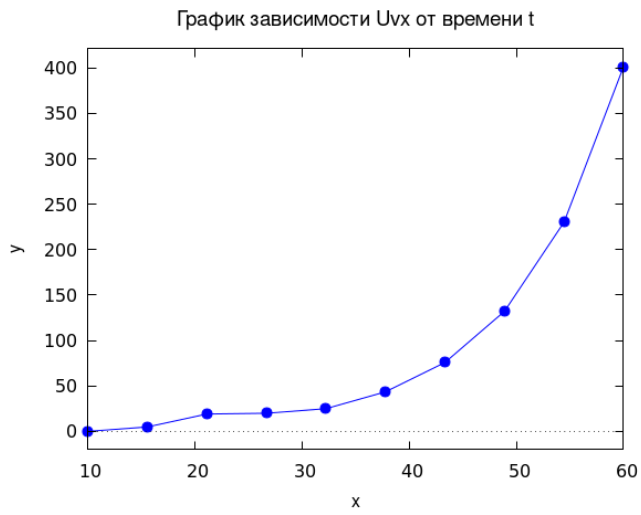
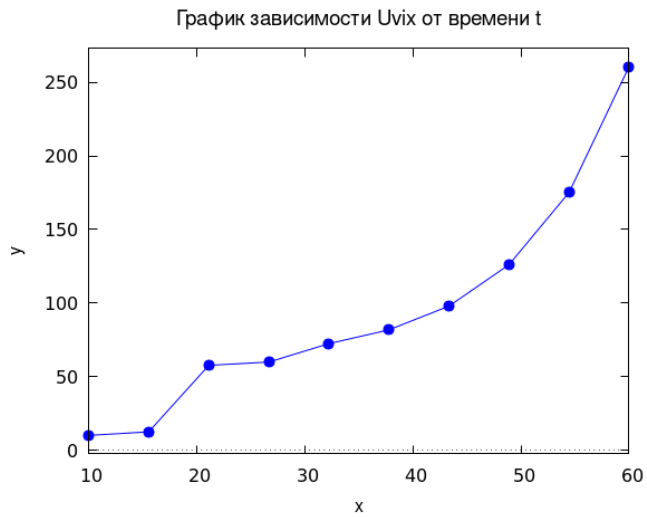
График Uvx	Параметры
	$a = 50В/с;\ b = 0,07(1/с);$ $c = 0,1\ (1/с);$ $t_{нач} = 10с;\ t_1 = 15с;$ $t_2 = 30с;$ $t_{кон} = 60с;$ n = 25

Таблица 8.2.4:

Таблица “График Uvix и параметры”

График Uvix	Параметры
	$a_1 = 0,5;$ $b_1 = 10В;$ $a_2 = 2,5;$ $b_2 = 10В;$ $a_3 = 0,5;$ $b_3 = 60В;$ $U_{вх1} = 10В;$ $U_{вх2} = 30В$ n = 25

8.3 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.

Плисов Кирилл Константинович ИКПИ - 41

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

Пример выходных данных:

_ (файл massic_Uvx.txt):

- 37.5
- 100.3

9 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 Си находится в приложение 1.

Код Bash находится в приложение 2.

Код Wxmaxima находится в приложение 3.

10 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.

11 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 С. — 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/

12 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.xxx).
- **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 10^6 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

13 Приложения

13.1 Приложение 1

a main.c

```c
#include "app.h"

int main(int count, char* arg[]
    run_app(count, arg);
    return 0;
}
```

b app.c

```
 GNU nano 7.2                                                    src/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[]) {

    // Выбор действия по номеру варианта, переданного в аргументах
    switch (atoi(arg[1])) {
        case 1:
            // Вариант 1: выполнить прямой расчёт
            control_calc(atoi(arg[2]), atoi(arg[3]));
            break;

        case 2:
            // Вариант 2: установить погрешность и выполнить приближённый расчёт
            approx_value(atoi(arg[2]), atoi(arg[3])/100);
            break;
    }
}
```

c funct.c

GNU nano 7.2 src/funct.c

```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(float* t, int n) {
    float tn = TN;
    float tk = TK;
//  printf("|%f|", tk);

    float dt = (tk - tn) / (n - 1);  // Шаг между точками времени
    for (int i = 0; i < n; i++) {
        t[i] = tn + i * dt;                     // t[i] = начальное + шаг * номер
    }
}

// Формирование массива значений Uvx по заданному закону
void form_Uvx(float* t, float* Uvx, int n) {
    for (int i = 0; i < n; i++) {
        if (t[i] <= T1) Uvx[i] = 0;
        else if (t[i] <= T2) Uvx[i] = A_VX*(1-exp(-B_VX*(t[i]-T1)));
        else Uvx[i] = A_VX*(1-exp(-B_VX*(T2-T1))) * exp(C_VX*(t[i]-T2));
    }
}


// Формирование массива значений Uvix на основе Uvx по кусочной линейной аппроксимации
void form_Uvix(float* Uvx, float* Uvix, int n) {
    for (int i = 0; i < n; i++) {
        if (Uvx[i] <= UVX1) Uvix[i] = A1*Uvx[i]+B1;
        else if (Uvx[i] <= UVX2) Uvix[i] = A2*Uvx[i]+B2;
        else Uvix[i] = A3*Uvx[i]+B3;
    }
}

// Функция вычисляет продолжительность (в секундах), когда сигнал превышает порог
float parametr(int n, float sum, float *U, float *t) {
    // 1. Определяем точку перехода по Uvx (где происходит скачок)
    int transition_point = 0;
    for (int i = 1; i < n; i++) {
        if (fabs(U[i] - U[i-1]) > 5.0f) {  // Порог для обнаружения перехода
            transition_point = i;
            break;
        }
    }

    // 2. Вычисляем параметры перехода
    if (transition_point > 0) {
        // Основной параметр - время перехода
```

```
        // Основной параметр - время перехода
        float transition_time = t[transition_point];

        // Дополнительные характеристики:
        float Uvix_before = U[transition_point-1];
        float Uvix_after = U[transition_point];

        // Комбинированный параметр (можно адаптировать под ваши нужды)
        float param = transition_time * (Uvix_before - Uvix_after);

        return param;
    }

    // Если переход не обнаружен
    return 0.0f;

}

// 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("%g ", t[i]);
            else printf("%g\n", t[i]);
        } else if (i < n * 2) {
            if (i < (n * 2 - 1)) printf("%g ", Uvx[i - n]);
            else printf("%g\n", Uvx[i - n]);
        } else {
            if (i < (n * 3 - 1)) printf("%g ", Uvix[i - n * 2]);
            else printf("%g\n", Uvix[i - n * 2]);
        }
    }
}

void control_calc(int n, int eps) {
    float t[N], Uvx[N], Uvix[N];       // Массивы для времён, промежуточного и результирующего напряжения

    form_time(t, n);              // Заполнение массива времени
    form_Uvx(t, Uvx, n);          // Расчёт промежуточного напряжения Uvx
    form_Uvix(Uvx, Uvix, n);      // Расчёт результирующего напряжения Uvix

    if (eps == 100) file_out_data(n, t, Uvx, Uvix);
    else form_tabl1(n, t, Uvx, 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 %g",t[i]);
        fprintf(f2,"\n %g", Uvx[i]);          //Запись данных в файл
        fprintf(f3,"\n%g",Uvix[i]);
      }
      fclose(f1);
      fclose(f2);                                          //Закрытие файлов
      fclose(f3);
}

// Функция приближённого расчёта значения параметра с заданной точностью
void approx_value(int n, int eps) {
    float t[N], Uvx[N], Uvix[N];
    float p = 1;
    float par = 1e10;
    float par1 = 0;

    printf("n   parametr   pogrechnost\n");

    while (p > eps && n < N) {
        form_time(t, n);
        form_Uvx(t, Uvx, n);
        form_Uvix(Uvx, Uvix, n);

        par1 = parametr(n, 0, Uvix, t);
        p = fabs(par - par1) / fabs(par1);
        if (p > 1) p = 1;

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

        par = par1;
        n = 2 * n;
    }

}
```

d Заголовочные файлы

```
 GNU nano 7.2                                                src/include/app.h
#ifndef APP_H
#define APP_H

void run_app(int count, char* arg[]);

#endif
```

```
 GNU nano 7.2                                                src/include/funct.h
#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);

void file_out_data(int n, float* t, float* Uvx, float* Uvix);

void control_calc(struct AppParams ap_pr);
void approx_value(struct AppParams ap_pr);

#endif
```

```
 GNU nano 7.2                                                    src/include/globals.h
#ifndef GLOBALS_H
#define GLOBALS_H

// Максимальное количество точек (для массивов)
#define N 10000

#define A1 0.5
#define A2 2.5
#define A3 0.5

#define B1 10
#define B2 10
#define B3 60

#define UVX1 10
#define UVX2 30

#define A_VX 20
#define B_VX 0.5
#define C_VX 0.1

#define TN 10
#define T1 15
#define T2 30
#define TK 60

#endif
```

13.2 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
```

```
Плисов Кирилл Константинович ИКПИ - 41
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
```

13.3 Приложение 3

```
 GNU nano 7.2                                                    scripts/Wxmax_scr/make_graphs.mac
/* 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]
)$
```