ФГБОУ ВО «Уфимский университет науки и технологий»

Институт информатики, математики и робототехники

Кафедра высокопроизводительных вычислений и дифференциальных уравнений

**ОТЧЕТ О ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

**ТИП ПРАКТИКИ**

для выполнения производственной работы

**ОБУЧАЮЩЕГОСЯ**

3 курса группы МКН-318Б

Халитовой Айгуль Азатовны

(фамилия имя отчество в род. п.)

|  |  |
| --- | --- |
| Уровень образования: | высшее образование – бакалавриат |
| Направление подготовки (специальность) | 02.03.01 Математика и компьютерные науки |
| Направленность (профиль) программы | Анализ данных и компьютерное моделирование |
| Срок проведения практики: | с 12 мая 2025 по 7 июня 2025 |

**1. МЕТОДИЧЕСКИЕ УКАЗАНИЯ**

1. База практики – профильная организация или структурное подразделение УУНиТ.
2. Обучающийся – физическое лицо, осваивающее образовательную программу среднего профессионального или высшего образования.
3. Вид практики – учебная, производственная.
4. Каждый обучающийся, находящийся на практике, обязан вести отчет по практике.
5. Отчет по практике служит основным и необходимым материалом для составления обучающимся отчета о своей работе на базе практики.
6. Заполнение отчета по практике производится регулярно, аккуратно и является средством самоконтроля. Отчет можно заполнять рукописным и (или) машинописным способами.
7. Иллюстративный материал (чертежи, схемы, тексты и т.п.), а также выписки из инструкций, правил и других материалов могут быть выполнены на отдельных листах и приложены к отчету.
8. Записи в отчете о практике должны производиться в соответствии с программой по конкретному виду практики.
9. После окончания практики обучающийся должен подписать отчет у руководителя практики, руководителя от базы практики и сдать свой отчет по практике вместе с приложениями (при наличии) на кафедру.
10. При отсутствии сведений в соответствующих строках ставится прочерк.

**2. ОБЩИЕ ПОЛОЖЕНИЯ**

|  |  |
| --- | --- |
| Фамилия, инициалы, должность руководителя практики от факультета (института, колледжа, техникума) | Заместитель директора по развитию проектов и программ Белова А.С. |
| Фамилия, инициалы, должность руководителя практики от кафедры | Поречный С.С., доцент кафедры ВВиДУ |
| Полное наименование базы практики | ООО «РН-БашНИПИнефть» |
| Наименование структурного подразделения базы практики | Отдел разработки геологических проектов |
| Адрес базы практики (индекс, субъект РФ, район, населенный пункт, улица, дом, офис) | 450103, респ. Башкортостан, г. Уфа, ул. Бехтерева 3/1 |
| Фамилия, инициалы, должность руководителя практики от профильной организации | Феоктистов Б.А., главный специалист |
| Телефон руководителя практики от базы практики | 83472936010, доп. 3455 |

**3. РАБОЧИЙ ГРАФИК (ПЛАН) ПРОВЕДЕНИЯ ПРАКТИКИ**

Срок проведения практики: с 12 мая 2025 по 7 июня 2025

|  |  |  |  |
| --- | --- | --- | --- |
| № | Разделы (этапы) практики | Виды и содержание работ, в т.ч. самостоятельная работа обучающегося в соответствии с программой практики | График (план) проведения практики  (начало – окончание) |
| 1. | Подготовительный этап | – организационное собрание;  – установочная лекция;  – получение индивидуального задания на практику;  – проведение инструктажа обучающегося по ознакомлению с требованиями охраны труда, техники безопасности, пожарной безопасности, а также правилами внутреннего трудового распорядка. | 12.05.2025 –15.05.2025 |
| 2. | Основной этап | Работа обучающихся в профильной организации  – выполнение индивидуального задания | 16.05.2025 – 03.06.2025 |
| 3. | Заключительный этап | – подготовка и оформление отчёта по практике, содержащего итоги прохождения практики;  – подготовка к защите, в том числе оформление презентации, и защита отчета. | 04.06.2025 – 07.06.2025 |

|  |  |
| --- | --- |
| Руководитель практики от кафедры | \_\_\_\_\_\_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись И.О. Фамилия |
| Руководитель практики от профильной организации [[1]](#footnote-1) | \_\_\_\_\_\_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись И.О. Фамилия |

**4. ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ**

Содержание задания на практику (перечень подлежащих рассмотрению вопросов, выполняемых работ, связанных с будущей профессиональной деятельностью):

1. Разработать алгоритм создания ступенчатого графика по непрерывным данным.

2. Реализовать форматированный ввод и вывод данных на языке Python.

3. Реализовать вычислительную часть на языке C++.

4. Реализовать интеграцию вычислительного ядра на C++ с языком Python.

5. Реализовать пользовательский интерфейс с помощью фреймворка Qt Designer и библиотеки PySide6.

6. Создать исполняемый файл (.exe) для реализованной программы.

|  |  |
| --- | --- |
| Руководитель практики от кафедры | \_\_\_\_\_\_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись И.О. Фамилия |
| Руководитель практики от профильной организации | \_\_\_\_\_\_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись И.О. Фамилия |
| ОЗНАКОМЛЕН:  Обучающийся | \_\_\_\_\_\_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись И.О. Фамилия |

**5. ИНСТРУКТАЖ ПО ОХРАНЕ ТРУДА**

Наименование и реквизиты локального нормативного акта, регламентирующего систему управления охраной труда, техники безопасности, пожарной безопасности профильной организации:

Политика компании № ПЗ-05 П-11 "В области пожарной безопасности, охраны труда и окружающей среды. Реестр опасностей, рисков и мер управления в области промышленной безопасности, охраны труда и окружающей среды ООО "РН-БашНИПИнефть" стандарт компании "Интегрированная система управления ПБОТОС". Положение компании № ПЗ-05 Р-0809 "Система обеспечения пожарной безопасности компании".

Наименование и реквизиты локального нормативного акта, устанавливающего правила внутреннего трудового распорядка профильной организации:

Положение ООО «РН-БашНИПИнефть» «Правила внутреннего трудового распорядка».

Инструкция о мерах пожарной безопасности в Уфимском университете науки и технологий, утвержденная приказом УУНиТ.

Правила внутреннего трудового распорядка обучающихся в Уфимском университете науки и технологий, утвержденные приказом УУНиТ от 23.05.2023 №1285 " Об утверждении Правил внутреннего распорядка обучающихся".

Перед началом практики инструктаж по ознакомлению с требованиями охраны труда, техники безопасности, пожарной безопасности, а также правилами внутреннего трудового распорядка прошел:

обучающийся \_\_\_\_\_\_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись И.О. Фамилия

Перед началом практики инструктаж обучающегося по ознакомлению с требованиями охраны труда, техники безопасности, пожарной безопасности, а также правилами внутреннего трудового распорядка провел:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

должность подпись И.О. Фамилия

**6. ДНЕВНИК РАБОТЫ ОБУЧАЮЩЕГОСЯ**

|  |  |
| --- | --- |
| Дата | Информация о проделанной работе, использованные источники и литература (при наличии) |
| 12.05.2025 | Прохождение инструктажей по технике безопасности и охране труда. |
| 13.05.2025 | Получение задания. |
| 14.05.2025-16.05.2025 | Изучение профильной литературы и библиотек для разработки программы. |
| 19.05.2025-20.05.2025 | Разработка структуры графического интерфейса пользователя. |
| 21.05.2025-23.05.2025 | Подключение элементов управления интерфейса. |
| 26.05.2025-28.05.2025 | Разработка алгоритма для построения ступенчатого графика. |
| 29.05.2025 | Реализация трудоемких вычислительных операций на языке C++ для повышения производительности. |
| 30.05.2025 | Связать вычислительное ядро C++ с языком Python. |
| 02.06.2025-03.06.2025 | Тестирование разработанной программы на большой выборке данных. |
| 04.06.2025 | Создание исполняемого файла программы. |
| 05.06.2025 | Оформление отчета. |
| 06.06.2025 | Защита практики на базе практики. |
| 07.06.2025 | Защита практики на кафедре. |

|  |  |
| --- | --- |
| Руководитель практики от кафедры | \_\_\_\_\_\_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись И.О. Фамилия |
| Руководитель практики от профильной организации | \_\_\_\_\_\_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись И.О. Фамилия |

**7. ОТЧЕТ ОБУЧАЮЩЕГОСЯ О ПРАКТИКЕ**

Я, Халитова Айгуль Азатовна, прошла производственную практику с 12 мая 2025 по 7 июня 2025.

В соответствии с программой практики и индивидуальным заданием я выполняла следующую работу:

1) изучила основные принципы работы с Qt Designer для создания пользовательского интерфейса.

2) разработала алгоритм построения ступенчатого графика и реализовала его перенос на язык программирования C++. Проверила работоспособность разработанной программы на большой выборке данных.

3) создала исполняемый файл (.exe) из разработанной программы.

В результате прохождения практики поставленные задачи были решены в полном объеме, профессиональные компетенции (профессиональные умения, навыки и опыт профессиональной деятельности) приобретены.

|  |  |
| --- | --- |
| Обучающийся | \_\_\_\_\_\_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись И.О. Фамилия |

**8. ЗАКЛЮЧЕНИЕ РУКОВОДИТЕЛЯ ПО ПРАКТИЧЕСКОЙ ПОДГОТОВКЕ О ПРАКТИКЕ**

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ прошел производственную практику с 12 мая 2025 по 7 июня 2025.

Перед обучающимся во время прохождения практики были поставлены следующие профессиональные задачи:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Краткая характеристика проделанной работы и полученных результатов:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Во время прохождения практики обучающийся проявил себя как (достоинства, уровень теоретической подготовки, дисциплина, недостатки, замечания) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Рекомендации (пожелания) по организации практики:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| Руководитель практики от профильной организации | \_\_\_\_\_\_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  М.П. подпись И.О. Фамилия  «\_\_\_»\_\_\_\_\_\_\_\_\_20\_\_\_ |

**9. РЕЗУЛЬТАТ ЗАЩИТЫ ОТЧЕТА**

В результате прохождения практики поставленные задачи были решены в полном объеме, профессиональные компетенции (профессиональные умения, навыки и опыт профессиональной деятельности) приобретены.

Результат прохождения практики обучающимся оценивается на:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| Руководитель практики от кафедры | \_\_\_\_\_\_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись И.О. Фамилия |

**ПРИЛОЖЕНИЕ А  
ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

**А1. Введение**

В работе использовались данные, полученные при геофизических исследованиях скважин, а именно — измерения удельного электрического сопротивления горных пород по глубине скважины. Такие данные представляют собой дискретный сигнал, где первый столбец содержал значения по оси Z — координаты в метрах (глубина скважины), а второй столбец — значения по оси V, соответствующие измерениям удельного электрического сопротивления в Ом·м. Такой формат данных представляет собой дискретную последовательность, являющуюся функцией, отражающей значения сигнала в определённые дискретные моменты времени.

Измерения удельного сопротивления выполняются с помощью каротажных приборов, которые регистрируют изменения электрического сопротивления жидкости и пород в скважине [1]. Эти данные отражают физико-химические свойства горных пород и жидкости, заполняющей скважину, и используются для определения различных характеристик пласта.

Данные имеют дискретный характер, так как измерения проводятся через определённые интервалы по глубине, при этом в данных могут встречаться бланковочные значения — отсутствующие измерения, которые исключаются из обработки.

**А2. Пользовательский интерфейс**

Для разработки программного обеспечения применялись следующие библиотеки и инструменты:

* PySide6 — библиотека для создания графического интерфейса на языке Python, предоставляющая современные средства для разработки приложений с использованием объектно-ориентированного подхода. Она использовалась для реализации визуальной части программы, обеспечивая удобное взаимодействие пользователя с алгоритмами обработки данных [2].
* Qt Designer — инструмент для проектирования графических интерфейсов, входящий в состав фреймворка Qt [3]. С его помощью был разработан макет интерфейса приложения, который затем интегрировался с программным кодом на языке Python. Использование Qt Designer [4] позволило ускорить процесс создания и настройки элементов управления интерфейса.

На рисунке 1 приведён пример пользовательского интерфейса разработанного приложения, отображающего основное окно программы с элементами управления.

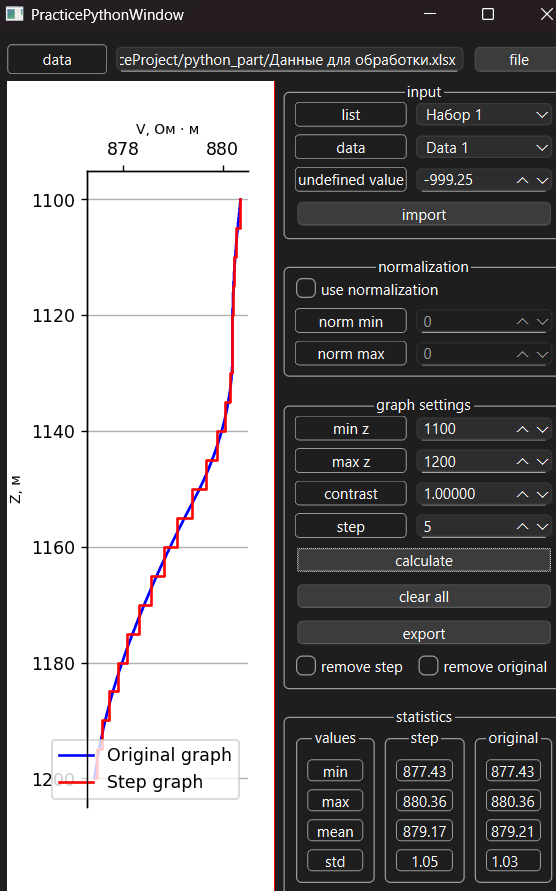


Рисунок 1 – пример работы программы

**А3. Обработка данных.**

Ступенчатый график представляет собой дискретную функцию, отображающую значения сигнала в виде кусочно-постоянных участков

Обработка данных включала фильтрацию по заданному диапазону значений, нормировку и построение ступенчатого графика. Также учитывались бланковочные значения, которые исключались.

Нормировка данных производилась по формуле, которая масштабирует исходные значения *v* из диапазона [] в новый диапазон [], сохраняя пропорции между значениями.

где

*v –* исходное значение

*–* максимальное и минимальное значение исходных данных

*–* новыемаксимальное и минимальное значения, которое вводится пользователем в окне программы.

В алгоритме обработки данных коэффициент контрастности используется для сравнения усреднённых значений удельного сопротивления на соседних интервалах глубины.

Отношение этих значений вычисляется по формуле

где

*–* усреднённых значений удельного сопротивления на соседних интервалах глубины.

*contrast* – коэффициент контрастности, определяющий порог значимых изменений сигнала, вводится пользователем в окне программы.

Если отношение этих значений выходит за пределы,

это означает, что изменения на графике значительные и формируется новая ступень. В противном случае интервал расширяется для сглаживания мелких колебаний, что позволяет выделить только существенные изменения сигнала.

Ступенчатый график формировался путём усреднения значений функции на динамически изменяемых интервалах по оси Z (рисунок 2). Такой подход позволил выявить значимые изменения сигнала с учётом установленного коэффициента контрастности, что соответствует построению дискретной ступенчатой последовательности, отражающей основные особенности исходного сигнала.

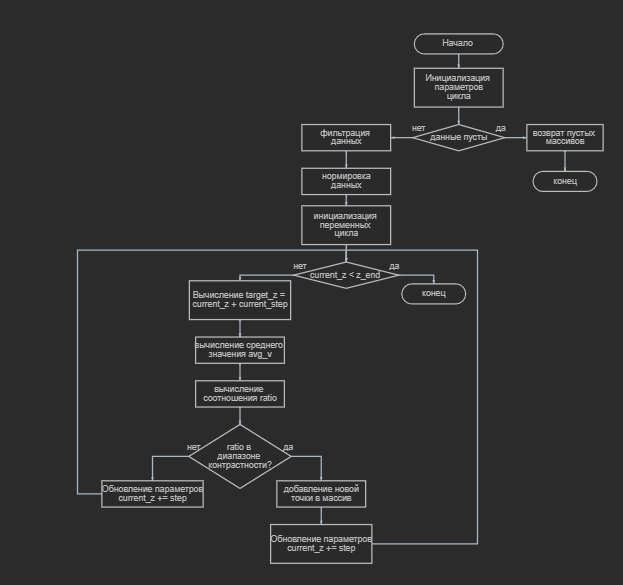


Рисунок 2 – блок схема алгоритма

По обработанным данным находятся параметры:

* минимальное значение
* максимальное значение
* среднее значение

где

*vi –* исходные значение

* стандартное отклонение.

где

*vi –* исходные значение

*–* среднее значение

**ПРИЛОЖЕНИЕ B**

**ПРОГРАММНАЯ РЕАЛИЗАЦИЯ**

**B1. Описание пользовательского интерфейса**

Главное окно программы является основным средством взаимодействия пользователя с функционалом приложения. Оно содержит несколько логически выделенных областей, каждая из которых отвечает за определённый набор задач — ввод данных, настройку параметров обработки, управление выполнением алгоритмов и отображение результатов.

Элементы управления в окне программы реализуют взаимодействие с помощью сигналов и слотов. При изменении возникновении какого-либо события соответствующий сигнал испускается и передаётся в связанный с ним слот — функцию или метод, который обрабатывает это событие [5].

**Область ввода данных** (рисунок 3) предназначена для загрузки и выбора необходимых данных. В этой области пользователь может выбрать файл формата (.xlsx), после чего элементы управления типа combobox динамически заполняются в зависимости от выбранного файла и листа. Также предусмотрена возможность указания бланковочного значения. Для подтверждения выбора и загрузки данных предусмотрена отдельная кнопка import, запускающая процесс импорта и подготовки данных к дальнейшему анализу. Загружаются данные с помощью библиотеки Pandas.

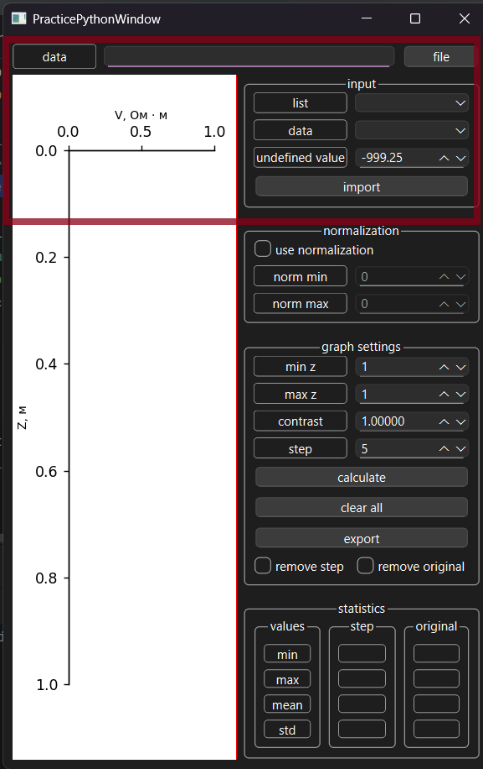


Рисунок 3 – область ввода данных

**Область нормировки данных** (рисунок 4) предназначена для масштабирования исходных данных в заданный пользователем диапазон. В области присутствует элемент управления типа checkbox, который позволяет включать или отключать функцию нормировки. Если флажок не установлен, поля для ввода новых минимального и максимального значений остаются неактивными, и данные обрабатываются без нормировки. При установке галочки поля ввода становятся активными, и данные будут нормированы согласно формуле (1), что позволяет привести значения к выбранному диапазону для улучшения визуализации и дальнейших вычислений.

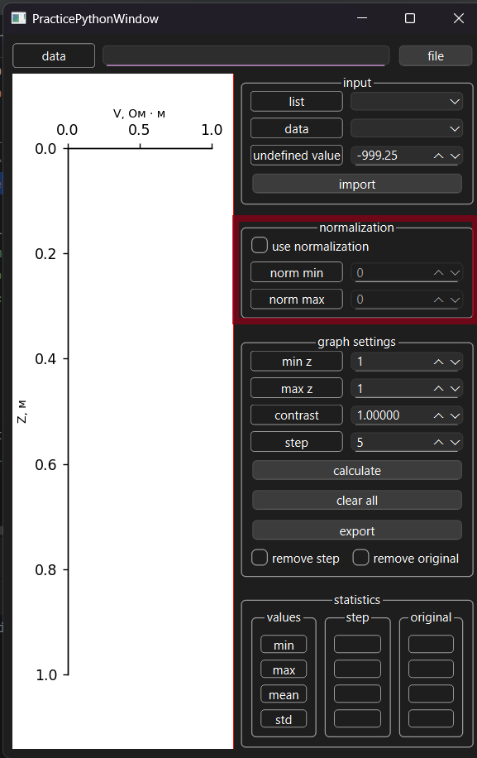


Рисунок 4 – область нормировки данных

**Область выбора параметров обработки** (рисунок 5) содержит элементы управления, позволяющие настраивать параметры алгоритма обработки данных, такие как диапазон значений, коэффициент контрастности, диапазон нормировки и минимальный шаг. Для запуска обработки предусмотрена кнопка calculate. Данные обрабатываются с использованием формул (2), (3). Также в этой области располагаются кнопки для управления визуализацией: кнопка очистки области визуализации, кнопка удаления обоих графиков, а также два элемента типа checkbox, позволяющие по отдельности скрывать или отображать каждый из графиков. При загрузке данных автоматически определяются минимальное и максимальное значения исходных данных, которые устанавливаются в качестве ограничений в соответствующие объекты типа spinbox, для обеспечения корректного ввода параметров. Для экспорта результатов предусмотрена кнопка export выгрузки данных ступенчатого графика в файл формата (.xlsx). Экспорт данных также происходит с помощью библиотеки Pandas.

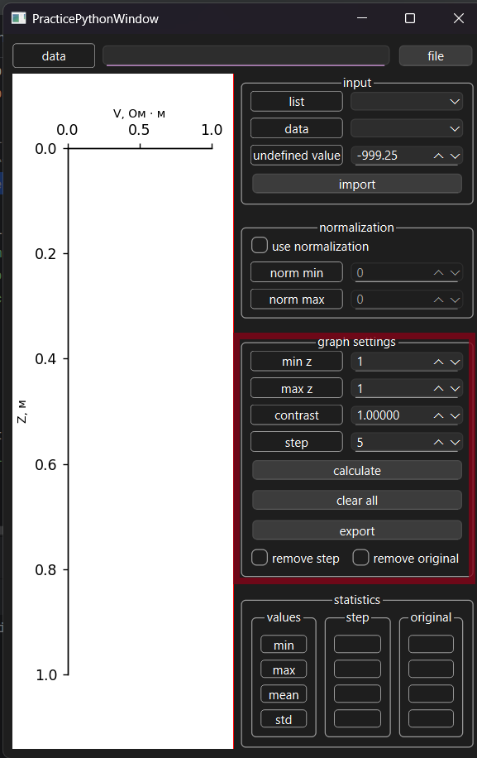


Рисунок 5 - Область выбора параметров обработки

**Область отображения статистических** данных (рисунок 6) предназначена для вывода рассчитанные по формулам (4) и (5) параметров, полученных на основе обработанных данных.

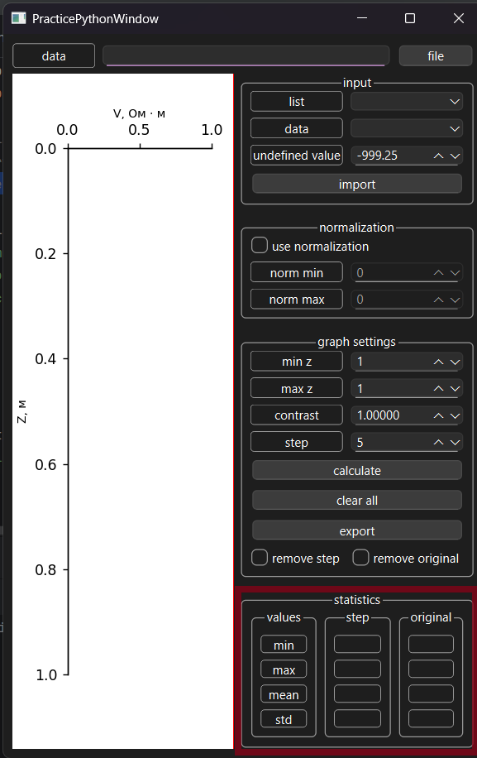


Рисунок 6 - Область отображения статистических данных

**Область визуализации** данных (рисунок 7) предназначена для отображения исходных и преобразованных данных в виде графиков с помощью библиотеки Matplotilb.

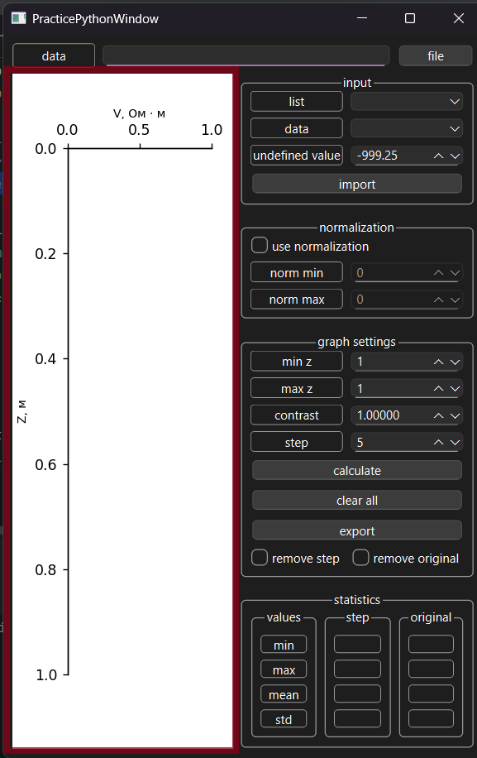


Рисунок 7 - Область визуализации данных

**B2. Использование библиотеки Pybind11**

Для повышения производительности вычислительной части алгоритма критичные по скорости операции были перенесены на язык C++ [7]. Связь между C++ и Python обеспечивалась с помощью библиотеки Pybind11, которая позволяет эффективно интегрировать модули на C++ в Python-приложение, сохраняя удобство разработки и гибкость Python при высокой скорости выполнения вычислений [6].

Для создания модуля в проекте необходимо создать файл setup.py с необходимыми настройками, далее прописать в терминал команду

python python\_part\setup.py build\_ext –inplace

Эта команда выполняет действия:

* build\_ext — команда для сборки C/C++ расширений.
* --inplace — указывает, что скомпилированные расширения должны быть размещены в той же директории, что и исходные файлы.

Такое сочетание технологий позволило объединить преимущества удобства и быстроты разработки на Python с высокой производительностью C++ в вычислительной части, а также создать функциональный и интуитивно понятный графический интерфейс с помощью Qt Designer и PySide6. Это обеспечило эффективную обработку и визуализацию дискретных данных, поступающих из измерений, и удобство взаимодействия пользователя с программой.

**B3. Создание исполняемого файла с помощью библиотеки PyInastller**

Для удобного распространения и запуска разработанного приложения на компьютерах пользователей без необходимости установки интерпретатора Python и всех зависимостей была использована утилита PyInstaller.

PyInstaller — это библиотека, которая позволяет упаковать Python-скрипт и все необходимые библиотеки в один автономный исполняемый файл.

Для удобного распространения и запуска разработанного приложения на компьютерах пользователей без необходимости установки интерпретатора Python и всех зависимостей была использована утилита PyInstaller.

Для создания исполняемого файла в терминале необходимо выполнить команду:

pyinstaller --onefile --windowed --add-data "example.cp313-win\_amd64.pyd;." --icon=favicon.ico --name GraphPlotter python\_part/main.py

Эта команда выполняет действия:

* Параметр --onefile упаковывает всё в один исполняемый файл.
* Параметр -- windowed отключает появление консольного окна при запуске графического приложения.
* Параметр --add-data "example.cp313-win\_amd64.pyd;." добавляет необходимый для корректной работы библиотеки Pybind11бинарный модуль в пакет.
* Параметр --icon=favicon.ico задаёт иконку для исполняемого файла.
* Параметр --name GraphPlotter устанавливает имя итогового файла.
* python\_part/main.py — путь к основному скрипту приложения.

После успешного выполнения команды исполняемый файл будет находиться в папке dist в корне проекта.

**B4. Вывод**

В ходе выполнения работы была разработана программа для обработки и визуализации данных удельного электрического сопротивления горных пород, полученных при исследованиях скважин.

* Разработан алгоритм построения ступенчатого графика на основе дискретных данных, учитывающий фильтрацию, нормировку и выделение значимых изменений сигнала с помощью коэффициента контрастности.
* Реализован форматированный ввод и вывод данных на языке Python, что обеспечило удобство загрузки, обработки и экспорта данных.
* Разработано вычислительное ядро на C++ для повышения производительности обработки, интегрированное с языком Python, что позволило сохранить гибкость разработки при высокой скорости вычислений.
* Создан пользовательский интерфейс на основе PySide6 и Qt Designer, обеспечивающий понятное взаимодействие пользователя с программой, настройку параметров и визуализацию результатов.
* Создана сборка автономного исполняемого файла (.exe), для обеспечения простоты распространения и запуска приложения на компьютерах других пользователей без необходимости установки дополнительных компонентов.

Таким образом, разработанная программа эффективно сочетает высокую производительность вычислительной части с удобством пользовательского интерфейса и обеспечивает обработку и визуализацию исходных и обработанных данных. Программа может быть применена для анализа и интерпретации данных геофизических исследований скважин, а также при проектировании и контроле качества разработки месторождений.

**ПРИЛОЖЕНИЕ C**

**СПИСОК ЛИТЕРАТУРЫ**

1. Геофизические исследования скважин: учебник для вузов / Ю. И. Горбачев. — Москва: Недра, 1990. — 398 с. — ISBN 5-247-01972-5.
2. Python 3 и PyQt 5. Разработка приложений / Н. Прохоренок, В. Дронов. — Санкт-Петербург: БХВ-Петербург, 2016. — 832 с. — ISBN 978-5-9775-3648-6.
3. Об интерфейсе. Основы проектирования взаимодействия / Купер А., Рейман Р., Кронин Д. — Санкт-Петербург: Символ-Плюс, 2009. — 688 с.
4. Qt 5.10. Профессиональное программирование на С++ / М. Шлее. — Санкт-Петербург: БХВ-Петербург, 2018. — 1072 с. — ISBN 978-5-9775-3678-3.
5. Однострочники Python. Лаконичный и содержательный код / К. Майер. — Санкт-Петербург: Библиотека программиста 2022. — 256 с. — ISBN 978-5-4461-2966-9.
6. Программирование. Принципы и практика использования C++ / Б. Страуструп. — 2-е изд.: Вильямс, 2018. — 1328 с. — ISBN 978-5-6040724-8-6.
7. Программирование на Python в примерах и задачах / Алексей Васильев. — Москва: Эксмо, 2021. — 616 с. ISBN 978-5-04-103199-2

**ПРИЛОЖЕНИЕ D**

**ЛИСТИНГ ПРОГРАММЫ**

main.py

import sys  
  
from PySide6.QtWidgets import QApplication  
  
from main\_window import MainWindow  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = QApplication(sys.argv)  
 window = MainWindow()  
 window.show()  
 sys.exit(app.exec())

perform\_calculation.py

import example  
import numpy as np  
import pandas as pd  
  
class Model:  
 def \_\_init\_\_(self) -> None:  
 self.file\_path = None  
 self.current\_excel = None # объект ExcelFile  
 self.sheet = None  
 self.column = None  
 self.data\_frame = None  
 self.min\_z = None  
 self.max\_z = None  
 self.undefined\_value = None  
 self.new\_min = None  
 self.new\_max = None  
 self.statistics\_min = None  
 self.statistics\_max = None  
 self.statistics\_mean = None  
 self.statistics\_std = None  
 self.result = None  
  
 def load\_excel(self, file\_path: str) -> list[str]: # загрузка данных  
 self.file\_path = file\_path  
 self.current\_excel = pd.ExcelFile(file\_path)  
 return self.current\_excel.sheet\_names  
  
 def select\_sheet(self, sheet: str) -> list[str]: # выбор страницы в файле экселя  
 self.sheet = sheet  
 df = pd.read\_excel(self.current\_excel, sheet\_name=sheet, nrows=0)  
 return df.columns.tolist()[2:]  
  
 def select\_column(self, column: str) -> None: # выбор колонки  
 self.column = column  
  
 def import\_data(self) -> None: # импорт данных  
 if not self.file\_path or not self.sheet or not self.column:  
 print('File, sheet or column not chosen')  
 return  
 df = pd.read\_excel(self.current\_excel, sheet\_name=self.sheet, usecols=[self.column, 'MD'])  
 self.data\_frame = df  
  
 self.min\_z = self.data\_frame.iloc[:, 0].min()  
 self.max\_z = self.data\_frame.iloc[:, 0].max()  
 print(self.min\_z, self.max\_z)  
  
 def \_norm\_data(self, data: np.array, mask: np.array) -> np.array: # нормировка данных  
 print('normalizing data')  
  
 if self.new\_min == self.new\_max:  
 return data.copy()  
  
 v\_norm\_data = data.copy()  
 if np.any(mask):  
 v\_min = data[mask].min()  
 v\_max = data[mask].max()  
  
 v\_norm\_data[mask] = (data[mask] - v\_min) / (v\_max - v\_min) \* (self.new\_max - self.new\_min) + self.new\_min  
 return v\_norm\_data  
  
 def perform\_calculation(self, min\_z: float, max\_z: float, contrast: float, step: float, undef\_val: float,  
 new\_min: int, new\_max: int) \  
 -> tuple[  
 np.ndarray, np.ndarray, np.ndarray, np.ndarray]: # вычисления без контрастности с нормировкой, питон  
 print('perform calculation python')  
 self.undefined\_value = undef\_val  
 self.new\_min = new\_min  
 self.new\_max = new\_max  
 if self.data\_frame is not None and not self.data\_frame.empty:  
  
 z\_data = self.data\_frame.iloc[:, 0].to\_numpy()  
 v\_data = self.data\_frame.iloc[:, 1].to\_numpy()  
  
 valid\_data = (v\_data != self.undefined\_value)  
  
 v\_data = self.\_norm\_data(v\_data, valid\_data) # нормировка данных  
  
 z\_data\_filtered = z\_data[valid\_data]  
 v\_data\_filtered = v\_data[valid\_data]  
  
 range\_data = (z\_data\_filtered >= min\_z) & (z\_data\_filtered <= max\_z)  
 v\_data\_filtered = v\_data\_filtered[range\_data]  
 z\_data\_filtered = z\_data\_filtered[range\_data]  
  
 z\_steps = np.arange(z\_data\_filtered[0], max\_z + step, step)  
  
 x\_steps = np.interp(z\_steps, z\_data\_filtered, v\_data\_filtered)  
  
 self.result = (x\_steps, z\_steps)  
  
 print(z\_data\_filtered, v\_data\_filtered)  
  
 return z\_data\_filtered, v\_data\_filtered, z\_steps, x\_steps  
 else:  
 print('No data to calculate')  
 return np.array([]), np.array([]), np.array([]), np.array([])  
  
 def perform\_calculation1(self, min\_z: float, max\_z: float, contrast: float, step: float, undef\_val: float,  
 new\_min: int, new\_max: int) \  
 -> tuple[  
 np.ndarray, np.ndarray, np.ndarray, np.ndarray]: # вычисления с контрастностью и с нормировкой, питон  
 print('perform calculation python')  
 self.undefined\_value = undef\_val  
 self.new\_min = new\_min  
 self.new\_max = new\_max  
 if self.data\_frame is not None and not self.data\_frame.empty:  
 z\_data = self.data\_frame.iloc[:, 0].to\_numpy()  
 v\_data = self.data\_frame.iloc[:, 1].to\_numpy()  
  
 valid\_mask = (v\_data != undef\_val) & (z\_data >= min\_z) & (z\_data <= max\_z)  
  
 v\_data = self.\_norm\_data(v\_data, valid\_mask) # нормировка данных  
  
 z\_filtered = z\_data[valid\_mask]  
 v\_filtered = v\_data[valid\_mask]  
  
 if len(z\_filtered) == 0:  
 return np.array([]), np.array([]), np.array([]), np.array([])  
  
 z\_start = z\_filtered[0]  
 z\_end = z\_filtered[-1]  
  
 z\_steps = []  
 v\_steps = []  
  
 current\_index = 0  
 current\_z = z\_start  
 last\_v = v\_filtered[0]  
 initial\_step = step  
 current\_step = initial\_step  
  
 while current\_z < z\_end and current\_index < len(z\_filtered):  
 target\_z = min(current\_z + current\_step, z\_end)  
  
 sum\_v = 0.0  
 count = 0  
 temp\_index = current\_index  
  
 while temp\_index < len(z\_filtered) and z\_filtered[temp\_index] < target\_z:  
 sum\_v += v\_filtered[temp\_index]  
 count += 1  
 temp\_index += 1  
  
 if count == 0 or current\_index == 0:  
 z\_steps.extend([current\_z, target\_z])  
 v\_steps.extend([last\_v, last\_v])  
 current\_z = target\_z  
 current\_index = temp\_index  
 current\_step = initial\_step  
 continue  
  
 avg\_v = sum\_v / count  
 ratio = avg\_v / last\_v if last\_v != 0 else float('inf')  
 print(ratio, current\_step)  
  
 if target\_z >= z\_end:  
 z\_steps.extend([current\_z, target\_z])  
 v\_steps.extend([avg\_v, avg\_v])  
 break  
  
 if ratio <= 1 / contrast or ratio >= contrast:  
 z\_steps.extend([current\_z, target\_z])  
 v\_steps.extend([avg\_v, avg\_v])  
 current\_z = target\_z  
 last\_v = avg\_v  
 current\_step = initial\_step  
 current\_index = temp\_index  
 else:  
 current\_step += 1.0  
  
 z\_steps.append(z\_filtered[-1])  
 v\_steps.append(v\_filtered[-1])  
  
 self.result = (z\_steps, v\_steps)  
  
 return np.array(z\_filtered), np.array(v\_filtered), np.array(z\_steps), np.array(v\_steps)  
 else:  
 print('No data to calculate')  
 return np.array([]), np.array([]), np.array([]), np.array([])  
  
 def perform\_calculation2(self, min\_z: float, max\_z: float, contrast: float, step: float, undef\_val: float,  
 new\_min: int, new\_max: int) \  
 -> tuple[np.ndarray, np.ndarray, np.ndarray, np.ndarray]: # вычисления с контрастностью и с нормировкой c++  
 print('perform calculation pybind2')  
  
 self.undefined\_value = undef\_val  
 self.new\_min = new\_min  
 self.new\_max = new\_max  
  
 if self.data\_frame is not None and not self.data\_frame.empty:  
 input\_data = self.data\_frame.iloc[:, :2].to\_numpy(dtype=np.float64)  
  
 z\_array = input\_data[:, 0]  
 v\_array = input\_data[:, 1]  
  
 z\_filtered, v\_filtered, z\_steps, v\_steps = example.perform\_calculation(  
 z\_array, v\_array, min\_z, max\_z, step, contrast, self.undefined\_value, self.new\_min, self.new\_max  
 )  
  
 self.result = (z\_steps, v\_steps)  
  
 print(z\_filtered)  
 print(v\_filtered)  
 print(z\_steps)  
 print(v\_steps)  
 return z\_filtered, v\_filtered, z\_steps, v\_steps  
 else:  
 print('No data to calculate')  
 return np.array([]), np.array([]), np.array([]), np.array([])  
  
 def compute\_statistics2(self, z\_steps: np.array) -> tuple[float, float, float, float]: # вычисления статистики c++  
 if self.result is None:  
 print('No results to calculate statistics2')  
 return 0.0, 0.0, 0.0, 0.0  
  
 # z\_steps = self.result[1]  
  
 self.statistics\_min, self.statistics\_max, self.statistics\_mean, self.statistics\_std = (  
 example.calculate\_statistics2(z\_steps))  
 return self.statistics\_min, self.statistics\_max, self.statistics\_mean, self.statistics\_std  
  
 def compute\_statistics(self, z\_steps: np.array) -> tuple[float, float, float, float]: # вычисления статистики питон  
 if self.result is None:  
 print('No results to calculate statistics')  
 return 0.0, 0.0, 0.0, 0.0  
  
 # z\_steps = self.result[0]  
  
 self.statistics\_min = np.min(z\_steps)  
 self.statistics\_max = np.max(z\_steps)  
 self.statistics\_mean = np.mean(z\_steps)  
 self.statistics\_std = np.std(z\_steps)  
  
 return self.statistics\_min, self.statistics\_max, self.statistics\_mean, self.statistics\_std  
  
 def save\_to\_file\_xlsx(self, filename: str) -> None: # выгрузка данных  
 if self.result is None:  
 print('No results to save')  
 return  
 x\_steps, z\_steps = self.result  
 df\_export = pd.DataFrame({  
 'X\_steps': x\_steps,  
 'Z\_steps': z\_steps  
 })  
 df\_export.to\_excel(filename, index=False)

widget\_drawer.py

from matplotlib.backends.backend\_qt5agg import FigureCanvasQTAgg  
from matplotlib.figure import Figure  
  
class MplCanvas(FigureCanvasQTAgg):  
 def \_\_init\_\_(self, parent: object = None, width: int = 10, height: int = 5, dpi: int = 100) -> None:  
 self.fig = Figure(figsize=(width, height), dpi=dpi)  
 self.ax = self.fig.add\_subplot(111)  
 super().\_\_init\_\_(self.fig)  
 self.setParent(parent)  
  
 self.ax.invert\_yaxis()  
  
 # скрываем нижнюю ось x и правую границу  
 self.ax.spines['bottom'].set\_visible(False)  
 self.ax.spines['right'].set\_visible(False)  
  
 # левые и верхние границы видимы  
 self.ax.spines['left'].set\_visible(True)  
 self.ax.spines['top'].set\_visible(True)  
  
 self.ax.yaxis.set\_ticks\_position('left') # Ось y слева  
  
 # дополнительная ось сверху  
 self.ax\_top = self.ax.twiny()  
  
 # Настраиваем верхнюю ось  
 self.ax\_top.spines['top'].set\_visible(True)  
 self.ax\_top.spines['bottom'].set\_visible(False)  
  
 # скрываем правую границу верхней оси дополнительной оси  
 self.ax\_top.spines['right'].set\_visible(False)  
 self.ax\_top.yaxis.set\_ticks\_position('left')  
  
 self.ax\_top.xaxis.set\_ticks\_position('top')  
 self.ax\_top.xaxis.set\_label\_position('top')  
 self.ax\_top.set\_xlabel('V, Ом · м', fontsize=8)  
  
 # Скрываем нижнюю ось x основной оси  
 self.ax.xaxis.set\_visible(False)  
  
 self.ax.set\_ylabel('Z, м', fontsize=8)  
  
 self.fig.subplots\_adjust(left=0.25, top=0.89)

main\_window.py

import numpy as np  
from PySide6.QtWidgets import QFileDialog, QMainWindow, QVBoxLayout  
  
from model import Model  
from ui\_window import Ui\_MainWindow  
from widget\_drawer import MplCanvas  
  
class MainWindow(QMainWindow):  
 def \_\_init\_\_(self) -> None:  
 super().\_\_init\_\_()  
  
 self.original\_line = None  
 self.step\_line = None  
  
 self.model = Model()  
 self.ui = Ui\_MainWindow()  
 self.ui.setupUi(self)  
  
 self.ui.PB\_Choose\_file.clicked.connect(self.choose\_file\_name)  
 self.ui.PB\_Import\_Data.clicked.connect(self.import\_data)  
 self.ui.PB\_Calculate.clicked.connect(self.calculate)  
 self.ui.PB\_Export.clicked.connect(self.export\_data)  
 self.ui.PB\_Clear\_All.clicked.connect(self.clear\_all\_graphs)  
  
 self.ui.SB\_Contrast.valueChanged.connect(self.contrast\_changed)  
 self.ui.SB\_Step.valueChanged.connect(self.step\_val\_changed)  
 self.ui.SB\_Max\_Z.valueChanged.connect(self.max\_z\_changed)  
 self.ui.SB\_Min\_Z.valueChanged.connect(self.min\_z\_changed)  
 self.ui.SB\_uv.valueChanged.connect(self.undef\_val\_changed)  
 self.ui.SB\_Max\_Norm.valueChanged.connect(self.norm\_max\_val\_changed)  
 self.ui.SB\_Min\_Norm.valueChanged.connect(self.norm\_min\_val\_changed)  
  
 self.ui.CB\_list\_choose.currentIndexChanged.connect(self.combobox\_list\_choose\_change)  
 self.ui.CB\_data\_choose.currentIndexChanged.connect(self.combobox\_data\_choose\_change)  
  
 self.ui.CheckBox\_Remove\_Original.toggled.connect(self.remove\_original\_graph)  
 self.ui.CheckBox\_Remove\_Step.toggled.connect(self.remove\_step\_graph)  
 self.ui.cb\_use\_normalization.toggled.connect(self.use\_normalization)  
  
 self.ui.SB\_Min\_Norm.setDisabled(True)  
 self.ui.SB\_Max\_Norm.setDisabled(True)  
  
 self.ui.lineEdit\_data.textChanged.connect(self.text\_changed)  
 self.ui.lineEdit\_data.textEdited.connect(self.text\_edited)  
 self.ui.lineEdit\_data.returnPressed.connect(self.return\_pressed)  
  
 self.canvas = MplCanvas(self.ui.widget)  
 layout = QVBoxLayout(self.ui.widget)  
 layout.setContentsMargins(0, 0, 0, 0)  
 layout.addWidget(self.canvas)  
  
 def choose\_file\_name(self) -> None: # выбор файла для считывания данных и динамическое заполнение именами листов  
 file, \_ = QFileDialog.getOpenFileName(self, '', '', 'Excel Files (\*.xlsx)')  
 if not file:  
 return  
 self.ui.lineEdit\_data.setText(file)  
 sheet\_names = self.model.load\_excel(file)  
 self.ui.CB\_list\_choose.blockSignals(True)  
 self.ui.CB\_list\_choose.clear()  
 self.ui.CB\_list\_choose.addItems(sheet\_names)  
 self.ui.CB\_list\_choose.setCurrentIndex(0)  
 self.ui.CB\_list\_choose.blockSignals(False)  
 self.combobox\_list\_choose\_change()  
  
 def combobox\_list\_choose\_change(self) -> None: # динамическое заполнение именами колонок и выбор листа  
 sheet = self.ui.CB\_list\_choose.currentText()  
 print(f'Chosen sheet {sheet}')  
  
 columns = self.model.select\_sheet(sheet)  
  
 self.ui.CB\_data\_choose.blockSignals(True)  
 self.ui.CB\_data\_choose.clear()  
 self.ui.CB\_data\_choose.addItems(columns)  
 self.ui.CB\_data\_choose.setCurrentIndex(0)  
 self.ui.CB\_data\_choose.blockSignals(False)  
 self.combobox\_data\_choose\_change()  
  
 def combobox\_data\_choose\_change(self) -> None: # выбор колонки  
 column = self.ui.CB\_data\_choose.currentText()  
 print(f'Chosen column {column}')  
 self.model.select\_column(column)  
  
 def import\_data(self) -> None: # загрузка данных  
 print('Importing data')  
 self.model.import\_data()  
 if self.model.data\_frame is not None:  
 print(self.model.data\_frame.head())  
  
 self.set\_min\_z()  
 self.set\_max\_z()  
  
 def calculate(self) -> None: # вычисления  
 self.ui.CheckBox\_Remove\_Original.setChecked(False)  
 self.ui.CheckBox\_Remove\_Step.setChecked(False)  
 v\_data, z\_data, x\_steps, z\_steps = self.model.perform\_calculation2(  
 self.ui.SB\_Min\_Z.value(),  
 self.ui.SB\_Max\_Z.value(),  
 self.ui.SB\_Contrast.value(),  
 self.ui.SB\_Step.value(),  
 self.ui.SB\_uv.value(),  
 self.ui.SB\_Min\_Norm.value(),  
 self.ui.SB\_Max\_Norm.value(),  
 )  
  
 self.plot\_results(v\_data, z\_data, x\_steps, z\_steps)  
  
 v\_step\_min, v\_step\_max, v\_step\_mean, v\_step\_std = self.model.compute\_statistics2(z\_steps) # подсчет статистики  
 self.perform\_step\_statistics(v\_step\_min, v\_step\_max, v\_step\_mean, v\_step\_std)  
  
 v\_orig\_min, v\_orig\_max, v\_orig\_mean, v\_orig\_std = self.model.compute\_statistics2(z\_data) # подсчет статистики  
 self.perform\_orig\_statistics(v\_orig\_min, v\_orig\_max, v\_orig\_mean, v\_orig\_std)  
  
 def perform\_step\_statistics(self, v\_min: float, v\_max: float, v\_mean: float, v\_std: float) -> None: # результаты ступенчатого графика  
 self.ui.lb\_step\_min\_val.setText(str(format(v\_min, '.2f')))  
 self.ui.lb\_step\_max\_val.setText(str(format(v\_max, '.2f')))  
 self.ui.lb\_step\_mean\_val.setText(str(format(v\_mean, '.2f')))  
 self.ui.lb\_step\_std\_val.setText(str(format(v\_std, '.2f')))  
  
 def perform\_orig\_statistics(self, v\_min: float, v\_max: float, v\_mean: float, v\_std: float) -> None: # результаты ступенчатого графика  
 self.ui.lb\_orig\_min\_val.setText(str(format(v\_min, '.2f')))  
 self.ui.lb\_orig\_max\_val.setText(str(format(v\_max, '.2f')))  
 self.ui.lb\_orig\_mean\_val.setText(str(format(v\_mean, '.2f')))  
 self.ui.lb\_orig\_std\_val.setText(str(format(v\_std, '.2f')))  
  
 def export\_data(self) -> None: # выгрузка данных  
 self.model.save\_to\_file\_xlsx('Обработанные данные.xlsx')  
 print('Data export')  
  
 def plot\_results(self, v\_data: np.ndarray, z\_data: np.ndarray, x\_steps: np.ndarray, z\_steps: np.ndarray) -> None: # построение графика  
 ax = self.canvas.ax  
 ax\_top = self.canvas.ax\_top  
  
 # Очищаем содержимое обеих осей  
 ax.cla()  
 ax\_top.cla()  
  
 # Инвертируем ось y заново после очистки  
 ax.invert\_yaxis()  
  
 # Скрываем нижнюю ось x и правую границу основной оси  
 ax.spines['bottom'].set\_visible(False)  
 ax.spines['right'].set\_visible(False)  
 ax.spines['left'].set\_visible(True)  
 ax.spines['top'].set\_visible(True)  
  
 # Восстанавливаем подписи и положение осей  
 ax.set\_ylabel('Z, м', fontsize=8)  
 ax.xaxis.set\_visible(False)  
 ax.yaxis.set\_ticks\_position('left')  
  
 # Строим графики на основной оси  
 self.original\_line = ax.plot(z\_data, v\_data, 'b-', label='Original graph')[0]  
 self.step\_line = ax.plot(z\_steps, x\_steps, 'r-', label='Step graph')[0]  
  
 # Настраиваем верхнюю ось  
 ax\_top.spines['top'].set\_visible(True)  
 ax\_top.spines['bottom'].set\_visible(False)  
 ax\_top.xaxis.set\_ticks\_position('top')  
 ax\_top.xaxis.set\_label\_position('top')  
 ax\_top.set\_xlabel('V, Ом · м', fontsize=8)  
  
 # Устанавливаем одинаковые пределы по X для обеих осей  
 ax\_top.set\_xlim(ax.get\_xlim())  
  
 # Легенда и сетка на основной оси  
 ax.legend(loc='lower right')  
 ax.grid(True)  
  
 # Отступы  
 self.canvas.fig.subplots\_adjust(left=0.3, top=0.89)  
  
 # Обновляем холст  
 self.canvas.fig.canvas.draw\_idle()  
  
 def clear\_all\_graphs(self) -> None: # удаляет все графики  
 ax = self.canvas.ax  
 ax.clear()  
 ax.figure.canvas.draw()  
 self.original\_line = None  
 self.step\_line = None  
  
 def use\_normalization(self, checked: bool) -> None:  
 if checked:  
 self.ui.SB\_Min\_Norm.setEnabled(True)  
 self.ui.SB\_Max\_Norm.setEnabled(True)  
 else:  
 self.ui.SB\_Min\_Norm.setValue(0)  
 self.ui.SB\_Max\_Norm.setValue(0)  
  
 def remove\_original\_graph(self, checked: bool) -> None: # скрывает исходный график  
 if self.original\_line is not None:  
 self.original\_line.set\_visible(not checked)  
 self.canvas.draw()  
  
 def remove\_step\_graph(self, checked: bool) -> None: # скрывает ступенчатый график  
 if self.step\_line is not None:  
 self.step\_line.set\_visible(not checked)  
 self.canvas.draw()  
  
 def set\_min\_z(self):  
 self.ui.SB\_Min\_Z.setMinimum(int(self.model.min\_z))  
 self.ui.SB\_Min\_Z.setValue(int(self.model.min\_z))  
 print(f'Min z {self.model.min\_z}')  
  
 def set\_max\_z(self):  
 self.ui.SB\_Max\_Z.setMaximum(int(self.model.max\_z))  
 self.ui.SB\_Max\_Z.setValue(int(self.model.max\_z))  
 print(f'Max z {self.model.max\_z}')  
  
 def norm\_max\_val\_changed(self, value: int) -> None:  
 print(f'Value of max norm {value}')  
  
 def norm\_min\_val\_changed(self, value: int) -> None:  
 print(f'Value of min norm {value}')  
  
 def contrast\_changed(self, value: int) -> None:  
 print(f'Value of contrast {value}')  
  
 def step\_val\_changed(self, value: int) -> None:  
 print(f'Value of step {value}')  
  
 def max\_z\_changed(self, value: int) -> None:  
 print(f'Max z = {value}')  
  
 def min\_z\_changed(self, value: int) -> None:  
 print(f'Min z = {value}')  
  
 def text\_changed(self, text: str) -> None:  
 print(f'Text changed {text}')  
  
 def text\_edited(self, text: str) -> None:  
 print(f'Text edited {text}')  
  
 def return\_pressed(self) -> None:  
 print('Enter was pressed')  
  
 def undef\_val\_changed(self, value: int) -> None:  
 print(f'Undefined value changed {value}')

setup.py

from pybind11.setup\_helpers import Pybind11Extension, build\_ext

from setuptools import setup

# pyside6-uic ui\_window.ui -o ui\_window.py

# python python\_part\setup.py build\_ext --inplace

# pyinstaller --onefile --windowed --add-data "example.cp313-win\_amd64.pyd;." --icon=favicon.ico --name GraphPlotter python\_part/main.py

ext\_modules = [

Pybind11Extension(

'example', [r'C:\Python\_projects\PracticeProject\cpp\_part\pybind\source.cpp'], # Путь к исходному файлу

language='c++',

),

]

setup(

name='example',

ext\_modules=ext\_modules,

cmdclass={'build\_ext': build\_ext},

)

pybind.cpp

#include <vector>

#include <pybind11/pybind11.h>

#include <pybind11/numpy.h>

#include <algorithm>

namespace py = pybind11;

std::vector<double> norm\_data(std::vector<double>& data, int min\_v, int max\_v) {

if (min\_v == max\_v) {

return data;

}

else {

double min\_v\_val = \*std::min\_element(begin(data), end(data));

double max\_v\_val = \*std::max\_element(begin(data), end(data));

for (size\_t i = 0; i < data.size(); i++) {

data[i] = (data[i] - min\_v\_val) / (max\_v\_val - min\_v\_val) \* (max\_v - min\_v) + min\_v;

}

return data;

}

}

py::tuple perform\_calculation(

py::array\_t<double> z\_array,

py::array\_t<double> v\_array,

double min\_z,

double max\_z,

double step,

double contrast,

double undef\_val,

int new\_min,

int new\_max)

{

auto v\_buf = v\_array.request();

auto z\_buf = z\_array.request();

double\* v\_ptr = static\_cast<double\*>(v\_buf.ptr);

double\* z\_ptr = static\_cast<double\*>(z\_buf.ptr);

size\_t n = v\_buf.size;

std::vector<double> v\_filtered, z\_filtered;

for (size\_t i = 0; i < n; i++) {

double z = z\_ptr[i];

double v = v\_ptr[i];

if (v != undef\_val && z >= min\_z && z <= max\_z) {

v\_filtered.push\_back(v);

z\_filtered.push\_back(z);

}

}

if (z\_filtered.empty() || v\_filtered.empty()) {

py::array\_t<double> empty\_array(0);

return py::make\_tuple(empty\_array, empty\_array, empty\_array, empty\_array);

}

v\_filtered = norm\_data(v\_filtered, new\_min, new\_max);

const double z\_start = z\_filtered.front();

const double z\_end = z\_filtered.back();

std::vector<double> z\_steps;

std::vector<double> v\_steps;

size\_t current\_index = 0;

double current\_z = z\_start;

double last\_v = v\_filtered.front();

double initial\_step = step;

double current\_step = initial\_step;

while (current\_z < z\_end && current\_index < z\_filtered.size()) {

double target\_z = std::min(current\_z + current\_step, z\_end);

double sum\_v = 0.0;

size\_t count = 0;

size\_t temp\_index = current\_index;

while (temp\_index < z\_filtered.size() && z\_filtered[temp\_index] < target\_z) {

sum\_v += v\_filtered[temp\_index];

count++;

temp\_index++;

}

if (count == 0 || current\_index == 0) {

z\_steps.push\_back(current\_z);

z\_steps.push\_back(target\_z);

v\_steps.push\_back(last\_v);

v\_steps.push\_back(last\_v);

current\_z = target\_z;

current\_index = temp\_index;

current\_step = initial\_step;

continue;

}

double avg\_v = sum\_v / count;

double contrast\_val;

if (last\_v != 0) {

contrast\_val = avg\_v / last\_v;

}

else {

contrast\_val = std::numeric\_limits<double>::infinity();

}

if (target\_z >= z\_end) {

z\_steps.push\_back(current\_z);

z\_steps.push\_back(target\_z);

v\_steps.push\_back(avg\_v);

v\_steps.push\_back(avg\_v);

break;

}

if (contrast\_val <= (1.0 / contrast) || contrast\_val >= contrast) {

z\_steps.push\_back(current\_z);

z\_steps.push\_back(target\_z);

v\_steps.push\_back(avg\_v);

v\_steps.push\_back(avg\_v);

current\_z = target\_z;

last\_v = avg\_v;

current\_step = initial\_step;

current\_index = temp\_index;

}

else {

current\_step += 1.0;

}

}

z\_steps.push\_back(z\_end);

v\_steps.push\_back(v\_filtered.back());

py::array\_t<double> v\_result(v\_filtered.size());

py::array\_t<double> z\_result(z\_filtered.size());

py::array\_t<double> v\_steps\_result(v\_steps.size());

py::array\_t<double> z\_steps\_result(z\_steps.size());

std::copy(v\_filtered.begin(), v\_filtered.end(), v\_result.mutable\_data());

std::copy(z\_filtered.begin(), z\_filtered.end(), z\_result.mutable\_data());

std::copy(v\_steps.begin(), v\_steps.end(), v\_steps\_result.mutable\_data());

std::copy(z\_steps.begin(), z\_steps.end(), z\_steps\_result.mutable\_data());

return py::make\_tuple(

z\_result,

v\_result,

z\_steps\_result,

v\_steps\_result

);

}

py::tuple calculate\_statistics(py::array\_t<double> arr) {

auto buf = arr.request();

double\* ptr = static\_cast<double\*>(buf.ptr);

size\_t n = buf.size;

if (n == 0) {

return py::make\_tuple(0.0, 0.0, 0.0, 0.0);

}

double min\_val = std::numeric\_limits<double>::max();

double max\_val = std::numeric\_limits<double>::lowest();

double sum = 0.0;

for (size\_t i = 0; i < n; i++) {

double val = ptr[i];

if (val < min\_val) {

min\_val = val;

}

if (val > max\_val) {

max\_val = val;

}

sum += val;

}

double mean = sum / n;

double diff\_sum = 0.0;

for (size\_t i = 0; i < n; i++) {

double diff = ptr[i] - mean;

diff\_sum += diff \* diff;

}

double stddev = std::sqrt(diff\_sum / n);

return py::make\_tuple(min\_val, max\_val, mean, stddev);

}

PYBIND11\_MODULE(example, m) {

m.def("perform\_calculation", &perform\_calculation,

py::arg("z\_array"),

py::arg("v\_array"),

py::arg("min\_z"),

py::arg("max\_z"),

py::arg("step"),

py::arg("contrast"),

py::arg("undef\_val"),

py::arg("new\_min"),

py::arg("new\_max"));

m.def("calculate\_statistics2", &calculate\_statistics);

}

1. При проведении практики в профильной организации руководителем практики от кафедры и руководителем практики от профильной организации составляется совместный рабочий график (план) проведения практики. [↑](#footnote-ref-1)