МИНИСТЕРСТВО ОБРАЗОВАНИЯ КИРОВСКОЙ ОБЛАСТИ

Кировское областное государственное профессиональное образовательное

бюджетное учреждение

«Слободской колледж педагогики и социальных отношений»

**ОТЧЕТ**

**по производственной практике**

**ПМ01. Разработка модулей программного обеспечения для компьютерных систем**

**Тема: «Разработка программного модуля «Численные методы»**

Студент

Семакина Лада Владиславовна

Группа 21П-1

Специальность 09.02.07 Информационные системы и программирование

Руководитель практики от колледжа:

*Калинин Арсений Олегович*

Руководитель практики от организации:

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Махнев Александр Анатольевич*

подпись

УТВЕРЖДАЮ:

Директор

Шеренцова Ольга Михайловна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
 Наименование организации

КОГПОБУ «Слободской колледж педагогики и

социальных отношений»

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

подпись расшифровка

М. П.

2025 уч. год

**Содержание**

1. Сведения об организации (базе практики)
2. Разработка и анализ технического задания
3. Разработка программного модуля по алгоритму в соответствии техническому заданию
4. Оформление документации на программный модуль
5. Выполнение отладки программного модуля
6. Выполнение тестирования программного модуля
7. Оформление результатов тестирования
8. Рефакторинг и оптимизации программного кода
9. Разработка программного модуля для мобильного устройства
10. Заключение.
11. Приложения к отчету: ссылка на репозиторий отчет в электронном виде, презентация для выступления, проект программного модуля, проект мобильного приложения и др. материалы.
12. **СВЕДЕНИЯ ОБ ОРГАНИЗАЦИИ (БАЗЕ ПРАКТИКИ)**
13. Описание базы практики:

Наименование базы практики – Кировское областное государственное профессиональное образовательное бюджетное учреждение «Слободской колледж педагогики и социальных отношений» (КОГПОБУ СКПиСО).

* Адрес – Кировская область, г. Слободской, ул. Рождественская, д. 69.
* ФИО директора – Шеренцова Ольга Михайловна.
* Номера телефона:
  + +7 (83362) 4-71-54
  + +7 (901) 479-17-07
* Веб-сайт: [slobkoll.ru](http://slobkoll.ru/)
* ФИО руководителя практики: Махнев Александр Анатольевич.

Образование в КОГПОБУ СКПиСО ориентировано на массовое обучение специалистов с рабочими навыками. Половина учебного времени отводится на практические работы в лабораториях. Это помогает выпускникам органично и быстро включаться в трудовые процессы на новых местах работы.

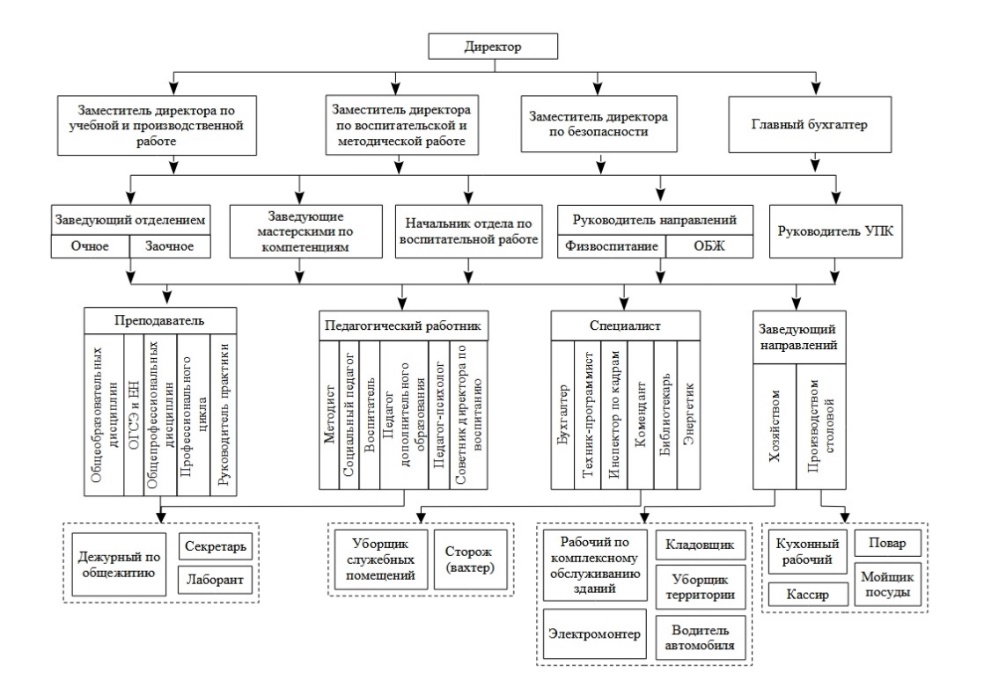
Схема организационной структуры представлена на рисунке 1.

Рисунок 1 – Схема организационной структуры.

1. Описание рабочего места.

Колледж предоставил отдельный кабинет для выполнения практических работ (Рисунок 2). Помимо этого, в данном кабинете проводились оцифровка документов, помощь подготовки к мероприятиям и прочие работы.



Рисунок 2 – Рабочее место на базе практики.

Комплектующие рабочего компьютера:

* Имя устройства: LAPTOP-R5L7BM3C
* Процессор: AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx 2.10 GHz
* Оперативная память: 8,00 ГБ
* Тип системы: 64-разрядная операционная система, процессор x64

1. **РАЗРАБОТКА И АНАЛИЗ ТЕХНИЧЕСКОГО ЗАДАНИЯ**

На данном этапе было проведено детальное изучение предоставленного технического задания (ТЗ). В рамках анализа ТЗ были уточнены ключевые требования к базе данных, определены основные сущности, их атрибуты и взаимосвязи. Для визуализации процессов и структуры данных были разработаны следующие диаграммы:

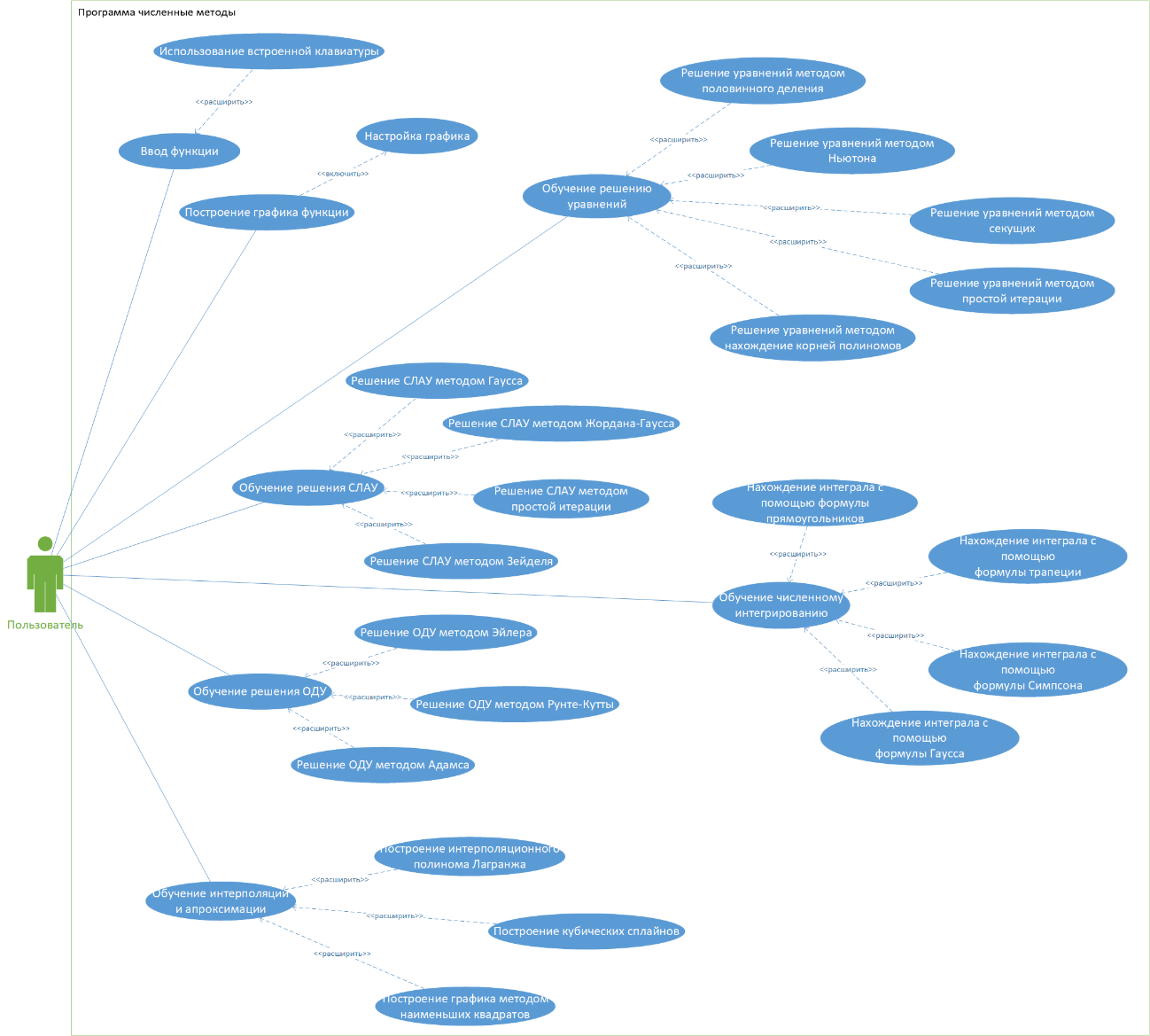
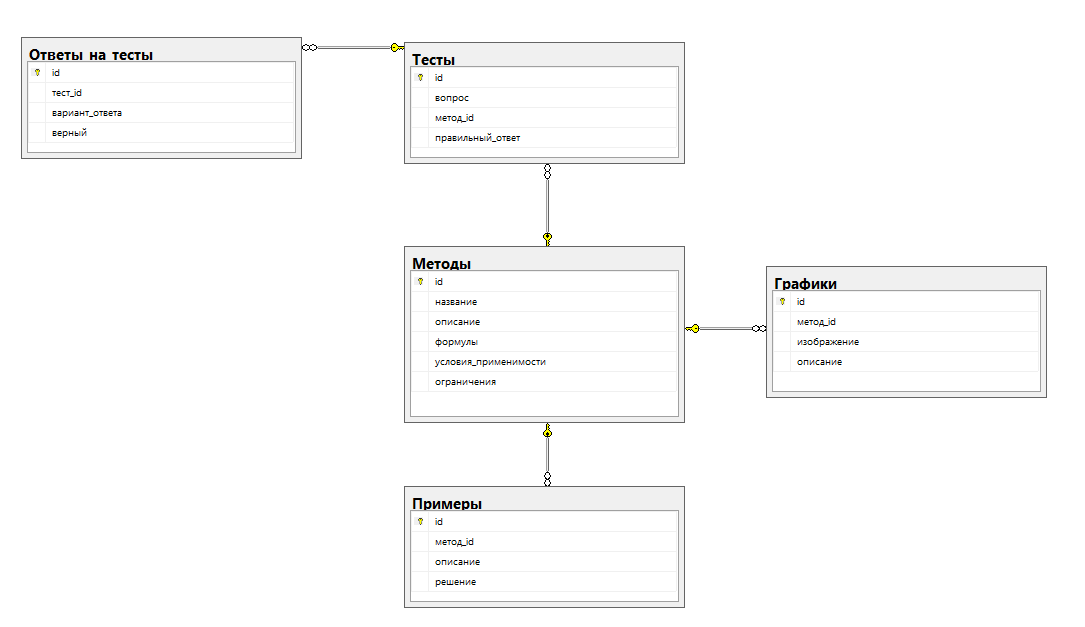
**Диаграмма вариантов использования** — для отображения основных сценариев взаимодействия пользователей с системой (Рисунок 3.1).

Рисунок 3.1 – Диаграмма вариантов использования.

**ER-диаграмма** — для проектирования структуры базы данных, включая сущности, их атрибуты и связи (Рисунок 3.2).

Рисунок 3.2 – Диаграмма базы данных.

На основе анализа ТЗ и разработанных диаграмм были сформулированы уточненные требования к реализации базы данных. Также был произведен выбор СУБД (MSServer) и подготовлена основа для следующего этапа — непосредственного создания базы данных.

1. **РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ ПО АЛГОРИТМУ В СООТВЕТСТВИИ ТЕХНИЧЕСКОМУ ЗАДАНИЮ**

На данном этапе была выполнена разработка программного модуля, включающего API и настольное приложение, в строгом соответствии с техническим заданием (ТЗ). Основной задачей было реализовать функционал, который обеспечивает корректную работу системы и ее интеграцию с другими компонентами.

**Разработка API**

API был разработан с использованием ASP.NET для C# и с использованием БД по технологии ADO NET.EDM. Были реализованы следующие методы:

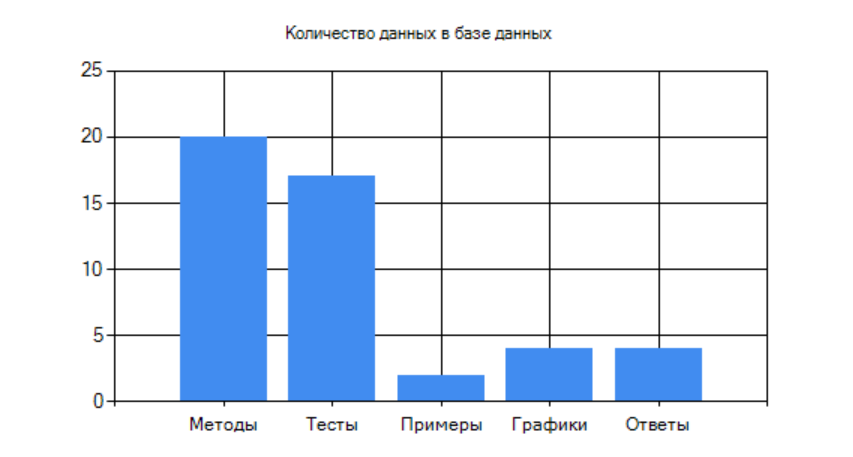
* Диаграмма, отображающая количество данных в базе данных (Рисунок 4.1).
* Получение списка вопросов теста — метод GET, возвращающий список всех вопросов с их основными характеристиками (Рисунок 4.2).
* Работа с графиками — методы POST для отправки данных о графике и GET для получения списка графиков (Рисунок 4.3).

Рисунок 4.1 – Диаграмма количества данных в БД.

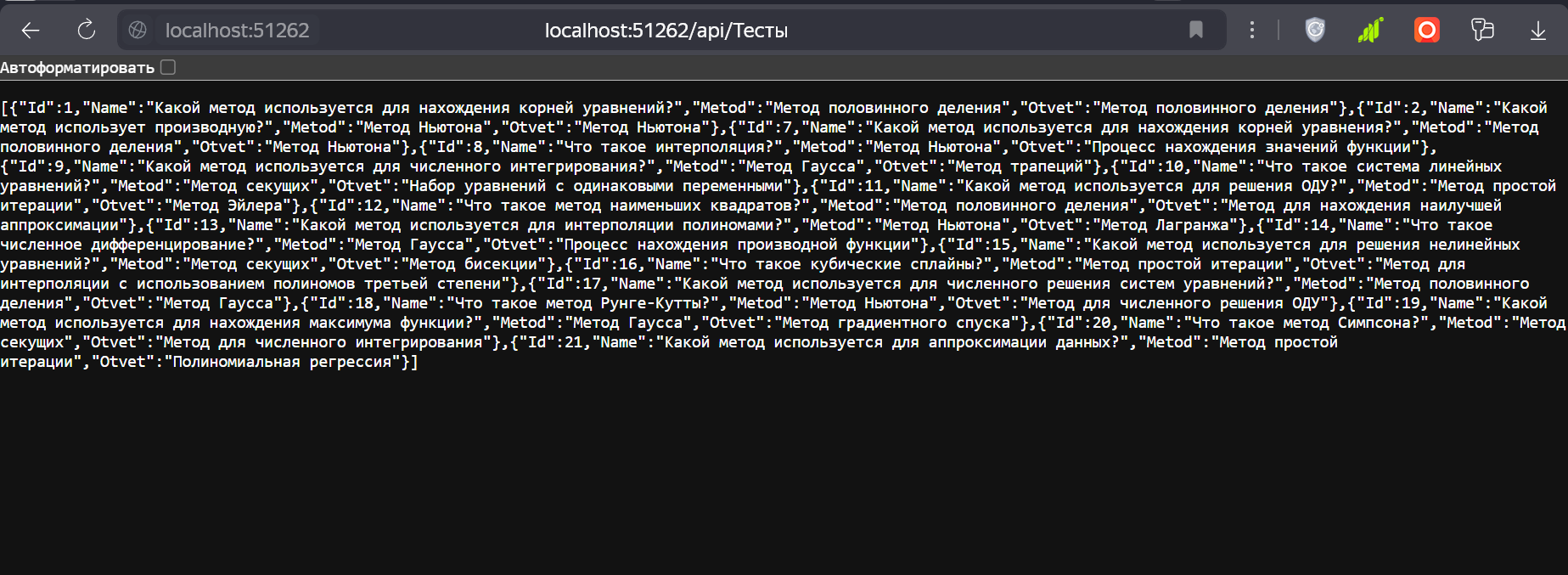
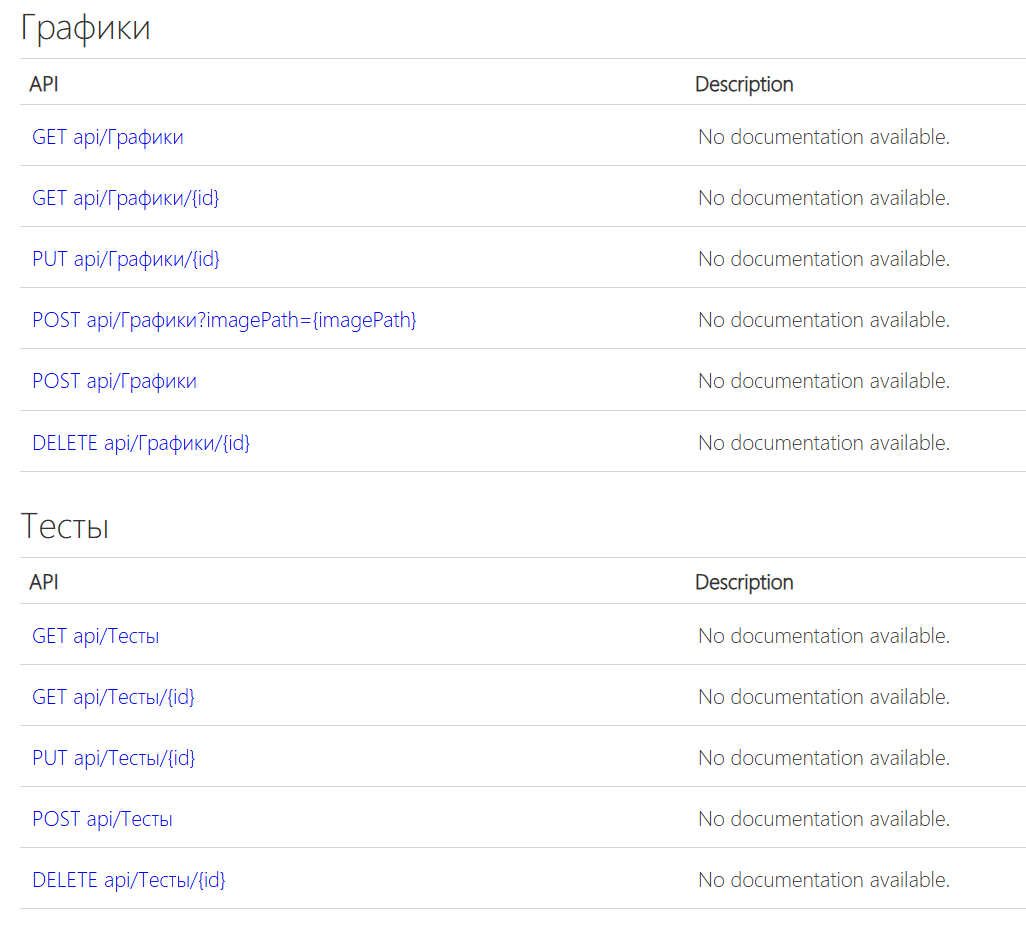
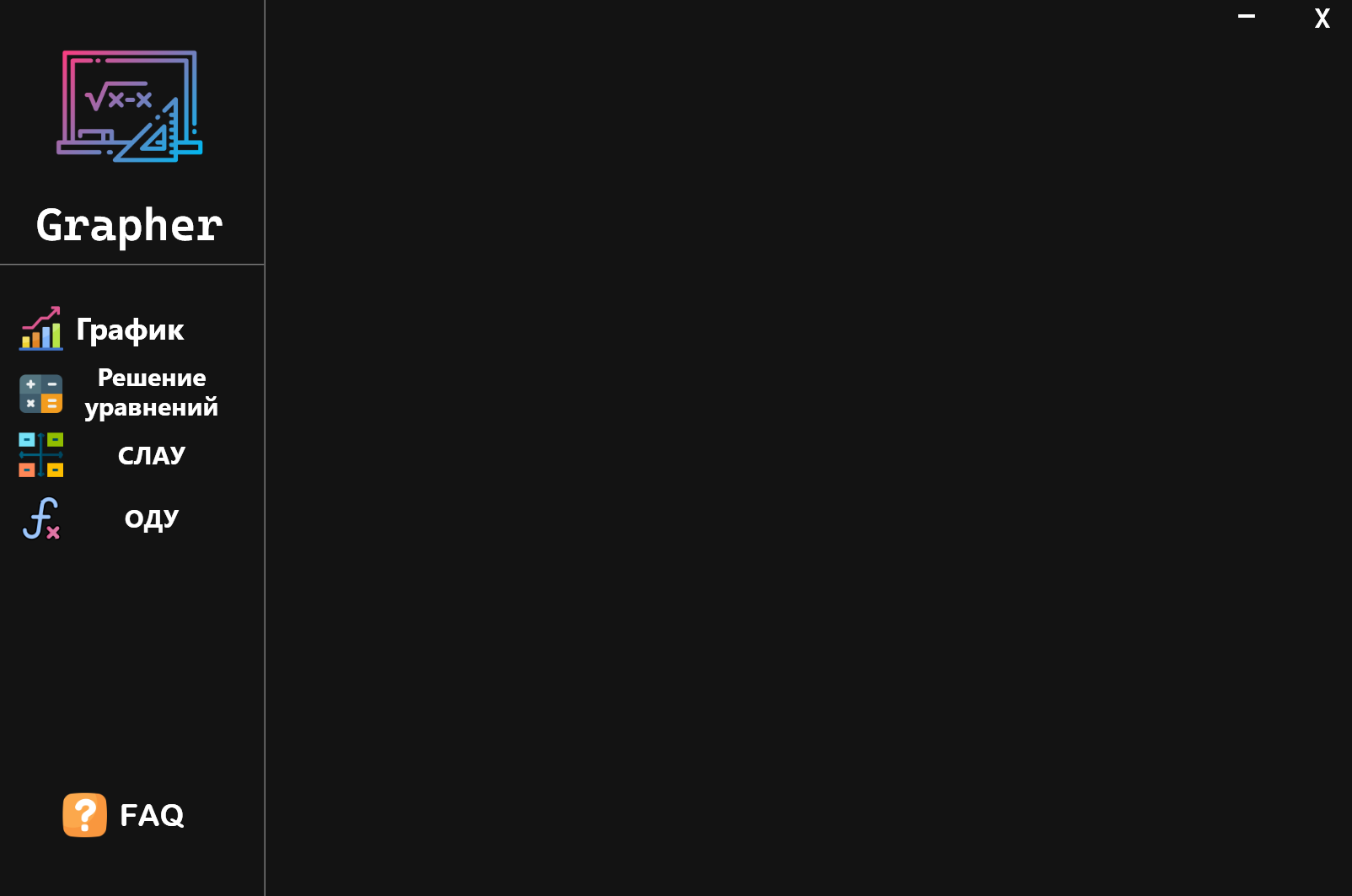
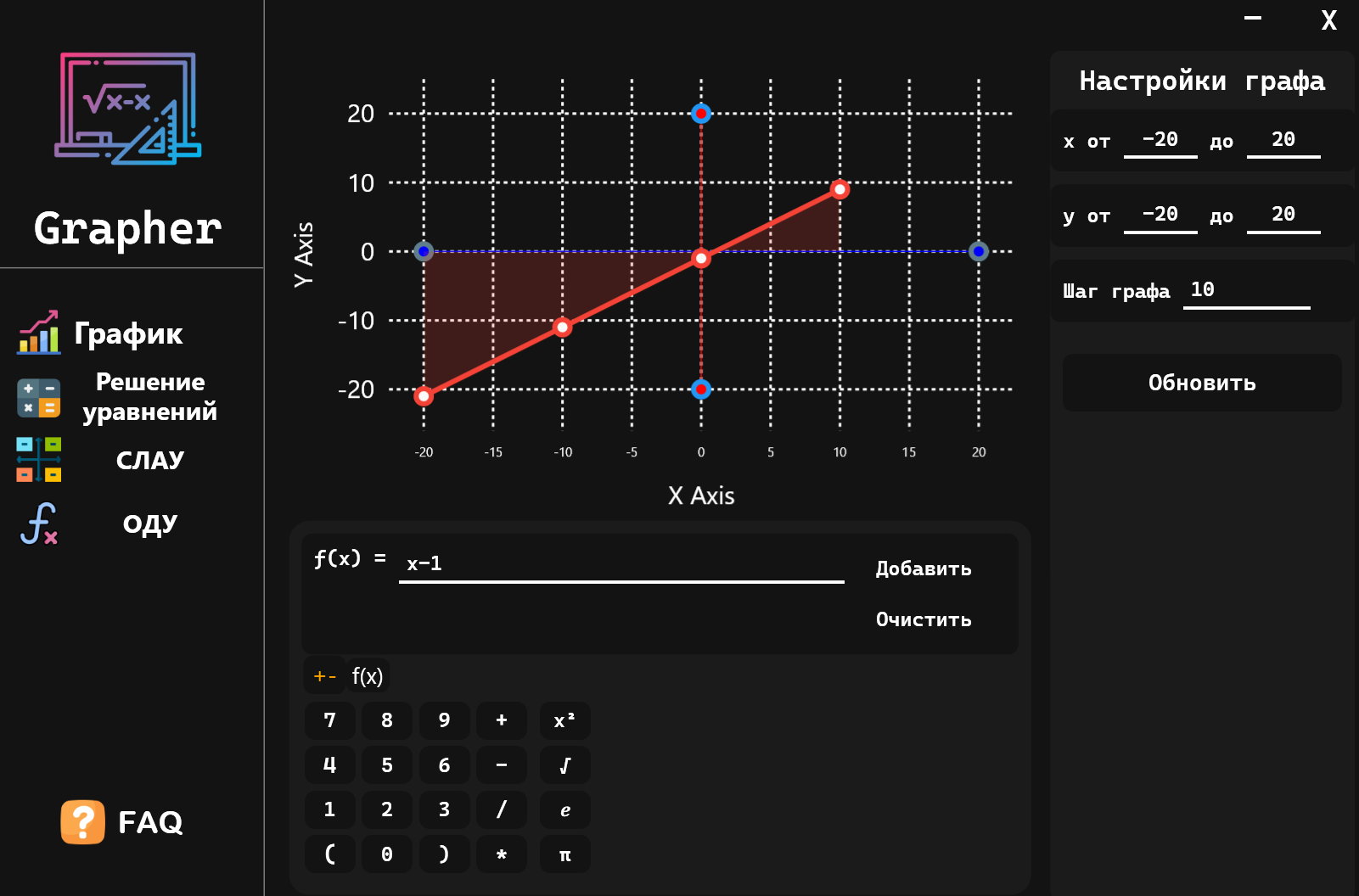
Рисунок 4.2 – Список вопросов.

Рисунок 4.3 – Методы POST и GET.

**Создание настольного приложения**

Настольное приложение было разработано с использованием WPF для C#. Основные функции приложения:

1. Интуитивно понятный пользовательский интерфейс (Рисунок 5.1).
2. Многоуровневая навигация.
3. Ввод и обработка данных (Рисунок 5.2).
4. Вычисления математических задач.
5. Обучение пользователей теоретическим материалом, краткое описание каждого метода (Рисунок 5.3).
6. Обработка ошибок.

Рисунок 5.1 – Главное окно программы.

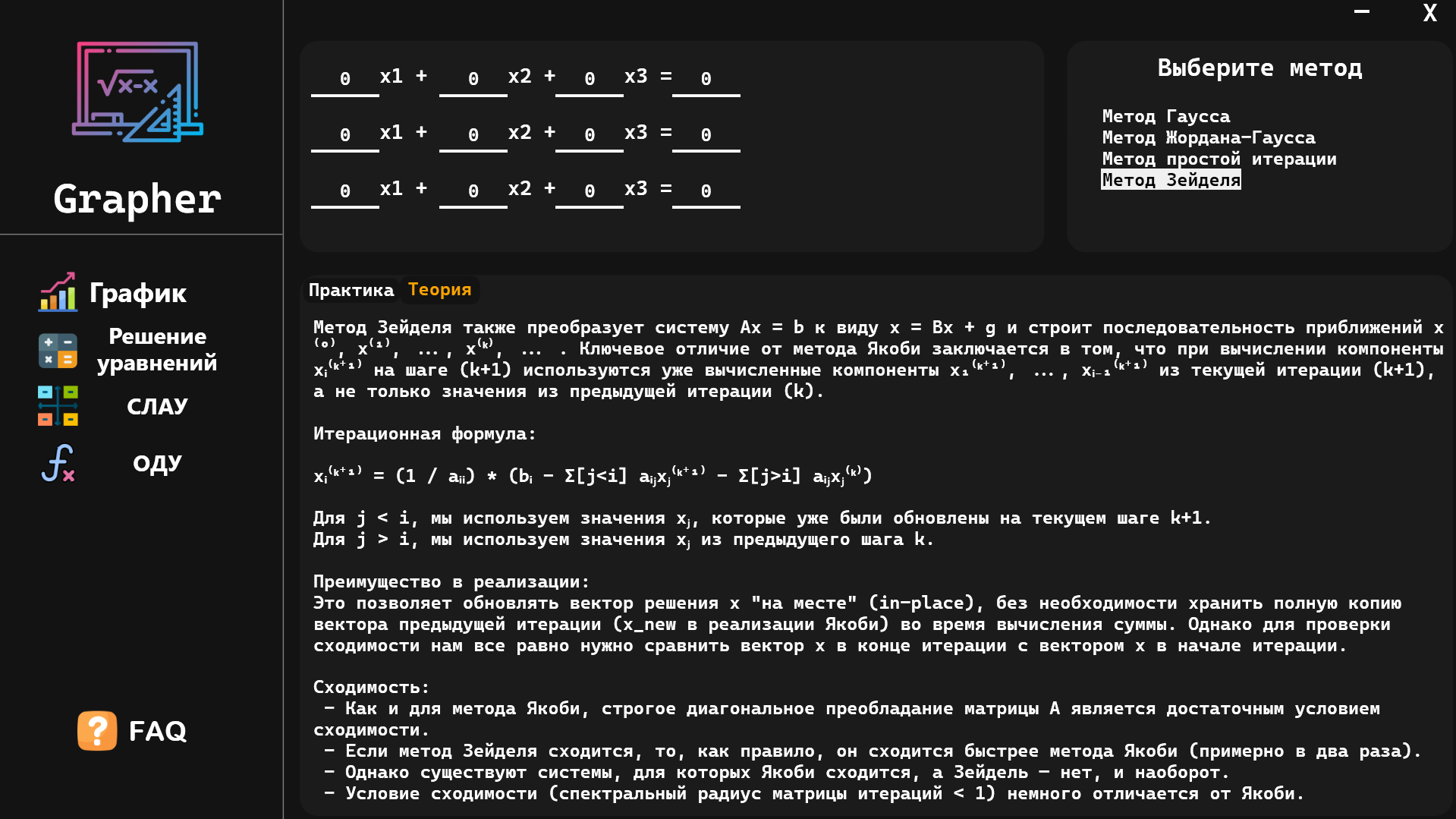
Рисунок 5.2 – Ввод функции и построение графика.

Рисунок 5.3 – Теоретический материал.

Результатом этапа стал полностью функциональный программный модуль, включающий API и настольное приложение, соответствующие всем требованиям ТЗ. Модуль готов к дальнейшему тестированию и отладке.

1. **ОФОРМЛЕНИЕ ДОКУМЕНТАЦИИ НА ПРОГРАММНЫЙ МОДУЛЬ**

На данном этапе была выполнена разработка и оформление документации на программный модуль в соответствии с требованиями технического задания (ТЗ) и стандартами разработки. Документация включает в себя описание всех компонентов системы, инструкции по использованию и технические спецификации.

**Руководство оператора настольного приложения**

Программа «Численные методы» предназначена для сопровождения занятий по данной дисциплине, включая решение математических задач с использованием различных численных методов и теоретический материал. Она предоставляет пользователям возможность выполнять вычисления и визуализировать результаты.

**Функциональное назначение программы**

Программа «Численные методы» обеспечивает выполнение следующих функций:

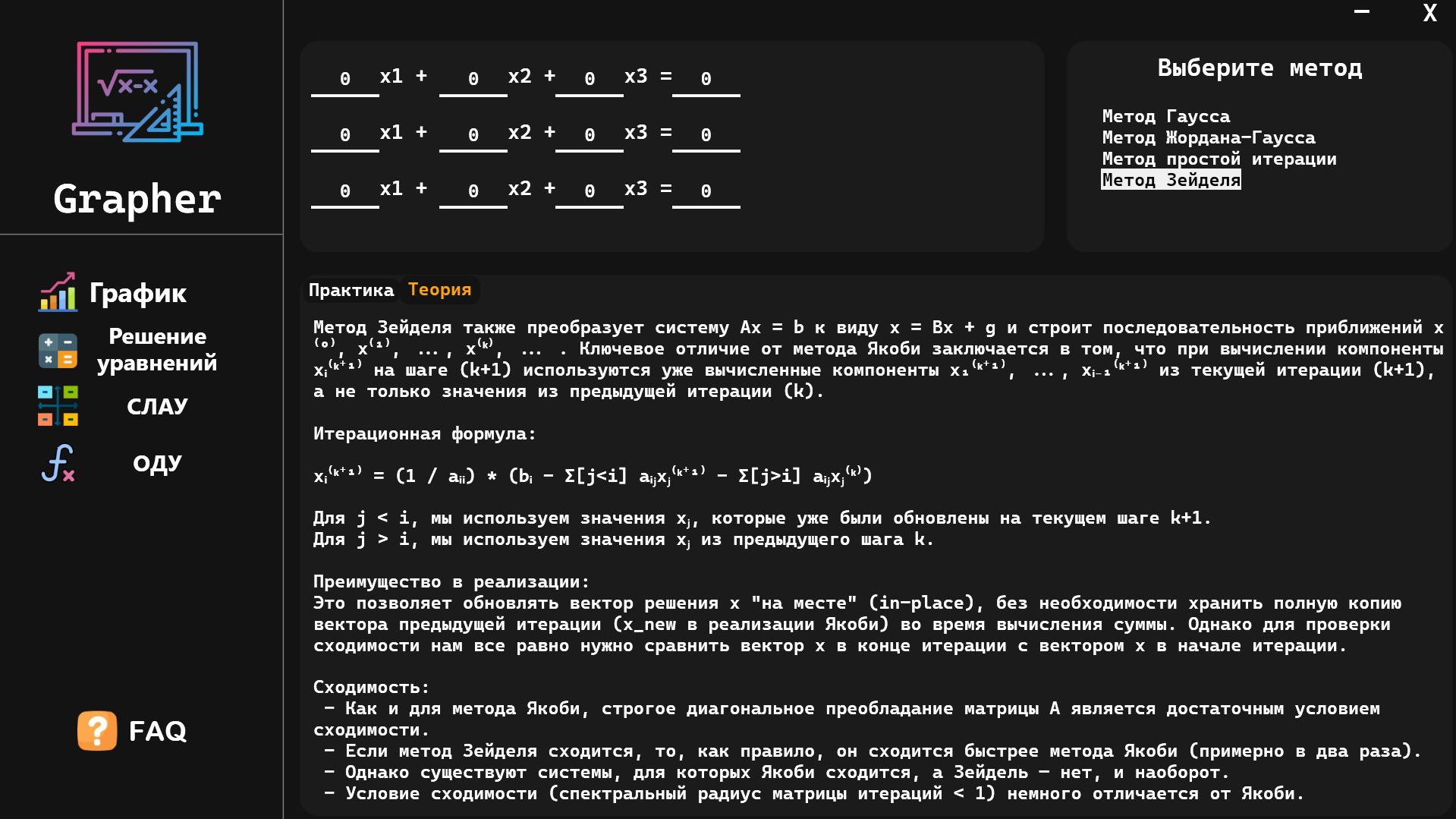
* Обучение пользователей решения задач численными методами (Рисунок 6.1).

Рисунок 6.1 – Теоретический материал.

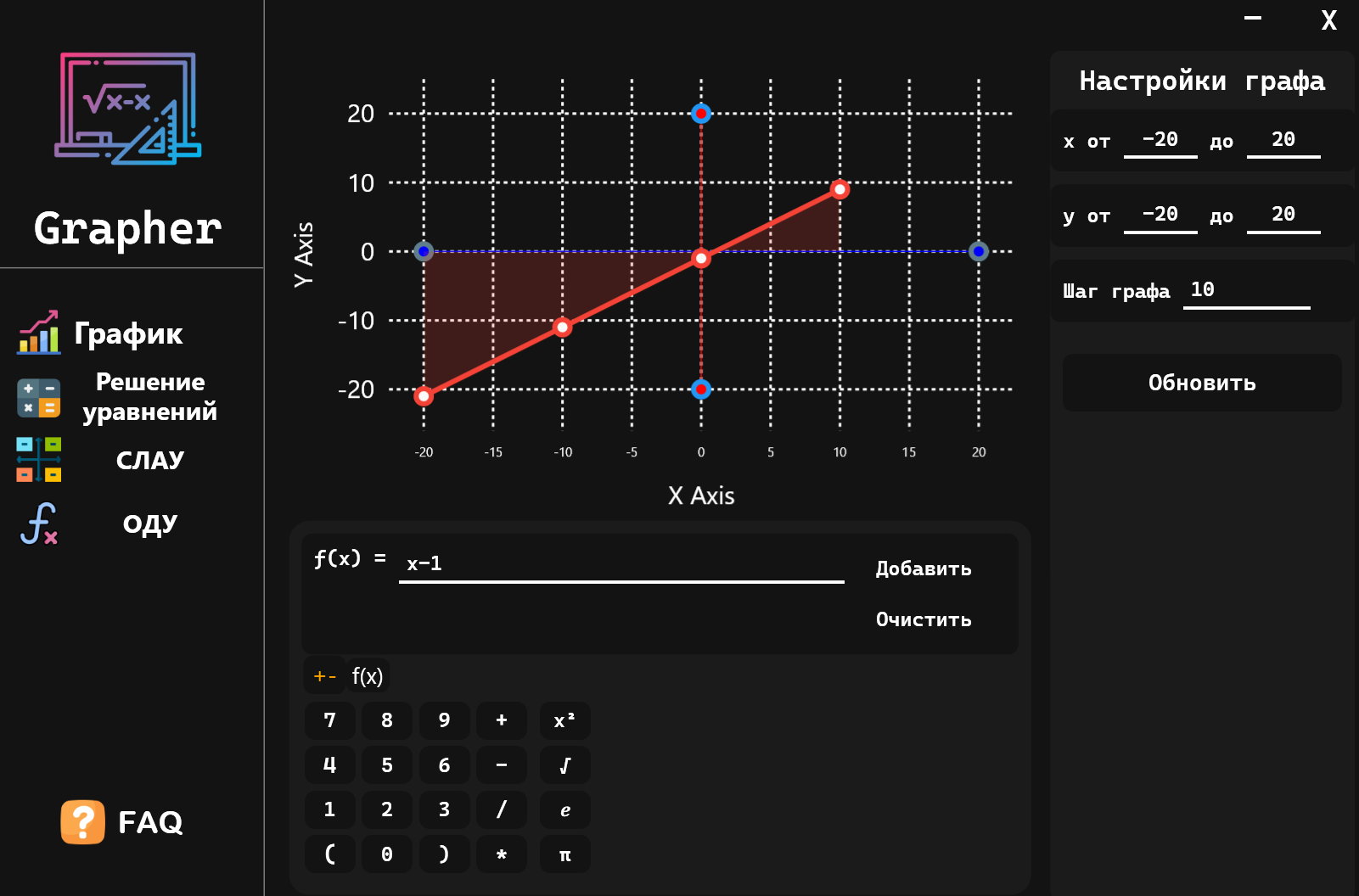
* Ввод функции (Рисунок 6.2).

Рисунок 6.2 – Ввод функции и построение графика.

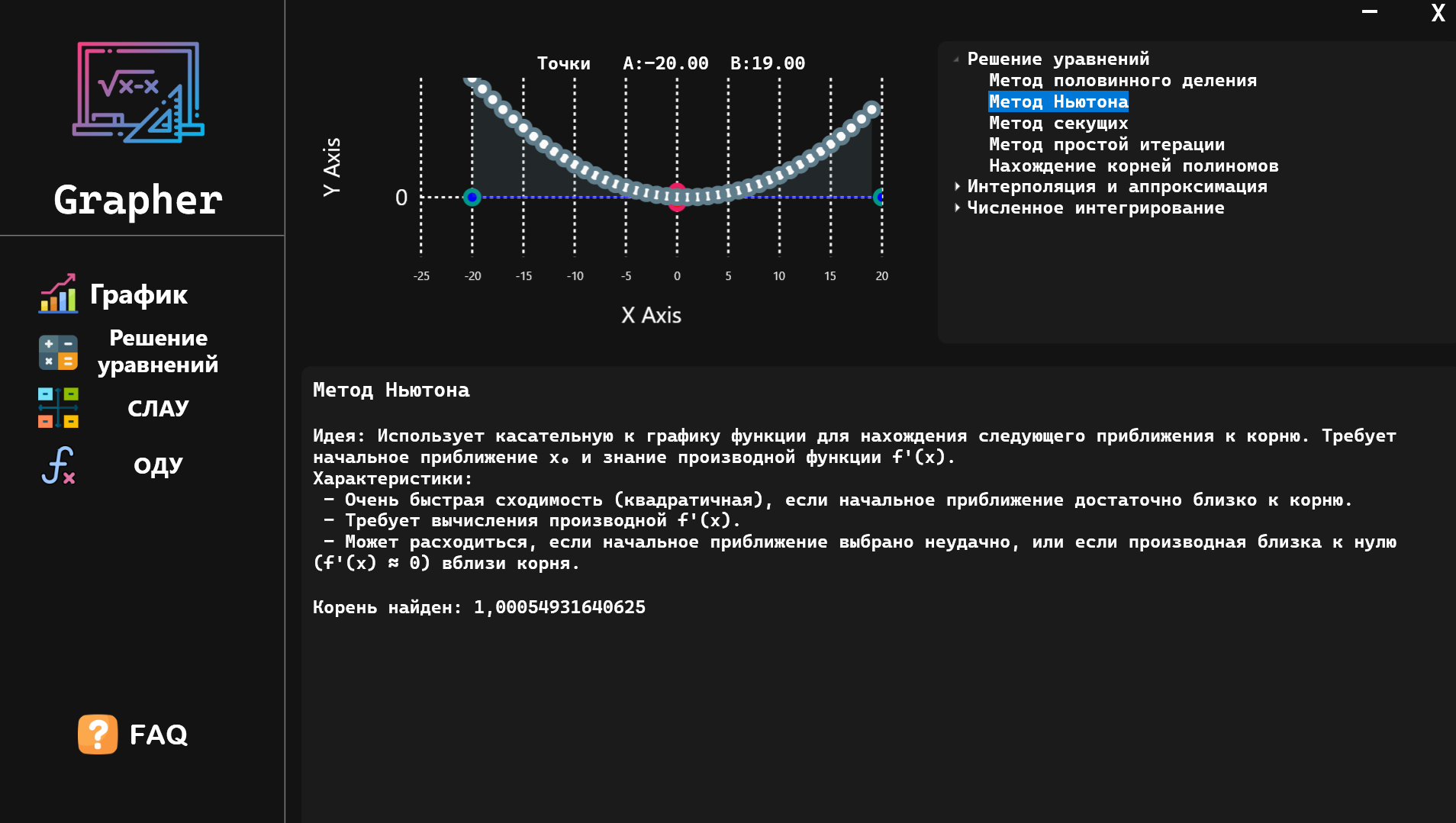
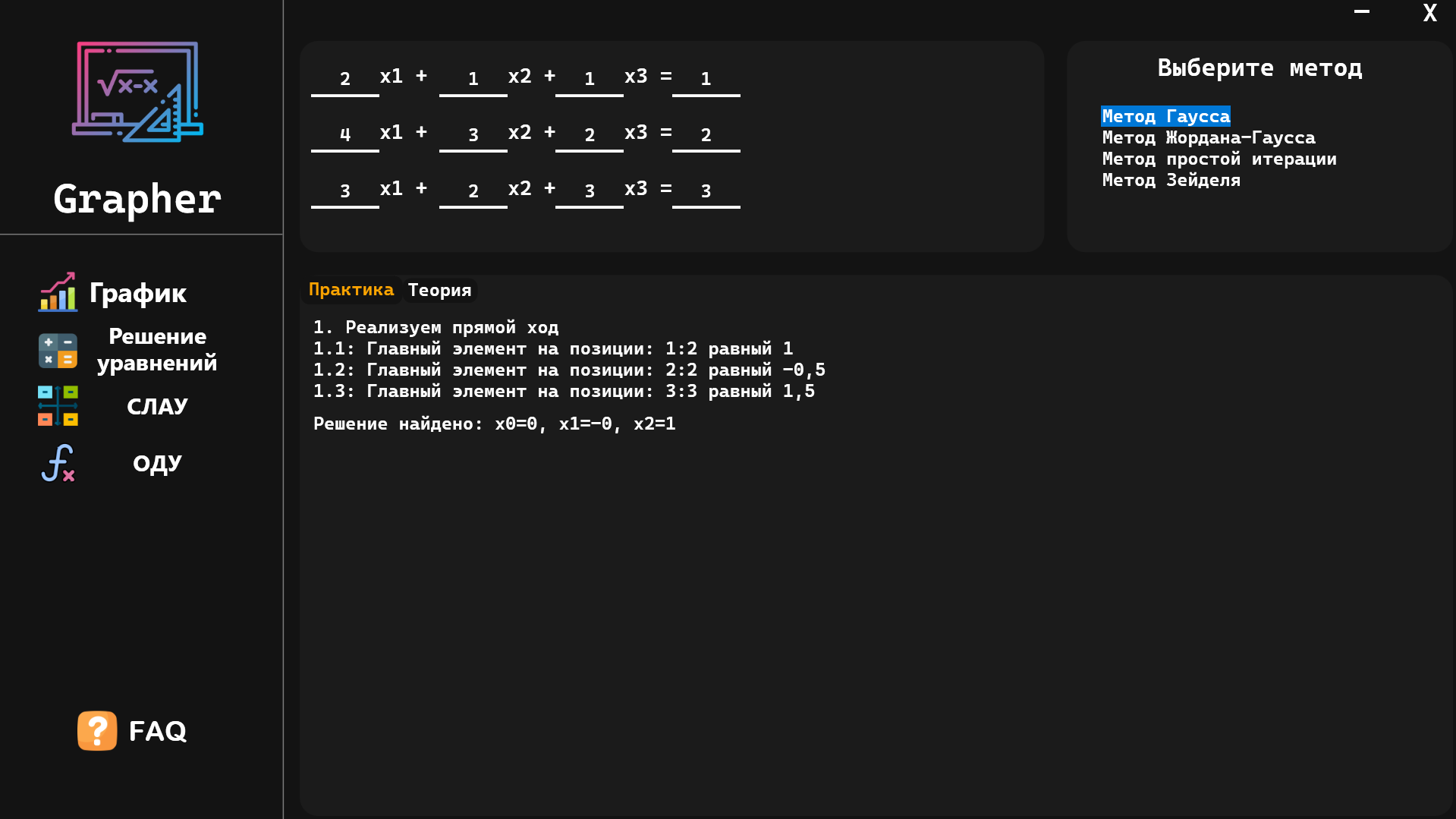
* Решение уравнений: Использование методов Ньютона, секущих, половинного деления, простой итерации, корней полиномов (Рисунок 6.3).

Рисунок 6.3 – Решение функции методом Ньютона.

* Решение систем линейных уравнений: Применение методов Гаусса, Жордана-Гаусса, простой итерации, Зейделя (Рисунок 6.4).

Рисунок 6.4 – Решение СЛАУ методом Гаусса.

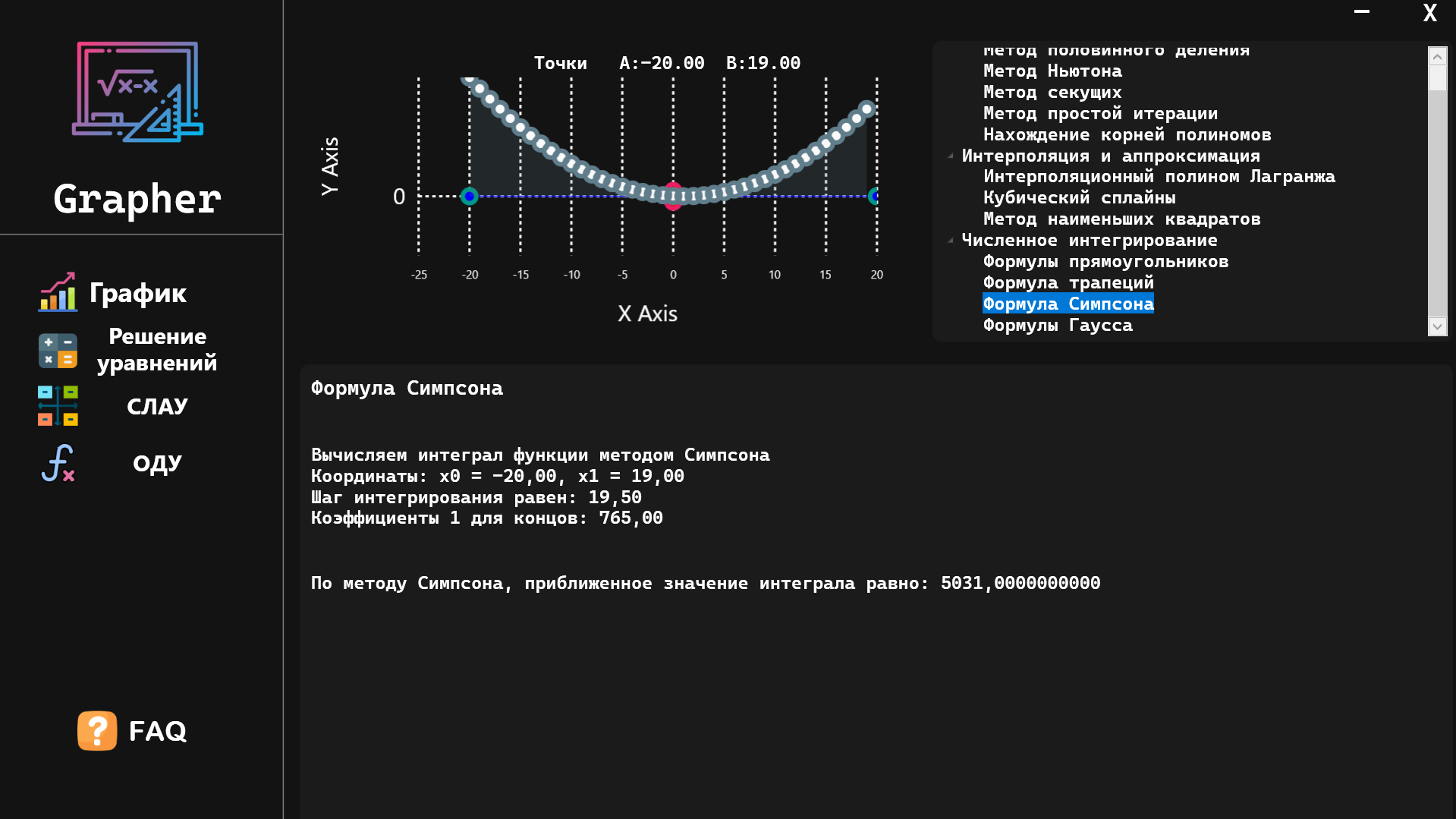
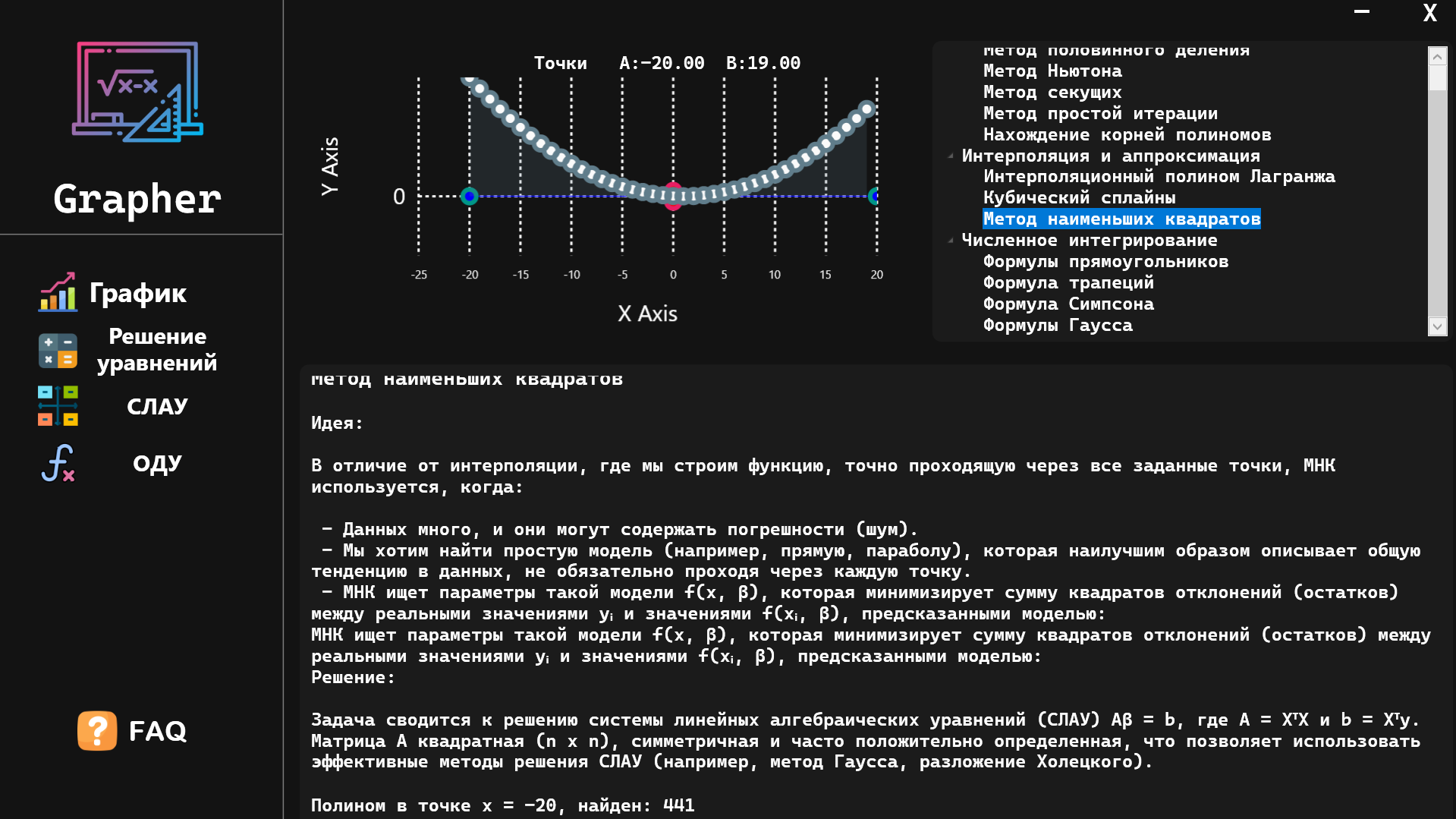
* Численное интегрирование: Использование формул прямоугольников, трапеций, Симпсона и Гаусса (Рисунок 6.5).

Рисунок 6.5 – Вычисление интеграла методом Симпсона.

* Интерполяция: Применение полинома Лагранжа, кубических сплайнов и метода наименьших квадратов (Рисунок 6.6).

Рисунок 6.6 – Вычисление полинома методом наименьших квадратов.

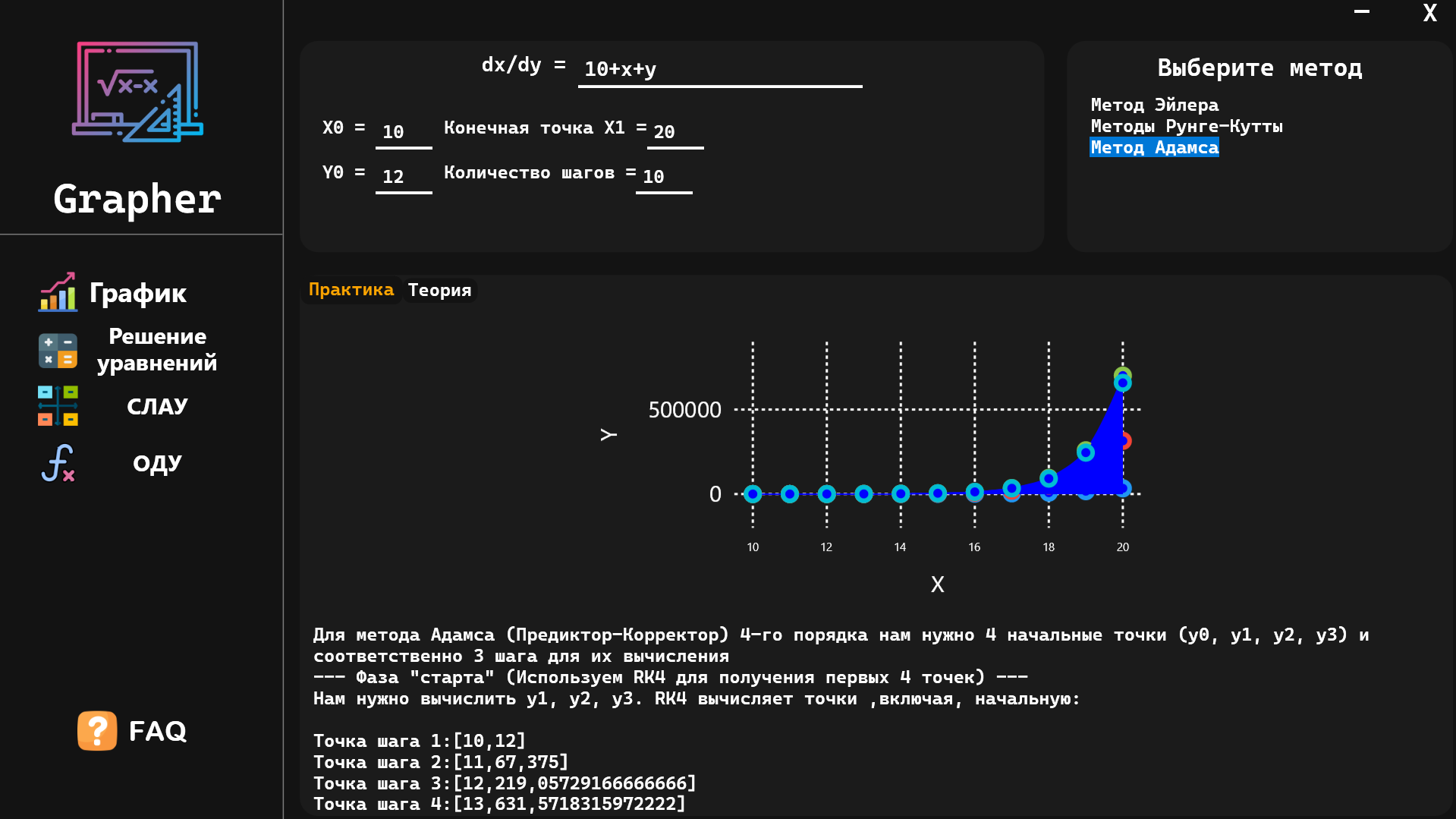
* Решение обыкновенных дифференциальных уравнений: методов Эйлера, Рунте-Курты и Адамса (Рисунок 6.7).

Рисунок 6.7 – Вычисление ОДУ методом Адамса.

* Визуализация результатов: Графическое представление данных и результатов вычислений.
* Настройка построенных графиков.

1. **ВЫПОЛНЕНИЕ ОТЛАДКИ ПРОГРАММНОГО МОДУЛЯ**

На данном этапе была проведена отладка программного модуля, в ходе которой были исправлены ошибки в коде, оптимизирована работа API и проверена корректность функционирования всех компонентов системы. Основной задачей было обеспечить стабильную работу модуля в соответствии с требованиями технического задания (ТЗ).

**Основные этапы отладки**

**Исправление ошибок в коде**

Проведен анализ кода на наличие синтаксических, логических и runtime-ошибок.

Устранены ошибки, связанные с обработкой данных, работой с базой данных и взаимодействием между компонентами системы.

Исправлены проблемы, выявленные в процессе тестирования (например, некорректная обработка исключений, ошибки в расчетах и т.д.).

**Оптимизация работы API**

Проведена оптимизация методов API для повышения производительности и уменьшения времени обработки запросов.

Улучшена обработка ошибок (например, при неверных входных данных или отсутствии доступа).

Проверена корректность работы всех методов:

* Отображение диаграммы.
* Получение списка вопросов теста.
* Работа с графиками.

**Проверка работы всех функций настольного приложения**

Проведено тестирование функционала настольного приложения для каждого раздела:

* Обучение пользователей теоретическим материалом: Краткое описание каждого метода представлено в теоретическом материале.
* Ввод данных: Проверена функция ввода функции и построения графика.
* Решение уравнений: Использованы методы Ньютона, секущих, половинного деления, простой итерации и корней полиномов для решения уравнений.
* Решение систем линейных уравнений: Применены методы Гаусса, Жордана-Гаусса, простой итерации и Зейделя для решения систем линейных уравнений.
* Численное интегрирование: Использованы формулы прямоугольников, трапеций, Симпсона и Гаусса для численного интегрирования.
* Интерполяция: Применены полином Лагранжа, кубические сплайны и метод наименьших квадратов для интерполяции данных.
* Решение обыкновенных дифференциальных уравнений: Использованы методы Эйлера, Рунге-Кутты и Адамса для решения обыкновенных дифференциальных уравнений.
* Визуализация результатов: Проверено графическое представление данных и результатов вычислений, что позволяет пользователям наглядно видеть результаты работы программы.

**Используемые инструменты**

* Отладчики (Visual Studio).
* Логирование — для отслеживания выполнения кода и выявления ошибок.
* Тестовые данные — для проверки всех возможных сценариев.
* Инструменты для тестирования API (Postman).

**Результаты отладки**

* Все выявленные ошибки были успешно исправлены.
* Работа API оптимизирована, что повысило производительность системы.
* Подтверждена корректность работы всех функций приложения.
* Обеспечено соответствие модуля требованиям ТЗ.

Программный модуль был успешно отлажен, все компоненты системы работают стабильно и готовы к дальнейшему тестированию.

1. **ВЫПОЛНЕНИЕ ТЕСТИРОВАНИЯ ПРОГРАММНОГО МОДУЛЯ**

На этапе тестирования была проведена проверка программного модуля для обеспечения его корректной работы и соответствия требованиям технического задания (ТЗ).

Тестирование API было выполнено с использованием инструмента Postman, что позволило проверить основные функциональные возможности модуля. В процессе тестирования были отправлены запросы на создание графиков с различными наборами данных. Все тестовые случаи, включая как корректные, так и некорректные данные, были успешно обработаны.

#### Шаги для тестирования API с помощью Postman:

1. **Установка Postman**.
2. **Создание нового запроса**:
   * Открыть Postman и создать новый запрос (New Request).
   * Выбрать метод POST.
   * Ввести URL API: http://localhost:5162/api/Графики.
3. **Настройка заголовка**:
   * Добавление заголовка Content-Type со значением application/json.
4. **Создайте тело запроса**:
   * В разделе «Body» выберать raw и установите формат на JSON.
   * Ввести JSON-объект – Графики:

{

"метод\_id": 2,

"изображение": "data:image/png;base64,iVBORw0KGgoAAAA

NSUhEUgAAAfQAAAH9CAMAAADbB35wAAAAGXRFWHRTb2Z0d2FyZQBBZG9iZSBJbWFnZVJlYWR5ccllPAAAABJQTFRF",

"описание": "проверка отправки"

}

1. **Отправить запрос**.
2. **ОФОРМЛЕНИЕ РЕЗУЛЬТАТОВ ТЕСТИРОВАНИЯ**

Результаты тестирования были систематизированы и оформлены в виде отчетов, которые включают:

**Описание тестовых сценариев**

В данном разделе представлен подробный перечень всех проверенных функций и сценариев использования программного модуля. Например, были проверены сценарии проверки расчета интеграла и корней уравнений.

**Отчеты были оформлены в виде онлайн документации:**

Все результаты тестирования, включая отчеты, были загружены в репозиторий на GitHub. Это обеспечивает удобное хранение, версионность и доступ к документам для всех участников проекта. Каждый отчет был размещен в папке репозитория, что позволяет легко находить нужные материалы и отслеживать изменения.

Таким образом, результаты тестирования были тщательно проанализированы и оформлены в удобном для использования виде, что позволяет эффективно работать с ними на всех этапах жизненного цикла проекта.

1. **РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ ДЛЯ МОБИЛЬНОГО УСТРОЙСТВА**

На данном этапе было разработано мобильное приложение, предназначенное для справочника по численным методам. Приложение предоставляет удобный интерфейс для работы с разделами решения уравнений, систем линейных уравнений, интерполяцию и аппроксимацию, численное интегрирование и численное дифференцирование.

**Основные функции мобильного приложения:**

**Меню приложения**

Приложение включает удобное меню и четыре раздела:

* Теоретический материал по каждому разделу.
* Теоремы каждого раздела.
* Тест по каждому разделу.
* Результаты тестов.

**Работа с локальной базой данных**

Для хранения данных используется локальная база данных SQLite, которая интегрирована в приложение через NuGet-пакет.

**Технологии и инструменты разработки:**

* Среда разработки: Xamarin (кроссплатформенная разработка для iOS и Android).
* База данных: SQLite (легковесная и производительная локальная БД).

**NuGet-пакеты:**

* SQLite-net-pcl — для работы с SQLite.

**Особенности реализации:**

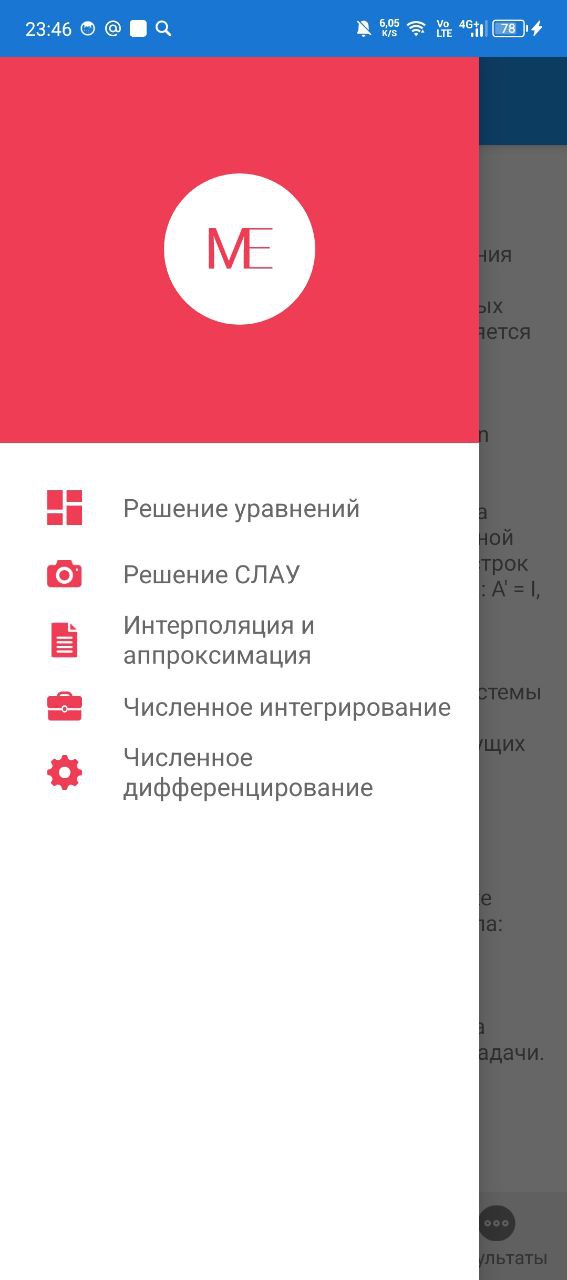
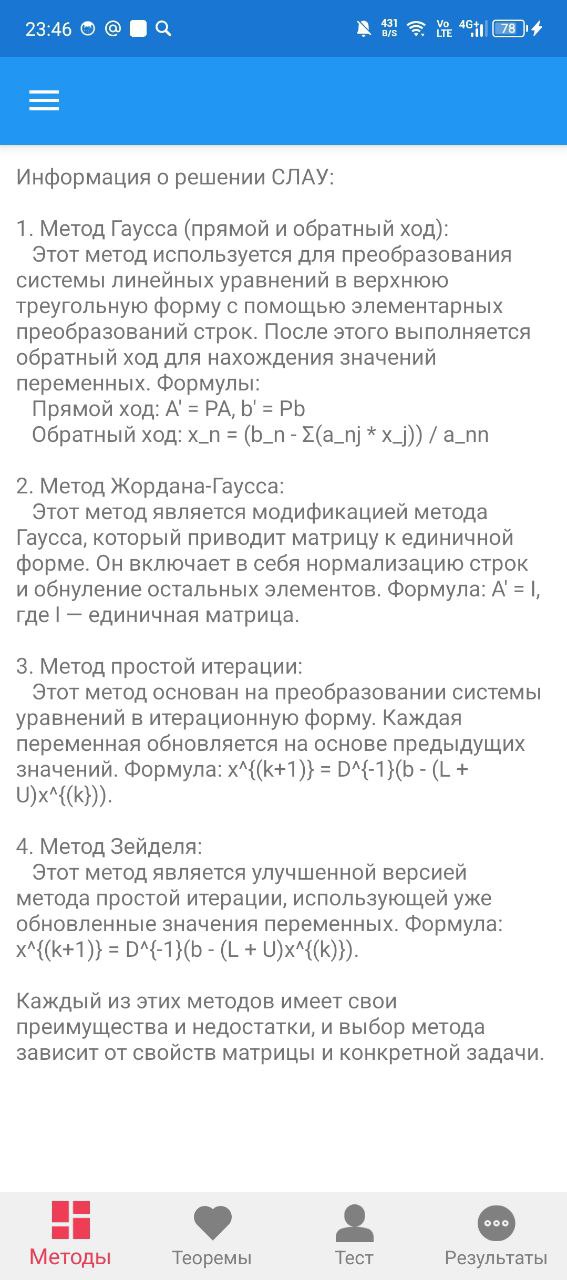
* ****Интерфейс: Приложение имеет интуитивно понятный и отзывчивый интерфейс, адаптированный под мобильные устройства (Рисунки 7.1 и 7.2).

Рисунок 7.1 – Разделы приложения. Рисунок 7.2 – Меню приложения.

* Офлайн-режим: Возможность работы с данными без подключения к интернету благодаря локальной базе данных.

Разработано мобильное приложение, которое соответствует всем требованиям технического задания. Обеспечена удобная работа для роли кладовщика, а также возможность просмотра данных другими ролями. Приложение готово к тестированию на реальных устройствах и эмуляторах.

1. **РЕФАКТОРИНГ И ОПТИМИЗАЦИИ ПРОГРАММНОГО КОДА**

Рефакторинг и оптимизация кода – это важный этап в разработке программного обеспечения, направленный на улучшение его внутренней структуры, читаемости, производительности и надёжности, при этом сохраняя функциональность. При разработке программы были проведены следующие методы рефакторинга и оптимизации:

1. Разделение логики на методы

Каждый метод, такой как Newton, Secant, FixedPointIteration и др., отвечает за конкретную задачу. Это делает код более модульным и упрощает его тестирование и поддержку. Например, если нужно изменить логику метода Ньютона, это можно сделать, не затрагивая другие методы.

1. Использование комментариев

В каждом методе написано описание операции, которое объясняет, как работает метод и его характеристики. Это улучшает документацию кода и помогает быстрее понять, что делает каждый метод.

1. Проверка условий и обработка ошибок

Идет проверка условий, такие как наличие разных знаков на концах интервала в методе Bisection, и обрабатываю случаи, когда производная близка к нулю в методе Newton (Рисунок 7.1). Это помогает избежать ошибок во время выполнения и делает код более надежным.

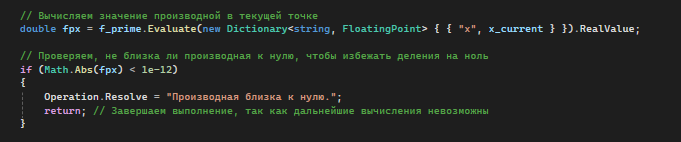
****

Рисунок 8.2 – Рефакторинг и оптимизация кода (проверка условий)

1. Использование лямбда-выражений

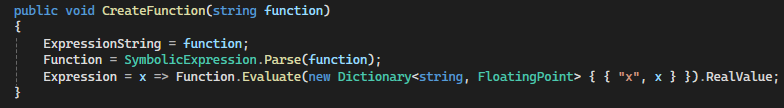
В методе CreateFunction используется лямбда-выражение для определения функции Expression (Рисунок 7.2). Это позволяет динамически создавать функции на основе входных данных, что делает код более гибким.

Рисунок 7.2 – Рефакторинг и оптимизация кода (лямбда-выражение)

1. Использование try-catch для обработки исключений

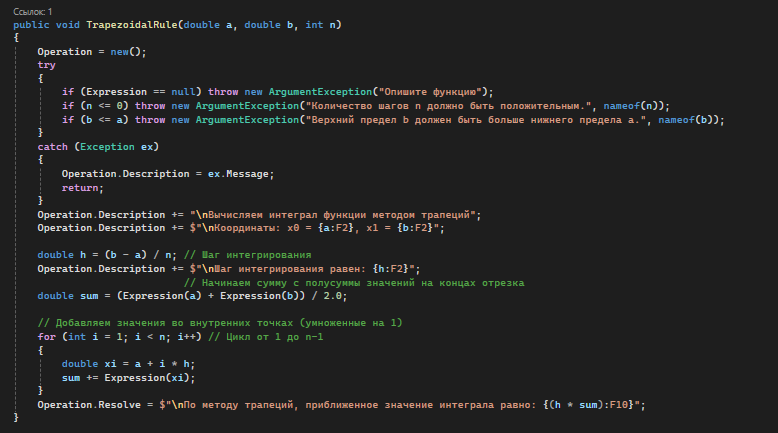
В методе TrapezoidalRule используется блок try-catch для обработки возможных исключений (Рисунок 7.3). Это позволяет коду оставаться устойчивым к ошибкам и предоставляет пользователю информацию о том, что пошло не так.

Рисунок 7.3 – Рефакторинг и оптимизация кода (блоки try-catch)

1. Избежание дублирования и удаление «мёртвого» кода.

Неиспользуемый код сам по себе не влияет на работоспособность программного обеспечения, но он отнимает время у разработчиков.

1. **ЗАКЛЮЧЕНИЕ.**

В ходе производственной практики была успешно реализована разработка нескольких ключевых компонентов, включая API, базу данных, WPF-приложение и мобильное приложение. Каждый из этих элементов был создан с учетом современных стандартов и технологий, что позволило обеспечить высокую производительность и удобство использования.

Создание API обеспечило возможность интеграции различных систем и упрощение взаимодействия между ними. База данных была спроектирована для эффективного хранения и обработки данных, что является основой для работы всех приложений. WPF-приложение предоставило пользователям интуитивно понятный интерфейс и функциональность, необходимую для выполнения задач. Мобильное приложение расширило доступность сервиса, позволяя пользователям работать с системой в любом месте и в любое время.

Производственная практика дала мне ценный опыт, который значительно расширил мои профессиональные знания и навыки. Я научился работать с современными технологиями разработки, улучшил свои навыки программирования и проектирования, а также приобрел опыт командной работы и управления проектами. Этот опыт станет важной основой для моего дальнейшего профессионального роста и развития в области информационных технологий.

1. **ПРИЛОЖЕНИЯ К ОТЧЕТУ.**

Все материалы, программные модули и документация, созданные в рамках проекта, были систематизированы и загружены в репозиторий на GitHub.

**Ссылка на репозиторий:**

https://github.com/Lada-Sem/PM01.git