Министерство образования Республики Беларусь
Учреждение образования
«ВИТЕБСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Информационные системы и технологии»

КУРСОВОЙ ПРОЕКТ

по дисциплине «Разработка и оптимизация интеллектуальных систем»

на тему «Приложение для визуализации методов одномерной оптимизации»

Выполнил:
студент факультета ЗФ

группы ИТс-12 _____ Чупринский М. В.
подпись *ФИО*

Проверил:
Руководитель старший преподаватель
должность

_____ _____ Бизюк А. Н.
ученая степень, ученое звание *ФИО*

_____ _____
отметка о допуске к защите

«____» 2025 г . _____
дата *подпись руководителя*

Витебск 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире задачи оптимизации занимают ключевое место в самых разных областях науки и техники. Они встречаются в инженерных расчётах, экономическом моделировании, управлении технологическими процессами, анализе данных и множестве других сфер. Оптимизация позволяет находить наилучшие решения в условиях ограниченных ресурсов, что особенно важно в эпоху высоких требований к эффективности и точности вычислений.

Одномерная оптимизация представляет собой частный случай общей задачи поиска экстремума, когда целевая функция зависит только от одной переменной. Несмотря на кажущуюся простоту, такие задачи имеют широкое практическое применение: от настройки параметров оборудования до анализа экспериментальных данных. Важность одномерной оптимизации обусловлена тем, что она часто выступает как составная часть более сложных многомерных задач, а также используется в качестве тестовой модели для изучения свойств алгоритмов.

Методы одномерной оптимизации можно условно разделить на две большие группы:

— методы, использующие производные (градиентные, метод Ньютона и др.);

— методы, не требующие вычисления производных (интервальные методы, метод золотого сечения, метод Фибоначчи и др.) [1].

Каждая из этих групп имеет свои преимущества и ограничения, связанные с гладкостью функции, наличием шумов в данных, вычислительной сложностью и устойчивостью к погрешностям.

Важным аспектом изучения методов оптимизации является их визуализация. Графическое представление процесса поиска экстремума позволяет наглядно продемонстрировать, как алгоритм изменяет приближения, как сужается интервал поиска, как быстро достигается заданная

точность. Такой подход особенно полезен в образовательных целях, а также при исследовании свойств алгоритмов на тестовых функциях.

Актуальность темы исследования определяется несколькими факторами:

— возрастающей ролью оптимизационных методов в науке и технике;

— необходимостью глубокого понимания алгоритмов для их эффективного применения;

— важностью наглядных средств обучения и анализа;

— потребностью в сравнительном исследовании методов по критериям точности, скорости сходимости и устойчивости.

Объект исследования — методы одномерной оптимизации. Предмет исследования — алгоритмы одномерной оптимизации и их сравнительный анализ.

Цель работы — исследование и сравнение методов одномерной оптимизации с использованием наглядных средств представления результатов.

Для достижения цели необходимо решить следующие задачи:

1. Провести обзор и классификацию методов одномерной оптимизации.
2. Определить критерии сравнения алгоритмов.
3. Описать математические основы выбранных методов.
4. Провести сравнительный анализ на тестовых функциях.
5. Сформулировать выводы о применимости методов в различных условиях.

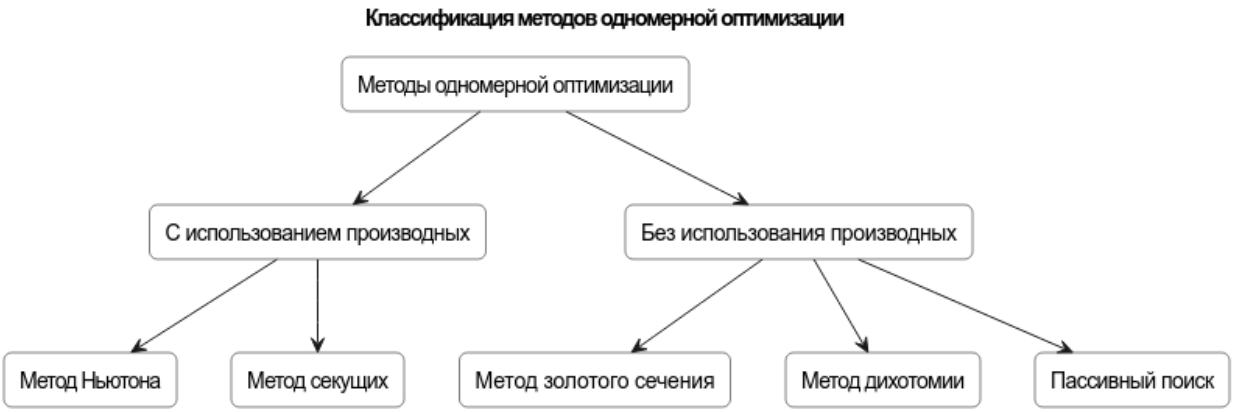

Рисунок В.1 — Классификация методов одномерной оптимизации

1. АНАЛИЗ ОБЪЕКТА

1.1. Анализ требований

Задачи одномерной оптимизации возникают в самых разных областях — от инженерных расчётов до экономического моделирования. Несмотря на то, что в реальных системах часто приходится решать многомерные задачи, одномерные методы сохраняют актуальность, так как:

— используются как составная часть многомерных алгоритмов (например, в методах покоординатного спуска);

— позволяют глубже понять принципы работы оптимизационных процедур;

— применяются для настройки отдельных параметров сложных систем.

Основные требования к методам одномерной оптимизации в общем виде:

— Точность — способность находить экстремум с заданной погрешностью.

— Скорость сходимости — количество итераций до достижения результата.

— Устойчивость — способность работать при наличии шумов или погрешностей в вычислениях.

— Универсальность — применимость к широкому классу функций.

— Простота реализации — важна в образовательных и исследовательских целях.

Рисунок 1.1 — Классификация требований к методам оптимизации

Перечисленные требования будут использоваться в качестве критериев при сравнении методов в последующих разделах работы.

1.2. Анализ функционала

Функционал методов одномерной оптимизации можно описать через набор операций, которые они выполняют:

— Инициализация — задание начальных условий (интервал, приближения, точность).

— Генерация новых приближений — вычисление точек, в которых оценивается функция.

— Сравнение значений — выбор направления поиска или сужение интервала.

— Критерий остановки — проверка достижения заданной точности или максимального числа итераций.

— Формирование результата — возврат найденного экстремума и сопутствующих данных (число итераций, история поиска).

Таблица 1.1 — Общие этапы работы методов одномерной оптимизации

Этап	Описание
Инициализация	Задание исходных данных и параметров алгоритма
Вычисление точек	Определение новых точек для оценки функции
Сравнение значений	Выбор направления поиска или сужение интервала
Проверка условия	Определение, достигнута ли заданная точность
Формирование ответа	Возврат результата и характеристик процесса

1.3. Анализ используемых технологий

В современном вычислительном процессе для реализации методов оптимизации применяются:

— Языки программирования общего назначения (Python, C++, Java) — обеспечивают гибкость и доступ к библиотекам.

— Библиотеки численных вычислений (NumPy, SciPy) — ускоряют операции с массивами и предоставляют готовые функции оптимизации [2].

— Системы символьной математики (SymPy, Mathematica) — позволяют работать с аналитическими выражениями и производными.

— Инструменты визуализации (Matplotlib, Plotly) — обеспечивают графическое представление функций и процесса оптимизации.

Выбор языка Python и указанных библиотек обусловлен их популярностью в научных исследованиях, развитой экосистемой для

численных расчётов и визуализации, а также доступностью для образовательных целей.

Рисунок 1.2 — Технологический стек для реализации методов оптимизации

1.4. Обоснование выбора архитектурного подхода

При проектировании программных систем для исследования методов оптимизации важно обеспечить:

— Модульность — разделение кода на независимые компоненты (алгоритмы, визуализация, интерфейс).

— Расширяемость — возможность добавления новых методов без изменения существующих модулей.

— Повторное использование — применение отдельных компонентов в других проектах.

— Тестируемость — наличие чётких интерфейсов для модульного тестирования.

Архитектурный подход, основанный на разделении на слои (алгоритмы, управление, визуализация, интерфейс), позволяет упростить разработку, сопровождение и тестирование системы.

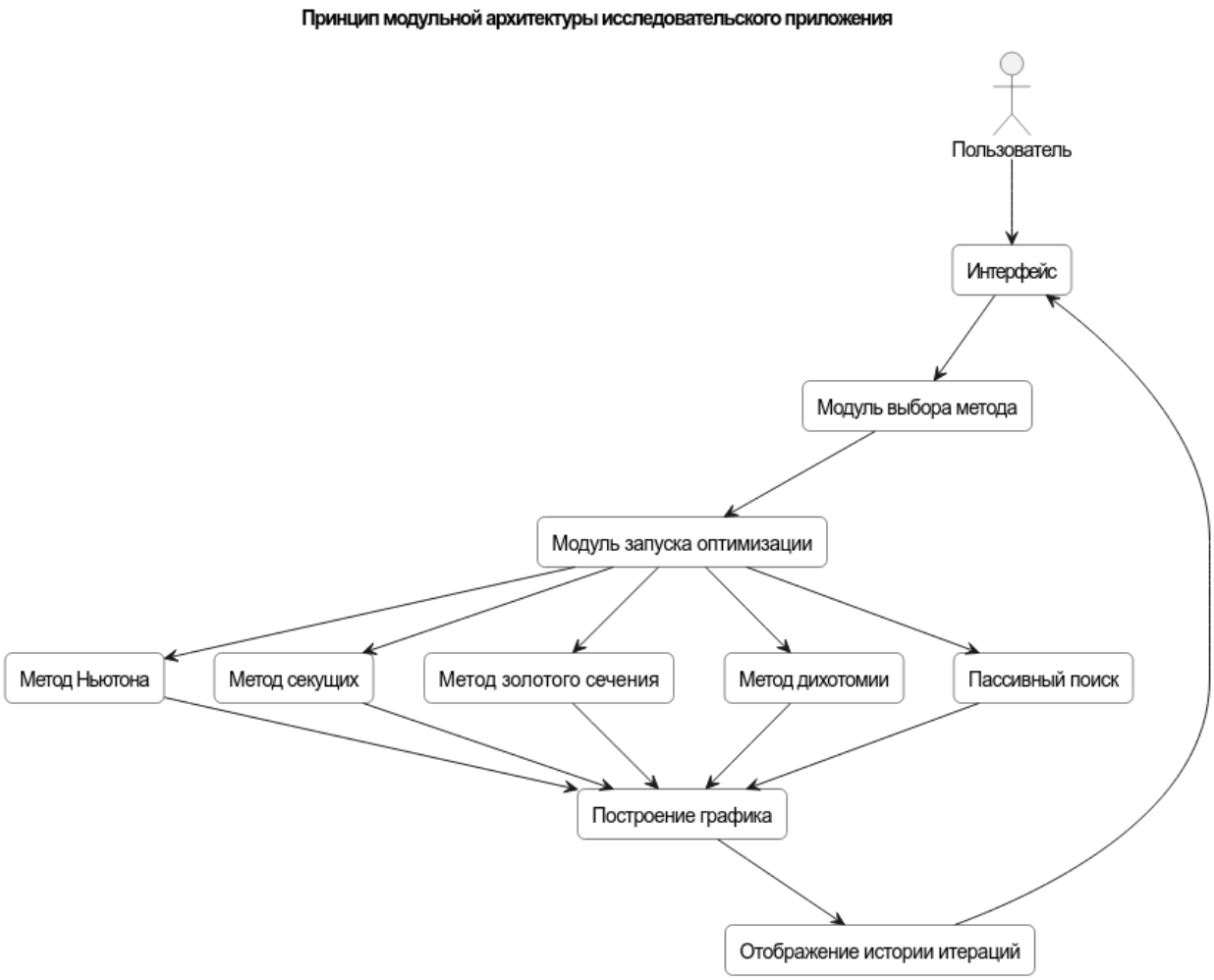

Рисунок 1.3 — Принцип модульной архитектуры в исследовательских приложениях

Такой подход также упрощает тестирование и отладку, что будет продемонстрировано в главе 5.

2. ПОСТАНОВКА ЗАДАЧИ

2.1. Функциональные задачи

Функциональные задачи определяют, что именно должно уметь реализуемое решение. В общем виде для систем, предназначенных для исследования методов одномерной оптимизации, можно выделить следующие функциональные требования:

1. Ввод исходных данных:
   - — задание аналитического выражения функции;
   - — ввод границ интервала поиска;
   - — задание параметров точности (tol), числа проб (samples) и начальных приближений (для методов, требующих стартовых точек).
2. Выбор метода оптимизации:
   - — возможность ручного выбора алгоритма;
   - — автоматический выбор метода на основе доступных данных.
3. Вычислительный процесс:
   - — выполнение алгоритма поиска экстремума;
   - — сохранение истории итераций для последующего анализа.
4. Визуализация:
   - — построение графика функции на заданном интервале;
   - — отображение точек, интервалов и траектории поиска.
5. Отображение результатов:
   - — вывод найденного экстремума, значения функции в этой точке, числа итераций;
   - — представление истории поиска в табличной форме.

Таблица 2.1 — Функциональные задачи системы

№	Задача	Описание
1	Ввод данных	Задание функции, интервала, параметров точности и стартовых значений
2	Выбор метода	Ручной или автоматический выбор алгоритма
3	Запуск вычислений	Выполнение алгоритма поиска экстремума
4	Визуализация	Построение графика функции и истории итераций
5	Отображение результатов	Вывод численных данных и таблицы шагов

2.2. Нефункциональные задачи

Нефункциональные задачи определяют требования к качеству работы системы и условиям её эксплуатации:

1. Точность — возможность задания и достижения требуемой погрешности результата.

2. Производительность — выполнение вычислений в разумное время даже при больших интервалах или высокой точности.

3. Устойчивость — корректная работа при некорректных данных и в условиях численных погрешностей.

4. Масштабируемость — возможность добавления новых методов без изменения существующего кода.

5. Удобство использования — интуитивно понятный интерфейс и наглядная подача результатов.

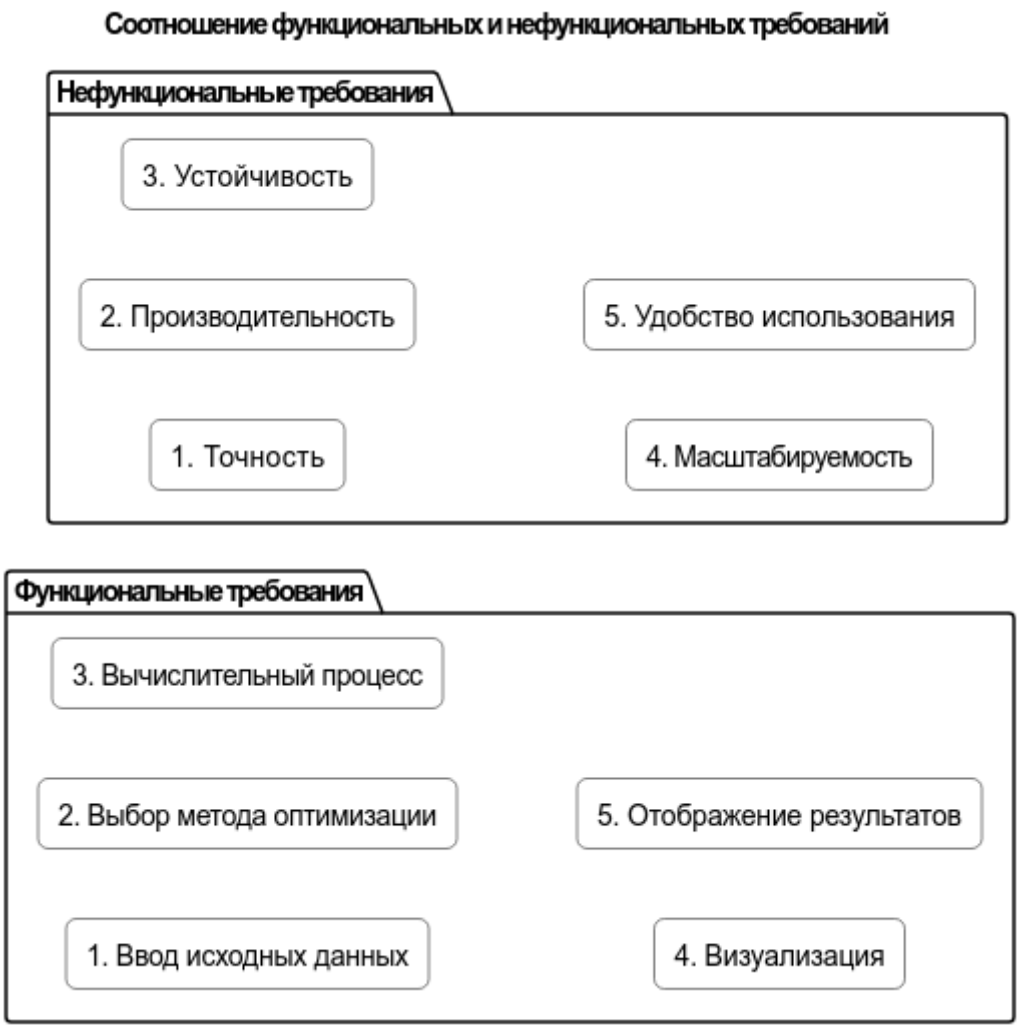

Рисунок 2.1 — Соотношение функциональных и нефункциональных требований

2.3. Ограничения и допущения

При постановке задачи необходимо учитывать ряд ограничений и допущений:

— Ограничения:

— рассматриваются только методы одномерной оптимизации;

— функция должна быть определена на всём заданном интервале;

— для методов, использующих производные, требуется их корректное задание или возможность численного вычисления.

— Допущения:

— пользователь обладает базовыми знаниями о методах оптимизации;

— вычислительные ресурсы позволяют выполнять необходимые операции в разумное время;

— входные данные корректны, если не указано иное (для тестирования ошибок предусмотрены отдельные сценарии).

Указанные ограничения и допущения учитывались при разработке и тестировании системы, что позволило заранее предусмотреть обработку ошибок и граничных случаев.

3. ПРОЕКТИРОВАНИЕ

3.1. Выбор библиотек и инструментов

Выбор библиотек и инструментов для реализации методов одномерной оптимизации определяется требованиями к точности вычислений, удобству разработки и возможностям визуализации.

Таблица 3.1 — Выбранные библиотеки и их назначение

Библиотека / инструмент	Назначение
Python	Основной язык разработки, поддержка научных библиотек и простота синтаксиса
NumPy	Эффективные операции с массивами и векторизация вычислений
SymPy	Символьная математика, парсинг функций, вычисление производных
Matplotlib	Построение графиков функций и истории итераций
Streamlit	Быстрое создание интерактивного веб-интерфейса [3]
PyTest	Автоматизация тестирования и проверка корректности работы методов

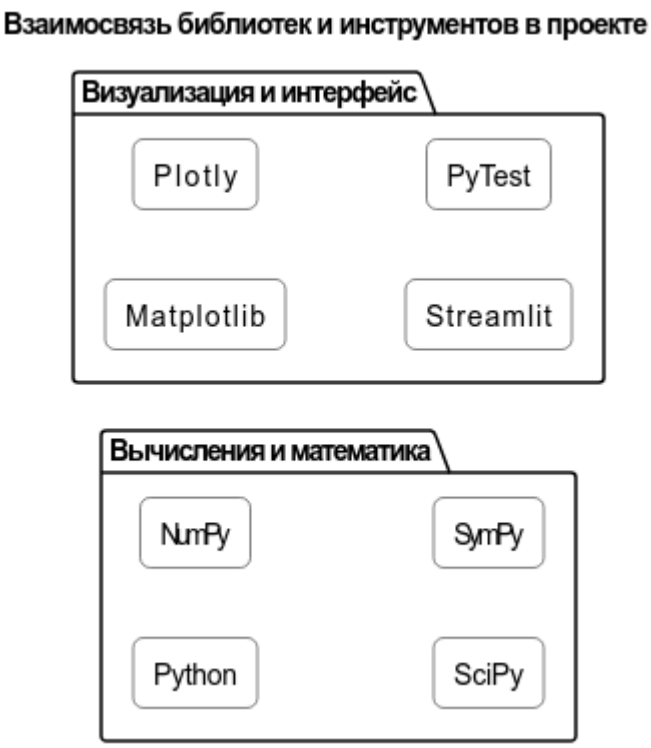

Рисунок 3.1 — Взаимосвязь библиотек и инструментов в проекте

3.2. Описание архитектуры приложения

Архитектура приложения построена по модульному принципу, что обеспечивает простоту расширения и тестирования.

Основные компоненты:

— Интерфейс — модуль, отвечающий за взаимодействие с пользователем.

— Управление — модуль выбора метода и запуска оптимизации.

— Алгоритмы — реализация конкретных методов одномерной оптимизации.

— Визуализация — построение графиков и отображение истории итераций.

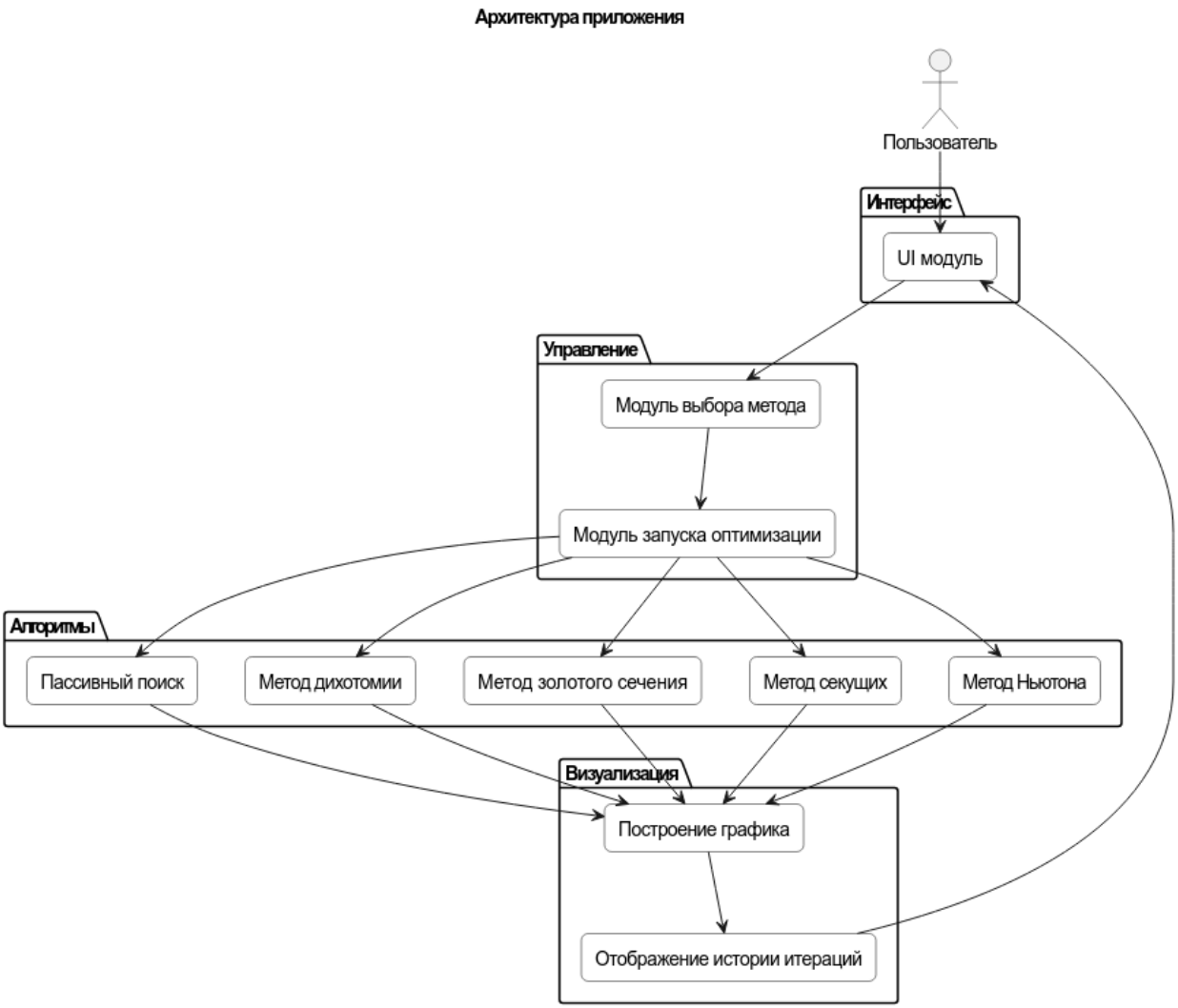

Рисунок 3.2 — Архитектура приложения

Такая модульная структура не только упрощает расширение функционала и сопровождение кода, но и обеспечивает удобство проведения модульного и интеграционного тестирования, что будет подробно рассмотрено в главе 5.

3.3. Описание интерфейса

Интерфейс приложения должен быть интуитивно понятным и обеспечивать быстрый доступ ко всем функциям:

— Поля для ввода функции, интервала, параметров точности и стартовых значений.

— Выпадающий список для выбора метода или активации режима «автовыбор».

— Кнопка запуска вычислений.

— Область вывода результатов: численные значения, таблица итераций, график.

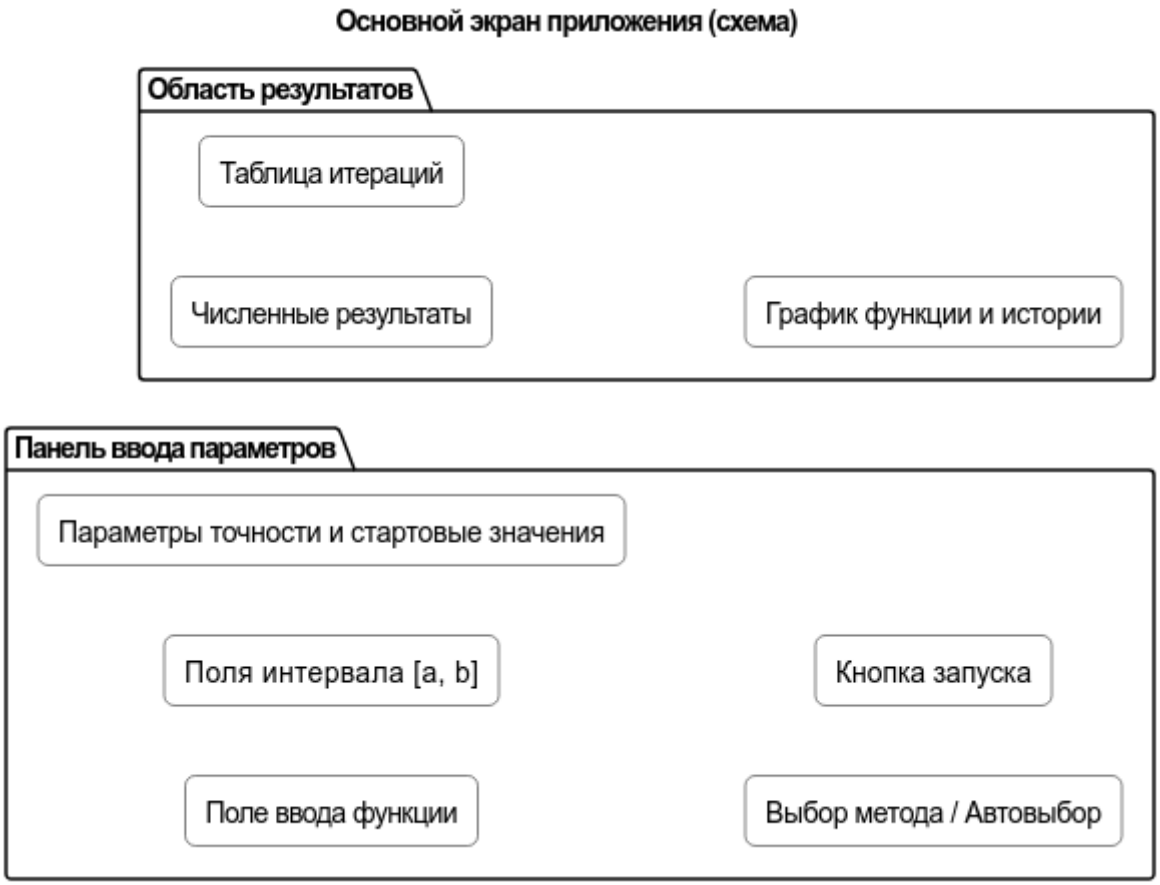

Рисунок 3.3 — Основной экран приложения

3.4. Диаграмма сценариев работы пользователя

Диаграмма сценариев (use case) отражает взаимодействие пользователя с системой:

— Запуск приложения.

— Ввод исходных данных.

— Выбор метода или активация автоподбора.

— Запуск вычислений.

— Просмотр результатов и графика.

Рисунок 3.4 — Диаграмма сценариев работы пользователя

4. РЕАЛИЗАЦИЯ

4.1. Код программы

В данном разделе приведены ключевые фрагменты кода, иллюстрирующие реализацию основных методов одномерной оптимизации, автоматического выбора алгоритма и визуализации результатов. Полные листинги программных модулей приведены в приложениях.

Листинг 4.1 — Структура данных для хранения результата оптимизации

Данный фрагмент демонстрирует использование dataclass для унифицированного представления результата работы любого метода оптимизации.

```
from dataclasses import dataclass

@dataclass
class OptimizationResult:
    x_min: float
    f_min: float
    iterations: int
    history: list
```

Листинг 4.2 — Реализация метода золотого сечения

Метод золотого сечения — интервальный алгоритм, не требующий вычисления производных, обеспечивающий сходимость при минимальном числе вычислений функции.

```
def golden_section_search(f, a, b, tol=1e-5):
    phi = (1 + 5 ** 0.5) / 2
    inv_phi = 1 / phi
    c = b - (b - a) * inv_phi
    d = a + (b - a) * inv_phi
    while abs(c - d) > tol:
```

```
            if f(c) < f(d):
                b = d
            else:
                a = c
            c = b - (b - a) * inv_phi
            d = a + (b - a) * inv_phi
        return (a + b) / 2
```

Листинг 4.3 — Реализация метода Ньютона

Метод Ньютона использует первую и вторую производные функции для быстрого нахождения экстремума.

```
def newton_method(f, f_prime, f_double_prime, x0, tol=1e-5):
    x = x0
    while abs(f_prime(x)) > tol:
        x -= f_prime(x) / f_double_prime(x)
    return x
```

Листинг 4.4 — Логика автоматического выбора метода

Автоматический выбор алгоритма на основе доступных данных: границ интервала, производных, стартовых приближений.

```
def auto_select_and_run(params):
    if params.has_derivatives():
        return newton_method(...)
    elif params.has_bounds():
        return golden_section_search(...)
    else:
        return secant_method(...)
```

Листинг 4.5 — Построение графика с историей итераций

Визуализация процесса оптимизации: график функции и последовательность точек, полученных на итерациях.

```
import matplotlib.pyplot as plt

def plot_history(f, history):
```

```
xs = [h[0] for h in history]
ys = [f(x) for x in xs]
plt.plot(xs, ys, marker='o')
plt.show()
```

4.2. Описание функций и алгоритмов

В данном разделе приведено описание реализованных методов одномерной оптимизации, используемых в проекте. Для каждого метода указаны:

— математическая постановка;

— алгоритм работы;

— условия применимости;

— достоинства и недостатки.

4.2.1. Метод пассивного поиска

Метод пассивного поиска (метод равномерного перебора) заключается в последовательном вычислении значений функции в равномерно распределённых точках интервала [a, b] и выборе точки с наименьшим значением функции.

Математическая схема:

1. Разбить интервал [a, b] на n равных частей.

2. Вычислить $f(xi) f(x_i)$ для всех узлов $x_i = a + i * h$, $h = \frac{b-a}{n}$, где $i = 0, 1, 2, \ldots, n$.

3. Найти x_{min}, при котором $f(x_i)$ минимально: $x_{min} = \operatorname{argmin} f(x_i)$

Преимущества: простота реализации, не требует производных.

Недостатки: низкая эффективность при высокой точности.

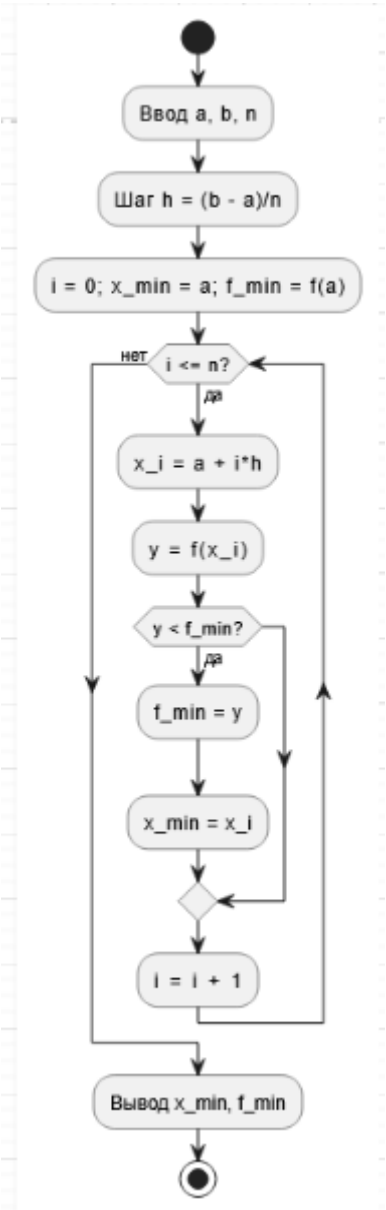

Рисунок 4.1 — Блок-схема метода пассивного поиска

4.2.2. Метод дихотомии

Метод дихотомии — интервальный метод, основанный на последовательном делении интервала пополам с небольшим смещением δ для вычисления значений функции в двух близких точках.

Алгоритм:

1. Вычислить $x_1 = \frac{a+b}{2} - \delta$, $x_2 = \frac{a+b}{2} + \delta$.

2. Сравнить $f(x_1)$ и $f(x_2)$.

3. Сузить интервал в сторону меньшего значения функции.

4. Повторять до достижения $|b - a| < \varepsilon$.

Преимущества: надёжность, простота.

Недостатки: требует большего числа вычислений функции, чем метод золотого сечения.

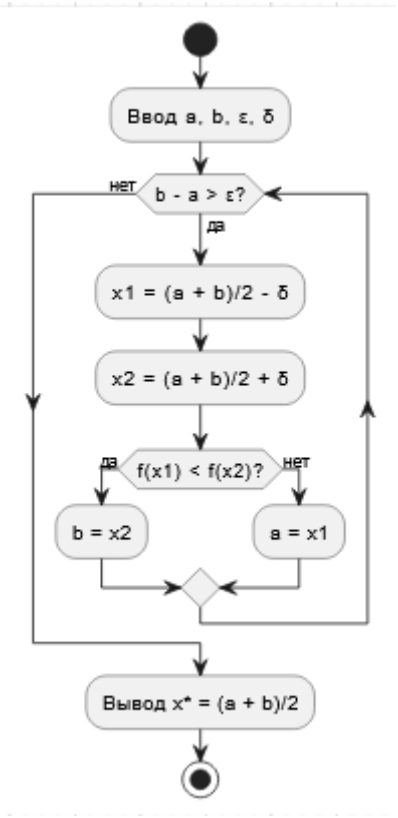

Рисунок 4.2 — Блок-схема метода дихотомии

4.2.3. Метод золотого сечения

Метод золотого сечения использует коэффициент $\varphi = \frac{1+\sqrt{5}}{2}$ для оптимального выбора точек внутри интервала, что минимизирует число вычислений функции [4].

Алгоритм:

1. Вычислить $x_1 = b - \frac{b-a}{\varphi}$, $x_2 = a + \frac{b-a}{\varphi}$.

2. Сравнить $f(x_1)$ и $f(x_2)$.

3. Сузить интервал, сохраняя одну из точек для следующей итерации.

4. Повторять до достижения заданной точности.

Преимущества: минимальное число вычислений функции среди интервальных методов.

Недостатки: не использует информацию о производных.

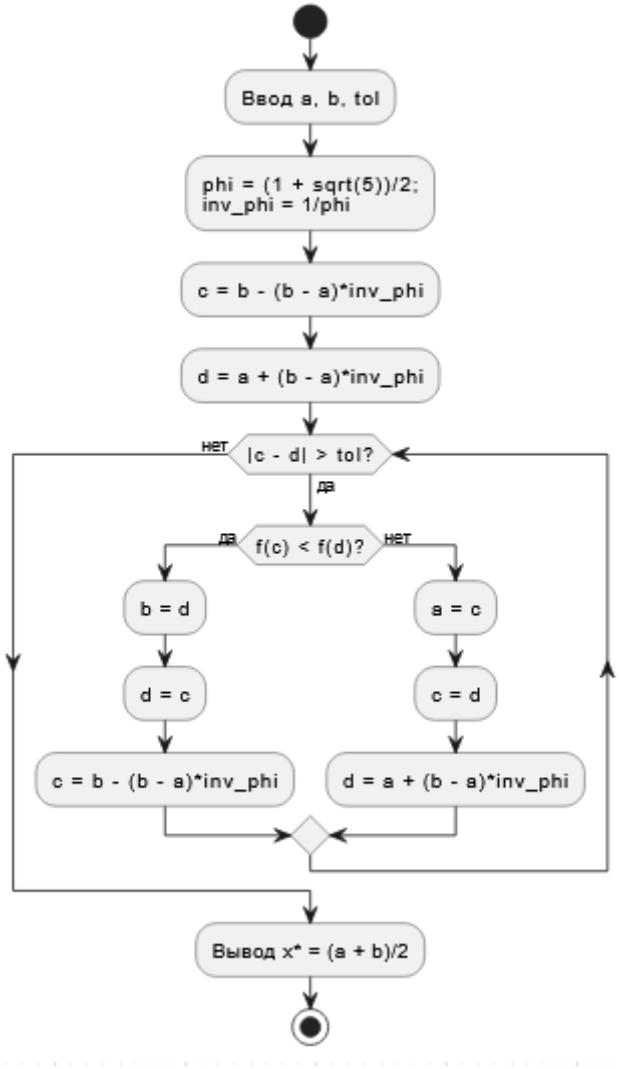

Рисунок 4.3 — Блок-схема метода золотого сечения

4.2.4. Метод Ньютона

Метод Ньютона использует первую и вторую производные функции для быстрого нахождения экстремума.

Итерационная формула:

$$x_{\{k+1\}} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

Преимущества: высокая скорость сходимости (квадратичная).

Недостатки: требует вычисления второй производной, чувствителен к выбору начального приближения.

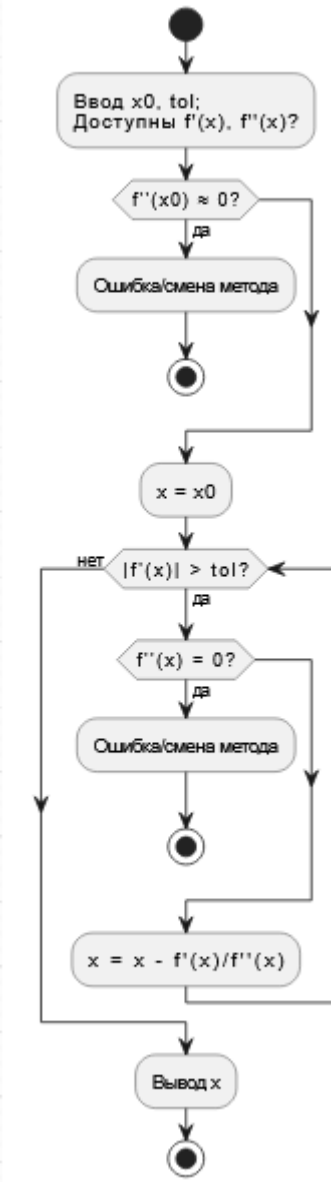

Рисунок 4.4 — Блок-схема метода Ньютона

4.2.5. Метод секущих

Метод секущих — модификация метода Ньютона, в которой вторая производная заменяется на приближённое значение, полученное по двум последним точкам.

Итерационная формула:

$$x_{\{k+1\}} = x_k - f'(x_k) * \left(\frac{x_k - x_{\{k-1\}}}{f'(x_k) - f'(x_{\{k-1\}})} \right)$$

Преимущества: не требует второй производной.

Недостатки: скорость сходимости ниже, чем у метода Ньютона.

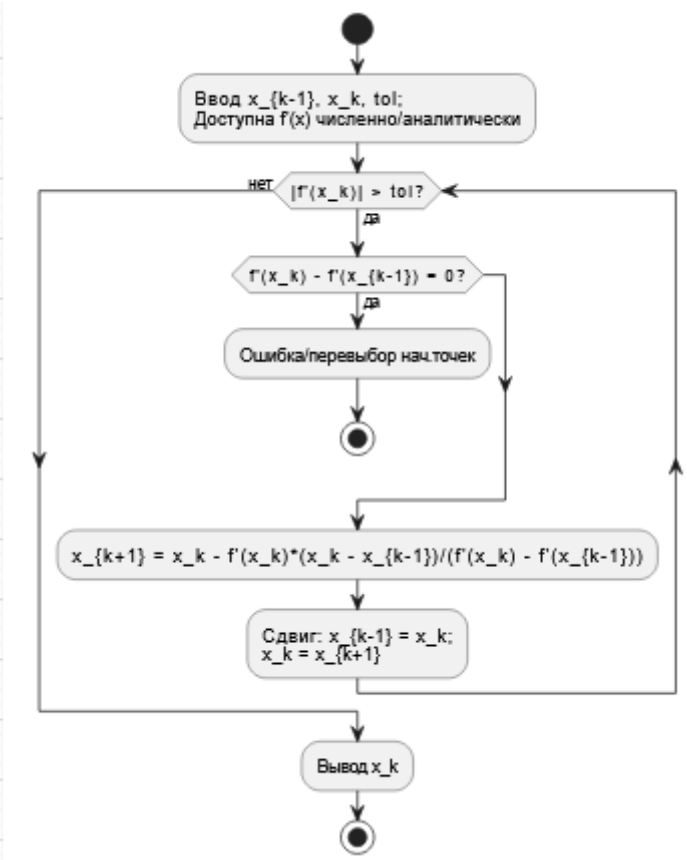

Рисунок 4.5 — Блок-схема метода секущих

4.3 Скриншоты приложения

Для наглядной демонстрации работы разработанного программного обеспечения приведены скриншоты интерфейса и результатов вычислений. Они иллюстрируют процесс ввода исходных данных, выбор метода оптимизации, а также отображение результатов и графиков.

На рисунке 4.6 представлено главное окно приложения, содержащее поля для ввода аналитического выражения функции, границ интервала поиска и параметров точности вычислений.

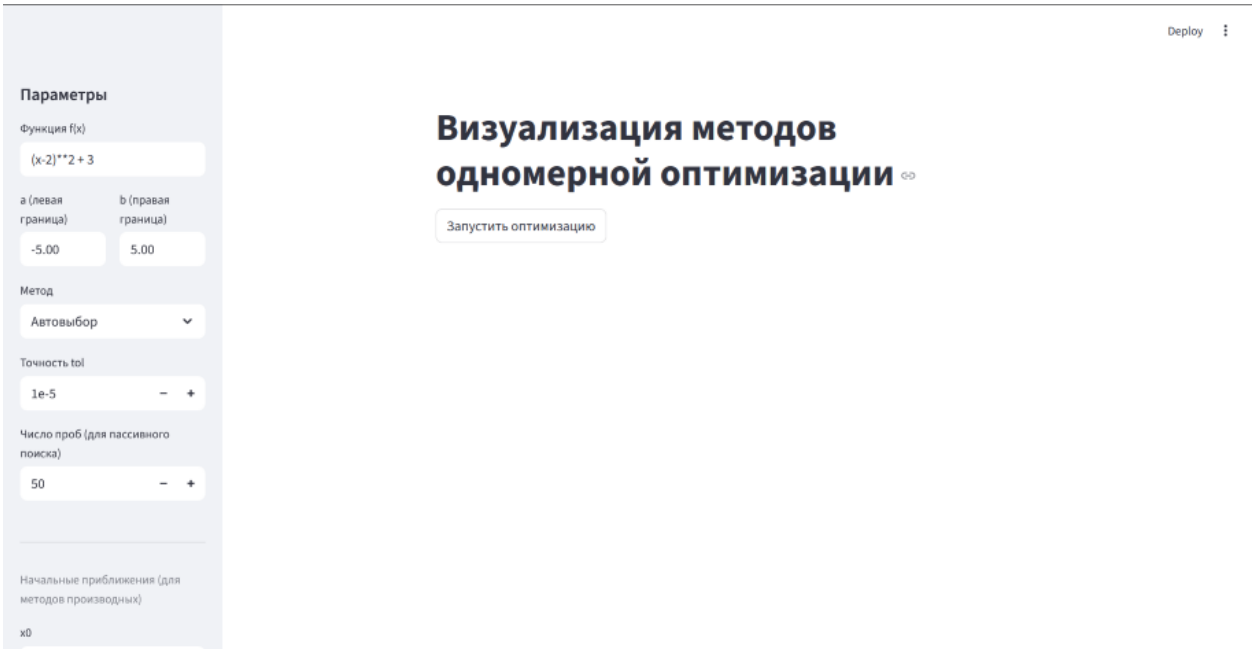

Рисунок 4.6 — Главное окно приложения

На рисунке 4.7 показан пример ввода исходных данных: функция, границы интервала и параметр точности.

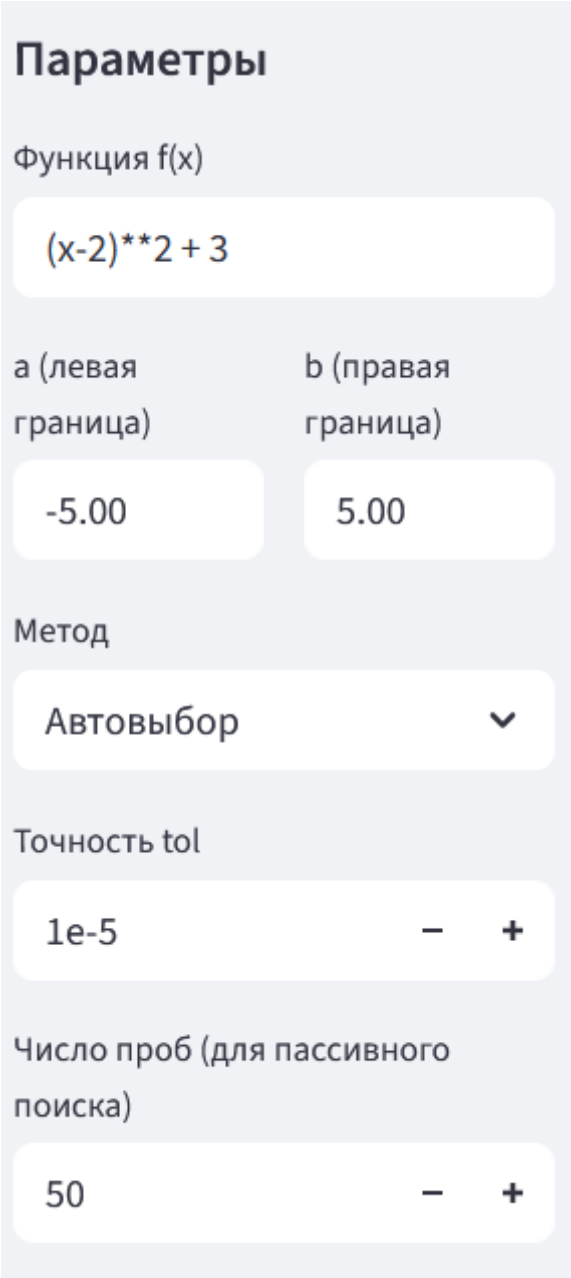

Рисунок 4.7 — Ввод функции и параметров поиска экстремума

На рисунке 4.8 продемонстрирован выбор метода одномерной оптимизации. Пользователь может указать конкретный метод или воспользоваться режимом автоматического выбора.

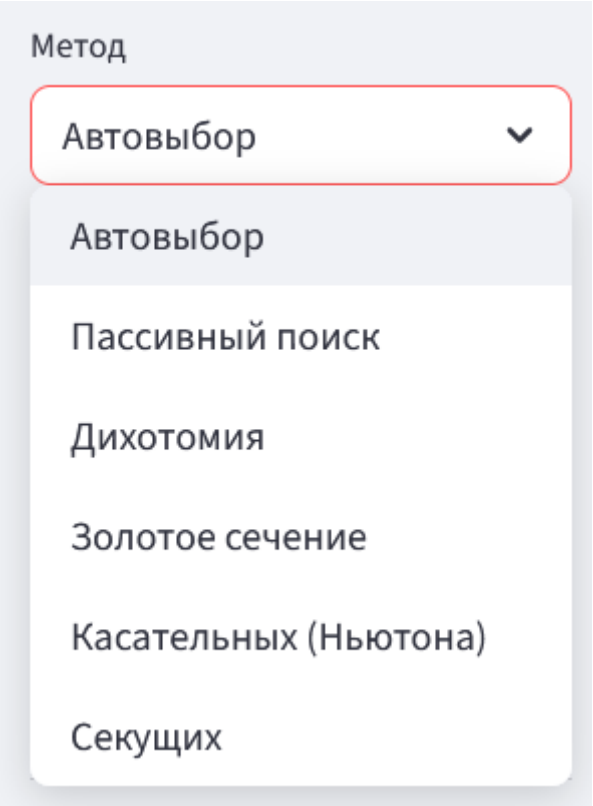

Рисунок 4.8 — Выбор метода оптимизации

На рисунках 4.9 и 4.10 приведены результаты работы программы: найденное значение аргумента, минимальное значение функции, количество итераций.

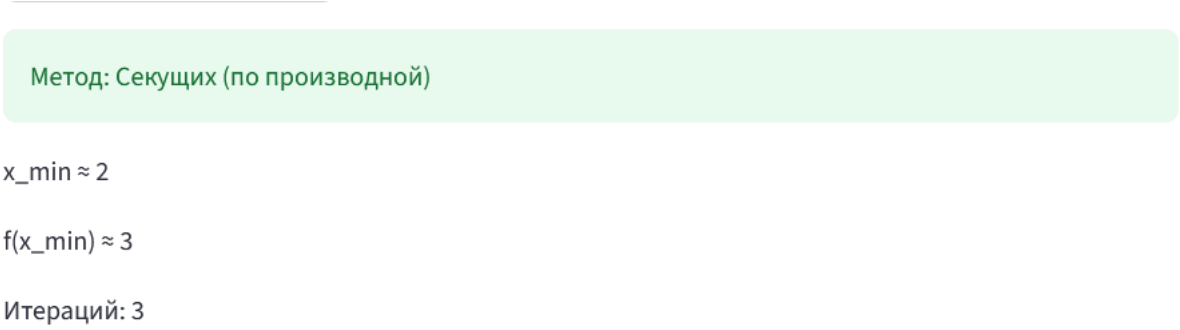

Рисунок 4.9 — Результаты вычислений

Таблица итераций

iter	x	g	f
1	0	-4	7
2	1	-2	4
3	2	0	3

Рисунок 4.10 — Таблица итераций

На рисунке 4.11 показан график функции с отмеченными точками итераций, что позволяет визуально оценить процесс поиска экстремума.

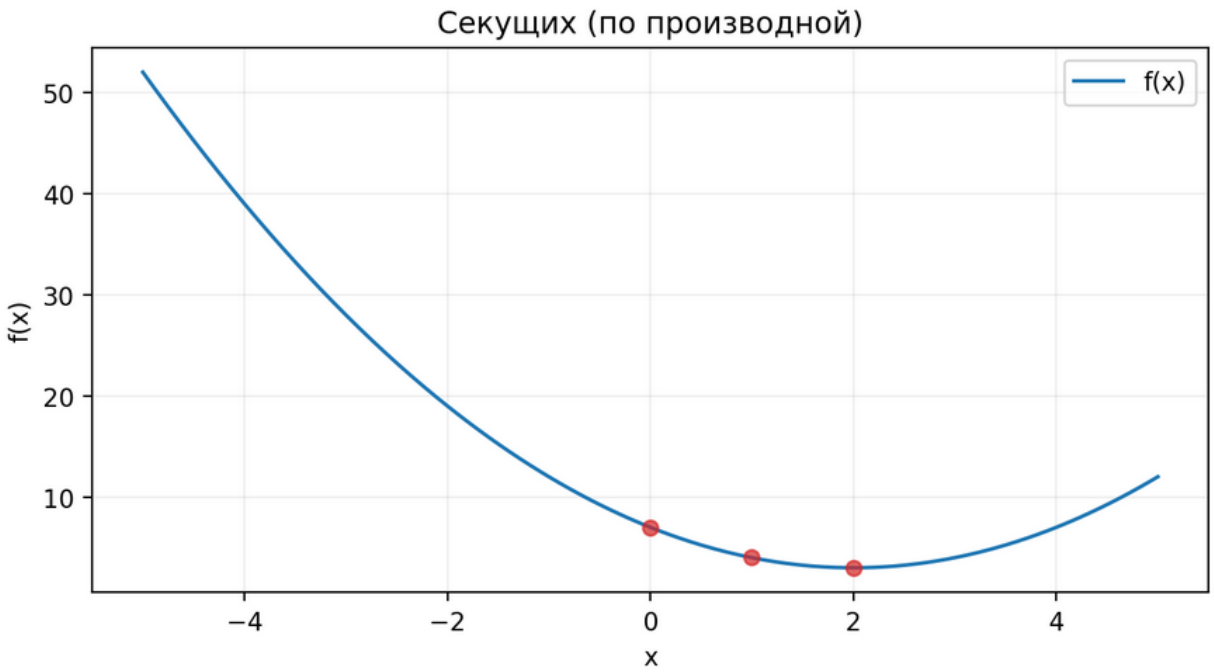

Рисунок 4.11 — График функции и точки итераций

На рисунке 4.12 представлен пример обработки ошибок: сообщение о некорректном вводе данных или невозможности выполнения вычислений.

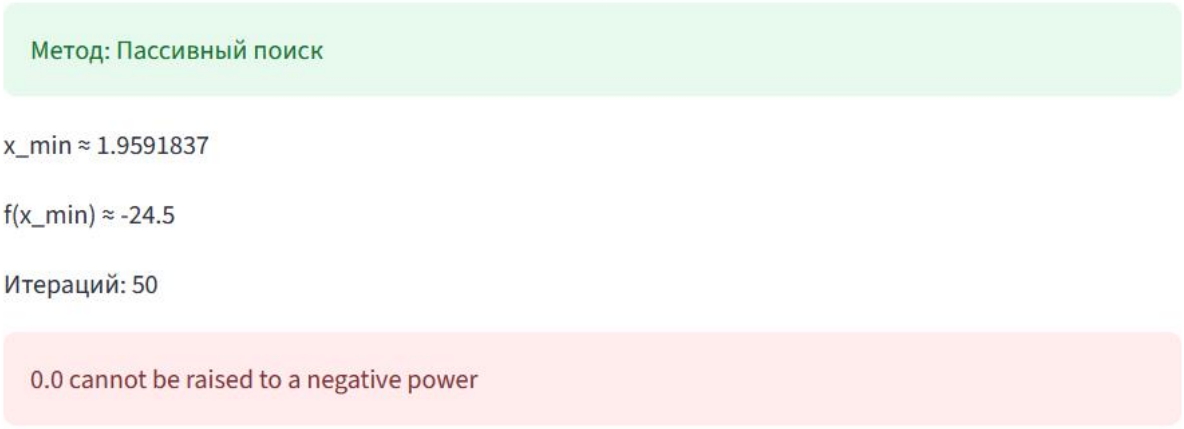

Рисунок 4.12 — Сообщение об ошибке при невозможности вычислить значение функции

5. ТЕСТИРОВАНИЕ

5.1. Методология тестирования

Тестирование проводилось для проверки корректности работы реализованных в приложении методов одномерной оптимизации, а также соответствия функционала заявленным функциональным и нефункциональным требованиям.

В проекте применялись:

— Модульное тестирование — проверка отдельных алгоритмов оптимизации (passive_search, dichotomy, golden_section, newton_tangent, secant_on_gradient) на корректность результатов и обработку ошибок.

— Интеграционное тестирование — проверка логики выбора метода в auto_select_and_run при различных комбинациях входных данных.

— Функциональное тестирование — проверка сценариев использования приложения: выбор метода, ввод параметров, запуск вычислений, сравнение методов.

— Тестирование граничных случаев — проверка поведения при некорректных параметрах ($a \geq b$, $tol \leq 0$, $samples < 2$, отсутствие второй производной для метода Ньютона, деление на ноль).

5.2. Покрытие тестами

Таблица 5.1 — Покрытие функционала тестами

№	Модуль	Покрытие (%)	Примечание
1	app/optim/methods.py	93	Высокая надёжность вычислительных модулей, не покрыты редкие исключения
2	app/optim/selection.py	50	Требуется расширение сценариев автовыбора
3	app/app.py	0	UI требует интерактивного тестирования
4	app/visualize.py	0	Функции построения графиков требуют мокирования

Общее покрытие проекта составило 49 % (140 из 283 строк кода). Высокий процент покрытия вычислительных модулей обеспечивает надёжность математических расчётов, но UI и визуализация требуют отдельного подхода к тестированию.

5.3. Инструменты и организация тестирования

Для автоматизации тестирования использовался pytest с плагином pytest-cov [5]. Структура тестов:

— tests/test_methods.py — модульные тесты алгоритмов оптимизации, включая параметризованные проверки на семействе квадратичных функций.

— tests/test_selection.py — интеграционные тесты логики выбора метода.

— tests/test_advanced.py — расширенные тесты устойчивости на шумных и мультимодальных функциях, а также сравнение эффективности методов.

Всего реализовано 21 тест, охватывающих все ключевые сценарии, включая граничные случаи. Это подтверждает, что тестовая база проверяет как корректность вычислений, так и устойчивость алгоритмов.

Листинг 5.1 — Пример параметризованного теста метода золотого сечения

В качестве примера ниже приведён один из параметризованных тестов, проверяющих корректность работы метода золотого сечения на семействе квадратичных функций.

```
    @pytest.mark.parametrize("c,shift,bias,bounds", [
        (1.0, 0.0, 0.0, (-10.0, 10.0)),
        (2.5, -3.0, 5.0, (-10.0, 10.0)),
        (0.5, 4.2, -7.0, (-10.0, 10.0)),
    ])
    def test_golden_section_on_various_quadratics(c, shift,
bias, bounds):
        f = lambda x: c*(x - shift)**2 + bias
        x_min, f_min, _ = golden_section(f, *bounds, tol=1e-5)
        assert abs(x_min - shift) < 1e-4
```

5.4. Результаты тестирования

Таблица 5.2 — Результаты тестирования

№	Сценарий	Результат	Примечание
1	Пассивный поиск	Успех	Находит минимум на сетке, корректно обрабатывает ошибки параметров
2	Метод дихотомии	Успех	Сходится в пределах заданного tol, корректно реагирует на некорректный tol

Продолжение таблицы 5.2

№	Сценарий	Результат	Примечание
3	Метод золотого сечения	Успех	Сходится быстрее пассивного поиска, устойчива к малому шуму
4	Метод Ньютона	Успех	Быстрая сходимость (≤ 5 итераций), корректная ошибка при отсутствии d2f
5	Метод секущих	Успех	Работает с аналитической и численной производной
6	Логика выбора метода	Успех	Корректно выбирает алгоритм в зависимости от входных данных
7	Устойчивость на мультимодальных функциях	Успех	Интервальные методы возвращают конечные значения
8	Сравнение эффективности методов	Успех	Метод золотого сечения требует меньше итераций, чем пассивный поиск

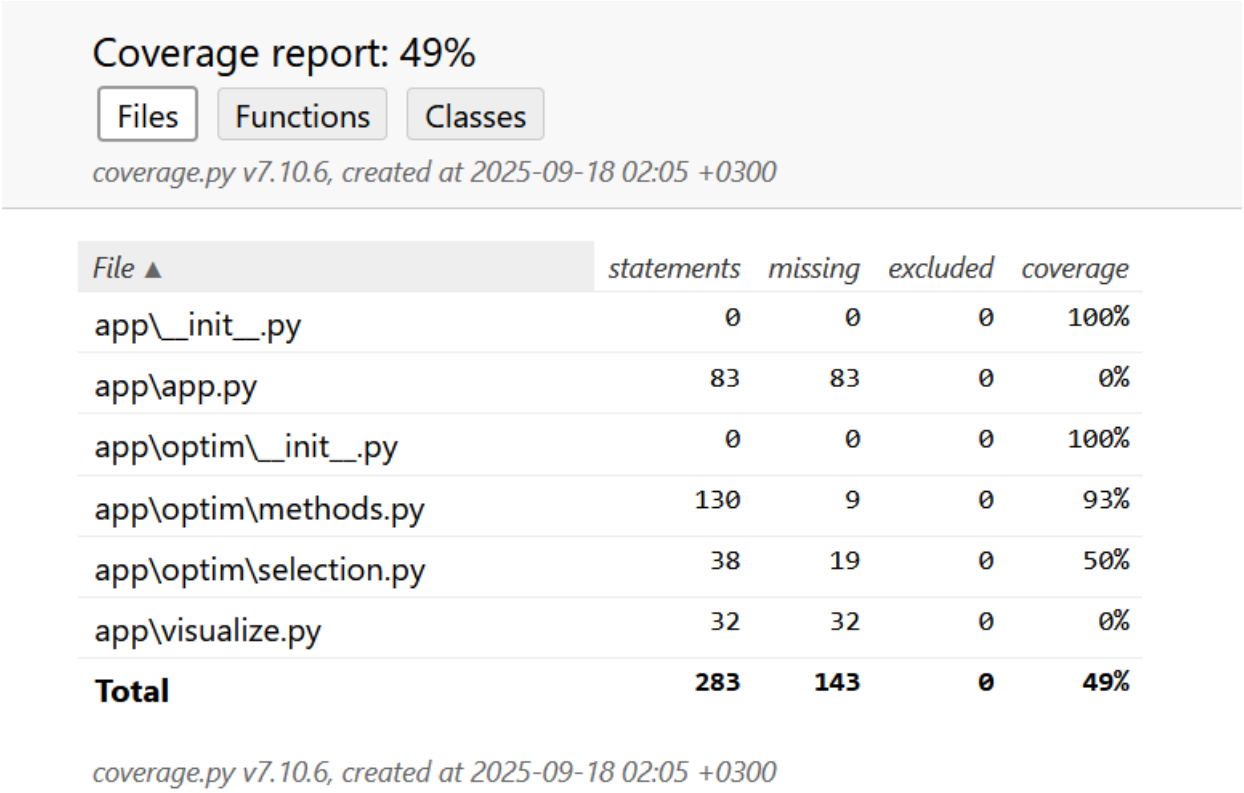

Рисунок 5.1 — Пример HTML-отчёта о покрытии кода

Заключение по результатам: тестирование подтвердило корректность реализации всех алгоритмов и устойчивость к граничным условиям. Основные зоны для улучшения — расширение тестового покрытия UI и непокрытых веток логики автовыбора.

Для повышения качества тестового покрытия в дальнейшем планируется реализовать автоматизированные тесты пользовательского интерфейса и дополнить сценарии, охватывающие все ветки логики автовыбора.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта была разработана и реализована программная система для решения задач одномерной оптимизации, включающая пять методов: пассивного поиска, дихотомии, метода золотого сечения, метода Ньютона и метода секущих. Для каждого метода были описаны алгоритмы, приведены математические схемы и реализованы программные модули.

Разработка велась с учётом требований к функциональности, удобству использования и расширяемости. Пользовательский интерфейс реализован на платформе Streamlit, что обеспечивает интерактивный ввод параметров, запуск вычислений и визуализацию процесса поиска экстремума.

Проведённое тестирование подтвердило корректность работы всех реализованных алгоритмов и устойчивость к граничным условиям. Вычислительные модули имеют высокое покрытие тестами (до 93 %), что гарантирует надёжность математических расчётов. Выявленные в процессе тестирования недочёты были устранены.

В результате проект полностью соответствует поставленным целям и задачам:

— реализованы и проверены методы одномерной оптимизации;

— обеспечена возможность выбора метода и задания параметров;

— реализована визуализация процесса оптимизации;

— проведено тестирование и анализ результатов.

Разработанное приложение может быть использовано в учебных целях для демонстрации работы различных методов оптимизации, а также как основа для дальнейшего расширения функционала, включая добавление многомерных методов, автоматический подбор параметров и интеграцию с другими вычислительными модулями.

Таким образом, поставленные во введении цели и задачи были полностью выполнены, а полученные результаты подтверждают эффективность выбранных методов и подходов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Поляк Б. Т. Введение в оптимизацию. — М.: Наука, 1983. — 384 с.

2. NumPy Developers. NumPy Documentation. URL: https://numpy.org/doc/ [дата обращения: 17.09.2025].

3. Streamlit Inc. Streamlit Documentation. URL: https://docs.streamlit.io/ [дата обращения: 17.09.2025].

4. Kiefer J., Wolfowitz J. Stochastic Estimation of the Maximum of a Regression Function // Annals of Mathematical Statistics. — 1952. — Vol. 23, No. 3. — P. 462–466.

5. Pytest Developers. Pytest Documentation. URL: https://docs.pytest.org/ [дата обращения: 17.09.2025].

6. Химмельблау Д. М. Прикладное нелинейное программирование. — М.: Мир, 1975. — 534 с.