Оптимизация конфигурации счетчика нейтронных совпадений

Никифоров Д.О.

Отчет, представленный в рамках НИРС
кафедра №5, ИЯФиТ
НИЯУ МИФИ
Москва, Россия

Декабрь 2025

Руководитель Пугачев П.А.

Аннотация

Рассматривается задача оптимизации конфигурации счётчика нейтронных совпадений, моделируемого в транспортном коде Serpent. Модель счётчика представлена как задача оптимизации типа «чёрного ящика», где значение целевой функции определяется результатами моделирования методом Монте-Карло. Для решения задачи исследуется эффективность трёх подходов: алгоритма деревьев Парзена (TPE), байесовской оптимизации (BO) и активного обучения. Проведено сравнение методов по скорости сходимости, устойчивости и качеству найденных решений при ограниченном числе вычислительных экспериментов. Результаты показывают, что активное обучение обеспечивает наилучший баланс между точностью и вычислительными затратами, а байесовская оптимизация демонстрирует высокую стабильность в условиях сложного параметрического пространства. Полученные данные могут быть использованы для автоматизации проектирования нейтронных детекторов и систем регистрации.

Оглавление

Список таблиц

Список иллюстраций

Листинги

Глава 1

Введение

1.1 Актуальность и текущее состояние

На определенных этапах технологического процесса изготовления ТВС переход к следующему циклу в силу требований ядерной и радиационной безопасности невозможен без оценки количества делящихся материалов в изделии неразрушающими методами.

Поэтому разработка новых и оценка применимости существующих методов неразрушающего контроля содержания делящихся материалов в топливе ВТГР реакторов является актуальной задачей.

Разрабатываемая измерительная система состоит из ^{3}He счетчиков в полиэтиленовом замедлителе, источника нейтронов и анализатора импульсов.

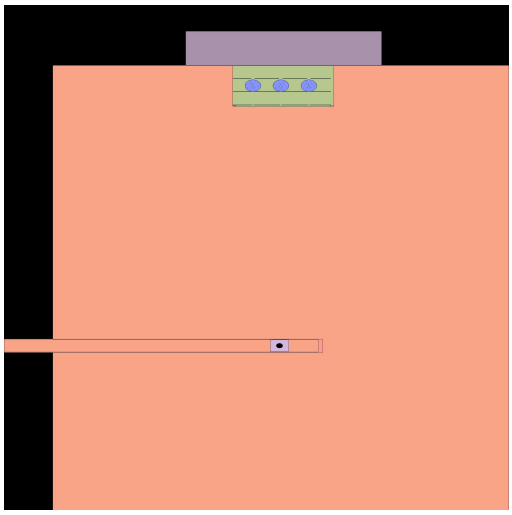


Рис. 1.1: Изображение установки, полученное в Serpent.

Идея установки состоит в следующем: нейтроны, испускаемые источником замедляются до тепловых, взаимодействуют с топливом. Так как первичные нейтроны – тепловые, а вторичные – быстрые, то через кадмиевую пластину в основном проходят только вторичные, взаимодействуют с гелиевыми счетчиками, позволяя снимать измерения.

1.2 Постановка задачи

Задача заключается в подборе оптимальной конфигурации установки, при которой относительная ошибка счета будет минимальной:

$$\mathcal{F} = \frac{\sqrt{N_{\text{ф}} t_{\text{ф}} + N_{tot} t_{tot}}}{N_{tot} t_{tot} - N_{\text{ф}} t_{\text{ф}}}$$

$\text{t}_{\text{ф}} = \text{t}_{tot} = 3900$ sec
$\text{N}_{\text{ф}}$ - скорость счета без топлива
N_{tot} - скорость счета с топливом

Следует отметить, что значение целевой функции $\mathcal{F}$ в рамках нейтронно-физического моделирования методом Монте-Карло не является детерминированной величиной. Для фиксированного набора параметров $x \in \mathcal{X}$ наблюдаемое значение функционала представляет собой случайную величину

$$\widehat{\mathcal{F}(x)} = \mathcal{F}(x) + \epsilon(x),$$

где $\mathcal{F}(x) = \mathbb{E}[\widehat{\mathcal{F}(x)}]$ – математическое ожидание, а $\epsilon(x)$ – статистическая погрешность, обусловленная конечным числом моделируемых нейтронов. Дисперсия шума $\epsilon(x)$ зависит от выбранной точности расчёта и, в общем случае, является функцией точки параметрического пространства, что соответствует модели гетероскедастического шума.

Таким образом, задача оптимизации сводится не к поиску экстремума детерминированной функции, а к поиску минимума математического ожидания стохастического функционала при ограниченном вычислительном бюджете и неоднородной точности наблюдений. Данное обстоятельство существенно ограничивает применимость классических методов оптимизации и требует использования алгоритмов, способных учитывать статистическую природу отклика и стоимость получения одного наблюдения.

Первоначально поиск оптимальной конфигурации счетчика нейтронов проходил поиском по сетке с равным шагом. Так как выполнение нейтронно-физических

расчетов с использованием Serpent является вычислительно дорогой задачей (~ 2 часа для получения значения с точностью $\sim 3\%$) полное исследование системы с целью нахождения точки максимума целевой функции становится трудновыполнимой задачей.

Время полного расчета увеличивается экспоненциально с ростом числа параметров и квадратично с устанавливаемой точностью:

$$T = \mathcal{O}(k^P \Delta^2) \tag{1.1}$$

где:

- P – размерность пространства параметров
- Δ – точность целевой функции
- $A, k \in \Re$

То есть при числе параметров более 2 использование поиска по сетке или случайного поиска не представляется возможным.

«На фундаментальном уровне перенос нейтронов через вещество описывается по своей сути как стохастический процесс. Полное сечение представляет собой вероятность (на единицу длины пути и единицу плотности атомов), но не уверенность того, что нейтрон испытает столкновение при прохождении определенного слоя пространства. Если нейтрон действительно сталкивается, сечения для различных процессов представляют собой вероятности, но не уверенность того, что столкновение окажется актом рассеяния, радиационного захвата, деления и так далее. Нейтронный поток на самом деле является средним значением или математическим ожиданием функции распределения нейтронов. Метод Монте-Карло напрямую моделирует перенос нейтронов как стохастический процесс.» Stacey 2007

Из-за природы процесса и способа расчета не представляется возможным получить промежуточные результаты или получить целевую функцию аналитически. Это позволяет охарактеризовать данный процесс как расчет по схеме "черного ящика".

С учетом данной информации можно изменить подход в поиске максимума целевой функции и решать задачу оптимизации черного ящика.

1.3 Цель работы, задачи и научная новизна

Целью данной работы является разработка и исследование методов оптимизации параметров нейтронно-физических моделей, представляемых в виде вычислительно дорогого стохастического «чёрного ящика», с учётом статистической погрешности Монте-Карло расчётов.

Для достижения поставленной цели в работе решаются следующие задачи:

- анализ применимости методов байесовской оптимизации к нейтронно-физическим расчётам;
- разработка многостадийной схемы оптимизации с адаптивной точностью вычислений;
- реализация гибридного метода на основе суррогатных моделей и активного обучения;
- сравнительный анализ эффективности различных оптимизационных стратегий при фиксированном вычислительном бюджете.

Научная новизна работы заключается в следующем:

- предложена схема байесовской оптимизации с адаптивным управлением точностью Монте-Карло расчётов;
- разработан универсальный фреймворк оптимизации нейтронно-физических моделей типа «чёрный ящик»;
- проведено систематическое сравнение методов BO, TPE и гибридных подходов в условиях шумной целевой функции;
- показано, что устойчивые ансамблевые суррогатные модели обеспечивают более быстрое схождение при фиксированном вычислительном бюджете.

Глава 2

Методы

2.1 Особенности оптимизации стохастических физических моделей

В отличие от типичных задач оптимизации гиперпараметров, в которых значение целевой функции может быть получено с высокой точностью при фиксированной вычислительной стоимости, в нейтронно-физических расчётах на основе метода Монте-Карло значение целевой функции является случайной величиной.

Дисперсия оценки функционала определяется числом моделируемых нейтронов и убывает пропорционально 1/N, что приводит к необходимости учитывать статистическую погрешность при оптимизации.

В данной работе оптимизация рассматривается в условиях гетероскедастического шума и ограниченного вычислительного бюджета, что накладывает ограничения на прямое применение классических методов байесовской оптимизации.

В отличие от классических задач байесовской оптимизации, в которых предполагается либо детерминированный отклик, либо аддитивный гомоскедастический шум, в нейтронно-физических расчётах методом Монте-Карло наблюдается принципиально иная ситуация. Точность оценки целевой функции напрямую зависит от вычислительных затрат и может целенаправленно изменяться в процессе оптимизации.

Это приводит к необходимости рассматривать оптимизацию в условиях управляемой гетероскедастичности, где алгоритм должен одновременно принимать решение о выборе следующей точки параметрического пространства и о допустимой точности вычисления значения функционала в данной точке. Стандартные реализации

байесовской оптимизации, как правило, не учитывают данную особенность, что снижает их эффективность при фиксированном вычислительном бюджете и мотивирует разработку адаптивных многостадийных стратегий.

2.2 Существующие подходы

Наиболее популярными методами оптимизации являются фреймворки, основанные на байесовской оптимизации Gardner и др. 2023. Этот подход позволяет как исследовать неизвестную функцию с минимумом запросов значений черного ящика, так и находить оптимумы функции при наименьших затратах.

Разновидностью BO является TPEWatanabe 2023. Данный метод позволяет решать задачи с категориальными переменными. На данный момент это один из наиболее популярных методов, применяемых для подбора гиперпараметров моделей машинного обучения.

Также для данной задачи применима парадигма машинного обучения, называемая активным обучением. По существу мы согласно BO моделируем суррогатную модель, предсказывающую наиболее перспективные точки в пространстве параметров, которые будем передавать далее в Serpent.

2.3 Байесовская оптимизация с гауссовским процессом (BO with GP)

BO — это итерационный метод, позволяющий оценить оптимум функции, не дифференцируя её. Он имеет две основные компоненты: Elistratova б.г.

- вероятностная модель, которая приближает распределение значений целевой функции в зависимости от имеющихся исторических данных (часто в качестве такой модели выбирают гауссовские процессы);
- функция, позволяющая по некоторым статистикам текущей вероятностной модели функции $\mathcal{F}$ указать, в какой следующей точке нужно вычислить значение $\mathcal{F}$. Эта функция называется acquisition function.

Простой пример acquisition function — сумма среднего вероятностной модели и стандартного отклонения с некоторым весом:

$$\alpha(x) = \mu(x) + \beta\sigma(x)$$

Алгоритм процесса оптимизации выглядит следующим образом:

- На итерации t + 1 вычисляется точка x_{t+1}, в которой нужно провести следующее вычисление целевой функции:

$$x_{t+1} = \underset{x \in X}{argmax}\ \alpha(x|S_t)$$

- Вычисляется значение $\mathcal{F}(x_{t+1})$, и обновляется множество наблюдений $S_{t+1} = (S_t, \mathcal{F}(x_{t+1}))$.
- Обновляется статистическая модель.

2.4 Tree-structured Parzen Estimator (TPE)

Алгоритм TPE итерационный: на каждой итерации принимается решение о том, какие следующие значения гиперпараметров нужно выбрать, исходя из результатов предыдущих итераций. Но идейно имеет довольно сильные отличия. Elistratova б.г.

- По значениям точек из выборки определяется квантиль y^γ, который делит данные на хорошие и остальные.
- Значение целевой функции оцуниваем на основе Parzen window density estimation:

$$\hat{\mathcal{F}}_h(x) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x - x_i}{h}),$$

  где K – ядро с дисперсией h(обычно соответствует нормальному распределению)
- Далее определяем функцию ожидаемого улучшения:

$$EI = \frac{l(x)}{g(x)},$$

  где l(x) и g(x) – это взвешенные суммы распределений целевой функции на основе хороших и плохих точек соответственно.
- Вычисляется значение $\mathcal{F}(\underset{x \in X}{argmax}\ EI(x))$, и обновляется известная выборка.

2.5 Гибридный подход

Основная идея гибридного подхода заключается в разделении задач глобального поиска перспективных областей параметрического пространства и локального уточнения оптимума. При ограниченном вычислительном бюджете прямое применение байесовской оптимизации на высокоточной физической модели оказывается неэффективным, поскольку большая часть ресурсов расходуется на локальное уточнение без существенного улучшения значения функционала.

Использование суррогатных моделей машинного обучения позволяет на ранних этапах оптимизации аппроксимировать форму целевой функции по дешёвым и шумным наблюдениям, выделяя области пространства параметров с повышенной вероятностью нахождения оптимума. Последующее ограничение поиска и применение более точных физических расчётов в выбранных подпространствах обеспечивает ускорение сходимости при сохранении качества конечного решения.

Гибридный подход подразумевает применение готового фреймворка с TPE и суррогатной модели. В качестве суррогата подойдут модели классического машинного обучения (случайный лес, SVM и т.д.) или нейросеть с небольшим числом параметров. Комбинируя суррогат с TPE можно реализовать активное обучение:

1. Обучаем суррогатную модель на дешевых данных, собранных из черного ящика случайным поиском;
2. С помощью TPE находим точку оптимума суррогатной модели;
3. Полученное множество точек кластеризуем;
4. Сужаем пространство параметров до границ каждого подмножества и запускаем сбор более дорогих данных на новом подпространстве;
5. Из полученных групп данных выбираем то, для которого среднее значение функции минимально;
6. Сужаем пространство параметров до границ лучшего подпространства;
7. Переобучаем модель на выбранном датасете и с помощью TPE предсказываем оптимум черного ящика;
8. Передаем предсказанную точку в черный ящик, полученное значение функции с целевой точностью добавляем в датасет;
9. Возвращаемся к пункту 7 заданное число итераций.

Следует подчеркнуть, что предлагаемый гибридный метод не претендует на строгую оптимальность в теоретическом смысле и носит эвристический характер. Тем не менее, его применение оправдано в условиях дорогостоящих стохастических расчётов, где приоритетом является эффективное распределение вычислительного бюджета, а не асимптотические гарантии сходимости. Корректность и практическая применимость метода подтверждаются экспериментальными результатами, приведёнными в главе 5.

Следует отметить, что предлагаемый гибридный подход носит эвристический характер и ориентирован на практическое повышение эффективности оптимизации в условиях ограниченного бюджета. Его корректность и применимость подтверждаются экспериментальными результатами, представленными в главе 5.

Глава 3

Структура фреймворка

3.1 Архитектура

Разработанный фреймворк предназначен для проведения автоматизированной оптимизации параметров нейтронно-физических систем на основе вычислительного кода Serpent. Архитектура системы построена модульным образом и обеспечивает возможность использования различных стратегий оптимизации, включая принципиально отличающиеся подходы. Таким образом, архитектура не привязана к конкретному алгоритму, а представляет собой универсальную инфраструктуру, которая обеспечивает единый цикл "предложение — моделирование — обновление" для любого подключаемого оптимизатора.

Общая схема системы

На концептуальном уровне архитектура состоит из следующих основных компонентов:

- **Parameter Space Manager** — менеджер пространства параметров, определяющий оптимизируемые величины, их типы, пределы, зависимости и правила валидации.
- **Optimizer** — абстрактный модуль, реализующий алгоритм выбора новых точек в параметрическом пространстве. Фреймворк поддерживает несколько реализаций данного интерфейса: метод адаптивной точности, основанный на стохастической регрессии и TPE, а также метод активного обучения, использующий эвристику неопределённости. Детальное описание этих методов приведено в следующих разделах.
- **Input Generator** — модуль изменения входного файла Serpent. Он преобразует

предложенные оптимизатором параметры в корректный набор команд, определяющий геометрию.

- **Serpent Runner** — слой взаимодействия с симулятором, включающий запуск вычислительных задач, контроль завершения, многопроцессное выполнение, обработку ошибок и повторные запуски при нестабильных результатах.
- **Result Parser** — механизм извлечения параметров целевой функции (скорость счета и статистическая ошибка) из выходных файлов Serpent и их передача в оптимизатор.
- **Storage and Logging** — подсистема сохранения логов, траекторий оптимизации, промежуточных результатов, состояний оптимизаторов и метаданных моделирования.

Унификация подходов и абстрактный интерфейс оптимизатора

Несмотря на то что используемые методы оптимизации существенно отличаются по своей природе и используют различные фреймворки, архитектура предоставляет им единый интерфейс взаимодействия. В основе лежит абстракция `Optimizer`, определяющая следующие операции:

- `suggest(`n`)` — генерация одного или нескольких наборов параметров для последующих моделирований;
- `update(`x, y, σ`)` — обновление модели оптимизатора на основе новых данных (параметров, отклика и статистической ошибки Serpent).

Такой интерфейс позволяет подключать любые алгоритмы как *плагины*, при этом не изменяя остальную часть системы. Различие между методами отражается исключительно в том, какую модель они поддерживают (gaussian process, TPE-модель, стохастическая регрессия, ансамбль деревьев и т.д.) и по каким правилам они формируют следующие точки. Все остальные компоненты архитектуры остаётся неизменными.

Единый вычислительный цикл

Архитектурно система реализует единый вычислительный цикл, общий для всех методов:

1. Оптимизатор предлагает параметры: $x \leftarrow$ `suggest()`.
2. Генератор формирует Serpent-вход.
3. Происходит запуск моделирования с учётом выбранной точности.

4. Парсер извлекает результат моделирования y и оценку статистической ошибки σ.
5. Оптимизатор обновляется по данным (x, y, σ).
6. Цикл повторяется до достижения критерия остановки.

Несмотря на то что различные оптимизаторы используют разные математические модели и критерии, их взаимодействие с системой полностью унифицировано. Это обеспечивает корректность сравнения подходов, снижает сложность разработки и позволяет повторно использовать компоненты.

Масштабируемость и расширяемость

Модульная архитектура позволяет легко:

- добавлять новые оптимизаторы без изменения существующего кода;
- изменять структуру входного файла Serpent, не затрагивая оптимизационные алгоритмы;
- повышать степень параллелизма моделирования;
- интегрировать новые метрики и ограничения;
- сохранять согласованность и воспроизводимость экспериментов.

Таким образом, архитектура выступает в роли универсального каркаса, обеспечивающего независимость инфраструктуры от конкретного оптимизационного метода и позволяющего использовать несколько взаимодополняющих стратегий в едином программном комплексе.

Архитектура фреймворка ориентирована не только на решение конкретной прикладной задачи, но и на воспроизводимость вычислительных экспериментов и корректное сравнение различных оптимизационных методов. Это позволяет рассматривать разработанную систему как исследовательскую платформу для анализа методов оптимизации вычислительно дорогих физических моделей.

3.2 Связь с Serpent

Интеграция оптимизационного фреймворка с транспортным кодом `Serpent 2` реализована через набор специализированных функций, предназначенных для автоматического изменения входных файлов, запуска расчётов на вычислительном кластере и последующего сбора результатов. Все функции являются строго привязанными к конкретной структуре файла `experiment.txt`, используемого в данной

работе. Изменение геометрии или номенклатуры поверхностей влечёт необходимость полного пересмотра логики генерации строк.

3.2.1 Автоматическое формирование входных файлов

Каждая параметризация реализована в виде метода класса `set_par`. Функции выполняют синтаксически корректную замену строк в исходном файле Serpent, опираясь на поиск по фиксированным шаблонам:

- изменение положения источника нейтронов;
- модификация толщины замедлителя и поглотителя (кадмий, полиэтилен);
- изменение геометрии активной зоны (призма, камера с топливом);
- алгоритмическая генерация набора детекторов с учётом их количества и координат.

Работа функций основана на следующем принципе:

1. входной файл считывается построчно;
2. строки, содержащие определённые ключевые префиксы (`surf`, `cell`, `sp`) сравниваются с перечнем шаблонов;
3. соответствующие строки заменяются вновь сформированными, содержащими параметры текущей итерации;
4. обновлённый файл перезаписывается.

Таким образом обеспечивается воспроизводимое формирование входных данных без необходимости ручного редактирования геометрии.

3.2.2 Запуск Serpent на кластере

После генерации входного файла выполняется автоматический запуск модели в терминале вычислительного узла. Для этого используется модуль `subprocess`:

```
subprocess.run(["sss2", "experiment.txt"], check=True)
```

Данная схема позволяет встроить Serpent в цикл оптимизации: каждая новая точка пространства параметров приводит к формированию новых входных данных и немедленному запуску транспортного расчёта.

3.2.3 Сбор и обработка результатов

После завершения вычислений фреймворк автоматически извлекает необходимые величины из выходных файлов Serpent:

- скорость реакции в детекторах;
- статистическая ошибка.

Полученные данные переводятся в формат `NumPy` и передаются в оптимизатор. Это обеспечивает возможность использовать методы машинного обучения и байесовской оптимизации поверх ресурсоёмкого физического моделирования.

3.3 Класс адаптивных TPE и BO with GP

Представленный класс объединяет два подхода к байесовской оптимизации — TPE (Tree-structured Parzen Estimator) и BO with GP (Gaussian Process Regression) — обеспечивая переключение между ними в зависимости от локальной сложности пространства поиска. Такой механизм позволяет динамически изменять как модель аппроксимации целевой функции, так и стратегию выбора следующей точки, сохраняя эффективность как на гладких, так и на сильно неоднородных участках.

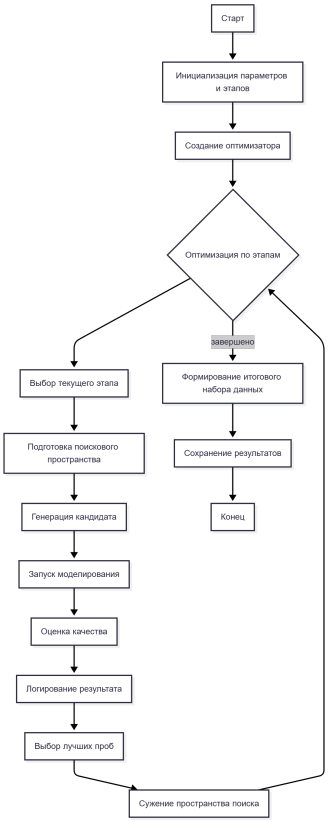


Рис. 3.1: Блок-схема Adaptive optimizer

В режиме TPE плотности $\ell(x)$ и $g(x)$ обновляются по адаптивным порогам качества, а выбор новой точки основан на максимизации отношения $g(x)/\ell(x)$. В

режиме Gaussian Process BO используется регрессионная модель с ядром Матерна, что обеспечивает более строгую аппроксимацию локальных структур. Значение acquisition function регулируется текущим уровнем неопределённости, что позволяет адаптивно переключаться между исследованием и эксплуатацией.

Класс отслеживает:

- Степень локальной вариативности функции (через кластеризацию и дисперсионные метрики).
- Стабильность предсказаний выбранной модели.
- Локальную плотность наблюдений вокруг текущей точки.

На этой основе выполняется переключение между TPE и GP-моделью. Такой подход минимизирует недостатки каждого метода по отдельности: TPE остаётся устойчивым на участках с резкими скачками, а GP обеспечивает более точные предсказания на гладких регионах.

Алгоритм работы класса целиком представлен на размещённой блок-схеме и включает циклическое обновление модели, расчёт целевой функции выбора точки и перегрузку параметров оптимизации с учётом текущей точности аппроксимации.

3.4 Гибридный метод

Гибридный метод объединяет идеи адаптивной оптимизации и активного обучения, позволяя динамически выбирать стратегию поиска оптимума в зависимости от текущего состояния пространства параметров. В отличие от базовых подходов, использующих лишь один оптимизатор, гибридный метод рассматривает совокупность критериев качества модели-посредника, распределений неопределённости и локальных свойств отклика.

Основная идея заключается в том, что для каждого шага оптимизации производится оценка информативности точки как в модели на основе деревьев, так и в гауссовском процессе. На основе этой оценки осуществляется выбор наиболее подходящего механизма генерации кандидатов. Такой подход позволяет объединить глобальную чувствительность TPE с локальной гладкостью GP, уменьшая вероятность попадания в неоптимальные локальные области и ускоряя сходимость.

В использовании гибридного метода особое значение имеет процедура адаптации: оптимизатор не только выбирает точки, но и контролирует, какая модель-посредник является более надёжной в текущей части пространства. Благодаря

этому метод демонстрирует устойчивость при сложных, разреженных и мульти-экстремальных задачах.

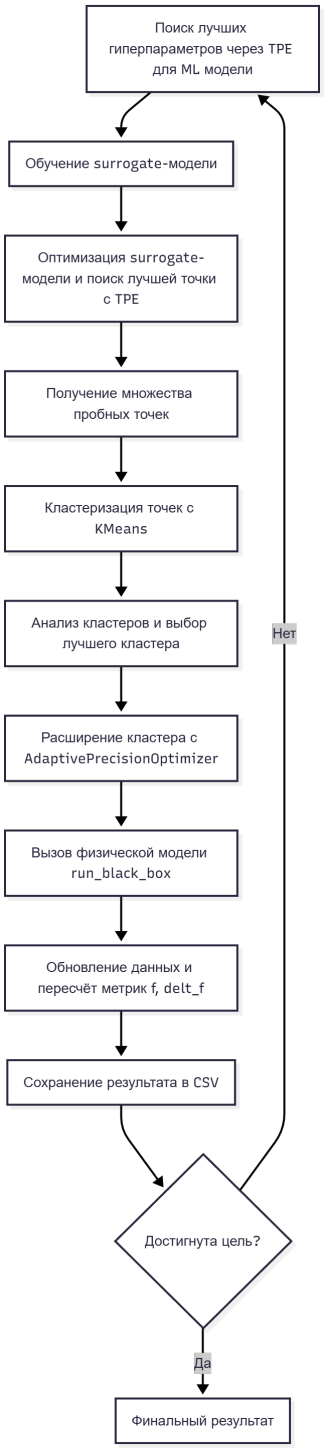


Рис. 3.2: Блок-схема activeL

Глава 4

Реализация оптимизационных алгоритмов

В данной главе описана программная реализация адаптивных и гибридных оптимизационных алгоритмов, применяемых для решения задачи оптимизации параметров нейтронно-физической модели. Основное внимание уделено архитектурным решениям, обеспечивающим масштабируемость, воспроизводимость вычислительных экспериментов и корректный учёт физических ограничений задачи.

4.1 Принципы реализации оптимизационных алгоритмов

При реализации оптимизационных алгоритмов основное внимание уделялось обеспечению воспроизводимости, модульности и корректного учёта физических ограничений задачи. Все проектные решения принимались с учётом высокой стоимости одного запроса оракула и наличия статистического шума в значениях целевой функции.

4.2 Реализация адаптивного оптимизатора

4.2.1 Общая архитектура оптимизатора

Все методы байесовской оптимизации, включая Tree-structured Parzen Estimator (TPE) и байесовскую оптимизацию с гауссовским процессом (BO with GP), реализованы в рамках единого класса `AdaptivePrecisionOptimizer`. Данный класс

выполняет роль оркестратора между библиотекой Optuna и внешним нейтронно-физическим расчётным кодом.

```
class AdaptivePrecisionOptimizer:

    def __init__(self, bounds, stages, sampler_name):
        self.bounds = bounds
        self.stages = stages
        self.sampler_name = sampler_name
        self.study = None
        self.history = []
```

Листинг 4.1: Скелет класса AdaptivePrecisionOptimizer

Такое архитектурное решение позволяет использовать идентичные условия эксперимента для различных алгоритмов оптимизации и проводить их корректное сравнение.

4.2.2 Инициализация и выбор сэмплера

Выбор алгоритма байесовской оптимизации осуществляется с помощью фабричного метода, что упрощает расширение системы новыми сэмплерами.

```
def _create_sampler(self):

    if self.sampler_name == 'TPE':
        return optuna.samplers.TPESampler()

    if self.sampler_name == 'GP':
        return optuna.samplers.GPSampler(
            n_startup_trials=0
        )

    raise ValueError("Unsupported sampler")
```

Листинг 4.2: Фабричный метод выбора сэмплера Optuna

4.2.3 Многоэтапная стратегия оптимизации

Оптимизация проводится в несколько этапов, каждый из которых характеризуется числом испытаний и числом моделируемых нейтронов. Это позволяет постепенно увеличивать точность расчётов и уточнять область поиска.

```
for stage_id, (n_trials, neutrons) in enumerate(self.stages):

    self.current_stage = stage_id
    self.neutrons = neutrons
```

```
5
6       self.study = optuna.create_study(
7           sampler=self._create_sampler(),
8           direction='minimize'
9       )
10
11      self.study.optimize(
12          self._objective,
13          n_trials=n_trials
14      )
15
16      self._update_bounds()
```

Листинг 4.3: Запуск многоэтапной оптимизации

4.2.4 Адаптивное сужение пространства параметров

После завершения каждого этапа выполняется автоматическое сужение пространства поиска на основе лучших найденных решений.

```
1  def _update_bounds(self):
2
3       trials = sorted(
4           self.study.trials,
5           key=lambda t: t.value
6       )
7
8       best = trials[:max(5, len(trials)//10)]
9       new_bounds = {}
10
11      for name, (low, high) in self.bounds.items():
12          values = [t.params[name] for t in best]
13          mean = np.mean(values)
14          width = 0.25 * (high - low)
15
16          new_bounds[name] = (
17              max(low, mean - width),
18              min(high, mean + width)
19          )
20
21      self.bounds = new_bounds
```

Листинг 4.4: Алгоритм адаптивного сужения границ

Используемый алгоритм сужения пространства параметров не является оптимальным в строгом математическом смысле, однако обеспечивает устойчивое уменьшение области поиска без преждевременной потери глобального минимума, что

подтверждается экспериментальными результатами.

4.2.5 Формирование параметров с физическими ограничениями

При генерации параметров учитываются явные физические зависимости между величинами, что предотвращает появление нефизических конфигураций.

```
def _suggest_params(self, trial):

    params = {}

    params['counters_n'] = trial.suggest_int(
        'counters_n',
        *self.bounds['counters_n']
    )

    h_poly = trial.suggest_float(
        's_poly',
        *self.bounds['s_poly']
    )

    h_cnt = trial.suggest_float(
        'counters_h',
        *self.bounds['counters_h']
    )

    params['s_poly'] = max(h_poly, h_cnt + 2.0)
    params['counters_h'] = min(h_cnt, params['s_poly'] - 1.5)

    return params
```

Листинг 4.5: Генерация параметров с физическими зависимостями

4.2.6 Целевая функция и физический расчёт

Целевая функция реализует обёртку над нейтронно-физическим расчётом и включает вычисление функционала качества и его статистической погрешности.

```
def _eval_physics(self, params):

    settings.set_neutrons(self.neutrons)
    a, b, da, db = settings.run_black_box(params)

    f = np.sqrt(a + b) / np.sqrt(3900) / (a - b)

    df = (
        f / (a**2 - b**2)
        * np.sqrt(
```

```
            (3*a + b)**2 * (a * da)**2 +
            (3*b + a)**2 * (b * db)**2
        ) / 2
    )

    return f, df
```

Листинг 4.6: Физическая часть целевой функции

```
def _objective(self, trial):

    params = self._suggest_params(trial)
    f, df = self._eval_physics(params)

    trial.set_user_attr('df', df)
    trial.set_user_attr('stage', self.current_stage)

    return f
```

Листинг 4.7: Полная целевая функция Optuna

4.2.7 Логирование и воспроизводимость

Все результаты оптимизации сохраняются в структурированном виде, что обеспечивает воспроизводимость экспериментов.

```
def save_history(self, path):

    records = []

    for t in self.study.trials:
        rec = dict(t.params)
        rec['value'] = t.value
        rec['df'] = t.user_attrs.get('df')
        rec['stage'] = t.user_attrs.get('stage')
        records.append(rec)

    pd.DataFrame(records).to_csv(path)
```

Листинг 4.8: Сохранение результатов оптимизации

4.3 Реализация гибридного метода оптимизации

4.3.1 Общая схема гибридного подхода

Гибридный метод сочетает машинное обучение и байесовскую оптимизацию. Суррогатная модель используется для быстрого анализа пространства параметров, после чего проводится уточнение оптимума на основе физической модели.

4.3.2 Реестр суррогатных моделей

Для обеспечения модульности реализован реестр суррогатных моделей.

```
MODEL_REGISTRY = {
    'bayesian_ridge': BayesianRidge,
    'random_forest': RandomForestRegressor,
    'catboost': CatBoostRegressor
}
```

Листинг 4.9: Реестр суррогатных моделей

4.3.3 Подготовка обучающей выборки

```
X = df[PARAMETER_COLUMNS].values
y = df['value'].values
w = 1.0 / df['df'].values
```

Листинг 4.10: Формирование обучающей выборки

4.3.4 Подбор гиперпараметров суррогатной модели

```
def surrogate_objective(trial):

    params = {
        'depth': trial.suggest_int('depth', 3, 8),
        'learning_rate': trial.suggest_float(
            'learning_rate', 0.01, 0.3
        )
    }

    model = CatBoostRegressor(**params, verbose=0)

    score = cross_val_score(
        model, X, y,
        cv=5,
        scoring='neg_mean_squared_error'
    ).mean()

```

```
    return score
```

Листинг 4.11: Целевая функция подбора гиперпараметров

4.3.5 Глобальный поиск оптимума суррогата

```
study = optuna.create_study(direction='minimize')
study.optimize(surrogate_objective, n_trials=100)
```

Листинг 4.12: Оптимизация суррогатной модели

4.3.6 Кластеризация и локальная оптимизация

```
kmeans = KMeans(n_clusters=3)
labels = kmeans.fit_predict(X)
```

Листинг 4.13: Кластеризация найденных решений

```
cluster_bounds = estimate_cluster_bounds(X, labels)

local_optimizer = AdaptivePrecisionOptimizer(
    cluster_bounds,
    stages=[(50, 1e7)],
    sampler_name='TPE'
)
```

Листинг 4.14: Локальная оптимизация в кластере

4.3.7 Активное обучение

```
for iteration in range(5):

    model.fit(X, y, sample_weight=w)

    x_new = propose_next_point(model)
    f_new, df_new = run_physics(x_new)

    X = np.vstack([X, x_new])
    y = np.append(y, f_new)
    w = np.append(w, 1.0 / df_new)
```

Листинг 4.15: Цикл активного обучения

4.3.8 Преимущества и ограничения

Предложенная реализация отличается модульной архитектурой, корректным учётом физических ограничений и эффективным использованием вычислительных ресурсов. К ограничениям относятся зависимость от качества начальной выборки и вычислительная стоимость обучения сложных суррогатных моделей.

Глава 5

Эксперимент & Результаты

В данной главе представлены результаты вычислительных экспериментов, направленных на сравнение различных методов оптимизации в условиях стохастических нейтронно-физических расчётов методом Монте-Карло. Эксперименты проводились при фиксированном вычислительном бюджете и одинаковых условиях моделирования, что обеспечивает корректность сопоставления рассматриваемых подходов.

Особое внимание уделено анализу эффективности гибридного метода оптимизации, сочетающего суррогатное моделирование, кластеризацию параметрического пространства и адаптивное управление точностью вычислений. Ввиду высокой вычислительной стоимости полного перебора вариантов и невозможности проведения исчерпывающего набора численных экспериментов, анализ результатов дополняется рассмотрением вклада отдельных компонентов алгоритма в общую эффективность оптимизации. Такой подход позволяет не только сравнить методы между собой, но и выявить ключевые механизмы, обеспечивающие преимущество гибридной стратегии при ограниченном вычислительном бюджете.

5.1 Гипотезы и дизайн эксперимента

В рамках экспериментального исследования проверялись следующие гипотезы:

- методы байесовской оптимизации обладают большей устойчивостью к статистическому шуму;
- гибридные методы обеспечивают более быстрое схождение при фиксированном вычислительном бюджете;

- использование устойчивых суррогатных моделей снижает вероятность неудачных расчётов.

Для проверки гипотез каждый метод запускался независимо несколько раз при одинаковых условиях эксперимента.

5.2 Условия

При постановке задачи было решено найти минимум искомого функционала с относительной погрешностью менее 5 %. Для этого необходимо при запуске Serpent симулировать перенос 5e7 нейтронов.

Все запуски кода Serpent выполнялись на кластере, используя один процессор Intel(R) Xeon(R) Gold 6230 на 40 потоков. Так, один запрос значения черного ящика при максимальной установленной точности занимает 1,5-2 часа.

Так как каждый запуск Serpent на установленной точности – это очень дорогостоящая операция, было решено применять адаптивную точность измерений. Она заключается в разделении запросов оракула на стадии. Каждая стадия имеет собственные точность и число запросов. Значения подобраны так, чтобы все стадии занимали приблизительно одинаковое время.

	Число нейтронов				**Число запросов**			
Способ	**Эт.1**	**Эт.2**	**Эт.3**	**Эт.4**	**Эт.1**	**Эт.2**	**Эт.3**	**Эт.4**
TPE/BO	1e5	1e6	1e7	5e7	500	100	25	10
Гибрид	1e5	1e7	5e7	-	1000	150	6	-

Таблица 5.1: Сравнение методов по характеристикам на каждом этапе

5.3 Метрики

Для оценки качества методов оптимизации сравнивались:

1. Суммарное время схождения к результату;
2. Наименьшее достигнутое значение целевой функции при заданной точности;
3. Робастность;
4. Дисперсия результата.

5.4 Сравнение методов

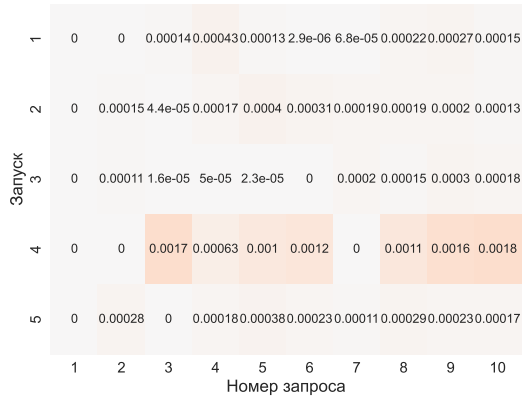
Чтобы получить представление об устойчивости, оптимизация каждым методом проводилась 5 раз. Для сравнения методов по заранее определенным метрикам были построены:

- Графики зависимостей лучшего найденного значения от номера запуска черного ящика на последней стадии;
- Тепловые карты с разностями между лучшим и текущим значениями на каждом шаге;
- Таблицы времени схождения к результату;
- Таблицы с наименьшим достигнутым значением.

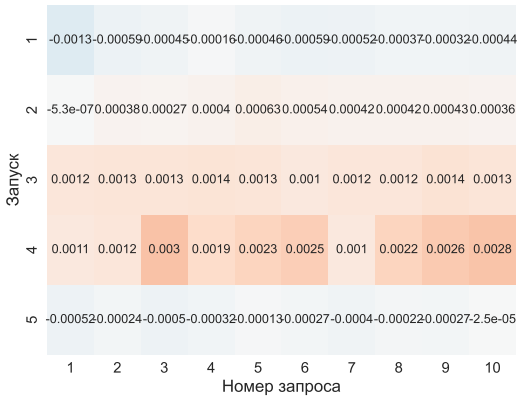
Способ	**TPE**	**GP**	**Forest**	**Boosting**	**Bayesian Ridge**
Среднее время, час	32	30	16	14	10

Таблица 5.2: Сравнение времени схождения к оптимуму

Долгое время схождения к результату у GP и TPE связано с большим числом запросов оракула на последней стадии. При переходе к испытаниям гибридного метода было решено снизить число самых дорогих запросов с целью ускорения исследования методов.

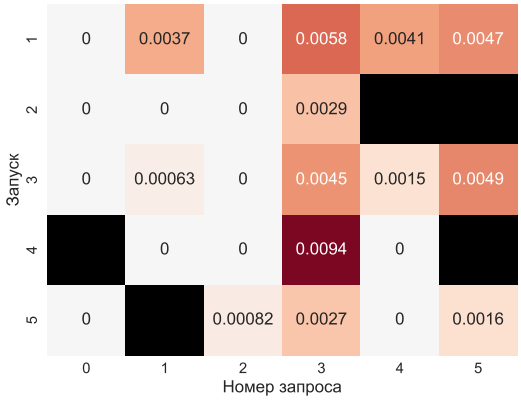


(a) TPE (Tree-structured Parzen Estimator)

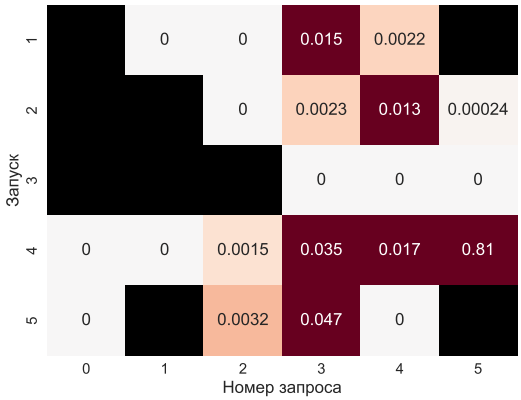


(b) Bayesian Optimization с Gaussian Process

(c) Случайный лес



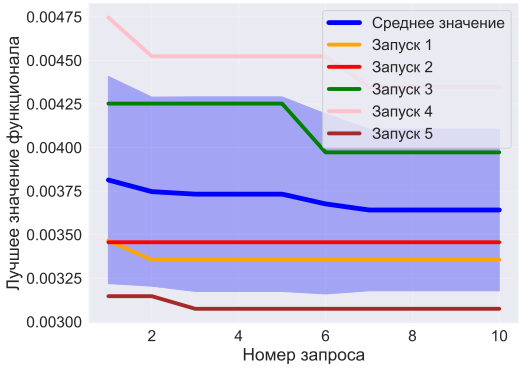
(d) Градиентный бустинг



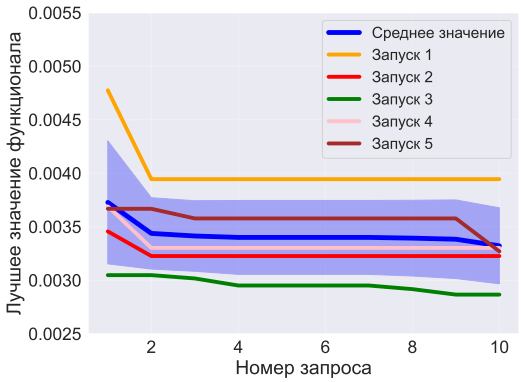
(e) Байесовская регрессия с полиномиальными признаками

Рис. 5.1: Робастность

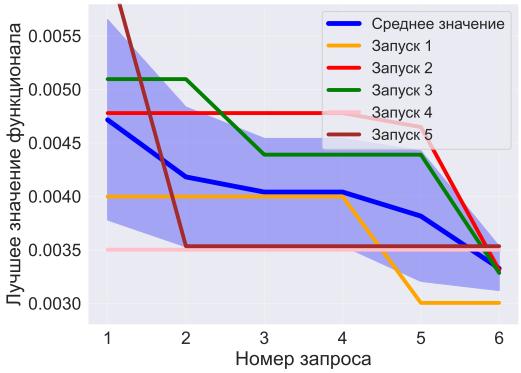
Черным выделены неудавшиеся попытки, то есть значение целевой функции либо не удалось получить, либо оно оказалось меньше ноля. Видно, что методы BO являются более устойчивыми, чем гибридные, однако модель со случайным лесом также показала высокую устойчивость к шумным данным.



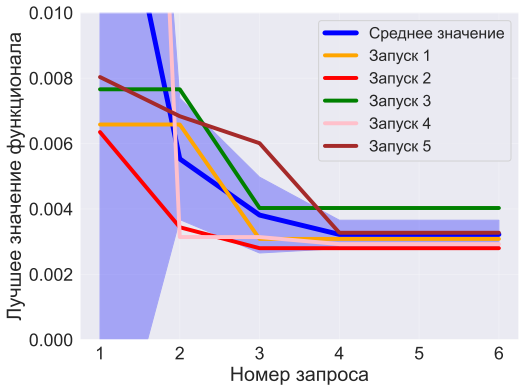
(a) TPE (Tree-structured Parzen Estimator)



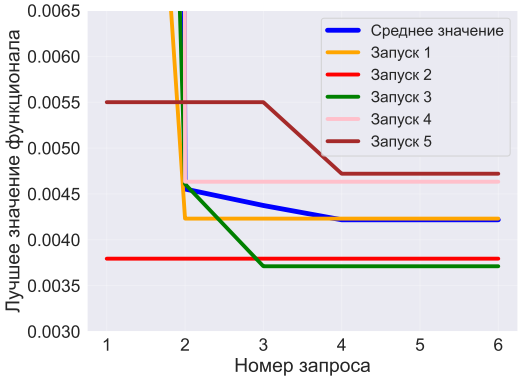
(b) Bayesian Optimization с Gaussian Process



(c) Случайный лес



(d) Градиентный бустинг



(e) Байесовская регрессия с полиномиальными признаками

Рис. 5.2: Результаты оптимизации разными методами

Самые устойчивые модели BO также демонстрируют невысокое улучшение результата на последней стадии. Самые нейстойчивые Boosting и Bayesian Ridge из-за частых ошибок также не показывают большого улучшения на последней стадии. Наиболее устойчивый к шуму Random Forest демонстрирует самую высокую динамикку схождения.

Способ	TPE	GP	Forest	Boosting	Bayesian Ridge
Средний минимум	3.64e-03	3.32e-3	3.33e-03	3.22e-03	4.22e-03
Дисперсия	4.6e-04	3.5e-4	2.1e-04	4.3e-04	4.3e-04

Таблица 5.3: Сравнение достигнутых минимумов

5.5 Анализ результатов

В данном разделе приведён анализ поведения различных методов оптимизации на основе полученных экспериментальных данных. Особое внимание уделено устойчивости методов к статистическому шуму, скорости схождения и качеству найденных решений.

При интерпретации результатов следует учитывать, что сравниваемые методы оптимизации находятся в принципиально разных режимах работы с шумной целевой функцией. Методы байесовской оптимизации ориентированы на аккуратное локальное уточнение экстремума и демонстрируют высокую устойчивость, однако при фиксированном вычислительном бюджете склонны к преждевременной эксплуатации уже найденных областей.

Гибридные методы, напротив, допускают более агрессивное исследование пространства параметров за счёт использования суррогатных моделей, что повышает вероятность нахождения более глубокого минимума, но сопровождается увеличением дисперсии результатов. Таким образом, сравнение методов следует рассматривать как анализ компромисса между устойчивостью и скоростью сходимости.

5.5.1 Поведение методов байесовской оптимизации

Методы байесовской оптимизации, основанные на TPE и гауссовском процессе, продемонстрировали наиболее устойчивое поведение в условиях шумной целевой функции. На тепловых картах (рис. 5.1) видно, что доля неудачных запусков для данных методов минимальна, что свидетельствует о корректной работе механизмов учёта неопределённости.

В то же время, оба метода характеризуются относительно слабой динамикой улучшения целевой функции на последней стадии оптимизации. Это связано с тем, что при высокой точности расчётов стоимость одного запроса оракула существенно возрастает, а байесовские методы склонны к локальному уточнению уже найденного минимума. В результате дополнительные испытания не приводят к значимому

улучшению результата.

Для TPE это проявляется в стабилизации лучшего значения после ограниченного числа запусков, что указывает на достижение плато по качеству решения. Для метода с гауссовским процессом ситуация усугубляется высокой вычислительной сложностью обновления модели и чувствительностью к шумным наблюдениям, что объясняет невозможность получения устойчивых результатов на последней стадии при заданном бюджете вычислений.

5.5.2 Гибридные методы на основе суррогатных моделей

Гибридные методы, использующие суррогатные ML-модели, продемонстрировали существенно различное поведение в зависимости от выбранного типа модели.

Суррогатная модель на основе случайного леса показала наилучшее сочетание устойчивости и скорости схождения. Это объясняется высокой устойчивостью ансамблевых методов к шуму в обучающих данных, а также их способностью аппроксимировать нелинейные зависимости без явного предположения о форме целевой функции. В результате предлагаемые моделью точки редко приводят к нефизическим или неудачным расчётам, что подтверждается минимальным числом чёрных областей на тепловых картах.

Градиентный бустинг, напротив, демонстрирует наиболее агрессивную стратегию поиска. Предлагаемые им точки часто приводят к резкому улучшению значения функционала, однако сопровождаются высокой дисперсией результата. Это указывает на переобучение модели на ограниченном и шумном наборе данных, в результате чего предсказания оказываются нестабильными. Таким образом, несмотря на потенциально более глубокий минимум, данный метод характеризуется низкой робастностью.

Байесовская линейная регрессия с полиномиальными признаками оказалась наименее устойчивой моделью. Её предположение о гладкости и глобальной структуре зависимости плохо согласуется с реальной формой целевой функции, что в сочетании со статистическим шумом приводит к значительному разбросу результатов и худшему среднему значению минимума.

5.5.3 Сравнение качества найденных решений

Анализ табл. 5.3 показывает, что наименьшие значения целевой функции в среднем достигаются гибридными методами на основе случайного леса и градиентного бустинга. Однако с учётом дисперсии результатов наиболее предпочтительным является метод со случайным лесом, обеспечивающий компромисс между качеством

решения и его воспроизводимостью.

Методы байесовской оптимизации демонстрируют более консервативное поведение, что делает их надёжными в условиях строгих ограничений на допустимость решений, однако менее эффективными при фиксированном вычислительном бюджете.

5.5.4 Влияние адаптивной точности

Использование многостадийной схемы с адаптивным числом моделируемых нейтронов позволило существенно сократить общее время экспериментов без потери качества финального решения. Ранние стадии обеспечивают грубое исследование пространства параметров, в то время как финальные стадии используются для уточнения найденных оптимальных областей.

Результаты показывают, что дальнейшее увеличение числа запусков на максимальной точности не приводит к существенному улучшению результата, что подтверждает корректность выбранной стратегии распределения вычислительного бюджета.

5.5.5 Анализ вклада отдельных компонентов алгоритма

Для оценки вклада отдельных компонентов предложенного гибридного метода был проведён абляционный анализ на концептуальном уровне. Рассматривались три ключевых элемента алгоритма: использование суррогатной модели для предварительного исследования пространства параметров, кластеризация параметрического пространства с последующим ограничением области поиска, а также адаптивное управление точностью нейтронно-физических расчётов.

Исключение суррогатного моделирования на начальных этапах оптимизации приводит к необходимости выполнения высокоточных физических расчётов на всём пространстве параметров, что при фиксированном вычислительном бюджете существенно снижает эффективность глобального поиска и увеличивает вероятность преждевременной сходимости к локальному минимуму. Отказ от этапа кластеризации приводит к рассеиванию вычислительных ресурсов между несколькими конкурирующими областями пространства параметров, что замедляет локальное уточнение оптимума. Наконец, использование фиксированной точности расчётов на всех стадиях оптимизации приводит либо к избыточным вычислительным затратам на ранних этапах, либо к недостаточной статистической достоверности на этапах локального уточнения.

Таким образом, каждый из рассмотренных компонентов играет существенную роль

в общей эффективности гибридного метода, а их совместное использование обеспечивает наблюдаемое преимущество по скорости сходимости и качеству получаемых решений при ограниченном вычислительном бюджете.

5.5.6 Анализ пространства параметров

Помимо различий в абсолютных значениях достигнутого минимума целевого функционала, рассматриваемые методы демонстрируют различия в локализации оптимальных решений в пространстве параметров. Анализ распределений оптимальных значений позволяет разделить все варьируемые параметры на две характерные группы.

1. Параметры, для которых наблюдается чётко выраженный экстремум функционала:
   - толщина полиэтиленового слоя;
   - ширина камеры с топливом;
   - относительное расстояние между счётчиками;
   - вертикальное положение счётчиков.
2. Параметры, для которых в окрестности минимума наблюдается широкое плато:
   - толщина кадмиевого слоя;
   - высота матрицы замедлителя;
   - ширина матрицы замедлителя;
   - положение источника нейтронов;
   - высота камеры с топливом;
   - число счётчиков.

Для параметров первой группы оптимальное значение может быть однозначно выбрано как соответствующее положению экстремума целевого функционала. В случае параметров второй группы наличие протяжённого плато указывает на слабую чувствительность функционала к их вариациям вблизи минимума. В связи с этим в качестве предпочтительных принимаются значения нижнего квантиля соответствующих распределений, что обусловлено соображениями снижения материальных и конструктивных затрат.

Указанная классификация параметров подтверждается анализом распределений оптимальных значений, представленных в виде диаграмм размаха в приложении C.

5.5.7 Общие выводы

Таким образом, проведённые эксперименты подтверждают, что в условиях шумной и дорогостоящей целевой функции гибридные методы оптимизации, основанные на устойчивых суррогатных моделях, способны обеспечивать более быстрое схождение и лучшее качество решений по сравнению с классическими методами байесовской оптимизации.

Количественная верификация абляционного анализа представляет собой отдельное направление дальнейших исследований и требует дополнительных вычислительных ресурсов.

Глава 6

Заключение

6.1 Основные выводы

В рамках данной работы была рассмотрена задача оптимизации конфигурации нейтронного детектора на основе ^3He в условиях стохастических нейтронно-физических расчётов методом Монте-Карло. Показано, что данная задача относится к классу оптимизации стохастических функционалов при ограниченном вычислительном бюджете и гетероскедастическом шуме, что существенно ограничивает применимость классических методов оптимизации и стандартных реализаций байесовской оптимизации.

Разработан модульный программный фреймворк, обеспечивающий корректное и воспроизводимое сравнение оптимизационных алгоритмов в условиях дорогостоящих физических расчётов. Архитектура фреймворка позволяет гибко комбинировать физические модели, суррогатные модели машинного обучения и различные оптимизационные стратегии, что делает его применимым к широкому классу задач параметрической оптимизации в нейтронной физике.

Проведено сравнительное исследование методов байесовской оптимизации с гауссовскими процессами, алгоритмов на основе TPE, а также гибридных подходов с использованием суррогатных моделей. Показано, что при фиксированном вычислительном бюджете методы байесовской оптимизации с GP обеспечивают высокую устойчивость, однако уступают по скорости сходимости в условиях выраженного стохастического шума и высокой стоимости одного вычислительного эксперимента. Алгоритмы на основе TPE демонстрируют лучшую масштабируемость и устойчивость к шуму, однако не учитывают явно различную точность наблюдений.

Ключевым результатом работы является разработка и экспериментальная верификация гибридного метода оптимизации, сочетающего суррогатное моделирование, кластеризацию пространства параметров и адаптивное управление точностью нейтронно-физических расчётов. Показано, что предлагаемый подход позволяет более эффективно распределять вычислительный бюджет, концентрируя высокоточные расчёты в наиболее перспективных областях параметрического пространства. В вычислительных экспериментах гибридный метод демонстрирует преимущество по достижению меньших значений целевой функции по сравнению с классическими методами при сопоставимых вычислительных затратах.

Отдельно установлено, что использование адаптивной точности расчётов является принципиально важным элементом оптимизации в условиях Монте-Карло моделирования. Управление числом моделируемых нейтронов в зависимости от стадии оптимизации и локальной неопределённости суррогатной модели позволяет существенно снизить суммарные вычислительные затраты без ухудшения качества конечного решения.

6.2 Ограничения и направления дальнейших исследований

Полученные в работе результаты формируют задел для дальнейшего развития предложенного подхода. В первую очередь перспективным направлением является более строгий учёт стохастической природы целевой функции, включая явное моделирование гетероскедастического шума в рамках байесовской оптимизации и использование методов, позволяющих совместно оптимизировать положение точки и требуемую точность вычислений.

Дополнительным направлением развития является внедрение многоуровневых стратегий оптимизации, таких как BOHB Falkner, Klein и Hutter 2018 и hyperband Li и др. 2018, позволяющих автоматически распределять вычислительный бюджет между стадиями различной точности без ручного задания числа запусков и параметров моделирования. Это особенно актуально для задач с существенно различающейся стоимостью одного вычислительного эксперимента.

Перспективным также является расширение пространства оптимизируемых параметров, включая геометрические характеристики детектора, параметры материалов и условия облучения. В сочетании с предложенным гибридным подходом это позволит перейти от оптимизации отдельных конфигураций к систематическому исследованию классов детекторных систем.

Наконец, дальнейшее развитие фреймворка возможно в направлении интеграции с высокопроизводительными вычислительными платформами и облачными сервисами, а также в сторону автоматизации постановки вычислительных экспериментов и анализа результатов. Это открывает возможность применения разработанных методов в задачах практического проектирования и оптимизации сложных нейтронно-физических систем.

Список литературы

Elistratova, E. (б.г.). «Hyperparameter selection». В: *Yandex Handbook, Machine learning tutorial* (). URL: https://education.yandex.ru/handbook/ml/article/podbor-giperparametrov.

Falkner, S., A. Klein и F. Hutter (2018). «BOHB: Robust and Efficient Hyperparameter Optimization at Scale». В: *ICML2018*.

Gardner, J. R. и др. (2023). «Bayesian Optimization with Inequality Constraints». В: *Advances in Neural Information Processing Systems*.

Li, L. и др. (2018). «Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization». В: *Journal of Machine Learning Research* 18, с. 1—52.

Stacey, W. (2007). *Nuclear Reactor Physics*. Wiley-VCH.

Watanabe, S. (2023). *Tree-Structured Parzen Estimator: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance*. arXiv preprint arXiv:2304.11127.

Приложение А

Спецификация проекта

№	Функциональное требование	Описание
1	Генерация входных файлов Serpent	Система принимает параметры задачи (геометрию, материалы, источники) и автоматически формирует корректный входной файл Serpent.
3	Автоматический запуск симуляций	Система запускает Serpent на основе сгенерированного файла, отслеживает выполнение, собирает выходные данные.
4	Формирование датасета	Автоматическое сохранение входных файлов, параметров и результатов симуляций в стандартизированном формате.
5	Обучение модели	Поддержка обучения и дообучения модели для улучшения качества генерации входных файлов.
6	Управление экспериментами	Единый интерфейс для генерации, запуска симуляций, логирования и анализа результатов.

№	Нефункциональное требование	Описание
1	Производительность	Генерация файла занимает менее 1 секунды; симуляции стартуют без задержек; обучение возможно на GPU (например, AWS g4dn).
2	Надёжность	Система корректно обрабатывает некорректные входные данные, сохраняет логи ошибок и обеспечивает воспроизводимость экспериментов.
3	Масштабируемость	Лёгкое добавление новых типов входных файлов, расширение датасета и внедрение новых модулей.
4	Совместимость	Работа в средах Python 3.10+, Linux-сервер, Google Colab; экспорт данных в CSV/JSON.
5	Поддерживаемость	Модульная архитектура, документация, комментарии и структура кода, облегчающие сопровождение.
6	Юзабилити	Удобный интерфейс (CLI или Jupyter Notebook); настройка параметров без ручного редактирования файлов.
7	Безопасность	Запуск симуляций в изолированной среде; контроль версий данных и входных файлов.

Приложение B

Управление проектом

План работ

№	Этап	Подзадачи	Планируемый результат
1	Подготовка	• Анализ литературы	
		• Постановка задачи	Литобзор
2	Разработка	• Реализация фреймворков	
		• Апробация методов	Работоспособные фреймворки
3	Эксперименты	• Запуск симуляций Serpent	
		• Сбор данных	Датасет результатов
4	Анализ	• Обработка результатов	
		• Визуализация	Кривые сходимостей
5	Документация	• Написание отчета	
		• Оформление кода	Оформленный проект

Диаграмма Ганта

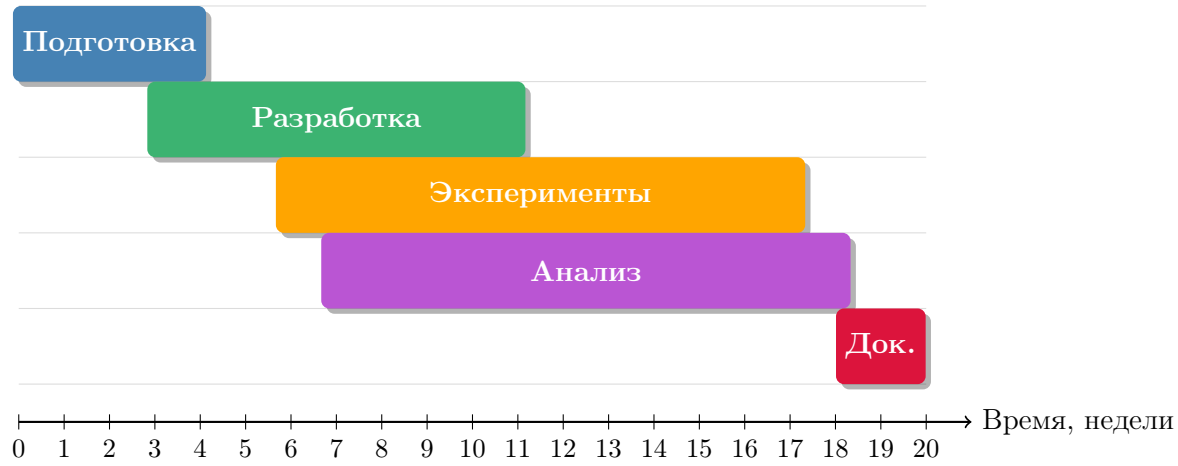


Рис. В.1: Диаграмма Ганта проекта

Приложение C

Boxplot

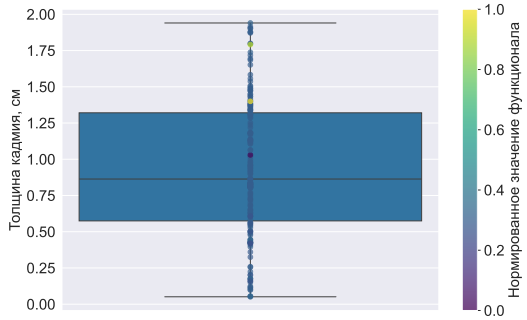


Рис. C.1: Толщина кадмиевого листа

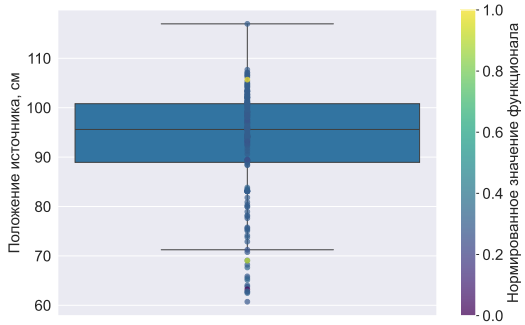


Рис. C.2: Расстояние между кадмием и источником

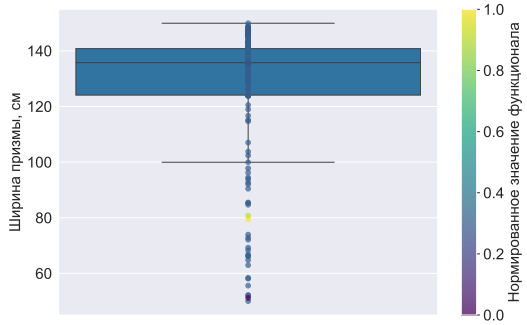


Рис. C.3: Ширина матрицы замедлителя

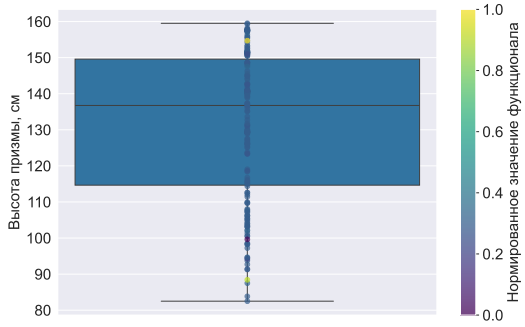


Рис. C.4: Высота матрицы замедлителя

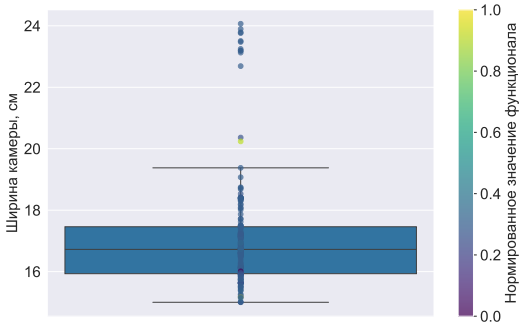


Рис. С.5: Ширина камеры с топливом

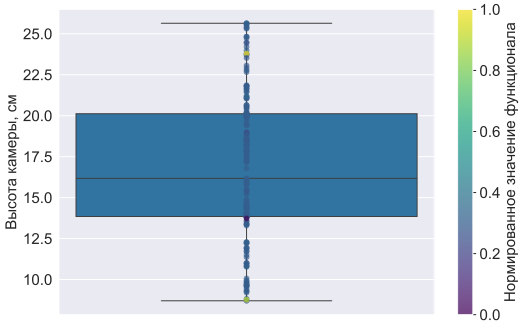


Рис. С.6: Высота камеры с топливом

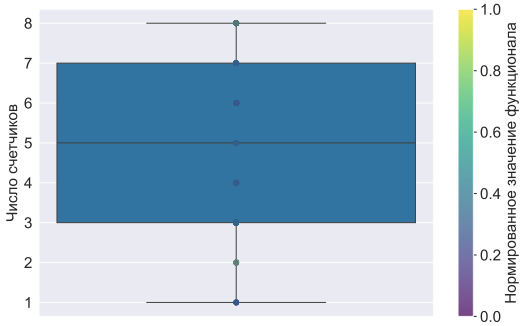


Рис. С.7: Число счетчиков

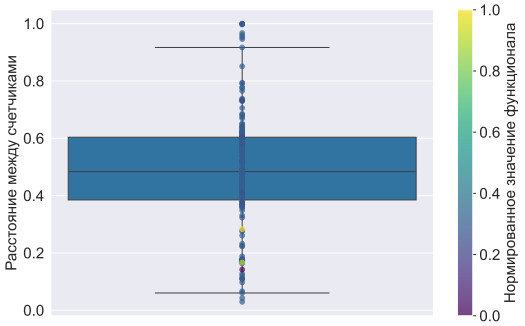


Рис. С.8: Относительное расстояние между счетчиками

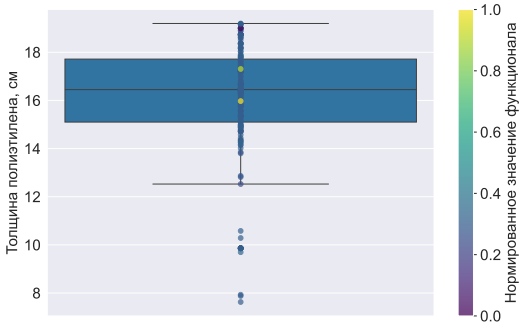


Рис. С.9: Толщина блока полиэтилена

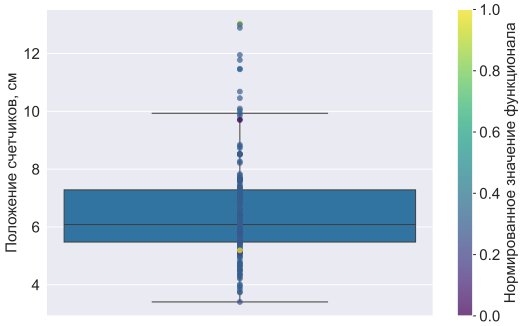


Рис. С.10: Расстояние между нижней границей полиэтилена и счетчиками