Министерство образования и науки Российской Федерации

Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа управления киберфизическими системами

|  |
| --- |
|  |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА**

**ОПРЕДЕЛЕНИЕ КОЛИЧЕТСВА РАБОТАЮЩИХ ГОРЕЛОК ПО РАСХОДУ МАЗУТА И ИЗМЕНЕНИЯ ИХ КОЛИЧЕСТВА ПРИ ИЗМЕНЕНИ ДАВЛЕНИЯ МАЗУТА**

направление 27.03.04 – Управление в технических системах

профиль 27.03.04\_05 - Интеллектуальные системы обработки информации и

управления

|  |  |
| --- | --- |
| Выполнил  студент гр. 5132704/00501  Руководитель  к.т.н, доцент | А. В. Александров  В. Н. Хохловский |

Санкт-Петербург

2024

Содержание

[Введение 5](#_Toc136866882)

[1. Общие положение об АСУ ТП и работе готлоагрегатов 7](#_Toc136866883)

[1.1. Общие положения о котлоагрегатаъ и принципе их работы 7](#_Toc136866884)

[1.2. Режимная карта и её роль в эксплуатации котлоагрегата 10](#_Toc136866885)

[1.3. Ключевые параметры работы в определении количества включенных горелок 12](#_Toc136866886)

[2. Математические методы в обработке и анализе данных работы котлоагрегата 16](#_Toc136866888)

[2.1. Интерполяция 17](#_Toc136866889)

[2.2. Аппроксимация 20](#_Toc136866890)

[2.3. Метод скользящего среднего 23](#_Toc136866891)

[2.4. Применение кластеризации в контексте определения режима работы котлоагрегата 23](#_Toc136866891)

[2.5. Определение оптимального подхода в сглаживании исходных графиков 23](#_Toc136866891)

[3. Реализация программы 25](#_Toc136866892)

[3.1. Алгоритм определения количества работающих горелок 26](#_Toc136866893)

[3.2. Серверная часть 31](#_Toc136866894)

[3.2. Клиентская часть 31](#_Toc136866894)

[4. Перспективы и направления дальнейшей разработки 25](#_Toc136866892)

[4.1. Ограничения разработанного алгоритма определения количества работающих горелок 26](#_Toc136866893)

[4.2. Интеграция методов машинного обучения 31](#_Toc136866894)

[4.2. Контроль со стороны оператора 31](#_Toc136866894)

[Заключение 36](#_Toc136866899)

[Список использованной литературы 38](#_Toc136866900)

[Приложение А 40](#_Toc136866901)

[Листинг метода интерполяции 40](#_Toc136866902)

[Приложение Б 46](#_Toc136866903)

[Листинг метода полиномиальной аппроксимации 46](#_Toc136866904)

[Приложение В 46](#_Toc136866905)

[Листинг метода скользящей средней 46](#_Toc136866906)

[Приложение Г 46](#_Toc136866905)

[Листинг метода расчёта ближайшей центроиды 46](#_Toc136866906)

[Приложение Д 46](#_Toc136866905)

[Листинг метода расчёта оптимальных параметров аппроксимации 46](#_Toc136866906)

[Приложение Е 46](#_Toc136866905)

[Листинг метода расчёта оптимальной длины окна скользящего среднего 46](#_Toc136866906)

**Введение**

В современной промышленной эксплуатации одной из критически важных задач является контроль за работой технологических устройств, в частности, горелочных устройств, использующих мазут в качестве топлива.

Определение количества работающих горелок котлоагрегата является важной задачей для обеспечения эффективной работы тепловой установки. Горелки играют ключевую роль в процессе сжигания топлива и преобразования энергии в тепло, которое затем передается носителю тепла. Правильное определение количества работающих горелок напрямую влияет на эффективность и безопасность работы котельной.

**Целью данной бакалаврской работы** является разработка программного решения, которое позволит определять количество работающих горелок на основе анализа паропроизводительности, расхода и давлении мазута. Внедрение такой системы способно значительно повысить контроль за процессом сжигания мазута, что, в свою очередь, позволит не только экономить топливо, но и повысить общую безопасность работы оборудования.

Актуальность исследования обусловлена необходимостью повышения эффективности производственных процессов и минимизации человеческого фактора в контроле за критически важными параметрами работы оборудования. Разработка такой программы дает возможность операторам получать достоверную информацию о состоянии горелок, что способствует более точной настройке процессов и уменьшению времени реагирования на аварийные ситуации.

# Общие положения об АСУ ТП и работе котлоагрегатов

АСУ ТП - это система, состоящая из персонала и совокупности оборудования с программным обеспечением, использующихся для автоматизации функций этого самого персонала по управлению промышленными объектами: электростанциями, котельными, насосными, водоочистными сооружениями, пищевыми, химическими, металлургическими заводами, нефтегазовыми объектами и т.д.

В данной главе будут рассмотрены общие положения о котлоагрегатах и режимной карте на основе которой в котлоагрегатах выставляют оптимальные параметры работы.

## Общие положения о котлоагрегатах и принципе их работы

Котельная – здание (в том числе блок-модульного типа) или комплекс зданий и сооружений с котельными установками и вспомогательным технологическим оборудованием, предназначенных для выработки тепловой энергии.

Котельная установка – котел (котлоагрегат) совместно с горелочными, топочными тягодутьевыми устройствами, механизмами для удаления продуктов горения и использования тепловой энергии уходящего мазута (газа) и оснащенный.

Принцип действия котлов на мазуте следующий:

1. Подача топлива в форсунку под давлением, в данном случае мазут
2. Распыление жидкого топлива на выходе с образованием мелких частиц
3. Частицы растворяются, смешиваются с воздухом и образуют горючую смесь
4. Тепло и горячий выхлоп нагревают воду в котловом баке
5. Циркуляционный насос перемешает разогретую воду по контуру

## Режимная карта и её роль в эксплуатации котлоагрегата

Режимная карта является важным инструментом в управлении и эксплуатации котлоагрегатов на тепловых электростанциях и в других промышленных установках. Это документ, составляемый организацией-подрядчиком по завершению пуско-наладочных и режимно-наладочных работ. Каждый котел должен иметь заполненную режимную карту и эксплуатироваться согласно ей.

Роль режимной карты в работе котлоагрегата:

1. Определение рабочих параметров.

Режимная карта устанавливает основные параметры работы котла, такие как давление, температура, скорость потока теплоносителя и другие критически важные показатели. Эти параметры определяются на основе проектных характеристик агрегата и его эксплуатационных возможностей.

1. Обеспечение безопасности.

Режимная карта помогает контролировать условия работы котлоагрегата таким образом, чтобы предотвратить аварийные ситуации. Она указывает на допустимые границы изменения параметров, выход за которые может привести к сбоям или несчастным случаям.

1. Оптимизация процессов.

Использование режимных карт позволяет оптимизировать процесс сгорания топлива, повышать тепловую эффективность и уменьшать потребление топлива и выбросы вредных веществ.

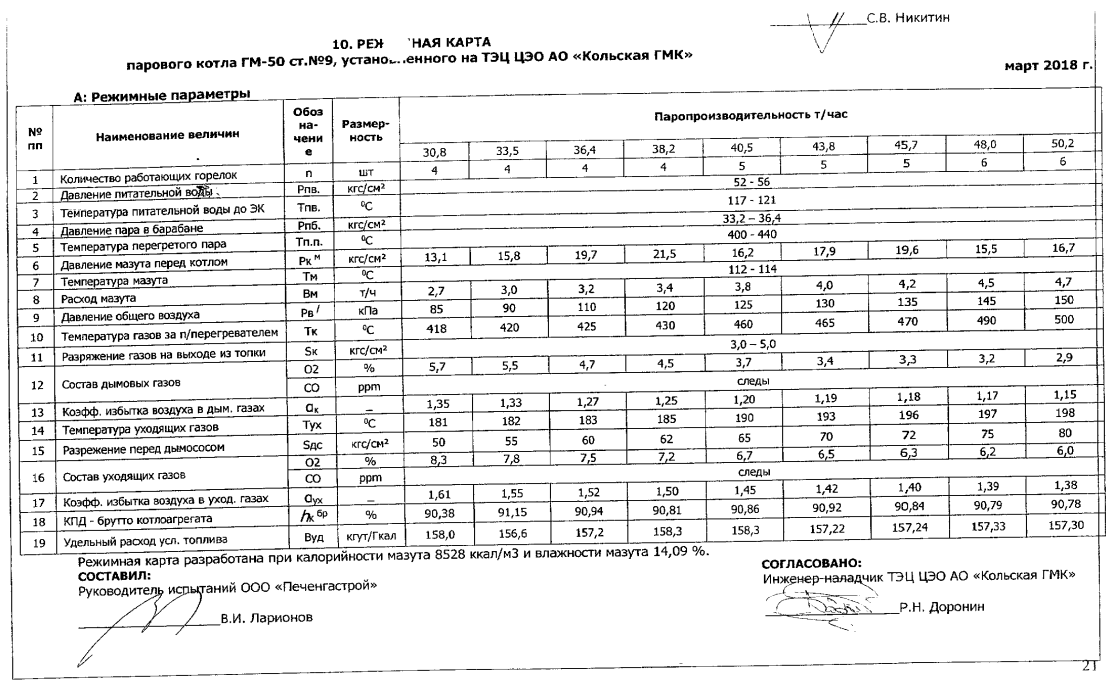
1. Контроль и мониторинг.

Режимная карта используется операторами и инженерами для мониторинга текущих условий работы котлоагрегата и его соответствия установленным нормам. Это позволяет своевременно выявлять отклонения и принимать меры для их корректировки.

Пример содержания режимной карты:

* Основные параметры работы: давление пара, температура пара, уровень воды в барабане.
* Режимы работы: номинальный, пониженный, повышенный.
* Лимиты и предупреждения: максимально и минимально допустимые значения параметров.
* Инструкции по регулировке: как реагировать на изменение параметров, в том числе настройки контрольно-измерительных приборов и автоматики.

В зависимости от типа и мощности котлоагрегата, а также от специфики производственного процесса, содержание режимной карты может значительно различаться, однако её основная цель — обеспечение безопасной, эффективной и стабильной работы остаётся неизменной. На рис.1.2.1 представлена режимная карта для котельной установки №9, предоставленная Кольской ГМК.



*Рисунок 1.2.1 – Режимная карта котельной установки №9 Кольской ГМК*

## Ключевые параметры работы в определении количества включенных горелок

Количество работающих горелок в котлоагрегате напрямую зависит от текущего давления, расхода мазута и требуемой тепловой мощности. Для определения оптимального количества работающих горелок будем учитывать следующие параметры:

1. Производительность пара.

Зависит от текущих потребностей в тепловой энергии. В периоды пиковых нагрузок может потребоваться задействование всех доступных горелок, тогда как в периоды низкой нагрузки часть горелок может быть отключена.

1. Расход мазута.

Чем выше расход мазута, тем больше горелок должно быть задействовано для обеспечения необходимой тепловой мощности.

1. Давление мазута.

При изменении давления мазута необходимо корректировать количество работающих горелок для поддержания стабильного процесса сгорания. При снижении давления мазута ухудшается распыление топлива, что приводит к образованию крупных капель и снижению эффективности сгорания. Для компенсации снижения эффективности сгорания может потребоваться увеличение количества работающих горелок. Повышение давления мазута способствует более тонкому распылению топлива, что улучшает процесс сгорания и повышает тепловую эффективность. При повышении давления мазута можно сократить количество работающих горелок без ущерба для тепловой мощности котлоагрегата.

# Математические методы в обработке и анализе данных работы котлоагрегата

В данной главе рассматриваются основные математические методы, используемые для обработки и анализа данных работы котлоагрегата. Эти методы позволяют повысить точность и надежность анализа, что в свою очередь способствует повышению эффективности и безопасности эксплуатации котлоагрегатов.

## 2.1. Интерполяция

Интерполяция — это метод математического анализа, используемый для нахождения промежуточных значений функции на основе известных значений. Проще говоря, интерполяция позволяет предсказать значения функции в точках, которые находятся между известными значениями. Этот метод широко используется в различных областях, таких как обработка сигналов, численные методы, компьютерная графика и многие другие.

Существует несколько методов интерполяции, каждый из которых имеет свои преимущества и недостатки. Рассмотрим 5 методов интерполяции.

1. Линейная интерполяция.

Самый простой и широко используемый метод. Предполагает, что промежуточные значения лежат на прямой линии, соединяющей два известных значения.

1. Полиномиальная интерполяция.

Использует полиномы для нахождения промежуточных значений. Более точная, чем линейная интерполяция, но может быть подвержена эффекту Рунге (осцилляции) при использовании высоких степеней полиномов.

1. Сплайн-интерполяция.

Использует кусочно-полиномиальные функции (сплайны) для интерполяции. Наиболее распространенный тип — кубические сплайны. Обеспечивает гладкость и непрерывность первой и второй производных.

1. Интерполяция методом ближайшего соседа.

Самый простой метод, при котором значение в неизвестной точке принимается равным значению ближайшей известной точки. Подходит для дискретных данных, но не обеспечивает гладкости.

1. Интерполяция методом Лагранжа.

Использует полиномы Лагранжа для нахождения промежуточных значений. Подходит для небольшого количества точек, так как вычислительная сложность растет с увеличением числа точек.

В нашей работе для заполнения пропущенных значений в данных используется линейная интерполяция:

Где и – значения функции в точках и , y – неизвестное значение функции в известной нам точке x.

Важно отметить, что линейная интерполяция применяется для всех параметров работы котлоагрегата отдельно. В условиях большого объема данных, даже если метод интерполяции не идеален для отдельных точек, его применение позволяет избежать искажений, связанных с нулевыми значениями, и поддерживать целостность и непрерывность графиков, что критически важно для точного анализа и визуализации параметров работы котлоагрегата. Код метода линейной интерполяции приведён в приложении Б.

## 2.2. Аппроксимация

Аппроксимация — это метод математического анализа, используемый для нахождения приближенных значений функции или данных на основе известных значений. В отличие от интерполяции, которая стремится точно восстановить промежуточные значения, аппроксимация направлена на нахождение функции, которая наилучшим образом описывает набор данных или поведение системы. Аппроксимация широко используется в численных методах, машинном обучении, обработке сигналов и других областях.

Существует несколько методов аппроксимации, каждый из которых имеет свои преимущества и недостатки. Рассмотрим некоторые из них.

1. Линейная аппроксимация.

Самый простой метод аппроксимации. Использует линейную функцию для приближения данных.

1. Полиномиальная аппроксимация.

Использует полиномы для приближения данных. Более гибкая, чем линейная аппроксимация, так как позволяет учитывать нелинейные зависимости.

1. Аппроксимация сплайнами.

Использует кусочно-полиномиальные функции (сплайны) для аппроксимации данных. Наиболее распространенный тип — кубические сплайны. Обеспечивает гладкость и непрерывность первой и второй производных.

1. Экспоненциальная аппроксимация.

Использует экспоненциальные функции для приближения данных. Подходит для данных, которые демонстрируют экспоненциальный рост или спад.

1. Аппроксимация методом наименьших квадратов.

Общий метод, используемый для нахождения коэффициентов аппроксимирующей функции. Минимизирует сумму квадратов отклонений между аппроксимирующей функцией и данными.

Важной особенностью аппроксимации, которая важна в нашей работе, является сглаживание данных. В исходных данных часто присутствуют шумы и выбросы, которые могут затруднять анализ. Аппроксимация помогает сгладить данные, убирая случайные колебания и выделяя основные тенденции.

В нашей работе для упрощения графика используется полиномиальная аппроксимация, предоставленная математической библиотекой apache на языке java. Код приведён в приложении Б.

Алгоритм полиномиальной аппроксимации:

1. Сбор данных.

Сначала собираются данные, которые нужно аппроксимировать. В нашем случае это параметр работы котлоагрегата и время. Так как аппроксимация предполагает работу с числами, а не с объектами, то возникла необходимость преобразования объекта LocaDateTime в число. Для этого используется метод для пересчёта текущей даты и времени в минуты, относительно 1 января 1970 года.

1. Формирование системы уравнений.

На основе данных формируется система линейных уравнений, которая выражает сумму квадратов отклонений.

1. Решение системы уравнений.

Система уравнений решается для нахождения коэффициентов полинома.

1. Применение полинома.

Найденные коэффициенты используются для построения полинома, который затем применяется для аппроксимации данных.

Полиномиальная аппроксимация методом наименьших квадратов является мощным инструментом для анализа данных о производительности котлоагрегата. Она позволяет сгладить данные и выявить основные тренды, что может быть полезно для дальнейшего анализа и принятия решений.

## Метод скользящего среднего

Метод «скользящего среднего» (Simple Moving Average, SMA) — это статистический инструмент, используемый для анализа временных рядов путём создания серии средних значений из различных подмножеств полного набора данных. Суть метода заключается в вычислении среднего значения набора данных за определенный период времени, который "скользит" по мере добавления новых данных. Этот метод широко применяется в финансах, экономике, метеорологии и инженерии для сглаживания краткосрочных колебаний и выделения долгосрочных тенденций из временных рядов.

Формула простого скользящего среднего для точки :

где ​ — текущее значение, а n — количество точек с каждой стороны от текущей.

Скользящее среднее помогает уменьшить "шум" в данных, вызванный случайными колебаниями, и облегчает восприятие основных тенденций и моделей. Метод прост в реализации и интерпретации, что делает его доступным для широкого круга пользователей, не обладающих специальными техническими знаниями. Код приведён в приложении В.

## 2.4. Применение кластеризации в контексте определения режима работы котлоагрегата

Кластеризация — это метод анализа данных, который используется для группировки объектов в кластеры, так чтобы объекты внутри одного кластера были более похожи друг на друга, чем на объекты из других кластеров. Это один из ключевых методов машинного обучения и анализа данных, который находит широкое применение в различных областях, таких как маркетинг, биоинформатика, обработка изображений и многие другие.

Рассмотрим основные методы кластеризации.

1. K-means

Один из самых популярных методов кластеризации. Алгоритм делит данные на k кластеров, минимизируя сумму квадратов расстояний между объектами и центроидами кластеров. Требует заранее заданного числа кластеров k.

1. Иерархическая кластеризация

Делится на агломеративную и дивизивную. В агломеративной кластеризации каждый объект начинается как отдельный кластер, и кластеры последовательно объединяются. В дивизивной кластеризации все объекты начинаются в одном кластере, который затем последовательно делится.

1. DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

Основан на плотности данных. Может выявлять кластеры произвольной формы и справляться с шумом (выбросами). Не требует заранее заданного числа кластеров.

1. Gaussian Mixture Models (GMM)

Предполагает, что данные генерируются из смеси нескольких гауссовых распределений. Использует метод максимального правдоподобия для оценки параметров распределений.

В нашем случае, когда центроиды уже заданы, мы используем метод, который можно назвать "кластеризация с фиксированными центроидами". Это действительно можно рассматривать как форму кластеризации, хотя и с некоторыми особенностями. В отличие от традиционных методов, где центроиды определяются в процессе кластеризации, здесь они заданы заранее.

Применение в контексте котлоагрегата заключается в том, чтобы определить режим работы в каждый момент времени. Принцип работы:

1. Определение центроид

Режимная карта содержит заранее определенные режимы работы котлоагрегата, которые можно рассматривать как центроиды кластеров.

1. Вычисление расстояние до центроид

Считаем евклидово расстояние параметров работы в определённый момент времени относительно параметров работы котлоагрегата режимной карты по формуле:

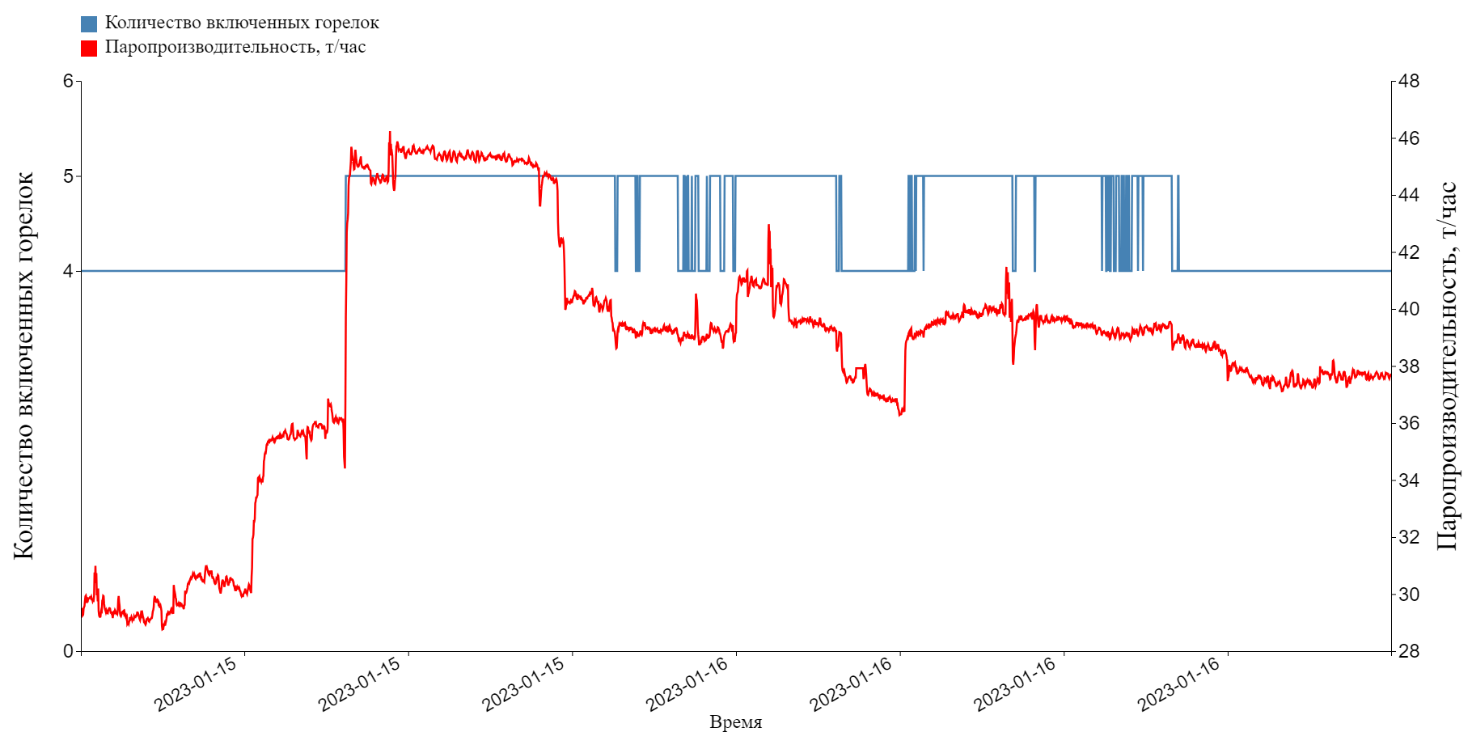
Где A, B, C – рекомендуемые параметры работы котлоагрегата для производительности пара, давления мазута и расхода мазута соответственно, а a, b, c – параметры работы котлоагрегата в текущий момент времени для производительности пара, давления мазута и расхода мазута соответственно.

1. Определение ближайшей центроиды

Режим с минимальным расстоянием считается текущим режимом работы. Код определения режима работы в определённый момент времени приведён в приложении Г.

## Определение оптимального подхода в сглаживании исходных графиков

Вопрос о включении или выключении горелки на производственном предприятии, таком как Кольская ГМК, обычно решается на основе ряда факторов, включая производственные потребности, техническое состояние оборудования, требования безопасности и экологические нормы. В общем случае, частота включения и выключения горелок может варьироваться от нескольких раз в час до нескольких раз в день, в зависимости от упомянутых факторов. На рис.2.5.1 мы можем увидеть, как выглядит график изменения производительности пара и включения, выключения горелок до применения сглаживания.



*Рисунок 2.5.1 – График изменения количества горелок и производительности пара до применения сглаживания*

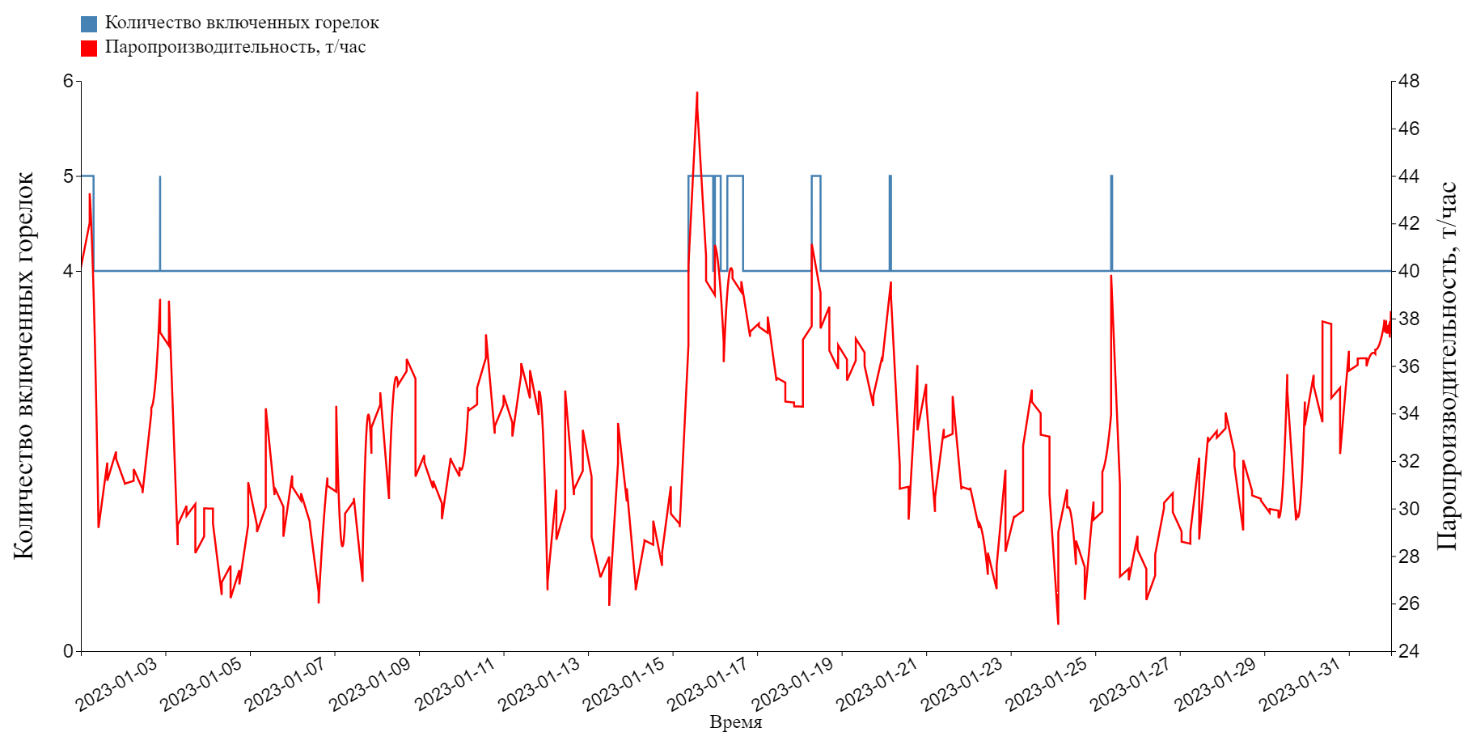
Частота включения и выключения пятой горелки слишком большая. Связано это с тем, что в течение перехода с одного режима на другой параметры работы изменяются не последовательно, происходят скачки в изменении параметров работы, которые могут быть вызваны, как и котлоагрегатом, так и неточностью работы датчика.С целью минимизации неоднозначных участков частого включения или выключения горелок были рассмотрены два подхода в сглаживании исходных графиков. В рамках исследования были рассмотрены полиномиальная аппроксимация и метод скользящего среднего. Задачей исследования было подобрать оптимальный метод с такими параметрами работы, при которых количество неоднозначных участков было бы как можно меньше.

Полиномиальная аппроксимация была протестирована для различных степеней полинома (от 2 до 5) и различной длины промежутков аппроксимации (от 50 до 500 с шагом 50). Результаты тестирования отражены в таблице 1. Код метода тестирования приведён в приложении Д.

*Таблица 1 – Результаты тестирования аппроксимаций*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | Степень полинома | | | |
| 2 | 3 | 4 | 5 |
| Длина промежутка аппроксимации | 50 | 232 | 207 | 214 | 208 |
| 100 | 138 | 139 | 132 | 127 |
| 150 | 111 | 105 | 89 | 96 |
| 250 | 66 | 65 | 63 | 66 |
| 300 | 57 | 56 | 45 | 79 |
| 350 | 41 | 42 | 40 | 41 |
| 400 | 48 | 47 | 44 | 54 |
| 450 | 35 | 47 | 56 | 36 |
| 500 | 35 | 77 | 27 | 31 |

На рис.2.5.2 приведён пример как выглядит график с полиномиальной аппроксимацией 4 степени и длиной промежутка в 500 значений.



*Рисунок 2.5.2 – График изменения количества горелок и производительности пара после применения полиномиальной аппроксимации*

Разбиение исходных данных на промежутки для аппроксимации делается по причине того, что исходные данные могут быть довольно большими, например, данные за 1 год снятые поминутно будут содержать более 500000 позиций. Найти аппроксимирующую функцию для такого количества точек не предоставляется возможным.

Результаты показали, что количество неоднозначных участков уменьшается с увеличением степени полинома и размера пропуска данных. Однако, даже при высоких степенях полинома и больших размерах пропусков, количество неоднозначных участков оставалось значительным. Например, при размере пропуска 500 и степени полинома 4 количество неоднозначных участков составило 27, что все еще является высоким значением.

Метод скользящего среднего был протестирован на тех же данных, что и полиномиальная аппроксимация. Результаты тестирования представлены на рис.2.5.2. Код метода тестирования приведён в приложении Е.

*Рисунок 2.5.2 – Зависимость количества неоднозначных участков от длины окна*

Метод скользящего среднего был протестирован для различных размеров окна. Результаты показали, что количество неоднозначных участков значительно уменьшается с увеличением размера окна. Например, при размере окна 20 количество неоднозначных участков составило 133, а при размере окна 100 — всего 21. При дальнейшем увеличении размера окна количество неоднозначных участков продолжало уменьшаться, но незначительно.

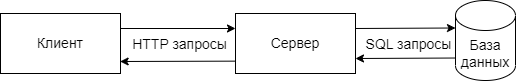
Сравнение двух подходов показало, что метод скользящего среднего превосходит полиномиальную аппроксимацию по способности уменьшать количество неоднозначных участков. Полиномиальная аппроксимация, несмотря на свою гибкость и возможность точного подбора степени полинома, не смогла обеспечить достаточное сглаживание данных для минимизации неоднозначных участков. В то время как метод скользящего среднего, благодаря своей простоте и эффективности, показал лучшие результаты при меньших вычислительных затратах.

На основании проведенного анализа и тестирования было принято решение использовать метод скользящего среднего для сглаживания исходных графиков паропроизводительности, давления и расхода мазута с длиной окна в 120 минут. Этот метод продемонстрировал наилучшие результаты в уменьшении количества неоднозначных участков, что позволяет более точно и надежно анализировать данные и определять принадлежность текущих точек к определенным режимам работы.

Таким образом, метод скользящего среднего был выбран в качестве основного инструмента для сглаживания данных в рамках данного проекта, обеспечивая высокую точность и надежность анализа.

# Реализация программы

Результатом написания программного решения является полноценное клиент-серверное приложение. Клиент-серверная архитектура представляет собой модель взаимодействия между двумя основными компонентами: клиентом и сервером. В этой модели клиент запрашивает ресурсы или услуги, а сервер предоставляет их. Принцип работы приведён на рис.3.1.



*Рисунок 3.1 – Принципы работы клиент-серверной архитектуры*

Особенности клиент-серверной архитектуры заключаются в следующем:

1. Запрос-ответ. Клиент отправляет запрос серверу, используя определенный протокол (например, HTTP/HTTPS для веб-приложений). Сервер обрабатывает запрос, выполняет необходимые действия (например, извлечение данных из базы данных) и отправляет ответ клиенту. Этот цикл запрос-ответ является основой взаимодействия в клиент-серверной архитектуре.

2. Разделение обязанностей. Клиент и сервер имеют четко определенные роли. Клиент отвечает за взаимодействие с пользователем и отправку запросов, а сервер — за обработку запросов и предоставление ресурсов. Такое разделение позволяет улучшить масштабируемость и управляемость системы, так как каждый компонент может быть разработан, развернут и масштабирован независимо.

3. Масштабируемость. Клиент-серверная архитектура позволяет легко масштабировать систему. Например, можно добавить больше серверов для обработки увеличенного числа запросов или использовать балансировщики нагрузки для распределения трафика. Масштабируемость также достигается за счет возможности горизонтального масштабирования, когда добавляются новые серверы, и вертикального масштабирования, когда увеличиваются ресурсы существующих серверов.

## Алгоритм определения количества работающих горелок

Исходные данные собираются с котлоагрегата с интервалом в одну минуту. Эти данные включают в себя довольно много параметров, в том числе интересующие нас: производительность пара, давление мазута и расход мазута. Использование трех параметров — паропроизводительности, давления и расхода мазута — в ннашей работе обусловлено их фундаментальной значимостью для оценки и управления режимами работы котлоагрегата. Вот почему эти параметры являются основополагающими: Данные представлены в виде таблицы, где каждая строка соответствует одному моменту времени. Пример продемонстрирован в таблице 2.

*Таблица 2 – Пример исходных данных*

|  |  |  |  |
| --- | --- | --- | --- |
| Время | Давление мазута на котел №9, кг/ | Расход мазута в котёл №9, т/ч | Расход перегретого пара от котла №9 (факт.), т/ч |
| 01.01.2023 | 19,216 | 4,066 | 44,336 |
| 01.01.2023 0:01 | 19,219 | 4,066 | 44,215 |
| 01.01.2023 0:02 | 19,216 | 4,066 | 44,061 |
| 01.01.2023 0:03 | 19,216 | 4,067 | 43,958 |
| 01.01.2023 0:04 | 19,216 | 4,065 | 43,865 |
| 01.01.2023 0:05 | 19,216 | 4,066 | 43,799 |
| … | … | … | … |

Алгоритм определения количества работающих горелок состоит из следующих этапов:

1. Восстановление пропущенных и некорректных значений. В данных могут встречаться пропущенные значения (NULL), как для одного из параметров, так и для всех. Пример приведён в таблице 3.

*Таблица 3 – Примеры пропущенных значений*

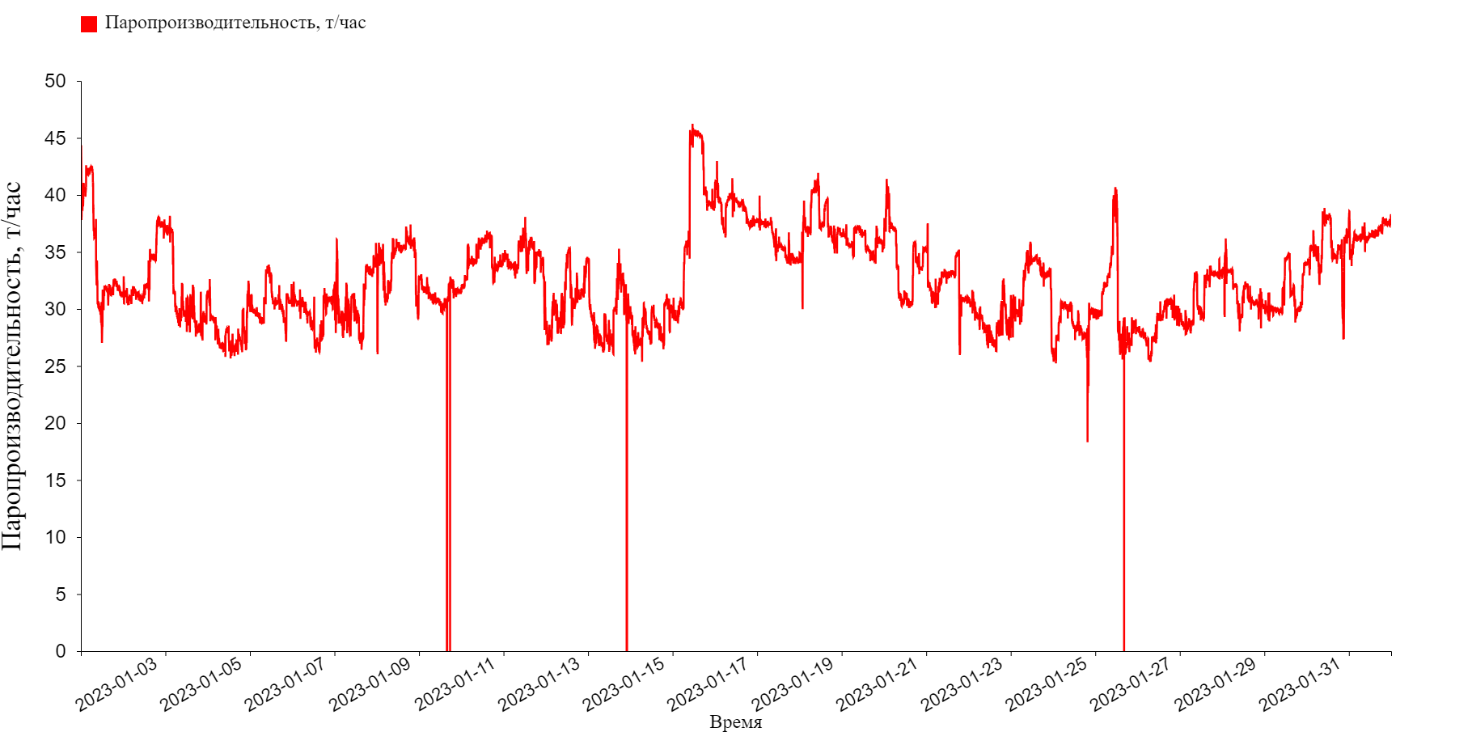
|  |  |  |  |
| --- | --- | --- | --- |
| Время | Давление мазута на котел №9, кг/ | Расход мазута в котёл №9, т/ч | Расход перегретого пара от котла №9 (факт.), т/ч |
| 01.01.2023 | 19,216 | 4,066 | 44,336 |
| 09.01.2023 17:39 | 16,837 | 2,978 | 32,181 |
| 09.01.2023 17:40 | Null | Null | Null |
| 09.01.2023 17:41 | 16,837 | 2,978 | 32,119 |

Также могут встречаться значения, равные 0.0, в те моменты, когда котлоагрегат функционирует на определённых мощностях и можно точно сказать, что в эти моменты времени датчики некорректно сняли параметры работы. Пример приведён в таблице 4.

*Таблица 4 – Примеры некорректной работы датчиков*

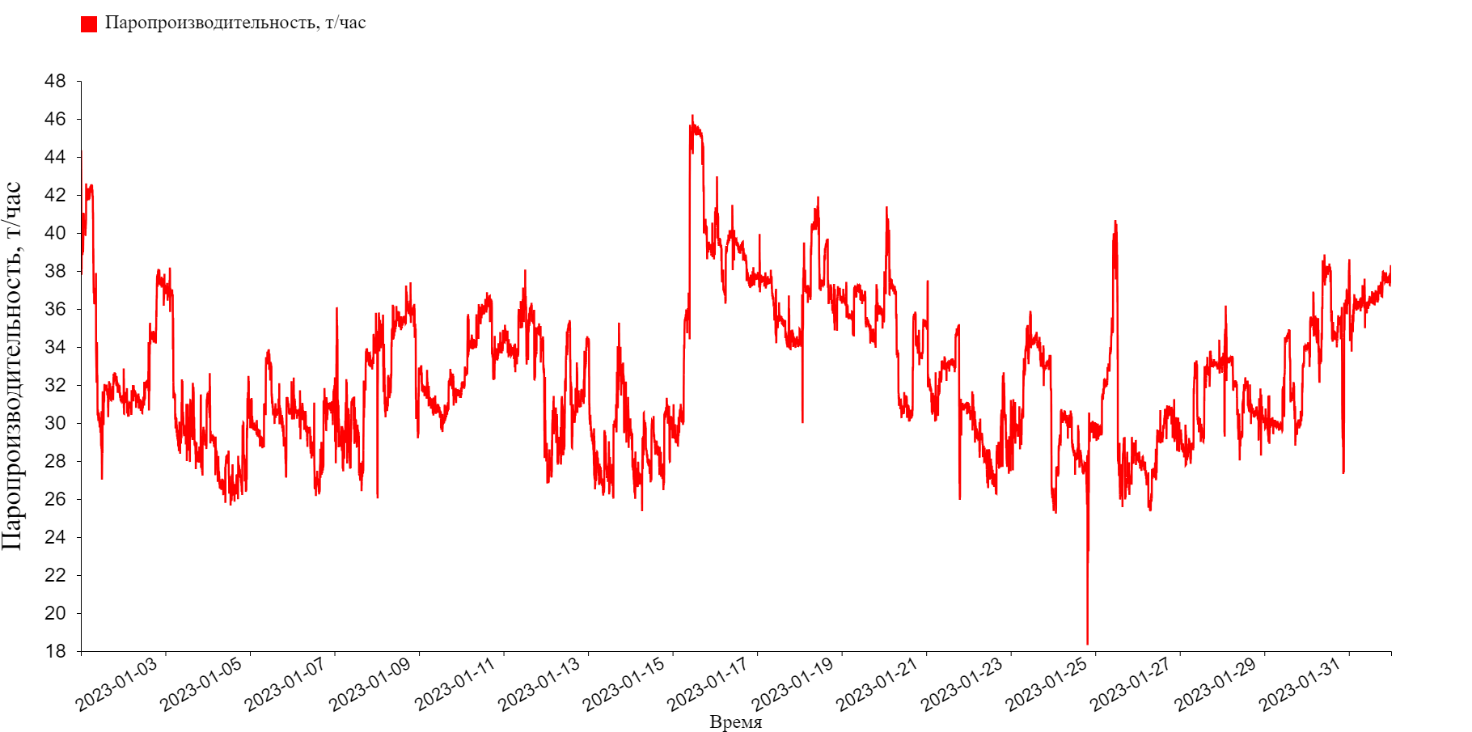
|  |  |  |  |
| --- | --- | --- | --- |
| Время | Давление мазута на котел №9, кг/ | Расход мазута в котёл №9, т/ч | Расход перегретого пара от котла №9 (факт.), т/ч |
| 13.01.2023 21:52 | 15,1 | 2,8 | 30,028 |
| 13.01.2023 21:53 | 15,097 | 2,801 | 30,227 |
| 13.01.2023 21:54 | 15,094 | 0 | 0 |
| 13.01.2023 21:55 | 15,087 | 2,801 | 30,563 |
| 13.01.2023 21:56 | 0 | 0 | 30,593 |
| 13.01.2023 21:57 | 15,087 | 2,8 | 30,609 |

В эти моменты времени график резко принимает значение 0. На рис.3.1.1 приведён пример с производительностью пара до применения интерполяции.



*Рисунок 3.1.1 – Производительность пара до применения интерполяции*

Как мы видим, котлоаграгат на протяжении всего января работал на определённых мощностях, от 25 до 45 тонн в час. Однако в моменты некорректной работы датчиков он резко принимает значение ноль. Эти значения необходимо обработать, чтобы они не искажали дальнейший анализ. При работе с большими объемами данных, даже если метод интерполяции не всегда точен для отдельных точек, его использование помогает избежать искажений, вызванных нулевыми значениями. Это также позволяет поддерживать целостность и непрерывность графиков, что крайне важно для точного анализа и визуализации параметров работы котлоагрегата. Пропущенные и некорректные значения заполняются с помощью линейной интерполяции. Интерполяция позволяет плавно восстановить недостающие данные, основываясь на соседних значениях. Результат применения интерполяции приведён на рис.3.1.2.

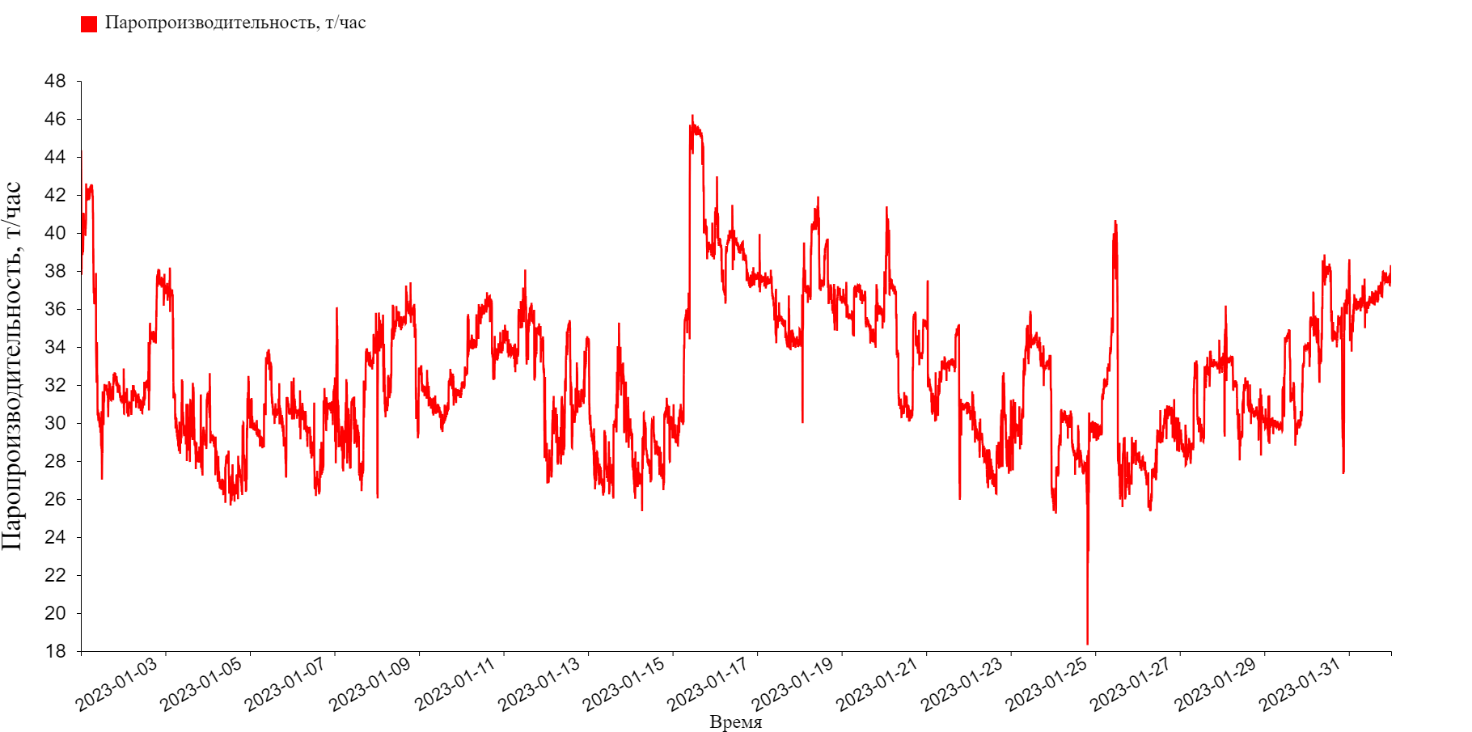


*Рисунок 3.1.2 – Производительность пара после применения интерполяции*

Как мы видим значения некорректно снятые датчиками исчезли. Это помогает компенсировать сбои в работе датчиков и получить более полные и корректные данные. Интерполяция применяется для каждого параметра работы котлоагрегата отдельно. В данном случае приведён пример с производительностью пара.

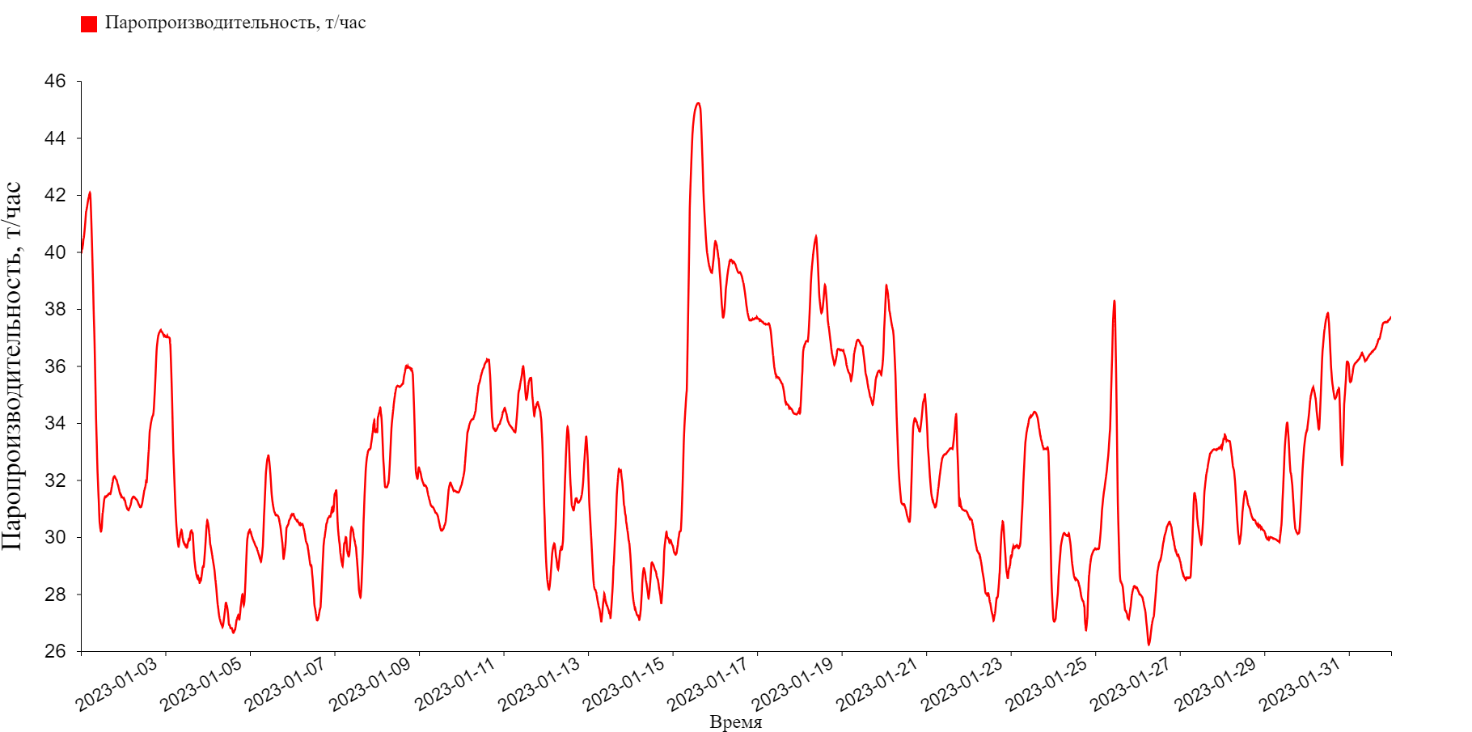
2. Сглаживание данных:

Данные работы котлоагрегата собираются с довольно высокой частотой, а именно раз в минуту, поэтому даже небольшие колебания параметров будут видны на графике. Это может создавать видимость "кардиограммы". После интерполяции данные будут содержать резкие скачки, которые негативно влияют на результаты анализа. Пример представлен на рис.3.1.3.



*Рисунок 3.1.3 – Производительность пара до применения скользящего среднего*

Для сглаживания данных используется метод скользящего среднего – оптимальный метод для получения наименьшего количества неоднозначностей, выявленный по результатам исследования. Скользящее среднее с окном в 120 минут вычисляется для каждого параметра работы котлоагрегата отдельно. Результат применения скользящего среднего на производительности пара представлен на рис.3.1.4.

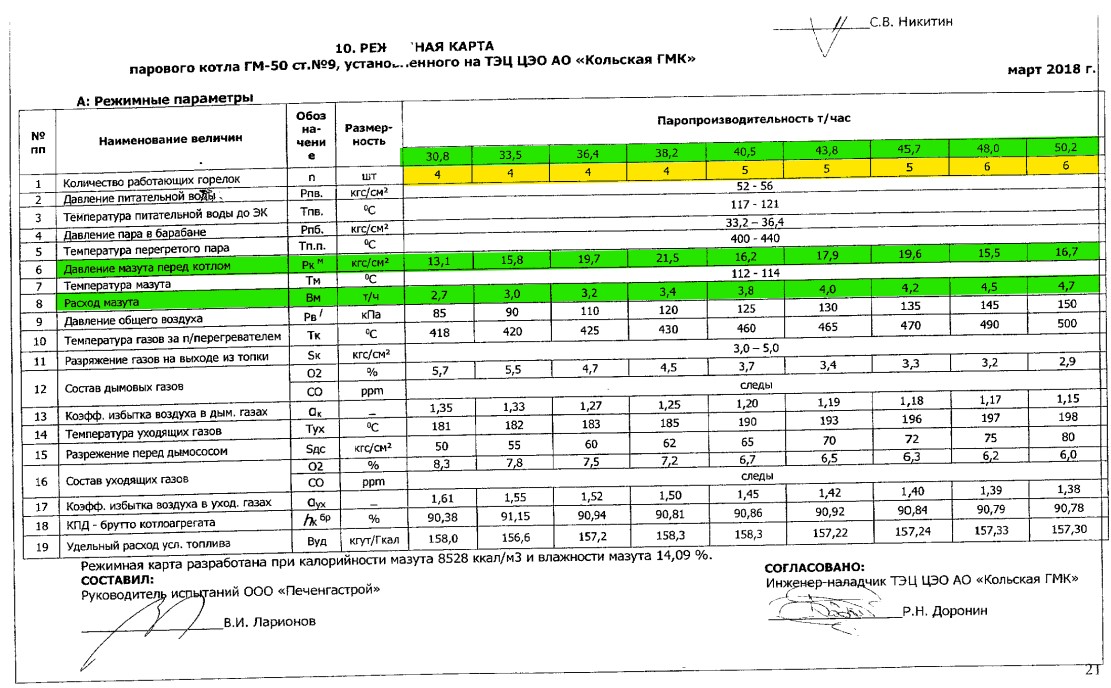


*Рисунок 3.1.4 – Производительность пара после применения скользящего среднего*

Как мы видим, скользящее среднее позволяет уменьшить влияние случайных колебаний и получить более стабильные значения параметров.

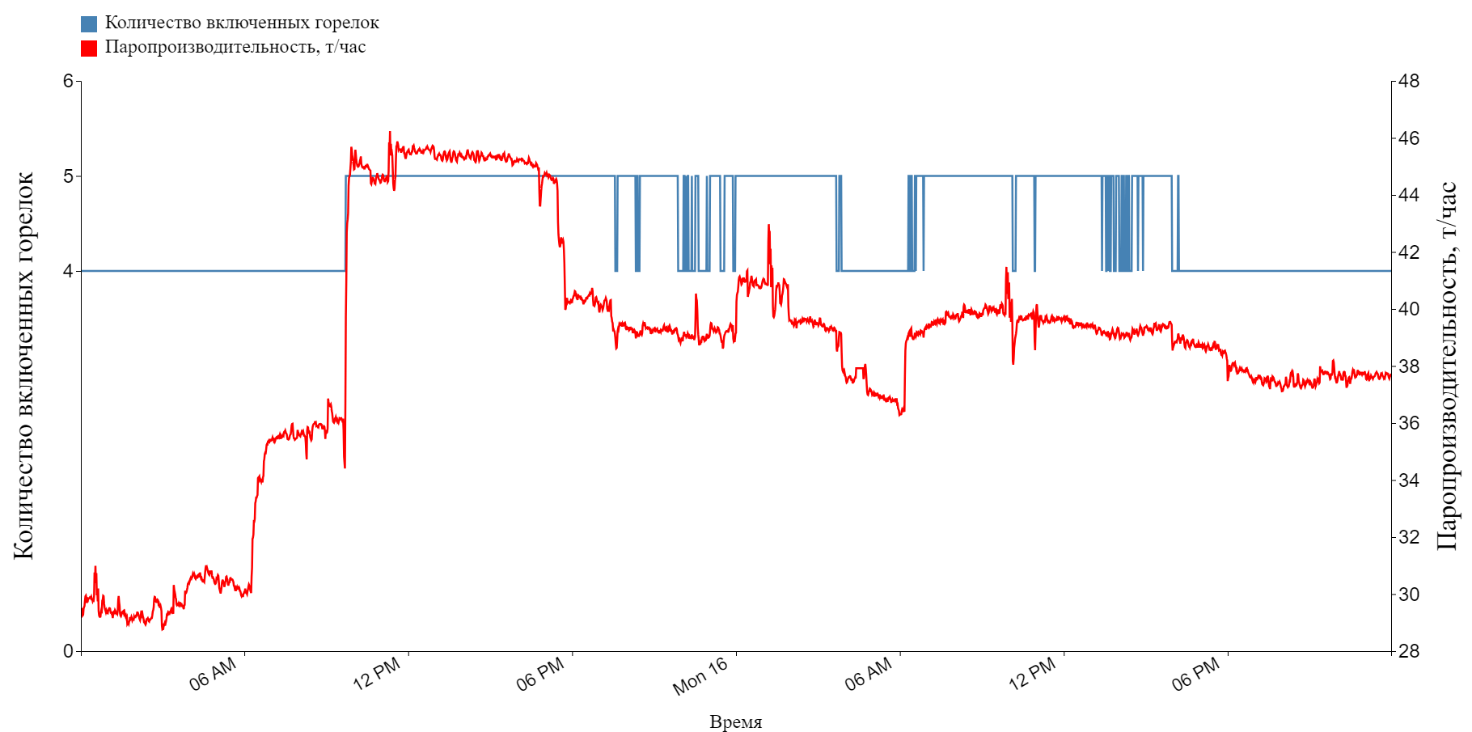
3. Определение текущего режима работы:

После сглаживания данных необходимо определить текущий режим работы котлоагрегата. Для этого используется кластеризация с заранее заданными центроидами. Режимная карта содержит заранее определенные режимы работы котлоагрегата, которые можно рассматривать как центроиды кластеров. На рис.3.1.5 представлена режимная карта, где для удобства зелёным цветом выделены интересующие нас параметры – производительность пара, давление и расход мазута, жёлтым – количество включенных горелок.



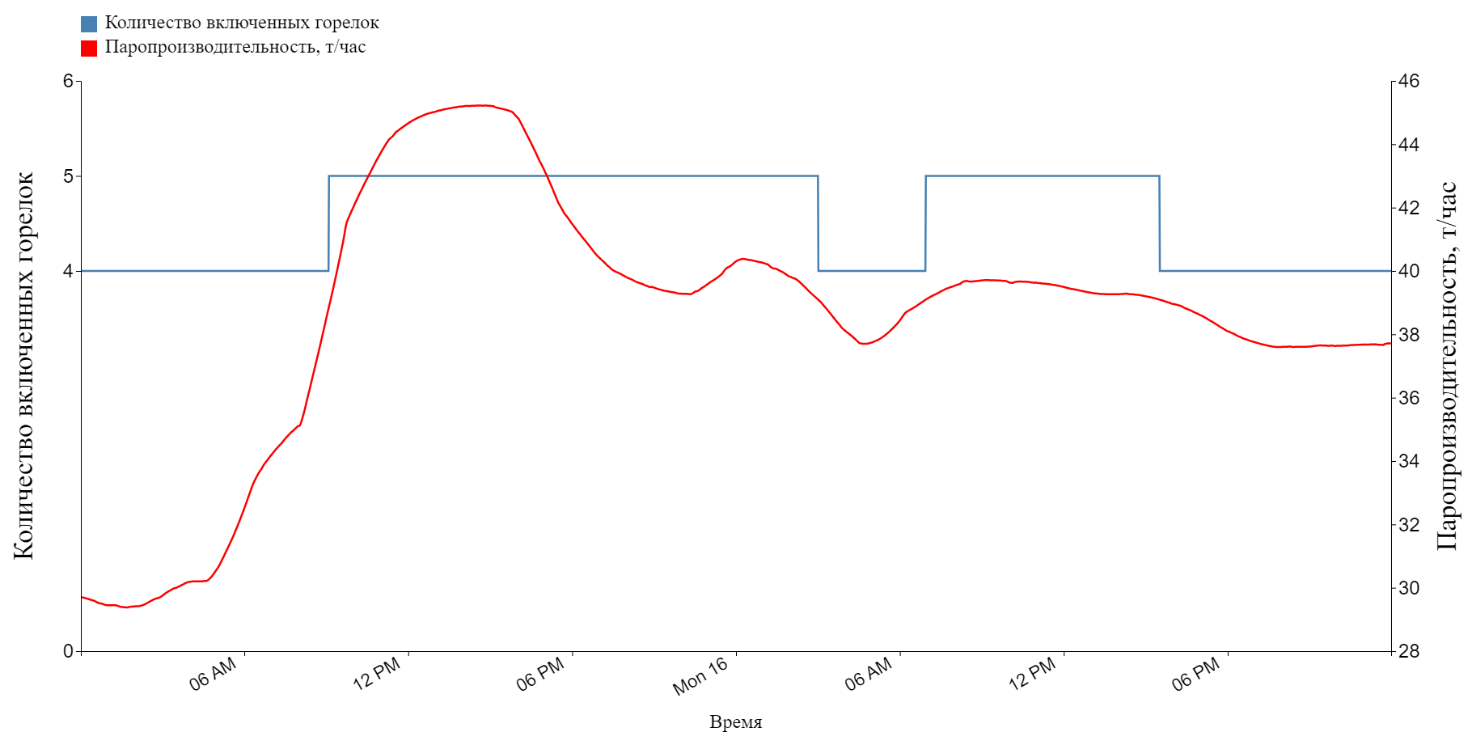
*Рисунок 3.1.5 – Режимная карта котельной установки №9 Кольской ГМК*

Каждый режим характеризуется определенными значениями паропроизводительности, давления мазута, расхода мазута и количеством включенных горелок. Для каждой позиции исходных данных считается евклидово расстояние до каждого из режимов. Режим с минимальным евклидовым расстоянием считается текущим режимом работы котлоагрегата. На рис.3.1.6 представлены результаты расчёта количества работающих горелок для котлоагрегата №9 Кольской ГМК до применения сглаживания.



*Рисунок 3.1.6 – Р*езультаты расчёта количества работающих горелок для котлоагрегата до применения сглаживания

Как можно заметить, на синем графике присутствуют неоднозначные участки, когда тот или иной режим работы включается или выключается слишком часто. В реальности это маловероятно. На данном же графике смена режимов работы местами осуществляется меньше чем 10 минут. Именно с этой целью делаем наши исходные графики производительности пара, давлении и расхода мазута более гладким, однозначными. На рис.3.1.7 представлены результаты расчёта количества работающих горелок после применения скользящего среднего.

****

*Рисунок 3.1.7 – Р*езультаты расчёта количества работающих горелок для котлоагрегата после применения сглаживания

Как можно заметить, синий график принял более однозначный вид в тех местах, которые вызывали сомнения.

## Серверная часть

Сервер — это программа, которая обрабатывает запросы от клиентов и предоставляет необходимые ресурсы или услуги. Серверы могут выполнять различные функции, такие как хранение данных, обработка бизнес-логики, аутентификация пользователей и многое другое. В данной работе для написания серверной части использовались язык программирования Java и библиотека Spring Framework.

Java - это объектно-ориентированный язык программирования, широко используемый для разработки серверных приложений. Spring Framework - это мощная библиотека для разработки Java-приложений. Она предоставляет комплексные решения для создания веб-приложений. Для сохранения данных использовалась объектно-реляционная система управления базами данных PostgreSQL.

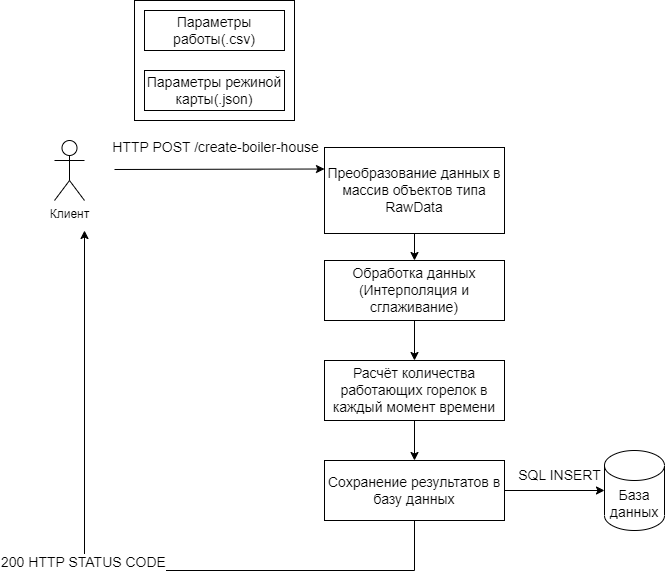
На рис.3.2.1 представлена диаграмма классов, используемых в работе.



*Рисунок 3.2.1 – Диаграмма классов*

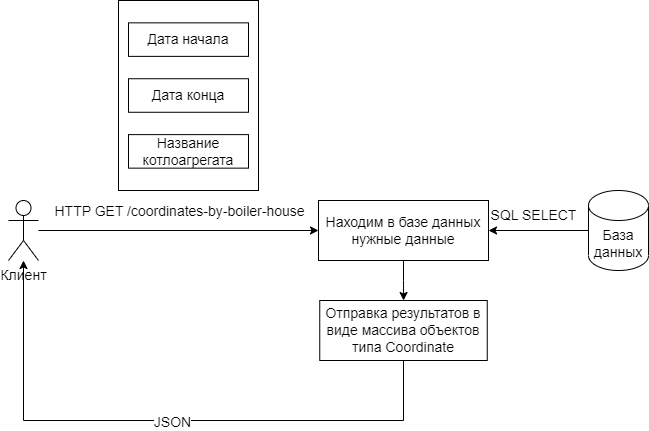
* LocalDateTime – класс для сохранения даты и времени. Встроен в Java по умолчанию.
* RawData – пользовательский класс. Один объект этого класса содержит данные параметров работы котлоагрегата в определённый момент времени.
* Coordinate – пользовательский класс для сохранения результатов вычисления количества работающих горелок.
* BoilerHouse – пользовательский класс, котельная установка.

Эти классы являются основополагающими в данной работе, так как они обеспечивают основную функциональность и структуру системы. Каждый класс был разработан с учетом принципов объектно-ориентированного программирования, что позволяет достичь высокой степени модульности, повторного использования кода и легкости сопровождения. Диаграмма классов наглядно демонстрирует взаимосвязи между различными компонентами системы, что способствует лучшему пониманию архитектуры и логики работы приложения. На рис.3.2.2 представлен алгоритм сохранения котлоагрегата клиентом.



*Рисунок 3.2.2 – Алгоритм сохранения котлоагрегата*

Клиент отправляет HTTP POST запрос на метод сервера /create-boiler-house, к которому прикрепляет csv файл с данными работы котлоагрегата за определённый промежуток времени и параметры режимной карты в формате json. После этого сохранения клиент в любой момент может открыть интересующий его котлоагрегат и посмотреть результаты вычислений. На рис3.2.3 продемонстрирована диаграмма получения результатов вычислений со стороны клиента.



*Рисунок 3.2.3 –* Алгоритм получения результатов вычислений

Как было сказано ранее, клиент может в любой момент посмотреть результаты вычислений, сохранённых котлоагрегатов. Это позволяет в любой момент времени получить доступ к историческим данным. Получение данных реализуется с помощью HTTP GET запроса на метод сервера /coordinates-by-boiler-house. Полностью со всеми параметрами запрос выглядит следующим образом - http://localhost:8080/coordinates-by-boiler-house?dateFrom=2023-01-01 00:01&dateTo=2023-01-31 23:59&name=9 котёл. Здесь мы в параметрах запроса указываем даты начала и конца временного интервала и название ранее сохранённого котлоагрегата. На клиентскую часть приходят данные в формате JSON, который представляет из себя текстовый формат с неупорядоченным множеством пар ключ-значение. Пример на рис.3.2.4.



*Рисунок 3.2.4 –* Данные приходящие с сервера

Каждая позиция содержит информацию о работе котлоагрегата в определённый момент времени. В поля которые содержит одна позиция входят:

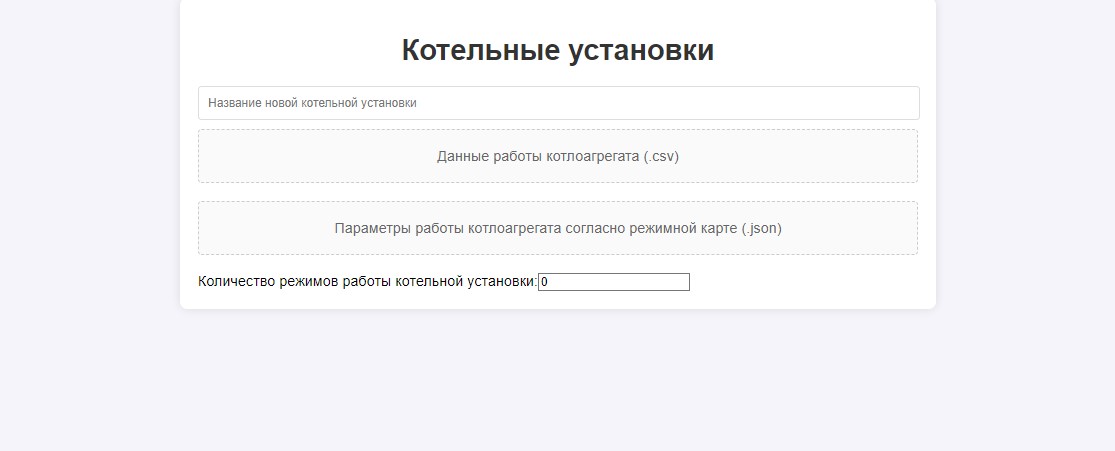
* time: Время, когда были зафиксированы данные. Формат времени соответствует стандарту ISO 8601.
* burnersNum: Количество работающих горелок в данный момент времени.
* steamCapacity: Паропроизводительность котлоагрегата в данный момент времени, измеренная в соответствующих единицах (например, тонны пара в час).

## Клиентская часть

Клиент — это устройство или программа, которая инициирует запросы к серверу. Клиенты могут быть различными: от веб-браузеров и мобильных приложений до специализированных программных решений. Клиенты обычно взаимодействуют с пользователем, предоставляя интерфейс для ввода данных и отображения результатов. В данной работе для написания клиентской части использовались язык программирования JavaScript и библиотека React.

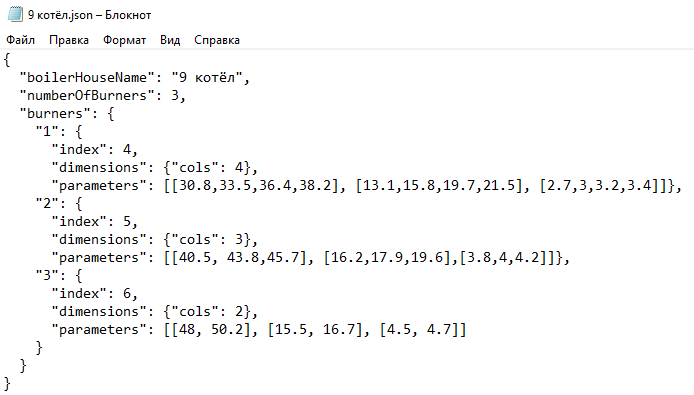
JavaScript - это язык программирования, используемый для создания интерактивных веб-страниц. React - это библиотека для создания пользовательских интерфейсов.

Первое, что видит пользователь – главное меню. Пример на рис.3.3.1.



*Рисунок 3.3.1 – Главное меню*

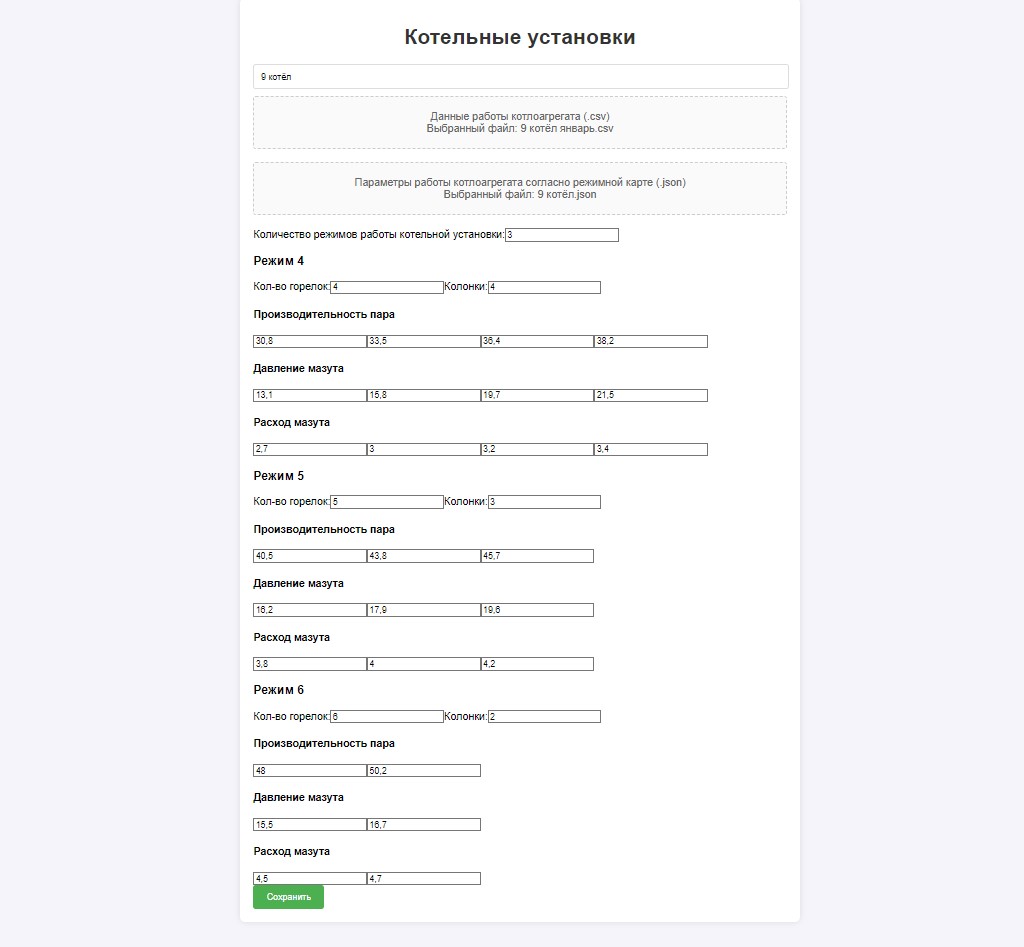
Здесь отображаются сохранённые ранее данные о работе котлоагрегатов и формы для загрузки файлов с целью сохранения новых. Чтобы сохранить новые данные, надо выбрать csv файл с данными работы за определённый временной интервал и json файл с параметрами работы, взятыми из режимной карты. Также требуется задать имя новой записи. Пример того как должен выглядеть файл представлен на рис.3.3.2.



*Рисунок 3.3.2 – Параметры работы котлоагрегата по режимной карте*

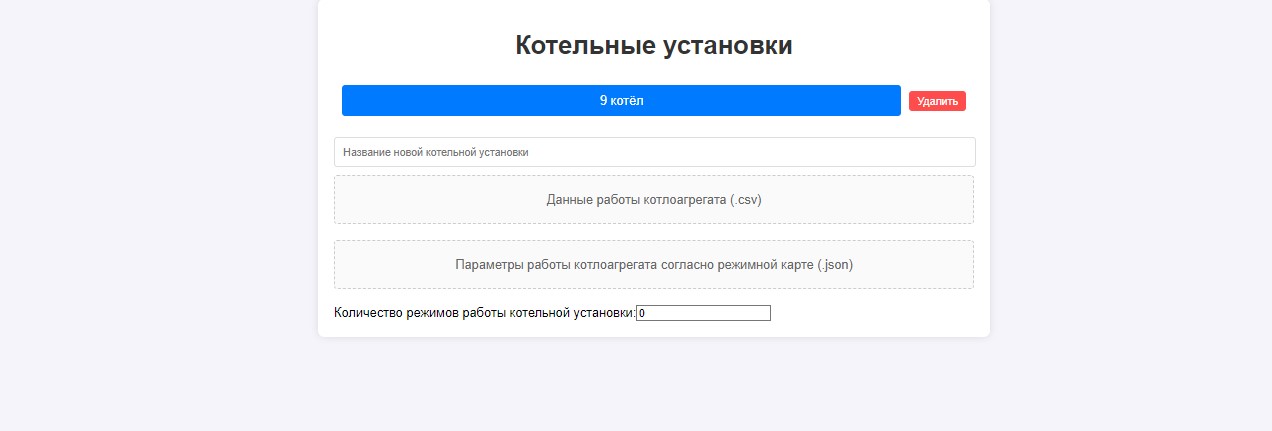
После загрузки файла параметров, данные из файла автоматически заполняют соответствующие поля в интерфейсе. Пользователь может изменить количество режимов работы, количество горелок для каждого из этих режимов, а также сами значения параметров производительности пара, давлении или расхода мазута. Этот функционал позволяет гибко настраивать параметры работы котлоагрегата, обеспечивая удобство и точность в управлении данными.

После внесения изменений пользователь нажимает кнопку "Сохранить", и данные отправляются на сервер для создания новой котельной установки. Форма сохранения котлоагрегата приведена на рис.3.3.3.



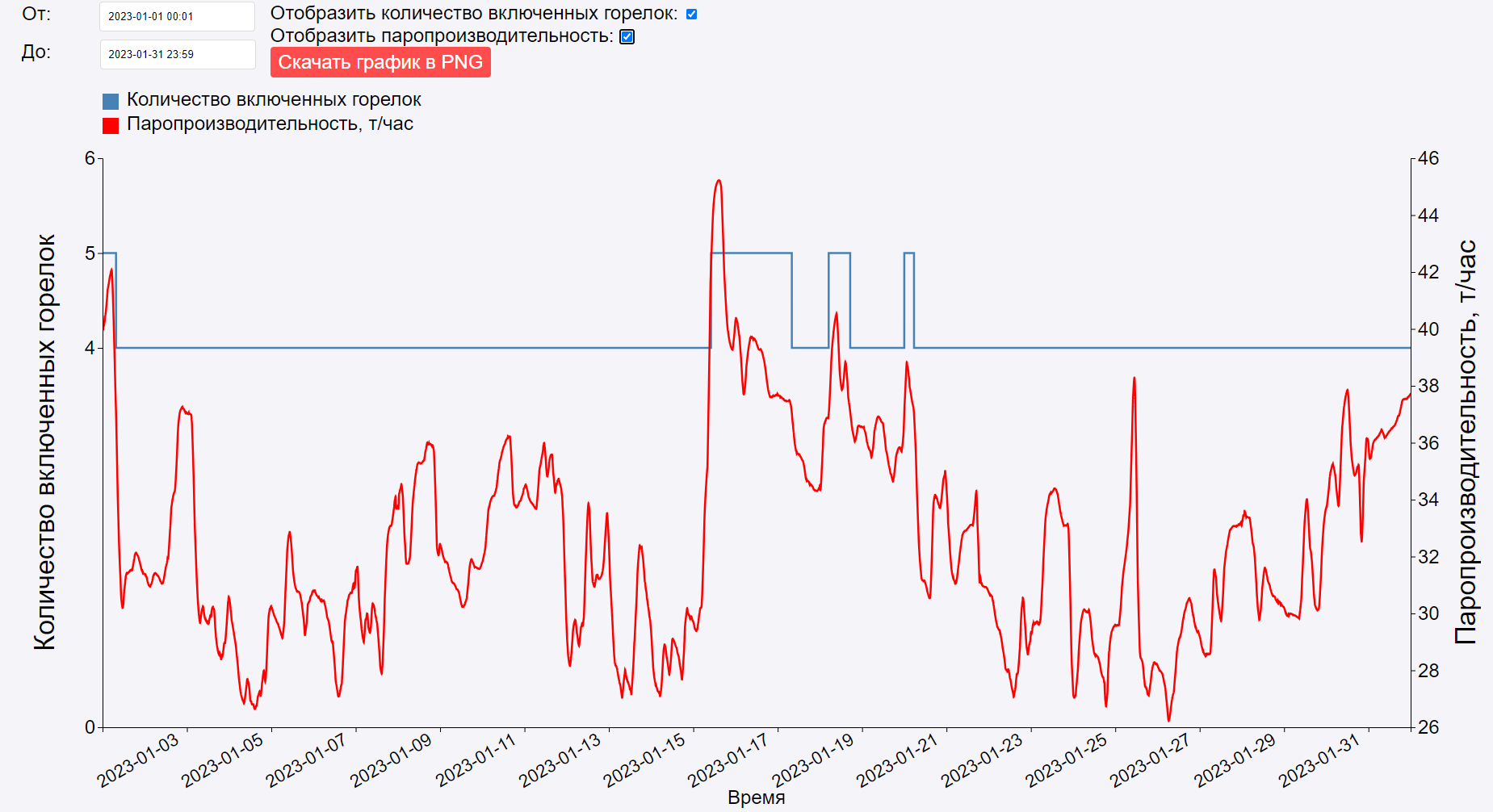
*Рисунок 3.3.3 – Форма сохранения котлоагрегата*

Все необходимые параметры выставлены, файл с данными загружен. Клиент нажимает зелёную кнопку сохранить, на сервере происходят вычисления и сохранение результатов в базу данных. Клиент переходит назад в главное меню и видит результат сохранения, пример приведён на рис.3.3.4



*Рисунок 3.3.4 – Главное меню после сохранения котлоагрегата*

Котлоагрегат сохранён и можно посмотреть результаты вычислений, нажав на синий блок, в котором размещено название котлоагрегата, пример на рис.3.3.5.



*Рисунок 3.3.5 – Страница визуализации результатов котлоагрегата*

Клиент имеет возможность выбирать определённые промежутки времени для отображения, так же имеется возможность опционального отключения отображения графика паропроизводительности или количества включенных горелок. По графику можно масштабироваться с целью лучшего рассмотрения конкретного промежутка. Помимо прочего, сверху интерфейса имеется кнопка для сохранения графика в PNG, все графики данной работы были получены этим способом.

# Перспективы и направления дальнейшей разработки

В ходе проведенной работы удалось достичь значительных результатов в области сглаживания графиков паропроизводительности, давления и расхода мазута. Применение методов сглаживания позволило существенно уменьшить количество неоднозначных графиков, что положительно сказалось на общей точности анализа данных. Однако, несмотря на достигнутые успехи, следует отметить, что полученные результаты не являются абсолютно точными, так как они были получены косвенным образом. В данной главе будут рассмотрены перспективы и направления дальнейшей разработки программы.

## Ограничения разработанного алгоритма определения количества работающих горелок

Метод скользящего среднего продемонстрировал свою эффективность в уменьшении количества неоднозначных участков в данных. Однако, несмотря на достигнутые успехи, существует ряд условностей. Полученные результаты не являются абсолютно точными, так как они были получены косвенным образом. Это означает, что существует вероятность наличия ошибок и неточностей в данных. Текущий метод сглаживания имеет ограниченную способность адаптироваться к изменяющимся условиям и требованиям, это может привести к снижению точности анализа данных в долгосрочной перспективе.

## Интеграция методов машинного обучения

Одним из направлений дальнейшей разработки программы является интеграция методов машинного обучения. Целью такого подхода является обучение модели на данных, которые производит текущая версия программы. Это позволит автоматизировать процесс анализа и повысить точность результатов. Основные этапы интеграции машинного обучения:

* Сбор данных.

Необходимо собрать достаточное количество данных для обучения модели. Эти данные должны включать как исходные данные, так и результаты, полученные с помощью текущих методов сглаживания.

* Обучение модели.

На основе собранных данных будет обучена модель машинного обучения. Модель должна быть способна выявлять закономерности и аномалии в данных, что позволит повысить точность анализа.

* Валидация и тестирование.

После обучения модель должна быть протестирована и валидирована на новых данных. Это позволит оценить её точность и надежность.

## Контроль со стороны оператора

Учитывая, что исходные данные могут содержать ошибки и неточности, необходим контроль со стороны оператора во избежание неправильного обучения модели. Оператор смог бы проверять корректность исходных данных и результатов, полученных с помощью разработанного программного решения. В случае обнаружения ошибок или неточностей оператор должен вносить необходимые коррективы в данные или настройки модели.

Дальнейшее развитие программы предполагает не только интеграцию методов машинного обучения, но и техническое совершенствование существующих алгоритмов.

Таким образом, дальнейшее развитие программы предполагает интеграцию методов машинного обучения, активное участие оператора в процессе верификации данных и техническое совершенствование существующих алгоритмов. Это позволит создать более надежную и точную систему анализа, способную адаптироваться к изменяющимся условиям и требованиям. В результате, программа будет способна обеспечивать высокую точность и надежность анализа данных, что является ключевым фактором для успешного выполнения задач в рамках данного проекта.

# Приложение А

## Листинг метода интерполяции

public static <T> void interpolateAtZerosAndNull(List<T> data, Function<T, Double> getter, BiConsumer<T, Double> setter) {  
 int n = data.size();  
  
 for (int i = 0; i < n; i++) {  
 if (Math.*abs*(getter.apply(data.get(i))) < *NEAR\_ZERO*) {  
 int counter = i;  
 while (counter < n && Math.*abs*(getter.apply(data.get(counter))) < *NEAR\_ZERO*) {  
 counter++;  
 }  
 if (counter - i > *MAXIMUM\_ZERO\_GAP*) {  
 i = counter - 1;  
 continue;  
 }

*// Находим близлижайщее ненулеове значение до и после* int left = i - 1;  
 int right = i + 1;  
  
 while (left >= 0 && Math.*abs*(getter.apply(data.get(left))) < *NEAR\_ZERO*) {  
 left--;  
 }  
  
 while (right < n && Math.*abs*(getter.apply(data.get(right))) < *NEAR\_ZERO*) {  
 right++;  
 }  
  
 if (left >= 0 && right < n) {  
 *// Считаем значение* double interpolatedValue = getter.apply(data.get(left)) +  
 (getter.apply(data.get(right)) - getter.apply(data.get(left))) (i - left) / (right - left);  
 setter.accept(data.get(i), interpolatedValue);  
 } else if (left >= 0) {setter.accept(data.get(i), getter.apply(data.get(left)));  
 } else if (right < n) {  
setter.accept(data.get(i), getter.apply(data.get(right)));  
 }  
 }  
 }  
}

# Приложение Б

## Листинг метода полиномиальной аппроксимации

public static <DATA> void approximate(List<DATA> dataList, Function<DATA, LocalDateTime> dataTimeGetter, Function<DATA, Double> dataSteamCapacityGetter, BiConsumer<DATA, Double>

coordinateSetter, int degree) {

*// Создаём объект аппроксимации* PolynomialCurveFitter fitter = PolynomialCurveFitter.*create*(degree);  
 WeightedObservedPoints points = new WeightedObservedPoints();

*Окончание приложения Б*

for (DATA data : dataList) {  
 double x = dataTimeGetter.apply(data).toEpochSecond(ZoneOffset.*UTC*);  
 double y = dataSteamCapacityGetter.apply(data);  
 points.add(x, y);  
 }  
  
 *// Находим коэфициенты* double[] coefficients = fitter.fit(points.toList());

*// Для каждой точки устанавливаем новое значение функции* for (DATA data : dataList) {  
 double y = 0.0;  
 double x = dataTimeGetter.apply(data).toEpochSecond(ZoneOffset.*UTC*);  
 for (int i = 0; i < coefficients.length; i++) {  
 y += coefficients[i] Math.*pow*(x, i);  
 }  
 coordinateSetter.accept(data, y);  
 }  
 }

# Приложение В

# Листинг метода скользящей средней

public static void movingAverage(List<RawData> rawDataList, int border) {  
 RawData[] rawDataArray = rawDataList.toArray(new RawData[0]);  
 double sumSteamCapacity = 0.0;  
 double sumMasutPresure = 0.0;  
 double sumMasutConsumption = 0.0;  
 for (int outerIndex = 0; outerIndex < rawDataList.size(); outerIndex++) {  
 int from = Math.*max*(0, outerIndex - border);  
 int to = Math.*min*(rawDataList.size() - 1, outerIndex + border);  
 for (int innerIndex = from;  
 innerIndex <= to;  
 innerIndex++) {  
 sumSteamCapacity += rawDataArray[innerIndex].getSteamCapacity();  
 sumMasutConsumption += rawDataArray[innerIndex].getFuelOilConsumption();  
 sumMasutPresure += rawDataArray[innerIndex].getFuelOilPressure();  
 }  
 int quantity = to - from + 1;  
 rawDataList.get(outerIndex).setSteamCapacity(sumSteamCapacity / quantity);  
 rawDataList.get(outerIndex).setFuelOilPressure(sumMasutPresure / quantity);  
 rawDataList.get(outerIndex).setFuelOilConsumption(sumMasutConsumption / quantity);  
 sumSteamCapacity = 0.0;  
 sumMasutConsumption = 0.0;  
 sumMasutPresure = 0.0;  
 }  
}

# Приложение Г

# Листинг метода расчёта ближайшей центроиды

public static List<Coordinate> getBurnersAmountByClusterization(List<RawData> rawDataList,

*Окончание приложения Г*

BoilerHouse boilerHouse,  
 Map<Integer, double[][]> idealParameters) {  
 List<Coordinate> coordinates = new ArrayList<>();  
 double minDistance = Double.*MAX\_VALUE*;  
 double currentDistance;  
 int burnersNum = -1;  
  
 Map<Integer, double[]> idealParametersNew = new HashMap<>();  
 for (int key : idealParameters.keySet()) {  
 double[][] array = idealParameters.get(key);  
 double[] arrayNew = new double[3];  
 for (int i = 0; i < array.length; i++) {  
 double[] arr = array[i];  
 double sum = 0;  
 for (double v : arr) {  
 sum += v;  
 }  
 sum /= arr.length;  
 arrayNew[i] = sum;  
 }  
 idealParametersNew.put(key, arrayNew);  
 }  
  
  
 *//Проходим по всем значениям* for (RawData rawDataPoint : rawDataList) {  
 *//После выполнения это цикла, мы точно можем сказать сколько горелок было включено в текузий момент* for (int burnersNumByTable : idealParametersNew.keySet()) {  
 double[] array = idealParametersNew.get(burnersNumByTable);  
 currentDistance = Math.*sqrt*(Math.*pow*(rawDataPoint.getFuelOilPressure() - array[1], 2)  
 + Math.*pow*(rawDataPoint.getFuelOilConsumption() - array[2], 2)  
 + Math.*pow*(rawDataPoint.getSteamCapacity() - array[0], 2));  
 if (currentDistance < minDistance) {  
 minDistance = currentDistance;  
 burnersNum = burnersNumByTable;  
 }  
 }  
 minDistance = Double.*MAX\_VALUE*;  
 coordinates.add(

Coordinate.*builder*()  
 .time(rawDataPoint.getTime())  
 .burnersNum(burnersNum)  
 .steamCapacity(rawDataPoint.getSteamCapacity())  
 .boilerHouse(boilerHouse)  
 .build());  
 }  
 return coordinates;

}

# Приложение Д

# Листинг метода расчёта оптимальных параметров аппроксимации

@Test  
public void findOptimalApproximation() {  
 BoilerHouse boilerHouse = boilerHouseRepository.findByName("7 котёл 2022 DEFAULT").get();  
 int maxGap = 500;  
 int maxDegree = 5;  
 int minGap = 50;  
 int gapStep = 50;  
  
 int gapSteps = (maxGap - minGap) / gapStep + 1;  
 int degreeSteps = maxDegree - 2 + 1;  
  
 int[][] implicitGapsTable = new int[gapSteps][degreeSteps];  
  
 for (int gapIndex = 0, gap = minGap; gap <= maxGap; gapIndex++, gap += gapStep) {  
 for (int degree = 2; degree <= maxDegree; degree++) {   
 List<RawData> rawDataList = dataRepository.findDataByTimeBetweenAndBoilerHouse(boilerHouse.getMinDate(), boilerHouse.getMaxDate(), boilerHouse);  
  
 for (int start = 0, end = gap; end < rawDataList.size(); start += gap, end += gap) {  
 *approximate*(rawDataList.subList(start, end), RawData::getTime, RawData::getFuelOilPressure, RawData::setFuelOilPressure, degree);  
 *approximate*(rawDataList.subList(start, end), RawData::getTime, RawData::getFuelOilConsumption, RawData::setFuelOilConsumption, degree);  
 *approximate*(rawDataList.subList(start, end), RawData::getTime, RawData::getSteamCapacity, RawData::setSteamCapacity, degree);  
 }  
  
 List<Coordinate> coordinates = *getBurnersAmountByClusterization*(rawDataList, boilerHouse, idealParameters);  
  
 int implicitGapsCounter = *countGaps*(coordinates);  
  
 implicitGapsTable[gapIndex][degree - 2] = implicitGapsCounter;  
 }  
 }  
  
 System.*out*.println("Gap\\Degree");  
 System.*out*.print(" ");  
 for (int degree = 2; degree <= maxDegree; degree++) {  
 System.*out*.printf("%8d", degree);  
 }  
 System.*out*.println();  
  
 for (int gapIndex = 0, gap = minGap; gap <= maxGap; gapIndex++, gap += gapStep) {  
 System.*out*.printf("%4d ", gap);  
 for (int degree = 2; degree <= maxDegree; degree++) {  
 System.*out*.printf("%8d", implicitGapsTable[gapIndex][degree - 2]);  
 }  
 System.*out*.println();  
 }  
}

# Приложение Е

# Листинг метода расчёта оптимальной длины окна скользящего среднего

@Test  
public void findOptimalWindow() {  
 BoilerHouse boilerHouse = boilerHouseRepository.findByName("7 котёл 2022 DEFAULT").get();  
 for (int border = 2; border <= 200; border+=10) {  
 *//Находим все нужные нам значения параметров работы за определённый промежуток времени* List<RawData> rawDataList = dataRepository.findDataByTimeBetweenAndBoilerHouse(boilerHouse.getMinDate(), boilerHouse.getMaxDate(), boilerHouse);  
  
 *//"Сглаживаем" данные методом кользящего среднего  
 movingAverage*(rawDataList, border);  
  
 *//Считаем кол-во включенных горелок для каждого момента времени* List<Coordinate> coordinates = *getBurnersAmountByClusterization*(rawDataList, boilerHouse, idealParameters);  
  
 *//Печатаем все промежутки времени* int count = *countGaps*(coordinates);  
 System.*out*.println(border + "," + count);  
 }  
}