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

**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.

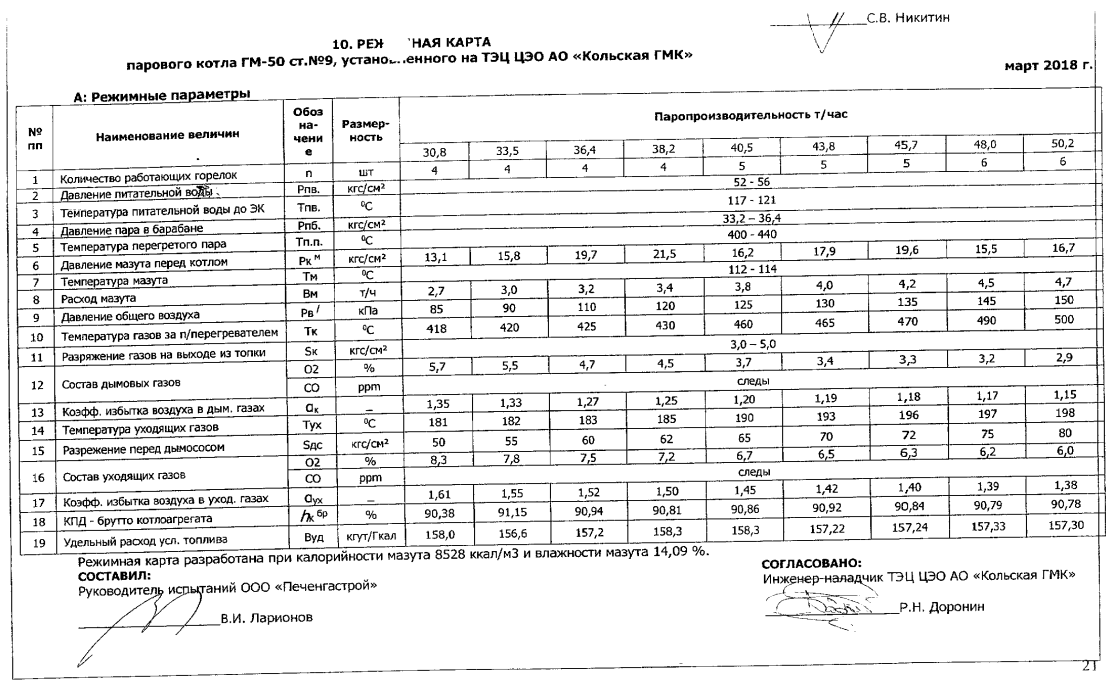
1. Контроль и мониторинг.

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*

## Параметры работы играющие ключевую роль 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.

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 – начало существования Unix подобной системы.

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.

Преимущества

1. Уменьшение шума: Скользящее среднее помогает уменьшить "шум" в данных, вызванный случайными колебаниями, и облегчает восприятие основных тенденций и моделей.
2. Простота и понятность: Метод скользящей средней прост в реализации и интерпретации, что делает его доступным для широкого круга пользователей, не обладающих специальными техническими знаниями.

Недостатки

1. Фазовое запаздывание: Из-за особенностей расчёта скользящие средние отстают от текущих значений, особенно при больших размерах окон.
2. Потеря данных на краях: На начале и конце временных рядов скользящее среднее не может быть рассчитано для всех точек из-за отсутствия достаточного количества предшествующих или последующих данных.
3. Неадаптивность: Скользящее среднее не адаптируется к изменениям в поведении данных, что может быть критично в случаях, когда в данных присутствуют важные экономические или финансовые события.

## 2.4. Кластеризация

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.

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 \(k\).

2. Иерархическая кластеризация:

- Делится на агломеративную (снизу вверх) и дивизивную (сверху вниз).

- 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.

3. 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.

4. Gaussian Mixture Models (GMM):

- Предполагает, что данные генерируются из смеси нескольких гауссовых распределений.

- Использует метод максимального правдоподобия для оценки параметров распределений.

В вашем случае, когда центроиды уже заданы, вы используете метод, который можно назвать "кластеризация с фиксированными центроидами". Это действительно можно рассматривать как форму кластеризации, хотя и с некоторыми особенностями:

1. Фиксированные центроиды:

- В отличие от традиционных методов, где центроиды определяются в процессе кластеризации, здесь они заданы заранее.

- Это может быть полезно, когда у вас есть априорные знания о структуре данных или когда вы хотите использовать заранее определенные эталонные точки.

2. Евклидово расстояние:

- Вы используете евклидово расстояние для определения ближайшего кластера.

- Это стандартный подход в задачах кластеризации и классификации, особенно когда данные имеют метрическую структуру.

Применение в контексте котлоагрегата

Ваш подход к определению текущего режима работы котлоагрегата действительно можно рассматривать как форму кластеризации. Вот как это работает в вашем случае:

1. Режимная карта:

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

2. Текущие параметры:

- Вы измеряете текущие параметры (паропроизводительность, давление, расход мазута) и сравниваете их с режимами из режимной карты.

3. Определение ближайшего режима:

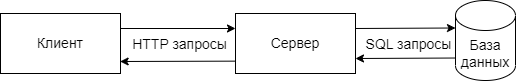
- Вы вычисляете евклидово расстояние между текущими параметрами и каждым из режимов.

- Режим с минимальным расстоянием считается текущим режимом работы.

Определение оптимального подхода в сглаживания исходных графиков

# 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. Данные представлены в виде таблицы, где каждая строка соответствует одному моменту времени. Пример в таблице ниже

*Таблица 1 – Основные функции активации*

|  |  |  |  |
| --- | --- | --- | --- |
| 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.

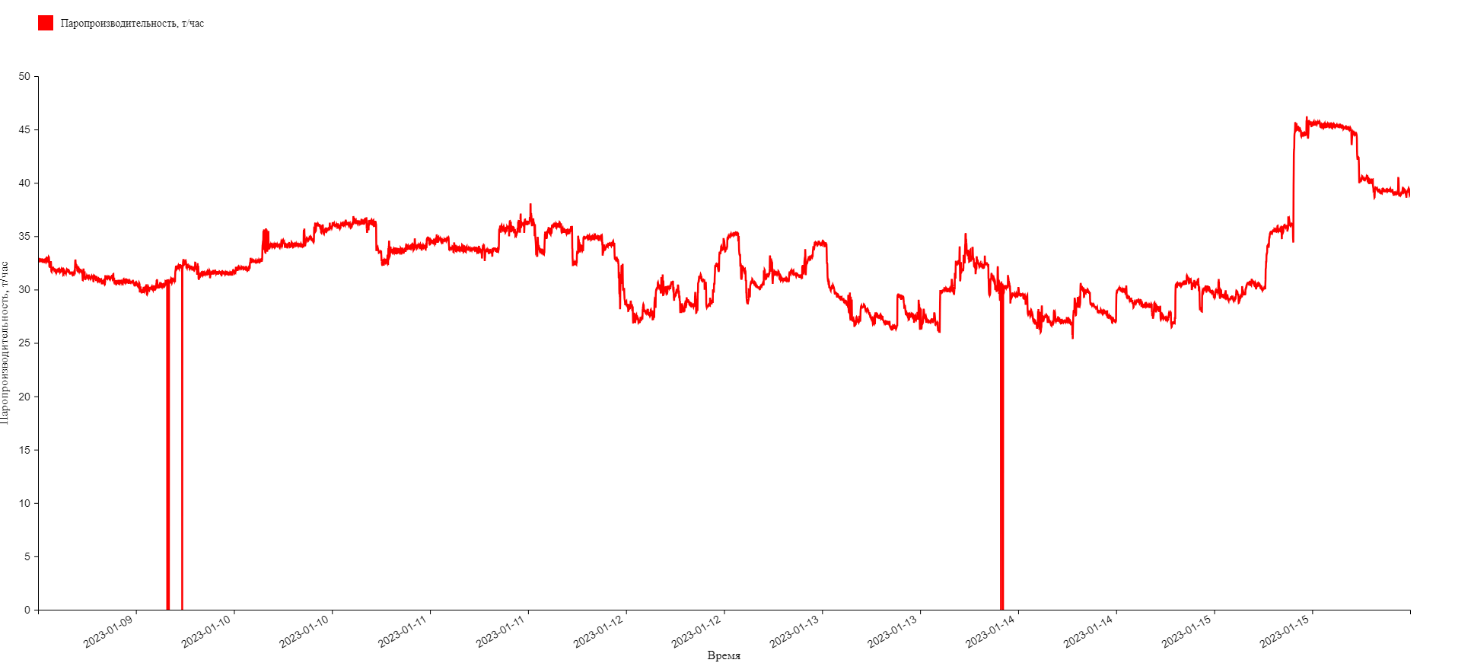
Пример1

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |

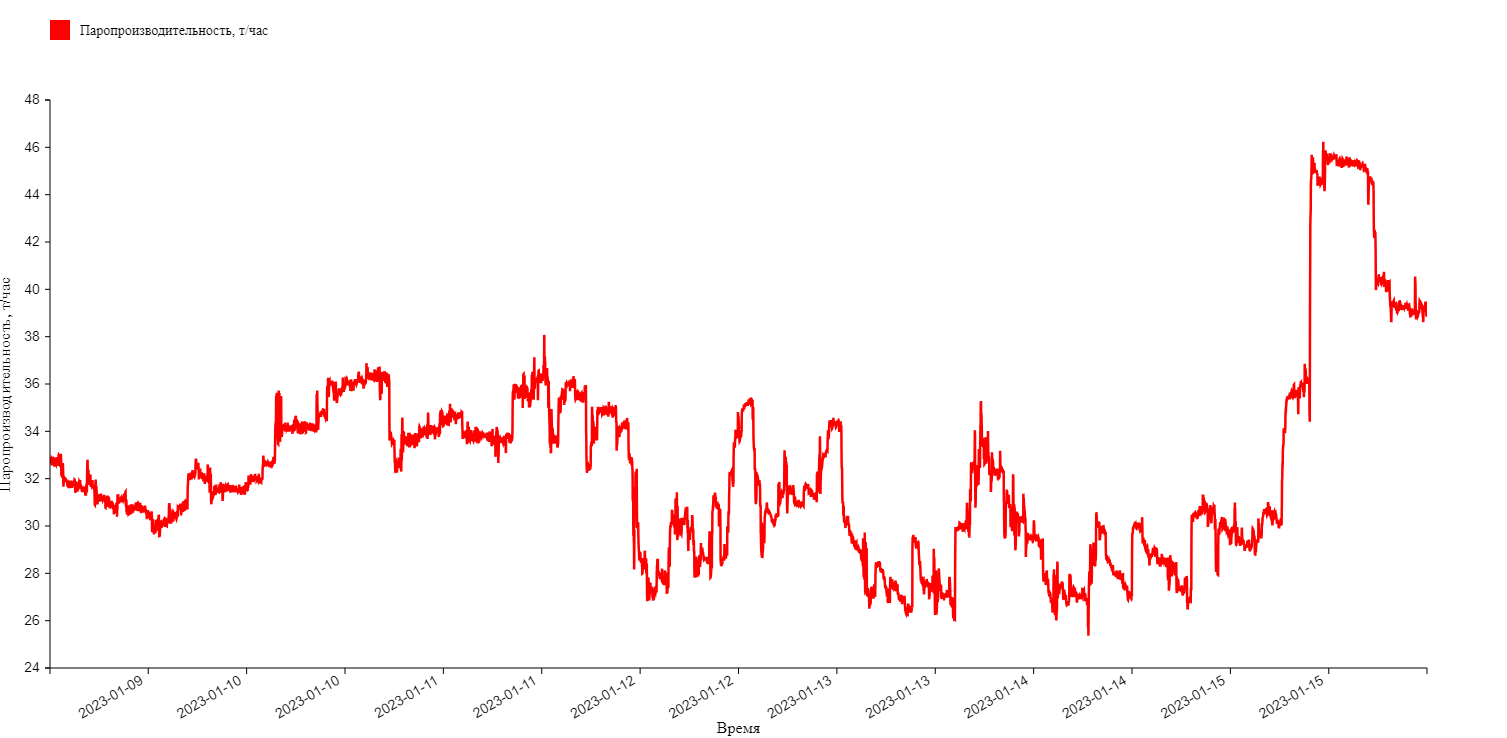
Также могут встречаться значения, равные 0.0, в те моменты, когда котлоагрегат функционирует на определённых мощностях и можно точно сказать, что в эти моменты времени датчики некорректно сняли параметры работы. Пример снизу

|  |  |  |  |
| --- | --- | --- | --- |
| 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 данном рисунке пример с производительностью пара. Эти значения необходимо обработать, чтобы они не искажали дальнейший анализ.



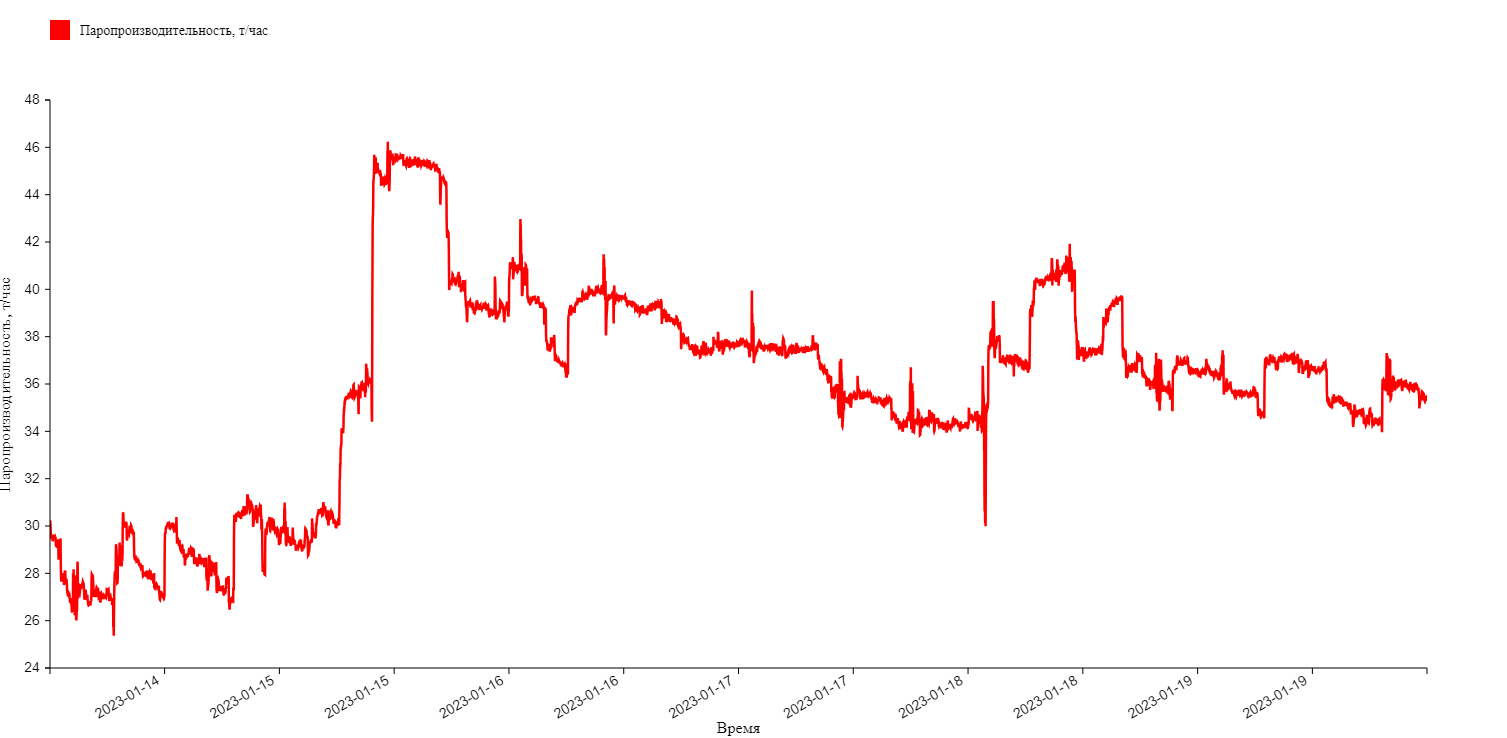
Пропущенные и некорректные значения заполняются с помощью линейной интерполяции. Интерполяция позволяет плавно восстановить недостающие данные, основываясь на соседних значениях.



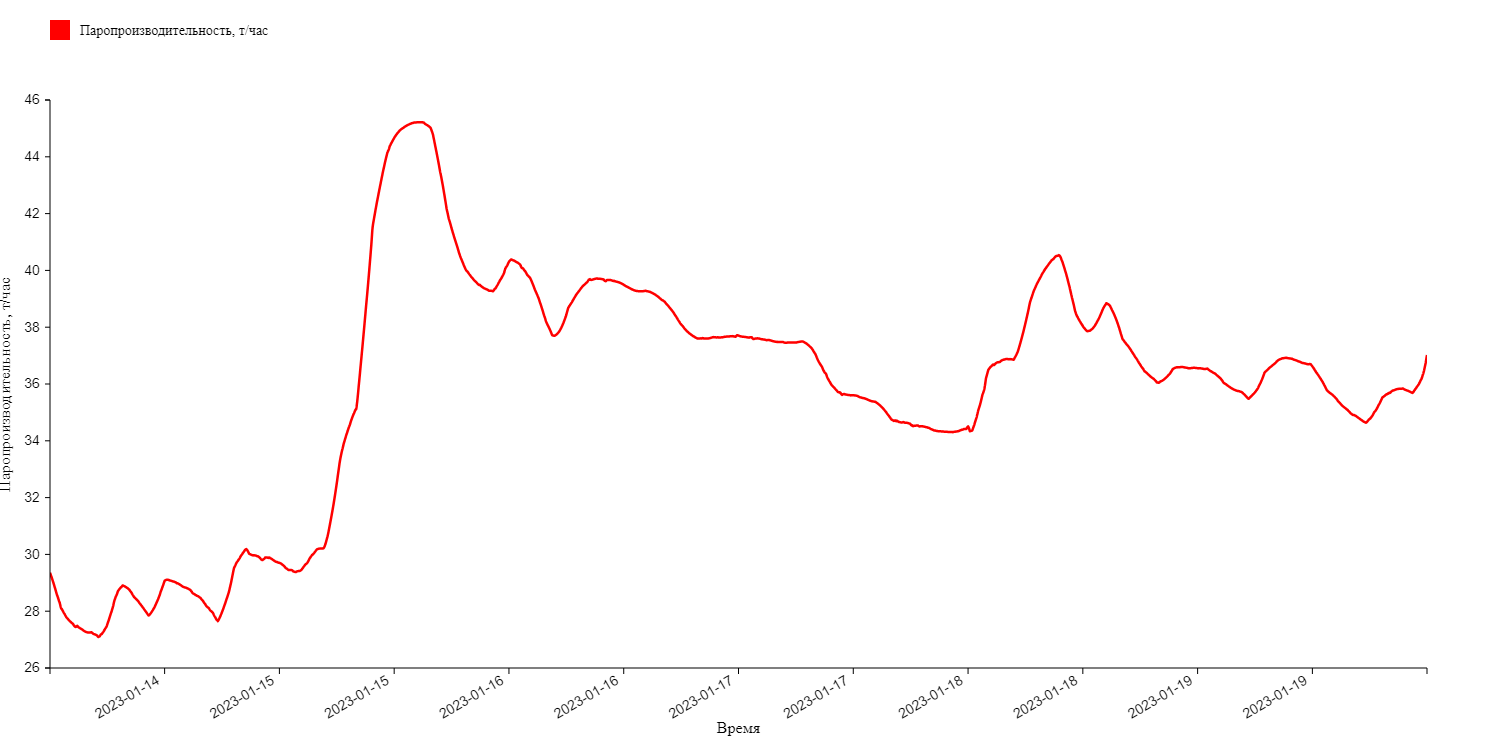
Как мы видим некорректно снятые значения исчезли. Это помогает компенсировать сбои в работе датчиков и получить более полные и корректные данные. Интерполяция применяется для каждого параметра работы котлоагрегата отдельно. В данном случает приведён пример с производительностью пара.

2. Сглаживание данных:

После интерполяции данные могут все еще содержать резкие скачки, которые негативно влияют на результаты анализа. Для сглаживания данных используется метод скользящего среднего.



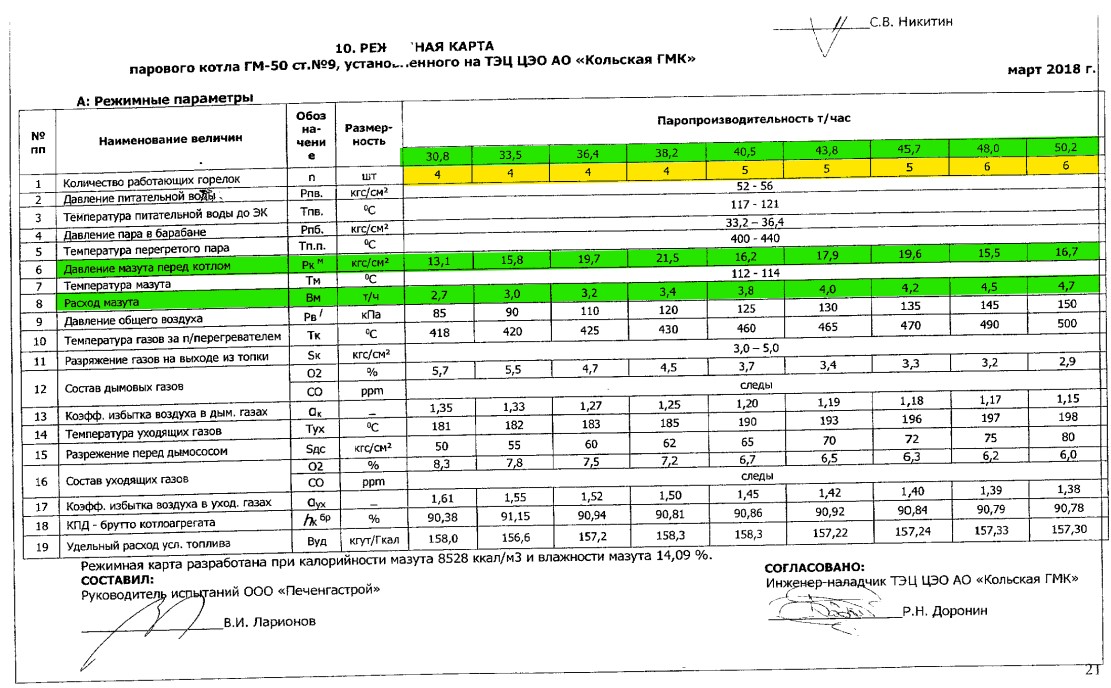
Скользящее среднее с окном в 100 минут вычисляется для каждого параметра работы котлоагрегата отдельно с заданным размером окна.



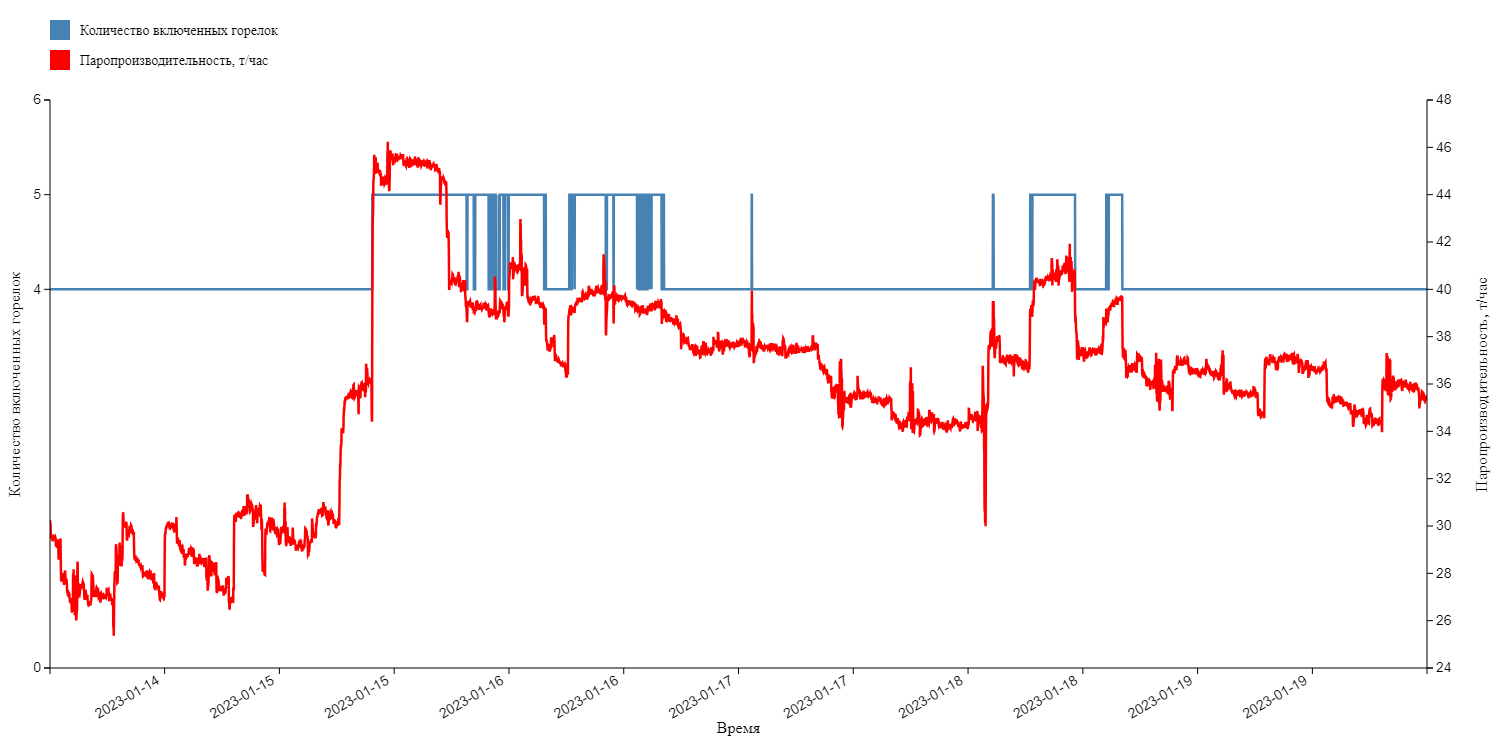
Это позволяет уменьшить влияние случайных колебаний и получить более стабильные значения параметров.

3. Определение текущего режима работы:

После сглаживания данных необходимо определить текущий режим работы котлоагрегата. Для этого используется метод кластеризации с заранее заданными центроидами. Режимная карта содержит заранее определенные режимы работы котлоагрегата, которые можно рассматривать как центроиды кластеров. На режимной карта на рис.3.1. для удобства зелёным цветом выделены интересующие нас параметры – производительность пара, давление и расход мазута, жёлтым – количество включенных горелок.

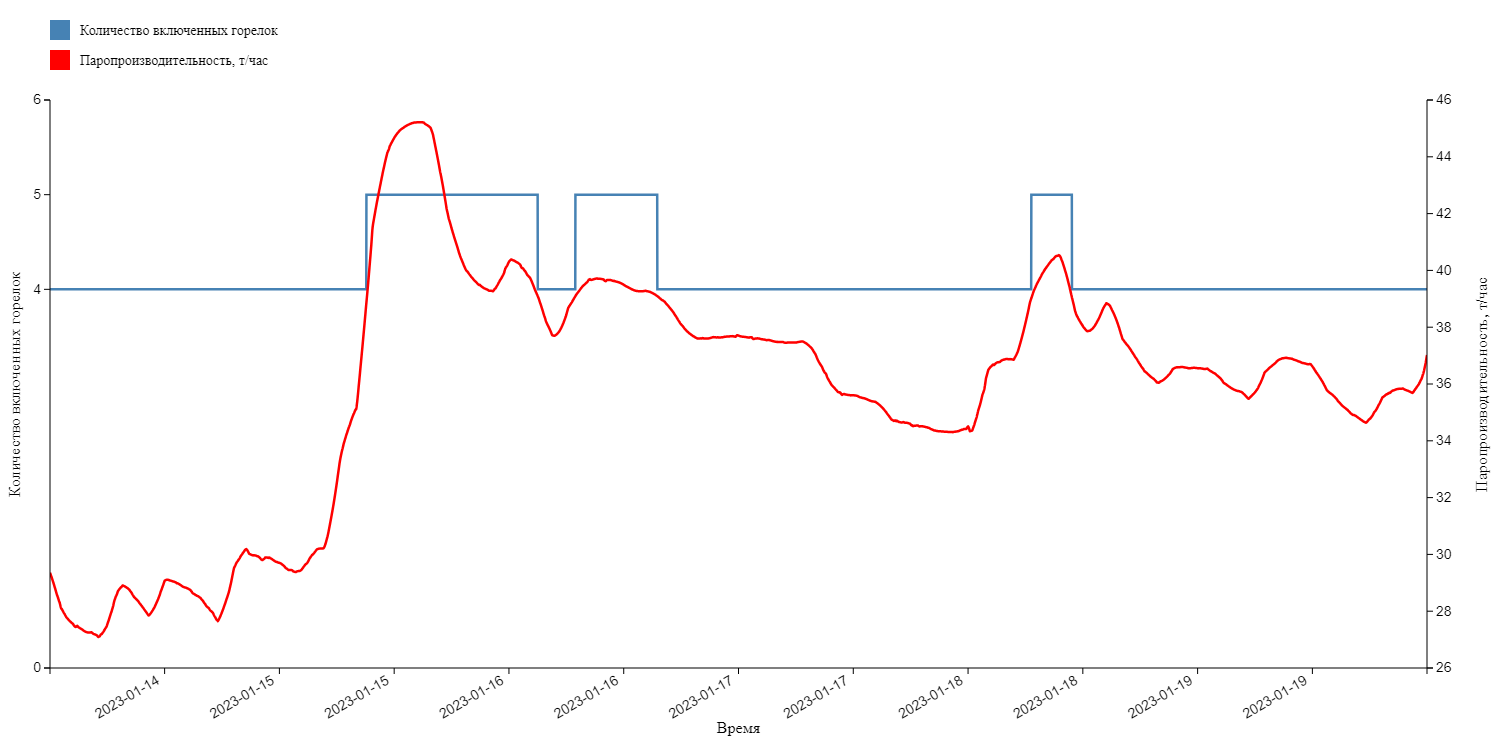


Каждый режим характеризуется определенными значениями паропроизводительности, давления мазута и расхода мазута. Для каждого значения исходных данных считается евклидово расстояние до каждого из режимов. Режим с минимальным евклидовым расстоянием считается текущим режимом работы котлоагрегата. На данном рис.3.1. представлены результаты расчёта количества работающих горелок для котлоагрегата №9 Кольской ГМК 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 же это маловероятно. 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. На данном же графике смена режимов работы местами осуществляется меньше чем каждые 10 минут. 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.

На рис.3.1. представлены результаты расчёта количества работающих горелок после применения скользящего среднего.



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 данном рисунке представлена диаграмма классов, используемых в работе.



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 один момент времени.
* Coordinate – пользовательский класс для сохранения результатов вычисления количества работающих горелок.
* BoilerHouse – пользовательский класс, котельная установка.

На данной диаграмме приведён алгоритм сохранения работы котлоагрегата за определённый временной интервал.

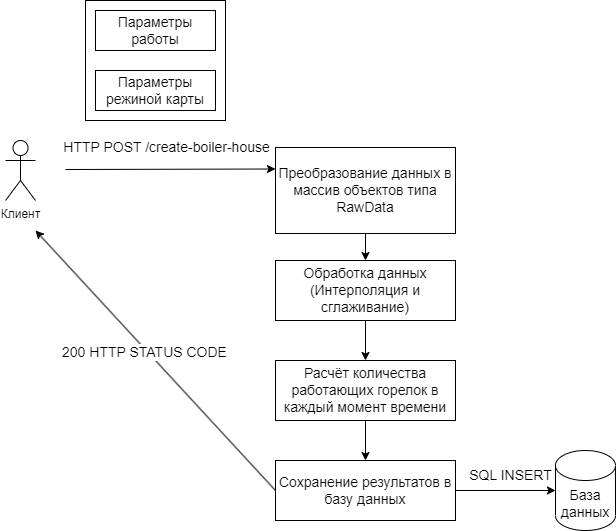


Диаграмма сохранения котлоагрегата

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. После этого сохранения клиент в любой момент может открыть интересующий его котлоагрегат и посмотреть результаты вычислений.

На данной диаграмме продемонстрирован алгоритм получения результатов вычислений со стороны клиента.

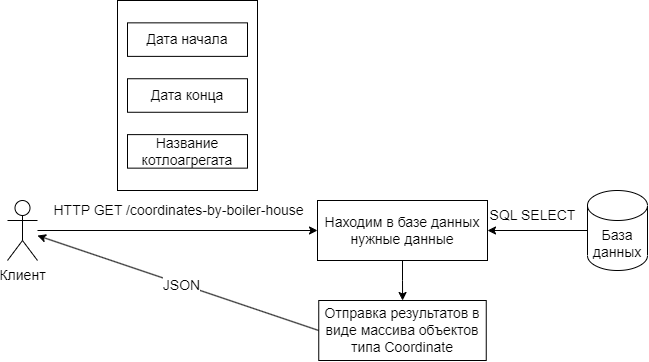


Диаграмма получения результатов вычислений

Как было сказано ранее, клиент может в любой момент посмотреть результаты вычислений сохранённый 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. На клиентскую часть приходят данные в формате JSON, который представляет из себя текстовый формат с неупорядоченным множеством пар ключ-значение.

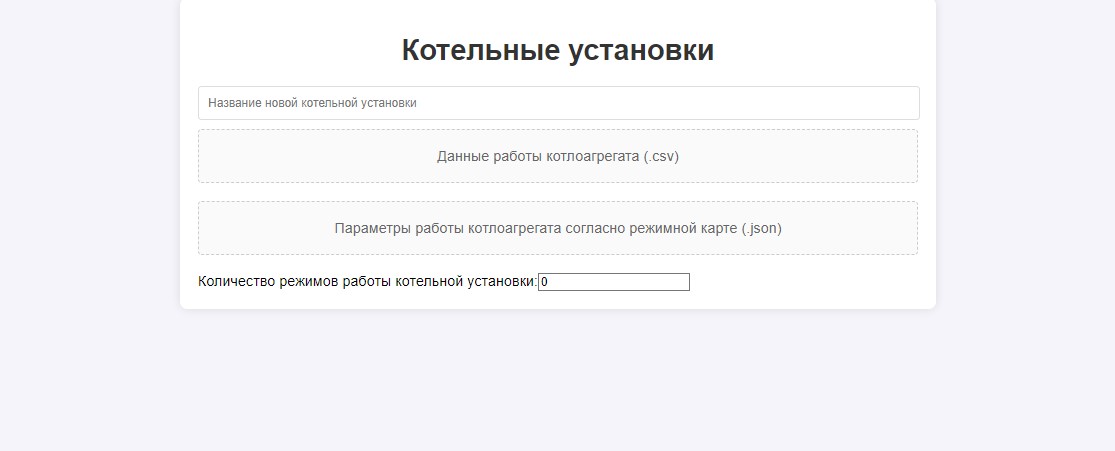
## 

## 3.2 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.

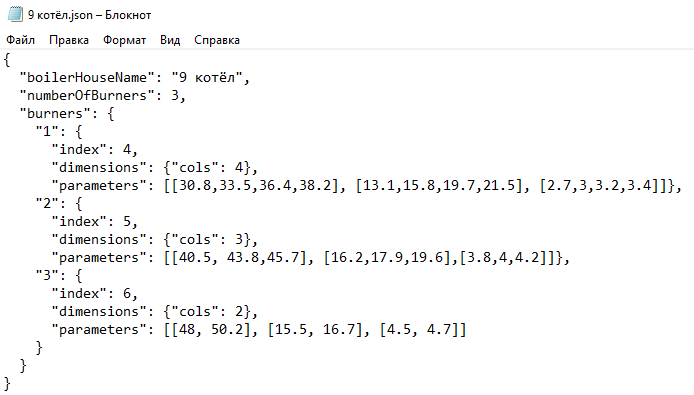
JavaScript - это язык программирования, используемый для создания интерактивных веб-страниц. React - это библиотека для создания пользовательских интерфейсов.

Первое, что видит пользователь – главное меню.

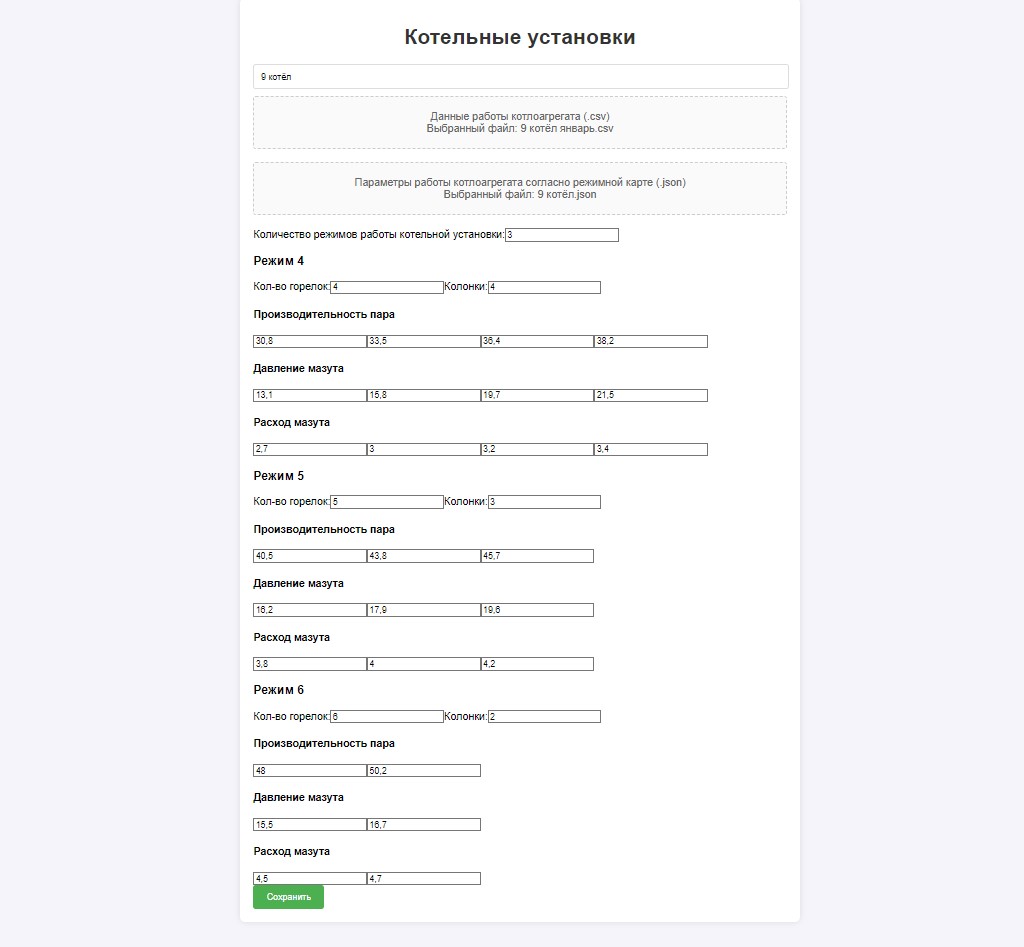


Главная страница

Здесь отображаются сохранённые ранее данные о работе котлоагрегатов и формы для загрузки файлов с целью сохранения новых. Чтобы сохранить новые данные, выберем csv файл с данными работы за определённый временной интервал и json файл с параметрами работы, взятыми из режимной карты.

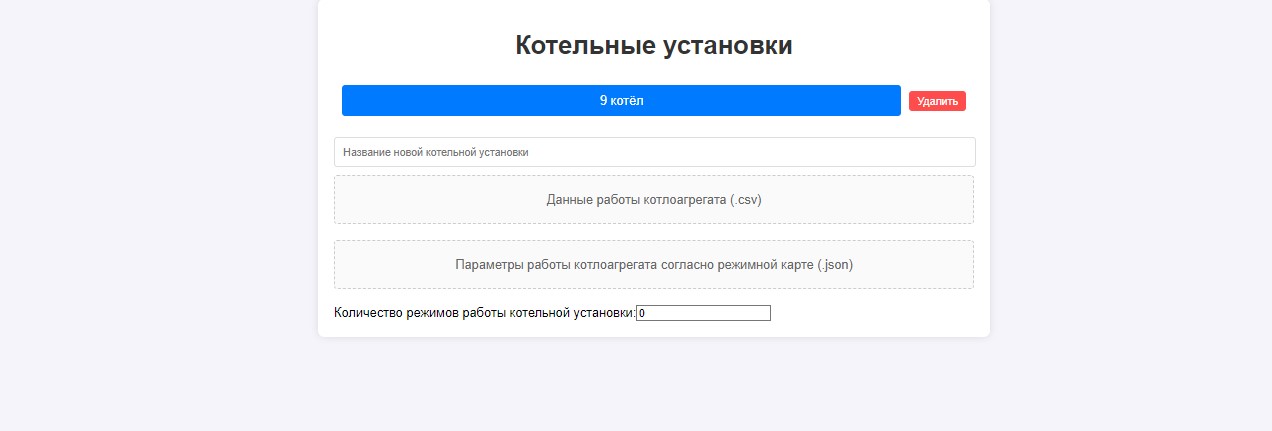


При необходимости можно изменить параметры работы в полях ниже. Нажимаем кнопку «Сохранить» и переходим в главное меню.



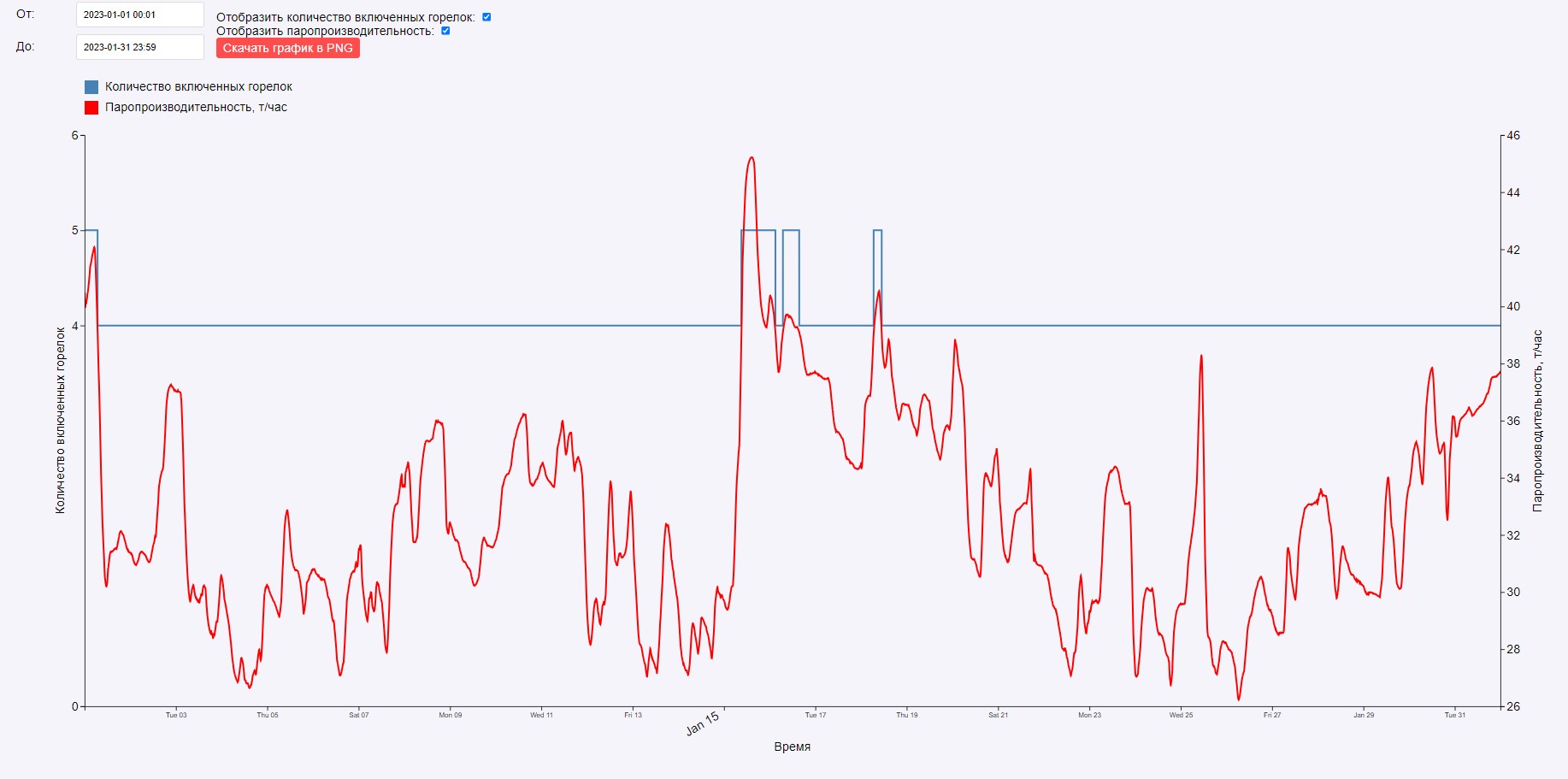
Процесс сохранения котлоагрегата

Как мы видим, данные 9 котлоагрегата успешно сохранены.



Результат сохранения котлоагрегата

Теперь мы можем выбрать его и посмотреть результаты вычислений. Клиенту предоставляется 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.



Визуализация результатов вычислений

# Тестирование и оценка точности

# Приложение А

## Код 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

## Код 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);  
 }  
 }