



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)  
(МГТУ им. Н.Э. Баумана)

---

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

# РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

*НА ТЕМУ:*

*Метод сравнения*

*многомерных траекторий*

*на примере кольцевых гонок*

---

---

---

---

---

Студент ИУ7-83Б  
(Группа)

(Подпись, дата)

**А. О. Степанов**  
(И.О.Фамилия)

Руководитель ВКР

(Подпись, дата)

**Д. С. Бабарыкин**  
(И.О.Фамилия)

Консультант

(Подпись, дата)

(И.О.Фамилия)

Консультант

(Подпись, дата)

(И.О.Фамилия)

Нормоконтролер

(Подпись, дата)

**Ю. В. Строганов**  
(И.О.Фамилия)

2021 г.

**Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет имени Н.Э. Баумана  
(национальный исследовательский университет)  
(МГТУ им. Н.Э. Баумана)**

---

УТВЕРЖДАЮ

Заведующий кафедрой ИУ7  
(Индекс)

Рудаков И.В.  
(И.О.Фамилия)

«        » 20 20 г.

## **З А Д А Н И Е на выполнение выпускной квалификационной работы бакалавра**

Студент группы ИУ7-83Б

Степанов Александр Олегович  
(фамилия, имя, отчество)

Тема квалификационной работы Метод сравнения многомерных траекторий на примере  
кольцевых гонок

---

---

Источник тематики (НИР кафедры, заказ организаций и т.п.)

НИР кафедры

---

---

Тема квалификационной работы утверждена распоряжением по факультету  
ИУ № 03.02.01-04.03/14 от « 11 » декабря 2020 г.

### **Часть 1. Аналитический раздел**

Провести анализ предметной области. Формализовать цель работы и перечень задач для ее  
достижения. Дать общую постановку задачи. Провести анализ существующих методов  
сравнения многомерных траекторий и сравнить их. Выбрать оптимальные методы для  
использования в работе.

---

### **Часть 2. Конструкторский раздел**

Определить формат входных и выходных данных. Спроектировать метод сравнения  
многомерных траекторий. Привести схемы для предложенного метода.

---

**Часть 3. Технологический раздел**

Выбрать технологический стек для реализации предложенного метода. Подготовить необходимые данные. Спроектировать программное обеспечение, привести диаграммы. Выполнить программную реализацию метода.

---

**Часть 4. Исследовательский раздел**

Провести исследование применимости предложенного метода и оценить его эффективность.

---

***Оформление квалификационной работы:***

Расчетно-пояснительная записка на 40-80 листах формата А4.

Перечень графического (илюстративного) материала (чертежи, плакаты, слайды и т.п.)

На защиту квалификационной работы должна быть представлена презентация,

---

---

Дата выдачи задания « 18 » сентября 2020 г.

В соответствии с учебным планом выпускную квалификационную работу выполнить в полном объеме в срок до « 25 » мая 20 21 г.

**Руководитель квалификационной работы**

Д.С. Бабарыкин  
(Подпись, дата)

**Студент**

А.О. Степанов  
(Подпись, дата)

**Примечание:**

1. Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

**Министерство науки и высшего образования Российской Федерации**  
**Федеральное государственное бюджетное образовательное учреждение**  
**высшего образования**  
**«Московский государственный технический университет имени Н.Э. Баумана**  
**(национальный исследовательский университет)**  
**(МГТУ им. Н.Э. Баумана)**

---

**ФАКУЛЬТЕТ** ИУ

УТВЕРЖДАЮ

**КАФЕДРА** ИУ7

Заведующий кафедрой ИУ7  
(Индекс)

**ГРУППА** ИУ7-83Б

И.В. Рудаков  
(И.О.Фамилия)

«        » 20        г.

**КАЛЕНДАРНЫЙ ПЛАН**  
**выполнения выпускной квалификационной работы**  
**студента:** Степанова Александра Олеговича  
(фамилия, имя, отчество)

Тема квалификационной работы Метод сравнения многомерных траекторий на примере кольцевых гонок

№ п/п	Наименование этапов выпускной квалификационной работы	Сроки выполнения этапов		Отметка о выполнении	
		план	факт	Должность	ФИО, подпись
1.	Задание на выполнение работы. Формулирование проблемы, цели и задач работы	24.10.2020 Планируемая дата		Руководитель ВКР	
2.	1 часть Аналитический раздел	24.12.2020 Планируемая дата		Руководитель ВКР	
3.	Утверждение окончательных формулировок решаемой проблемы, цели работы и перечня задач	24.12.2020 Планируемая дата		Заведующий кафедрой	
4.	2 часть Конструкторский раздел	28.04.2021 Планируемая дата		Руководитель ВКР	
5.	3, 4 части Технологический и Исследовательский разделы	18.05.2021 Планируемая дата		Руководитель ВКР	
6.	1-я редакция работы	20.05.2021 Планируемая дата		Руководитель ВКР	
7.	Подготовка доклада и презентации	22.05.2021 Планируемая дата			
8.	Заключение руководителя	24.05.2021 Планируемая дата		Руководитель ВКР	
9.	Нормоконтроль	27.05.2021 Планируемая дата		Нормоконтролер	
10.	Внешняя рецензия	24.05.2021 Планируемая дата			
11.	Защита работы на ГЭК	Планируемая дата			

Студент \_\_\_\_\_

Руководитель работы \_\_\_\_\_

## **РЕФЕРАТ**

В аналитическом разделе рассматриваются алгоритмы необходимые для проведения сравнения двух траекторий в многомерном пространстве, а именно алгоритмы фильтрации, алгоритмы преобразования в непрерывную траекторию и алгоритмы разделения набора данных по тенденции.

В конструкторском разделе ставится задача и рассматривается разработанный метод сравнения многомерных траекторий, подробно описываются используемые алгоритмы

В технологическом разделе выбраны средства реализации метода для последующей демонстрации работы, приведена структура разработанного программного обеспечения, рассмотрены важные детали реализации, а также приведен пользовательский интерфейс.

В исследовательском разделе приведено описание поставленных экспериментов по рассмотрению применимости метода и выбору алгоритма для преобразования траектории в непрерывную величину, а так же их результаты.

Отчет содержит 46 стр., 20 рис., 1 табл., 21 источн., 1 прил.

## СОДЕРЖАНИЕ

Реферат .....	5
Введение .....	8
1 Аналитический раздел .....	9
1.1 Автоспорт .....	9
1.1.1 Гоночные трассы .....	9
1.1.2 Пределы устойчивости .....	9
1.2 Траектории .....	11
1.3 Метод сравнения многомерных траекторий .....	12
1.3.1 Существующие аналоги .....	12
1.3.2 Фильтрация .....	14
1.3.3 Преобразование в непрерывную траекторию .....	19
1.3.4 Разделение значений на части с общей тенденцией .....	23
1.4 Вывод .....	25
2 Конструкторский раздел .....	27
2.1 Функциональная модель .....	27
2.2 Описание метода .....	27
2.2.1 Декомпозиция функциональной модели .....	27
2.2.2 Фильтрация траекторий .....	28
2.2.3 Преобразование траектории в непрерывную .....	29
2.2.4 Расчет отставания .....	29
2.2.5 Деление отставания на промежутки со схожей тенденцией .....	32
2.3 Вывод .....	33
3 Технологический раздел .....	34
3.1 Средства реализации .....	34
3.1.1 Языки программирования .....	34

3.1.2 Используемые библиотеки и фреймворки . . . . .	34
3.2 Структура разработанного ПО . . . . .	35
3.3 API для обращения клиента к серверу . . . . .	36
3.4 Формат входного файла . . . . .	37
3.5 Интерфейс программы . . . . .	37
3.6 Вывод . . . . .	39
4 Исследовательский раздел . . . . .	40
4.1 Выбор алгоритма преобразования в непрерывную величину . . . . .	40
4.2 Представление результата . . . . .	40
4.3 Размеченные данные . . . . .	40
4.4 Метрики применимости . . . . .	41
4.5 Вывод . . . . .	43
Заключение . . . . .	44
Список использованных источников . . . . .	45
Приложение А ЛИСТИНГИ . . . . .	47

## ВВЕДЕНИЕ

В настоящее время очень популярным видом спорта являются автомобильные гонки. Одной из разновидностей подобных соревнований являются кольцевые гонки, в которых на успешное выступление участника влияет не скорость автомобиля, а умение гонщика проходить данную трассу за минимальный промежуток времени. Для улучшения навыков прохождения конкретного трека гонщику необходимо знать где он совершил ошибки, которые могут выражаться в самых различных видах, например, неправильно подобранная траектория прохождения участка дороги или выбран неудачный момент для поворота, слишком сильное торможение или наоборот, черезсчур большое ускорение, повлекшее за собой потерю управления на короткий промежуток времени. Для поиска всех совершенных ошибок при прохождении необходимо сравнить свой заезд с самым лучшим прохождением.

Для сравнения двух заездов производится оценка путем нахождения отклонения одной траектории от другой в многомерном пространстве.

Целью данной работы является разработать метод сравнения многомерных траекторий на кольцевых трассах.

Для достижения этой цели ставятся следующие задачи:

- а) проанализировать подходы к сравнению набора точек в многомерном пространстве;
- б) реализовать разработанный метод;
- в) визуализировать результат работы метода;
- г) оценить применимость метода.

# **1 Аналитический раздел**

## **1.1 Автоспорт**

### **1.1.1 Гоночные трассы**

Трассы и их участки, используемые для проведения автомобильных гоночных соревнований или спокойного движения автомобилей, классифицируются:

- длина (протяженность) и ширина;
- скоростной режим;
- тип дорожной поверхности (асфальт, бетон, гравий, бульдозер, грунт, песок, снег, лед);
- категория (от 0 до 7; одиночные или повторяющиеся простые и сложные повороты).

Оптимальная траектория является идеальной замкнутой линией по которой рекомендуется двигаться гонщику. Так как ее геометрия определена совместно наивысшей средней скоростью и минимальным временем прохождения трассы, она характерна высоким «держаком» – параметром дорожного полотна, вызванным многократным прохождением автомобилей с износом покрышек, оставляющих частицы разложения в маленьких и больших открытых углублениях трассы.

### **1.1.2 Пределы устойчивости**

В автоспорте использование активных систем контроля показателей устойчивости, запрещено правилами соревнований. Такие правила требуют от гонщиков высокого мастерства и знания, как проходить кривые по оптимальным траекториям. Поэтому автогонщики профессионалы хорошо разбираются в устройстве эксплуатируемых ими автомобилей, и проходят кривые на скоростях, определяющих пределы устойчивости

Устойчивостью скоростного безрельсового транспортного средства (в частности автомобиля) принято называть его способность сохранять заданную траекторию движения при воздействии факторов внешней среды [1].

Принято различать траекторную и курсовую устойчивость автомобиля, а так же устойчивость к его опрокидыванию.

Траекторная устойчивость автомобиля определяет способность его центра масс сохранять заданное направление движения. Оценивается смещением центра масс автомобиля относительно заданной траектории движения.

Курсовая устойчивость – способность автомобиля сохранять ориентацию его продольной оси. Определяется курсовым углом. При этом предельный случай потери траекторной и курсовой устойчивости возникает в момент начала скольжения одного или всех мостов автомобиля.

В случае опрокидывания автомобиля, предельным случаем потери устойчивости является отрыв колес автомобиля от опорной плоскости дороги. При этом различают продольное (автомобиль поворачивается в продольной плоскости) и поперечное (автомобиль поворачивается в поперечной плоскости) опрокидывание. В связи с этим различают поперечную и продольную потерю устойчивости.

Поперечная потеря устойчивости возникает либо в случае бокового опрокидывания автомобиля, либо в случае его заноса (скольжения). В основном этот тип потери устойчивости возникает при движении по криволинейным участкам дороги (за счет действия сил инерции), косогорам (за счет действия дополнительной составляющей от силы тяжести) и при сильном боковом ветре.

Продольная потеря устойчивости автомобиля возникает при сильном возрастании буксованияния его ведущих колес. Однако современные автомобили характеризуются низким расположением центра масс, что способствует буксированию автомобилей на уклонах, которые значительно меньше по кривизне, чем уклоны, на которых возможно продольное опрокидывание. Тем не менее, для коротко базовых автомобилей повышенной проходимости с высоким рас-

положением центра тяжести, при езде в горной местности продольная потеря устойчивости все же возможна.

С точки зрения сохранения высокой скорости при движении в кривых, интерес представляет именно поперечная устойчивость автомобиля. Показатели оценки устойчивости автомобиля представляют собой критические значения параметров положения и движения автомобиля. При воздействии на автомобиль боковых сил, для анализа его устойчивости используют следующие показатели [1]:

- критическая скорость по боковому скольжению и критическая скорость по боковому опрокидыванию;
- критическая скорость по траекторной и курсовой устойчивости;
- коэффициент поперечной устойчивости;
- критический угол косогора по боковому скольжению и критический угол по боковому опрокидыванию;
- угол крена;
- угол дрейфа.

Для обеспечения устойчивости автомобилей, все выше приведенные параметры нормируются согласно ГОСТ Р 52302-2004, ЕЭК ООН №107 и ОСТ 37. 001. 487-89 [1].

## 1.2 Траектории

Траектория движения объекта является последовательностью точек в многомерном евклидовом пространстве в разные моменты времени [2]. Значение каждой оси пространства является значением одного из параметров, которые имеют значение при сравнении двух траекторий. На изображении 1.1 изображен пример двух двумерных траекторий.

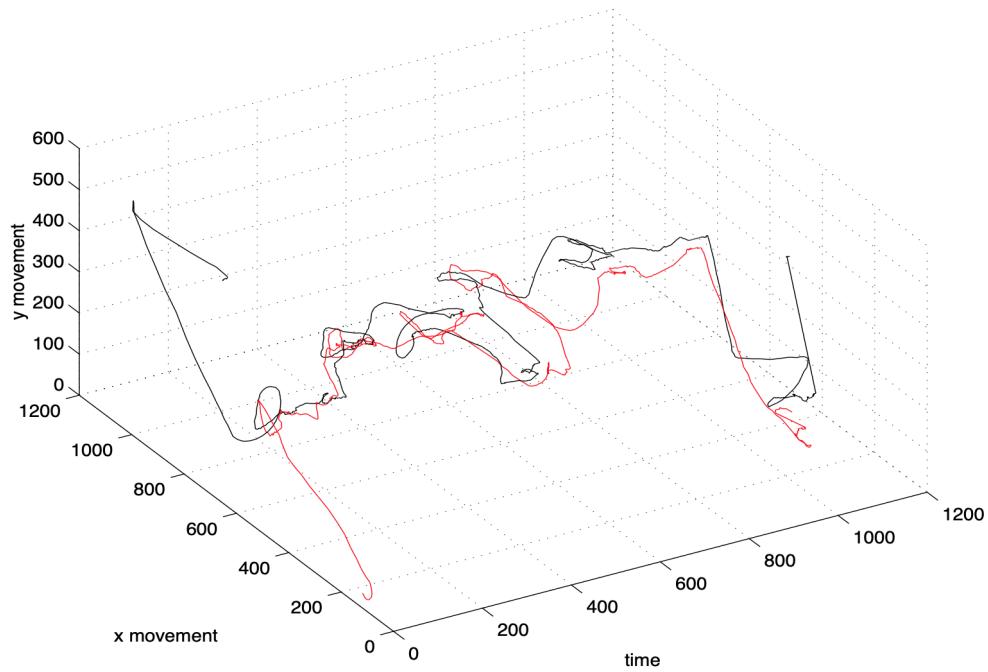


Рисунок 1.1 — Пример двумерных траекторий

### 1.3 Метод сравнения многомерных траекторий

#### 1.3.1 Существующие аналоги

##### Kartchrono

Система хронометража для картинга и авто-мотоспорта – Kartchrono устанавливается на картодромах/автодромах и позволяет пилотам получать необходимую информацию о временах кругов и секторов (сектор – деление круга на части). Эта информация необходима пилотам и тренерам для оттачивания своего мастерства. Приложение для смартфона, разработанная kartchrono, позволяет пилотам во время заезда видеть и слышать информацию о улучшении/ухудшении результатов в режиме реального времени [3].

Главное преимущество данной системы – бесплатное использование для пилотов и тренеров [3]. Но она дает только информацию о времени кругов и секторов [3]. Что в свою очередь не позволяет самостоятельно заниматься полноценным обучением. Необходимо присутствие тренера, который должен искать ошибки на трассе (траектория, торможения и т.д.).

## **MyChron5**

MyChron работает с помощью 2 датчиков GPS и не нуждается в сторонних системах и магнитных полосах на трассах [4]. MyChron5 более точен, чем большинство существующих систем GPS, поскольку он был разработан для добавления к сигналу спутника глобальной навигационной спутниковой системы (ГЛОНАСС). Имея в среднем почти двадцать спутников, работающих совместно, MyChron5 гарантирует абсолютную точность [4].

MyChron имеет много преимуществ таких как [4]:

- среднее значение допуска менее одного метра, что означает абсолютную точность определения положения картины на трассе,
- спутниковый сигнал активируется очень быстро. Через несколько секунд после включения MyChron5 GPS готов к работе,
- нет риска пропустить сигнал, так как - в случае "шума" или помех в одной датчике - другой датчик гарантирует непрерывность сигнала.

Помимо GPS-датчиков в системе MyChron5 используются еще [4]:

- датчики температуры,
- датчики давления,
- датчики положения,
- датчики скорости,
- радиомаяковое оборудование,
- датчики RPM.

## **Alfano 6**

Alfano 6 – устройство для регистрации данных с картов, к которому можно подключить до 12 устройств для считывания данных [5].

Программное обеспечение, идущее с данным устройством имеет следующий функционал [5]:

- отображение времени прохождения круга и разделения, максимальных и минимальных значений основных датчиков, а также сравнение выбранного круга с другими кругами в виде гистограммы. Также есть доступ к теоретическому кругу;
- возможность изменять амплитуду срезов оборотов в минуту, видеть зоны ускорения и замедления, менять цвет, кумулировать срезы оборотов в минуту;
- анализ коробки передач;
- анализ мощности двигателя;
- синхронизация с видео с камеры;
- анализ данных в графическом виде, т.е. визуализация всех образцов датчиков, расположения трека, силы  $G$ , промежутка между различными кругами различных сессий.

MyChron5 и Alfano 6 действительно позволяют находить ошибки в заездах без помощи тренера, рассматривая большое число данных, которые регистрируются при помощи датчиков, но данные решения являются платными [4] [5].

### 1.3.2 Фильтрация

Прежде чем работать с траекториями, необходимо отфильтровать те данные, в которых могут возникнуть шумы, например, из-за неточности считывающего устройства или кратковременных аномалий. Рассмотрим методы фильтрации.

#### Метод средних значений

Метод средних значений является одним из самых простых методов фильтрации шума, суть которого заключается в следующем: на каждом шаге  $k$ , значение  $v_k$  вычисляется как среднее из  $n$  предыдущих значений, то есть

$v_k = \frac{\sum_{i=0}^n a_{k-i}}{n}$ . Такой метод в среднем дает неплохое сглаживание, но имеет один значительный недостаток – задержку в значениях.

На рисунке 1.2 продемонстрирован пример фильтрации методом средних значений.

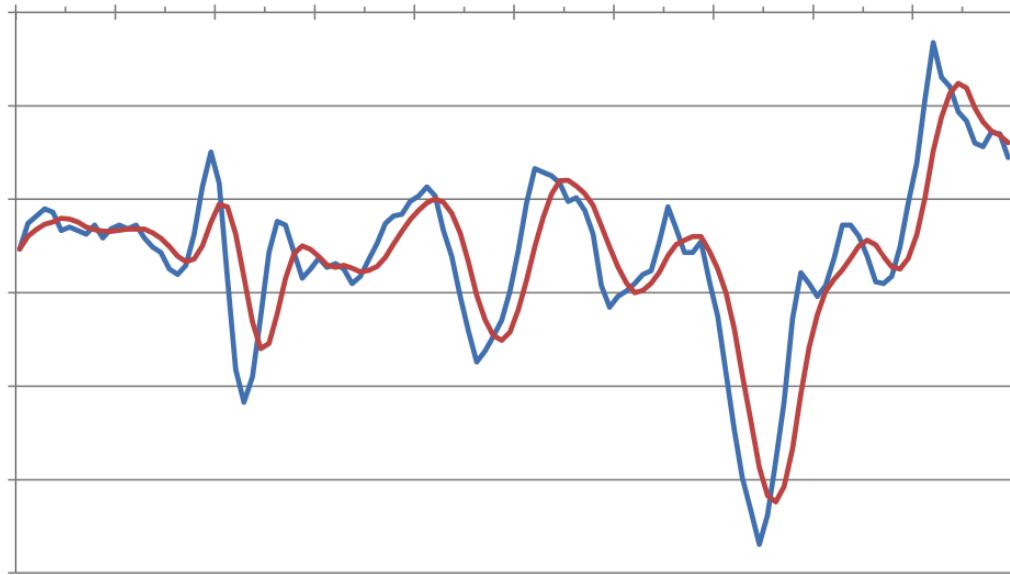


Рисунок 1.2 – Пример работы метода средних значений, синее – шумовые значения, красное – отфильтрованные

### Фильтр низких частот

Фильтры низких частот – набор фильтров, общей особенностью которых является возможность фильтрации сигналов выше заданной частоты, т.е. фильтр пропускает через себя низкочастотный сигнал, что позволяет избавиться от шумовых помех сигнала [6].

Простейший фильтр низких частот описывается формулой 1.1, в которой  $O$  – отфильтрованное значение сигнала,  $I_n$  – исходное значение,  $\alpha$  – коэффициент фильтрации, принимающий значения в диапазоне от 0 до 1 [6].

$$O_n = O_{n-1} + \alpha(I_n - O_{n-1}) \quad (1.1)$$

На рисунке 1.3 продемонстрирован пример фильтрации фильтром низких частот.

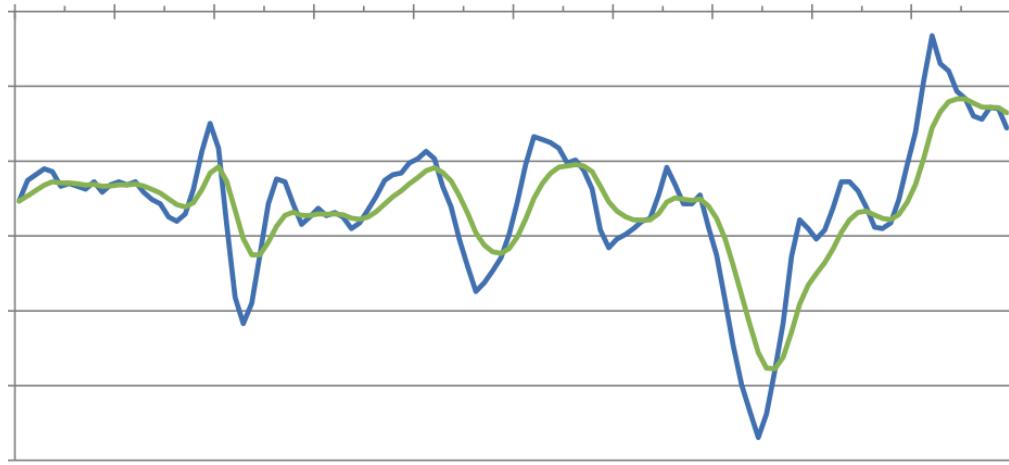


Рисунок 1.3 — Пример работы фильтра низких частот при  $\alpha = 0.25$ , синее — шумовые значения, зеленое — отфильтрованные

### Модифицированный фильтр низких частот

Для того чтобы уменьшить задержку в случае резких колебаний, необходимо ввести фильтр, зависящий от приращения  $n$ -го и  $n - 1$ -го значений сигнала. Необходимо подобрать такое пороговое значение  $\varepsilon$ , что при истинности условия  $|I_n - I_{n-1}| < \varepsilon$  отфильтрованное значение сигнала оказывается равным исходному [6].

На рисунке 1.4 продемонстрирован пример фильтрации модифицированным фильтром низких частот.

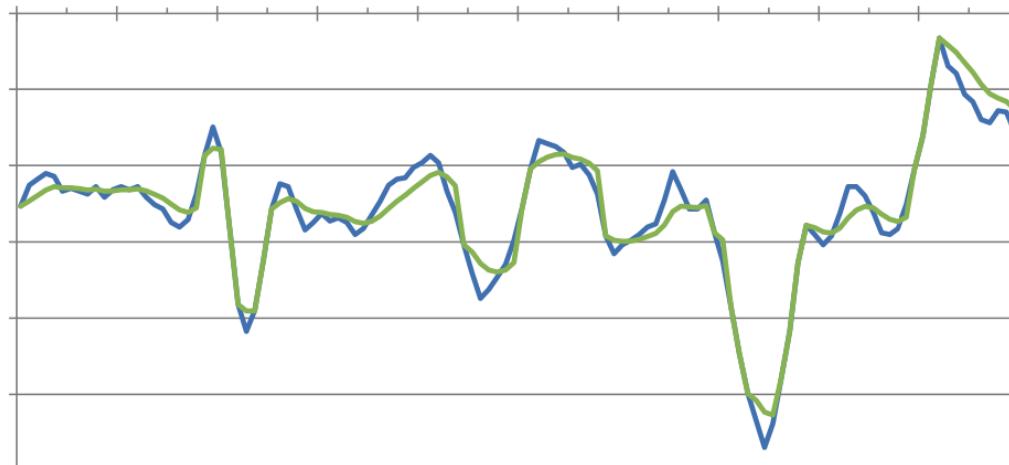


Рисунок 1.4 — Пример работы модифицированного фильтра низких частот при, синее — шумовые значения, зеленое — отфильтрованные

## Фильтр Калмана

Фильтры Калмана часто используются для фильтрации значений различных типов сигналов, которые можно найти в GPS-приемниках, обработчиках показаний датчиков и т.д. [7].

Фильтр Калмана – это эффективный рекурсивный фильтр, реализованный во временном представлении. Он позволяет оценивать сразу все данные, имеющиеся в системе. Алгоритм работает в два этапа. Первый этап – это прогнозирование следующего значения системы. Прогнозирование происходит при помощи экстраполяции. На втором этапе происходит уточнение предсказанного значения с помощью реального [7].

Для вычисления текущего значения фильтру Калмана необходима оценка системы на предыдущем шаге, а именно состояние системы и оценка погрешности определения этого состояния) [7].

Состояние фильтра задается двумя величинами:  $\hat{x}_{k|k-1}$  и  $P_{k|k}$ , где первая это оценка вектора состояния системы в момент времени  $k$ , а вторая – ковариационная матрица ошибок, которая задает оценку точности предсказания вектора состояния, а так же включает в себя оценку дисперсий погрешности и ковариации, показывающие взаимосвязи между параметрами состояния системы [7].

### 1. Этап предсказания [7]

На формуле 1.2 указана экстраполяция вектора состояния системы, в которой используется оценка системы с предыдущего шага.

$$\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} + B_k u_k \quad (1.2)$$

На формуле 1.3 указан расчет ковариационной матрицы для предсказанного значения на текущем шаге.

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k \quad (1.3)$$

### 2. Этап корректировки [7]

Расчет отклонения наблюдения на шаге  $k$  от предсказанного значения рассчитывается по формуле 1.4.

$$\tilde{y}_k = z_k - H_k \hat{x}_{k|k-1} \quad (1.4)$$

Ковариационная матрица для вектора ошибки рассчитывается по формуле 1.5.

$$S_k = H_k P_{k|k-1} H_k^T + R_k \quad (1.5)$$

На основании полученных ковариационных матриц можно получить матрицу коэффициентов усиления, формула расчета которой продемонстрирована на формуле 1.6.

$$K_k = P_{k|k-1} H_k^T + R_k \quad (1.6)$$

Получение фильтрованной оценки состояния системы на текущем шаге  $k$ , выполняющееся посредством корректировки ранее экстраполированного (предсказанного) значения, выполняется по формуле 1.7.

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k \quad (1.7)$$

Расчет ковариационной матрицы оценки вектора состояния системы указан на формуле 1.8.

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \quad (1.8)$$

На рисунке 1.5 продемонстрирован пример фильтрации фильтром Калмана.

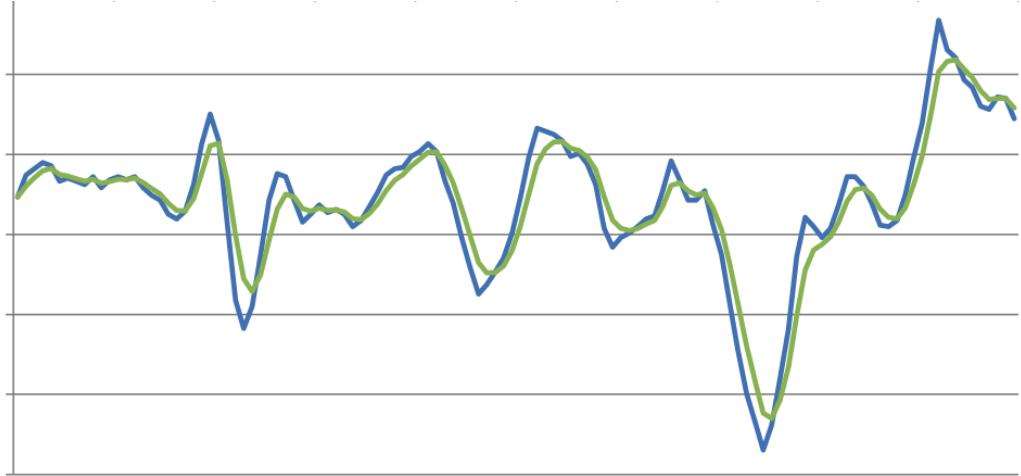


Рисунок 1.5 – Пример работы фильтра Калмана, синее – шумовые значения, зеленое – отфильтрованные

### Выбор алгоритма фильтрации

Исходя из проведенного анализа алгоритмов фильтрации был выбран фильтр Калмана.

#### 1.3.3 Преобразование в непрерывную траекторию

При сравнении траекторий может возникнуть ситуация, при которой ближайшая точка с эталонной траектории будет слишком далеко или будет использована при сравнении с другой точкой, поэтому для увеличения точности сравнения необходимо дискретную эталонную траекторию превратить в непрерывную.

#### Линейная интерполяция

Самым простым методом интерполяции является линейная интерполяция. Построение промежуточных значений происходит соединением соседних точек прямой с помощью уравнения прямой через две точки [8].

Для построения промежуточных значений используется уравнение прямой. Для двух точек  $(x_0; y_0)$  и  $(x_1; y_1)$  и точки  $(x; y)$ , в которой  $x_0 \leq x \leq x_1$ , справедливо уравнение 1.9 [8].

$$\frac{y - y_0}{x - x_0} = \frac{y_1 - y_0}{x_1 - x_0} \quad (1.9)$$

Из уравнения 1.9 можно выразить  $y$  через  $x$  и получить 1.10.

$$y = y_0 + (x - x_0) \frac{y_1 - y_0}{x_1 - x_0} = \frac{y_0(x_1 - x) + y_1(x - x_0)}{x_1 - x_0} \quad (1.10)$$

На рисунке 1.6 представлен пример линейной интерполяции.

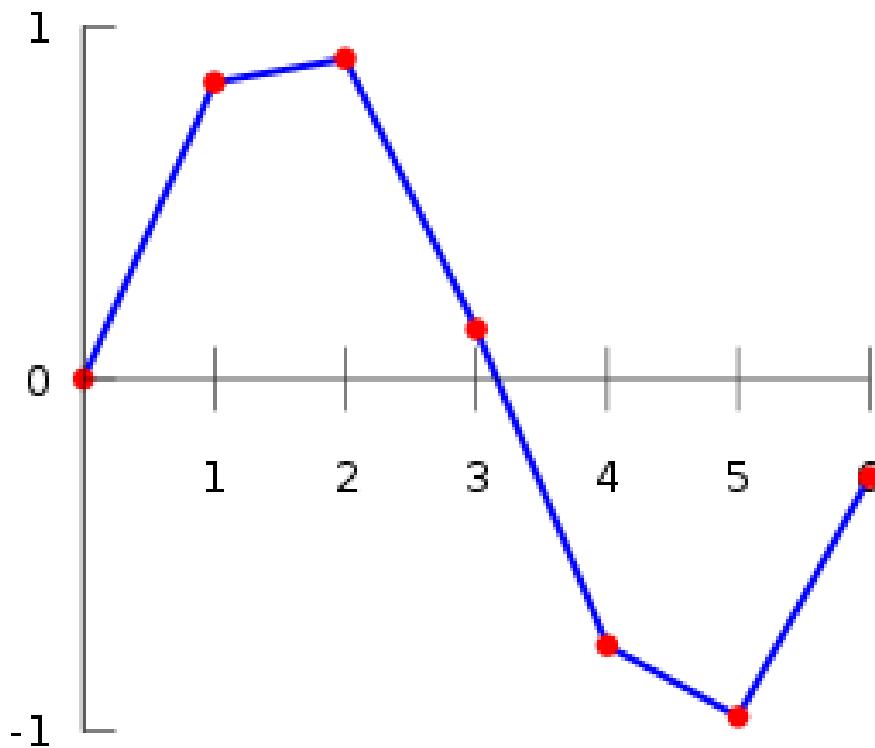


Рисунок 1.6 — Пример линейной интерполяции

## Интерполяция сплайнами

Интерполяция сплайнами является формой интерполяции, в которой интерполированные значения являются особым типом кусочного полинома, называемого сплайном [9].

Для отрезка  $[x_i; x_{i+1}]$  при  $i = \overline{1, N}$  функция  $S(x)$  является полиномом третьей степени  $S_i(x)$  и вычисляется по формуле 1.11 [9].

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 \quad (1.11)$$

Из формулы 1.11 получаем 1.12 [9].

$$\begin{aligned} S_i(x_i) &= a_i \\ S'_i(x_i) &= b_i, \quad t = \overline{1, N} \\ S''_i(x_i) &= 2c_i \\ S'''_i(x_i) &= 6d_i \end{aligned} \quad (1.12)$$

Для удобства примем обозначения на формулах 1.13, 1.14 [9].

$$h = x_i - x_{i-1}, i = \overline{1, N} \quad (1.13)$$

$$f_i = f(x_i), i = \overline{0, N} \quad (1.14)$$

Окончательные уравнения для расчета коэффициентов приведены на формулах 1.15, 1.16, 1.17 и 1.18 [9].

$$a_i = f_i \quad (1.15)$$

$$d_i = \frac{c_i - c_{i-1}}{3 \cdot h_i} \quad (1.16)$$

$$b_i = \frac{a_i - a_{i-1}}{h_i} + \frac{2 \cdot c_i + c_{i-1}}{3} \cdot h_i \quad (1.17)$$

$$c_{i-1} \cdot h_i + 2 \cdot c_i \cdot (h_i + h_{i-1}) + c_{i-1} \cdot h_{i+1} = 3 \cdot \left( \frac{a_{i+1} - a_i}{h_{i+1}} - \frac{a_i - a_{i-1}}{h_i} \right) \quad (1.18)$$

На рисунке 1.7 представлен пример интерполяции сплайнами, аналогичный тому, что был приведен для линейной интерполяции (1.6).

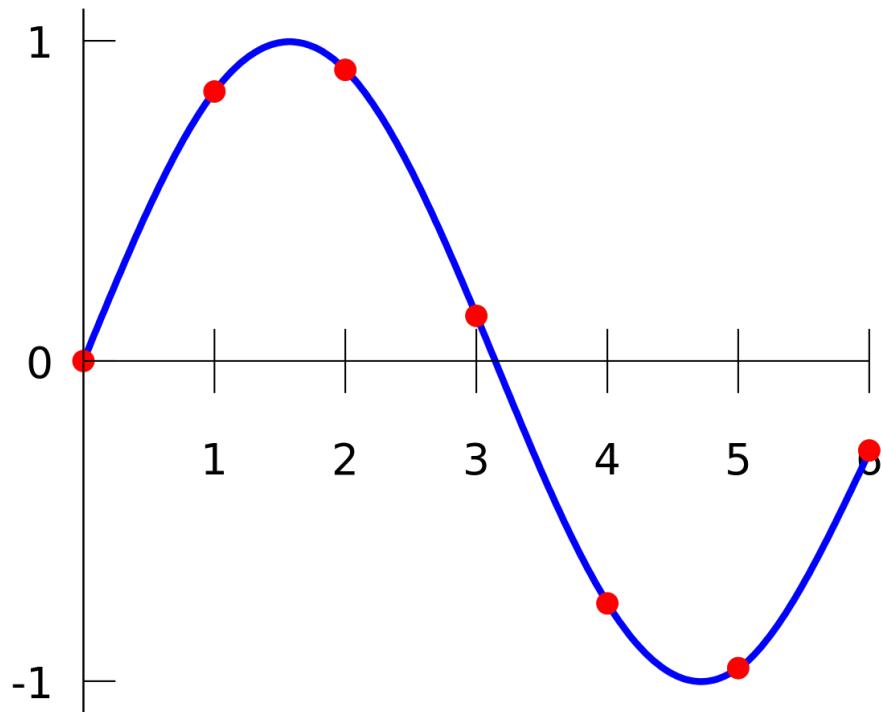


Рисунок 1.7 — Пример интерполяции сплайнами

## Аппроксимация

Аппроксимация или приближение – это алгоритм, в котором по набору точек из функции  $f(x)$  находится приближенная функция  $g(x)$ , исходя из различных параметров [10].

В качестве функции  $g(x)$  обычно выбирается полином, который называют интерполяционным полиномом [10]. На формуле 1.19 приведен классический полином.

$$y = \sum_{i=0}^n c_i \cdot x^i \quad (1.19)$$

На рисунке 1.8 представлен пример работы алгоритма аппроксимации.

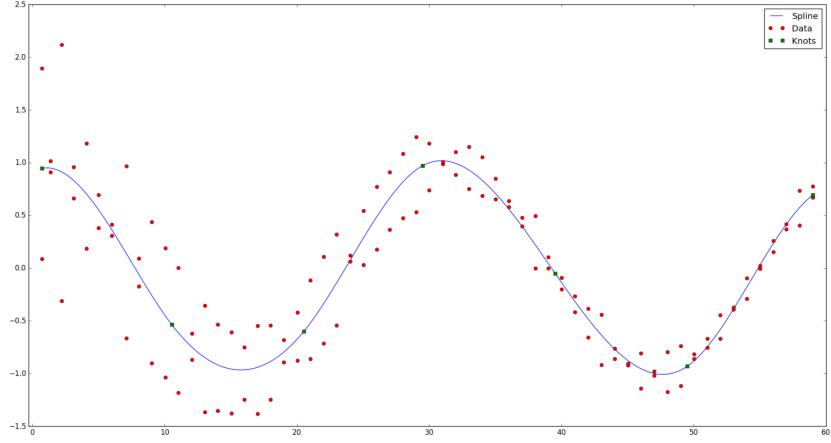


Рисунок 1.8 — Пример аппроксимации

## Выбор метода преобразования в непрерывную траекторию

В дальнейшем будет проведено исследование о применимости каждого из этих алгоритмов и будет выбран лучший из них.

### 1.3.4 Разделение значений на части с общей тенденцией

После расчета отставания в каждом моменте времени для сравниваемой траектории необходимо разделить ее на промежутки с общей тенденцией отставания, чтобы выделить участки, на которых отставание сокращалось или накапливалось.

## Алгоритмы на основе скользящих окон

Методы в этой категории разделяют данный временной ряд на окна фиксированного размера (подпоследовательности), чтобы локализовать причину исключения в одном или нескольких окнах. Мотивация такого подхода заключается в том, что смена тенденции во временном ряду может быть вызвана наличием одной или нескольких последовательностей исключений. Метод извлекает окно фиксированной длины ( $m$ ) (подпоследовательность) из временного ряда для обучения и тестирования, перемещая один или несколько

символов за раз. Оценка аномалии для временного ряда теста рассчитывается путем суммирования оценки аномалии для его окна [11].

Недостатком подхода, основанного на окнах, является то, что размер окна должен быть тщательно выбран, чтобы исключения могли быть четко обнаружены. Оптимальный размер окна зависит от длины области аномалии во временном ряду. Еще одним недостатком методов, основанных на использовании раздвижных окон, является их дороговизна. Поскольку сравнивается каждая пара окон тестирования и обучения, сложность равна  $O((nl)^2)$ , где  $l$  – средняя длина временного ряда, а  $n$  – количество временных рядов в наборе данных [11].

## Алгоритмы на основе прогнозирования

Идея этих методов заключается в том, что нормальный временной ряд генерируется из статистического процесса, а аномальный временной ряд не соответствует этому процессу. Поэтому ключевым компонентом является изучение параметров этого процесса из набора нормальных временных рядов для обучения, а затем оценка вероятности того, что данный временной ряд имеет те же характеристики, что и последовательность обучения [11].

Алгоритмы на основе предсказаний состоят из следующих двух этапов.

1. Модель обучается на  $t$  последовательных моментах времени, чтобы предсказать последующее  $t + 1$ -е значение.
2. Полученная модель применяется на дальнейших данных для тестирования обучения. Ошибка прогноза, соответствующая наблюдению, зависит от разницы между прогнозируемым значением, фактическим наблюдением и некоторыми параметрами модели таких как дисперсия модели.

Подобно подходу, основанному на скользящем окне, длина истории для обучения, выбранная здесь, важна для определения местоположения изменения тенденции. Поэтому, если длина истории  $t$  выбрана меньше длины цикла, производительность будет неудовлетворительной, а если  $t$  больше длины цикла, производительность улучшится. Но если значение  $t$  слишком

велико, то у нас есть очень большие размерные данные, что увеличивает вычислительную сложность [11].

## **Скрытые марковские модели**

Скрытая марковская модель – это статистическая модель, которая имитирует процесс, аналогичный марковскому процессу с неизвестными параметрами, с задачей решения неизвестных параметров на основе наблюдаемых параметров [12].

Основная идея методов этой группы заключается в том, что производится предположение о наличии скрытого марковского процесса в модели. При отсутствии этого процесса методы не смогут обнаруживать точки изменения тенденции [12].

## **Метод поиска сокращенного точного линейного времени**

Этот метод использует значение штрафа, чтобы определить количество точек изменения [13]. Таким образом, оптимальной сегментацией будет являться  $F(n)$ . Модель начинает вычисления с  $F(1)$ , а затем рекурсивно вычисляет  $F(2)$ ,  $F(3)$ , ...,  $F(n)$  [13]. При этом на каждой итерации сохраняется оптимальная сегментация. Когда модель достигает конца серии  $F(n)$ , то это значит была определена оптимальная сегментация для всех данных.

## **Выбор алгоритма**

Исходя из выводов, сделанных по каждому из алгоритмов, для кластеризации траектории по тенденции отставания будет использоваться метод поиска сокращенного точного линейного времени.

### **1.4 Вывод**

На основе проведенного анализа предметной области были выявлены основные этапы разрабатываемого метода, исследованы возможные подходы к

реализации каждого из этапов. Исходя из проведенного сравнительного анализа, для использования в разрабатываемом методе могут быть рекомендованы следующие алгоритмы.

1. Фильтрация исходных значений при помощи фильтра Калмана.
2. Для выбора алгоритма, для преобразование эталонной траектории из дискретной величины в непрерывную, будет проведено исследование.
3. Для поиска отставания в каждой точки сравниваемой траектории будет разработан собственный метод.
4. Для разделения последовательности отставаний на отрезки схожие по тенденции будет использоваться метод поиска сокращенного точного линейного времени.

## 2 Конструкторский раздел

### 2.1 Функциональная модель

На рисунке 2.1 представлена функциональная модель IDEF0 нулевого уровня.

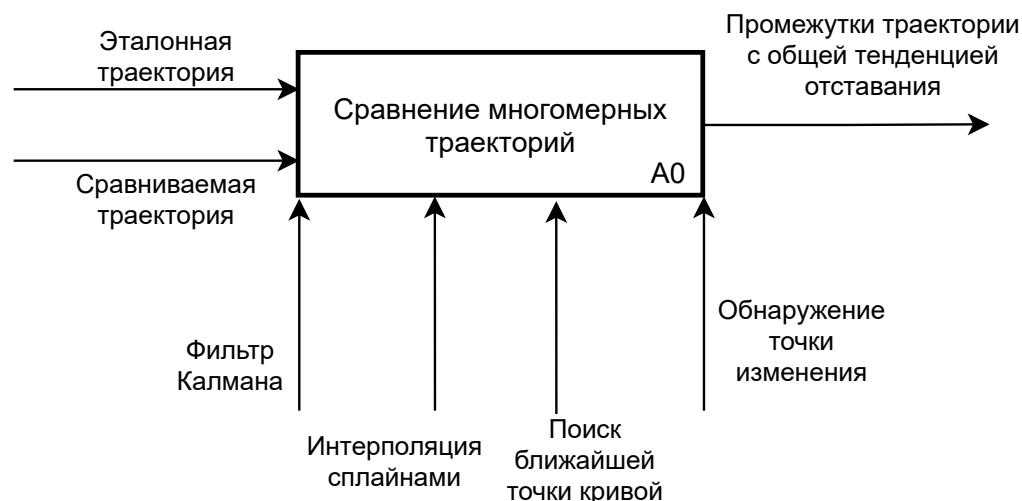


Рисунок 2.1 — Функциональная модель IDEF0 нулевого уровня

### 2.2 Описание метода

#### 2.2.1 Декомпозиция функциональной модели

На рисунке 2.2 представлена функциональная модель IDEF0 первого уровня.

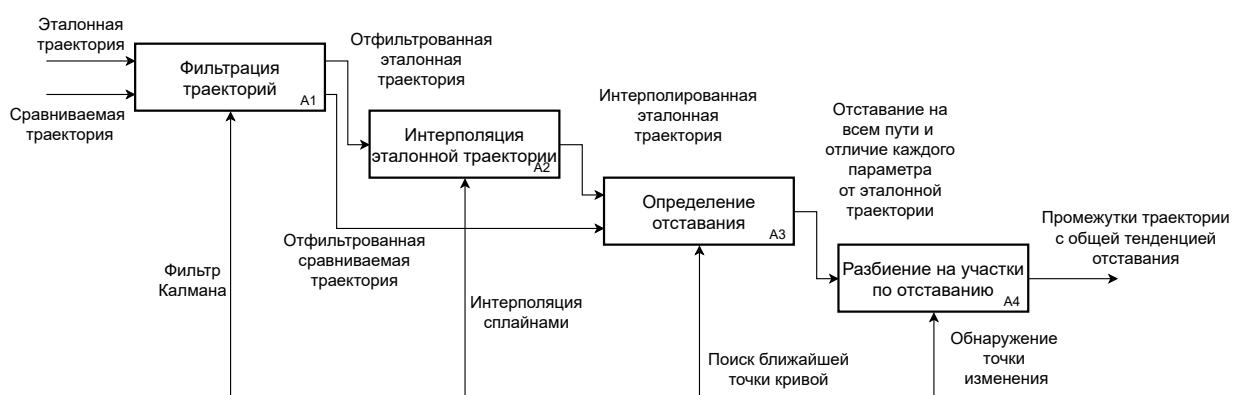


Рисунок 2.2 — Функциональная модель IDEF0 первого уровня

## 2.2.2 Фильтрация траекторий

Для устранения шумовых значений, возникающих впоследствии плохого соединения датчика gps со спутником, необходимо фильтровать значения, чтобы они не ухудшали точность работы алгоритма.

Для фильтрации значений был выбран фильтр Калмана. Схема данного алгоритма представлена на рисунке 2.3, на ней можно заметить исходные данные черного цвета и отфильтрованные зеленого.

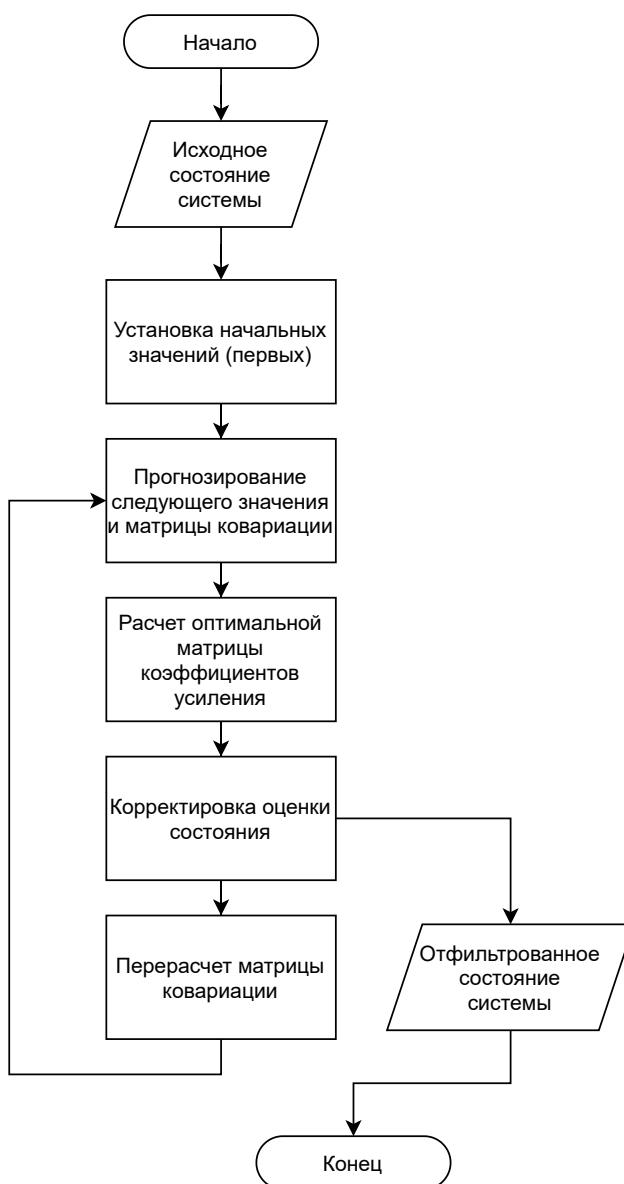


Рисунок 2.3 – Фильтр Калмана

Пример работы фильтра калмана на реальных данных представлен на рисунке 2.4.

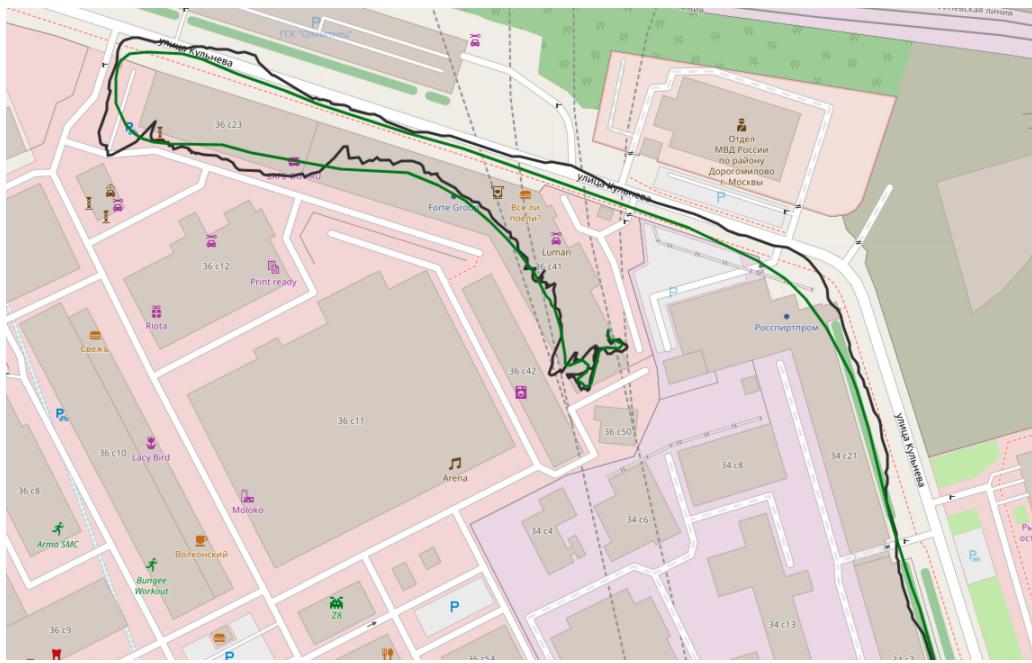


Рисунок 2.4 – Пример работы фильтра Калмана на реальных данных

### 2.2.3 Преобразование траектории в непрерывную

Для преобразования траектории из набора дискретных значений в непрерывную кривую стоит выбор между интерполяции сплайнами и аппроксимации сплайнами. Необходимо исследовать применимость каждого из двух алгоритмов для того, чтобы принять решение о выборе.

### 2.2.4 Расчет отставания

Для поиска отставания необходимо знать скорость движения автомобиля, но обычная скорость в данном случае не подойдет, так как на трассе возникает много ситуаций, при которых двигаясь с большей скоростью, все равно можно добраться медленнее. Например, на повороте проезд по внешнему кругу с большей скоростью затрачивает больше времени, чем по внутреннему. Поэтому введем скорость прогресса прохождения трассы.

Для каждой пары точек  $(t_i, x_i, y_i)$  и  $(t_{i+1}, x_{i+1}, y_{i+1})$ , где  $t$  – время в текущей точке,  $x$  – долгота,  $y$  – широта при  $i = \overline{1; N - 1}$  можно записать значение скорости прогресса трассы в текущей  $i$ -й точке (2.1).

$$\text{speed}(i) = \frac{\text{progress}(i)}{t_{i+1} - t_i}, \quad (2.1)$$

Значение прогресса  $\text{progress}(i)$  рассчитывается по формуле 2.2 и является значением в промежутке от 0 до 1, означающее на сколько много продвинулся гонщик за промежуток  $(i; i + 1)$  относительно длины всей трассы, таким образом вычисления не базируются на абсолютной скорости автомобиля.

$$\text{progress}(i) = \frac{\sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2}}{\text{length}} \quad (2.2)$$

Разница скоростей между эталонной траекторией и сравниваемой и будет отставанием. Помимо параметра отставания необходимо находить разность всех остальных параметров для дальнейшего изучения и поиска ошибок, совершенных пилотом.

На рисунке 2.5 представлена схема алгоритма расчета отставания.

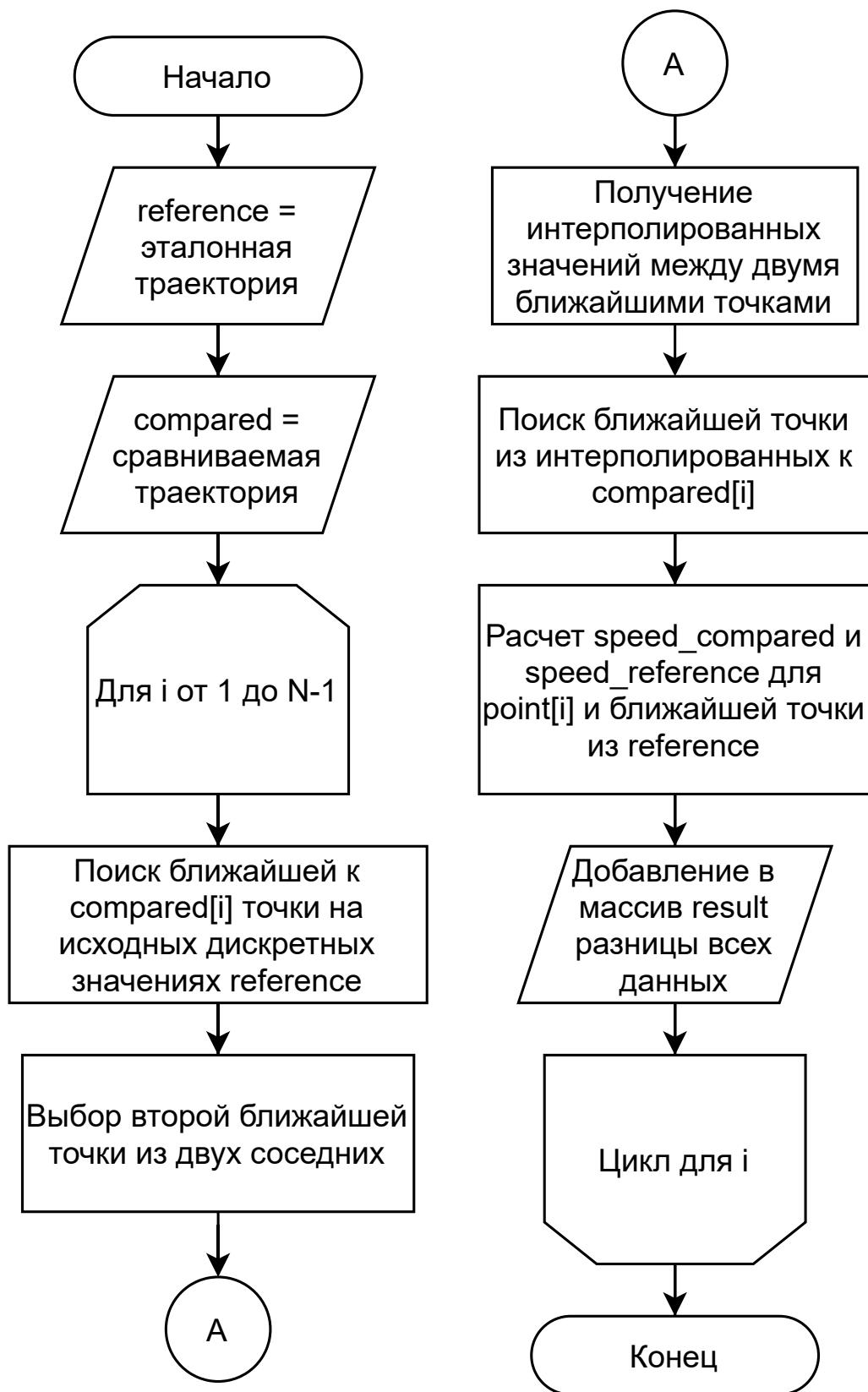


Рисунок 2.5 — Алгоритм расчета отставания

## 2.2.5 Деление отставания на промежутки со схожей тенденцией

Для выявления ошибок в заезде для пилота необходимо разделить его траекторию на промежутки, в которых происходило отставание или опережение с одинаковой тенденцией, то есть выделить точки, на которых происходит изменение скорости накапливания отставания, которая может быть даже отрицательной. Для данной задачи был выбран метод поиска сокращенного точного линейного времени.

Рассмотрим сам алгоритм.

**Ввод:**

- Набор данных  $(y_1, y_2, \dots, y_n)$ , где  $y_i \in R$
- Мера соответствия  $C(\cdot)$ , которая зависит от данных
- Константа штрафа  $\beta$ , которая не зависит от данных
- Константа  $K$ , удовлетворяющая  $C(y_{(t+1):s}) + C(y_{(s+1):T}) + K \leq C(y_{(t+1):T})$ , при  $t < s < T$ 
  1.  $n = \text{длина } y, F(0) = -\beta, cp(0) = \text{NULL}, R_1 = \{0\}$
  2. Цикл для  $\tau* = 1, \dots, n$ 
    - 2.1.  $F(\tau*) = \min_{\tau \in R_{\tau*}} [F(\tau) + C(y_{(\tau+1):\tau*}) + \beta]$
    - 2.2.  $\tau^1 = \arg\{\min_{\tau \in R_{\tau*}} [F(\tau) + C(y_{(\tau+1):\tau*}) + \beta]\}$
    - 2.3.  $cp(\tau*) = [cp(\tau^1), \tau^1]$
    - 2.4.  $R_{\tau*+1} = \{\tau \in R_{\tau*} \cup \{\tau*\} : F(\tau) + C(y_{(\tau+1):\tau*}) + K \leq F(\tau*)\}$

**Выход:** Точки изменения записаны в  $cp(n)$

## **2.3 Вывод**

Описанная в данном разделе структура метода позволяет выполнить его реализацию. Рассмотренные особенности использования выбранных алгоритмов не препятствуют реализации метода.

### **3 Технологический раздел**

#### **3.1 Средства реализации**

##### **3.1.1 Языки программирования**

Поскольку необходимо выводить результаты метода на карте, было принято решение об использовании клиент-серверной архитектуры с использованием web-интерфейсов для отображения карты с траекториями.

В качестве языка для реализации клиентской части, в которой происходит визуализация результата, был выбран самый популярный язык в этой области – JavaScript[14], имеющий большое число фреймворков и библиотек для разработки веб-приложений. JavaScript это мультипарадигменный язык программирования с динамической типизацией.

В качестве языка для реализации серверной части, на которой происходят все вычисления и воспроизводится разработанный алгоритм, был выбран относительно новый язык программирования Golang [15], разработанный компанией Google в 2009 году. Golang является компилируемым языком программирования со статической типизацией, созданный специально для разработки высоконагруженных сервисов.

##### **3.1.2 Используемые библиотеки и фреймворки**

###### **ReactJS**

В качестве основного фреймворка для разработки веб-приложения был выбран ReactJS [16], поскольку он является одним из самых популярных, простым и эффективным для использования.

###### **Leaflet**

Для отображения карты с введенными траекториями используется библиотека leaflet [17] для JavaScript, которая имеет модифицированную вер-

сию библиотеки специально для React [18], в которую входит множество компонентов для взаимодействия с библиотекой leaflet.

### gorilla/mux

Для роутинга и создания API на языке Golang была использована библиотека gorilla/mux [19], которая является оберткой для стандартной библиотеки Golang http, но реализующая больше возможностей без дополнительного написания кода.

## 3.2 Структура разработанного ПО

Для реализации программного обеспечения была спроектирована структура, представленная в виде UML-диаграмм компонентов на рисунке 3.1 для клиента и на 3.2 для сервера.

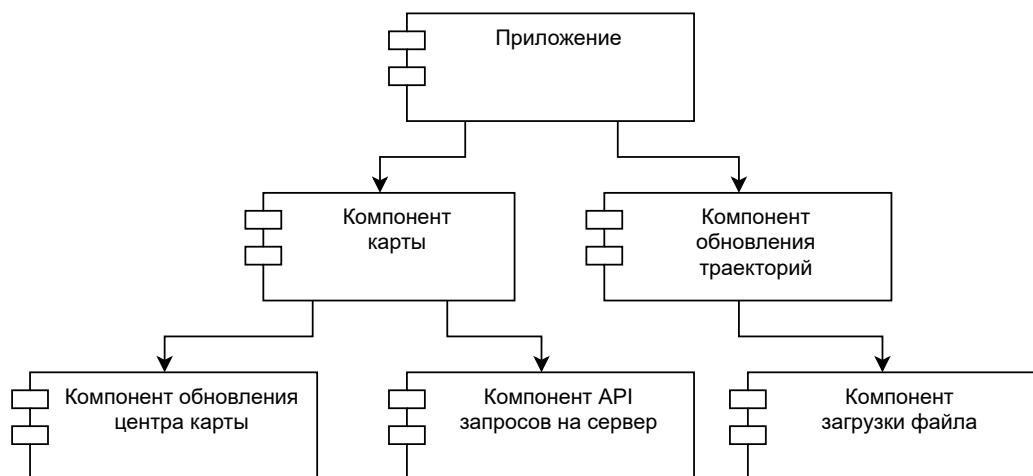


Рисунок 3.1 — UML-диаграмма компонентов для клиента

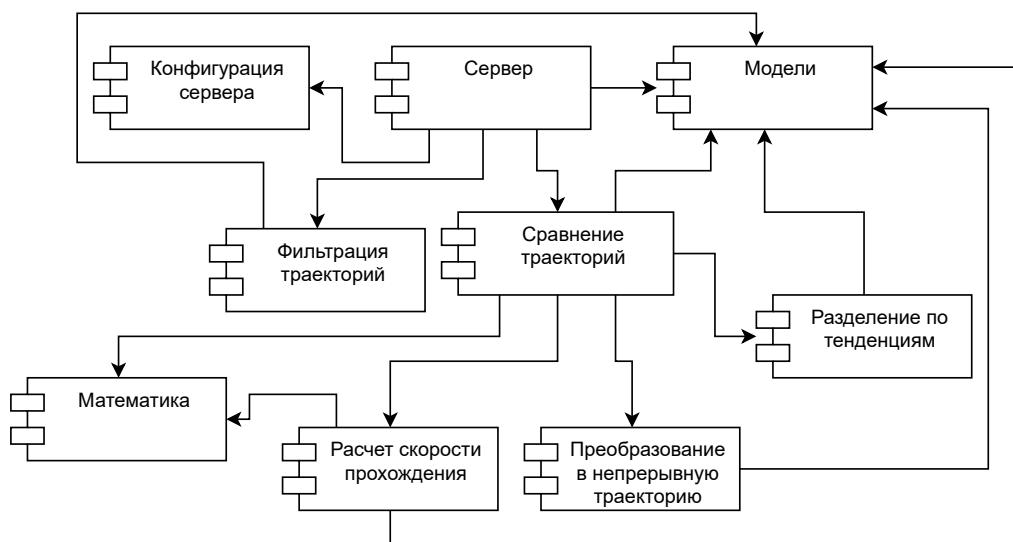


Рисунок 3.2 – UML-диаграмма компонентов для сервера

На листингах А.6 и А.7 продемонстрирована функция, отвечающая за нахождение отставания на сравниваемой траектории.

### 3.3 API для обращения клиента к серверу

Для передачи информации между клиентом и сервером были разработаны следующие модели, указанные на листингах А.1, А.2, А.3, А.4, А.5.

Для связи клиента и сервера было разработано следующее REST API [20].

– **/filter POST**

Тело – Модель Gps

Ответ – Модель Gps

– **/compare POST**

Тело – {"reference": Модель Trajectory, "compared": Модель Trajectory}

Ответ – Модель ComparedTrajectory

### 3.4 Формат входного файла

Для сравнения траекторий необходимо их загрузить в программу. Для загрузки траектории в программу используется формат json файла, в котором хранится объект модели Trajectory A.4.

### 3.5 Интерфейс программы

На рисунке 3.3 представлен интерфейс программы при ее запуске. На нем можно заметить карту, на которой будут отображаться загружаемые траектории и результат сравнения. Так же на правой части интерфейса видны две области для загрузки файлов в формате json, после загрузки карта автоматически переносится на место загруженной траектории.

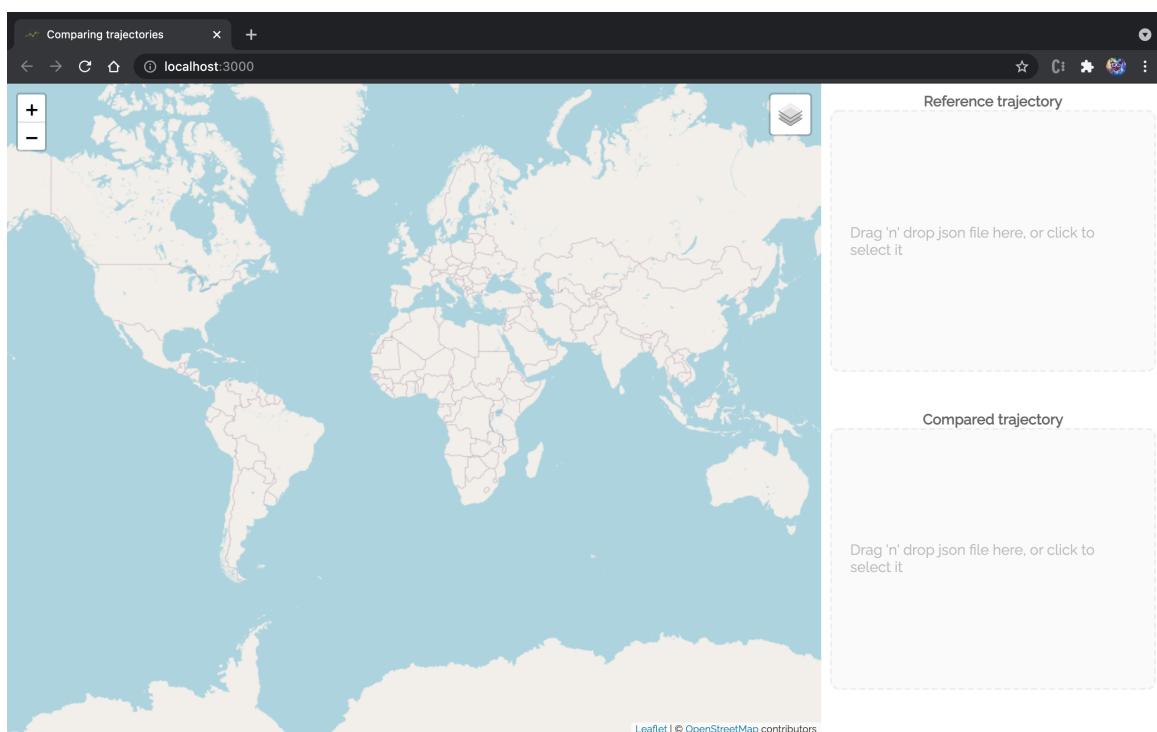


Рисунок 3.3 — Стартовый интерфейс программы

После загрузки одной из траекторий она отображается на карте сразу в отфильтрованном виде. Когда будут загружены обе траектории, то сравниваемая траектория будет разделена на промежутки разных цветов:

- зеленый – опережение эталонной траектории (прошел лучше),
- желтый – отставание не менялось (прошел так же),

- красный – оставание нарастает (прошел хуже).

На рисунке 3.4 представлен интерфейс с двумя загруженными траекториями.

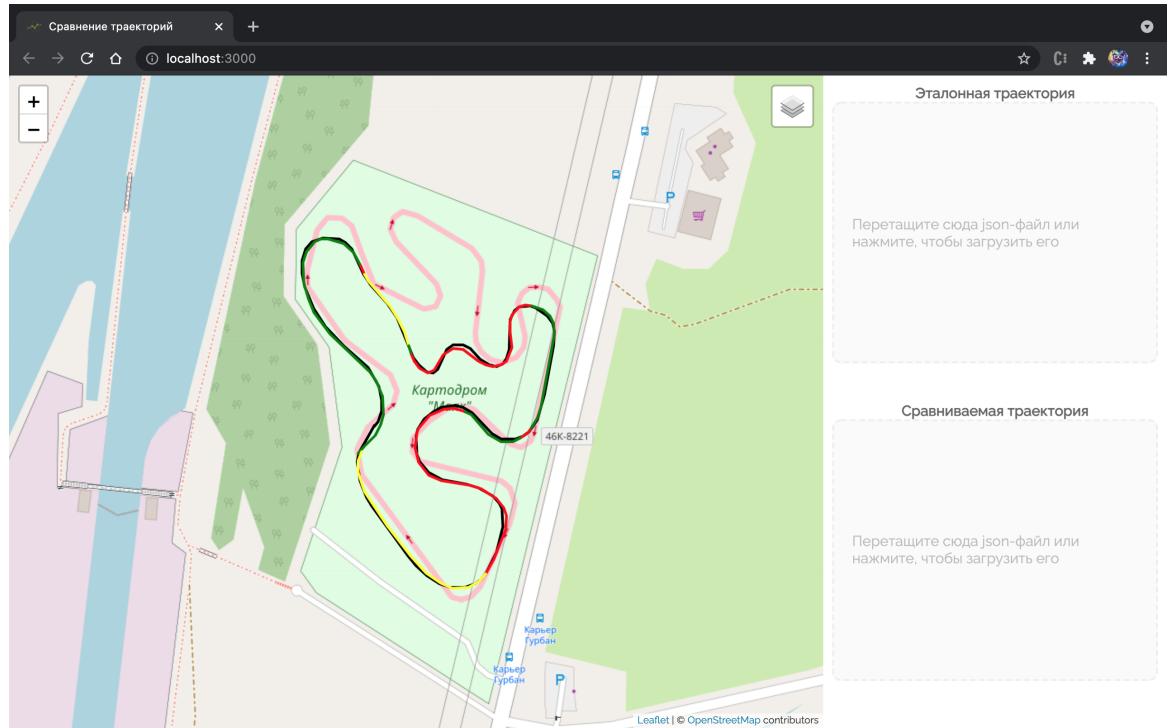


Рисунок 3.4 — Интерфейс программы с двумя загруженными траекториями

При наведении на каждый из промежутков можно увидеть дополнительную информацию о том, насколько сильно меняются разные параметры относительно эталонной траектории. На рисунке 3.5 представлен интерфейс при наведении на промежуток сравниваемой траектории.

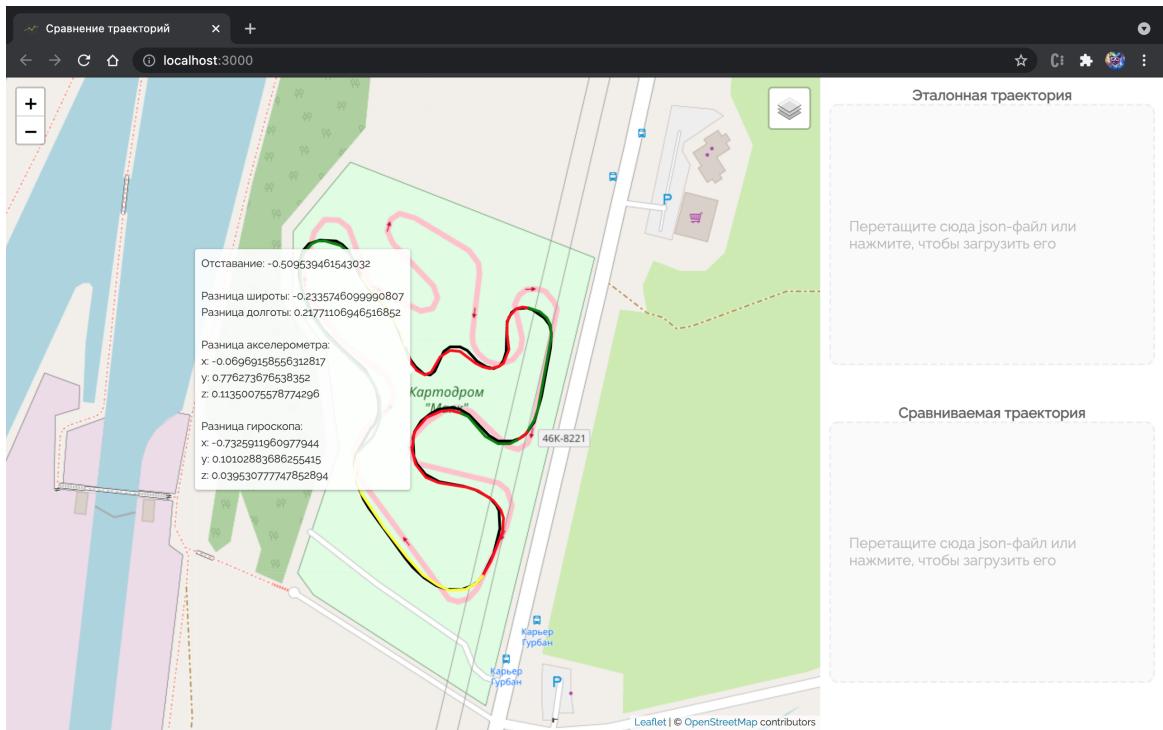


Рисунок 3.5 — Интерфейс программы с двумя загруженными траекториями

### 3.6 Вывод

В разделе проведены выбор средств реализации (Golang для алгоритма, ReactJS для демонстрации результатов), проектирование архитектуры программного обеспечения и его разработка. Помимо этого были описаны выбор библиотек и фреймворков для разработки, а также модели и API для связи клиента и сервера.

## **4 Исследовательский раздел**

### **4.1 Выбор алгоритма преобразования в непрерывную величину**

Так как нельзя однозначно сказать какой из двух алгоритмов преобразования траектории в непрерывную величину лучше, необходимо провести эксперимент, сравнивающий интерполяцию сплайнами и аппроксимацию по выбранной метрике.

### **4.2 Представление результата**

Результатом сравнения траекторий является сравниваемая траектория, разделенная на участки с общей тенденцией отставания от эталонной траектории. Зная численное значение отставания, можно разделить участки по трем категориям: участок, на котором отставание увеличивается (движение медленнее эталонного), участок, на котором отставание не меняется (движение примерно такое же быстрое как эталонное), и участок, на котором отставание сокращается (движение быстрее эталонного).

Таким образом результат можно представить в виде кластеризации точек сравниваемой траекторий по трем категориям.

### **4.3 Размеченные данные**

Поскольку результат метода был сведен к кластеризации, то можно рассматривать метрики классификации. В качестве размеченных данных была использована траектория, размеченная тренером, наблюдавшим за записанным в траекторию заездом. На рисунке 4.1 можно увидеть размеченную траекторию с большим числом кругов.

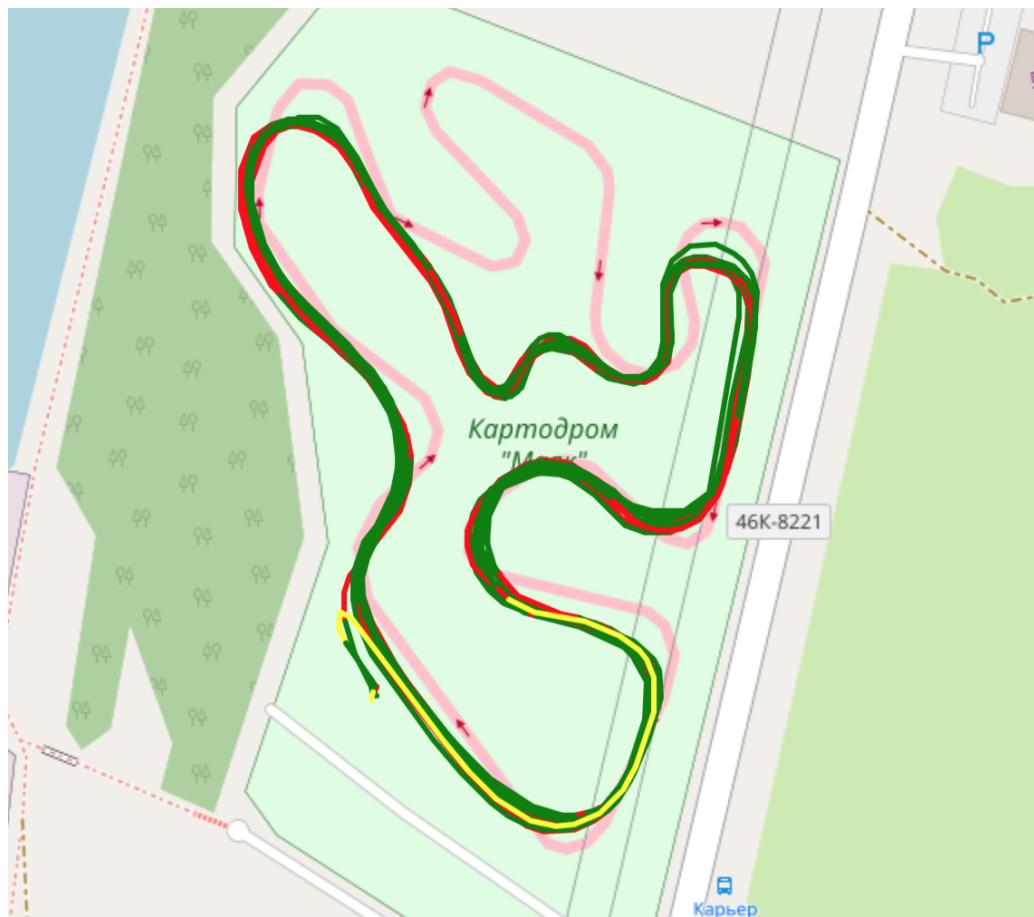


Рисунок 4.1 — Размеченная траектория

#### 4.4 Метрики применимости

В качестве метрик применимости были выбраны точность (precision) и полнота (recall). Точность в пределах типа отставания – это доля точек, действительно принадлежащих к этому типу. Полнота системы – отношение числа найденных точек, относящихся к этому типу, при помощи метода к числу всех точек данного типа в сравниваемой траектории.

В результате получаем четыре класса размеченных точек при помощи алгоритма.

- TP – истинно-положительное решение.
- TN – истинно-отрицательное решение.
- FP – ложно-положительное решение.
- FN – ложно-отрицательное решение.

На таблице 4.1 более наглядно продемонстрированы данные классы.

Таблица 4.1 — Классы решений метода

Тип отставания _i		Экспертная оценка	
Оценка метода	Положительная	Положительная	Отрицательная
	Отрицательная	TP	FP
		FN	TN

Тогда, относительно класса позитивных решений можно определить точность по формуле 4.1 и полноту по формуле 4.2.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (4.1)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (4.2)$$

Величина, равная среднему гармоническому от точности и полноты называется f-мерой (4.3) и характеризует точность определения одного класса точек [21].

$$F = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4.3)$$

Средним арифметическим f-мер всех классов будет эффективность классификатора в целом [21].

Результаты проведенного эксперимента можно увидеть на рисунке 4.2.

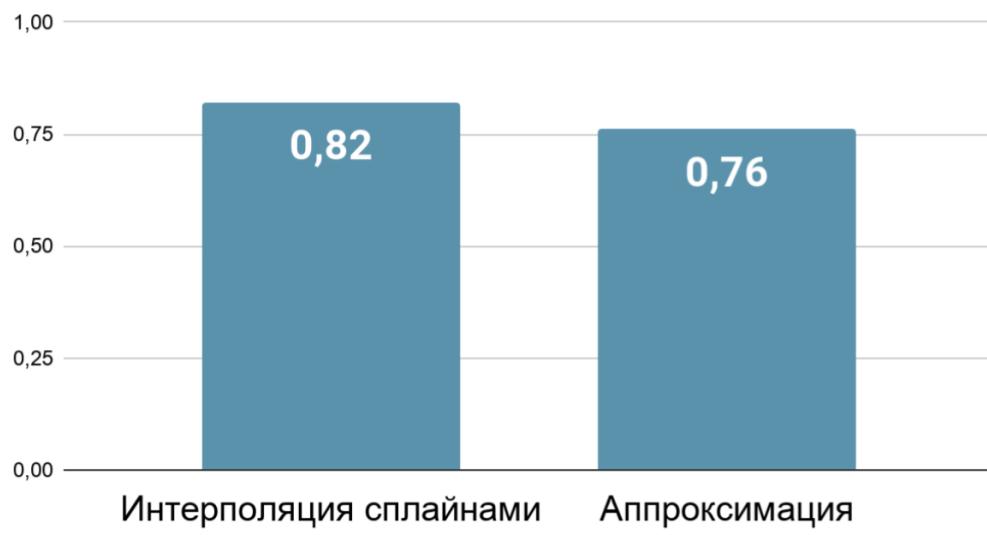


Рисунок 4.2 — Значение F-меры

#### 4.5 Вывод

В результате проведенного эксперимента можно сделать вывод о применимости разработанного алгоритма для сравнения многомерных траекторий.

Так же были рассмотрены два алгоритма преобразования дискретной траектории в непрерывную величину на применимость в методе, а именно интерполяция сплайнами и аппроксимация. Эксперимент показал, что оба метода могут быть применены в методе, но лучше работает именно метод интерполяции сплайнами, который и будет в дальнейшем применяться.

## **ЗАКЛЮЧЕНИЕ**

Таким образом был разработан и реализован метод сравнения многомерных траекторий на примере кольцевых гонок.

В ходе работы были выполнены следующие задачи:

- а) проанализированы подходы к сравнению набора точек в многомерном пространстве;
- б) реализован разработанный метод;
- в) визуализирован результат работы метода;
- г) оценена применимость метода.

Для дальнейшего развития работы планируется:

- реализовать математический расчет идеальной траектории;
- улучшить визуализацию результатов работы метода, добавить демонстрацию в разных измерениях;
- получить обратную связь от пилотов;
- реализовать возможность работы алгоритма в реальном времени.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. В.П. Тарасик. Теория движения автомобиля: учебник для вузов. — СПб: БХВ-Петербург, 2006.
2. Michail Vlachos George Kolios Dimitrios Gunopoulos. Discovering Similar Multidimensional Trajectories. — IEEE, 2002.
3. KartChrono. — Режим доступа: <https://kartchrono.com> (дата обращения: 2020-05-22).
4. MyChron5. — Режим доступа: <https://www.aimtechnologies.com/kart/aim-mychron-5/> (дата обращения: 2020-05-22).
5. Alfano. — Режим доступа: <https://www.alfano.com/en/> (дата обращения: 2020-05-22).
6. Gery Casiez Nicolas Roussel3, Vogel Daniel. 1e Filter: A Simple Speed-based Low-pass Filter for Noisy Input in Interactive Systems. — 2012.
7. Youngjoo Kim Hyochoong Bang. Introduction to Kalman Filter and Its Applications. — InTechOpen, 2018.
8. Needham Joseph. Mathematics and the Sciences of the Heavens and the Earth. — Cambridge University Press., 1959. — Т. 3. — ISBN: 978-0-521-05801-8.
9. В.Е. Герцман А.-В. И. Середа. Аппроксимация траекторных рядов с использованием кубических В-сплайнов. — Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина), 2019.
10. Kidron Ivy. Polynomial Approximation of Functions: Historical Perspective and New Tools. — International Journal of Computers for Mathematical Learning, 2003.
11. Викторович Соболев Константин. Автоматический поиск аномалий во временных рядах. — Московский физико-технический институт (государственный университет), 2018.

12. Farzad Eskandanian Bamshad Mobasher. Detecting Changes in User Preferences using Hidden Markov Models for Sequential Recommendation Tasks. — 2018.
13. The Power of the Pruned Exact Linear Time(PELT) Test in Multiple Changepoint Detection. — American Journal of Theoretical and Applied Statistics, 2015.
14. JavaScript. — Режим доступа: <https://www.javascript.com> (дата обращения: 2020-05-28).
15. Golang. — Режим доступа: <https://golang.org> (дата обращения: 2020-05-28).
16. ReactJS. — Режим доступа: <https://ru.reactjs.org> (дата обращения: 2020-05-28).
17. Leaflet. — Режим доступа: <https://leafletjs.com> (дата обращения: 2020-05-28).
18. React-Leaflet. — Режим доступа: <https://react-leaflet.js.org> (дата обращения: 2020-05-28).
19. Модуль mux для golang. — Режим доступа: <https://github.com/gorilla/mux> (дата обращения: 2020-05-28).
20. What is a REST API. — Режим доступа: <https://www.redhat.com/en/topics/api/what-is-a-rest-api> (дата обращения: 2020-05-28).
21. Yutaka Sasaki Research Fellow. The truth of the F-measure. — School of Computer Science, University of Manchester MIB, 2007.

## ПРИЛОЖЕНИЕ А

### ЛИСТИНГИ

Листинг А.1 — Точка в трехмерном пространстве

```
1 type Point struct {
2     X int `json:"x, string"`
3     Y int `json:"y, string"`
4     Z int `json:"z, string"`
5 }
```

Листинг А.2 — Данные с акселерометра и гироскопа

```
1 type Accelerometer struct {
2     Date [] ParsedDate `json:"date"`
3     Acc   [] Point      `json:"acc"`
4     Gyro  [] Point      `json:"gyro"`
5 }
```

Листинг А.3 — Данные с GPS-датчика

```
1 type Gps struct {
2     Date [] ParsedDate `json:"date"`
3     Long  [] float64   `json:"long"`
4     Lat   [] float64   `json:"lat"`
5 }
```

Листинг А.4 — Траектория

```
1 type Trajectory struct {
2     Gps        Gps        `json:"gps"`
3     Accelerometer Accelerometer `json:"acc"`
4 }
```

Листинг А.5 — Результат сравнения траекторий

```
1 type ComparedTrajectory struct {
2     Backlog    [] float64 `json:"backlog"`
3     Long       [] float64 `json:"long"`
4     Lat        [] float64 `json:"lat"`
5     DeltaLong  [] float64 `json:"dlong"`
6     DeltaLat   [] float64 `json:"dlat"`
7     DeltaAcc   [] Point   `json:"dacc"`
8     DeltaGyro  [] Point   `json:"dgyro"`
9 }
```

## Листинг А.6 — Нахождение отставания траектории

```
1 func difference(perfect interpolation.InterpolatedTrajectory, compared
                  speed.SpeedTrajectory, comparedAcc model.Accelerometer) (ct model.ComparedTrajectory,
                  err error) {
2     ct = model.ComparedTrajectory{
3         Backlog: make([] float64, compared.Gps.Len()),
4         Long:     make([] float64, compared.Gps.Len()),
5         Lat:      make([] float64, compared.Gps.Len()),
6         DeltaLong: make([] float64, compared.Gps.Len()),
7         DeltaLat:  make([] float64, compared.Gps.Len()),
8         DeltaAcc:   make([] model.FloatPoint, compared.Gps.Len()),
9         DeltaGyro:  make([] model.FloatPoint, compared.Gps.Len()),
10    }
11    for i := 0; i < compared.Gps.Len(); i++ {
12        var minDist float64 = -1
13        var minIndex int
14        x1 := compared.Gps.Lat[i]
15        y1 := compared.Gps.Long[i]
16
17        for j := 0; j < perfect.Trajectory.Gps.Len(); j++ {
18            x2 := perfect.Trajectory.Gps.Lat[j]
19            y2 := perfect.Trajectory.Gps.Long[j]
20            dist := math.Distance(x1, y1, x2, y2)
21            if minDist == -1 || dist < minDist {
22                minDist = dist
23                minIndex = j
24            }
25        }
26
27        var secondIndex int
28        if minIndex == 0 {
29            secondIndex = 1
30        } else if minIndex == perfect.Trajectory.Gps.Len() - 1 {
31            secondIndex = minIndex - 1
32        } else {
33            x2 := perfect.Trajectory.Gps.Lat[minIndex - 1]
34            y2 := perfect.Trajectory.Gps.Long[minIndex - 1]
35            x3 := perfect.Trajectory.Gps.Lat[minIndex + 1]
36            y3 := perfect.Trajectory.Gps.Long[minIndex + 1]
37
38            dist1 := math.Distance(x1, y1, x2, y2)
39            dist2 := math.Distance(x1, y1, x3, y3)
40
41            if dist1 < dist2 {
42                secondIndex = minIndex - 1
43            } else {
44                secondIndex = minIndex + 1
45            }
46        }
47
48        speed, lat, long, acc, gyro := perfect.TakeValues(minIndex, secondIndex)
49
50        minDist = -1
51        for j := 0; j < len(lat); j++ {
52            x2 := lat[j]
53            y2 := long[j]
54            dist := math.Distance(x1, y1, x2, y2)
```

### Листинг А.7 – Нахождение отставания траектории (продолжение)

```
56         if minDist == -1 || dist < minDist {
57             minDist = dist
58             minIndex = j
59         }
60     }
61
62     accIndex := 0
63     var minDateDiff float64 = -1
64     for j := 0; j < comparedAcc.Len(); j++ {
65         dateDiff := m.Abs(model.DateDiffSeconds(compared.Gps.Date[i],
66                                         comparedAcc.Date[j]))
67         if minDateDiff == -1 || dateDiff < minDateDiff {
68             minDateDiff = dateDiff
69             accIndex = j
70         } else {
71             break
72         }
73     }
74     ct.Backlog[i] = speed[minIndex] - compared.Speed[i]
75
76     ct.Long[i] = compared.Gps.Long[i]
77     ct.Lat[i] = compared.Gps.Lat[i]
78
79     ct.DeltaLat[i] = lat[minIndex] - compared.Gps.Lat[i]
80     ct.DeltaLong[i] = long[minIndex] - compared.Gps.Long[i]
81
82     ct.DeltaAcc[i].X = float64(acc[minIndex].X -
83                                 comparedAcc.Acc[accIndex].X)
83     ct.DeltaAcc[i].Y = float64(acc[minIndex].Y -
84                                 comparedAcc.Acc[accIndex].Y)
84     ct.DeltaAcc[i].Z = float64(acc[minIndex].Z -
85                                 comparedAcc.Acc[accIndex].Z)
85
86     ct.DeltaGyro[i].X = float64(gyro[minIndex].X -
87                                 comparedAcc.Gyro[accIndex].X)
87     ct.DeltaGyro[i].Y = float64(gyro[minIndex].Y -
88                                 comparedAcc.Gyro[accIndex].Y)
88     ct.DeltaGyro[i].Z = float64(gyro[minIndex].Z -
89                                 comparedAcc.Gyro[accIndex].Z)
90
91     return
92 }
```