**Разминка**

**#1** Выполнение задания мы начали с построения траектории движения по данным, которые вы любезно для нас выгрузили. Мы воспользовались вашим жирным намеком в постановке задачи и начали разбираться с данными Device Time, Longitude, Latitude, или Время, Долгота и Широта соответственно. С временем все относительно понятно, но за уточнением долготы и широты, чтобы ничего не напутать – обратились к гуглу. Из картинок (пример на рисунке 1) сразу стало понятно, что есть начальная точка отсчета обеих величин.

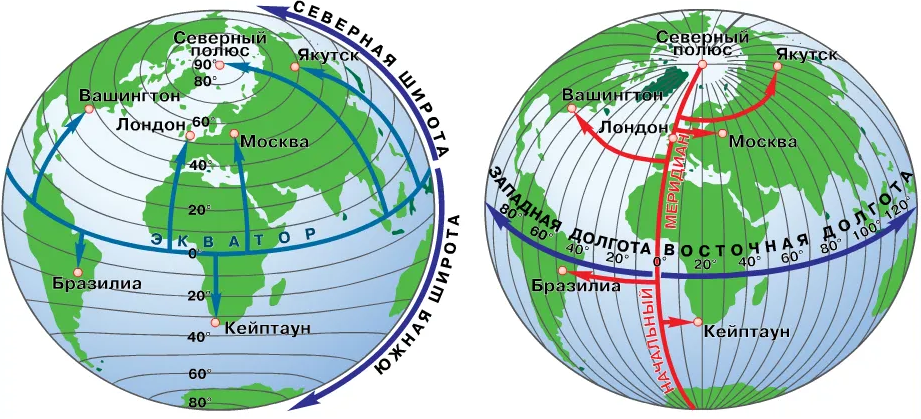


Рис. 1 – Широта и Долгота на глобусе

Уточнив то, что было понятно по 1му рисунку, мы двинулись дальше. В интернете было найдено огромное количество [сайт](https://www.wikiloc.com/outdoor-trails/data1-47957058)ов (это не реклама) которые были готовы нанести координаты на карту (рисунок 2) и фактически сделали всю грязную работу за нас. Ради приличия распишем, как можно высчитать координаты каждой точки вручную. Делим 40 000 км (длину меридиана) на 360 градусов и получаем в 1 градусе 111,111 км. У долготы учитывается широта местности. Математика делает невозможное и считает градус формулой (40000/360)\*cos(n), где n = градусу широты. В итоге была получена следуюшая траектория движения большого бензинового мобильного робота.

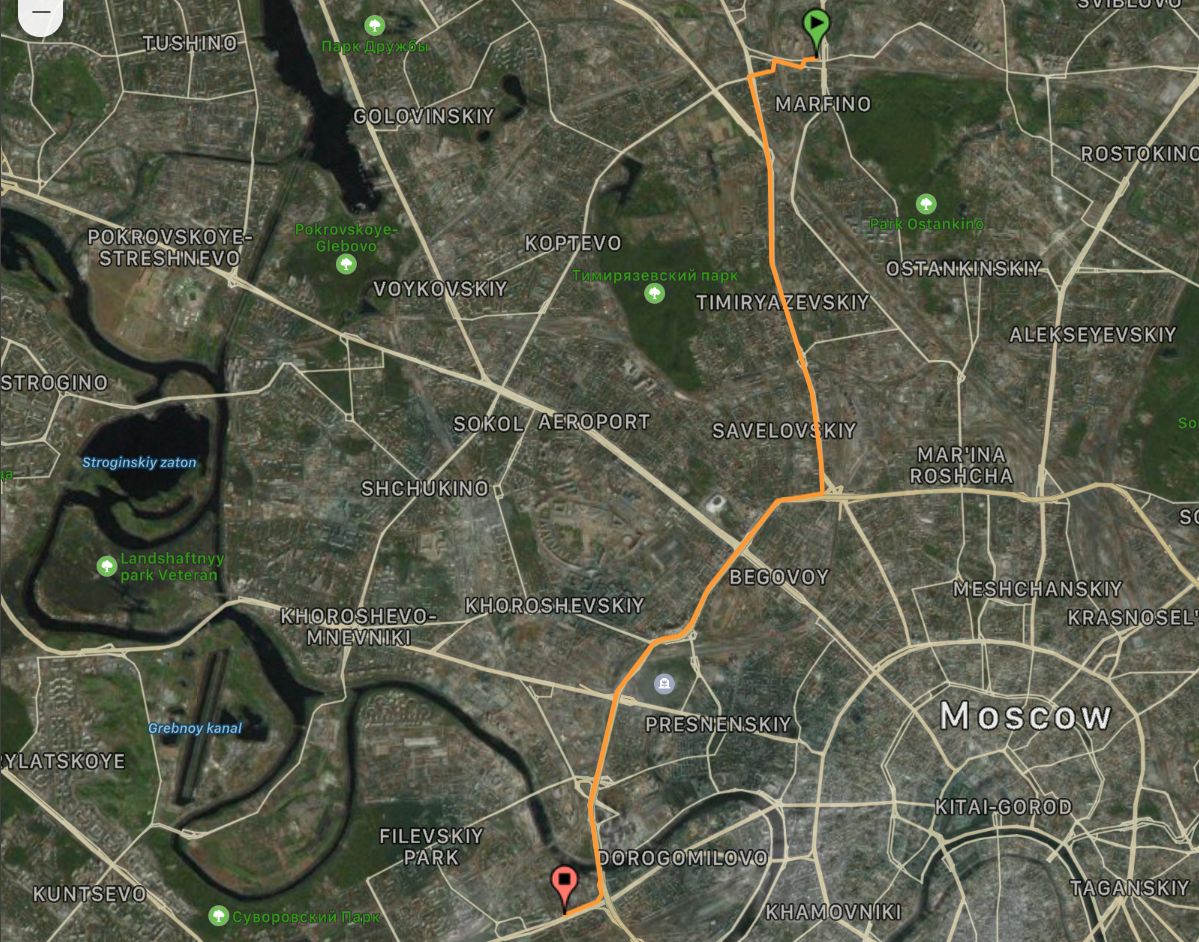


Рис. 2 – Наложение траектории движения на карту

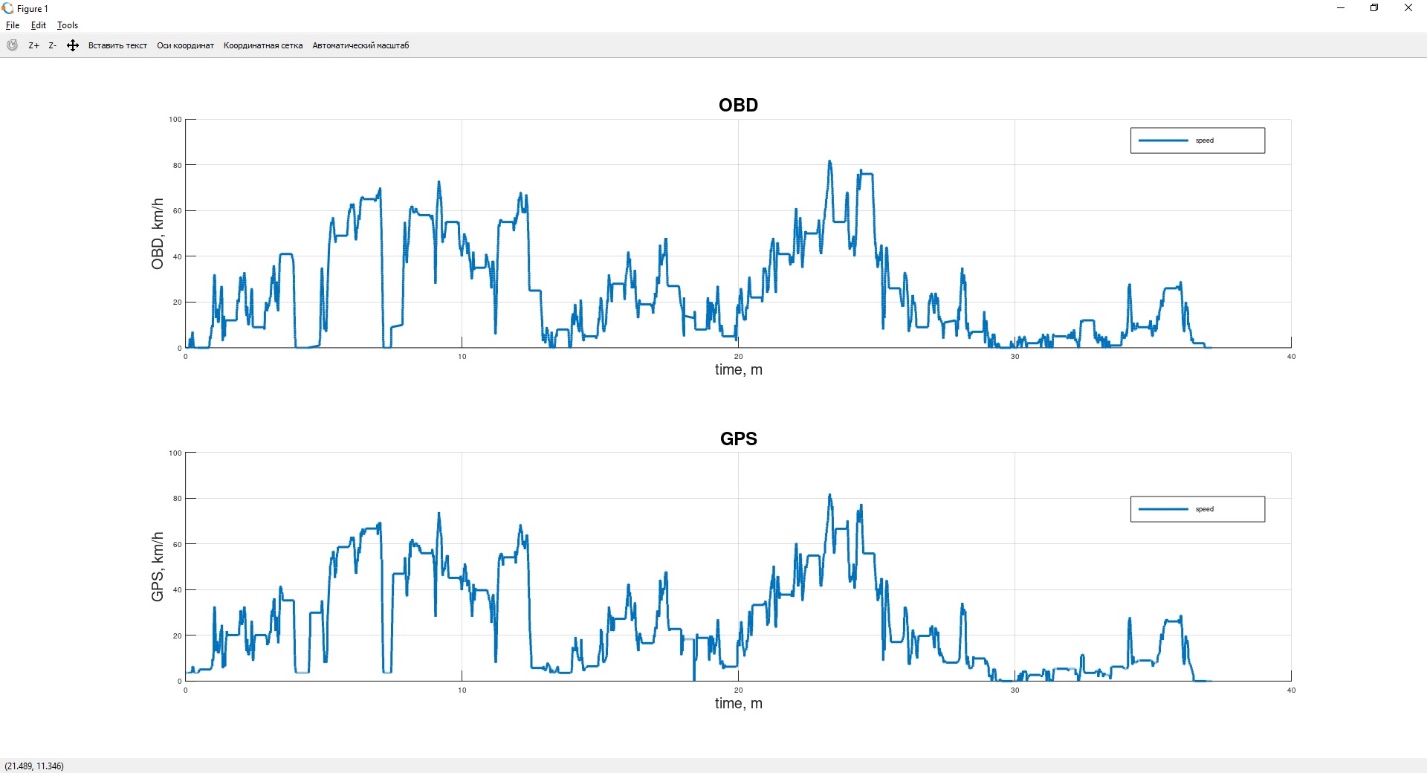
**#2** Отображение скорости в зависимости от времени, звучит очень просто. Но на деле пришлось повозиться с Октавом. Нам удалось в очередной раз убедиться, что в мире науки без Октава и шагу не ступить. Основной сложностью в этом задание стало то ,что начало времени отсчета у разных датчиков разное , а так и их скорость обновления. Код мы поместили в приложении, результат отобразили в максимально удобной форме (рисунок 3). Зависимость скорости от времени в пути. Кажется, вышло не дурно.

Рис. 3 – Графики зависимости OBD\_Speed/Time & GPS\_Speed/Time

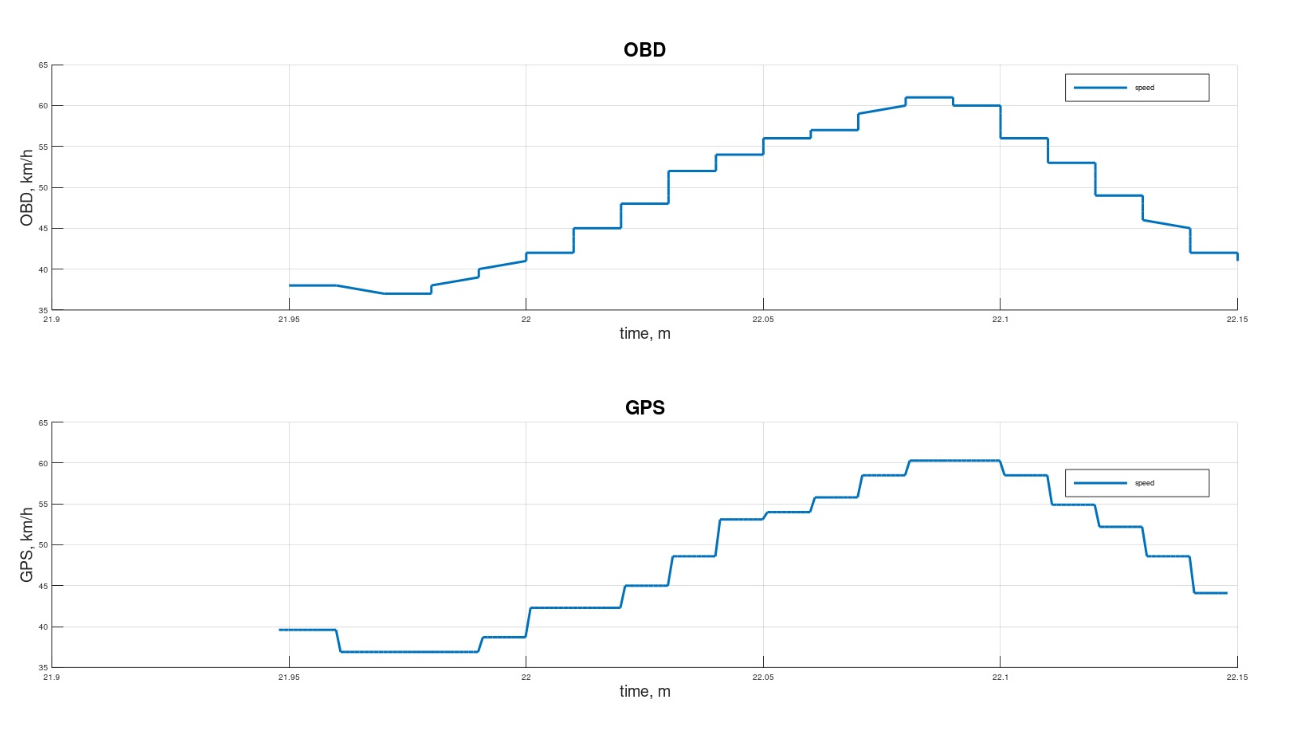


Рис. 4 – Те же графики но с приближением

У графиков есть одно сильное преимущество, что сильно выделяет их на фоне остальных способов визуализации данных, это их наглядность. Первые выводы, которые можно сделать это то что в целом датчики работают одинаково никто по дороге не “отвалился” и картина состояния у них обшая.

**#3** При выполнении задания 3 мнения в команде немного разошлись. Одной части ребят пришлось вспоминать что такое азимут , другой что такое синус и причем тут углы. Но лучше 1 раз показать, чем несколько раз рассказать. Наглядность ещё никому не повредила.

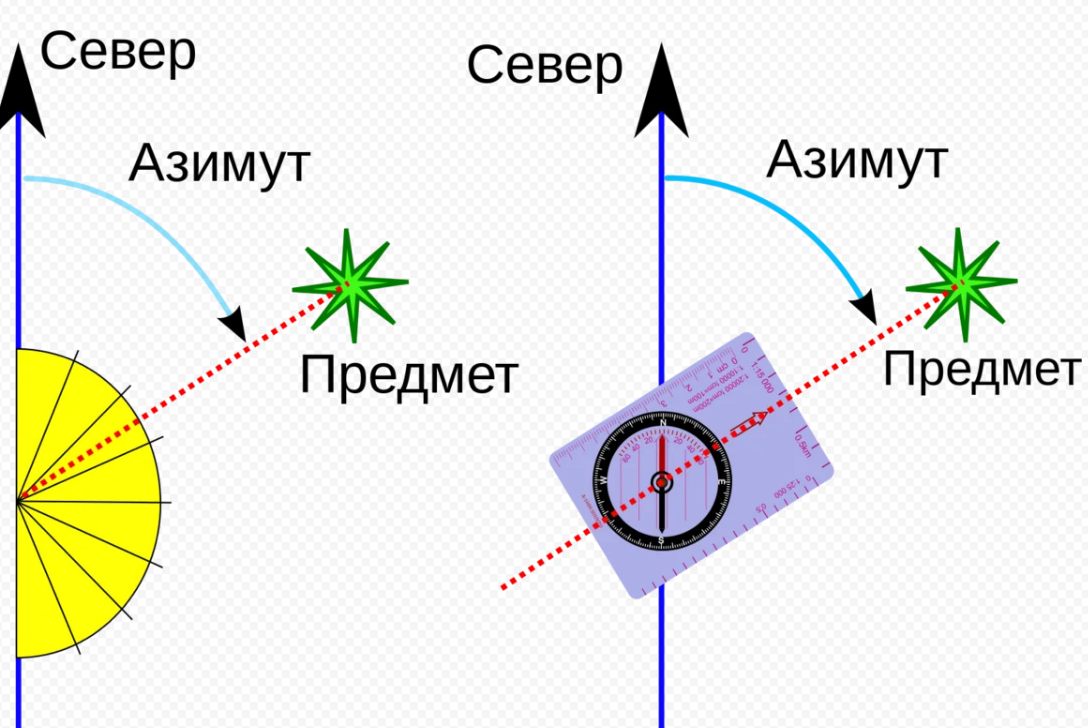


Рис. 4 – Вычисление азимута

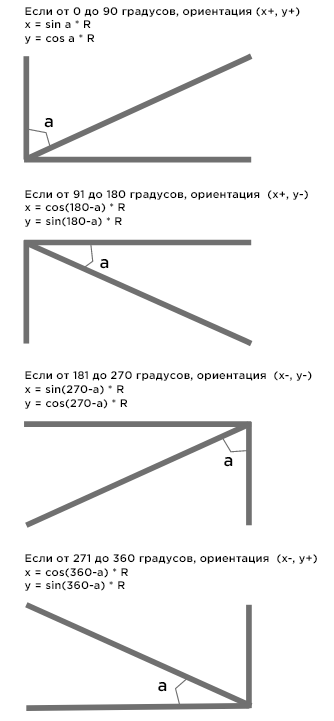
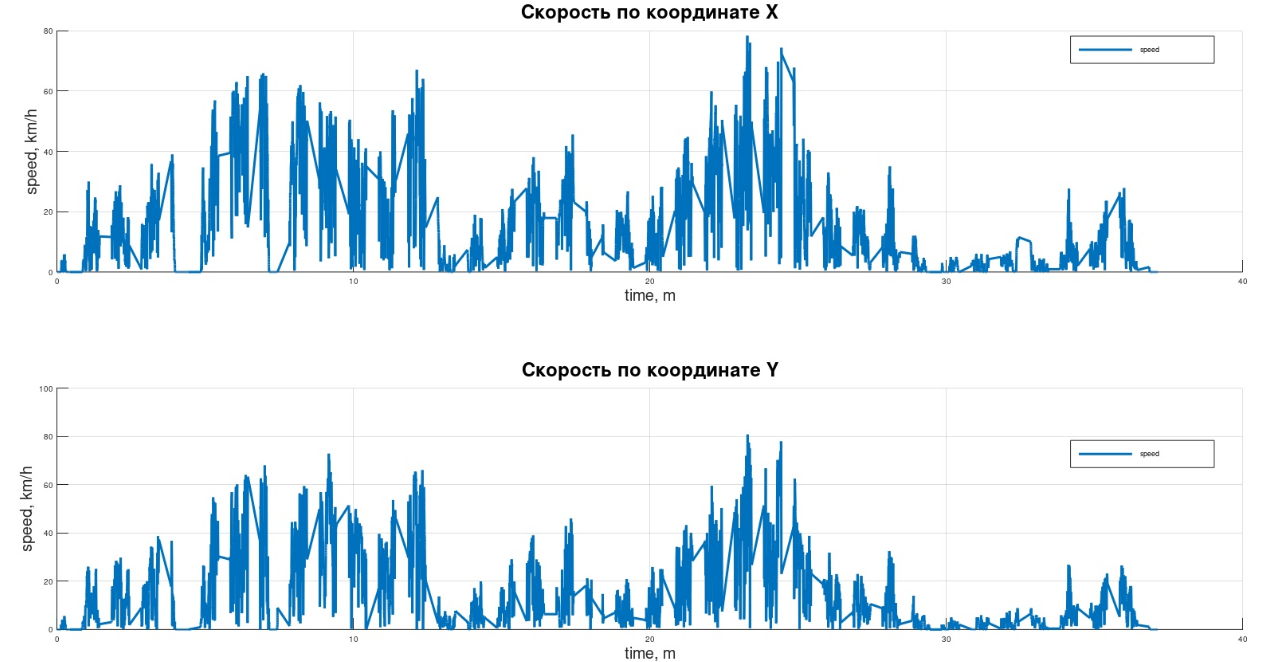


Рис. 5 – Разложение компоненты скорости на X/Y

Где R – скорость, а замысловатые углы – ее направление. Угол отсчитывается по часовой стрелке от нуля, если ориентироваться по часам.

Итог, общий алгоритм получился следуюший берем значение Bearing вычисляем для него синус и косинус после чего умножаем на значение скорости в данный момент времени и получаем два массива данных. Результат работы увидим на рисунке 6

Рис. 6 – график разложения компоненты скорости

Приложение с кодом для   
2 пункта - <https://pastebin.com/index/EF3U6w2G>

3 пункта - <https://pastebin.com/nD0Mikbq>

**Основное задание**

1) О работе фильтра Калмана много где было сказано до нас но это не помешает нам обобщить уже полученные знания. Основная мысль которая была вынесена за несколько дней работы с ним это то что фильтр Калмана — мощнейший инструмент обработки данных , ведь в основе стоит принцип о том, что при фильтрации используется информация о физике самого явления. Удобство работы с фильтром добавляет то что с каждой новой полученной информацией мы все больше и больше начинаем рассчитывать на уже на копленные знания что спасет нас в том случае , если например во время езды на датчик на липнет грязь или мы заедем под землю.

Убрав всю лирику получаем такую таблицу с формулами :

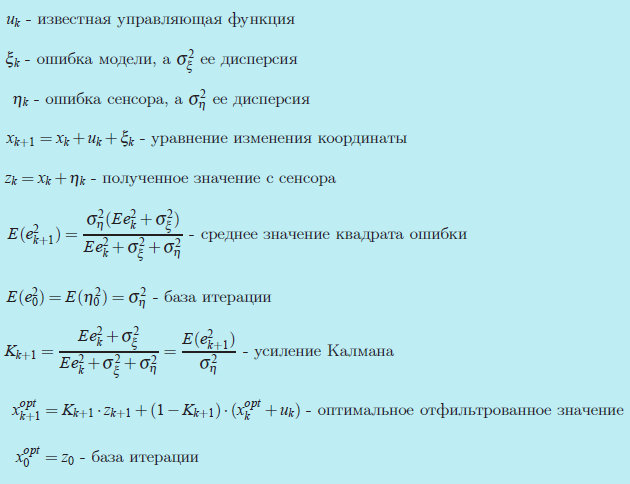
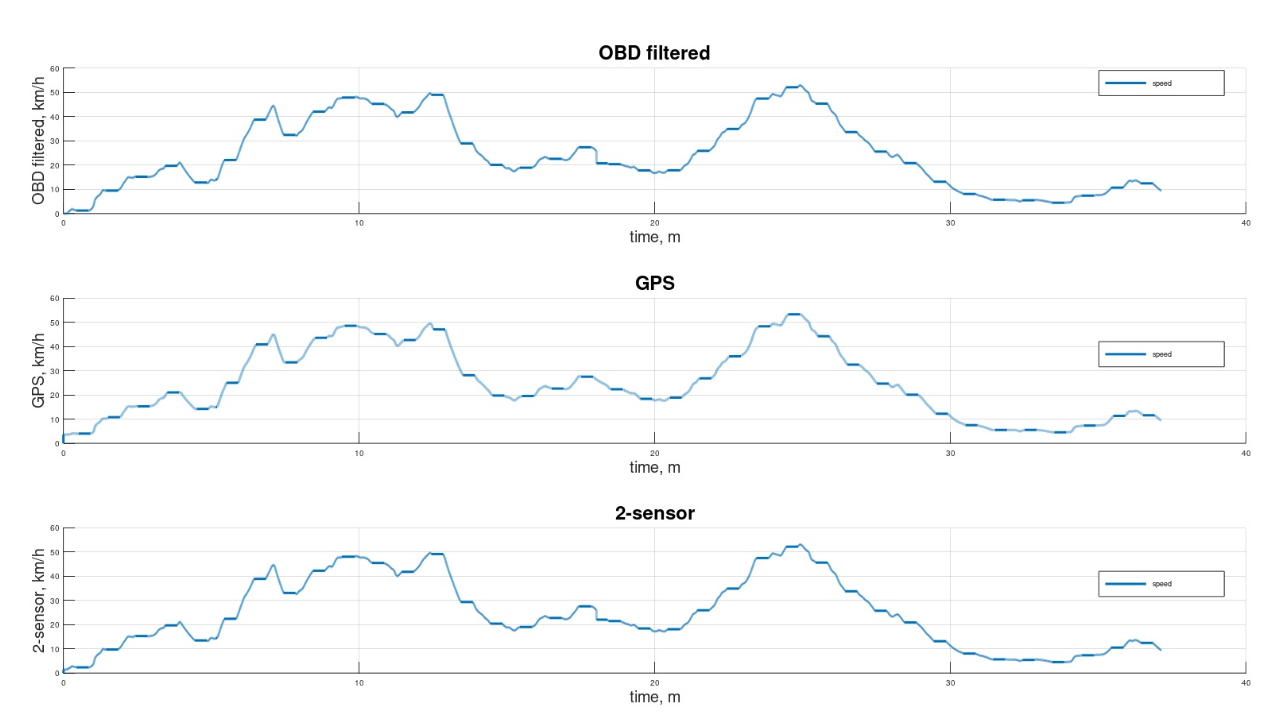


Рис. 7 – Все формулы в одном месте

Особый интерес здесь вызывает xk ведь это величиной может быть вообше что угодно , начиная температурой заканчивая скоростью машины как в нашем случае. В отличии от статьи на хабре в нашем случае необходимо было работать одновременно с двумя датчиками по этому код работает по следующему алгоритму. В начале применяется Калман для каждого датчика датчика в отдельности со своими собственными коэффициентами дисперсии и ошибки, после чего полученные данные преобразуются в один единственный массив с учетом того что OBD чуть-чуть более точный так как он установлен в машине и верим ему соответственно немного больше. В результате получаем такой график :

Рис. 8 – Работа фильтра

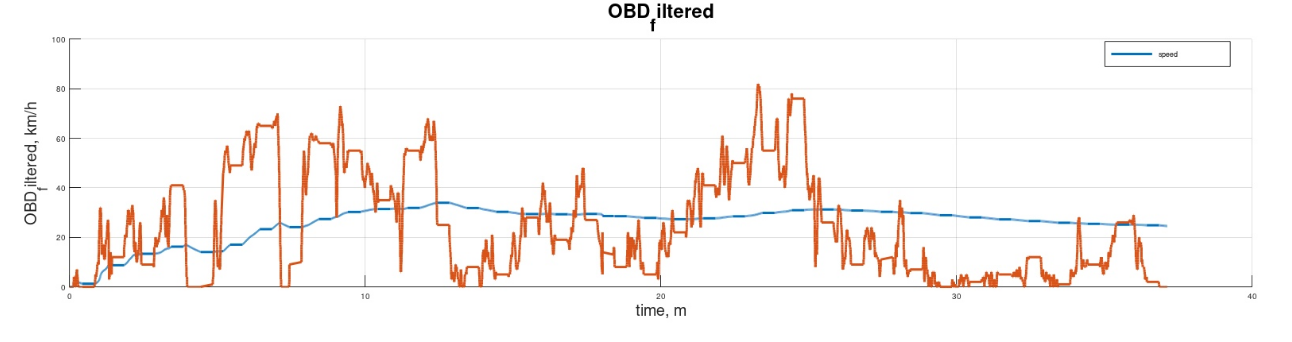
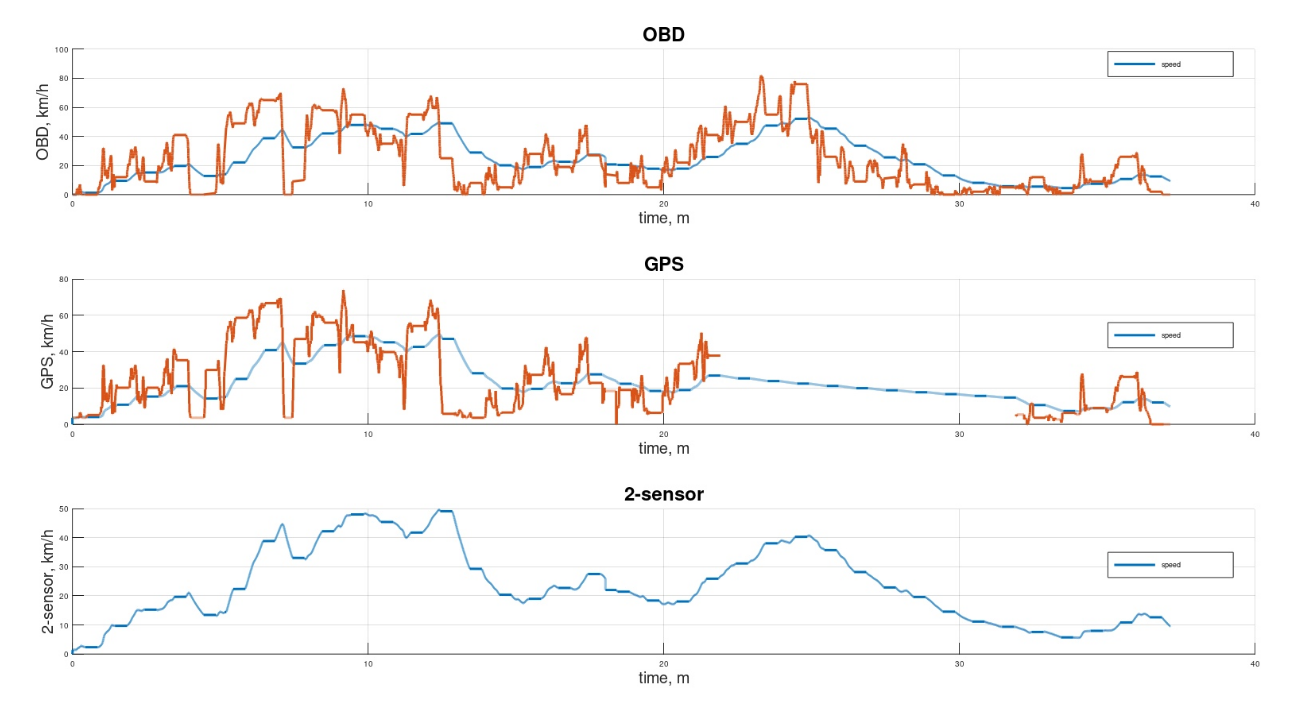
Основной минус в работе с этим фильтром , заключается в том что если сильно выкрутить коэффициенты то график получиться слишком “зализаным”. Т.е система будет максимально сильно противиться любому изменению из вне. Это отлично видно на рисунке 9

Рис. 9 – Неправильно заданный коэффициент

2) Окей фильтр написан и вроде даже работает. Выше было сказано про его сильную надежность. Ну чтож теперь пришло время ее провекрить! В первую очередь во время езды у нас ломается GPS и целых 10 минут шлет нам шляпу вместо данных (рисунок 10).

Рис. 10 – работоспособность алгоритма с пропаданием данных от GPS с 17:25 до 17:35

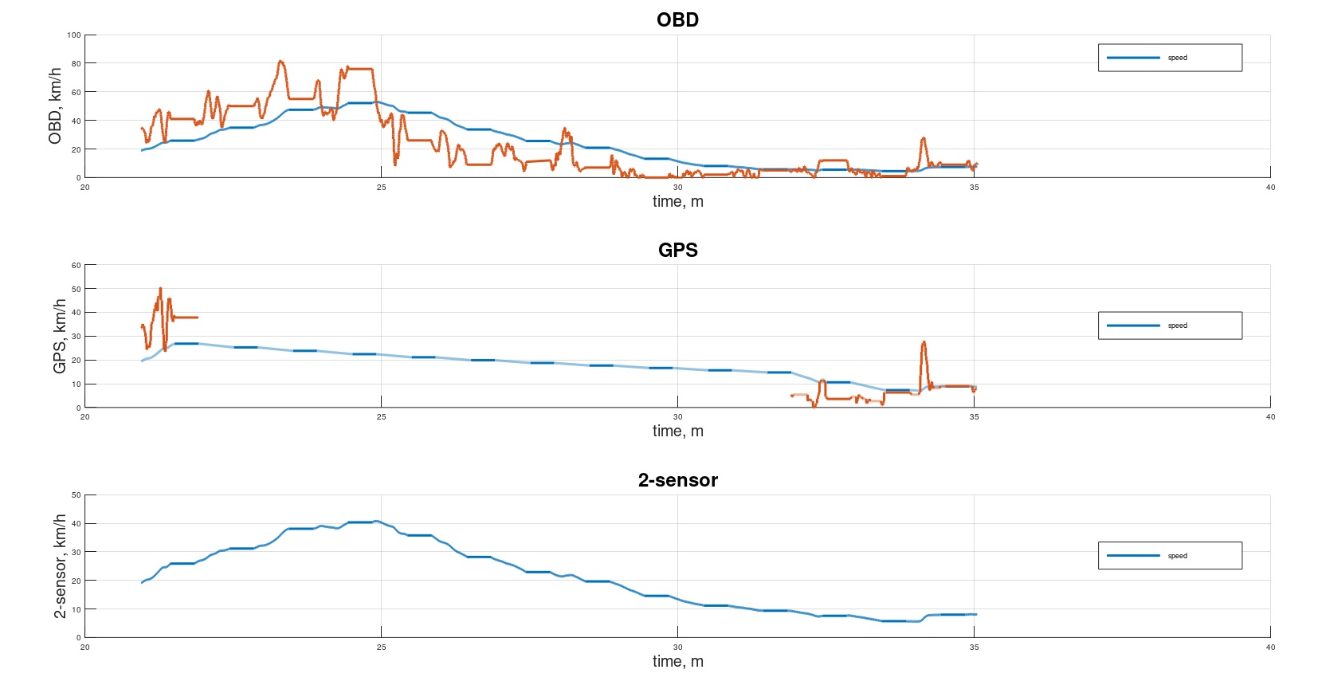


Рис. 11 – приближение

Красным цветом показаны значения с датчика , синим результат их обработки после фильтрации ,нижний график как воспринмает наша программа результат от обоих датчиков.

3) Теперь посмотрим как отреагирует система если отвалиться более точный OBD:

Рис. 12 – работоспособность алгоритма с пропаданием данных от OBD с 17:40 до 17:50



Рис. 13 – тоже самое но поближе

Да в этом случае итоговые данные пострадали побольше сразу заметно то что две амплитуды сразу исчезли. Но в конечном счете благодаря большой надежности фильтра Калмана в момент исчезновения данных с датчика система начала предугадывать новые значения что не позволило данным сильно испорться. Ведь если бы фильтра не было то алгоритм увидев что с одного датчика поступает условно 60км/ч а с другого 0км/ч решил бы что машина едет со скорость 30 километров что сильно бы повлияло , например на значения передвижения по карте.

4) На последок можно посмотреть на то как сильно измениться картина если вдруг сломаются поочередно два датчика сразу

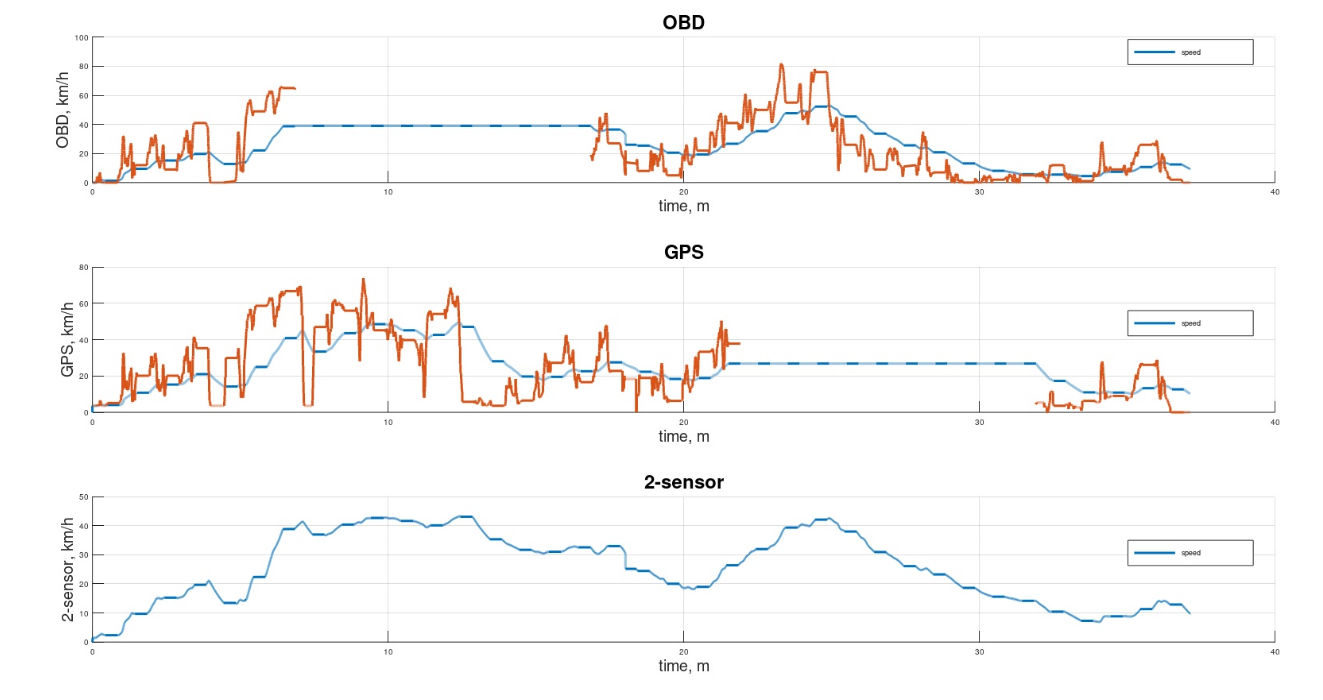


Рис. 14 – одновременная поломка сразу двух датчиков

В самом конце мы решили проверить на сколько сильно будет отличаться картина если на всем периоде эксперимента два датчика успеют сломаться и починаться и в чем отличие от “идельных” условий. Вывод получился следующий фильтр Калмана идеален для работы с датчиками ведь этот алгоритм можем работать в таких случаях когда , информация с датчика каждый раз сильно разниться или вовсе отсутствует. Такой запас прочности не каждый может себе позволить.

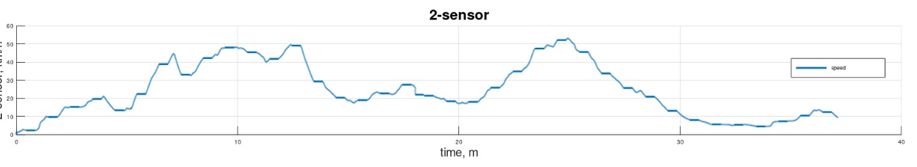


Рис. 15 – Работа фильтра без ошибок от датчиков

Приложение с кодом <https://pastebin.com/hqVGeTdk>

**Основное задание часть 2**

Хорошо, фильтр написан и он даже работает ! Теперь давайте проверим то насколько хорошо он предугадывает. Во второй части задания было необходимо построить одномерный фильтр Калмана для оценки расстояние от начальной точки по широте. Для этого нам понадобиться уже написанный код разложения скорости на две компоненты. Логика следуюшая, раз мы говорим про то что нас интересует только отклонение по широте , то разложим скорость по GPS при помоши того же bearing-a для X и будем по нему предсказывать пройдённый путь.

Во первых так как в этот раз мы работаем только с GPS из таблицы будем брать только каждое десятое значение (ведь GPS обновляется раз в секунду а в таблице они записаны с шагом 0.1) , во вторых необходимо преобразовывать скорость в путь. Конечно горячие умы с 3 курса сразу предложили проинтегрировать все это дело и не париться. Но потом мы вспомнили формулу попроще (рисунок 16):



Рис. 16 – Формула пути

Ведь путь это скорость помноженное на время. Это самое время будем брать равное 1 секунде вот только GPS cчитает в километрах в час по этому что бы преобразавать скорость по X координате умножим на 1 и поделим на 3.6 для того что бы получить путь. На каждом этапе будем прибавлять путь к уже имеющемуся отклонению и получим массив значений который остаётся вывести на график.

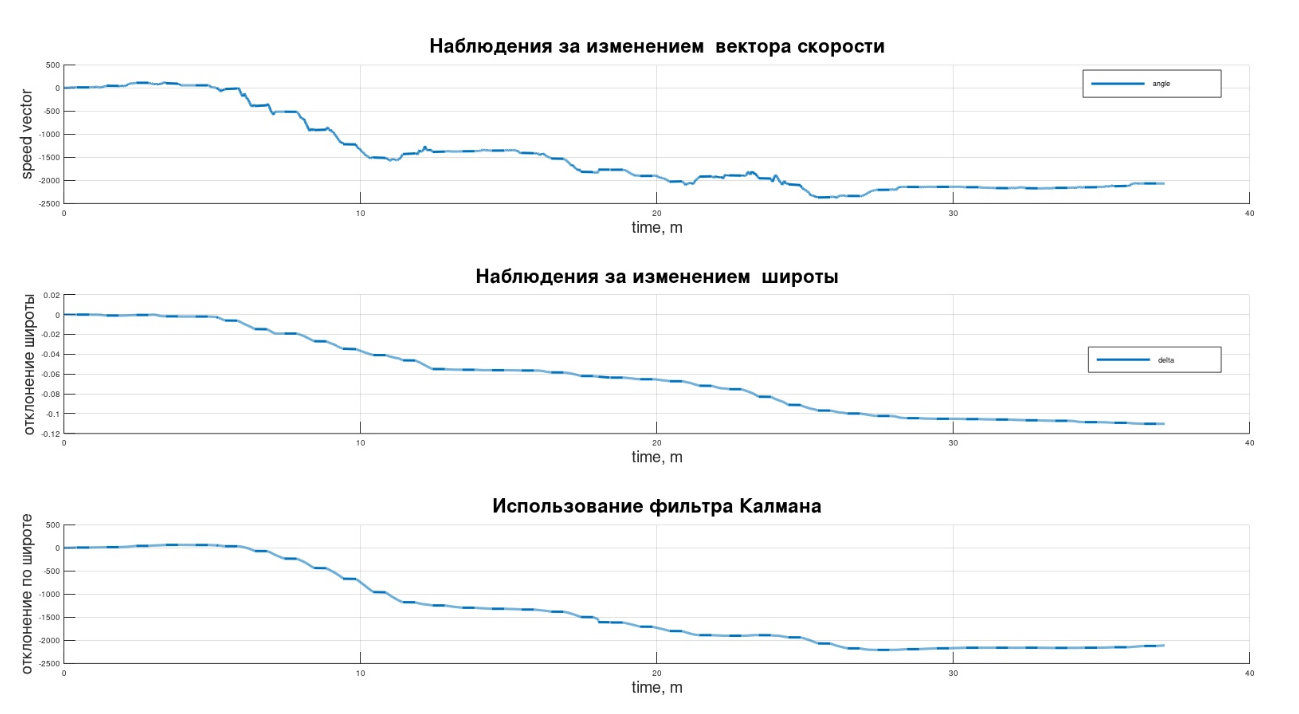


Рис. 17 – Внешний вид отклонения по ширине

На изображений 17 самый верхний график это отклонение высчитанное по формуле , по середине график полученный при помоши вычитания координат , последний применение фильтра к первым данным.

Для наглядности приблизим графики в разных участках (рисунок 18 и 19):



Рис. 18 – Те же графики в приближении

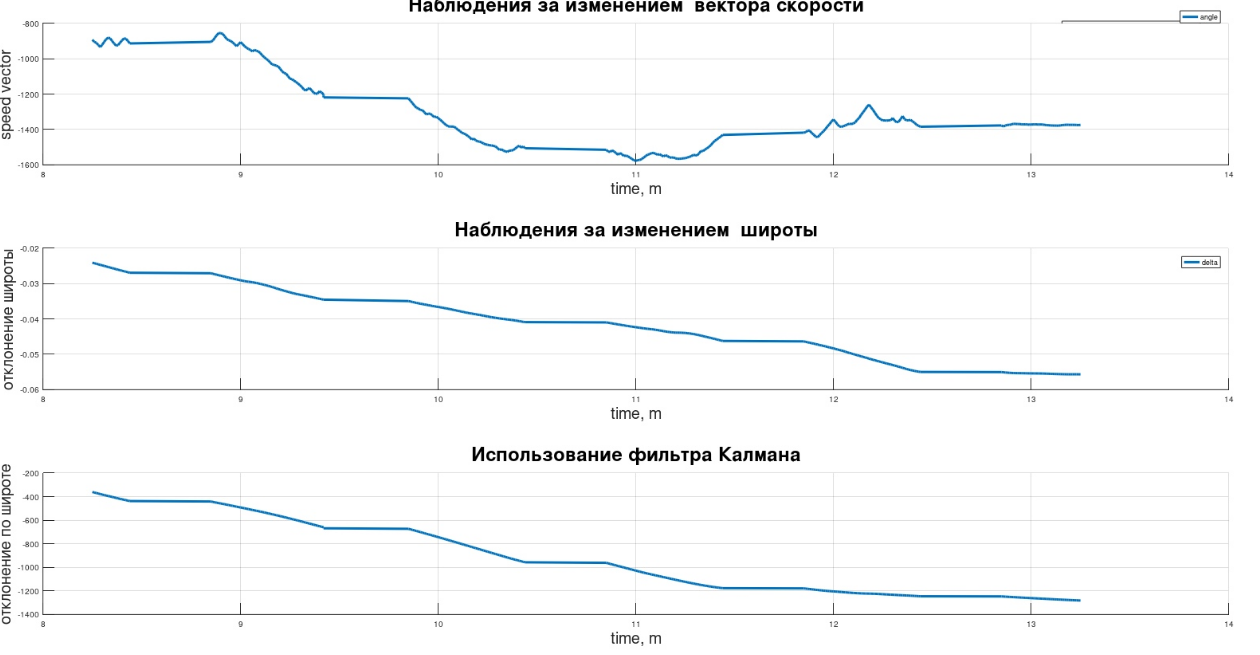


Рис. 19 – Те же графики в приближении

Так как отклонению по GPS мы берем за эталон и сравниваем с теми данными которые мы получили за счет математики. То можно сделать вывод от том что мы справились с заданием успешно.

Приложение с кодом <https://pastebin.com/BQGy7tJg>

**Обший вывод по работе**

За время работы с этим фильтром можно сделать ряд выводов. На него можно положиться в том случае если датчик сильно врет или отвалился вовсе. Благодаря фильтру Калмана можно предугадать последующее передвижение робота в направлении. Благодаря этой работе мы поняли почему многие научные журналы так лестно отзываются от таком способе фильтрации данных и в бедующем мы так же будем его использовать в своих разработках.