Движение транспортного средства с учётом профиля трассы (GPS-трек)

Генов А.В.

Аннотация.
В проекте произведен анализ *GPS*-трека с учетом профиля трассы. Демонстрируются графики зависимости параметров транспортного средства друг от друга, физические и математиские формулы.

Об авторах:
Генов Александр Владимирович,
НИУ ВШЭ МИЭМ им. А.Н.Тихонова
ул. Мясницкая, 20, г.Москва, Россия, 101000
e-mail: avgenov@edu.hse.ru

Введение

В данном проекте задачей было выполнить анализ *GPS*-трека, записать уравнение движения транспортного средства по треку с учетом трения качения и аэродинамического сопротивления и решить граничную задачу. Построить графики зависимости от времени, расстояния: скорости, расстояния пройденного ТС. Рассчитать максимальные, минимальные и средние скорости ТС. Рассчитать затраченное на преодоление всего расстояния время и энергию.
Решение было оформлено с помощью компонента программы Anaconda Navigator– *Jupyter Python ver 3.6.5*. Был использован метод численного интегрирования, а именно метод Рунге-Кутта-Дорманда-Принса 4-5 порядка.
Основной идеей статьи является продемонстрировать результаты анализа движения транспортного средства по профилю трассы с помощью графиков и математических выкладок с конкретными цифрами.
Структура статьи:
1. Немного о GPS: Рассматривается история и области применения навигационной системы GPS;
2. Описание решения: Описано подробное описание проделанной работы
3. Анализ результов: В третьей главе полученные в ходе опытов результаты
4. Заключение: В заключении содержатся выводы.

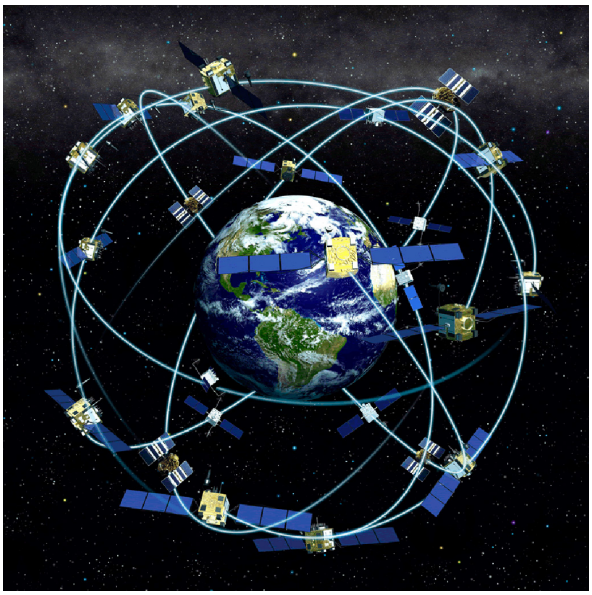

Рис. 1. «GPS»

1. Немного о *GPS*

Global Positioning System (GPS) — система глобального позиционирования, которая обеспечивает рассчёт времени и расстояния, и определяющая местоположение во всемирной системе координат. Система была создана Министерство обороны США в 1970-е. Изначально глобальная система позиционирования разрабатывалась и была направлена на военные проекты, но впоследствии стала доступна и для гражданских целей — достаточно иметь только *GPS*-приёмник, который встроен, например, в навигатор, смартфон или другой аппарат.
Действительно, сложно найти человека в современном мире, который не сталкивался бы с системой *GPS*. Она позволяет прокладывать маршруты с учетом дорожных знаков и автмобильных пробок, искать по карте конкретные здания, улицы, рестораны, атвозаправка, торговые центры и другие объекты инфраструктуры.
Области применения *GPS*:

- Геодезия
- Картография
- Навигация
- Спутниковый мониторинг транспорта
- Сотовая связь
- Геотегинг
- и др

2. Описание решения

Характристики выбранного транспортного средства

Легковой автомобиль:

- масса **(m):** 1500 кг
- площадь поперечного сечения **(S):** 2.5 м^2
- коэффицент аэродинамического сопротивления **(C_x):** 0.25
- мощность двигателя **(P):** 125 л.с $\approx$ 91.94 кВт
- КПД трансмиссии **(K_p):** 0.8
- коэффицент сопротивления качению **(K_r):** 0.05
- температура воздуха **tau:** +10°C

Этап 1

а) Выгружен трек в формате *CSV* и загружен в численный массив *numpy*, содержащий широту, долготу и высоту, при помощи функции *genfromtxt*.
б) Использована формула с *arctan* для определения расстояний между соседними точками и результатом получен массив *numpy*, содержащий для каждой точки трека расстояния от предыдущей точки до следующей и высоту.

$$\Delta\sigma = \arctan \frac{\sqrt{(\cos\phi_2 \sin\Delta\lambda)^2 + (\cos\phi_1 \sin\phi_2 - \sin\phi_1 \cos\phi_2 \cos\Delta\lambda)^2}}{\sin\phi_1 \sin\phi_2 + \cos\phi_1 \cos\phi_2 \cos\Delta\lambda} \tag{1}$$

$\phi_1, \lambda_1; \phi_2\lambda_2$ — широта и долгота двух точек в радианах
$\Delta\lambda$ — разница координат по долготе
$\Delta\sigma$ — угловая разница

в) С помощью функции *cumsum* для каждой точки трека, получено расстояние до нее от начала трека и результатом получен массив *numpy*, содержащий для каждой точки трека расстояния от начала трека и высоту.
г) Отображен график высота-расстояние:

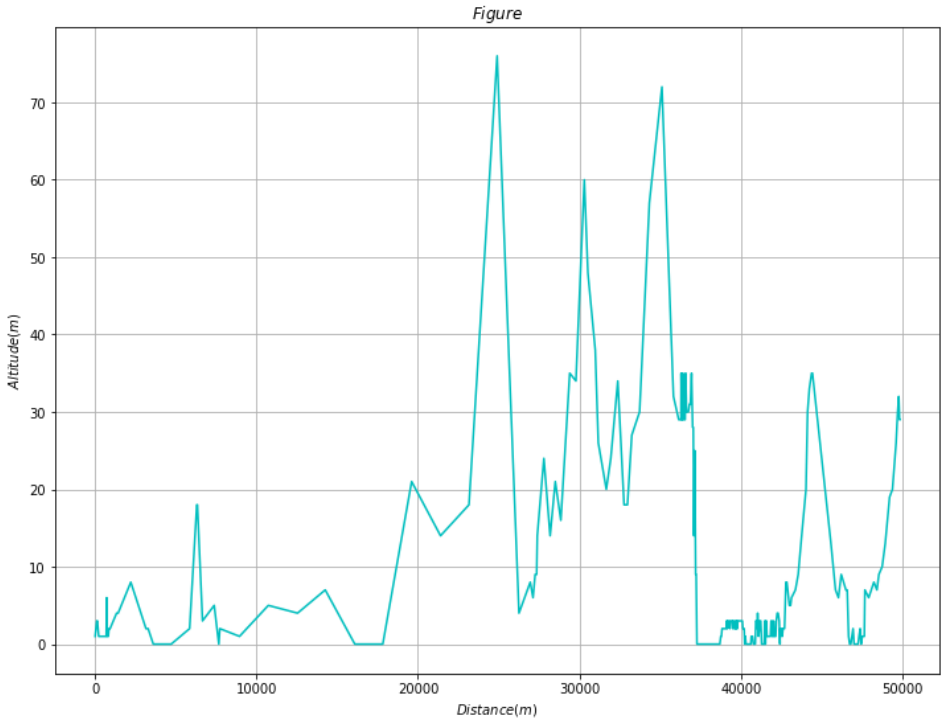

Рис. 2. «График зависимости высоты от пройденного растояния»

д) На основе полученной зависимости высоты от расстояния построен кубический интерполянт:
$h(x) = scipy.interpolate.InterpolatedUnivariateSpline(расстояния, высоты, k=3)$

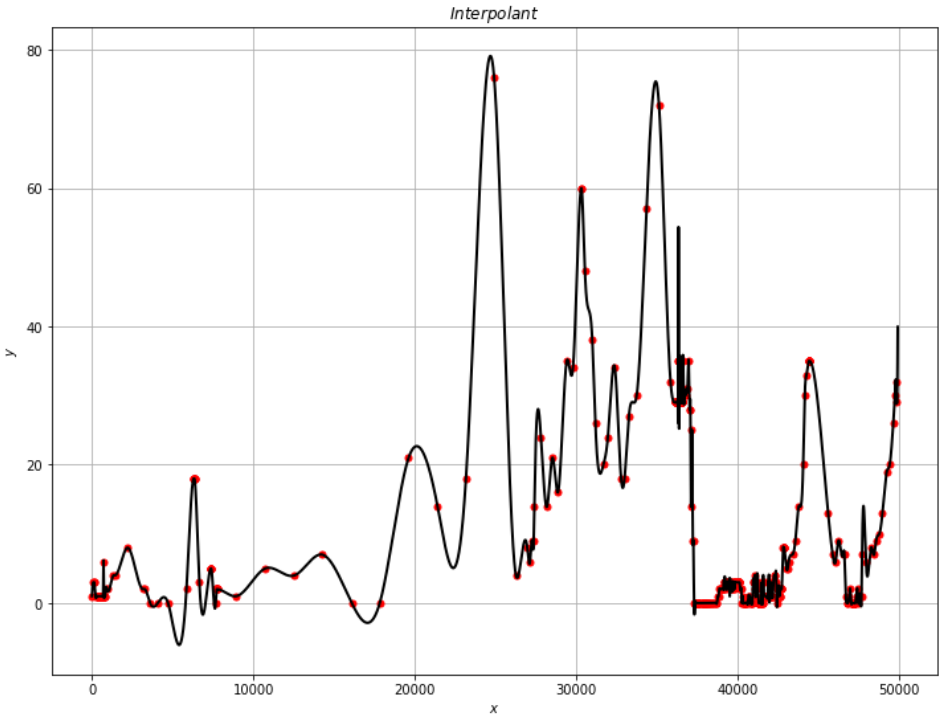

Рис. 3. «Кубический интерполянт»

Этап 2

а) Записано уравнение движения транспортного средства (Обыкновенное дифференциальное уравнение второго порядка) с учетом сопротивления качению и аэродинамического сопротивления.

$$m\ddot{x} = \sum_{i=1}^{n} F_i(x, v) \tag{2}$$

Для того, чтобы определить тангенс угла наклона трека в любой точке трека была вычеслена производная от функции интерполянта.

$$\alpha(x) = h(x).derivative \tag{3}$$

Отображен график производной:

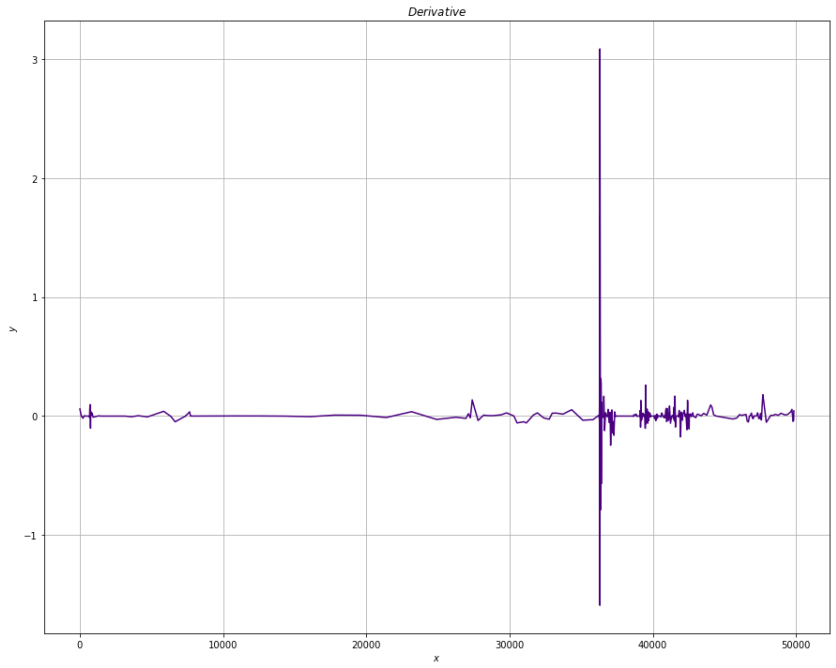

Рис. 4. «График производной»

б) Приведено уравнение движения второго порядка (2) к системе уравнений первого порядка путем замены переменных следующим образом:

$$\begin{cases} \dot{x} = v \\ \dot{v} = \frac{1}{m} \sum\limits_{i=1}^{n} F_i(x, v) \end{cases} \tag{4}$$

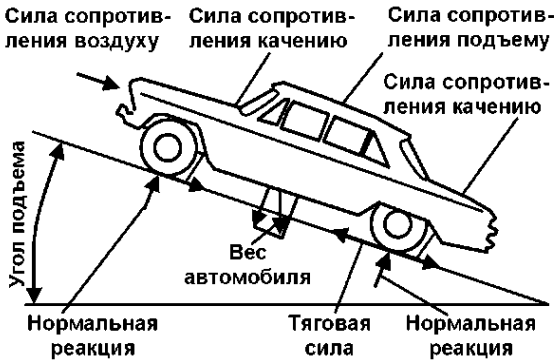

Рис. 5. «Схема сил, действующих на автомобиль, во время подъема»

$$\dot{v} = \frac{1}{m}(\frac{PK_p}{v} - C_x\frac{\rho v^2}{2} - K_r mg\cos\alpha - mg\sin\alpha) \quad (5)$$

,где

- $\frac{PK_p}{v}$ - сила тяги,
- $C_x\frac{\rho v^2}{2}$ - сила аэродинамического сопротивления (ρ - плотность воздуха при $+10°C$),
- $K_r mg\cos\alpha$ - проекция сопротивлению трения качения (α-угол подъема),
- $mg\sin\alpha$ - проекция силы тяжести

Этап 3

а) Написана функция, реализующая правую часть полученной системы уравнений. Выглядеть она может, например, так:

```
1  params=dict()
2  params={"m":1500,"g":9.82,"S":2.5,"Cx":0.25,
3          "P":91940,"Kk":0.05,"Kt":0.8,"p":1.2466}
4
5  def motion_ode(t,s,alpha_x,params):
6
7    x,v=s
8    sec_2=alpha_x(s[0])*alpha_x(s[0])+1
9    sqrtsec=math.sqrt(sec_2)
10   cos=1/sqrtsec
11
12   dx=s[1]
13   dv=(1/params["m"])*((params["P"]/s[1])*params["Kt"]-
14   -params["Cx"]*params["p"]*params["S"]*s[1]*s[1]/2-
15   -params["Kk"]*params["m"]*params["g"]*cos-params["m"]*
16   *params["g"]*alpha_x(s[0])*cos)
17
18   return [dx,dv]
```

,где alpha(x) - зависимость тангенса угла наклона от расстояния,s - вектор состояния ТС. В данном случае вектором состояния ТС будет вектор (x, v), где x - расстояние от начала трека до ТС (вдоль трека), v - скорость ТС вдоль трека. params - остальные параметры, необходимые для вычисления правой части уравнения движения.

б) Написана функция останова *solout*, позволяющая остановить интегрирование в момент достижения ТС конца трека. Эта функция также записывает каждый вектор состояния ТС и момент времени в список, ссылка на который передается в неё.

После интегрирования в указанном списке окажутся все векторы состояния и моменты времени для всех шагов интегрирования.
Функция останова может выглядеть так:

```
1 def stopLength(t, s, L, lst):
2   lst.append(np.hstack((s,t)))
3   if (s[0] >= L):
4     return -1
5   return 0
```

, где *s[0]* - компонента x вектора состояния ТС, L - длина всего трека.

в) При помощи *scipy.integrate.ode* и функции останова проинтегрировано уравнение движения ТС и получен список всех векторов состояния и моментов времени.

Сделать это можно например так:

```
1   L=sumdist[502]
2   prop = sp.integrate.ode(lambda t, s: motion_ode(t, s,
        alpha_x,params))
3   prop.set_initial_value(s0, 0)
4   prop.set_integrator('dopri5', nsteps=1e5)
5   lst = []
6   prop.set_solout(lambda t, s: stopLength(t, s, L, lst))
7   prop.integrate(1e10)
8   arr = np.asarray(lst)
```

где, *sumdist* - это список, который содержит для каждой точки трека расстояние до нее от начала трека. Последняя точка - 502, соответственно в ней и содержится длина всего трека. *s0* - вектор состояния ТС в начальный момент времени (t = 0). s0 = (0, v0), где v0 выбирается из условия возможности решения построенного ОДУ в начальный момент времени (например, v0=1e-16). После интегрирования массив arr будет содержать все вектора состояния и моменты времени, которые интегратор вычислял в процессе работы.
Погрешность при решении методом Дорманда-Принса:

49865.57307017194 - значение расстояния, полученное интегратором.

49849.516001435986 - искомое расстоние, полученное функцией cumsum

Таким образом, погрешность метода:
0.000322110825218

3. Анализ результатов

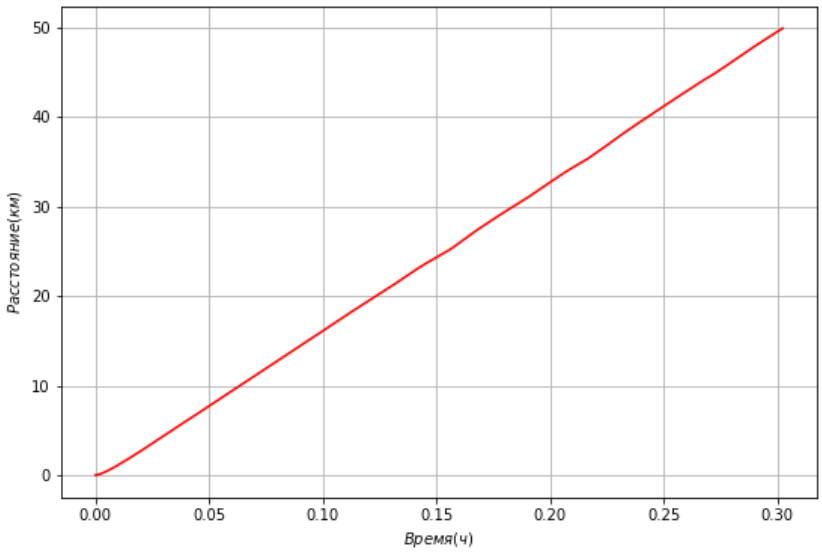

Рис. 6. «Зависимость расстояния от времени»

На Рис. 6 изображен график зависимости расстояния от времени, за которое автомобиль преодолел весь путь. Из этого графика видно, что расстояние зависит от времени линейно на протяжении всего расстояния.

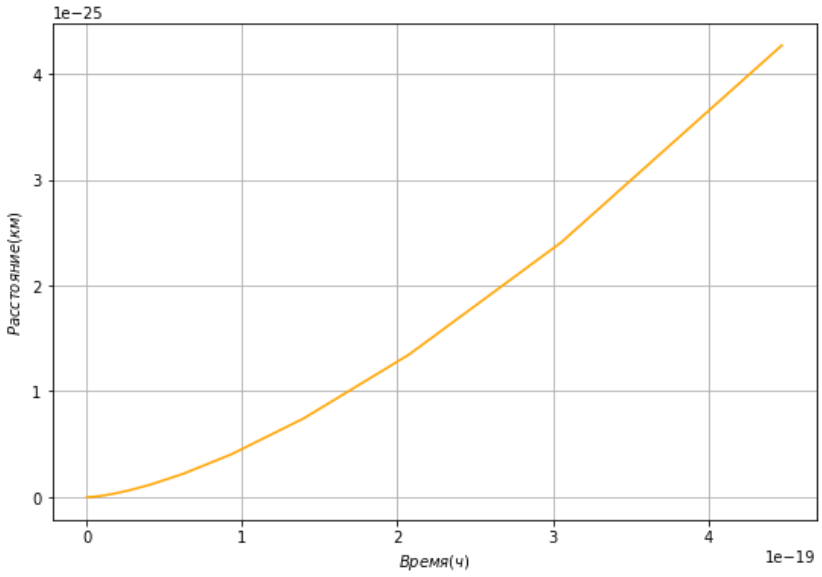

Рис. 7. «Зависимость расстояния от времени (первые 50 точек)»

На Рис. 7 изображен тот график зависимости расстояния от времени, но отображено только первые 50 точек, чтобы наглядно продемонстрировать, что зависимость расстояния от времени не линейная, а имеет вид кривой.

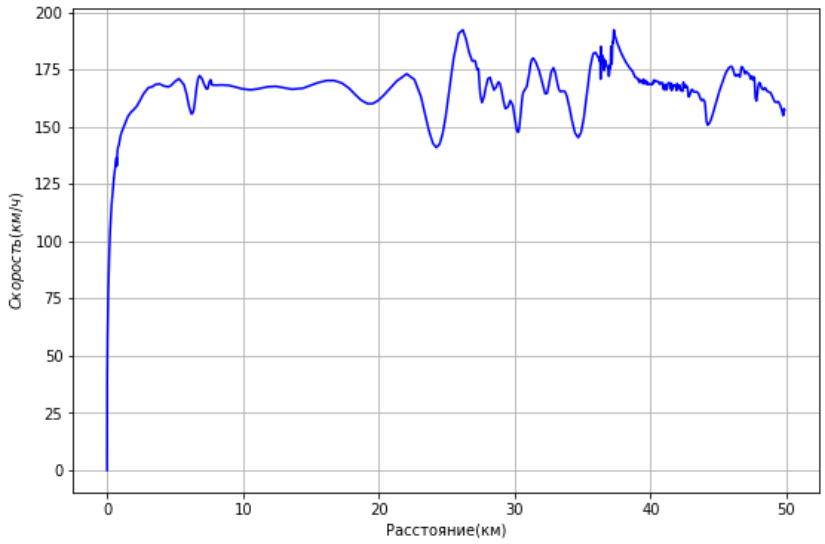

Рис. 8. «Зависимость расстояния от скорости»

На Рис. 8 изображен график зависимости расстояния от скорости, который показывает как изменялась скорость транспортного средства в течение всего пути. Из графика очевидно, что минимальная скорость достигалась автомобилем в самом начале движения и почти сразу достигла средней сокрости, которая сохранялась до достижения им конца трассы. Пик скорости достигается автомобилем примерно на середине участка пути.

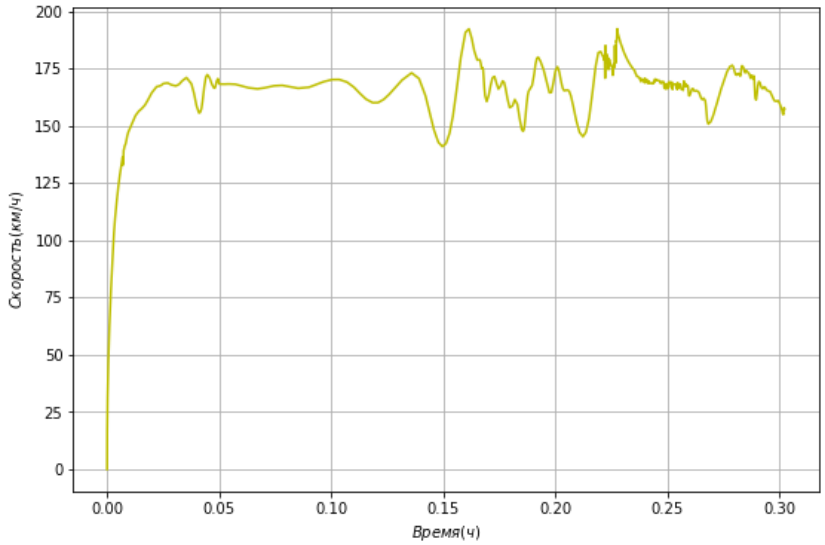

Рис. 9. «Зависимость скорости от времени»

На Рис. 9 изображен график зависимости скорости от времени, который демонстрирует как изменяется скорость легкового автомобиля по мере преодоления им всего пути. Из графика видно, что скорость транспортного средства быстро стала

больше 150 *км/ч*, но имеела значительный спад по прошествии половины временного промежутка и в течение второй половины колеблется, меняясь в пределах 50 *км/ч*.

Общий путь:

$S = 49.84$ *(км)*

Расстояние, пройденное легковым автомобилем по городу Сан-Франциско, следовательно вся трасса проходит через населенный пункт. В таком случае, моделирование движения ТС не учитывает множество городских факторов движения, таких как регулируемые перекрестки, пешеходы, ограничение скоростного режима, пробки и др. Такой подход к моделированию в большей степени подходит для автомагистрали.

Максимальная, минимальная и средняя скорость:

$V_{max} = 192.42$ *(км/ч)*
$V_{min} = 3.6 * 9.99e^{-17}$ *(км/ч)*
(Максимально близка к нулю, но не равна ему)
$V_{average} = 127.03$ *(км/ч)*

В результате расчетов с помощью функций max, *min* языка программирования *Python* и *np.mean* библиотеки *numpy* получены точные численные значения максимальной, минимальной и соответсвенно средней скорости движения транспортного средства.

Затраченное на преодоление всего расстояния, время и энергия.

Общее время, за которые автомобиль преодолел указанный выше общий путь. Двигаясь в среднем со скоростью 127 *км/ч* автомобиль расстояние 49.84 *км* за 0.3 часа или за 18 минут.

Time= 0.3 hours

$$L = \frac{P * T}{q * \eta} \tag{6}$$

,где P - мощность двигателя в кВт,
T - общее время работы двигателя,
q - удельный расход топлива (10кВт*ч/л),
η - КПД двигателя внутренного сгорания (=0.4)

Liters= 6.9 L

Столько литров топлива потратил легковой автомобиль, пройдя за 18 минут ≈ 50 *км*, разгонясь при этом до скорости ≈ 200 *км/ч*.

4. Заключение

В работе проведен анализ данных и найдены требуемые значения. Получены графики зависимости различных характеристик ТС друг от друга: скорости от времени, расстояния; расстояния, пройденного ТС от времени. Представлены все необходимые формулы на основе *GPS* - трека. Изучен язык програмирования *Python* и возможности его библиотек, таких как: *numpy*, *scipy*, *matplotlib*. Написано уравнения движения транспортного средства по треку с учетом трения качения, аэродинамического сопротивления. Решена граничная задача.

Список литературы / References

[1] D.E. Knuth, The TEXbook (Addison-Wesley, Reading, 1984).

[2] Марк Лутц, *Изучаем Python 4-е издание*, 2010.

[3] Dormand, J. R. and Prince, P.J.:, "High order embedded Runge-Kutta formulae", *J. Comp. Appl. Math.*, **7** (1981), 7.67–75.

[4] Щур Л. Н., "Система LaTex для подготовки научных публикаций: учебно- методическое пособие,", *М.: МИЭМ НИУ ВШЭ*, 2017, 27.

[5] "*GPS*-трек *San Francisco Tag 2*".
URL: https://www.gpsies.com/map.do;jsessionid=FB1359606E64E56FA7EC33753D376042.fe3?fileId=chasgwgivzdwkoxr

[6] "Репозиторий с лекциями по библиотекам Python".
URL: https://github.com/jrjohansson/scientific-python-lectures

[7] "Онлайн-курс по языку программирования *Python*".
URL: https://stepik.org/course/67/syllabus

[8] "Вычисление расстояния и начального азимута между двумя точками на сфере".
URL: http://gis-lab.info/qa/great-circles.html