Федеральное государственное автономное
образовательное учреждение
высшего образования
«Национальный исследовательский университет
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
Образовательная программа «Прикладная математика»
бакалавр

ОТЧЕТ

по проектной работе

Движение транспортного средства по дороге (gps-трек)

Выполнил студент группы БПМ 191
Любимов Георгий Владимирович

Руководитель проекта:
Бобер Станислав Алексеевич, старший преподаватель

Москва
27 мая 2020 г.

Введение

Расчёт движения транспортного средства (ТС) используется повсеместно, как в бытовых случаях, так и для организации логистики предприятия. В данном примере будет построена модель движения ТС, с учётом ограничения скорости на дороге, и примеры данных, которые можно получить из неё, например, расход топлива и минимальная, максимальная и средняя скорости на пути трека.

Цели работы:

- Записать уравнение движения ТС по треку с учётом трения качения, аэродинамического сопротивления, также учесть, что на треке максимально допустимая скорость 100 км/ч
- Указать начальные и граничные условия
- Решить граничную задачу
- Построить графики зависимости по времени, расстояния:
  - Скорости
  - Расстояния, пройденного ТС (только от времени)
- Рассчитать максимальную, минимальную и среднюю скорость
- Рассчитать затраченное на преодоление всего расстояния, время и энергию. Энергию перевести в литры топлива с коэффицентом 10 кВт*ч/л, с учётом КПД двигателя внутреннеого сгорания = 0.4.
- Построить графики, отражающие участки, где скорость ТС первысила 100 км/ч или опустилась ниже 10 км/ч.

Дано:

- Трек
- Характеристики транспортного средства:
  - Мощность двигателя ($P = 125$ л.с.$= 91937.5$ Вт)

– КПД трансмиссии ($k_p = 0.8$)

– Коэффициент сопротивления качению ($f/R = 0.05$)

– Коэффициент аэродинамического сопротивления ($C_x = 0.25$)

– Площадь поперечного сечения ($S = 2.5$ м^2)

– Температура воздуха ($\tau = -10$ °C)

– Масса ТС ($m = 1500$ кг)

Решение задачи

Подготовка данных трека

Скачиваем с сайта таблицу, содержащую в каждой строке широту, долготу и высоту для точки трека. Затем загружаем эти данные в массив Python при помощи функции genfromtxt библиотеки numpy.

С помощью формулы расчёта расстояния между соседними точками на сфере создаём функции, которая принимает на вход широту и долготу двух точек и возвращает расстояние между ними. Проходя через неё получаем массив содержащий расстояние до предыдущей точки (для первой точки оно равно 0) и высоту этой точки.

Используя функцию cumsum суммируем расстояния до предыдущих точек, таким образом получая массив, содержащий теперь расстояние от начала трека, до этой точки и также её высоту. Вот график получившегося массива:

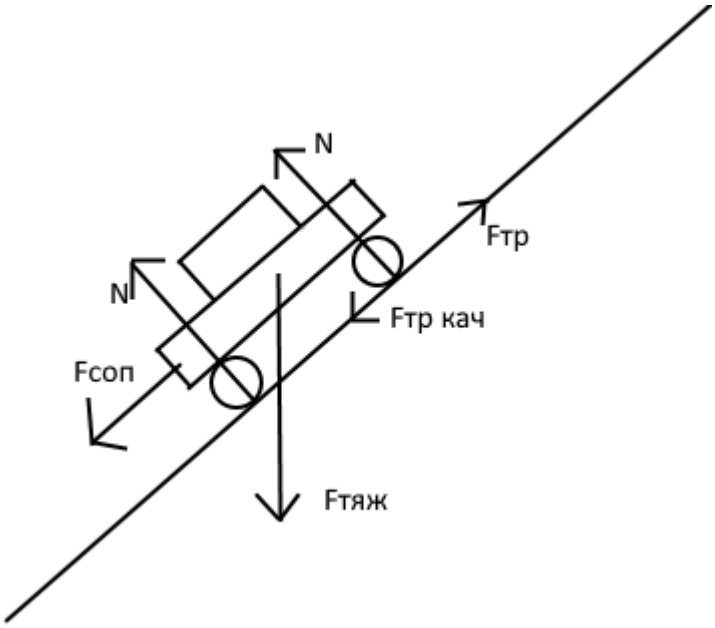
Далее получаем из этого массива кубический интеполянт и сравниваем его с предыдущим графиком:

Запись уравнения движения ТС

Уравнение при P=const

На ТС на треки действуют:

- Сила тяжести
- Сила движения (трение колёс, которое и приводит ТС в движение)
- Сила трения качения
- Сила сопротивления воздуха
- Сила реакции опоры



Нам также понадобится плотность воздуха для вычисления его сопротивления (ρ). Выгружаем данные с сайта в формате BeautifulSoup и парсим, получая переменную с таблицей плотности воздуха от температуры, затем с помощью цикла находим нужную температуру и плотность воздуха и используем её. Для определения угла α, мы поместим переменную alpha_x в которой будет хранится производная от интерполянта на любой точке трека.

Запишем ОДУ второго порядка движения ТС, мы будем считать начальную скорость $v \to 0$:

$$x = x_0 + x't + \frac{x"t^2}{2}$$

где

$$x'' = \frac{F}{m} = \frac{(\frac{P}{v_0} - mg\sin\alpha - \frac{f}{R}mg\cos\alpha - C_x\frac{\rho v_0^2}{2}S)t^2}{2m}$$

С помощью замены $x' = v_0$ и $v' = \frac{(\frac{P}{v_0} - mg\sin\alpha - \frac{f}{R}mg\cos\alpha - C_x\frac{\rho v_0^2}{2}S)t^2}{2m}$ мы можем привести это уравнение к системе уравнений первого порядка:

$$\begin{cases} x' = v_0 \\ v' = \frac{(\frac{P}{v_0} - mg\sin\alpha - \frac{f}{R}mg\cos\alpha - C_x\frac{\rho v_0^2}{2}S)t^2}{2m} \end{cases}$$

Посмотрим зависимость силы двигателя от скорости автомобиля:

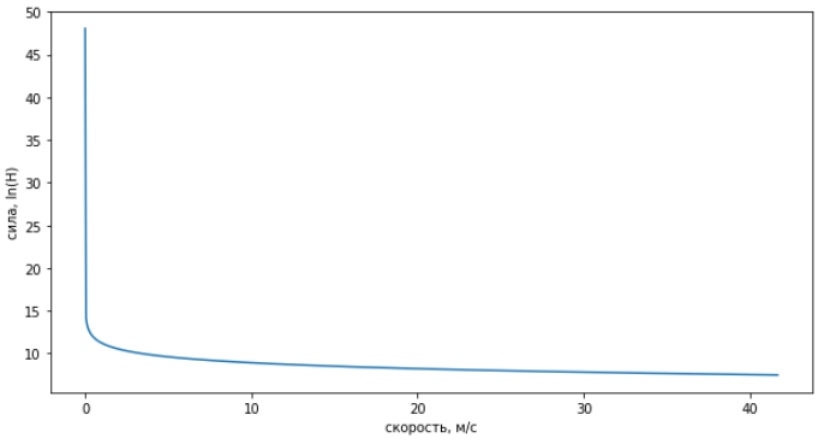


Рис. 1: зависимость силы от скорости ТС

Как можно увидеть сила двигателя всегда положительна, но в реальных условиях водитель ограничен максимальной скоростью на пути, но даже если сила двигателя будет 0, то может возникнуть ситуация, где водитель катиться со склона и скорость не будет уменьшаться.

Уравнение при динамической мощности

Добавим к движущей силе (Fтр) коффициент $1 - \frac{3.6v_0}{100}$, который показывает изменения мощности двигателя, в зависимости от текущей скорости к максимальной на треке (100 км/ч = (100 * 1000/3600) м/с = $\frac{100}{3.6}$ м/с). Посмотрим как коэффициент повлиял на зависимость силы от скорости:

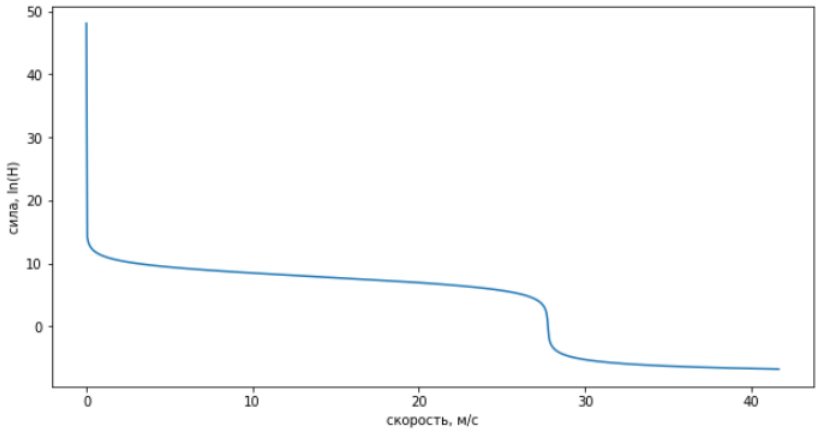


Рис. 2: зависимость силы от скорости ТС при динамической мощности

Как видно сила может становиться отрицательной, эту силу мы можем считать торможением двигателя, т.е. переход ТС на первую передачу для машин с двигателем внутреннего сгорания, для электрокаров это отключение двигателя и перевода механической энергии в элетрическую.

Новая система уравнений будет выглядеть так:

$$\begin{cases} x' = v_0 \\ v' = \dfrac{(\frac{P}{v_0}(1 - \dfrac{3.6v_0}{100}) - mg\sin\alpha - \frac{f}{R}mg\cos\alpha - C_x\frac{\rho v_0^2}{2}S)t^2}{2m} \end{cases}$$

Расчёт векторов состояния

Напишем функцию, возвращающую полученную систему уравнений в виде вектора состояния (x,v), где x - пройденное ТС расстояние и v - скорость в тот момент.

Далее напишем функцию останова (solout) для интегрирования, которая возвращает 0, если можно продолжать интегрирование и -1, если нельзя, также она будет добав-

лять вектор состояния в массив, а также момент временни и коэффициент мощности двигателя.

При помощи scipy.integrate.ode и функции останова интегрируем функцию движения ТС, в результате получаем массив со всеми векторами состояния, моментами времени и коэффициентами мощости.

Некоторые расчёты с помощью массива векторов состояния

Скорости:

- минимальная скорости в пути - 46 км/ч (поиск минимальной скорости идёт спустя 10 минут после старта)
- средняя скорость в пути - 74 км/ч
- максимальная скорость в пути - 104 км/ч

Также рассчитаем кол-во бензина затраченное на преодоления пути. Для этого найдём совершённую ТС работу, умножая мощность на коэффициент и на разность времени между двумя соседними точками и сложим получившиеся работы, далее делим результат на КПД двигателя внутреннего сгорания (0.4) и переводим в литры, поделив на 10 кВт*ч/л. Результат - 15.9 л

Графики

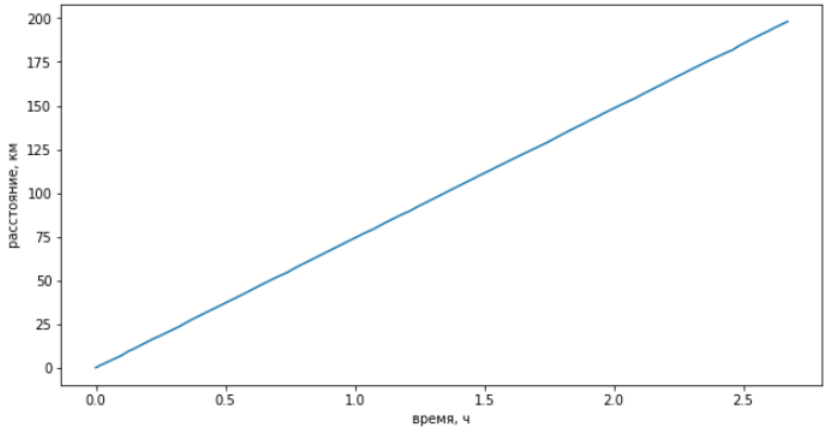


Рис. 3: Расстояние от времени ТС

Как видим расстояние имеет практические линейную зависимости от времени.

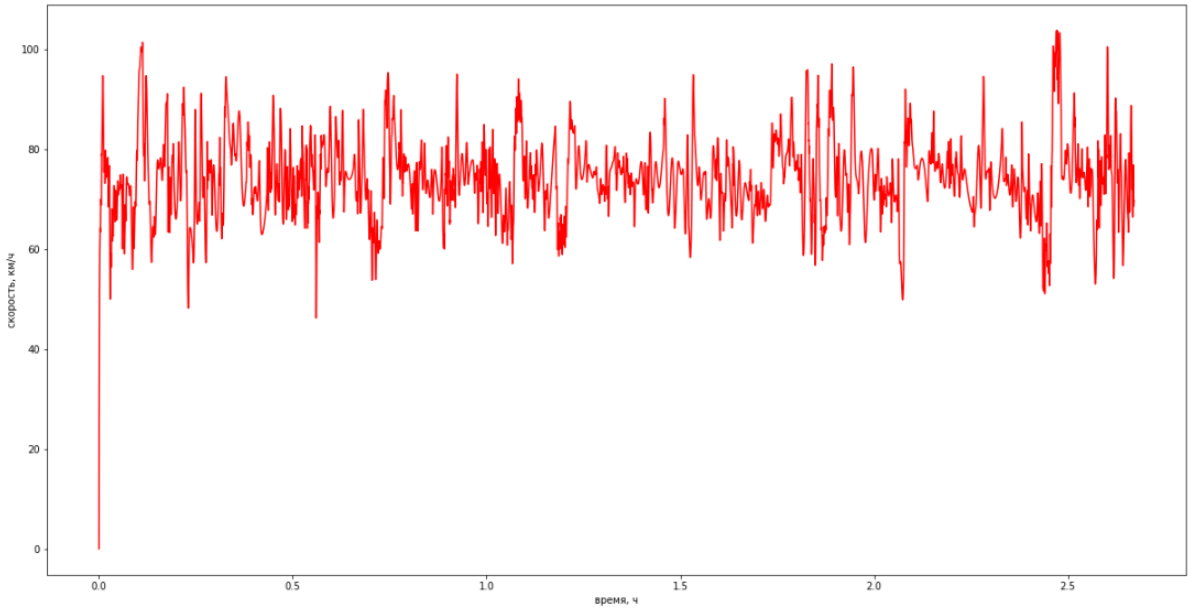


Рис. 4: Скорость от времени

Графики скорость от времени и расстояния являются практически идентичными, что ещё раз показывает их линейную зависимость.

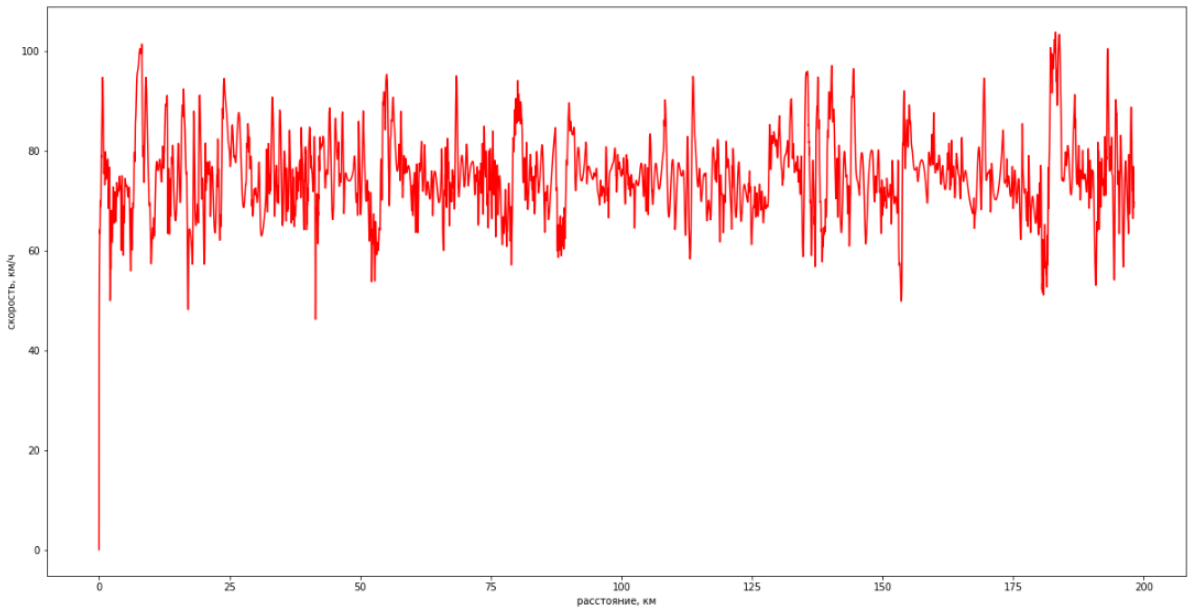


Рис. 5: Скорость от расстояния

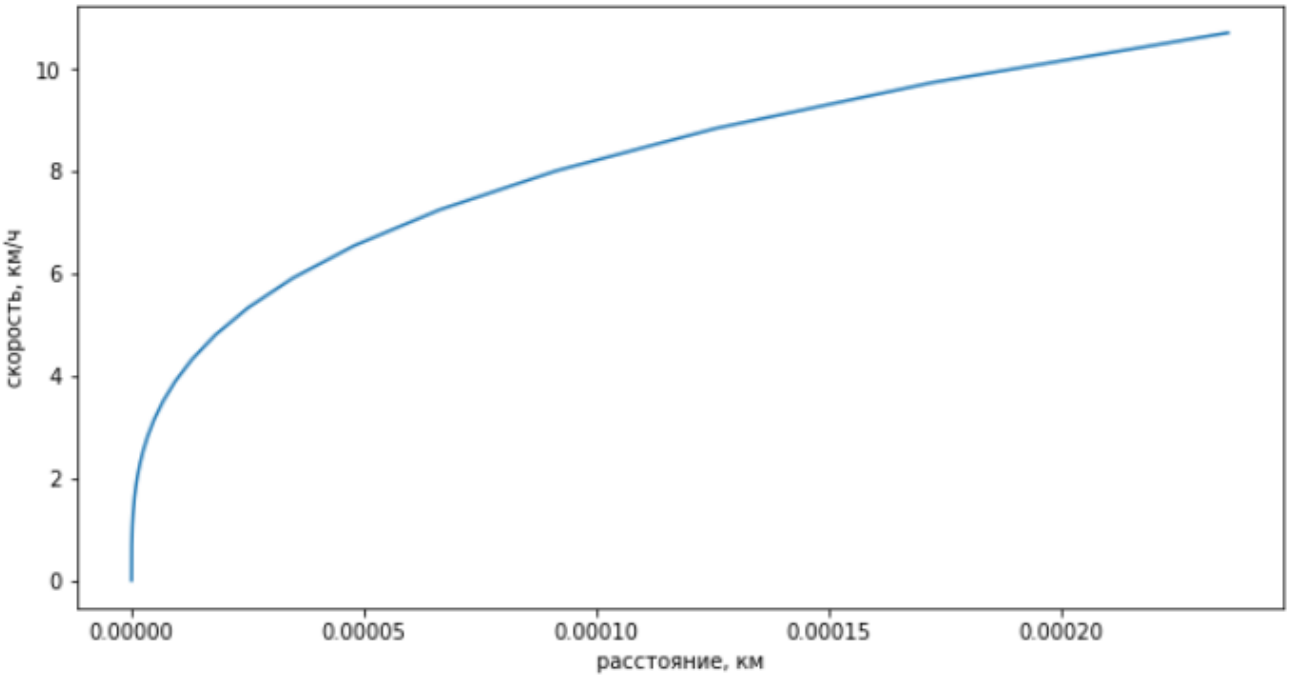


Рис. 6: График расстояния от скорости, где скорость была меньше 10 км/ч

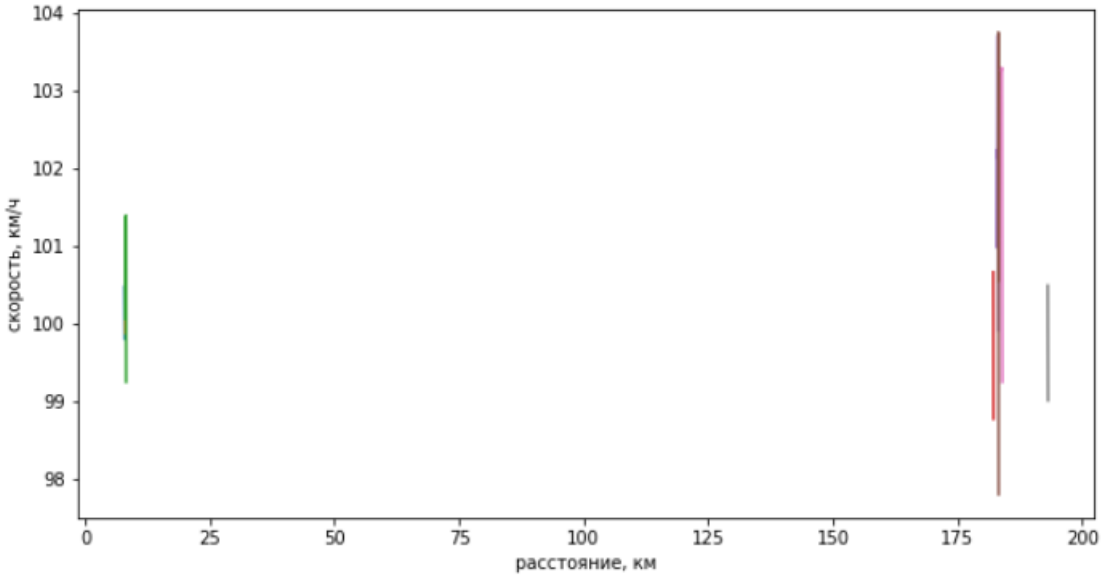


Рис. 7: График скорости от расстояния, где ТС движется более чем 100 км/ч

При достижении 100 км/ч ТС быстро сбрасывает скорость, в виду отрицательного ускорения (в реальности можно предположить, что водитель перешёл на первую передачу)

Проверим это, приблизив первый участок превышения скорости

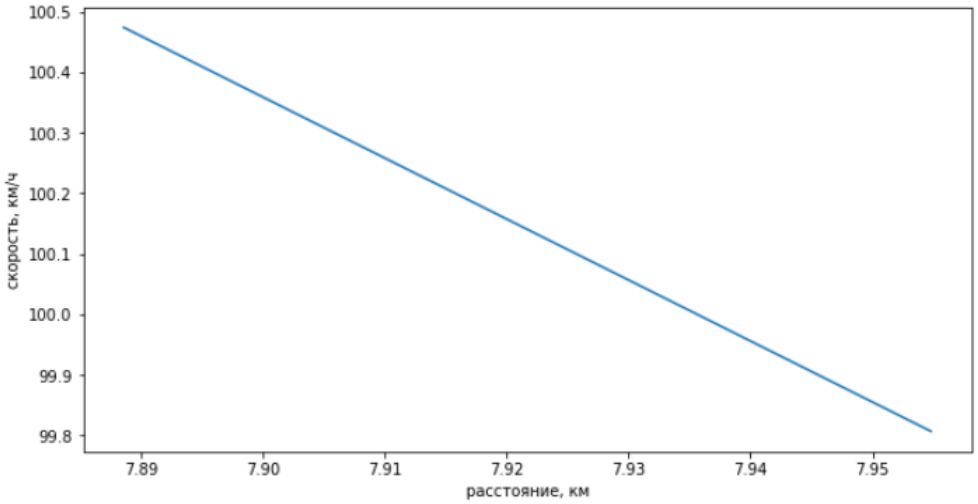


Рис. 8: Первый участок, где ТС превысил скорость

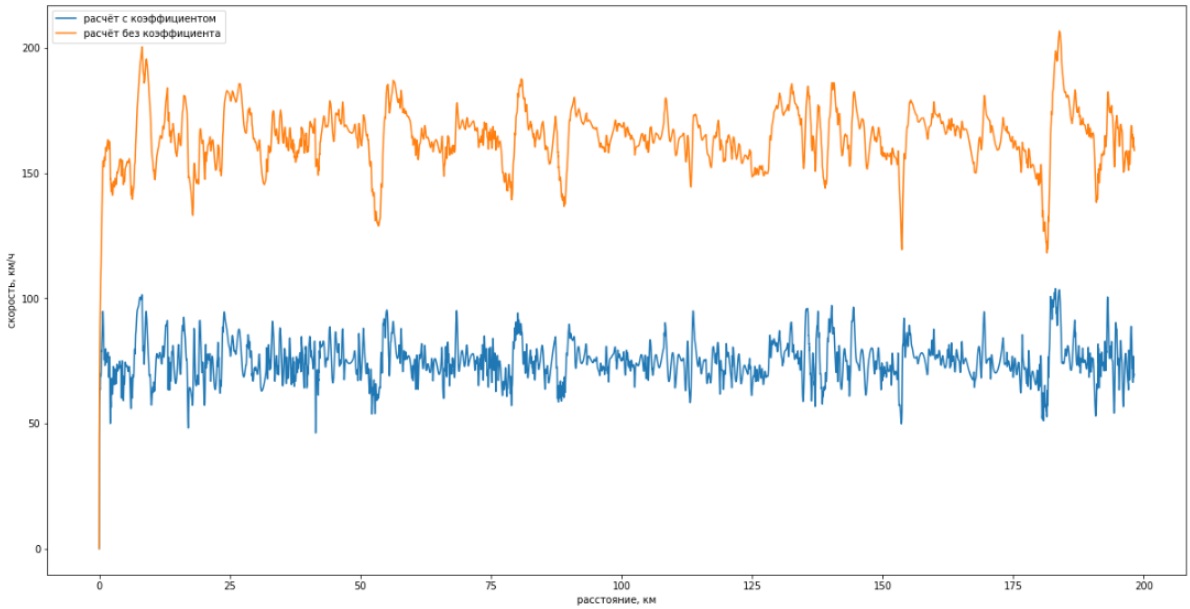


Рис. 9: Первый участок, где ТС превысил скорость

Из этого графика мы видим, как отличается скорость ТС без коэффициента, от чего такая модель имела бы сильную погрешность и её было бы сложно наложить на реальность, если максимальная скорость меньше 200 км/ч.

Заключение

Мы получили программу, способную рассчитать приблизительное движение ТС по треку и получили данные о его состоянии на протяжении всего пути, что позволяет использовать программу для организации маршрута и возможность выбора оптимального транспорта путём изменения характеристик ТС и сравнением результатов. Также есть возможность доработать уравнение движения для получения более точных данных а также использовать полученные векторы состояния для расчёта другой информации, если это необходимо.

Список литературы

[1] Онищенко О. Г., Коробко Б. А., Ващенко К. М. Структура, кинематика и динамика механизмов //Полтава: ПолНТУ. – 2010.

[2] Юрьев Б. Н. Экспериментальная аэродинамика. – Рипол Классик, 2013.

[3] Ильина В. А., Силаев П. К. Численные методы для физиков-теоретиков. – Институт компьютерных исследований, 2004. – С. 118-118.

[4] Bressert E. SciPy and NumPy: an overview for developers. – "O'Reilly Media, Inc. 2012.

[5] Tosi S. Matplotlib for Python developers. – Packt Publishing Ltd, 2009.

[6] Mitchell R. Web scraping with Python: Collecting more data from the modern web. – "O'Reilly Media, Inc. 2018.

[7] McKinney W. et al. pandas: a foundational Python library for data analysis and statistics //Python for High Performance and Scientific Computing. – 2011. – Т. 14. – №. 9.

Приложение

```
import numpy as np
import requests
import pandas as pd
from bs4 import BeautifulSoup
import math
import matplotlib.pyplot as plt
import scipy.interpolate
import scipy.integrate

def dist(llat1,llong1,llat2,llong2):
    rad = 6372795 lat1 = llat1*math.pi/180.
    lat2 = llat2*math.pi/180.
    long1 = llong1*math.pi/180.
    long2 = llong2*math.pi/180.
    cl1 = math.cos(lat1)
    cl2 = math.cos(lat2)
    sl1 = math.sin(lat1)
    sl2 = math.sin(lat2)
    delta = long2 - long1
    cdelta = math.cos(delta)
    sdelta = math.sin(delta)
    y = math.sqrt(math.pow(cl2*sdelta,2)+math.pow(cl1*sl2-sl1*cl2*cdelta,2))
    x = sl1*sl2+cl1*cl2*cdelta
    ad = math.atan2(y,x)
    dist = ad*rad
    return (dist)
data = np.genfromtxt('Vladivostok-Nahodka.csv',dtype=float,delimiter=',')
data = data[1:]
route = [[0,data[0][2]]]
for i in range(len(data)-1):
    a = [dist(data[i][0],data[i][1],data[i+1][0],data[i+1][1]),data[i+1][2]]
```

```
    route.append(a)
route=np.array(route)
route[:,0]=np.cumsum(route[:,0],dtype=float)
% matplotlib inline
xs = np.linspace(0,route[len(route)-1,0],1000000)
h_x = scipy.interpolate.InterpolatedUnivariateSpline(route[:,0],route[:,1], k=3)
fg = plt.figure(figsize=(20, 5), constrained_layout=True)
axes = fg.add_axes([0.1, 0.1, 0.8, 0.8])
plt.plot(route[:,0],route[:,1])
axes.set_xlabel('расстояние, м')
axes.set_ylabel('высота, м')
axes.set_title('График высоты от расстояния по точкам')
fg = plt.figure(figsize=(20, 5), constrained_layout=True)
xs = np.linspace(0,route[len(route)-1,0],1000000)
axes = fg.add_axes([0.1, 0.1, 0.8, 0.8])
plt.plot(xs,h_x(xs),label = 'кубический интерполянт')
plt.plot(route[:,0],route[:,1],label = 'график по точкам')
axes.legend(loc=2);
axes.set_title('Сравнение получившегося интерполянта с графиком по точкам')
axes.set_xlabel('расстояние, м')
axes.set_ylabel('высота, м')
fg = plt.figure(figsize=(20, 5), constrained_layout=True)
xs = np.linspace(0,route[len(route)-1,0],1000000)
axes = fg.add_axes([0.1, 0.1, 0.8, 0.8])
plt.plot(xs,h_x(xs))
axes.set_xlabel('расстояние, м')
axes.set_ylabel('высота, м')
alpha_x = h_x.derivative(n=1)
dens_url = "https://ru.wikipedia.org/wiki/Плотность_воздуха"
wiki_req = requests.get(dens_url).text
density = BeautifulSoup(wiki_req,'lxml')
density = density.find('table',class_='wikitable')
a=density.find_all('td')
```

```
table = []
for field in a:
    table.append(str(field))
for i in range(len(table)):
    table[i] = table[i].replace('<td>',"")
    table[i] = table[i].replace('</td>',"")
    table[i] = table[i].replace('\n',"")
    table[i] = table[i].replace(',','.')
    table[i] = table[i].replace(',','.')
    table[i] = table[i].replace('–','-')
for i in range(len(table)//4):
    table[i*4] = int(table[i*4])
    table[i*4+2] = float(table[i*4+2])
m = 1500
S = 2.5
Cx= 0.25
P = 735.5*125
kp= 0.8
kr = 0.05
t = -10
for i in range(len(table)//4):
    if(table[i*4]==t):
        ro = table[i*4+2]
def motion_ode(t,s,alpha_x,S,kp,m,kr,Cx,ro,P):
    alpha = math.atan(alpha_x(s[0]))
    g=9.81
    gx=g*math.sin(alpha)
    gy=g*math.cos(alpha)
    F = (P*kp)/(s[1]*m)
    k=(1-s[1]*3.6/100)
    return [s[1],F*k - gx - kr * gy - (Cx*ro*s[1]*s[1]*S)/(2*m)]
fig = plt.figure(figsize=(10, 5), constrained_layout=True)
axes = fig.add_axes([0.1, 0.1, 0.8, 0.8])
```

```
varr=np.linspace(1e-16,150/3.6,800)
Farr=[]
for v in varr:
    acc = (P*kp)/(v)
    if(acc > 0 ):
        Farr.append(math.log(acc))
    elif(acc < 0):
        Farr.append(-math.log(abs(acc)))
    else:
        Farr.append(0)
axes.plot(varr,Farr)
axes.set_xlabel('скорость, м/с')
axes.set_ylabel('сила, ln(H)')
fig = plt.figure(figsize=(10, 5), constrained_layout=True)
axes = fig.add_axes([0.1, 0.1, 0.8, 0.8])
Farr=[]
for v in varr:
    acc = (1-v*3.6/100)*(P*kp)/(v)
    if(acc > 0 ):
        Farr.append(math.log(acc))
    elif(acc < 0):
        Farr.append(-math.log(abs(acc)))
    else:
        Farr.append(0)
axes.plot(varr,Farr)
axes.set_xlabel('скорость, м/с')
axes.set_ylabel('сила, ln(H)')
def stopLength(t,s,L,lst):
    k=1-s[1]*3.6/100
    lst.append(np.hstack((s,t,k)))
    if (s[0] >= L):
        return -1
    return 0
```

```
L=route[-1,0]
v0 = 1e-16
s0 = (0,v0)
prop = scipy.integrate.ode(lambda t, s: motion_ode(t,s,alpha_x,S,kp,m,kr,Cx,ro,P))
prop.set_initial_value(s0, 0)
prop.set_integrator('dopri5', nsteps=1e5)
lst = []
prop.set_solout(lambda t, s: stopLength(t, s, L, lst))
prop.integrate(1e10)
arr = np.asarray(lst)
arr[:,0] = arr[:,0] / 1000
arr[:,1]=arr[:,1] * 3.6
arr[:,2]=arr[:,2] / 3600
fig = plt.figure(figsize=(10, 5), constrained_layout=True)
axes = fig.add_axes([0.1, 0.1, 0.8, 0.8])
axes.plot(arr[:,2],arr[:,0])
axes.set_xlabel('время, ч')
axes.set_ylabel('расстояние, км')
fig = plt.figure(figsize=(20, 10), constrained_layout=True)
axes = fig.add_axes([0.1, 0.1, 0.8, 0.8])
axes.plot(arr[:,2],arr[:,1],'r')
axes.set_xlabel('время, ч')
axes.set_ylabel('скорость, км/ч')
fig = plt.figure(figsize=(20, 10), constrained_layout=True)
axes = fig.add_axes([0.1, 0.1, 0.8, 0.8])
axes.plot(arr[:,0],arr[:,1],'r')
axes.set_xlabel('расстояние, км')
axes.set_ylabel('скорость, км/ч')
flag = False
vmax = arr[0,1]
for row in arr[1:]:
    if(row[2]>0.25 and flag == False):
        vmin = row[1]
```

```
            flag = True
        if (flag and vmin>row[1]):
            vmin = row[1]
        if(vmax<row[1]):
            vmax = row[1]
print("средняя скорость - round(arr[-1,0]/arr[-1,2],1))
print("максимальная скорость - round(vmax,1))
print("минимальная скорость - round(vmin,1))
A=0
for i in range(len(arr)-1):
    deltaT=(arr[i+1,2]-arr[i,2])
    A+=P*arr[i,3]*deltaT*3600/0.4
litres = A/36000000
print('Кол-во литров для преодоления пути -',round(litres,1))
fig = plt.figure(figsize=(10, 5), constrained_layout=True)
flag = False
start, end = 0,0
axes = fig.add_axes([0.1, 0.1, 0.8, 0.8])
for i in range(len(arr)):
    if(arr[i,1]<10 and flag == False):
        start = i
        flag = True
    if(arr[i,1]>=10 and flag == True):
        end=i+1
        flag = False
    if(end>0):
        axes.plot(arr[start:end,0],arr[start:end,1])
        start = 0
        end = 0
axes.set_xlabel('расстояние, км')
axes.set_ylabel('скорость, км/ч')
fig = plt.figure(figsize=(10, 5), constrained_layout=True)
flag = False
```

```
start, end = 0,0
axes = fig.add_axes([0.1, 0.1, 0.8, 0.8])
for i in range(len(arr)):
    if(arr[i,1]>100 and flag == False):
        start = i
        flag = True
    if(arr[i,1]<=100 and flag == True):
        end=i+1
        flag = False
    if(end>0):
        plt.plot(arr[start:end,0],arr[start:end,1])
        start = 0
        end = 0
axes.set_xlabel('расстояние, км')
axes.set_ylabel('скорость, км/ч')
fig = plt.figure(figsize=(10, 5), constrained_layout=True)
axes = fig.add_axes([0.1, 0.1, 0.8, 0.8])
flag = False
start, end = 0,0
for i in range(len(arr)):
    if(arr[i,1]>100 and flag == False):
        start = i
        flag = True
    if(arr[i,1]<=100 and flag == True):     end=i+1
        flag = False
    if(end>0):
        plt.plot(arr[start:end,0],arr[start:end,1])
        start = 0
        end = 0
        break
axes.set_xlabel('расстояние, км')
axes.set_ylabel('скорость, км/ч')
fig = plt.figure(figsize=(20, 10), constrained_layout=True)
```

```
axes = fig.add_axes([0.1, 0.1, 0.8, 0.8])
axes.plot(arr[:,0],arr[:,1],label = 'расчёт с коэффициентом')
axes.plot(arr2[:,0],arr2[:,1],label = 'расчёт без коэффициента')
axes.legend(loc=2);
axes.set_xlabel('расстояние, км')
axes.set_ylabel('скорость, км/ч')
```