ENGR 102 –Spring 2020

Exam #2 Take-home Component

Sharing Information about this exam with anyone will be considered an ethical violation

You will have 24 hours from the date/time of exam availability. Be sure to check eCampus for the deadline, the due date and time will be given on the same page where you downloaded this file.

This exam will be considered open notes and open book. In addition to the ZyBook and any material posted to the eCampus course page.

YOU MAY USE:

1. Your IDE of choice to write your solution program

2. **You may use limited online resources like the official numpy** and matplotlib tutorial/help pages, or StackOverflow, **with the one condition**. **You must develop and write your own code, do not copy / paste– we will know.**

YOU MAY NOT USE/ACCESS:

- Homework help sites where you post questions to be answered by "experts" (or by anyone else). In other words, **NO CHEGG** (or any other similar sites)! Use of these sites in any manner will be considered an ethical violation. **We will be actively checking for these solutions**.

Submission Requirements:

- All generated figures must include axis labels, legends, and a title. They must look nice
- Put all code into a single Python file called LastName_FirstName_track.py, where LastName is your last name and FirstName is your first name. Include the standard individual assignment header with the Aggie Code of Honor, your name, the date, section number, and exam number (instead of assignment).
- Put every single line that is printed to the console by your program (including any lines asking for input, or user input lines) into a solution document called LastName_FirstName_track.pdf. Also to be included in the solution document are your three required figures.
- Upload all generated files to the **Exam 2 Takehome Submission Box**.

Exam #2: (Take-home): 45 points

ACADEMIC INTEGRITY PLEDGE:

In submitting your completed exam solution materials to eCampus you pledge that you have neither given nor recieved aid in completing this exam. In addition, you are pledging that you have followed the stricutres of the Texas A&M University Aggie Code of honor throughout the examination process.

An Aggie does not lie, cheat or steal or tolerate those who do.

Question 1. (45 points)

Imagine that you ae working as a race engineer for Helio Castroneves in Team Penske. We are participating in Monterey SportsCar Championship race at WeatherTech Raceway Laguna Seca. After the last free practice, our hot lap is slower than our Teammate Juan Pablo Montoya. However, previous to change any set up in the Acura Team Penske ARX-05 car (Fig.1) we have to identify the locations in the RaceTrac, where Helio's car has been slower than Montoyas car.

Figure 1. Acura Team Penske ARX-05 racecar.

To help you in this analysis, you have been given a data file *hot_laps.csv* that contains lap information for both drivers. The data includes the sector numbering in the racetrack, an incremental lap time computed in each sector, distance of each sector from the starting point (miles) and the (*x,y*) position along the race track (feets). Refer to the file header to obtain the correct ordering of the data columns before reading the file. Note that in the data file, sector one is computed twice: start and end of the time measurement. You are going to develop a python program that will allow comparing the lap performance and providing guidance to your driver.

- First, you are going to create a **function that reads** the given file and outputs a set of lists. The function takes in as input the name of the file and returns 6 lists: file header, time and distance for driver 1, time and distance for driver 2, race sectors, and (*x,y*) location. In the **main program**, you are going to ask the user to **input the name of the file** and then you are going to **call the function**. Use a **try-except** statement to check that the users enter a valid name. If not, print an appropriate statement and the program will keep asking until the users enter the correct name.

- Second, you are going to create a **function that computes the speed** of the car for each sector. This function takes in two individual lists for time and distance and returns one list with velocities. In order to compute the speed for each sector you are going to create another **function that performs** the **numerical derivative using backward differences**:

$$v_i = \frac{x_i - x_{i-1}}{t_i - t_{i-1}}$$

this function takes in one list with the individual value for x_i, x_{i-1} and another list with the individual value for $t_i - t_{i-1}$ and returns v_i. Note that the first velocity you are computing is for sector 2. Once we append all velocities in the function that computes the speed you are going to consider that the

velocity for time 0 (sector 1 = start point) is the same as the last velocity computed in the list (sector 1 = end point).

- Third, you are going to create a **function that computes the velocity in any location in the track**. This function will take in a list of distances inputted by the user, and the list of velocity and distance for the driver. For each individual value you will need to find the interval corresponding to that position x_(i-1) ≤ x_user < x_i and save in new variables $[x_0\ x_1]$ as well as the corresponding velocity to i-1 and i $[v_0\ v_1]$ respectively. With this values you are going to obtain the coefficients *m,b* for the linear interpolation equation:

$$v = m * x + b$$

To obtain m and v you are going to solve the following linear system of equations using np.linalg.solve:

$$\begin{bmatrix} x_0 & 1 \\ x_1 & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} = \begin{bmatrix} v_0 \\ v_1 \end{bmatrix}$$

Finally you are going to create another function for the previously described linear equation, where the input will be m and v and the distance inputted by the user x_user and the output will be the velocity.

- Fourth, you are going to ask the user to input a list of distances (any length) and print the distance and velocity in the console with two decimal places. You have to check the validity of the input. Note that the user cannot enter values longer than the max distance of the racetrack.

- Fifth, you are going to create two **independent functions** that computes the **negative local min and positive local max** of a data set. Both functions will take in a list of values and will return the local values and position in the list. See Fig 2b for the illustrative representation of local min and local max.

- Sixth, you have to plot three figures i.e.(Fig 2): 1st speed vs distance for the two drivers, 2nd speed_castroneves-speed_montoya vs distance including the local min (points where Castroneves is slower) and max (points where Castroneves is faster), 3rd the racetrack with the local min and max locations (here you have to use the (x,y list)).

- Finally, you are going to create a **function** that takes in 6 list: header list, sector list, time and distance for driver 1, time and distance for driver 2, velocity list for drive 1, and the local min position list, and will **write in a file named yourname.csv**. This file will include the following columns separated by commas for Castroneves: Race_sector, Time[m:s] in minutes and seconds format, Time_split[s] Time Castroneves-Time Montoya, Velocity Castoneves [mph] and in Observation you are going to put 'slower sector' in the corresponding local minimum positions.

- Looking all figures and information, where is your driver slower?

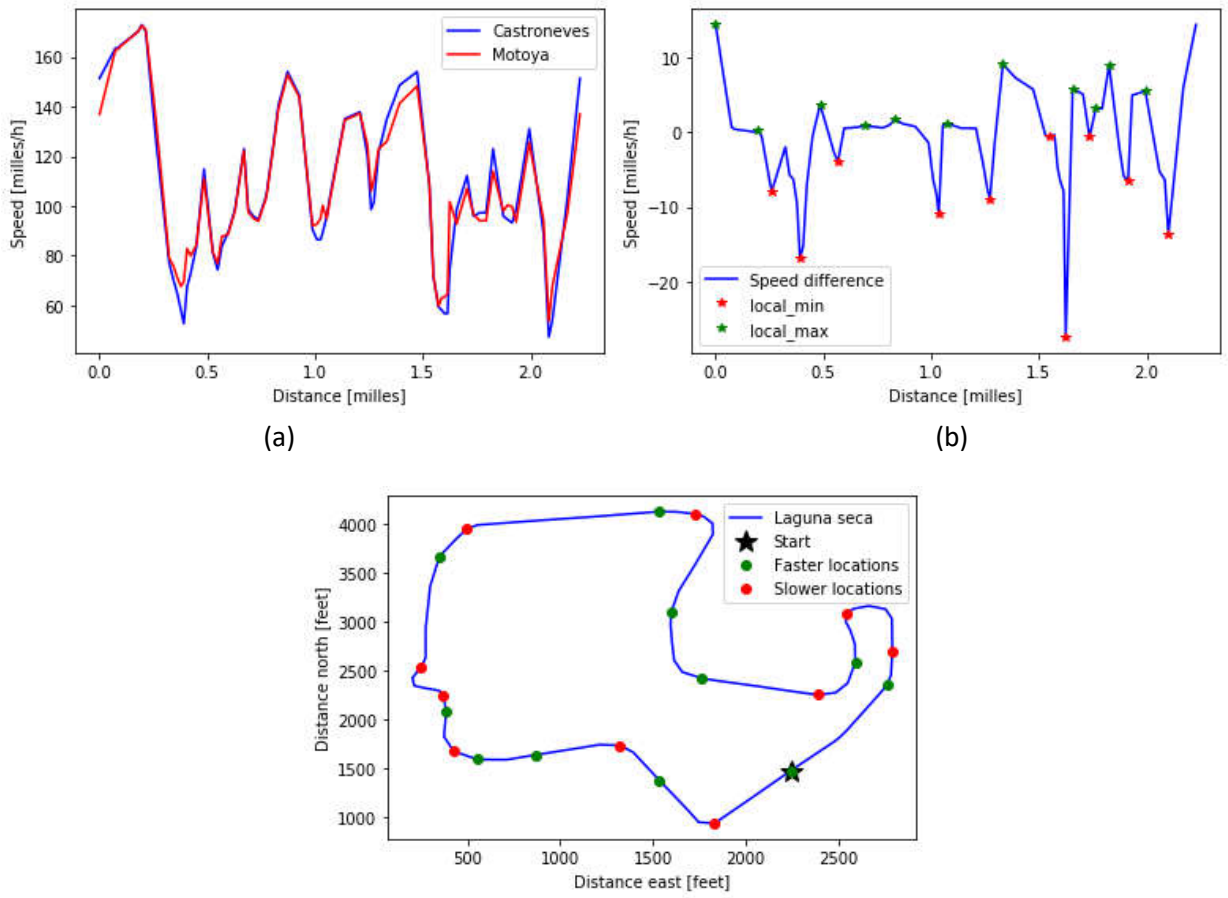

Figure 2. a) Speed vs Distance for both drivers, b) speed_castroneves-speed_montoya vs distance including the local min (green star) and max (red star), and Laguna Seca racetrack he racetrack with the local min (slower) and max locations (Faster).