

OpenLapSim

The Open Source Lap Time Simulation Software

Documentation

Autor: Davide Strassera
First release: 21th December 2019

Abstract

The aim of this project is to build an open source Lap Time Simulation software, which can be used as frame for further improvements and give an understanding of the main functionality of such a tool.

The software as it is can be used for preliminary study related to race car and help to compare some of the main trade off of setup configurations or design (aero, power, mass, grip), but as a community project can be further developed to a more advanced tool.

Intro

The *OpenLapSim* is a quasi-static Lap Time Simulator for a point mass vehicle, with aerodynamics forces, engine map, gear ratios and tire friction ellipse, written in Python.

The program simulates the maximum performance (accelerations and velocity) of the given vehicle-setup on all the sections of the circuit, computing the fastest lap time and highest speed trace over the lap the car could theoretically achieve.

Methodology

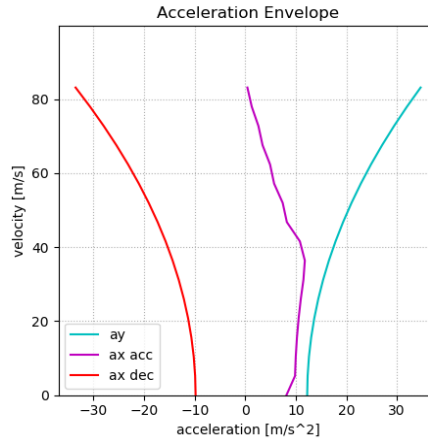
For the development of this Software it has been used Python 3.7 along with Spyder IDE, which allows to easily visualize the variable space (explorer). Python is a free programming language very powerful for mathematical modelling. The main external libraries used are NumPy, SciPy and Matplotlib.

The programming paradigm used in for this software is mainly Object Oriented, to facilitate the modular maintenance and the development of the program. Moreover, where possible, the functionalities have been divided in different classes and methods to help with an easier understand of the computational phases and help with testing and debugging as well.

Theory

The structure of the Lap time simulator is divided mainly in two components: the Acceleration Envelope calculator and the Lap Time Simulation calculator for the speed trace and lap time computations.

The *Acceleration Envelope*, also called GGK diagram [1], describes the maximum accelerations of the vehicle on the axis of velocity, from 0 to max speed, for positive (a_x acc), negative (a_x dec), which is braking phase and lateral direction (a_y).



To calculate it the program uses the setup parameters (defined by the user) and calculates the forces equilibrium for both longitudinal and lateral directions for each point of the speed vector.

The *Lap Time Simulator* uses both the results from the previous routine and the track file.

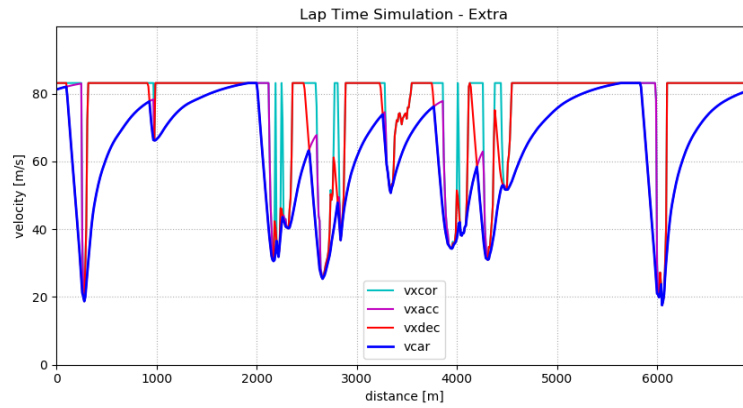
The TrackFile.txt is a two columns matrix which contains the distance [m] and curvature ($1/\text{corner radius}$) of the circuit.

This routine calculates for each segment of the circuit (row of the track file) the maximum lateral acceleration the car could performed based on the GGK diagram, and then integrates it over the distance to compute the fastest speed trace (in this phase for the corner) over the circuit [2].

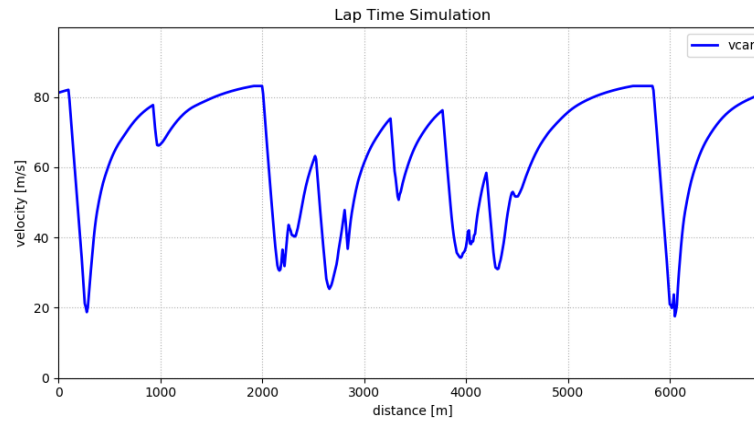
The same calculation is computed for the longitudinal, in both acceleration (positive) and deceleration (negative) phases. On the Long calculation we have to consider the amount of lateral acceleration the vehicle has, which means the car has a “combination” of accelerations and it is moving out from the pure longitudinal point of the GGK, if not in straight line.

In this tool the combined performance between long and lateral accelerations is calculated with the equation of the ellipse (simple ellipse of adherence).

Below in the chart we can see all the velocity traces calculated during the Lap Time Sim process: max corner speed (v_{xcor}), max acceleration speed (v_{xacc}) and max braking speed (v_{xdec}).



Finally, for each segment of the circuit, it is taken the minimum value between the three speed traces calculated. Below it is plotted the fastest car velocity (v_{car}) over the circuit distance.



Software Guide

The structure of the code inside `OpenLapSim/src` is as following:

- 📁 exportFiles
- 📁 setupFiles
- 📁 trackFiles
- 📁 utilities
- 📄 AccEnvCalc.py
- 📄 LapTimeSimCalc.py
- 📄 PostProc.py
- 📄 RunOpenLapSim.py
- 📄 SetupFileLoader.py

The file needed to run the simulation is **RunOpenLapSim.py**, where the user needs to specify two files: `SetupFile.json` and `TrackFile.txt`

```

-----
Select Files
-----
"""
# SetupFile.json
setupFileName = "SetupFile.json"

# TrackFile.txt
trackFileName = "TrackFile.txt"

bExport = 1
bPlotExtra = 0

```

Moreover, you can specify two other parameters: `bExport` and `bPlotExtra`. The first one if `== 1` will save the result of the simulation, as .txt file automatically named with the date, containing two columns: distance [m] and vcar [m/s] inside **/exportFiles** directory. The second parameter is used only to plot extra information as showed in a previous figure.

The directory **/trackFiles** stores the Trackfile.txt and is the location where you should save all the other track files. TrackFile.txt contains two columns: distance in meters and curvature (1/R) of the specific circuit (see utilities for info on how generate it).

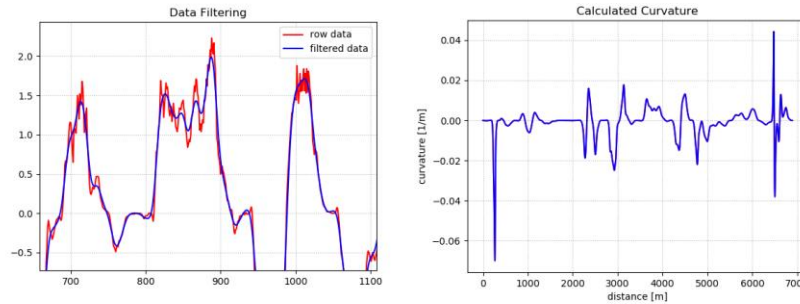
The directory **/setupFiles** stores the SetupFile.json. and is the location where you should save all the other setup files. The SetupFile.json contains all the parameters of the vehicle and the respective units are specified in the SetupFileLoader.py. (comments are not part of the JSON file and must be removed).

```

{
  "setupName" : "Gp2Dummy",
  "mcar"      : 728,          #[Kg]; total car mass
  "clt"       : 3.1,          #[100 pt.]; Lift coeffitien (-)
  "cx"        : 1.0,          #[100 pt.]; Drag coeffitien
  "afrcar"    : 1.0,          #[m2]; Frontal Area
  "rho"       : 1.22          #[Kg/m3]; air density
  "mbrk"      : 7000,         #[m2]; Max Braking Torque
  "gripX"     : 1.0,          #tyre friction coeff Long
  "gripY"     : 1.25,         #tyre friction coeff Lat
  "rtyre"     : 0.32,         #[m]; tyre radius
  "reff"      : 0.95,         # drive line efficiency
  "rGearRat"  : [10.0,7.8,6.1,7.8,5.2,4.5,4.0], #Final Gear Ratio
  "EngNm"     : [200,300,430,380], # [Nm]; Engine Torque
  "EngRpm"    : [0,3000,7000,10000], # [rpm]Engine rmp
}

```

The directory **/utilities** contain a routine “TrackFileBuilder.py” which given telemetryFile.csv (for example a real telemetry data export), containing at least distance, car speed and lateral acceleration, can automatically generate the TrackFile.txt in the format needed here. There is some low pass filtering to smooth noisy data, but you may need to adjust cut off frequency based on your necessity (see plot below, left). For more info on how to use it, see the comments inside the class.



After successful completion of the simulation you should see on the console these lines:

```
AccEnvCalc completed
LapSimTimeCalc completed
LapSimTimeCalc completed
PostProc completed
-----
LapTime: 121.836 [s]
TopSpeed: 299.2 [Km/h]
-----
```

Conclusions

This Project does not want to be an advanced tool for simulation given the large amount of assumption and simplifications (up to this first release). It is anyway a ready to use tool which can be furthermore developed or be used as an education software for passionate engineers and universities projects (such as Formula Students).

It is important to remember that the simulation results in general are both a combination of model approximation and correctness of the input data, especially in a Formula type car, where aerodynamics and tire model heavily modify the results.

Links

The source code is available on GitHub and is licensed under GNU General Public License: <https://github.com/dstrassera/OpenLapSim>

References

- [1] W. F. Milliken, D. L. Milliken, *Race Car Vehicle Dynamics*. Warrendale, Pa: SAE, 1995
- [2] J. Hakewill, *Lap Time Simulation*. 2000 [Online], available: http://www.jameshakewill.com/Lap_Time_Simulation.pdf
- [3] M. Guiggiani, *Dinamica del Veicolo*. De Agostini, Novara, It, 2007