Matlab Steady State Performance Simulator v1.1

César Álvarez Porras

4 – December – 2013

Content Index

1. Project Description

The Performance Simulator has been developed as a tool to understand the principles of vehicle dynamics and performance simulation.

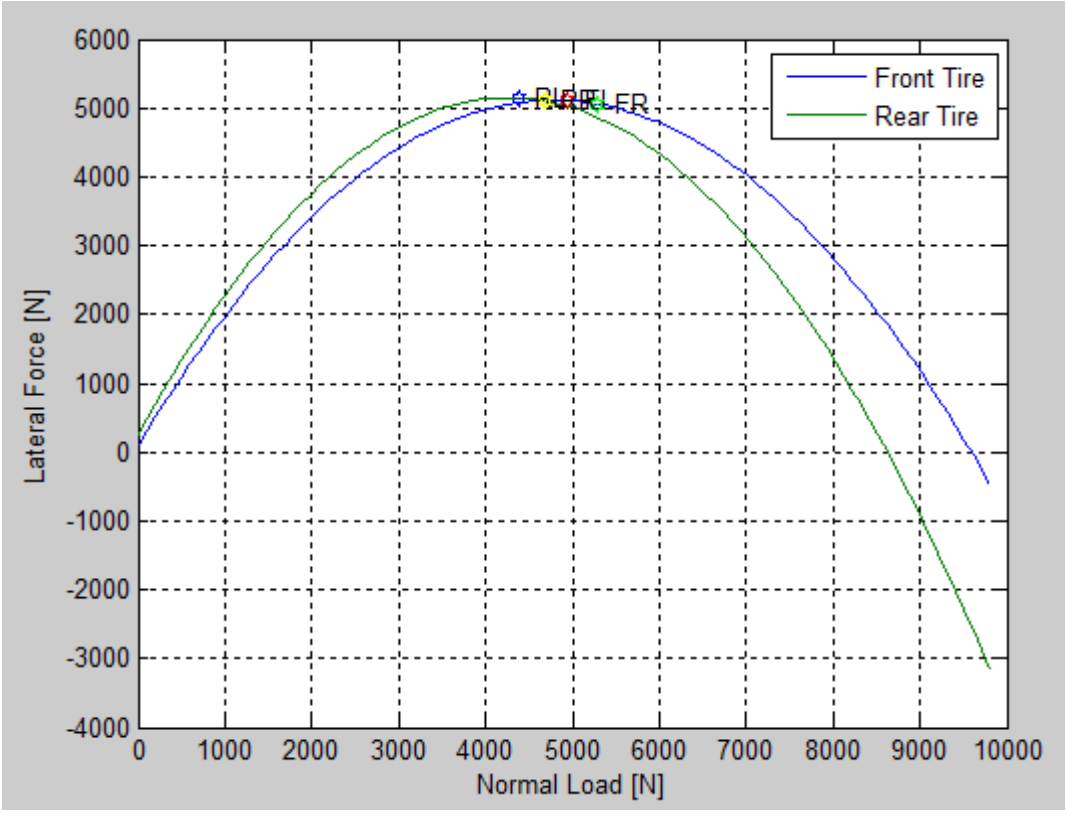
I considered Matlab as the best starting point since it gives the user a framework where one only has to make a correct use of the tools Matlab offers. However this code can be easily transformed to C, C++ or Python so that it can be executed as a part of a bigger application without any dependency to Matlab.

1.1 Steady State Corner Simulator

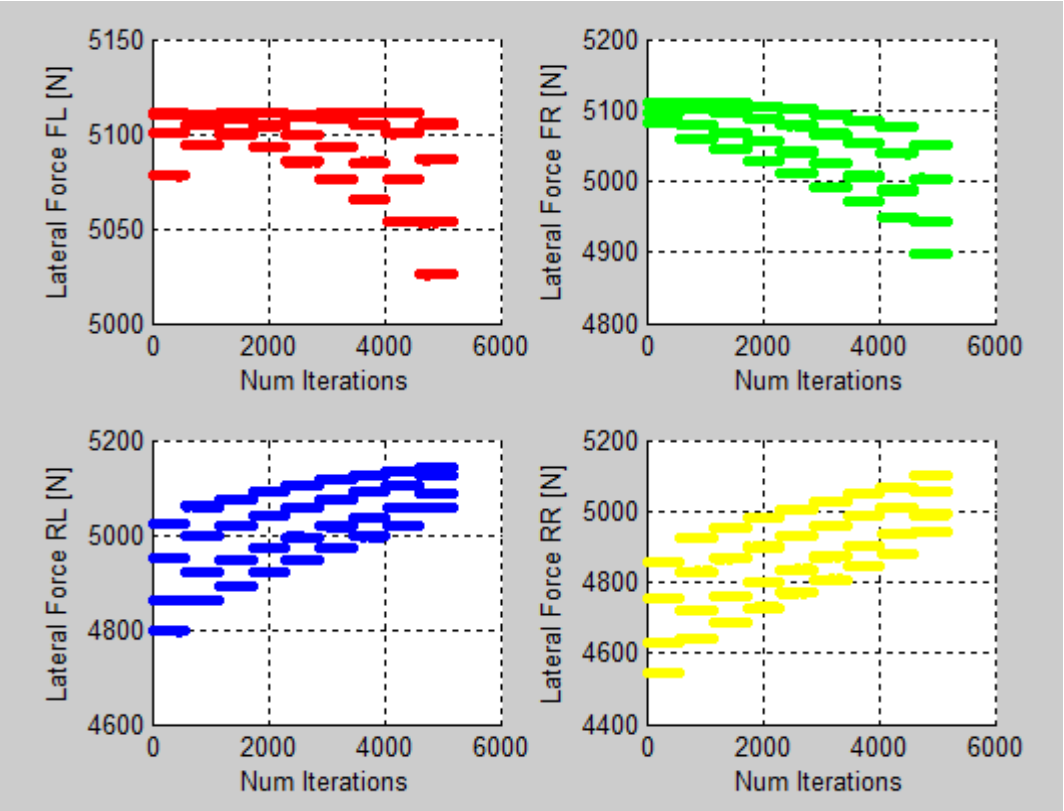
I first started programming a Steady State Corner Simulator. It gives the best configuration for a vehicle based on the different setup options, a steady state study in corner and a basic tire model. This Steady State Corner Simulator returns which setup configuration calculates the biggest lateral force generated by the tires. The tire model is a function of the vertical load of every tire.

To make this calculations I had to declare a set of initial conditions which are a vehicle speed – in order to calculate the down force –, and the lateral acceleration – so that I can calculate the lateral load transfer.

Since the vehicle speed and the lateral acceleration will depend on the maximum grip given by the tires (result of the simulator), these initial conditions will have to be changed for the result of the first iteration.



The aim of this code is to find the setup configuration that provides the best normal load in order to get the highest lateral force from the tires.



Evolution of the lateral force generated by every tire through out all the setup iterations. Every point on these four graphs represents the lateral force generated by the tire for a different setup combination.

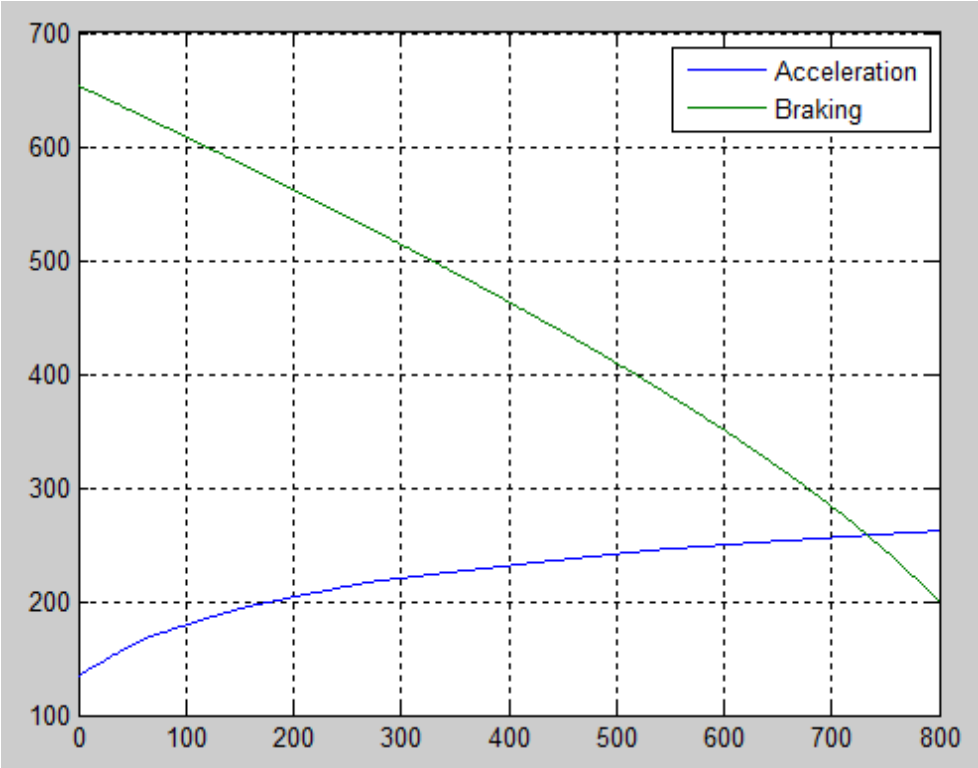
1.2 Steady State Straight Simulator

Next step was to make some kind of simulation for the straight segments. I started coding the acceleration moment on a straight basing my calculations on the Torque delivered by the engine at every RPM range. Few other parameters, such as Total Mass, traction tire radius and gearbox ratios are needed to calculate what is the acceleration capacity of the vehicle. The speed at the point *n* is calculated based on the speed in *n-1* and the acceleration given by the engine at the point *n*.

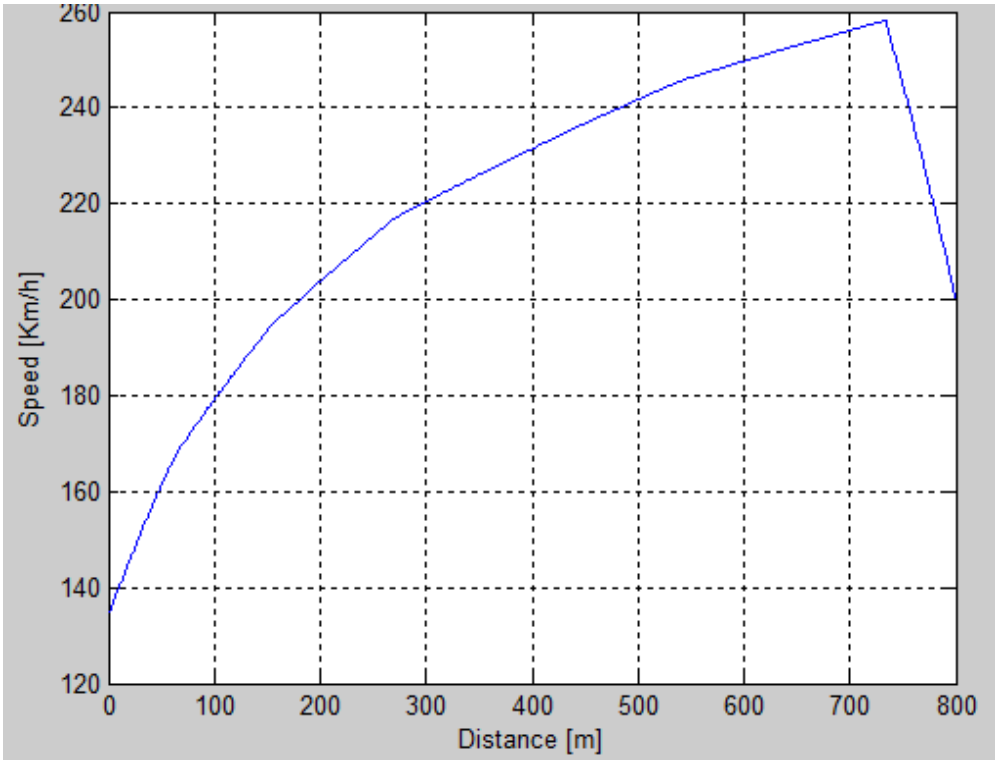
Besides the code calculates what is the best gear for the given gear ratios and the initial speed looking for the RPM range that gives the highest torque for a certain wheel speed.

Regarding the braking phase I supposed a constant braking force of 1G. The deceleration on every point is calculated based on the speed in the next point and the braking force provided by the brakes.

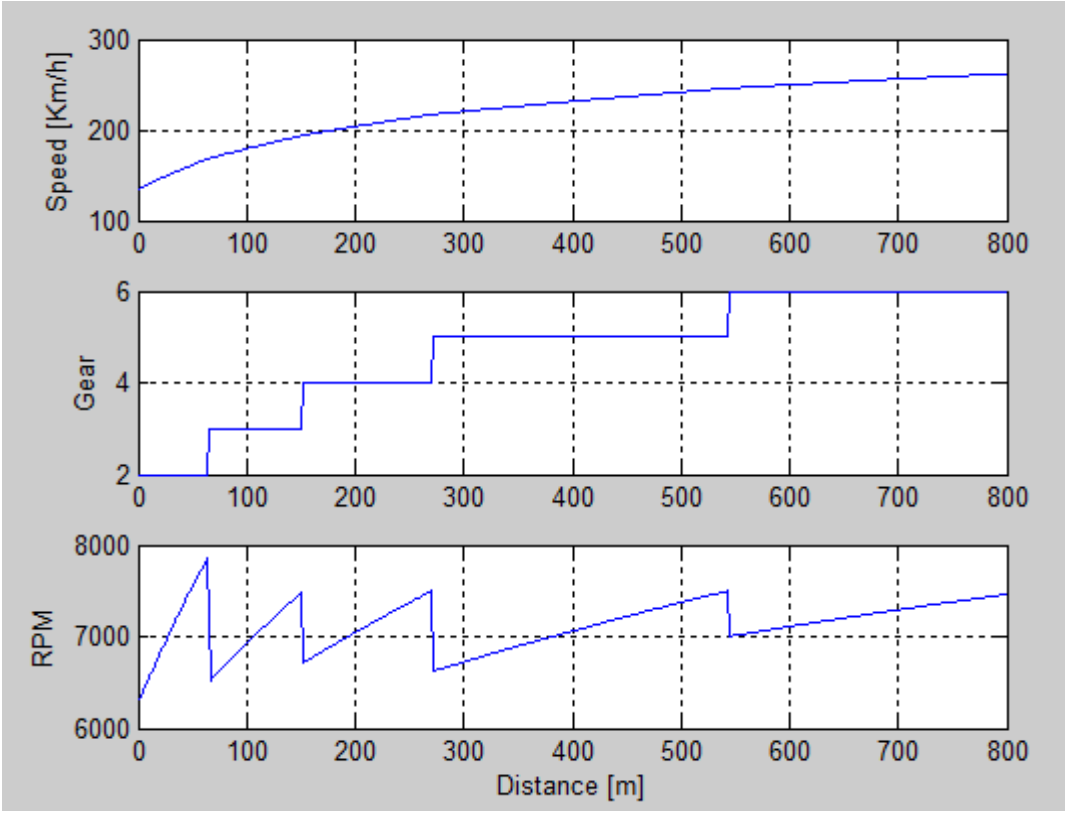
The initial conditions for the straight simulation are the straight length, initial speed and final speed. In the final Performance Simulator these conditions will be given by the maximum speed through the previous and next corners.



Straight Simulation for a 800 meter length segment, initial speed 135 Km/h and final speed 200 Km/h. Using the force from the engine I calculate the acceleration using the initial speed. The braking force is calculated backwards using the braking force and the final speed.



The resultant speed trace takes part of acceleration curve and part of the braking curve. The Steady State Simulator also calculates what is the RPM and Gear evolution through the straight.



1.3 Steady State Simulator

This code intends to merge the two previous simulators to calculate the best setup option.

For a given vehicle model where the following parameters are defined:

```
%Vehicle
Total_Mass = 1200;% Kg
US_Mass_f = 40;% Kg
US_Mass_r = 46;% Kg
Wheelbase = 2.890;% m
Weight_distribution = [0.45, 0.47, 0.48, 0.49, 0.5, 0.51, 0.52, 0.53, 0.55];
track_f = 1.698;% m
track_r = 1.620;% m
hcg = 0.45;% General CG height m
hrf = 0;% Front Roll Centre height (measured from the ground) m
hrr = 0;% rear Roll Centre height (measured from the ground) m
huf = 0.33;% Front Unsprung Mass CG height m
hur = 0.355;% Rear Unsprung Mass CG height m

% Drivetrain
Torque = [300, 330, 370, 400, 450, 470, 520, 580, 620, 660, 680, 670, 640,
          620, 600, 590, 570];
RPM = [2000,2500,3000,3500,4000,4500,5000,5500,6000,6500,7000,7500,
          8000,8500,9000,9500,10000];
% Torque and RPM arrays must have the same length
Gear_Ratios = [3.1, 2.3, 1.9, 1.7, 1.5, 1.4];
% Gear Ratios, Torque and RPM admit multiple sets i.e. Gear_Ratios = [[a, b, c, d, e,
   f];[g, h, i, j, k, l]] but these are taken in sets (a,b,c,d,e,f)  for one iteration
   and (g,h,i,j,k,l) for the next one.
Final_Ratio = 2.8;
Tire_Radious = hur;% m
Brakes_G_Force = 1; % G
rolling_coef = 0.396;
gearshift_delay = 0.02; % s

% Suspension
MR_wheel_spring_f = 1; % Front Motion Ratio Wheel-Spring
MR_wheel_spring_r = 1; % rear Motion Ratio Wheel-Spring
MR_ARB_f = 0.8; % Front Motion Ratio ARB
MR_ARB_r = 0.8; % Rear Motion Ratio ARB
K_spring_F = [507500, 545000, 595000, 625000];% Nm
K_spring_R = [507500, 545000, 595000, 625000];% Nm
K_ARB_F = [325000, 360000, 400000];% Nm
K_ARB_R = [325000, 360000, 400000];% Nm

% Aerodynamics
% The Downforce Coef, Drag Coef and Aerodynamic Area arrays must have
% the same length, assuming one change on any of these parameters also
% involves a change on the other two.
Downforce_Coef = [1.4, 1.5, 1.6, 1.65];%
Drag_Coef = [0.3, 0.35, 0.45 0.55];%
Area = [1.375, 1.400, 1.425, 1.45];% m^2

% Tire model
K_tire_f = 95000;% Nm
K_tire_r = 95000;% Nm
c2=[-0.00022, -0.00027];% Tire model parameters
c1=[2.1, 2.3];% Tire model parameters
c0=[100, 250];% Tire model parameters

% Scenario
% ========================
air_t = 23;% C degrees
atm_p = 1000;% mbar
```

The code calculates the maximum speed through the corners for every combination of these parameters.

The circuit model will be defined by segments in the following form:

```
track_length = 500; % m
Distance =      [0, 200, 210, 220, 225, 230, 280, 285, 300, 310, 325, 335];
Corner_Radius = [0, 120, 110, 100, 110, 0,   90,  85,  80,  70,  75,  0];
```

Where a *`Corner Radius(n) = 0`* means the segment is a straight of length *`Distance(n+1) - Distance(n)`*. *`Corner Radius(n) = 85`* means the segment is a corner of radius 85 metres and length *`Distance(n+1) - Distance(n)`*. The last segment's length is calculated using the *`track_length`* parameter.

Note the first and last segments must be straights and a straight is defined as the segment between two corners. There cannot be two straights together.

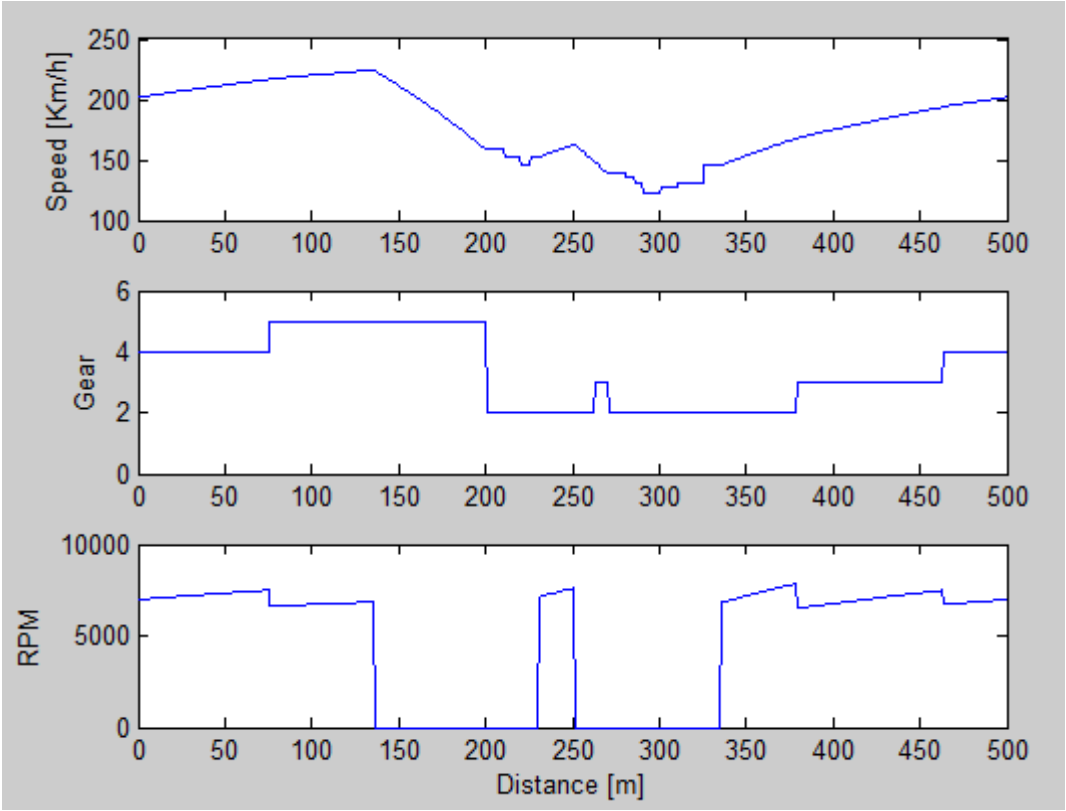
Once we know the maximum speeds the vehicle can handle on the corners we know the initial and final speeds for the straight sectors. We consider the speed is constant in the corner segments.

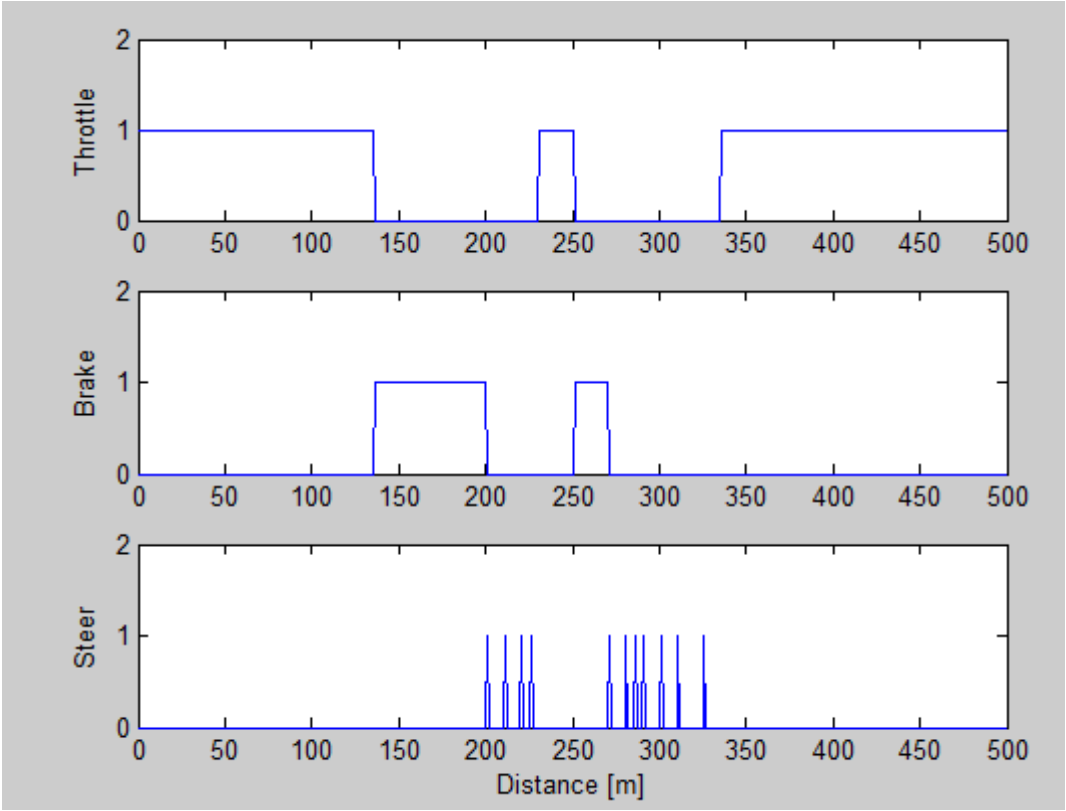
Knowing the speed and the distance of every segment we can easily calculate the time needed for every segment.

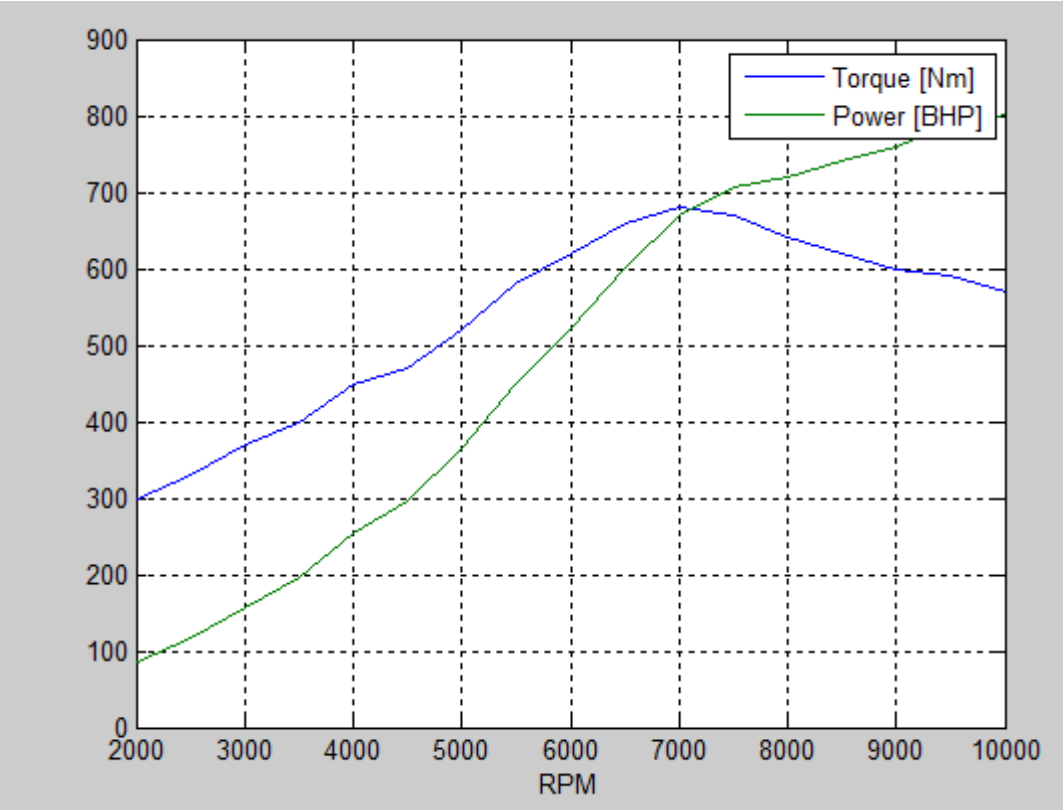
The best setup configuration will be the combination that has the smallest aggregated time to complete all the segments. The Simulator outputs the results in a report form and also graphically:

```
The quickest time for this circuit is 1.014755e+001 seconds for the
following configuration:
Total Mass: 1200 Kg, US Mass F: 40 Kg, US Mass R: 46 Kg
Wheelbase: 2.890000e+000 m, Weight Distribution: 5.500000e-001
Track F: 1.698000e+000 m, Track R: 1.620000e+000 m
Hcg: 4.500000e-001 m, Hrf: 0 m, Hrr: 0 m, Huf: 3.300000e-001 m, Hur:
3.550000e-001 m
Torque-RPM:
r_torque = 300 330 370 400 450 470 520 580 620 660 680 670 640 620 600 590
           570
r_rpm = 2000 2500 3000 3500 4000 4500 5000 5500 6000 6500 7000 7500 8000
           8500 9000 9500 10000
Gear Ratios: 3.1000    2.3000    1.9000    1.7000    1.5000    1.4000
Final Ratio: 2.800000e+000
Tire Radius: 3.550000e-001 m, Brakes G Force: 1 G, Rolling Coef: 3.960000e-
001
MR Wheel Spring F: 1, MR Wheel Spring R: 1
MR ARB F: 8.000000e-001, MR ARB R: 8.000000e-001
K Spring F: 507500 N, K Spring R: 625000 N
K ARB F: 325000 N, K ARB R: 400000 N
Downforce Coef: 1.400000e+000, Drag Coef: 3.000000e-001, Area:
1.375000e+000
K Tire F: 95000 N, K Tire R: 95000 N
Air temp: 23 °C, Atm P: 1000 mbar
```

The figures below show the graphical outputs.


Speed [Km/h]
250
200
150
100
0 50 100 150 200 250 300 350 400 450 500
Gear
6
4
2
0
0 50 100 150 200 250 300 350 400 450 500
RPM
10000
5000
0
0 50 100 150 200 250 300 350 400 450 500
Distance [m]


Throttle
2
1
0
0 50 100 150 200 250 300 350 400 450 500
Brake
2
1
0
0 50 100 150 200 250 300 350 400 450 500
Steer
2
1
0
0 50 100 150 200 250 300 350 400 450 500
Distance [m]


Torque [Nm]
Power [BHP]
900
800
700
600
500
400
300
200
100
0
2000
3000
4000
5000
6000
7000
8000
9000
10000
RPM

2. Limitations

I. The tire model is only based on vertical load, while it would be better described as a function of both vertical load and slip angle.

II. The circuit model does not admit two straight segments one after another since the initial and final speeds for the straight segments are mandatory parameters when calculating the straight performance. These speeds can only be calculated if there is a corner before and after a straight. However two straights together can be defined as a single straight with a length equal to the sum of the first two.

III. The segments' length must be at least 2 meters.

IV. The Straight Simulator code can only calculate the speed evolution when the initial speed in higher than 32 Km/h. Due to the algorithm used when launching the vehicle at a speed lower than 32 Km/h the vehicle can't overcome the drag and rolling resistances.

3. Further Development Areas

I. A better tire model should be developed before taking the results of this simulator serious.

II. The braking model can also be improved by modifying the braking force from a constant value to a function of the time the brakes have been pressed, or the braking distances. We do know the force of the brakes is not constant.

III. This simulator is based on Steady State models. This means the behaviour and reactions on every point are calculated as isolated events from the previous and next points. There are plenty of space to carry out further developments in Transient State Simulation.

IV. The vehicle model has also a great potential of development by adding independent downforces for the front and rear axes

V. An add-on to this application is a circuit model generator to create circuit models using a CSV logged data file of the main motorsport electronic manufacturers (MoTeC, PI, Bosch, Aim, Magneti Marelli...)

VI. Ideally this project developed in Matlab should be considered just as a dummy code prior to a real simulation software to be programmed in any language (C, C++, Python) with no dependencies to software packages.

If you are interested in working together with me enhancing this simulator on the areas listed above or any other, please do not hesitate to contact me: cesar@cesar-ap.com

4. Literature and Information Sources

I found great help on the following references in order to program the simulator. Most of the physic formulas used have been taken from different sources on internet , some of them:

- **OptimumG Advanced Vehicle Dynamics** seminar notes.
- **Scott Raymond**'s **Understanding Vehicle Performance** articles in **RaceTechMag**.
- **Vehicle Dynamics and Control (R. Rajamani)**.
- **Race Car Vehicle Dynamics (William F. Milliken and Douglas L. Milliken)**.
- **Lap Time Simulation (James Hakewill)**.
- **The Physics of Racing (Brian Beckman)**.

Special thanks to **Daniel Gratacós** and **Rob Arnott** for providing technical information of some of the vehicles they have worked with. Such information really helped to make a vehicle model used to develop the simulator.