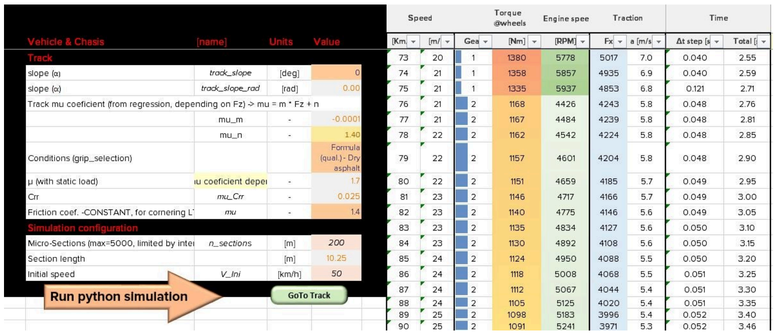


Vehicle & Chasis	[name]	Units	Value
Track			
slope (α)	track_slope	[deg]	0
slope (α)	track_slope_rad	[rad]	0.00
Track mu coeficient (from regression, depending on Fz) -> mu = m * Fz + n			
	mu_m	-	-0.0001
	mu_n	-	1.40
Conditions (grip_selection)			Formula (qual.) - Dry asphalt
μ (with static load)	ιu coeficient depeı	-	1.7
Crr	mu_Crr	-	0.025
Friction coef. -CONSTANT, for cornering L	mu	-	1.4
Simulation configuration			
Micro-Sections (max=5000, limited by inteı	n_sections	[m]	200
Section length		[m]	10.25
Initial speed	V_Ini	[km/h]	50

Speed			Torque @wheels	Engine spee	Traction		Time	
[Km.	[m/	Gea	[Nm]	[RPM]	Fx	a [m/s	Δt step [s	Total [
73	20	1	1380	5778	5017	7.0	0.040	2.55
74	21	1	1358	5857	4935	6.9	0.040	2.59
75	21	1	1335	5937	4853	6.8	0.121	2.71
76	21	2	1168	4426	4243	5.8	0.048	2.76
77	21	2	1167	4484	4239	5.8	0.048	2.81
78	22	2	1162	4542	4224	5.8	0.048	2.85
79	22	2	1157	4601	4204	5.8	0.048	2.90
80	22	2	1151	4659	4185	5.7	0.049	2.95
81	23	2	1146	4717	4166	5.7	0.049	3.00
82	23	2	1140	4775	4146	5.6	0.049	3.05
83	23	2	1135	4834	4127	5.6	0.050	3.10
84	23	2	1130	4892	4108	5.6	0.050	3.15
85	24	2	1124	4950	4088	5.5	0.050	3.20
86	24	2	1118	5008	4068	5.5	0.051	3.25
87	24	2	1112	5067	4044	5.4	0.051	3.30
88	24	2	1105	5125	4020	5.4	0.051	3.35
89	25	2	1098	5183	3996	5.4	0.052	3.40
90	25	2	1091	5241	3971	5.3	0.052	3.46

Laptime simulator: mass-point quasi-steady state

Victor Cortés Abad
🏁 Motorsport Engineer | 📈 Providing high-value solutions for teams and drivers | Trackside coaching,...

April 7, 2021

Along the following article, I will cover a brief tutorial on how to create a mass-point simulator, with quasi-steady state solver. Key points and formulation will be given in order to encourage the reader to reply my job and create his/her own tools!

Why mass-point and quasi-steady state simulator?

One of the most important transversal knowledge I learnt and developed since university is the skill of analysing resources vs usefulness relation before starting anything. This acknowledge is commonly studied with Pareto's law or 20/80 rule. The numbers may vary and the subjects of study may be disparate, but it always work. It is mandatory for me to find the optimum point between effort/resources and usefulness/results, before starting any project. In this particular case, as I learnt in Optimum-G's seminar, a mass-point simulator with quasi-steady state solver hit the mark in terms of usefulness vs complexity:

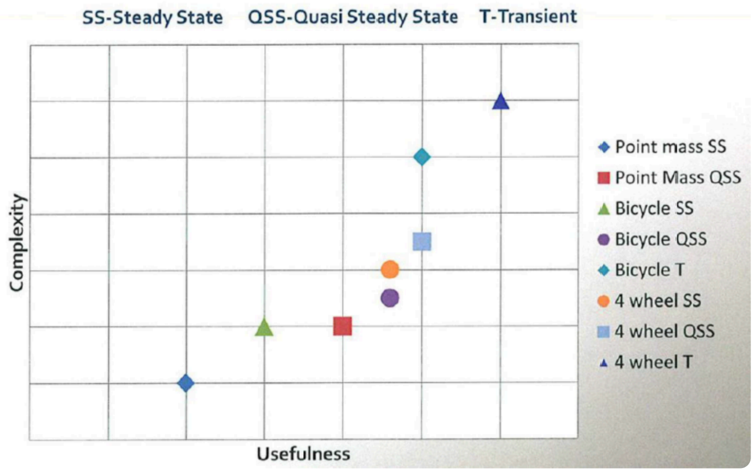


Figure 1: Types of simulators, complexity vs usefulness (Rouelle, 2018).

As reader may observe in figure 1, the chosen simulator holds one of the best usefulness/complexity ratio. This type of simulator, despite neglecting most of dynamic effects, returns satisfactory results in very short calculation times. This is the main reason that brings it to be the most common solution to simulate laptime in high competition environments, such as Formula 1, *(Catherall, 2017)*.

In conclusion, the chosen type of simulator seems to be the best option for my first simulator:

- Can be created in daily use programs such as Python and Excel.

- Returns satisfactory results with short calculation times.

- Useful for laptime analysis modifying important parameters such as vehicle mass, engine torque, track grip or basic aerodynamics.

Vehicle model

The mass-point model is the simplest model of a vehicle, it only need the following parameters:

· Total mass ‘m’: vehicle, fuel and driver.

· Basic aerodynamics: lift coefficient ‘CL’, drag coef. ‘CD’ and projected frontal area ‘A’.

· Basic engine parameters: power curve (torque vs RPM), gear ratios, rolling tyre diameter and intern loses.

· Braking capability: in terms of torque at wheel.

· Basic tyre model: friction coef. ‘µ’ in both longitudinal and transversal directions. In this particular case, as a first approach, it will depend only on tyre’s vertical load. Tyre model can be much more complex, considering tyre and track temperature, humidity, tyre wear and pressure, between others.

Note that a mass-point model has no yaw, pitch or roll inertias, nor suspension parameters. Therefore, this model will not have weight transfer nor transient states. Nevertheless, as simple as it is, it can be created with Excel or Python. In my particular simulator, I created the model on Excel and Python reads information from it.

Circuit model

Once the vehicle is modelled, it is time to model the track. This is done straightforward in three min steps, including hypothesis and simplifications:

1. Estimate grip level of the track, both friction and rolling resistance coefficients. In my case, for educational purpose, it is regarded as constant along the whole length of the circuit, as well as constant for any vertical load on tyres. In this case I took the following values:

· Friction coefficient = 1.4

· Rolling resistance coefficient = 0.025

2. Discretize driving line length into multiple sectors. Choosing sectors length need to be carefully decided. A higher rate (smaller sectors) will

lead to long calculation times, whereas too long sectors will lead to a lack of precision. A good starting point is 5m (*Hakewill, 2000).*

3. Determine curvature radius *"R"* of each sector, this is done by trigonometry calculations:

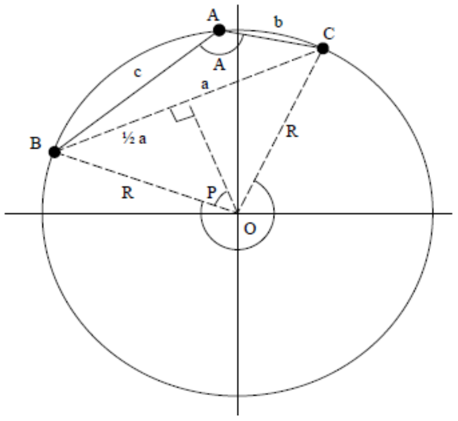


Figure 2: Curvature radius calculation scheme.

From three points in space, reader can calculate curvature radius as following:

$$R = \frac{a}{2 \cdot sin(180 - A)}$$

Reader must notice that coordinates and sectors are based on driving line and not on medium track's line. Nevertheless, given the precision level of this simulator, this fact might be non-determinant for the results.

Cornering maximum speed

Following step is to calculate cornering maximum speed *"Vm"*, which is proportional to curvature radius *"R"*. Given a point-mass describing a circular trajectory with a tangential speed *"v"*, the centripetal force *"Fy"* is defined as:

$$F_y = \frac{m \cdot v^2}{R}$$

Furthermore, the longitudinal force *"Fx"* that mass-point must produce to overcome air resistance (air density *"ρ"*, aerodynamic parameters *"A"* y *"Cd"*) is defined as:

$$F_x = \frac{1}{2} \cdot \rho \cdot A \cdot C_d \cdot v^2$$

Please note that rolling resistance is not included in *"Fx"* calculation due to educational purpose.

These two forces are regarded as orthogonal components of tyre's total grip force *"Ft"*. Moreover, given the simplified tyre model, total force is calculated as product of friction coefficient *"μ"* and normal force due to mass *"N"*. Please be aware that this hypothesis should be avoided when simulating higher freedom range models (such as bicycle and complete vehicle models), since they introduce slip angles.

In this particular case of a mass-point, friction ellipse is regarded a constant radius circle, since it trends to this geometric form for small slip angles:

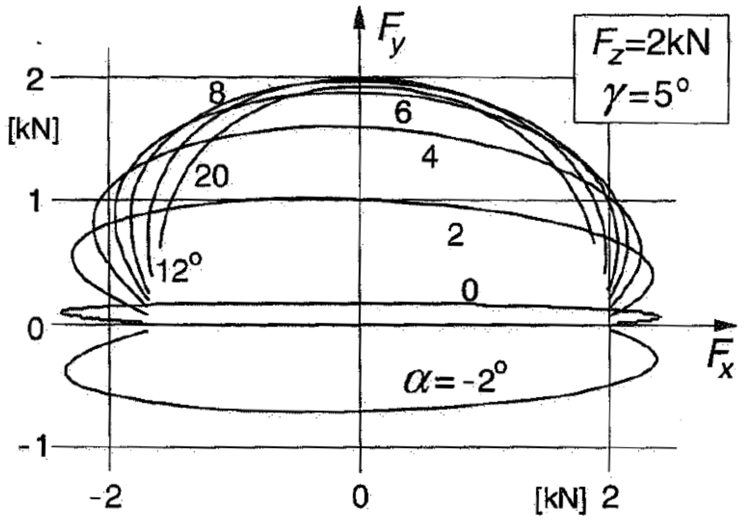


Figure 3: Grip ellipse for different slip angles, Pacejka (2006).

Furthermore, tyre's total friction force is calculated as:

$$F_t^2 = F_x^2 + F_y^2$$

$$(\mu \cdot N)^2 = \left(\frac{1}{2} \cdot \rho \cdot A \cdot C_D \cdot v^2\right)^2 + \left(\frac{m \cdot v^2}{R}\right)^2$$

User must be aware that aerodynamic vehicles generate significant downforce, thus normal force *"N"* will depend on downforce amount thus vehicle speed:

$$\left(\mu \cdot \left(m \cdot g + \frac{1}{2} \cdot \rho \cdot A \cdot C_L \cdot v^2\right)\right)^2 = \left(\frac{1}{2} \cdot \rho \cdot A \cdot C_D \cdot v^2\right)^2 + \left(\frac{m \cdot v^2}{R}\right)^2$$

Finally, attending to assumptions and simplifications, for a significant aerodynamic car, this equation allows obtaining maximum cornering speed at a steady state, named as *"Vm"* from now on.

Available acceleration & braking capability

Furthermore, from engine and transmission basic parameters, user can obtain a lookup table, which relates different speeds to their available straight-line acceleration (check VD-portfolio on my LinkedIn page for further details). Based on second's Newton law:

$$a_x = \frac{\sum F_x}{m}$$

Where the sum of forces includes tractive and resistive forces. User may take into account that tractive force is regarded as the minimum force between torque limited and grip limited forces:

$$F_{tractive} = min\left\{\frac{Engine\ torque\ on\ tyre}{Tyre\ rolling\ radius} \quad , \qquad \mu \cdot N\right\}$$

Usually, in most mass-point simulators, rolling radius of the tyre is regarded as constant. Nevertheless, in further simulations, it may be dependent on loads and tyre elasticity.

Finally, once the available straight-line acceleration lookup table is completed, user may create as well a straight-line braking capability table, in a parallel way but taking into account braking parameters.

Introduction to calculation process

Among a wide range of calculation methodologies, given the singularities of a quasi-steady mass-point simulator, the chosen for this article is a double iterative process along each sector.

Initially, from available acceleration lookup table, a first approach to exit speed for each sector is obtained. These speeds will be called precursor speeds.

Following, an inverse iterative process (from last to first sector) is conducted to identify braking points and tune speeds through braking capability lookup table. Those tuned speeds will be called definitive speeds from now on.

Calculation process: precursor speeds

First iterative process is executed in ascending sector order, from start to finish line:

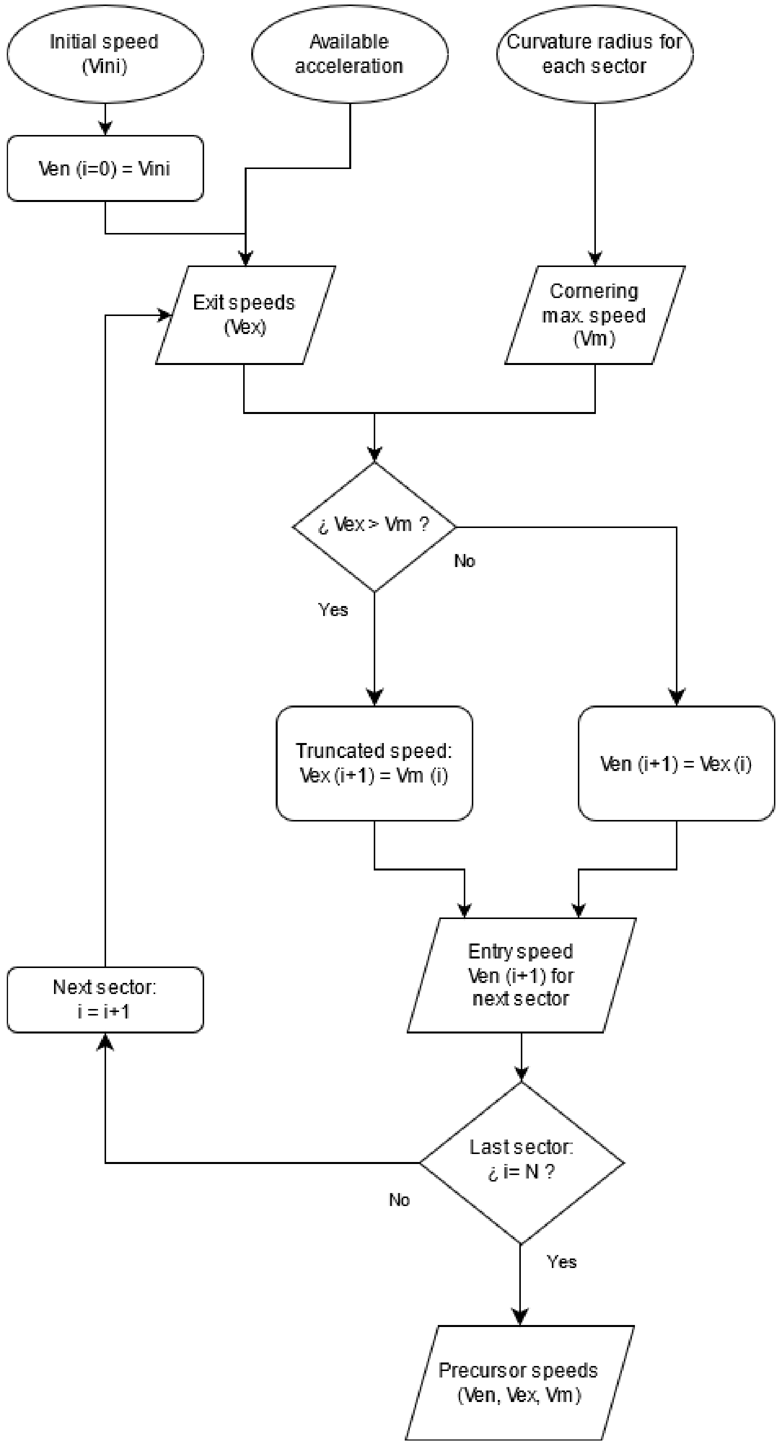

Initial speed (Vini)
Available acceleration
Curvature radius for each sector
Ven (i=0) = Vini
Exit speeds (Vex)
Cornering max. speed (Vm)
¿ Vex > Vm ?
Yes
No
Truncated speed: Vex (i+1) = Vm (i)
Ven (i+1) = Vex (i)
Entry speed Ven (i+1) for next sector
Next sector: i = i+1
Last sector: ¿ i= N ?
No
Yes
Precursor speeds (Ven, Vex, Vm)

Figure 4: Iteration process, flowchart for precursor speeds.

As seen in flowchart above, iteration process is started from three inputs:

· Initial speed *"Vini"*, which will be the entry speed *"Ven"* for first sector.

· Available acceleration obtained from lookup table described above.

· Curvature radius for each sector.

Initially, maximum cornering maximum speeds are calculated for every sector. These speeds will remain constant over iterations, since they are independent to entry and exit speeds for each sector.

Then, exit speed *"Vex"* for current sector is calculated, following the equation below (r.m.e.e.):

$$V_{ex} = V_{en} + a_x \cdot \Delta t$$

Please notice critical assumptions must be done in this equation for proper simplification of the model. On the one hand, acceleration along a single sector is regarded as constant, obtaining a unique acceleration value from lookup table, related to entry speed only.

On the other hand, the time interval required to overcome the sector *"Δt"* is calculated in a very simplistic way. It is regarded that vehicle travels at a constant speed, equal to entry speed. This assumption introduces a relatively small error if sector length is significantly short (being *"Δx"* the sector length), which is a good approach given the precision required from this simulator. Following those hypothesis, exit speed equation ends up as:

$$V_{ex} = V_{en} + a_x \cdot \frac{\Delta x}{V_{en}}$$

Once both speeds are calculated (exit and maximum cornering), they are compared against each other for every sector. If obtained *"Vex"* is greater than *"Vm"*, vehicle requires to brake along actual sector, consequently next sector's entry speed *"Ven(i+1)"* is truncated to current sector maximum cornering speed *"Vm(i)"*.

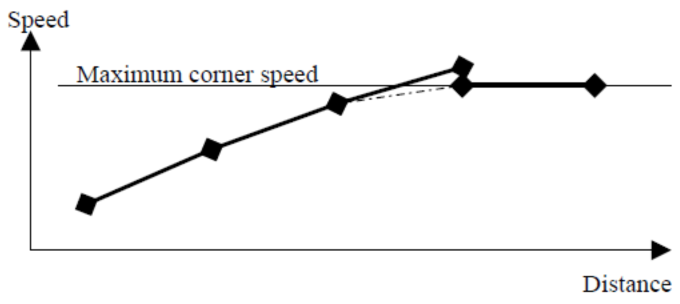


Figure 5: Graphic representation of first iterative process, precursor speeds (Hakewill, 2000).

Understandably, if speeds comparison above results in a greater *"Vm"* than *"Vex"*, next sector's entry speed *"Ven(i+1)"* is going to be calculated sector's exit speed *"Vex(i)"*.

Finally, once all sectors have been calculated, all precursor speeds, as well as maximum cornering speeds are defined, leading to next iteration process in inverse order.

Calculation process: definitive speeds

Second iterative process is executed in descending sector order, from finish to start line:

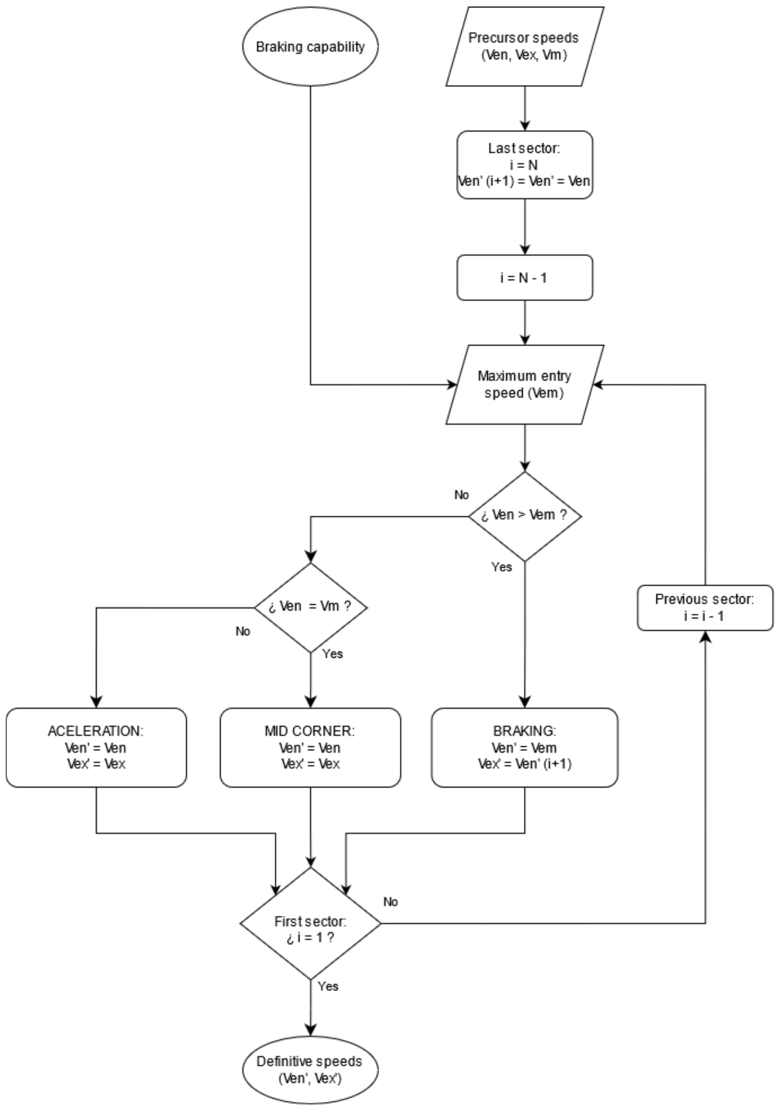

Braking capability
Precursor speeds (Ven, Vex, Vm)
Last sector: i = N Ven' (i+1) = Ven' = Ven
i = N - 1
Maximum entry speed (Vem)
¿ Ven > Vem ?
No
Yes
¿ Ven = Vm ?
No
Yes
Previous sector: i = i - 1
ACELERATION: Ven' = Ven Vex' = Vex
MID CORNER: Ven' = Ven Vex' = Vex
BRAKING: Ven' = Vem Vex' = Ven' (i+1)
First sector: ¿ i = 1 ?
No
Yes
Definitive speeds (Ven', Vex')

Figure 6: Iteration process, flowchart for definitive speeds.

Following process starts with precursor speeds, which define entry and exit speeds for each sector. Moreover, maximum cornering speed and braking capability (braking negative accelerations *"abx"*) lookup table are defined previously. Since real braking process is not instantaneous, it needs to be planned so vehicle reduces speed until maximum cornering speed when enters the first corner sector.

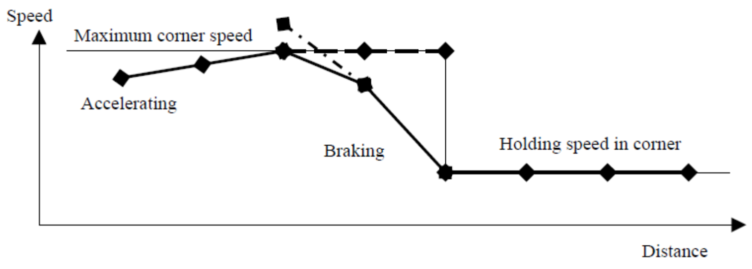


Figure 7: Graphic representation of second iterative process, definitive speeds (Hakewill, 2000).

Planning process is done by an inverse order iterative process (from last to first sector). From inputs described above, maximum entry speed *"Vem"* is calculated in order to ensure desired exit speed. Understandably, this desired exit speed for a given sector *(i)* is equal to definitive entry speed *(Ven')* assigned to next sector *(i+1)*. Nevertheless, there is an exception for first iteration, this is last sector *(i=N)*, thus there is no next sector, and stands as following:

$$V'^{\,N+1}_{en} = V'_{en} = V_{en}$$

Once this initial exception is calculated, *"Vem"* is normally calculated as:

$$V_{em} = V'^{\,(i+1)}_{en} - a_{bx} \cdot \frac{\Delta x}{V'^{\,(i+1)}_{en}}$$

Please notice *"Vem"* is obtained in a similar way to precursor exit speed, but this time from exit to sector entry. Furthermore, similar hypothesis are considered, despite elapsed time is calculated from desired exit speed, treated here as definitive entry speed for next sector *"Ve' (i+1)"*, which is regarded as constant along the sector.

Following, once maximum entry speed is defined to ensure required braking, it is compared against precursor entry speed (obtained in previous iterative process). In case *"Vem"* is greater, total or partial acceleration states are considered. In order to identify this aspect, an equality is required, between precursor entry speed *"Ve"* and maximum cornering speed *"Vm"*. On the one hand, if this equality is false, total acceleration state is considered and definitive speeds stand as following:

$$V'_{en} = V_{en}$$
$$V'_{ex} = V_{ex}$$

On the other hand, if *"Ve"* is different to *"Vm"*, cornering state is considered, and only a certain amount of acceleration is required, in order to keep maximum cornering speed *"Vm"*:

$$V'_{en} = V_{en}$$
$$V'_{ex} = V_{ex}$$

Last but not least, if previous inequality *(Vem > Ve)* is false, braking state is considered and definitive speeds are adjusted as following:

$$V'_{en} = V_{em}$$
$$V'_{ex} = V^{i+1}_{en}$$

Finally, next iteration is calculated, which corresponds to previous driving line sector. In the particular case of start line sector, the iterative process finishes, returning all entry and exit definitive speeds.

Future Lines

As end of document notes, please find identified several future lines of development and test:

- Test acceleration and braking straight-line lookup tables, comparing against straight-line acceleration and braking tests.

- Test simulator and analyse results with a logged lap.

- Generate driving line from GPS logged data. Issue: convert latitude and longitude degrees to UTM square coordinates.

- Implement a more accurate tyre model.

- Eau rouge particular case, for instance, where you cannot go under maximum cornering speed due to aerodynamic grip characteristics.

- Validate model and test mass, grip, aero and other parameters variations.

Bibliography

1. Catherall, M. (2017) *Canopy Simulations: The Structure and Use of Simulations in F1.*

2. Rouelle, C. (2018) *Optimum-G: Applied Vehicle Dynamics Seminar.*

3. Hakewill, J. (2000) Lap Time Simulation.

4. Pacejka, H. (2006) *Tyre and vehicle dynamics*. Oxford: Butterworth-Heinemann.

Victor Cortés Abad

🏁 Motorsport Engineer | 📈 Providing high-value solutions for teams and drivers | Trackside coaching, 📊 Data analysis & Eng. systems programmer | 🤓 Bots developer

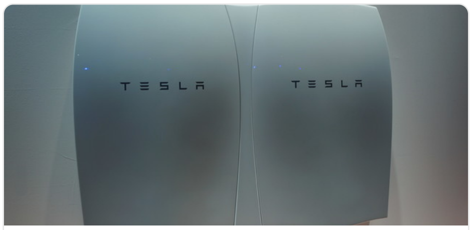
Follow

More articles for you

Crafting Tomorrow: Picking Simulation for Real Fruits at Tevel

Yair Rotem

38 · 8 comments · 2 reposts



TechNews (1/5): 10 Articles from the Tech World Today

Jamie Poole

Published on LinkedIn

Paradism: No Work, No Money

Sylvain Rochon