Quasi Steady State Lap Time Simulator

Jonathan Vogel (jovogel@g.clemson.edu)

Table of Contents

How to use a lap sim

Lap time simulations, at their very core, are tools used to predict the way that a vehicle performs on the track. However, it is critically important to understand that a lap simulation is just that – *only a simulation*. Thus, it cannot be treated as a representation of real-life performance.

This does not diminish the value of a lap sim – you just need to understand where its usefulness starts and ends. A well-developed simulation, while not always a perfect representation of absolute performance, is a powerful tool for evaluating relative performance between two concepts. This enables you to carry out sensitivity analyses and compare the different impacts that various design changes make on total performance. At the end of the day, a lap sim results will not be the be-all, end-all predictor of vehicle performance, but rather a tool to gather *insights* on vehicle performance.

In other words, you should not be using this lap sim to answer questions like:

- What is the exact camber setting I should run?
- How much downforce do we need to win Michigan?
- What is the ideal weight distribution?

Instead, you should be answering questions like:

- Does suspension gain more from prioritizing braking or cornering performance?
- Can we gain more performance from increasing downforce, or from improving aero efficiency?
- How does shifting weight distribution affect performance in different operating scenarios?

Then, once you have these insights, you can run tests, collect data, and refine your analysis before selecting a final design concept.

Lap Sim Contents

The lap sim is broken up into 19 sections overall, which I will briefly walk through. The first 5 pertain to entering in vehicle parameters that you want to evaluate and are the ones an end user will be interacting with the most.

Section 1: Input Tire Model

In this section, two tire models are loaded into the workspace - one to capture lateral performance, and one to capture longitudinal performance.

How to use this section

Currently, this model is pre-loaded to take in the tire model for our current tire (Hoosier R25B 18.0x7.5-10, 12 psi). However, if you want to evaluate a different tire, you can simply swap the file for the tire you are looking to analyze.

After you select a tire model, you can also play with the lateral and longitudinal grip scaling factors. These can be used to calibrate the simulation to the grip levels of the track, if you have lap time data to compare to. When in doubt though, leave them as is (sf_y = 0.47, sf_x =0.6).

Things you need to know

At the time of writing, this team employs two types of tire models; there is the fully empirical cubic spline interpolation fit (CSAPS), and the partially theoretical Pacejka Magic Formula MF5.2 model. The MF5.2 is a more robust model, but we only have a single model for our current tire. Meanwhile, the team has a full library of CSAPS models of various compounds and constructions. For this reason, there are two versions of the lap sim in this folder. "Lap_Sim.m" is written to use an MF5.2 tire, while "Lap_Sim_CSAPS.m" uses a CSAPS model.

One thing you must keep in mind, is that CSAPS tire models are very unreliable for normal loads of over 350 lbs. This means that you have to be *very careful* with simulating high-downforce cars on a CSAPS model, because you can get very inconsistent high-speed results.

Another piece of advice: If you are not explicitly comparing tires, do yourself a favor and don't use the CSAPS code. It takes much longer to evaluate and you'll be wasting your own time.

Section 2: Input Powertrain Model

In this section, you enter all the information pertaining to power delivery – torque curves, gearing, shift times, etc.

How to use this section

Refer to the Lap Sim Input Parameters table for instructions on the format required for each input. If you are not exploring drive-train specific concepts, you likely won't need to touch this section.

Things you need to know

Most of the parameters in this section are necessary. The only exception is "T_lock", which describes the locking torque of the rear differential. This is optional, and if you do not want to include differential effects, you can simply set it to 0.

Section 3: Vehicle Architecture

This section describes the primary architecture, dimensions and mass distribution characteristics of the vehicle.

How to use this section

All the variables in this section need to be defined. Refer to the Lap Sim Input Parameters table for instructions on the format required for each input. Most of your top-level vehicle concept explorations will happen here, such as exploring weight distribution, cg height, etc.

Things you need to know

The variable titled "LLTD" describes the lateral load transfer distribution of the suspension. This parameter is a primary characteristic used to define the understeer/oversteer balance of the car. If you do not know what to put here, or would not like to explore this variable, then set its value equal to "WDF" or the front weight distribution.

Section 4: Input Suspension Kinematics

This section contains most of the suspension parameters used in the lap sim.

How to use this section

Everything in this section is completely optional – so if you are not interested in exploring suspension interactions, simply set everything to zero. Within that, there are 3 primary sets of variables: Pitch/roll characteristics, camber alignment characteristics, and steering axis characteristics. These are mostly independent from each other, so you could change one section and set everything to zero and evaluate its effects in isolation.

Things you need to know

Be careful about exploring isolated suspension effects. Suspension kinematics are very coupled, and you may not be receiving the proper insight. In addition, make sure your sign conventions are correct; kingpin and caster should be positive, and any static camber should be negative.

Section 5: Input Aero Parameters

This section covers all the aerodynamic parameters and is the last section where you need to input vehicle characteristics.

How to use this section

This section is also optional, and you can set everything to zero and the lap sim will run just fine.

Things you need to know

For the lift and drag coefficient, you should actually input *the coefficient multiplied by the frontal area of the car*. I set up the sim this way to make comparing aero concepts more straightforward, and not having to account for an extra variable.

Section 6: Generate GGV Diagram

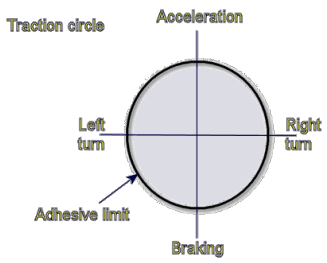
By this point, the vehicle has been fully defined, and as the user things are largely out of your hand. In this section, the vehicle gets compiled into a single model that will be used in the actual simulation.

How to use this section

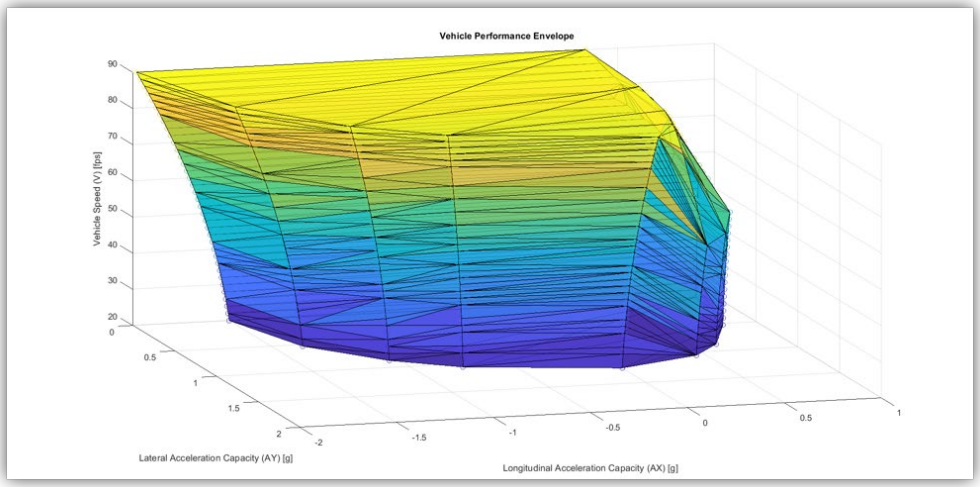
Unless you are re-writing the sim or expanding it to include new features, you really shouldn't be messing with this part of the code. I will make a different document to cover its technical details.

Things you need to know

This is arguably the most crucial piece of the sim. Using all of the parameters above, the code will create what is known as a "GGV diagram" for the vehicle. This is a compact way of describing the maximum acceleration, braking and cornering potential of the vehicle at any given velocity. You may be familiar with the concept of the traction circle, or friction ellipse:



A GGV diagram is basically the friction ellipse but expanded to include vehicle speed on the vertical axis. Pictured below is the GGV for our 2020 car:



A GGV diagram represents an idealized performance capability of the vehicle. If you are at any point along the 3-D surface of the envelope, it means you are operating the car at its absolute peak. Anything inside the surface and you are under-driving the car. When the virtual vehicle drives through the simulated track, it is moving exclusively along its GGV surface. This is, again, an idealized snapshot of vehicle performance capacity, and does not inherently translate to how the vehicle will behave on track. Peak performance is meaningless if the driver cannot consistently access that performance, which is why supplementary analysis should always be carried out to ensure the concept is stable and drive-able.

For Future Expansions

The beauty of the GGV diagram is in its modularity. Any number of vehicle models can be represented with a GGV, and the lap sim will run it just the same. So the GGV section of the code can be re-written into literally anything, so long as three things are generated:

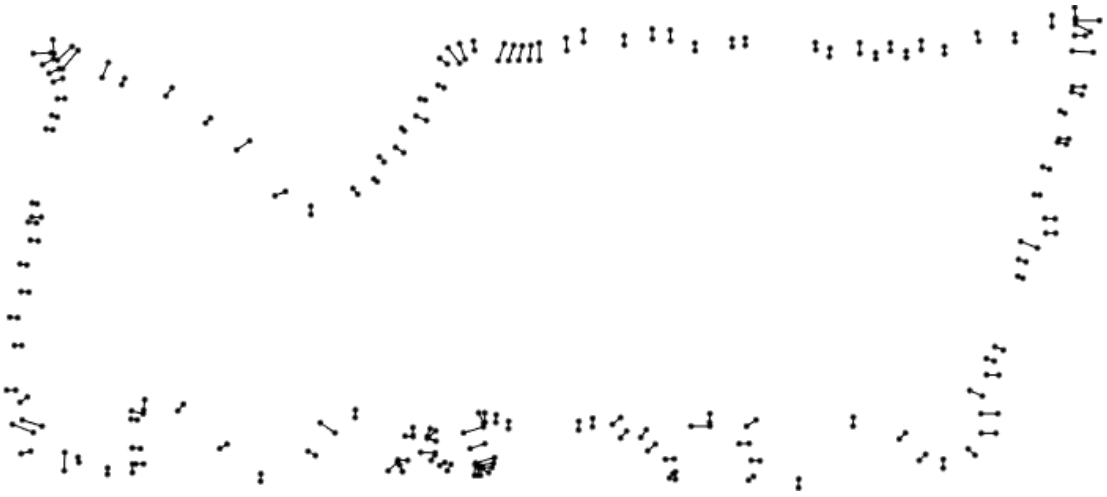
1) A function called “cornering”, that outputs lateral acceleration capacity in g’s as a function of an input vehicle velocity in ft/s
2) A function called “accel”, that outputs forward acceleration capacity in g’s as a function of an input vehicle velocity in ft/s
3) A function called “deccel”, that outputs braking acceleration capacity in g’s as a function of an input vehicle velocity in ft/s

So long as these three functions are generated as per the specifications above, and exist inside the workspace, the lap sim will run reliably – regardless of how they were generated.

Section 7: Load Endurance Track Coordinates

Things you need to know

The lap sim defines any track as a series of "gates" that the car needs to pass through. In order to define a track, you need a series of coordinates describing the "cone" positions that define each gate. By defining the set of "inside" and "outside" cones, the track boundary is also defined. See example of the 2019 Michigan endurance track below. The coordinates are loaded in from an excel spreadsheet.



How to use this section

If you are happy with running 2019 Michigan, there is no need to touch anything. However, if you want to use a new track, that is also possible. Simply create an excel spreadsheet containing the x-y coordinates of all the cone locations (in feet) in the following order:

- Column 1: Outside cones x
- Column 2: Outside cones y
- Column 3: Inside cones x
- Column 4: Inside cones y

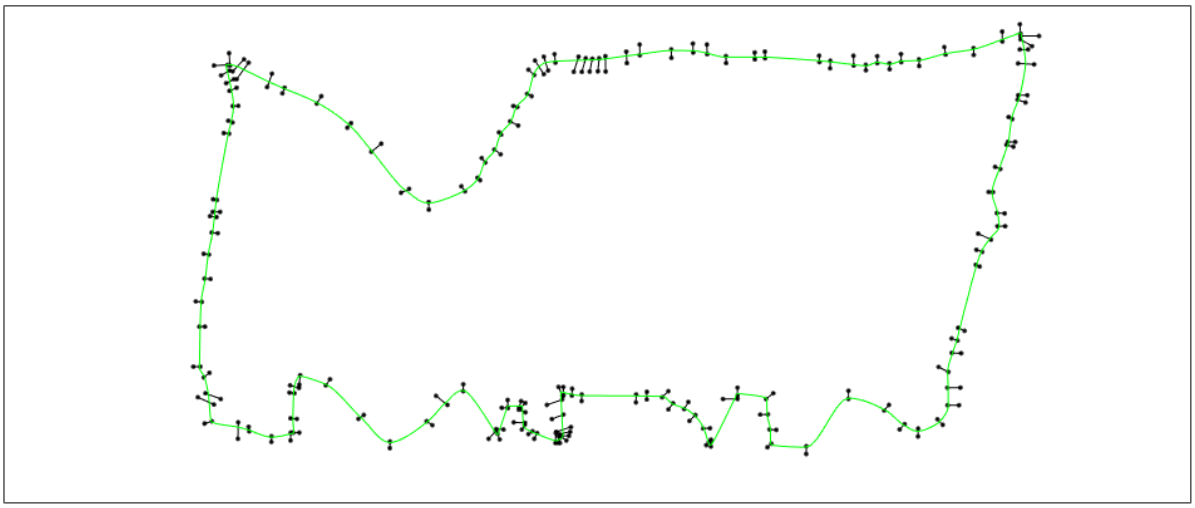
MATLAB will take it from there. Keep in mind that the more gates there are, the better the track definition will be. So it is advisable to "add" gates to the existing track map you may be referencing.

Section 8: Load Endurance Racing Line

The files folder already contains a file that describes an optimized trajectory through said endurance course. This section simply loads in that line.

Things you need to know:

Vehicle trajectory is defined relative to the gates. The racing line is defined as a series of values between zero and one. Zero means that for that given gate, the car drives right to the very inner boundary, and one means that the car is at the very outer edge. Stringing together a set of these values, along with the previously defined gate positions, is enough information to parameterize a vehicle path. See example below:



If you are loading in a brand-new track for the first time, you will need to generate a racing line to reference. Don't worry about it being optimal, that's what section 9 is for.

Section 9: Optimize Endurance Racing Line

This section will actually optimize the vehicle line taken through the gates, with the objective of minimizing the lap time.

How to use this section:

This section is optional and is normally commented out. If you want to use the section, simply highlight the whole section and un-comment it. You shouldn't ever really have to do this, except for a few scenarios:

- You have entered in coordinates for a new track, and need to find a fastest racing line for that track
- You are evaluating a vehicle concept that is such a departure from the previous, that it may be worth exploring new lines on the track (for example, switching to a single cylinder)

If you do decide to use the section, make sure to save your results so that next time you can just load it in section 8.

Things you need to know:

This part of the code is sooooooooo slow, and will take a very long time, so don't optimize the racing line unless you absolutely need to. Otherwise, best to just skip this section.

Section 10: Generate Final Endurance Trajectory

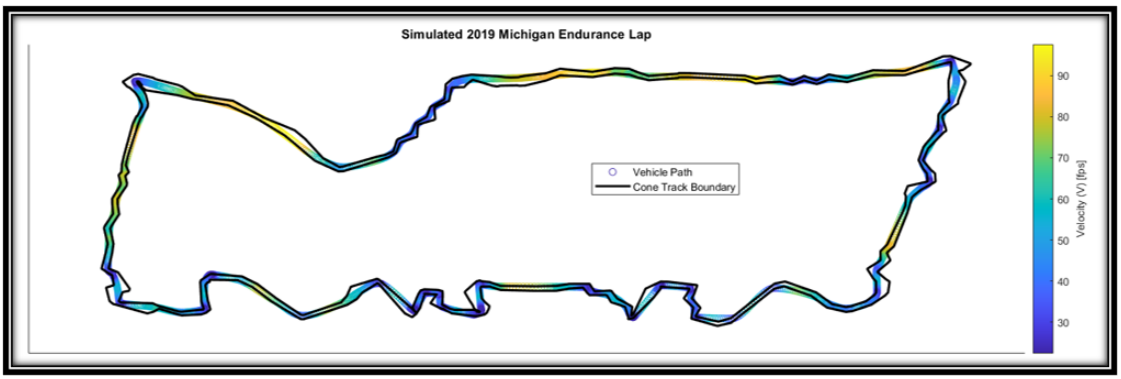
This is just a continuation of the previous sections. There's nothing you need to change or adjust here.

Section 11: Simulate Endurance Lap

Finally, we get a lap simulation!

How to use this section:

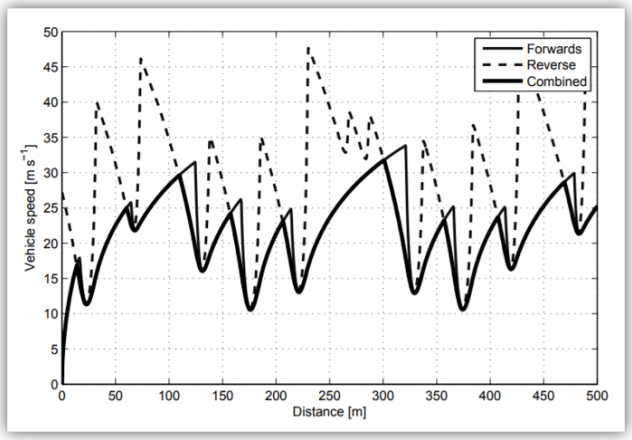
There is nothing you need to change or adjust here – just let the thing rip. That being said, the code outputs a lot of data that is useful to reference for potential design insights. For a full list, see the Lap Sim Output Parameters List. Also, it will make a nice pretty picture for you:



Things you need to know:

This lap sim is evaluated in quasi steady state. In other words, it assumes that the vehicle is always in a state of equilibrium, and as it traverses the track it is transitioning seamlessly from one equilibrium state to the next. This is, once again, an idealized representation of vehicle performance. In reality, the vehicle will exhibit a *transient* response to driver inputs before eventually reaching a steady state equilibrium. In its current state, the lap sim can not account for these transient effects.

The way the actual lap is generated is using something known as the “forward-reverse method”. It is a really fast way of evaluating a lap by avoiding logic trees. In essence, you first run the lap forwards, assuming that you have infinite braking capacity so you don’t need to worry about braking points, you just accelerate right up until the point you can’t anymore. Then you do the same thing in reverse, but braking aka “accelerating backwards”. What you get are two velocity traces, and if you overlay them on each other, their intersections define the braking point. Combining the two and extracting the combined minimum of both gives you your velocity across the full track. See the picture below.



Section 12-16: Autocross Simulation

This part of the code is identical to sections 7-12 in its functionality, so I won't cover it extensively here. Everything I said above applies here.

Things you need to know:

The only key difference is that in an autocross lap, the vehicle is starting from a standstill, while an endurance lap has a flying start. For that reason, the lap sim codes are slightly different. This is why, if you look in the files list, you'll see two of each code, one for each event. If you make a change to one code, just remember to make the same change to its twin as well.

Section 17: Calculate Dynamic Event Points

This section very quickly simulates an accel and skid-pad event, before adding up all points and predicting the dynamic event points scoring capacity of that vehicle.

How to use this section:

There is nothing you need to change or code here. The outputs include a total score, as well as a per-event breakdown. I encourage you to consider not just the total points haul of a vehicle concept, but to compare individual events to see where that performance gain is.

Things you should know:

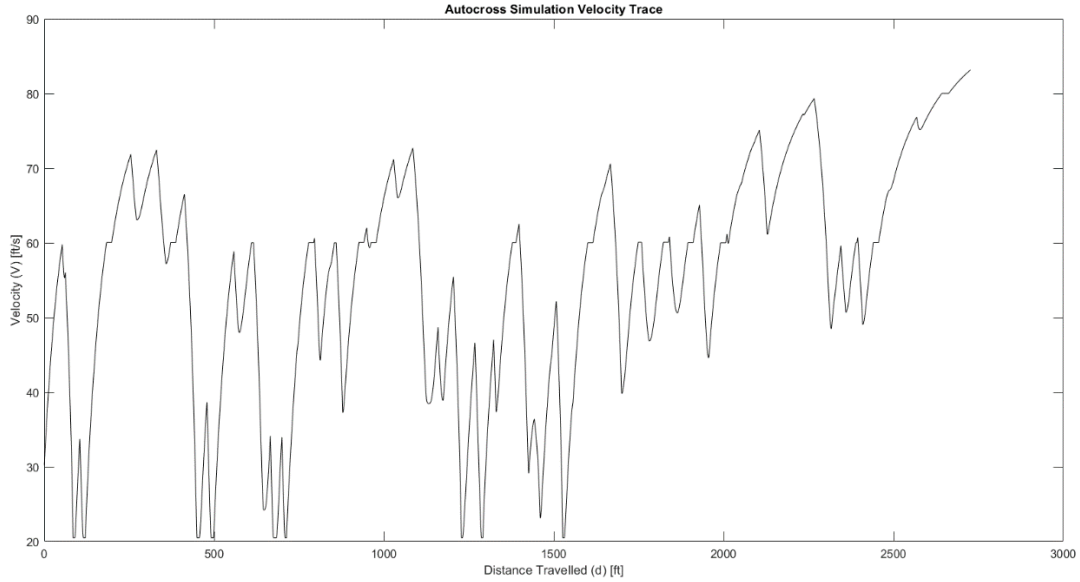
The points scored is defined by FSAE competition rules. These can be found in the latest revision of the rulebook. If you load in a track from a different event, you'll need to change the reference times used to determined points scored as well.

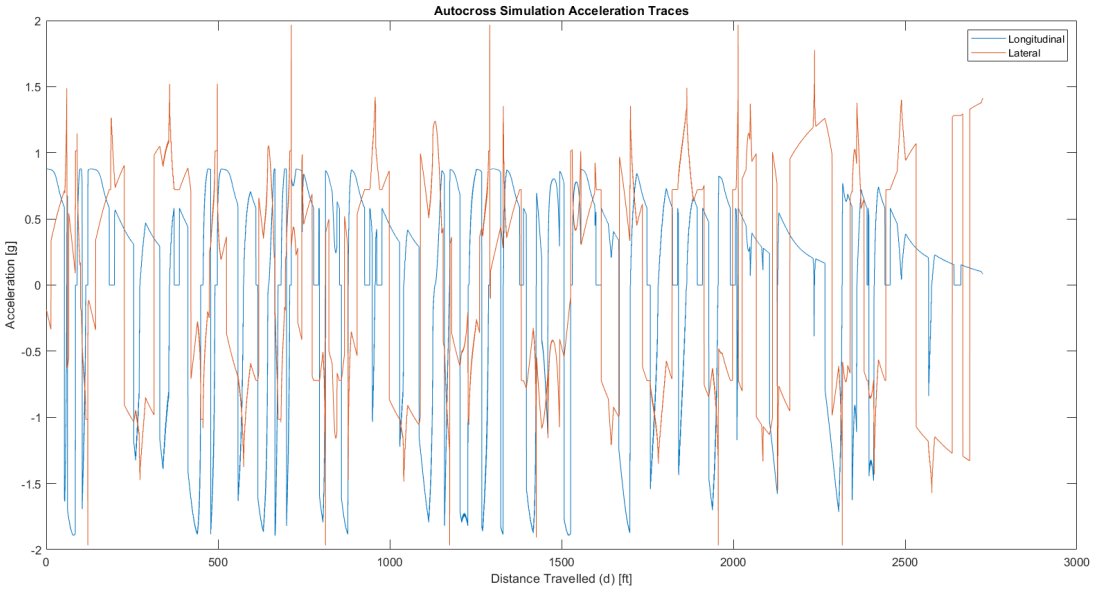
Section 18: Generate Load Cases

This section takes the maximum braking, cornering, and acceleration events from the endurance sim and uses them to generate peak load cases for the suspension and chassis. The load cases are presented as applied at the tire contact patch. For more information, see the Lap Sim Output Parameters List.

Section 19: Plot Results

This section just makes a few plots from the lap sim output data to show a few visualizations of the results, some examples on the next page:


Autocross Simulation Velocity Trace
Velocity (V) [ft/s]
Distance Travelled (d) [ft]
90
80
70
60
50
40
30
20
0
500
1000
1500
2000
2500
3000


Autocross Simulation Acceleration Traces
Longitudinal
Lateral
Acceleration [g]
Distance Travelled (d) [ft]
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
500
1000
1500
2000
2500
3000

Lap Simulation Parameters Index

This section is a reference for details regarding the inputs and outputs and general contents of the lap simulation.

List of codes contained

Lap Sim Files Index			
Category	**Function Name**	**Type**	**Purpose**
Lap Simulation	Lap_Sim.m	Program	Primary lap sim code
	lap_time.m	Program	Calculate endurance lap time for a specified vehicle and trajectory
	lap_time_sprint.m	Program	Calculate autocross lap time for a specified vehicle and trajectory
	lap_information.m	Program	Generate endurance lap data for a vehicle and trajectory
	lap_information_sprint.m	Program	Generate autocross lap data for a vehicle and trajectory
Tire Modelling	MF52_Fy_fcn.m	Program	MF5.2 Magic Formula tire model evaluator
	A1654run21_MF52_Fy_12.mat	Data	Pre-stored MF5.2 Variables for the R25B 18.0x7.5-10 tire
	A1654run21_MF52_Fy_GV12.mat	Data	Pre-stored MF5.2 Variables for the R25B 18.0x7.5-10 tire
	Hoosier_R25B_18.0x7.5-10_FX_12pis.mat	Data	Cubic spline model for the R25B 18.0x7.5-10 tire
	myvars.mat	Data	Pre-stored MF5.2 Variables for the R25B 18.0x7.5-10 tire
Powertrain	Powertrainlapsim.m	Program	Evaluates powertrain outputs for a given vehicle speed
Track Data	Endurance_Coordinates_1.xlsx	Data	Endurance track map cone coordinates
	Autocross_Coordinates_2.xlsx	Data	Autocross track map cone coordinates
Vehicle Trajectory	endurance_racing_line.mat	Data	Optimized endurance track racing line
	autocross_racing_line.mat	Data	Optimized autocross track racing line
	arclength.m	Program	Calculate distance travelled for a given vehicle racing line
	curvature.m	Program	Calculate path curvature for a given vehicle racing line
	circumcenter.m	Program	Calculate centerpoint of arbitrary arc trajectory
	track_curvature.m	Program	Constraint function to ensure vehicle does not exceed minimum turn radius in endurance
	track_curvature_sprint.m	Program	Constraint function to ensure vehicle does not exceed minimum turn radius in autocross

Input Parameters

Lap Simulation Input Parameters List

Section	Variable Name	Description	Type	Status	Range	Units	Notes
Section 1: Tire Model	tire_radius	Tire Radius	Scalar	Required	(0,∞)	ft	
	sf_x	Longitudinal tire grip correction factor	Scalar	Required	(0,1]	-	Use to calibrate braking/accel performance
	sf_y	Lateral tire grip correction factor	Scalar	Required	(0,1]	-	Use to calibrate cornering performance
Section 2: Powertrain Package	engineSpeed	Torque Curve RPM Vector	Vector	Required	(0:∞)	Rev/min	
	engineTq	Torque Curve Torque Vector	Vector	Required	(0:∞)	N-m	Dimensions should match engineSpeed
	primaryReduction	Primary engine speed reduction	Scalar	Required	(0,∞)	[ω_in:ω_out]	Probably will never have to change
	gear	Transmission gear ratios	Vector	Required	(0:∞)	[ω_in:ω_out]	Length should equal # of gears
	finalDrive	Final Drive Ratio	Scalar	Required	(0,∞)	[ω_in:ω_out]	Determined from engine and diff sprocket size
	shift_time	Shift Time	Scalar	Required	(0,∞)	sec	
	T_lock	Differential Locking Torque	Scalar	Optional	[0,100]	%	0 is a fully open/no diff, 100 is fully locked
Section 3: Vehicle Architecture	LLTD	Lateral Load Transfer Distribution to Front Axle	Scalar	Required	(0,100)	%	If you want to leave out LLTD effects, just make LLTD equal to WDF
	W	Vehicle+Driver Weight	Scalar	Required	(0,∞)	lbf	
	WDF	Front Weight Distribution	Scalar	Required	(0,100)	%	
	cg	Center of Gravity Height	Scalar	Required	(0,∞)	ft	
	l	Wheelbase Length	Scalar	Required	(0,∞)	ft	
	twf	Front Track Width	Scalar	Required	(0,∞)	ft	
	twr	Rear Track Width	Scalar	Required	(0,∞)	ft	
Section 4: Suspension Kinematics	rg_f	Front Roll Gradient	Scalar	Optional	(0,∞)	deg/g	If you don't want to include, put "0"
	rg_r	Rear Roll Gradient	Scalar	Optional	(0,∞)	deg/g	
	pg	Pitch Gradient	Scalar	Optional	(0,∞)	deg/g	
	WRF	Front Single Corner Ride Rate	Scalar	Required	(0,∞)	lbs/in	You have to put something, but if you set all gradients and kinematics to zero then it won't do anything
	WRR	Rear Single Corner Ride Rate	Scalar	Required	(0,∞)	lbs/in	
	IA_staticf	Front Static Camber Angle	Scalar	Optional	(-∞,∞)	deg	Negative camber means the tops of the tires are pointed in
	IA_staticr	Rear Static Camber Angle	Scalar	Optional	(-∞,∞)	deg	
	IA_compensationf	Front Camber Compensation	Scalar	Optional	(-∞,∞)	%	When positive camber is generated in roll, camber compensation is how much of that you gain back
	IA_compensationr	Rear Camber Compensation	Scalar	Optional	(-∞,∞)	%	
	casterf	Front Caster Angle	Scalar	Optional	(-∞,∞)	deg	Positive caster means the top of the steering axis is pointed backward
	casterr	Rear Caster Angle	Scalar	Optional	(-∞,∞)	deg	
	KPIf	Front Kingpin Axis Angle	Scalar	Optional	(-∞,∞)	deg	Positive KPI means the top of the steering axis is pointed inward
	KPIr	Rear Kingpin Axis Angle	Scalar	Optional	(-∞,∞)	deg	
Section 5: Aerodynamics Package	Cl	Coefficient of Lift * Frontal Area	Scalar	Optional	[0,∞)	ft^2	
	Cd	Coefficient of Drag * Frontal Area	Scalar	Optional	[0,∞)	ft^2	
	CoP	Front Downforce Distribution	Scalar	Optional	[0,100]	%	To leave CoP effects out, make CoP equal to WDF

Output Parameters

Lap Simulation Output Parameters List				
Variable Name	**Description**	**Type**	**Units**	**Notes**
laptime	Endurance lap time	Scalar	s	
time_elapsed	Endurance lap sim time trace	Vector	s	Use for plotting other variables below
distance	Endurance lap sim distance trace	Vector	ft	
velocity	Endurance sim velocity trace	Vector	ft/s	
acceleration	Endurance sim longitudinal acceleration trace	Vector	g	
lateral_accel	Endurance sim lateral acceleration trace	Vector	g	
gear_counter	Endurance sim gear selection trace	Vector	-	
path_length	Total distance travelled in endurance lap	Scalar	ft	
weights	Lap Weighting Scale	Vector	-	relative proportion of lap time spent accelerating, braking, and cornering, respectively
vehicle_path_EN	Endurance trajectory coordinates	Matrix	ft	[x coordinate;y coordinate]
laptime_ax	Autocross lap time	Scalar	s	
time_elapsed_ax	Autocross lap sim time trace	Vector	s	
distance_ax	Autocross lap sim distance trace	Vector	ft	
velocity_ax	Autocross sim velocity trace	Vector	ft/s	
acceleration_ax	Autocross sim longitudinal acceleration trace	Vector	g	
lateral_accel_ax	Autocross sim lateral acceleration trace	Vector	g	
gear_counter_ax	Autocross sim gear selection trace	Vector	-	
path_length_ax	Total distance travelled in autocross lap	Scalar	ft	
weights_ax	Autocross Lap Weighting Scale	Vector	-	relative proportion of lap time spent accelerating, braking, and cornering, respectively
vehicle_path_AX	Autocross trajectory coordinates	Matrix	ft	[x coordinate;y coordinate]
Endurance_Score	Endurance points prediction	Scalar	-	
Autocross_Score	Autocross points prediction	Scalar	-	
Skidpad_Score	Skidpad points prediction	Scalar	-	
Accel_Score	Accel points prediction	Scalar	-	
Total_Points	Total points prediction	Scalar	-	
accel_time	Acceleration event time	Scalar	s	
skidpad_time	Skidpad event time	Scalar	s	
t_accel	Accel event time trace	Vector	s	
v_f	Accel event velocity trace	Vector	s	
ax_f	Accel event acceleration trace	Vector	s	
frontF	worst case endurance load cases (as measured at the tire)	Matrix	lbf	Rows: FX, FY, FZ
rearF		Matrix	lbf	Columns: Accel, Braking, Cornering

Suggestions for Future Development

There is still plenty of work that can be used to improve this lap simulation. I'll include a few ideas here

Ackermann Steering

Ackermann steering effects are not yet accounted for in this sim. This is a relatively easy one to implement, and will enable another steering geometry parameter to optimize.

Aero Mapping

If aero maps existed that quantified the effects of pitch, heave and roll on the drag, downforce, and center of pressure migration, that could potentially be included in the GGV diagram generation to evaluate suspension stiffness targets.

Automatic Track Generation

My current process of generating track coordinates involves downloading a picture of the track map, opening it in Solidworks, sketching points onto the cones, and then painstakingly transfer those coordinates onto excel. There are definitely smarter and faster ways to go about this

Driver Limits

In the GGV simulation, the vehicle is instantly transitioning back and forth between states of acceleration, braking and cornering. A limit could be imposed on how quickly it is allowed to transition, to try and incorporate a more realistic driving representation.

Combined Load Cases

Despite the load cases currently provided, the true worst-case loading scenario for a suspension/frame member is a combination of both acceleration and cornering. One of the outputs automatically generated is a graph of accelerations over time, with some extra coding this could pretty easily be translated into a graph of suspension loads over time. This can provide more insight into where our worst load cases happen, and with what frequency, thus helping refine fatigue endurance targets.

And more

There are countless ways this work can be expanded upon! I am always happy to discuss ideas and potential improvements so don't hesitate to reach out or to just jump in and get your hands dirty yourself.