RoseLap v5: Lap Time and Point Simulation

Thaddeus Hughes (hughes.thad@gmail.com)
Evë Maquelin (evelyne@maquelin.com)

July 7, 2018

Abstract

RoseGPE is a student design competition team at Rose-Hulman that competes in the Formula SAE competitions, typically at the Lincoln and Michigan events. RoseGPE is driven by students learning and gaining work experience, and does this by aiming to score high at competition. To do this, the team builds a new prototype vehicle every year.

The competition is historically composed of 3/8 'static' events (cost judging, design judging, business case judging) and 5/8 'dynamic' events (acceleration, skidpad, autocross, endurance, and fuel economy)- meaning there are many facets to earning points and doing well.

Lap Time Simulation can be utilized to determine the configuration of vehicle which can produce the most points at competition. Determining points outcome from the acceleration, skipad, autocross, and endurance events is largely straightforward.

1 Problem Introduction

Lap Time Simulation is a numerical process. The general premise is to apply simplistic physics models to a car traveling over a small, finite stretch of track with given curvature to determine the next vehicle state. By stitching these segments together, with a strategy of when to brake, shift, or otherwise drive the vehicle, the operation of a racecar under idealized conditions can be simulated with useful accuracy.

These simulation results can be useful, potentially, to assist a driver in training for a track. They can also be useful and motivating for an engineer to understand how a vehicle drives around the track in detail, monitoring grip. The real value, however, lies in the ability to vary parameters and see how they impact lap times (and fuel consumption).

These lap times can be used with point formulas and other data to determine the overall point limit of a car at a particular competition.

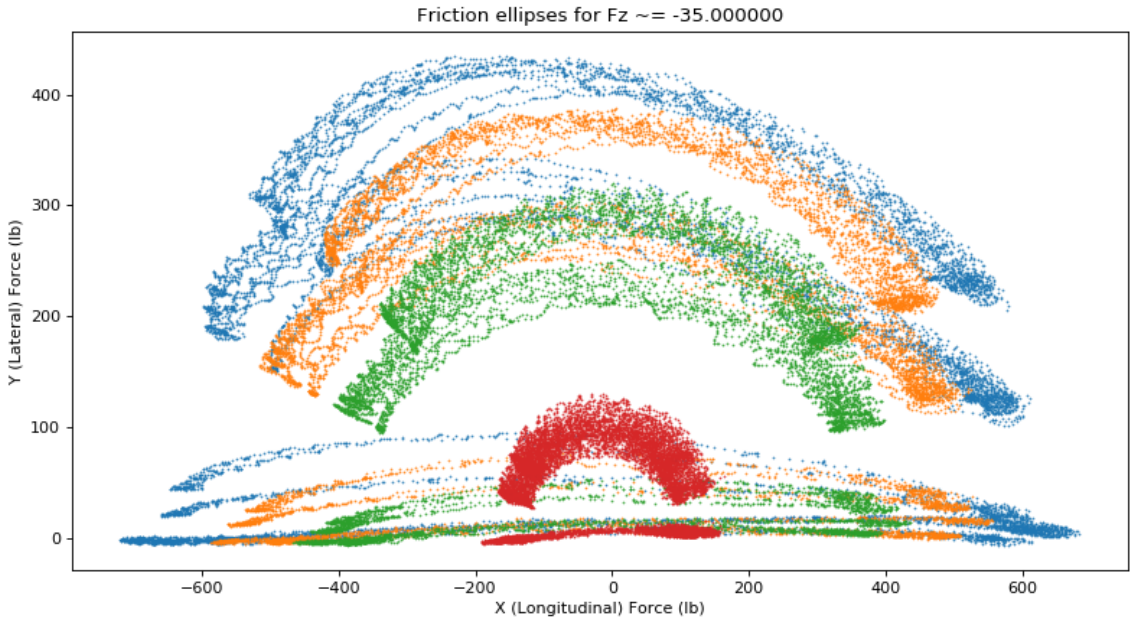
2 Pre-Processing and Input Data

2.1 Tire Data Fitting

As discussed in the 'Tires' section, tire models used in RoseLap v5 are rather simplistic, but are more sophisticated than the constant COF presumed in v4 and prior. To determine the four parameters, some fitting must be done.

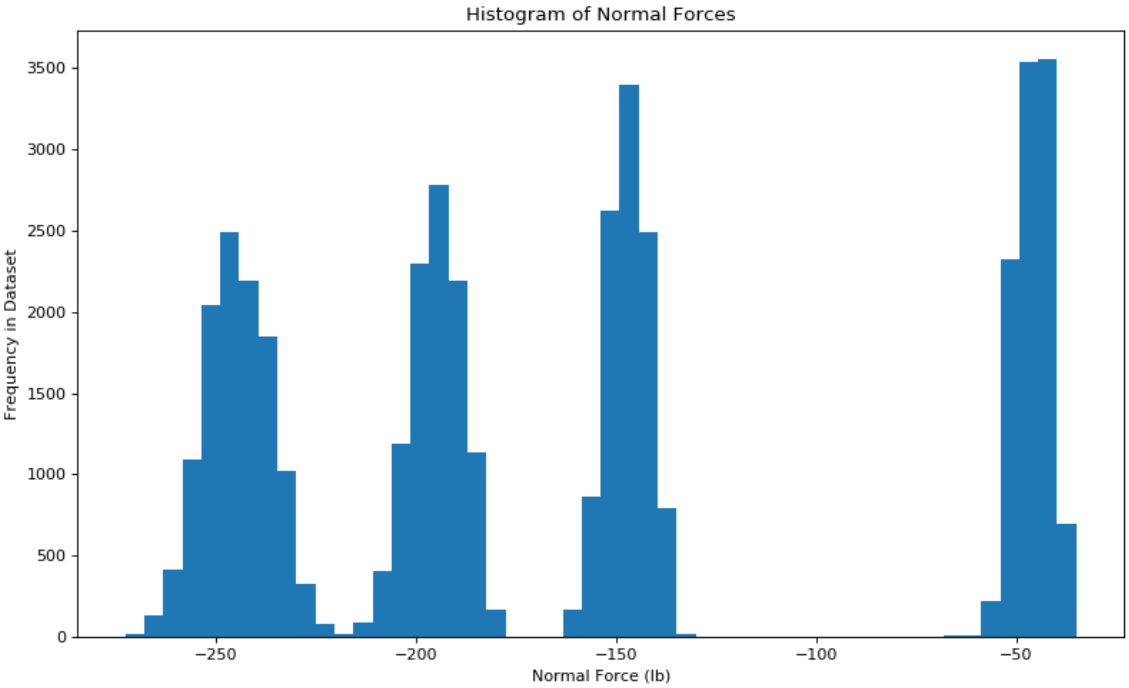
Tire data is obtained through physical testing. We use the data purchased from the Milliken Tire Test Consortium (TTC). This contains drive-brake and combined loading, with sweeps over various conditions.

With the python tool provided in `tire_utility`, we examine the relationship between lateral and longitudinal force as it relates to normal force.

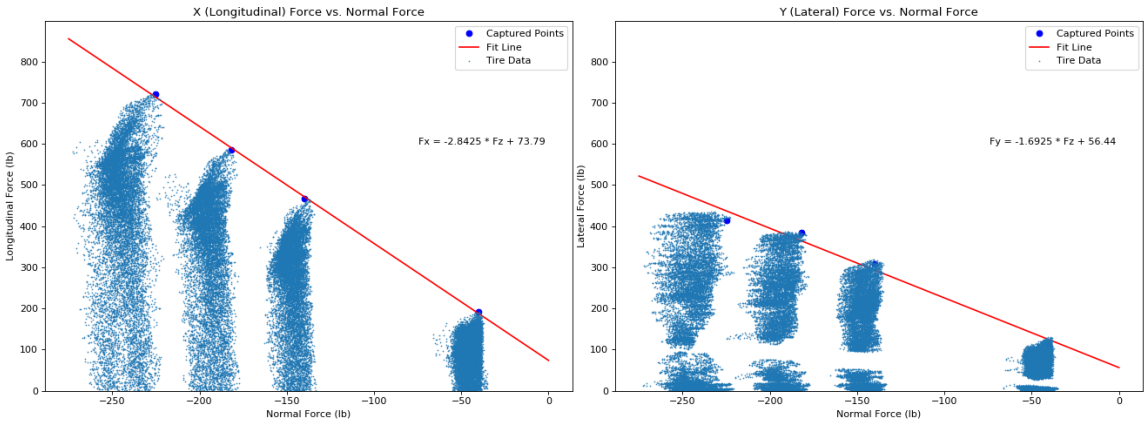


The multiple colors indicate different normal forces. While imperfect, it is easy to see how fitting an ellipse to this would provide a reasonable model of tire behavior.

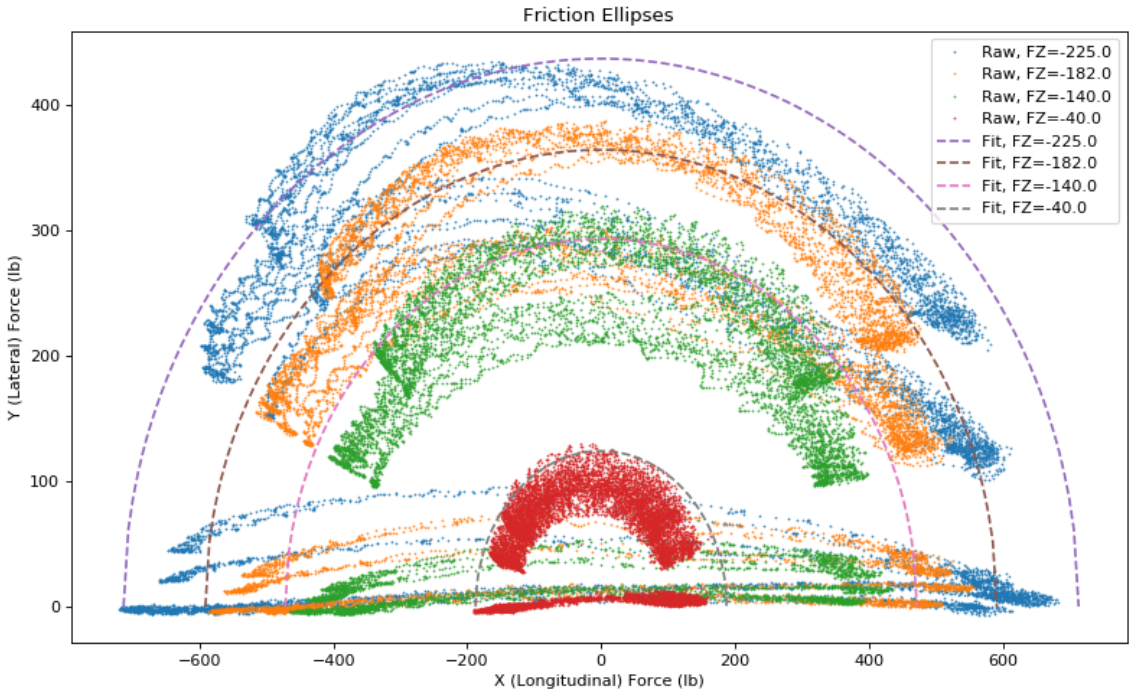
The first step is to determine the bins to which tire data vertical loads fall into. This is accomplished with a histogram.



Then, the data is analyzed in the Fz-Fx space and the Fz-Fy space. Outliers (outside of a 2-sigma confidence interval) are rejected. From here, the maximum grip datapoints from each bin are selected, and a linear regression is ran to fit these points.



This provides us with the four parameters for the tire: the offset and slope, in both x- and y-directions. This model is then overlaid onto the Fx-Fy data from before for comparison.



User adjustments can be made by changing confidence intervals, bins, or even overriding the regression results and using the charts to produce a linear fit.

2.2 Vehicle Data Entry

Vehicle data entry is accomplished with .YAML files. See the Example Vehicle Definition File in the Appendix.

2.3 Tracks

In RoseLap, tracks are sets of 'segments' analagous to an FEA mesh. A segment is parameterized as having a distance (Δx) and curvature (k).

Tracks may be defined in three ways:

- an SVG file (preferably using a spiro path)
- a trackwalker .LOG file with a header file
- a DXF file consisting of lines and circular arcs (NOT supported with $I_{yaw} \neq 0$)

Option 3 is becoming less supported but may come back into favor with some updates. This is because the introduction of yaw moment effects requires a track with continuous curvature; otherwise $\frac{dk}{dx}$ is, when numerically evaluated, effectively infinite, requiring infinite tire grip to initiate a corner.

SVG file import is accomplished with `svgpathtools`. This begins by loading all the SVG paths and computing the raw x, y, and curvature of each point at the specified mesh density. After this, a univariate spline of 2nd degree is fitted with a smoothing factor of 0.005 to the curvature data. This ensures that curvature is continuous and differentiable. The degree and smoothing parameter could be modified.

.LOG file import begins by reading a 1-line JSON file at the beginning with the following parameters:

- `cutoff_start`: how many entries to cut off from the start of a track
- `cutoff_end`: the entry number to stop at
- `d_nom`: the diameter of the wheels on the trackwalker
- `d_scale`: the ratio of diameters between two wheels
- `D`: the width of the trackwalker
- `maxcurv`: a hard limit on how much curvature can be allowed
- `smoothing`: the amount of smoothing to apply to the spline fit
- `savgol_amount`: the number of entries the savgol filter should average
- `savgol_dof`: the number of degrees of freedom to use with the savgol filter

An example of this is as follows:

```
{"cutoff_start": 30,"cutoff_end": 1280,"d_nom": 5.74,"d_scale": 1.0185, "D": 23.0, "
    maxcurv": 0.15, "smoothing":0.03, "savgol_amt":351, "savgol_dof":2}
0,0
15,4
26,11
36,22
50,36
...
```

The trackwalker log is then processed to determine distance, curvature, and cartesian coordinates for all points on the track.

$$\Delta l_i = 2(l_{1,i} - l_{1,i-1} - l_{2,i} + l_{2,i-1}) \tag{1}$$

$$d_i = \frac{l_{1,i} - l_{1,i-1} + l_{2,i} - l_{2,i-1}}{2} \tag{2}$$

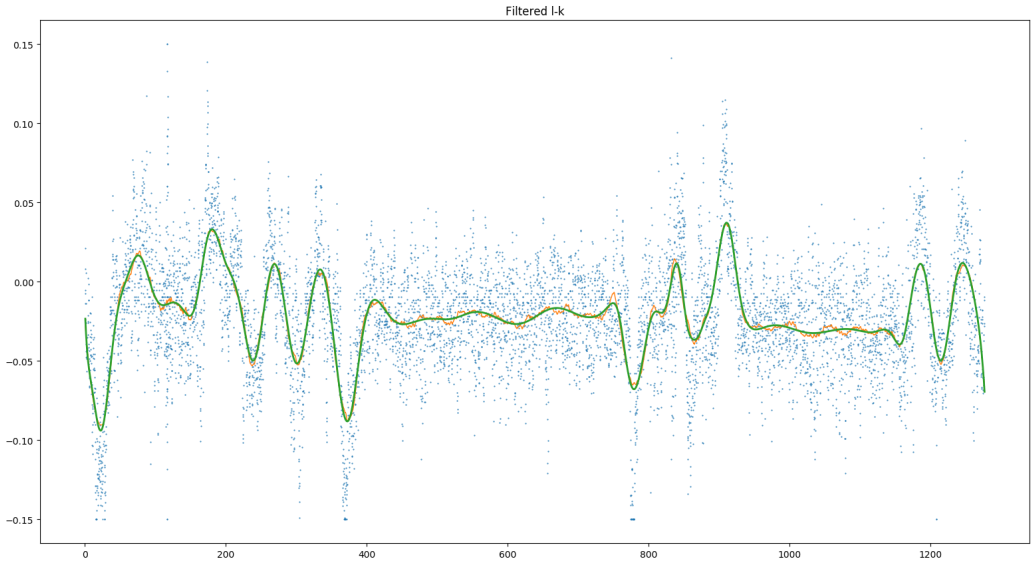
$$k_i = \frac{\Delta l_i}{D d_i} \tag{3}$$

$$\theta_i = theta_{i-1} + d_i k_i \tag{4}$$

$$x_i = x_{i-1} + \sin(\theta_i) d_i y_i = y_{i-1} + \cos(\theta_i) d_i l_i = l_{i-1} + d_i \tag{5}$$

The most important data is in the l, d, and k vectors. Any element in k greater in absolute value than `maxcurv` will be truncated to $\pm$ `maxcurv`. The k vector is passed through a savgol filter with the specified degrees of freedom and smoothing. This removes most of the noise but still leaves a curvature profile that is rough and jagged. This filtered data is then passed through a univariate spline with 5 degrees of freedom and the specified amount of smoothing.

Visually, this looks as follows:



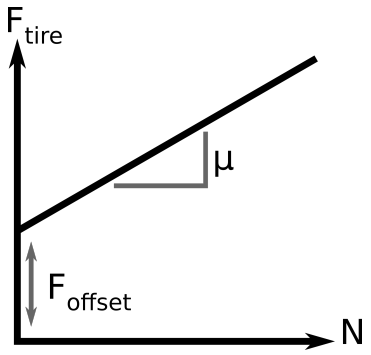
the unfiltered k vector is the blue dots, the filtered data is the orange line, and the smoothed data is the green curve.

The fitting process is done at runtime, but the track can be previewed by using `input_processing.py` as a command-line tool so that the fitting parameters may be adjusted. Usage: `input_processing.py path/to/logfile.log <segment size>`

3 Component Models

3.1 Tires

Tire models are extremely complicated if they are to be accurate- but this complexity can be a distraction from the major characteristics of tires that concern us. This behavior is that tires do not have a constant coefficient of friction- it decreases with additional normal force. Alternatively, we can model the tires as having some baseline, or 'offset' grip at zero normal force, and as normal force increases, maximum grip increases linearly.



$$F_{tire,max,lat} = F_{tire,offset,lat} + \mu_{tire,lat} N \tag{6}$$

$$F_{tire,max,long} = F_{tire,offset,long} + \mu_{tire,long} N \tag{7}$$

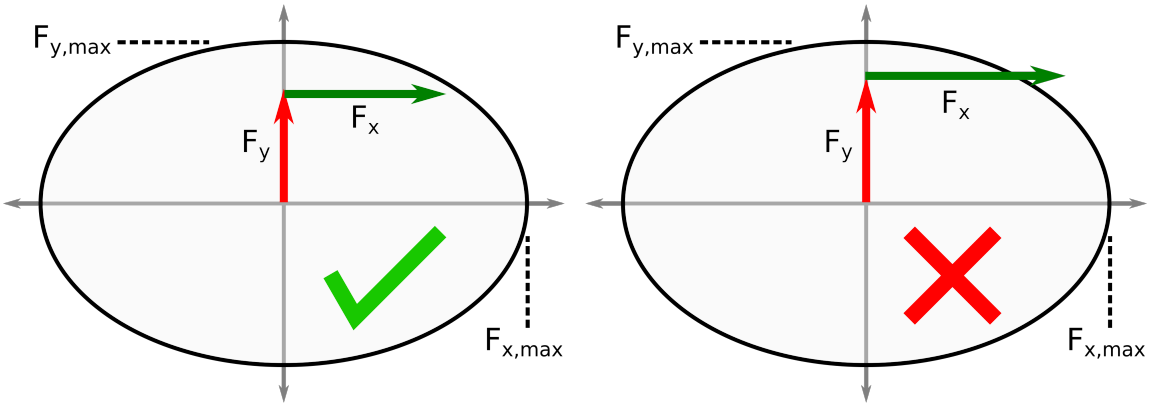
The parameters here are only for one tire. If we wish to look at the lumped characteristics of multiple tires, we will need to do some simple division, and at the end of it all, multiply by the number of tires.

$$F_{tire,max,lat} = F_{tire,offset,lat} + \mu_{tire,lat}\frac{N}{n_{tires}} \tag{8}$$

$$F_{tire,max,long} = F_{tire,offset,long} + \mu_{tire,long}\frac{N}{n_{tires}} \tag{9}$$

The parameters (μ and $F_{tire,offset}$) can be found from experimental tire data for various tires. The easiest method is to find lines that meet the outer capabilities of the tires, which will be at optimal operating conditions (camber, toe, slip rate, slip angle, and temperature).

Of course, a racecar tire is rarely purely in one of these conditions- we must determine the limits in a multi-axis loading. At a given normal force, a 'friction ellipse' can be produced that intercepts these limits.



This can be written as

$$\frac{F_{tire,lat}^2}{F_{tire,max,lat}^2} + \frac{F_{tire,long}^2}{F_{tire,max,long}^2} <= 1 \tag{10}$$

We can solve, then, for the 'remaining' lateral or longitudinal force that the tire can apply before slipping (this is still for one tire, but the lateral force is for the vehicle as a whole).

$$F_{tire,remaining,long} = F_{tire,max,long}\sqrt{1 - (\frac{F_{tire,lateral}}{n_{tires}F_{tire,max,lat}})^2} \tag{11}$$

3.2 Aerodynamics

Aerodynamic forces on the vehicle obey a square law relationship:

$$F_{down} = \alpha_{down}v^2, \tag{12}$$

$$F_{drag} = \alpha_{down}v^2, \tag{13}$$

$$\tag{14}$$

where α's are scaling constants computed from the aerodynamic forces at 35 miles per hour:

$$\alpha = \frac{F_{35MPH}}{(35MPH)^2} \tag{15}$$

The drag and downforce can be defined for three different aerodynamic 'modes': Full, Drag-Reduced (acceleration), and Airbrake (braking).

3.3 Powertrain

The force developed by the powertrain at the tires is modeled as a function of vehicle velocity and current gear.

$$\omega_{crank} = \frac{v}{r_{tire}} N_{final} N_{transmission,i_{gear}}, \tag{16}$$

$$F_{long,engine,limit} = T_{engine}(\omega_{crank}) N_{final} N_{transmission,i_{gear}}, \tag{17}$$

Where N is a gear ratio, i_{gear} is the gear number, and ω_{crank} is the angular velocity of the crankshaft. T_{engine} is a linear interpolation representing the torque curve of the engine. For velocities that cause the engine to hit its 'rev limiter', the engine is modeled as producing zero power. For velocities that are below the lowest datapoint specified in the torque curve, the engine is modeled as producing the torque at the lowest specified RPM.

It takes a finite amount of time, t_{shift} in order for a shift to occur. During this time, the powertrain produces no power. Another gear is chosen with a different gear than the currently selected one will yield a higher F_{tire}. When this happens, the driver shifts in the direction of this gear (the transmission is sequential, not fully manual). When the vehicle brakes, the gear is 'reset', and when the powertrain is used again, the best available gear is used, regardless of what was engaged prior.

3.4 Driver

The driver we assume (with some noted exceptions) is one which seeks to follow the ideal that racing is about maximizing acceleration in the correct direction. They use all available tire grip or engine power whenever possible, and brake as late and hard as possible. This driver does not exist in reality, but is a useful model allowing us to push the bounds for what the car can be driven to.

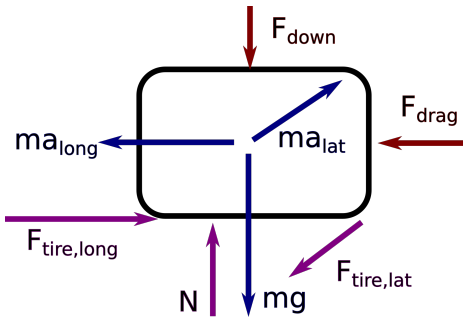
3.5 Code Implementation

These physics which are reused throughout different overall vehicle models are implemented in `vehicle.py`, which defines the `Vehicle` object type which contains methods to access this physics and has the attributes of the vehicle. Explanation of implementation is relatively straightforward and is left to the documentation in this file.

4 Single-Tire, Point-Mass Model

4.1 Physics

The physics behind a racecar are rather complicated. However, they can be simplified with the point mass assumption. This is the assumption that the car acts as a single point where all loads are applied, with no suspension to complicate tire grip and steering or tire characteristics.



Now, we can begin the vehicle physics. The forces in the vertical direction, with no vertical acceleration (I.E. perfectly flat ground) are:

$$\Sigma F_z = ma_z \tag{18}$$
$$-F_{down} - mg + N = 0 \tag{19}$$
$$N = mg + F_{down} \tag{20}$$

where m is the vehicle mass, g is the local acceleration due to gravity, and N is the normal force of the track on the tires.

Given a specific vehicle path, we will prescribe a curvature the vehicle must pass through. The sum of forces in the lateral direction is, then,

$$\Sigma F_{lat} = ma_{lat} \tag{21}$$
$$F_{tire,lat} = mv^2k, \tag{22}$$

where a_{lat} is lateral acceleration, v is instantaneous vehicle velocity, and k is curvature.

The longitudinal forces acting on the vehicle are that of the tire and drag:

$$\Sigma F_{longitudinal} = ma_{longitudinal} \tag{23}$$
$$a_{longitudinal} = \frac{\Sigma F_{longitudinal}}{m} \tag{24}$$
$$a_{longitudinal} = \frac{F_{long,tire} - F_{drag}}{m} \tag{25}$$

The force of the tire, though, can vary. In the case of braking, we assume the driver has the ability to lock up the tires, but does not; i.e., the tires apply all remaining grip:

$$F_{long,tire,braking} = -F_{tire,long,remaining}. \tag{26}$$

Acceleration, though, is more complicated, as the vehicle could be limited by either tire grip or engine power:

$$F_{long,tire,accel} = min(F_{tire,long,remaining}, F_{engine,limit}) \tag{27}$$

We can now compute the overall acceleration of the vehicle and its final velocity.

$$a_{long} = \frac{F_{long,tire,...} - F_{drag}(v0)}{m} \tag{28}$$
$$v_{final} = \sqrt{v_0^2 + 2a_{long}d_{segment}} \tag{29}$$

4.2 Code Implementation and Solver

After creating a vehicle with known parameters and a track made of segments, the process of solving can begin.

4.2.1 Steps

We will first examine a 'step'- how to get from the start state to end state of a given segment. The arguments for the 'step' function are as follows:

- `vehicle`: The vehicle and all its parameters
- `prior_result`: The simulation results from the previous step
- `segment`: The current segment properties

- `segment_next`: The next segment's properties
- `brake`: A flag indicating if braking is required
- `shifting`: A flag indicating if the vehicle is currently shifting inbetween gears
- `gear`: The current gear the vehicle is in

To begin, the final velocity and required accelerations, forces are calculated as described in the "Physics" section. If the tire grip is exceeded, an error is returned. The appropriate vehicle status is computed based on these values. To avoid oscillatory throttle-braking which also decreases cornering performance, a 'sustaining' strategy is utilized. If the vehicle is accelerating, running through a corner, and the prior status was braking, the next thing to do is sustain through the corner. If the vehicle was sustaining beforehand and curvature is relatively the same, continue to sustain. Sustaining is also triggered by running out of grip. Sustaining is simply solving for the steady-state velocity through a corner. Because of the nonlinearity of the equations, this is not a simple task. Something resembling bisection is employed. Luckily, the equations are simple enough that going faster will result in less remaining grip, and going slower will result in more.

Although this model does not allow for studying lateral load transfer, later ones do. To consistently study the effects of varying track, it is allowed to define an 'additional radius' that the vehicle is allotted to take a turn (if less track is provided, the radius is positive; if more track is used, the radius is negative).

$$k = \frac{k_{raw}}{1 + R_{additional} k_{raw}} \tag{30}$$

First, we will define remaining grip:

$$F_{long,remaining} = F_{long,remaining}(n_{tires} = 4, N = mg + F_{down}(v_f), F_{lat} = mv_f^2 k) - F_{drag}(v_f) \tag{31}$$

We will also determine the initial range of velocities we can use.

$$a_{max} = \frac{min(F_{engine}, F_{tire,long,remaining}) - F_{drag}}{m} \tag{32}$$

$$a_{min} = \frac{-F_{tire,long,remaining} - F_{drag}}{m} \tag{33}$$

$$v_{f,upper} = \sqrt{v_0^2 + 2a_{max}d} \tag{34}$$

$$v_{f,lower} = \sqrt{v_0^2 + 2a_{min}d} \tag{35}$$

The bisection algorithm, from here, is fairly straightforward:

Algorithm 1 Sustaining 'Bisection' Algorithm

while $excess < 0$ or $excess > \epsilon$ **do**
  $v_f \leftarrow \frac{v_{f,upper} + v_{f,lower}}{2}$
  run equations of motion
  if $F_{long,remaining}(n_{tires}, N, F_{lat}) - F_{drag}(v_f) < 0$ **then**
    $v_{f,upper} \leftarrow v_f$
  else
    $v_{f,lower} \leftarrow v_f$

4.2.2 Aerodynamic Strategy

The aerodynamic strategy is also quite straightforward. Run a step with one aero configuration, and with another. If only one works, use that one. If both work, use the one with the better longitudinal acceleration. When braking, the choice is between 'full' or normal aero kit and an airbrake. When accelerating, the choice is between a DRS aero kit and the full aero kit.

4.2.3 Overarching Solver

With the strategy for determining the optimal (and feasible) acceleration around a corner determined, we can begin to determine the overall driving strategy to determine the optimal drive around a track.

Each iteration of the solver begins by taking a step based on the previous conditions. If this fails, the braking algorithm is initialized. A shifting algorithm is also employed. Stitching these pieces together and managing the edge conditions results in a rather large loop.

4.2.4 Braking Algorithm

The braking algorithm runs inside the main solver. The overall strategy is to 'replay' the part of the track leading up to the failure (loss of grip), but not so far back that time is spent sustaining before a corner. The strategy to do this has two main stages. The first is a 'scanning' phase, where large steps backwards are made to find, roughly, a window in which braking can initialize. the second is the 'bisection' phase, where a bisection algorithm is used to find the point of optimal performance.

Algorithm 2 Braking Algorithm Pseudocode

1: Backup Output matrix (deep copy)
2: $failure\_point \leftarrow i$
3: $lower\_brake\_bound \leftarrow i - stepback\_amount$
4: $upper\_brake\_bound \leftarrow i$
5: **while** True **do**
6: $i = lower\_brake\_bound$
7: Re-step from i, with `brake` on, til failure or $i > failure\_point$
8: **if** failure **then**
9: $lower\_brake\_bound \leftarrow lower\_brake\_bound - stepback\_amount$
10: $upper\_brake\_bound \leftarrow upper\_brake\_bound - stepback\_amount$
11: **else**
12: Break out of loop
13: **while** $lower\_brake\_bound <= upper\_brake\_bound$ **do**
14: $i_{initial} = \frac{lower\_brake\_bound + upper\_brake\_bound}{2}$
15: $i = i_{initial}$
16: Re-step from i, with `brake` on, til failure or $i > failure\_point$
17: **if** failure **then**
18: $upper\_brake\_bound \leftarrow i_{initial}$
19: **else**
20: $lower\_brake\_bound \leftarrow i_{initial}$
21: Reset the Output matrix the the backup Output matrix from line 1

4.2.5 Shifting Algorithm

Shifting seems like a simple matter of always keeping the transmission moving to the best gear. However, this ignores an issue presented by engines with ever-increasing power curves, and frictional forces that slow down the vehicle when not in gear. This slowing down of the vehicle means that by the time the next gear is ready, the gear that was previously selected is now the better gear. While we want to be in the best gear at all times, this marginal performance increase is obviously outweighed by the result that no power would be delivered to the wheels at any time.

To combat this problem, we simply dictate that the previous speed must be exceeded with the new gear before any further shifting is allowed. The pseudocode for this can be summarized as follows, and is triggered by the best gear (as measured by theoretical force at the tires at the current vehicle velocity) being different than the current gear.

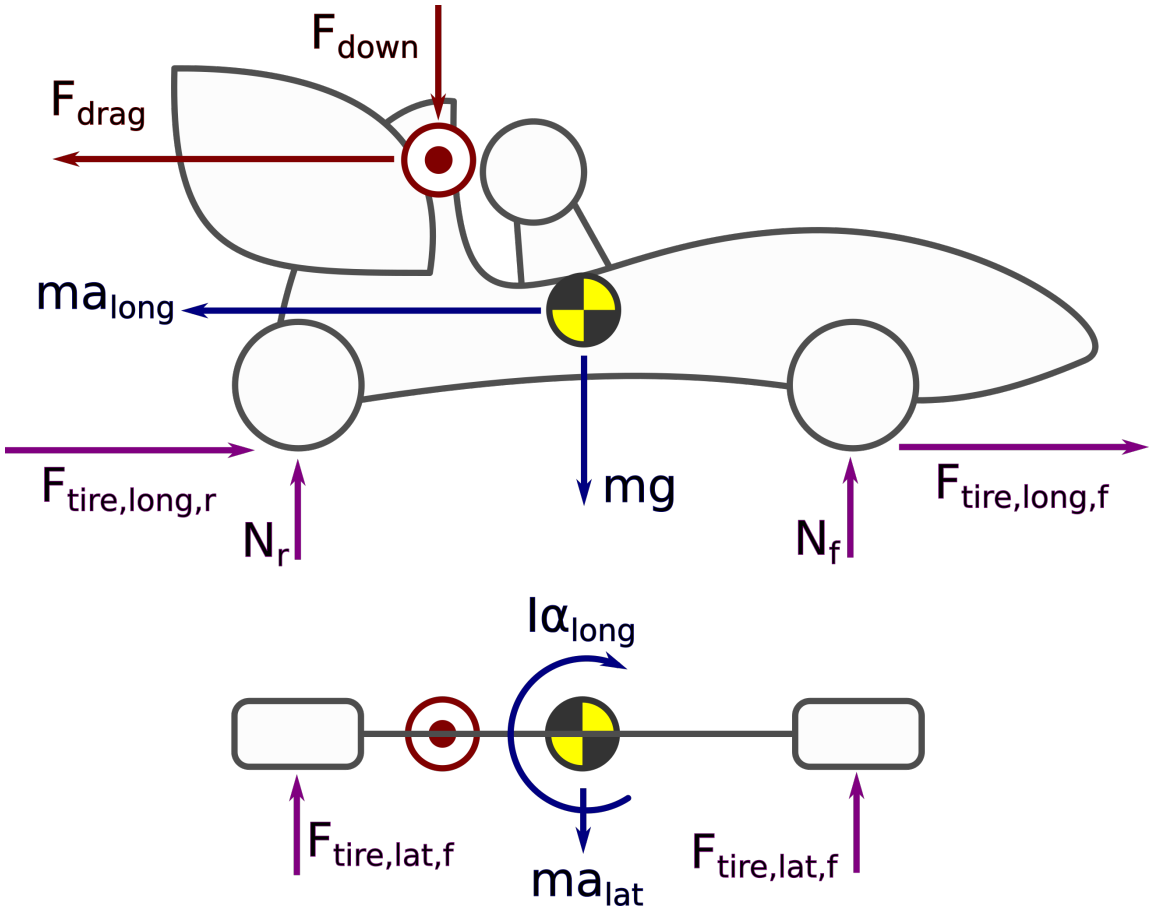
Algorithm 3 Shifting Algorithm Pseudocode

Compute *bestgear*
if ! `brake` and $bestgear \neq gear$ and $v > shift\_velocity$ **then**
    $gear \leftarrow gear + signof(bestgear - gear)$
    $shift\_velocity \leftarrow v_i \times (1 + \epsilon)$
    $t_{shift} \leftarrow t_i$
    while $t_i < t_{shift} + shift\_time$ **do**
        Run steps like normal, but disable engine power by setting the `shifting` to on.

5 Dual-Tire, Yaw-MOI Model

5.1 Physics

A slight improvement can be made by accounting for the longitudinal load transfer induced by acceleration and heave of the vehicle, and yaw moment effects. This enables us to study the effect of center-of-gravity (CG) and center-of-pressure (CP) locations, as well as brake biasing, and any other parameter that would effect or be effected by longitudinal load transfer.



The vehicle physics begin with the summation of forces and moments in the vertical, longitudinal, and lateral directions.

$$\Sigma F_{vert} = ma_{vert} \tag{36}$$
$$\Sigma F_{lat} = ma_{lat} \tag{37}$$
$$\Sigma F_{long} = ma_{long} \tag{38}$$
$$\Sigma M_{vert} = \Sigma ma \times r + I_{yaw}\alpha \tag{39}$$
$$\Sigma M_{lat} = \Sigma ma \times r + I_{yaw}\alpha \tag{40}$$
$$\tag{41}$$

$$\Sigma F_{vert} = N_f + N_r - mg - F_{down}(v_0) = 0 \tag{42}$$
$$\Sigma F_{lat} = F_{tire,lat,f} + F_{tire,lat,f} = ma_{lat} = mv^2k \tag{43}$$
$$\Sigma F_{long} = F_{tire,long,f} + F_{tire,long,f} - F_{drag}(v_0) = ma_{long} \tag{44}$$
$$\Sigma M_{vert} = F_{tire,lat,f} l_{wheelbase} = mv^2 k l_{CG} + I_{yaw}\alpha \tag{45}$$
$$\Sigma M_{lat} = N_f l_{wheelbase} + F_{drag}(v_0)h_{CP} - F_{down}(v_0)l_{CP} - mgl_{CG} = ma_{long}h_{CG} \tag{46}$$

where subscripts f denote relation to the front tires, and r denote relation to the rear tires. lCG and l_{CP} are the longitudinal distance from the rear tires to the CG and CP, respectively. h_{CG} and h_{CP} are the vertical height from the ground to the CG and CP, respectively. $l_{wheelbase}$ is the distance from the rear to front tires. I_{yaw} is the moment of inertia about the vertical axis passing through the CG. α is the yaw acceleration, which is computed as:

$$\omega = \frac{v}{r} = vk \tag{47}$$
$$\alpha = \frac{\mathrm{d}\omega}{\mathrm{d}t} = v\frac{\mathrm{d}k}{\mathrm{d}t} + k\frac{\mathrm{d}v}{\mathrm{d}t} \tag{48}$$
$$\alpha = v\frac{\mathrm{d}k}{\mathrm{d}x}\frac{\mathrm{d}x}{\mathrm{d}t} + ka_{long} \tag{49}$$
$$\alpha = v^2\frac{k_f - k_0}{\Delta x} + k_0 a_{long} \tag{50}$$

It should be noted that the summation of forces about the longitudinal axis results in the inertial force of the vehicle being the only thing to produce a moment, and as such, this is ignored, understanding that lateral load transfer is not modeled by this set of equations.

Solution of this is somewhat ugly, and most importantly, cannot be done explicitly. As such, the solution is attempted at first in a marginal fashion; this will be discussed later.

5.1.1 The Marginal Physics

This method begins by computing the required lateral force for each tire. $N_f, N_r, v0$ are presumed to be known from the previous iteration.

$$F_{tire,lat,f} = \frac{l_{CG}}{l_{wheelbase}}mv^2k + \frac{\alpha I_{yaw}}{l_{wheelbase}} \tag{51}$$
$$F_{tire,lat,r} = (1 - \frac{l_{CG}}{l_{wheelbase}})mv^2k - \frac{\alpha I_{yaw}}{l_{wheelbase}} \tag{52}$$

The remaining longitudinal forces $F_{tire,remaining,long}$ are computed as discussed previously (normal forces are presumed to be known).

Two models of braking may be employed: perfect brake biasing, and fixed brake biasing. Perfect brake biasing is simple:

$$F_{tire,long,f} = -F_{tire,remaining,long,f} \tag{53}$$
$$F_{tire,long,r} = -F_{tire,remaining,long,r} \tag{54}$$

Fixed brake biasing is less so. First, the maximum total braking force must be found:

$$F_{brake} = min(\frac{F_{tire,remaining,long,f}}{X_f}, \frac{F_{tire,remaining,long,r}}{X_r}), \tag{55}$$

where X is the brake bias for the front or rear tires ($X_f + X_r = 1$). Then, the brake force can be applied to the front and rear tires in the corresponding amount:

$$F_{tire,long,f} = -F_{brake}X_f \tag{56}$$
$$F_{tire,long,r} = -F_{brake}X_r \tag{57}$$

Acceleration is the same as the point-mass model, but it is important to note that the rear-wheel tires are considered; assuming RWD rather than FWD or AWD. The overall longitudinal force, longitudinal acceleration, and final velocity can then be computed.

$$\Sigma F_{longitudinal} = F_{tire,long,f} + F_{tire,long,r} - F_{drag}(v_0) \tag{58}$$
$$a_{long} = \frac{F_{tire,long,f} + F_{tire,long,r} - F_{drag}(v_0)}{m} \tag{59}$$
$$v_{final} = \sqrt{v_0^2 + 2a_{long}d_{segment}} \tag{60}$$

The normal force is computed for the next iteration, and for checks.

$$N_f = (1 - \frac{l_{CG}}{l_{wheelbase}})mg + (1 - \frac{l_{CP}}{l_{wheelbase}})F_{down} - ma_{long}\frac{h_{CG}}{l_{wheelbase}} - \frac{h_{CP}}{l_{wheelbase}}F_{drag} \tag{61}$$
$$N_r = \frac{l_{CG}}{l_{wheelbase}}mg + \frac{l_{CP}}{l_{wheelbase}}F_{down} + ma_{long}\frac{h_{CG}}{l_{wheelbase}} + \frac{h_{CP}}{l_{wheelbase}}F_{drag} \tag{62}$$

5.1.2 Working Backwards

This method determines the forces on the vehicle given a prescribed a_{long}.

The normal forces on the tires can be found by solving the summation of forces in the and vertical direction, and summation of moments about the lateral axis. This results in:

$$N_f = (1 - \frac{l_{CG}}{l_{wheelbase}})mg + (1 - \frac{l_{CP}}{l_{wheelbase}})F_{down} - ma_{long}\frac{h_{CG}}{l_{wheelbase}} - \frac{h_{CP}}{l_{wheelbase}}F_{drag} \tag{63}$$
$$N_r = \frac{l_{CG}}{l_{wheelbase}}mg + \frac{l_{CP}}{l_{wheelbase}}F_{down} + ma_{long}\frac{h_{CG}}{l_{wheelbase}} + \frac{h_{CP}}{l_{wheelbase}}F_{drag} \tag{64}$$

The required lateral forces from the tires are also computed:

$$F_{tire,lat,f} = \frac{l_{CG}}{l_{wheelbase}}mv^2k + \frac{\alpha I_{yaw}}{l_{wheelbase}} \tag{65}$$
$$F_{tire,lat,r} = (1 - \frac{l_{CG}}{l_{wheelbase}})mv^2k - \frac{\alpha I_{yaw}}{l_{wheelbase}} \tag{66}$$

From here, the required longitudinal tire grip can be computed.

$$F_{tire,long,total,required} = \frac{a_{long}}{m} + F_{drag}(v) \tag{67}$$

We must then compute the required longitudinal tire grip to maintain the desired a_{long} and counteract drag. For the acceleration case, this is straightforward:

$$F_{tire,long,r,required} = F_{tire,long,total,required} \tag{68}$$

For the braking case where fixed biasing is used, the process is very simple: distribute the required force between the tires.

$$F_{tire,long,r,required} = F_{tire,long,total,required} X_r \tag{69}$$
$$F_{tire,long,f,required} = F_{tire,long,total,required} X_f \tag{70}$$

For the braking case where perfect biasing is used, this requires some thinking. Since in this model, the front and rear braking forces are colinear, it makes no difference to where the braking force is distributed (provided that the maximum grip is not exceeded). We will use up all of the more-available tire's grip before using the less-available tire's grip.

Algorithm 4 Perfect Biasing Braking Algorithm

1: $F_{tire,long,more,required} = F_{tire,long,total,required}$
2: **if** $F_{tire,long,remaining,more} < F_{tire,long,more,required}$ **then**
3: $\quad F_{tire,long,remaining,less} = F_{tire,long,total,required}$ - $F_{tire,long,more,required}$

From here, checks may be performed to ensure that grip is used appropriately, and final velocity is calculated much like before.

5.2 Code Implementation and Solver

5.2.1 Take A Step

Taking a step is significantly more complicated in the two-tire model, because acceleration induces longitudinal load transfer. As mentioned before, it is difficult to solve the equations of motion. More complex checks must be employed. The equations described in "The Marginal Physics" are employed at first. After this, the remaining longitudinal grip is evaluated at the new vehicle velocity, and the next segment's curvature. If there is not sufficient grip on either of the tires, then we will need to iteratively solve for an acceleration that will work and still be optimal.

To solve for an acceleration that will work and still be optimal, a simple loop is employed to scan over a range of vehicle velocities (which are the same as the bounds for bisection used in the point-mass model). The physics discussed in the "Working Backwards" section are employed to check. This was intended to be bisection at first, as this would enable more precision. How to make this work was not immediately self-evident though, but Thad does have some ideas on how to make it work. As it stands, the range of possible vehicle velocities is swept through, and then the 'best' (highest amplitude of acceleration, depending on if braking is desired) velocity is chosen.

5.2.2 Putting it Together

The overall solver for the two-tire model is identical to the point-mass model, save for the braking algorithm employed. The braking algorithm employed here does not use the bisection stage, but uses smaller steps backwards. This was done because of solution stability. Bisection is the ideal algorithm, but stability issues meant having to forgo the computational efficiency of the bisection algorithm.

6 Point Simulations

Once multiple lap time simulations have been ran, the next important thing to do is put the pieces together and simulate points at a competition.

The points scored at each event can be found from the FSAE rulebook:

$$P_{acceleration} = 95.5\frac{1.5t_{best}}{0.5(t_{acceleration} - 1)} + 4.5 \tag{71}$$

$$P_{skidpad} = 71.5\frac{\frac{1.45t_{best}}{t_{skidpad}}^2 - 1}{1.45^2 - 1} + 3.5 \tag{72}$$

$$P_{autocross} = 118.5\frac{1.45t_{best}}{0.45(t_{autocross} - 1)} + 6.5 \tag{73}$$

$$P_{endurance} = 250\frac{1.45t_{best}}{0.45(t_{endurance} - 1)} + 25 \tag{74}$$

Efficiency score calculation is more complex.

$$EF_i = \frac{t_{best}}{t_{endurance}}\frac{CO2_{best}}{CO2_{endurance}} \tag{75}$$

$$P_{efficiency} = 100\frac{\frac{EF_{min}}{EF_i} - 1}{\frac{EF_{min}}{EF_{max}} - 1} \tag{76}$$

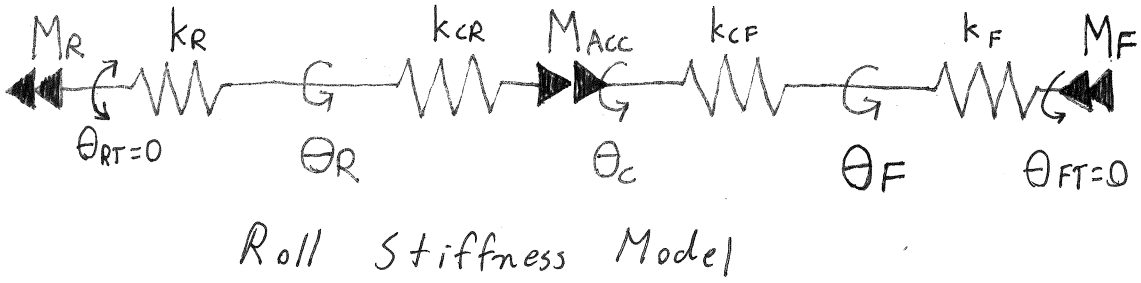
Lap times, best times at previous tracks, and estimated CO2 production can simply be plugged into these formulae. Estimation of the CO2 production, hopefully, is good enough.

7 Four-Tire, Yaw-MOI Model

This model is mostly a tweak of the two-tire model, so the physics changes only will be discussed here.

7.1 Physics

Adding four tires to the chassis is fairly simple from this point. We consider the chassis and suspension as a torsional spring assembly as shown below:



where:

- M_r and M_f are the moments by the rear and front tires
- M_f is the moment produced by the front tires
- M_{acc} is the moment produced by lateral acceleration
- k_f and k_r is the suspension roll stiffness for front and rear
- θ_f, $theta_r$, and $theta_c$ is the vehicle roll at the front and rear suspension points, and CG

We can write the following spring equations:

$$k_r\theta_r = M_r \tag{77}$$
$$k_f\theta_f = M_f \tag{78}$$
$$k_c r(\theta_c - \theta_r) = M_r \tag{79}$$
$$k_c f(\theta_c - \theta_f) = M_f \tag{80}$$

Including the torque balance equation leaves us with 5 equations and 5 unknowns (M_r, M_f, and θ's):

$$M_{acc} = M_r + M_f \tag{81}$$

Solving these equations yields:

$$M_f = M_{acc}k_{cf}k_f\frac{k_{cr}+k_r}{k_{cf}k_{cr}k_f + k_{cf}k_{cr}k_r + k_{cf}k_fk_r + k_{cr}k_fk_r} \tag{82}$$
$$M_r = M_{acc}k_{cr}k_r\frac{k_{cf}+k_f}{k_{cf}k_{cr}k_f + k_{cf}k_{cr}k_r + k_{cf}k_fk_r + k_{cr}k_fk_r} \tag{83}$$

Generally, k_r, k_f are roughly known, but k_cr and k_cf are not- but overall chassis k_c stiffness is. We will approximate the chassis as having linearly distributed stiffness. This gives us:

$$k_c r = \frac{k_c}{\frac{l_{CG}}{l_{wheelbase}}} \tag{84}$$
$$k_c f = \frac{k_c}{1 - \frac{l_{CG}}{l_{wheelbase}}} \tag{85}$$

It is important to note that this assumes that the ground is flat, and assumes steady state (that us, no dampers nor roll acceleration).

If we know the forces on the front tires and on the rear tires N_f and N_r we can compute the normal force on each tire. Inside tires have subscript 1, outside subscript 2.

$$N_{f,1} = \frac{N_f}{2} - \frac{M_f}{w_f} \tag{86}$$
$$N_{f,2} = \frac{N_f}{2} + \frac{M_f}{w_f} \tag{87}$$
$$N_{r,1} = \frac{N_r}{2} - \frac{M_r}{w_r} \tag{88}$$
$$N_{r,2} = \frac{N_r}{2} + \frac{M_r}{w_r} \tag{89}$$

Currently, the tire lateral grips are distributed between tires to try and equalize the longitudinal grip. If this is not possible (i.e. one tire has immense grip), the lateral load is taken completely by the better-loaded tire, and the maximum tire grip for each tire is assumed to be that of the lower one. This enforces the assumption of 50/50 left-right brake bias (a good assumption) and an open differential (a poor assumption).

8 Future Plans

RoseLap

9 Appendix

Listing 1: Example Vehicle Definition File

```
### Starter Vehicle File ###
# An example vehicle to help you get started in creating a vehicle
# units are lbf, ft, s
# updated 7/2/2018

### Getting started ###
# A vehicle definition file contains the baseline parameters of a vehicle.
# This file is written in YAML ('YAML Aint Markup Language').
# If you are confused by the syntax, https://learn.getgrav.org/advanced/yaml
    is an easy reference.

### Aerodynamics ###
# The first in the list is the number in the 'full' or default configuration,
# second is 'DRS' condition,
# third is 'airbrake' configuration.
# To 'omit' a configuration, simply make the values the same as the 'full'
    configuration.

downforce_35mph: [61.0, 61.0, 61.0] # lbf - the downforce produced at 35 MPH
drag_35mph:      [40.0, 40.0, 40.0] # lbf - the drag produced at 35 MPH
cp_height: [0.9, 0.9, 0.9]          # ft - the height of the center of
    pressure
cp_bias:   [0.4, 0.4, 0.4]          # fraction - the amount of the downforce
    that acts on the front tires

### Tires ###
# See the RoseLap documentation for explanation of these values.
# The front and rear tires are used for the two- and four- tire models.
# The lumped tire is used for the onetire (aka pointmass) model.

## Front Tires ##
# Longitudinal Parameters
front_tire_mu_x: 2.8196   # lbf/lbf
front_tire_offset_x: 4.73 # lbf
# Lateral Parameters
front_tire_mu_y: 2.0050   # lbf/lbf
front_tire_offset_y: 8.61 # lbf
# Tire radius
front_tire_radius: 0.75   # ft

## Rear Tires ##
# Longitudinal Forces
rear_tire_mu_x: 2.8196
rear_tire_offset_x: 4.73
# Lateral Forces
rear_tire_mu_y: 2.0050
rear_tire_offset_y: 8.61
# Tire radius
rear_tire_radius: 0.75

## Lumped Tire ##
# Longitudinal Forces
comb_tire_mu_x: 2.8196
comb_tire_offset_x: 4.73
# Lateral Forces
comb_tire_mu_y: 2.0050
```

```
comb_tire_offset_y: 8.61
# Tire radius - make this the rear tire radius (used for gear ratio calcs)
comb_tire_radius: 0.75

### Suspension / Chassis ###

## Roll stiffnesses ##
k_roll_front: 100.0 # ft-lbf / deg - the roll stiffness of the front
    suspension
k_roll_rear:  200.0 # ft-lbf / deg - the roll stiffness of the rear
    suspension
k_chassis:    800.0 # ft-lbf / deg - the roll stiffness of the chassis


## Mass Parameters ##
# Note: as long as we use imperial units, lbm is used, but internally, the
    value is converted to slugs.
mass: 550      # lbm - The overall mass of the vehicle including driver
moi_yaw: 1100  # lbm*ft^2 - The overall moment of inertia including driver
    about the vertical axis passing through the CG.
cg_height: 0.6 # ft - The height of the center of gravity from the ground at
    static height

## Size Parameters ##
wheelbase_length: 5.1666667 # ft - The length of the wheelbase
weight_bias: 0.45   # fraction - the amount of weight that is on the front
    tires
track_front: 4.5    # ft - the track width of the front tires
track_rear:  4.4    # ft - the track width of the rear tires
r_add: 0 # ft - additional radius that can be employed to make a turn. This
    allows us to see the impact of increasing/decreasing track width on
    tightness of turns.

### Brakes ###

perfect_brake_bias: true # boolean - if true, brake bias is set 'perfectly'
    so all available grip is used. If false, brake_bias is used.
brake_bias: 0.67         # fraction - the amount of braking force to have on
    the front tires compared to total braking force

### Engine ###
# An engine curve given by two lists.
engine_rpms: # RPM
  - 3500
  - 4500
  - 5500
  - 6500
  - 7500
  - 8500
  - 9500
engine_torque: # ft-lbf
  - 24.3
  - 26.2
  - 27.4
  - 26.5
  - 25.5
  - 23.8
  - 23.9

# Correlated efficiency factors for 100 Octane gasoline
```

```
co2_factor: 2.31 #kg/L - The amount of CO2 produced per liter of fuel
    consumed
e_factor: 3.5e6 #ft*lbf/L - The amount of useful mechanical energy produced
    per liter of fuel consumed

### Transmission ###
# Note: overall transmission ratio is calculated as engine_reduction*gear[
    gear_no]*final_drive_reduction
engine_reduction: 2.81 # ratio - A reduction applied to all gears
gears:    # ratio - a list of reductions available to the driver (any number
    of gears can be specified)
  - 2.416
  - 1.92
  - 1.562
  - 1.277
  - 1.05
final_drive_reduction: 2.7692 # ratio - the final drive reduction

shift_time: 0.2 # s - the time taken to complete a shift
```