

RoseLap v4: Lap Time and Point Simulation

Thaddeus Hughes

June 9, 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

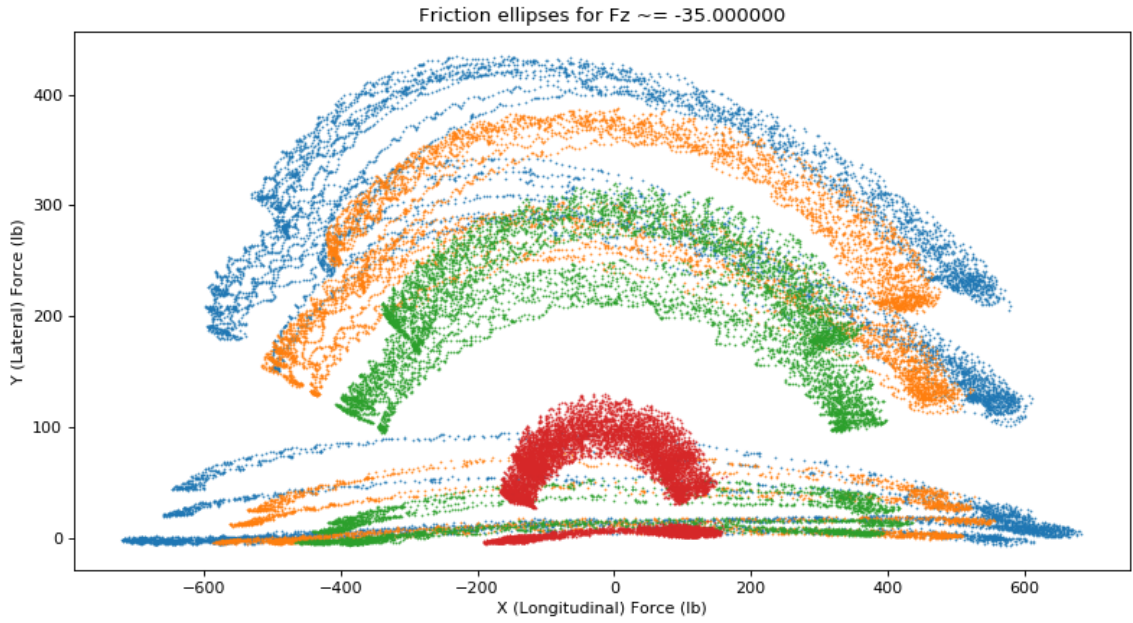
Blah blah dxfs and YAML

2.1 Tire Data Fitting

As discussed in the 'Tires' section, tire models used in RoseLap v4 are rather simplistic, but are more sophisticated than the constant COF presumed in v3 and prior. To determine the four parameters, some fitting must be done.

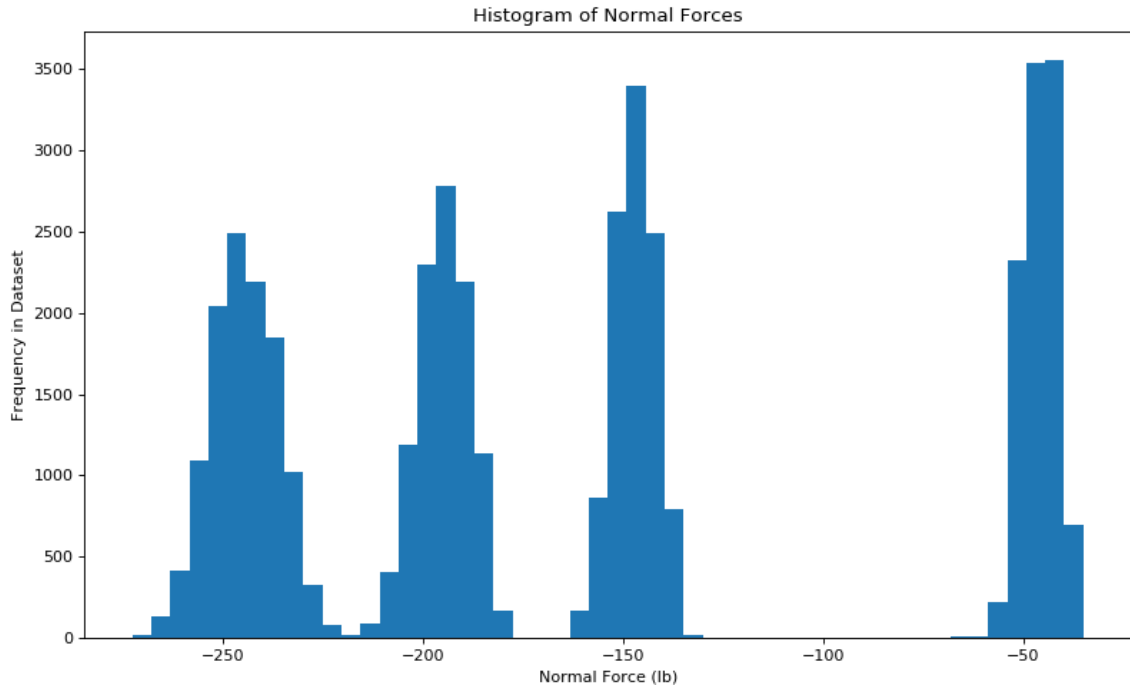
Tire data is obtained through physical testing. We use the data purchased from the Milliken Tire Test Consortium (TTC). This contains drive-brake and combined loading, with sweeps over various conditions.

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

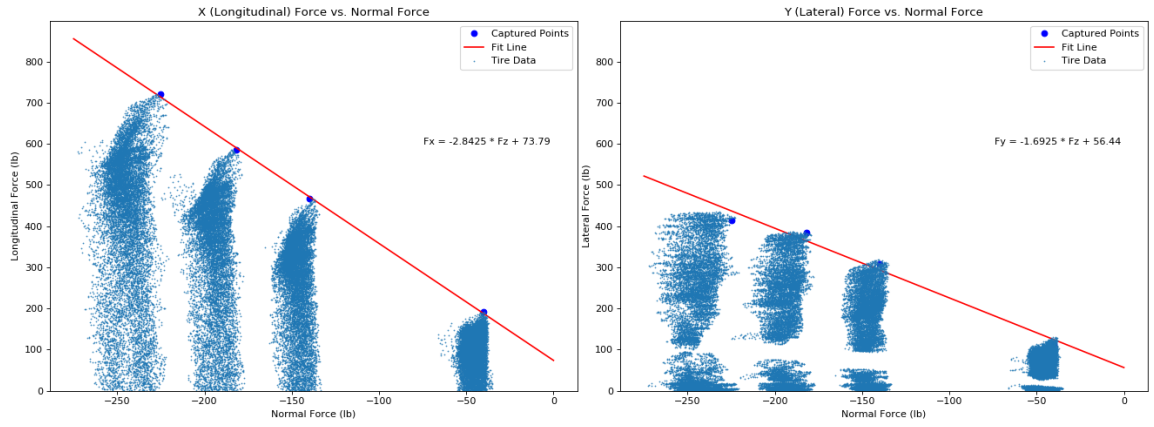


The multiple colors indicate different normal forces. While imperfect, it is easy to see how fitting an ellipse to this would provide a reasonable model of tire behavior.

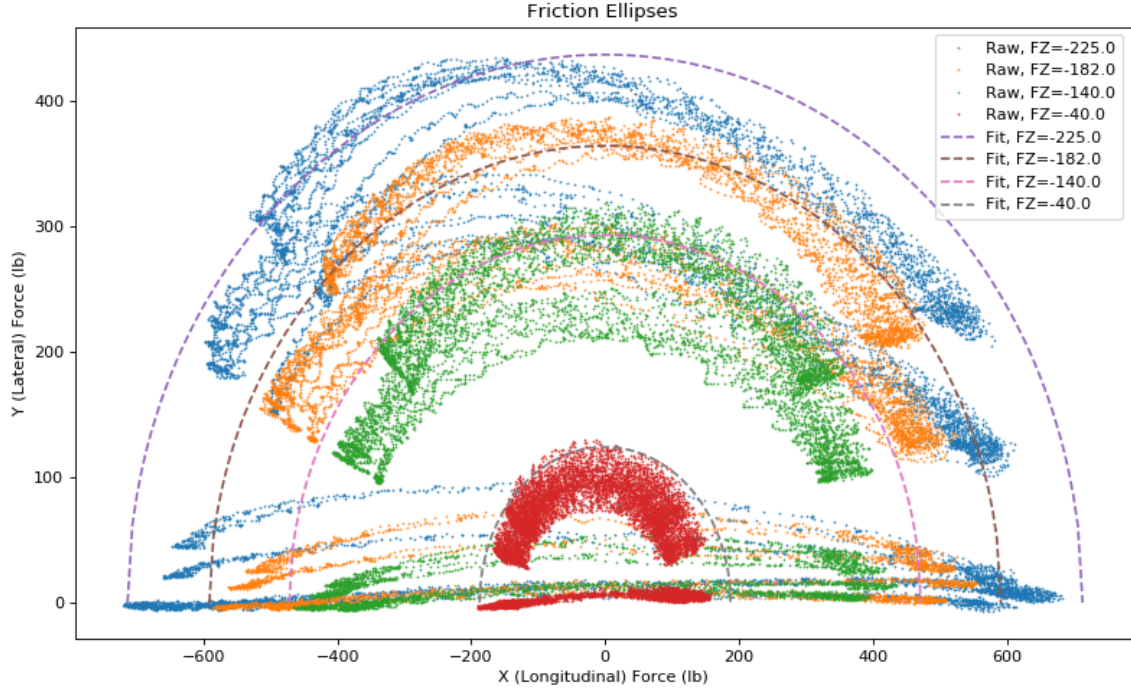
The first step is to determine the bins to which tire data vertical loads fall into. This is accomplished with a histogram.



Then, the data is analyzed in the F_z - F_x space and the F_z - F_y 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 F_x - F_y 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.

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.

$$F_{tire,max,lat} = F_{tire,offset,lat} + \mu_{tire,lat}N \quad (1)$$

$$F_{tire,max,long} = F_{tire,offset,long} + \mu_{tire,long}N \quad (2)$$

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.

$$F_{tire,max,lat} = F_{tire,offset,lat} + \mu_{tire,lat} \frac{N}{n_{tires}} \quad (3)$$

$$F_{tire,max,long} = F_{tire,offset,long} + \mu_{tire,long} \frac{N}{n_{tires}} \quad (4)$$

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

$$\frac{F_{tire,lat}^2}{F_{tire,max,lat}^2} + \frac{F_{tire,long}^2}{F_{tire,max,long}^2} \leq 1 \quad (5)$$

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

$$F_{tire,remaining,long} = F_{tire,max,long} \sqrt{1 - \left(\frac{F_{tire,lateral}}{n_{tires} F_{tire,max,lat}} \right)^2} \quad (6)$$

3.2 Aerodynamics

Aerodynamic forces on the vehicle obey a square law relationship:

$$F_{down} = \alpha_{down} v^2, \quad (7)$$

$$F_{drag} = \alpha_{down} v^2, \quad (8)$$

$$(9)$$

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

$$\alpha = \frac{F_{35MPH}}{(35MPH)^2} \quad (10)$$

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.

$$\omega_{crank} = \frac{v}{r_{tire}} N_{final} N_{transmission,igear}, \quad (11)$$

$$F_{long,engine,limit} = T_{engine}(\omega_{crank}) N_{final} N_{transmission,igear}, \quad (12)$$

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:

$$-F_{down} - mg + N = ma_z = 0, \quad (13)$$

$$N = mg + F_{down} \quad (14)$$

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,

$$\Sigma F_{lat} = ma_{lat} \quad (15)$$

$$F_{tire,lat} = mv^2k, \quad (16)$$

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:

$$a_{longitudinal} = \frac{\Sigma F_{longitudinal}}{m} \quad (17)$$

$$a_{longitudinal} = \frac{F_{long,tire} - F_{drag}}{m} \quad (18)$$

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:

$$F_{long,tire,braking} = -F_{tire,long,remaining}. \quad (19)$$

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

$$F_{long,tire,accel} = \min(F_{tire,long,remaining}, F_{engine,limit}) \quad (20)$$

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

$$a_{long} = \frac{F_{long,tire,...} - F_{drag}(v_0)}{m} \quad (21)$$

$$v_{final} = \sqrt{v_0^2 + 2a_{long}d_{segment}} \quad (22)$$

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.

4.3 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.

First, we will define remaining grip:

$$F_{excess} = F_{long,remaining}(n_{tires} = 4, N = mg + F_{down}(v_f), F_{lat} = mv_f^2 k) - F_{drag}(v_f) \quad (23)$$

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

$$a_{max} = \frac{\min(F_{engine}, F_{tire, long, remaining}) - F_{drag}}{m} \quad (24)$$

$$a_{min} = \frac{-F_{tire, long, remaining} - F_{drag}}{m} \quad (25)$$

$$v_{f, upper} = \sqrt{v_0^2 + 2a_{max}d} \quad (26)$$

$$v_{f, lower} = \sqrt{v_0^2 + 2a_{min}d} \quad (27)$$

The bisection algorithm, from here, is fairly straightforward:

Algorithm 1 Sustaining 'Bisection' Algorithm

```

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

```

4.4 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.

4.5 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.

4.5.1 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  $lower\_brake\_bound \leq upper\_brake\_bound$  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
```

4.5.2 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

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

5 Dual-Tire, Point-Mass 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. 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.

$$\Sigma F_{vert} = N_f + N_r - mg - F_{down}(v_0) = ma_{vert} = 0 \quad (28)$$

$$\Sigma F_{lat} = F_{tire,lat,f} + F_{tire,lat,r} = ma_{lat} = mv^2k \quad (29)$$

$$\Sigma F_{long} = F_{tire,long,f} + F_{tire,long,r} - F_{drag}(v_0) = ma_{long} \quad (30)$$

$$\Sigma M_{vert} = F_{tire,lat,f}l_{wheelbase} = ma_{lat}l_{CG} = mv^2kl_{CG} \quad (31)$$

$$\Sigma M_{lat} = N_f l_{wheelbase} + F_{drag}(v_0)h_{CP} - F_{down}(v_0)l_{CP} - mgl_{CG} = ma_{long}h_{CG} \quad (32)$$

where subscripts f denote relation to the front tires, and r denote relation to the rear tires. l_{CG} 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.

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.

5.2 The Marginal Physics

This method begins by computing the required lateral force for each tire. N_f, N_r, v_0 are presumed to be known from the previous iteration.

$$F_{tire,lat,f} = \frac{l_{CG}}{l_{wheelbase}}mv^2k \quad (33)$$

$$F_{tire,lat,r} = (1 - \frac{l_{CG}}{l_{wheelbase}})mv^2k \quad (34)$$

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:

$$F_{tire,long,f} = -F_{tire,remaining,long,f} \quad (35)$$

$$F_{tire,long,r} = -F_{tire,remaining,long,r} \quad (36)$$

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

$$F_{brake} = \min(\frac{F_{tire,remaining,long,f}}{X_f}, \frac{F_{tire,remaining,long,r}}{X_r}), \quad (37)$$

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:

$$F_{tire,long,f} = -F_{brake}X_f \quad (38)$$

$$F_{tire,long,r} = -F_{brake}X_r \quad (39)$$

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.

$$\Sigma F_{longitudinal} = F_{tire,long,f} + F_{tire,long,r} - F_{drag}(v_0) \quad (40)$$

$$a_{long} = \frac{F_{tire,long,f} + F_{tire,long,r} - F_{drag}(v_0)}{m} \quad (41)$$

$$v_{final} = \sqrt{v_0^2 + 2a_{long}d_{segment}} \quad (42)$$

5.3 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:

$$(43)$$

The required lateral forces from the tires are also computed:

$$F_{tire,lat,f} = \frac{l_{CG}}{l_{wheelbase}}mv^2k \quad (44)$$

$$F_{tire,lat,r} = (1 - \frac{l_{CG}}{l_{wheelbase}})mv^2k \quad (45)$$

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

$$F_{tire,long,total,required} = \frac{a_{long}m}{g} + F_{drag}(v) \quad (46)$$

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:

$$F_{tire,long,r,required} = F_{tire,long,total,required} \quad (47)$$

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

$$F_{tire,long,r,required} = F_{tire,long,total,required}X_r \quad (48)$$

$$F_{tire,long,f,required} = F_{tire,long,total,required}X_f \quad (49)$$

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 rear tire's grip before using the front tire's grip.

Algorithm 4 Perfect Biasing Braking Algorithm

- 1: $F_{tire,long,r,required} = F_{tire,long,total,required}$
 - 2: **if** $F_{tire,long,remaining,r} < F_{tire,long,r,required}$ **then**
 - 3: $F_{tire,long,remaining,f} = F_{tire,long,total,required} - F_{tire,long,r,required}$
-

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

5.4 Code Implementation and Solver

6 Point Simulations

6.1 Competition Formulae

6.2 Code Implementation

7 Post-Processing

8 Future Plans

Write your conclusion here.

Tire temps would be interesting??? CVTs maybe yo