LAP TIME SIMULATION OF A FORMULA STUDENT RACING CAR

Kester Broatch

2141351B

Final year project submitted in accordance with the requirements of
BEng Aeronautical Engineering

The University of Glasgow
School of Engineering
April 2019

Supervisors:
Dr David Anderson
Prof Konstantinos Kontis

Project ENG4110P

University of Glasgow

Abstract

The competitive nature and tight timescales in racing has pushed teams to rely heavily on computer simulation methods to design the overall vehicle package. An accurate and versatile Lap Time Simulator allows the best vehicle concept and set-up to be realised well before testing and racing takes place.

The overall objective of this thesis is to develop a transient lap time simulator for a Formula Student style racing car. This will allow the team to find the best vehicle set-ups while in the testing stage, and will allow future concept feasibility studies to be carried out when deciding upon future designs.

A Lap Time Simulator was developed in Simulink to represent a Formula Student car. The vehicle model subsystems were primarily based off of empirical data gathered from subsystem bench testing. The simulated lap times were compared to the real lap times for two test tracks. Both comparisons showed an accuracy of around 10% for initial un-optimised tests. Future work in the suspension, load transfer and gearing models was identified as a possible improvement in accuracy.

An optimisation tool was developed which allowed the fastest vehicle speed profile to be realised for a given track and vehicle. Future expansion of this optimisation tool will in theory allow the UGRacing Formula Student team to find the most competitive vehicle package attainable within their budget and capability.

University of Glasgow

Acknowledgments

I would like to take this opportunity to thank my supervisor Dr David Anderson for his advise, support and patience throughout my final year project. A special thanks goes to the UGRacing Formula Student team, who have provided endless advice and feedback throughout the projects development. Team members Miguel Cuni, Craig Davidson, Matt Koren, Peter Cochran and Callum Wrathall and many others offered assistance in the testing and validation of the model which is much appreciated. Thank you to the Forrestburn Hill climb track's staff, without whom much of the testing data could not have been collected. Lastly, I would like to thank SiTech Racing for providing dynamometer services and the Tyre Testing Consortium for providing tyre data free of charge.

University of Glasgow

Contents

University of Glasgow

List of Figures

List of Tables

University of Glasgow

Abbreviations

CG	Centre of Gravity
DOF	Degrees of Freedom
FSUK	Formula Student UK
IMeche	Institute of Mechanical Engineers
LTS	Lap Time Simulator
PI	Proportional Integration
SAE	Society of Automotive Engineers
TTC	Tyre Testing Consortium
UGR	University of Glasgow Racing
UGRacing	University of Glasgow Racing
UGR18	University of Glasgow Racing's 2018 Car

Nomenclature

Variables (English):

A	= Area
a	= CG distance from front axle, m
B	= Track width, m
BR	= Brake bias ratio
b	= CG distance from rear axle, m
C_D	= Drag coefficient
C_L	= Lift coefficient
e	= Lateral error vector, m
F	= Force, N
$F_{LatTran}$	= Lateral Load Transfer, N
$F_{LonTran}$	= Longitudinal Load Transfer, N
g	= Acceleration due to gravity, 9.81m/s^2
I_{wheel}	= Inertia of wheel, kgm^2
I_{zz}	= Yaw inertia of vehicle about z axis at CG, kgm^2
m	= Mass, kg
PR	= Pedal Ratio
r	= Wheel Radius, m
S	= Displacement, m
T	= Torque, Nm
V	= Velocity, m/s
w	= Lateral error weighting vector

Variables (Greek):

α	= Slip angle, rad
γ	= Wheel inclination (camber) angle, rad
δ	= Wheel Angle, rad
θ	= Steering Wheel Angle, rad
κ	= Slip ratio
μ	= Friction Coefficient
ψ	= Yaw Angle, rad
ω	= Wheel Angular Velocity, rad/s

Subscripts:

brake	= Produced by brakes
calliper	= Produced by brake calliper
drive	= Produced by Drivetrain
driver	= Produced by Driver
FL	= Front Left Wheel
FR	= Front Right Wheel
inert	= In the inertial coordinate system
master	= Master cylinder
pad	= Brake pad
RL	= Front Left Wheel
RR	= Front Left Wheel
tyre	= Produced by tyre
wheel	= at/of wheel
x	= along or around x axis
y	= along or around y axis
z	= along or around z axis

Axis Systems:

SAE vehicle dynamics standard axis system for wheels and 2 axle vehicles [1]

University of Glasgow

1 Introduction

In motorsport the results of a competition are often determined by tenths or hundredths of a second. The winner of a race is the vehicle and driver combination which can get around a track in the smallest amount of time or, in other words, the winner is the team which has the highest net forward acceleration. There are many parameters which affect the vehicle's overall forward acceleration, the main contributors of which are the driver, powertrain, tyres, suspension, aerodynamics and weight.

The Formula Student UK competition consists of five different dynamic events and four static events [2] each with varying points available, as seen in Table 1.

Table 1: FSUK competition points breakdown

Static Events:		
Business Plan	75	points
Cost and Manufacturing	100	points
Engineering Design	150	points
Dynamic Events:		
Skid Pad	75	points
Acceleration	75	points
Autocross	100	points
Endurance	325	points
Efficiency	100	points
Overall:	1000	points

67.5% of the points available in the FSUK competition are from the dynamic events. To be successful teams must be particularly competitive in this area. All of the dynamic events conflict each other in terms of vehicle performance, so it is vital to strike the right balance in the vehicle parameters which can maximise the points scored.

In practice there are a huge number of vehicle parameter combinations which can be used for a given track and conditions. To attempt to find the best set-up through testing alone proves to be an extremely expensive, risky and time consuming task and requires a very skilled team. F1 realised this as an issue in 2008 [3] and mandated a 30,000km testing limit

to keep the cost of development fair for smaller teams. This forced teams to rely more heavily on techniques such as wind tunnel testing, CFD, FEA and in particular, lap time simulations.

Lap Time Simulators typically combine a vehicle model, controller and track model to produce an estimated lap time. A Lap Time Simulator allows designers to observe and investigate the effect of changing parameters on the race car performance in an attempt to optimise the vehicle design and set up. Use of an LTS does not replace track testing, but it does reduce the amount of testing time and expense required to achieve the most competitive setup.

Lap time simulators in use in the motorsport industry vary a lot in complexity and capability. At the low end of the spectrum there exists 'steady state' lap time simulators which are often simple and cheap, but have an accuracy of around 10-20%. At the upper end of the spectrum are full transient lap time simulators which are often bespoke and very detailed, which can have an accuracy of 0.5%. Recent improvements in CPU power and simulation tools has seen an increase in full transient simulation in industry and academia. Full transient simulators allow all systems of the vehicle to be modelled and optimised, which gives designers a holistic insight into the racing package as a whole.

2 Aims of Project

The objective of the lap time simulator is to develop a tool which allows the UGRacing Formula Student team to make justified concept decisions early in the design process, and to allow the vehicle setup to be optimised after it has been manufactured. The main project objectives are as follows:

- Research the current Lap Time Simulators which are used in industry.
- Develop a transient vehicle model in Simulink which represents and links the various subsystems in the current Formula Student race car.
- Develop a lap time simulator which combines the vehicle model, a driver model and a track model
- Validate the model by comparing the simulation outputs to telemetry data collected from testing
- Carry out concept studies in an attempt to find the optimum vehicle concept within UGR's budget and resources
- Document the model to allow future development and understanding

3 Literature Review

3.1 Steady State simulators

When computers were still rather limited by their performance a steady state based simulation was favoured as it is much less computationally intensive. The earliest published example of the steady state approach was developed in 1971 [4]. This is where the simulation is divided into discrete segments – either a corner, accelerating straight or braking straight. Each segment is calculated at a constant acceleration defined by the vehicle g-g diagram [5].

The maximum corner speed at each corner is calculated by calculating the maximum lateral acceleration from the tyre data. Typically the tyre data is collected by a company such as Calspan [6] and provided as raw data, for example for Formula Student tyre data is provided by the TTC [7]. After the raw data is gathered it is typically converted into a semi empirical model such as the Pacejka magic formula tyre model [8]. The model assumes that the vehicle negotiates the corner at the peak friction coefficient of tyres and at a constant speed.

From here the vehicle model accelerates at a constant driven acceleration (determined by engine power, aero and tyres) up until the apex of the next corner. The model is then 'integrated backwards' from that apex using a constant braking deceleration. Where the lap distance of the driven acceleration and braking deceleration are equal this is determined as the braking point. The diagram below summarises this effectively - obtained from the 'Optimum Lap' software documentation [9].

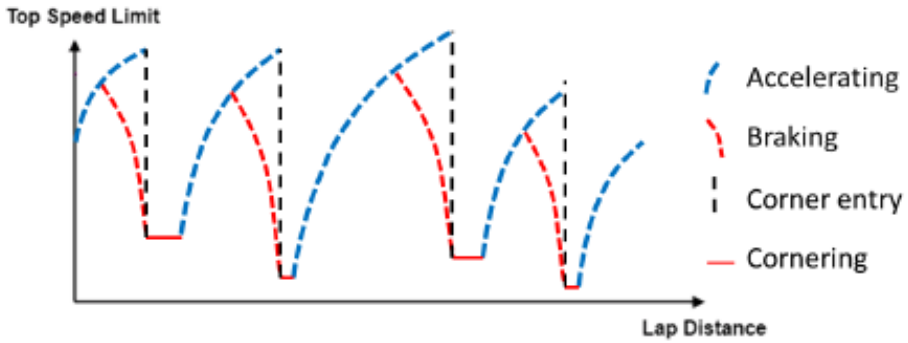


Figure 1: Steady state calculation of braking, diagram from Optimum G [9]

The steady state simulators are limited by the constant acceleration assumption. Any effect due to suspension or load transfer is not taken into account, for example roll, pitch, yaw and damping effects. This method also assumes that the driver has found the fastest racing line and does not vary from it.

3.2 Quasi-static Simulators

Quasi-static simulators are largely similar to the Steady state method described above accept each corner is treated as a series of smaller corners of varying radius and thus varying maximum allowable speed [10]. This effectively allows the model to model a corner of varying radius and allows the vehicle to accelerate or decelerate at the same time as cornering. Some quasi-static simulators also include a load transfer approximation and load sensitive tyre model.

The vast majority of commercially available LTS packages are Quasi-Static Simulators as they can achieve an adequate level of accuracy while maintaining low execution times, which is desirable for track testing purposes.

One of the more popular free Quasi-State lap time simulators is Optimum Lap, produced by Optimum G [9]. Optimum lap quotes that the results obtained are typically around 10% of the logged data, they also highlight that the results should be used for comparison and not

correlation. A point mass model is utilised, which uses a simple aerodynamics model, a linear tyre model, a linear braking model and a basic engine map to calculate the accelerations. The model is limited as it does not account for weight transfer, it doesn't use a real tyre model, it doesn't account for yawing in corners and it assumes the track is perfectly flat.

On the upper end of quasi-static simulators is the LTS, produced by Milliken Research Associates [11]. This model utilises a four wheel model with an approximated load transfer implementation. It also makes uses of a tyre lookup table and 3 dimensional track model as well as an aerodynamics model that varies with ride height. Another unique feature is a scalable driver which can be used to make the simulation 'non-perfect' by a given factor. Although this model is reasonably sophisticated it still has the same limitations as all quasi-static models – there is a constant acceleration assumption and no transient suspension effects.

3.3 Transient Simulators

Transient simulators use a more conventional modelling approach – the vehicle is modelled as a continuously integrated dynamic system. This allows load transfer, suspension, inertia and damping effects to be modelled, and allows more detailed tyre and aerodynamic models to be implemented. The vehicle model is then controlled by some form of driver controller which is coupled to a track model. The driver model can be controlled discretely, but more typically a parameter optimisation routine is used to find the best driver control inputs for a given track and conditions.

Transient vehicle simulators have been used in the general automotive industry for a long time for engineering analysis, but they are typically un-commercialised. Traditionally transient Lap Time Simulators were much less commonly found in the motorsport industry due to their long execution time, complexity and expense. Formula One teams make use of in-house or privately developed transient simulators, such as the Tag Heuer Vehicle data

package [12], because the limited testing time [3] has forced them to make use of much more capable and accurate simulators in the design phase. These packages however cannot be accessed by outside users due to their large expense and confidentiality.

RaceSim, produced by DATAS Ltd [13] is an example of a lower end commercialised lap time simulator. It should be noted that the software has two available modes – quasi-static and transient, just the transient mode will be discussed here. RaceSim utilises a 15 mass dynamic vehicle model with inertias, un-sprung and sprung masses, and non-linear spring suspension. A Pacejka Tyre model [8] is used along with tyre lag. No detailed description of the controller is given, but it does state that an optimisation routine is used. Judging by the relatively high price point of this software it is expected that it likely attains a high level of accuracy.

ADAMS Racecar, produced by MSC Software Corporation [14] uses a multi-body approach to model the vehicle. The multi-body has 53 DOF, full suspension model, and a detailed Pacejka tyre model [8]. Once the vehicle model has been created it can be used by any one of the ADAMS suite of automotive software, of particular interest is the lap time simulation software. Instead of using an optimisation routine to control the vehicle, Adams maps the predicted 'ideal' racing line and the driver model attempts to follow it. This software seems to be resource intensive and complex to set up, but seems to attain a high level of accuracy.

3.4 Vehicle Models Overview

The vehicle models used in steady state lap time simulators are effectively point mass representations which have two DOF; lateral and longitudinal acceleration. This model does not represent yawing behaviour of the vehicle, or the effect of individual tyres.

Milliken [5] showed that the point mass representation can be extended to a 'bicycle' model which models the front and rear axles as single wheels. This model can effectively

represent the yawing of the vehicle in order to calculate tyre slip angles and steering characteristics. This model however cannot be used to calculate load transfer effects.

The bicycle model can be expanded further to 4 wheels to form the 'Two Track' model. The two track model is 3DOF and can be used to find the slip angles of all 4 wheels. Load transfer can be modelled using a quasi-static suspension approximation. Two track models are the simplest model which can typically used in a full transient LTS.

The two track 3DOF model can now be expanded to any number of degrees of freedom to represent the sprung masses, spinning wheel masses, changing fuel mass etc. Ellis [15] represented the vehicle suspension system by splitting the vehicle into two unsprung masses and one sprung mass. This allowed to the model to affectively represent the effect of vehicle roll and pitch.

3.5 Controllers Overview

One of the simplest vehicle control methods is known Craig Reynolds Steering [16]. Here the vehicle 'looks ahead' on its current path distance depending upon its current speed. From here the perpendicular distance to the track is calculated and used to apply a steering command to the vehicle with the aim of approaching the track.

The Craig Reynolds method was expanded by Casanova [17] to use an optical lever. This effectively defines a 'set' of points ahead of the vehicle in which the distance to the track are calculated. The weightings of each distance are varied to achieve the most optimal steering characteristics.

Optimal preview methods can also be implemented for vehicle control. MacAdam [18] uses an optimal single-point preview control model to represent driver steering control behaviour. This adjusts the control behaviour based upon vehicle kinematics, and accounts for driver delay time. This model is accurate but complex to setup and slow to execute.

3.6 Conclusion of Literary Review

The following conclusions have been drawn after reviewing the literature on lap time simulators.

- Steady State Simulators are very simple, easy to implement and fast to execute. Their constant acceleration assumption, however, makes them very limited and lacking in accuracy
- Quasi-Static Simulators address some of the accuracy issues of steady state simulators by including partial transient models such as load transfer, while maintaining fast execution.
- Transient simulators are seeing a rise in use due to improving computing power, and desire for more capable packages in the industry. Transient simulators utilise continuous dynamic vehicle model controlled by a driver model
- Transient models have the most accuracy and versatility, but are computationally intensive.
- Transient models can be readily optimised using conventional optimisation techniques.
- A 3DOF two track vehicle model with quasi-static load transfer is required at the very least to represent a suspension system, more DOFs can be added for increased accuracy.

4 Methodology

4.1 Vehicle Model Design

The first major architecture decision is whether the model is calculated steady state, quasi-statically or dynamically. The steady state and quasi-static approaches are less computationally intensive, but do not take into account the effect of roll, pitch, yaw or load transfer. Dynamic simulation addresses the shortcomings of the steady state and quasi-static simulations and allows the addition or expansion of subsystems in the future such as suspension models. It was decided that a dynamic model would best suite the needs and skills of the Formula Student team.

The model was designed in Simulink and MATLAB mainly due to the author's familiarity with the software, but also because of its rising use within universities, which will ensure continued development within the Formula Student team. Formula Student receives support from MathWorks in the form of the complimentary powertrain block set [19] and advice through the Formula Student racing lounge [20].

The top level of the vehicle model ('*UGR_LTS*', see appendix A.1) has been divided into five subsystems – *Controller, PowerTrain, DriveTrain, Tyres + Brakes + Steering* and *VehicleDynamics* (See Figure 2 below). The logical flow from driver input to vehicle response is summarised as:

1. Control commands are issued from the *Controller* model, for instance steering angle and brake force
2. The *PowerTrain* model receives its torque command and uses its engine model to produce an actual engine torque
3. The *DriveTrain* model takes the engine torque and commanded gear and calculates the drive torque to each wheel
4. The *Tyre + Brakes + Steering* model uses the drive torque, braking torque and steering angle to calculate the force that each tyre produces using a tyre model.

5. The *Vehicle Dynamics* model uses the tyre forces to calculate the vehicle translational and rotational acceleration, velocity etc.

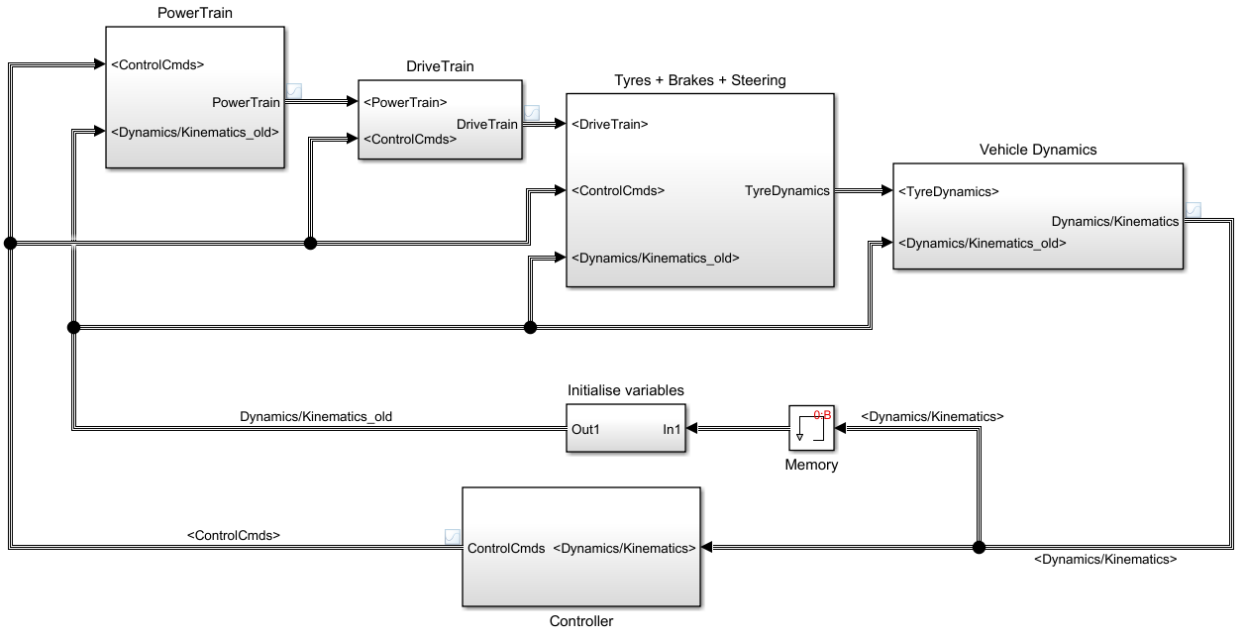


Figure 2: Top Level of *UGR_LTS* Simulink model

The vehicle parameters, track parameters and initial conditions are setup outside the model prior to running (see Appendix B for examples). This allows the user to simply change the setup files to run different scenarios, without having to modify the LTS.

The model runs using a time step of 0.001 seconds, this step size was found through testing to produce a stable output while maintaining fast execution. Currently the ODE3 fixed step solver is being used, but the model can also run on variable step and stiff solvers. Some experimentation is still required to decide upon the final solver to use.

4.2 Powertrain Model

The powertrain model (appendix A.2) uses the 'Mapped SI Engine' [21] model to calculate engine torque from a torque command and engine speed input. The torque data which is used in the engine model was collected primarily from dynamometer testing of the UGRacing's Honda CBR engine (appendix C.1) and some gaps in the data were supplemented from Ricardo WAVE [22] simulations for the Formula Student engine. The torque data in the engine model is stored as the following lookup table:

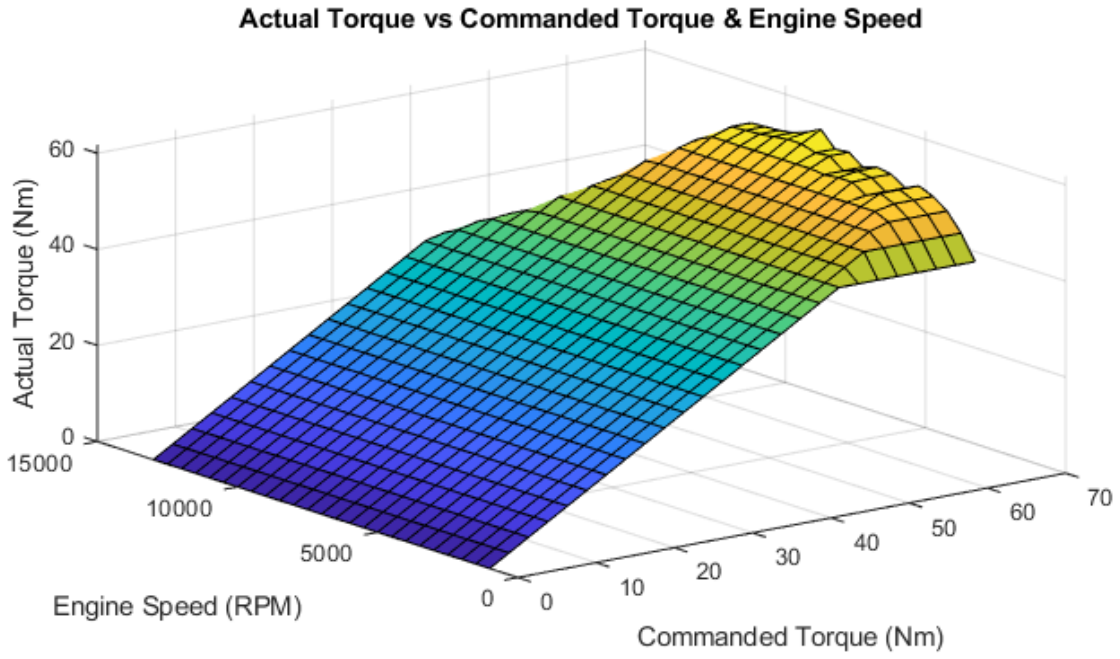


Figure 3: Engine Torque lookup table

Inside the Powertrain model the current Engine speed is calculated from the wheel speed and the current gear ratio. The engine speed and commanded torque are then fed into the engine model. The engine model calculates the actual torque which can be achieved using the engine torque look up table. In some scenarios the actual torque cannot match the commanded torque, for example when the engine speed reaches its maximum its torque drops off (over rev).

4.3 Drivetrain Model

The engine torque data passes into the drivetrain model (appendix A.3), which multiplies the torque by the gear ratio. The MATLAB Limited Slip Differential Model [23] is used to distribute the torque to each wheel depending upon the axle forces present from the tyres etc. An estimated drivetrain efficiency gain is also applied to the torque values.

4.4 Braking Model

The braking system (appendix A.4.1) is a simple hydraulic system which is actuated by the driver. The driver force (F_{driver}) is input via the braking pedal which is multiplied by a pedal force ratio (PR). The force is then distributed to two hydraulic master cylinders using a balance bar, the distribution depends on the bias ratio (BR). The master cylinders pressurise the front and rear hydraulic circuits, which then actuates the brake calliper pistons. The brake calliper pistons provide a normal force onto the brake pads which exert a frictional force onto the brake disc and thus create a braking torque if the wheel is rotating.

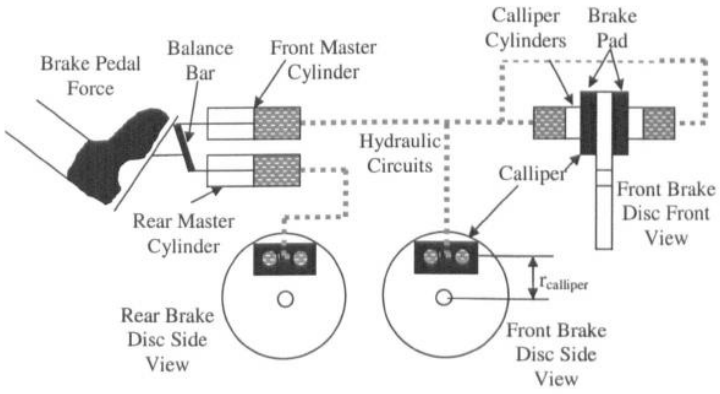


Figure 4: Hydraulic disc brake system, diagram from [24]

Shown below is the general braking torque calculation:

$$T_{brake} = F_{driver}\left(\frac{PR \cdot BR \cdot A_{calliper} \cdot \mu_{pad}}{A_{master}} \cdot r_{calliper}\right) \quad (1)$$

The braking command signal is a normalised value between 0 and 1. A command of 0 means no pedal force is applied and a command of 1 means maximum pedal force is applied. The maximum pedal force was found empirically to be 370N, by measuring the comfortable maximum braking force of UGR's drivers. The drivers were harnessed into the vehicle seat, where they pressed against a set of scales.

4.5 Steering Model

The steering system (appendix A.4.2) is a mechanical system which converts steering wheel angle displacement into wheel angle displacement via a rack and pinion and linkage system. It was found that the linkage system was difficult and complex to model analytically due to the 3d geometry. Instead it was decided to model the system empirically as it is more accurate and simpler.

The left and right wheel angles were measured for a range of steering wheel angles (appendix C.2). A lookup table was generated for the left and right wheels:

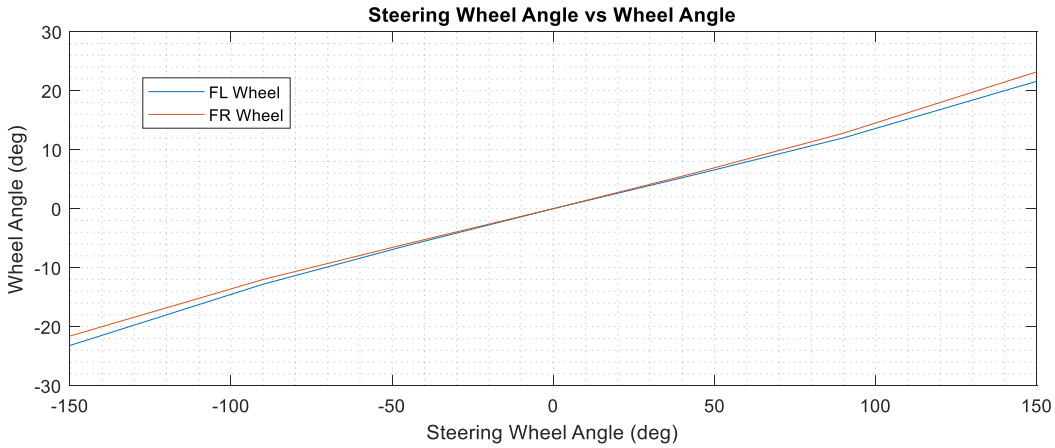


Figure 5: Steering system lookup table

4.6 Tyre Model

The tyre model (appendix A.4.3) was constructed using empirical data because the high non-linearity of tyres makes then very difficult to model accurately using analytical techniques. The tyre data was gathered by the tyre testing consortium [7] using a Calspan [6] belt testing rig and supplied as raw data (method can be seen in appendix D.1).

Pacejka 96 [8] curve fitting was used to obtain the parameterisation coefficients (appendix D.2) for the so called 'Magic Formula'. The Magic Formula was shown by Pacejka to closely model the tyre force with the following relationship:

$$F = D\sin[C\tan^{-1}\{Bx - E(Bx - \tan^{-1}Bx)\}] - S_v \quad (2)$$

For lateral force $x = \alpha + S_H$, for longitudinal force $x = \kappa + S_H$ (3)

Where:

- x = Input variable
- B = Stiffness factor, where $B = f(F_z, \gamma)$
- C = Shape factor
- D = Peak value, where $B = g(F_z, \gamma)$
- E = Curvature factor, where $E = h(F_z, \gamma, x)$
- S_H = Horizontal shift
- S_V = Vertical Shift

4.6.1 Lateral Tyre Force (Fy)

To calculate the tyre lateral force (Fy) the slip angle (α) of the wheel must first be calculated. The slip angle is the angle at which the road approaches the wheel x-axis:

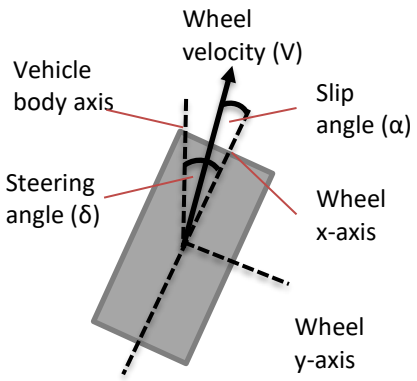


Figure 6: Tyre slip angle (α)

The slip angle is calculated from the wheel velocity and the steering angle:

$$\alpha = \delta - \tan\left(V_y / V_x\right) \quad (4)$$

The tyre slip angle (α), vertical load (F_z) and inclination angle (γ) are then input to the Magic Formula to calculate the lateral force (F_y) produced by the tyre. Shown below is the lateral force produced at various slip angles for a range of vertical load values:

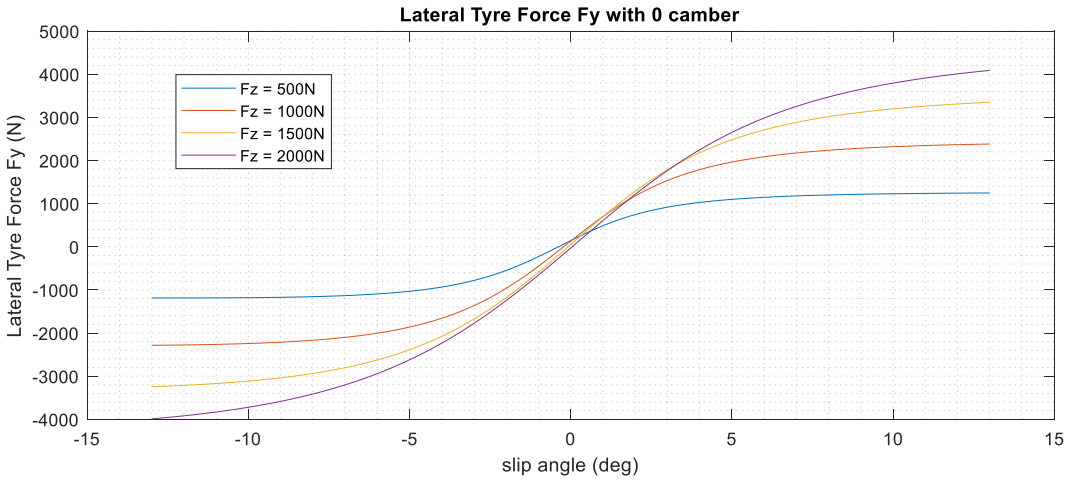


Figure 7: Lateral tyre force at zero camber

The lateral tyre model can be found in appendix A.4.3.2

4.6.2 Longitudinal Tyre Force (Fx)

To calculate the tyre longitudinal force (Fx) the slip ratio (κ) of the wheel must first be calculated.

To calculate the slip ratio (κ) of each wheel we must consider both the angular rate (ω) of the wheel and the forward velocity (V_x) of wheel with respect to the road.

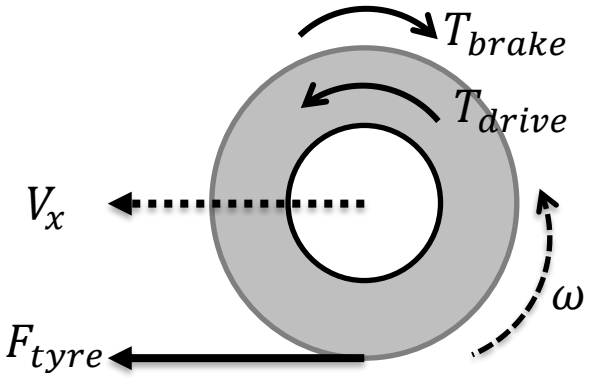


Figure 8: Torque and Velocity of driven wheel

The angular rate is calculated by integrating the angular acceleration produced by the torque imparted on the wheel by the drivetrain, brakes and reaction to tyre force:

$$\omega = \int \frac{T_{drive} - T_{brake} - T_{tyre}}{I_{wheel}} \, dt \qquad (5)$$

The tyre reaction force (F_{tyre}) is typically taken as the longitudinal tyre force (F_x) which was calculated in the previous time step of the simulation.

The slip ratio (κ) is then found by finding the ratio between the spinning wheel velocity at the contact patch and the wheel velocity with respect to the ground:

$$\kappa = \frac{\omega \times r_{wheel}}{V_x} \qquad (6)$$

For κ ≈ 1 wheel is in free spin, for κ ≈ -1 wheel is in locked braking, max force typically occurs at κ ≈ ±0.15.

The tyre slip ratio (κ), vertical load (F_z) and inclination angle (γ) are then input to the Magic Formula to calculate the longitudinal force (F_x) produced by the tyre. Shown below is the longitudinal force produced at various slip ratios for a range of vertical load values:

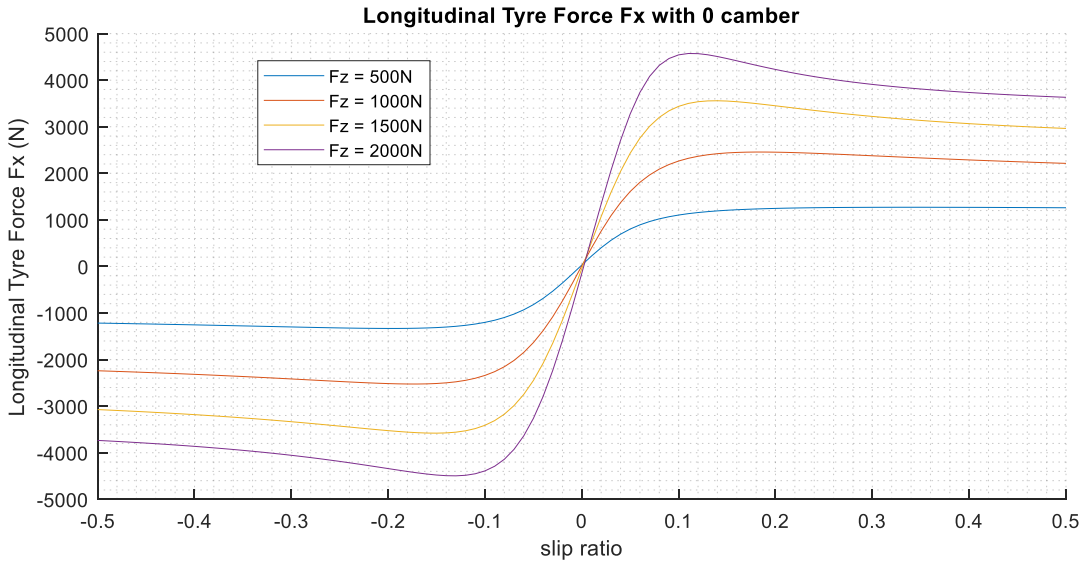


Figure 9: Longitudinal tyre force at zero camber.

The longitudinal tyre model can be found in appendix A.4.3.1

4.7 Vehicle Dynamics Model

The vehicle dynamics model (appendix A.5) is based off of a two track model. This is effectively a modified 'bicycle model' [5] which has been extended to have 4 wheels. The two track model uses the longitudinal and lateral forces of all 4 tyres and the aerodynamic forces to calculate the longitudinal, lateral and yaw accelerations exerted on the centre of gravity of the vehicle.

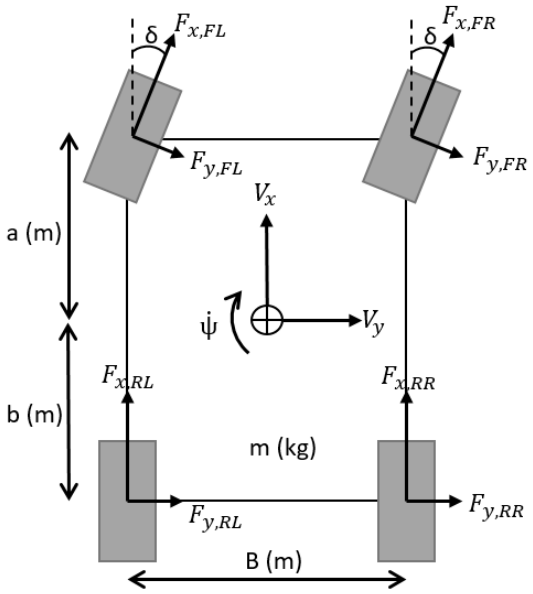


Figure 10: Two Track model

4.7.1 CG Dynamics Model

The centre of gravity acceleration in the longitudinal, lateral and yaw directions are found by combining the external forces produced by the tyres and aerodynamic drag.

$$\frac{d}{dt}v_x(t) = \frac{1}{m}(cos\delta * (F_{x,FL} + F_{x,FR}) - sin\delta * (F_{y,FL} + F_{y,FR}) + (F_{x,RL} + F_{x,RR}) - \frac{1}{2}\rho C_D A v_x^2) \quad (7)$$

$$\frac{d}{dt}v_y(t) = \frac{1}{m}(sin\delta * (F_{x,FL} + F_{x,FR}) + cos\delta * (F_{y,FL} + F_{y,FR}) + (F_{x,RL} + F_{x,RR})) \quad (8)$$

$$\frac{d}{dt}\dot{\psi}(t) = \frac{1}{I_{zz}}(a\left(sin\delta\left(F_{x,FL} + F_{x,FR}\right) + cos\delta\left(F_{y,FL} + F_{y,FR}\right)\right) - b\left(F_{x,RL} + F_{x,RR}\right) + \frac{B}{2}(cos\delta\left(F_{x,FL} - F_{x,FR}\right) + sin\delta\left(F_{y,FL} - F_{y,FR}\right) + \left(F_{x,RL} - F_{x,RR}\right))) \quad (9)$$

From here the velocity and displacement of the Centre of Gravity (Body Axes) is found by integrating (see appendix A.5.1).

Longitudinal displacement, m = $S_x(t) = \iint \frac{d}{dt} v_x(t)\, dtdt$ (10)

Lateral displacement, m = $S_y(t) = \iint \frac{d}{dt} v_y(t)\, dtdt$ (11)

Yaw angle, rad = $\psi(t) = \iint \frac{d}{dt} \dot{\psi}(t)\, dtdt$ (12)

4.7.2 Inertial Model

The displacements of the centre of gravity (Body Axes) are transformed to inertial axes for later use in the guidance model. The initial inertial position and yaw angle are determined by the start position of the track model.

$$S_{x,inert}(t) = S_x \sin(\psi) + S_y \cos(\psi) \quad (13)$$

$$S_{y,inert}(t) = S_x \cos(\psi) - S_y \sin(\psi) \quad (14)$$

4.7.3 Wheel Model

The wheel model (appendix A.5.2) calculates the load on each wheel due to the load transfer and aerodynamic downforce of the car, and it calculates the velocity of each wheel with respect to the ground.

The load on each tyre is determined by the static load at rest, the longitudinal load transfer (change in load in x direction from braking or accelerating), lateral load transfer (change in load in y direction from cornering) and aerodynamic downforce. It should be noted that the calculation of longitudinal and lateral load transfer are omitted here as they are determined by the suspension model which is still under development. The load on the front left tyre is calculated as:

$$F_{z,FL} = \frac{1}{2}\left(\frac{b}{a+b}\left((mg) + \frac{1}{2}\rho C_L A v_x^2\right) + F_{Lat\ Tran} + F_{Lon\ Tran}\right) \quad (15)$$

The x and y velocities of each wheel are calculated using the vehicle centre of gravity velocity and yaw rate, for example the velocity of the front left wheel is calculated as:

$$v_{x,FL} = v_x + \left(\frac{1}{2}B \times \dot{\psi}(t)\right) \quad (16)$$

$$v_{y,FL} = v_y + \left(a \times \dot{\psi}(t)\right) \quad (17)$$

4.8 Track Model

The track models were generated by collecting coordinates from GPS and/or from satellite imagery of the track. GPS values were collected from telemetry data from a UGR18 testing session at Forrestburn hill climb circuit [25]. The raw GPS is rather low resolution and seemed to drift between laps (as seen in Figure 11 below).

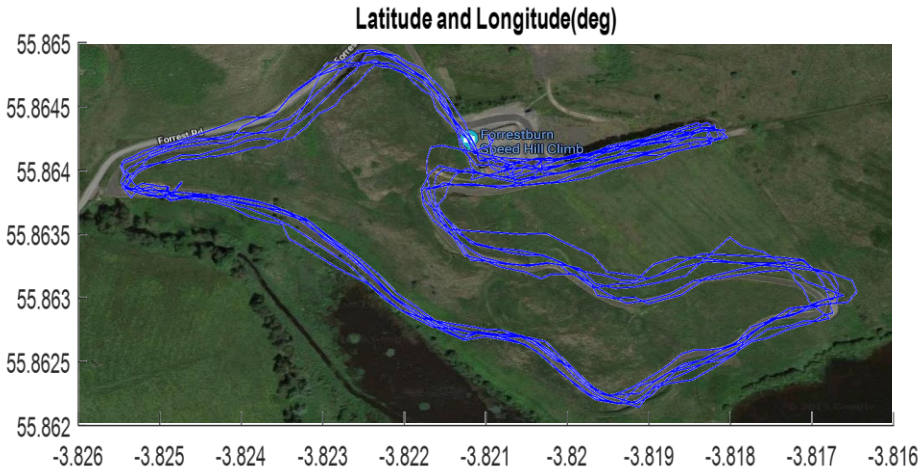


Figure 11: GPS Latitude and Longitude collected from testing at Forrestburn hill climb

It was decided that track points should be collected from satellite imaging instead of GPS, then scaled using two points with a known distance between them. A google maps snapshot of the track was taken, which was converted to a binary image using the MATLAB

'Color Thresholder' app. From here the binary image was converted to a standard plot and scaled using a known distance (straight line between start and finish line):

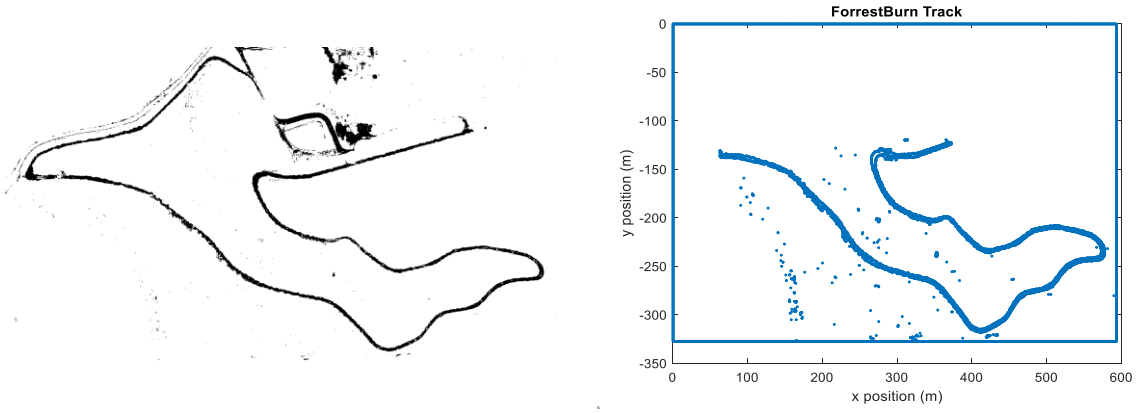


Figure 12: Conversion from track binary file to scaled track plot

The track plot is then processed using the 'Driving Scenario Designer' Matlab App [26]. The app is used to clean up any erroneous data points and to curve fit a track trajectory using a set of waypoints. Between the way points the track is represented by a series of straight segments 1m long (or whatever resolution the user specifies).

The driving scenario app assigns an initial target speed profile for the vehicle to attempt to reach at each waypoint on the track. The initial speed profile is a first approximation based on expected vehicle performance, usually determined from previous test results. The initial speed profile will later be updated during the optimisation routine in an attempt to minimise the lap time.

4.9 Guidance model

The guidance model (appendix A.6) calculates the lateral error from an optical lever [17] (see Figure 13 below) projected in front of the vehicle at a set of specified preview distances (d_i). The corresponding lateral errors (e_i) at these distances is calculated by interpolating the lateral intersection with the track, within a specified radius(R) of the

point. A radius of R = 3 was chosen as it was large enough to not lose range on the testing tracks and small enough to not require excessive computational time.

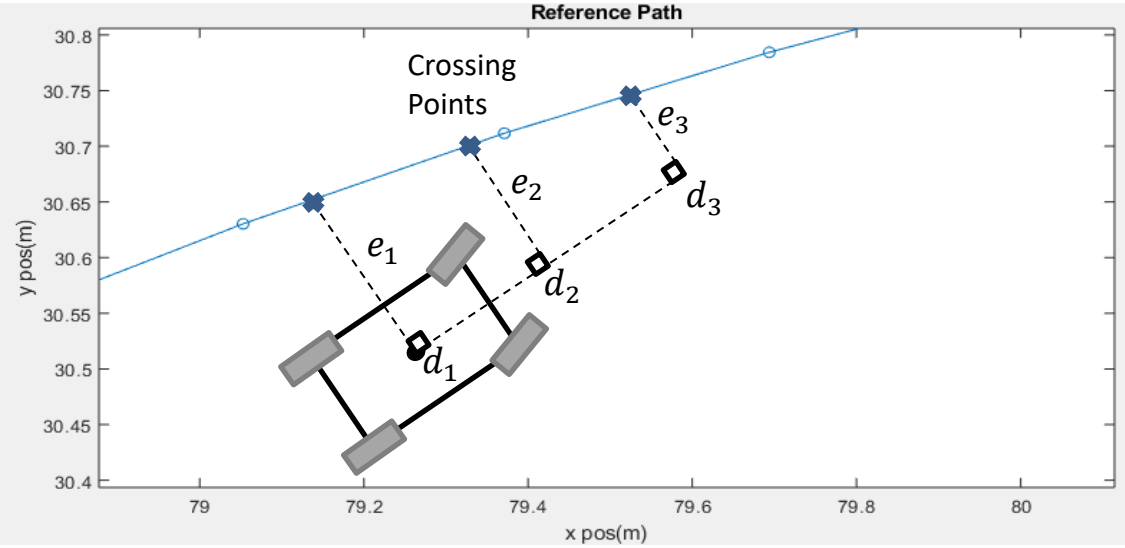


Figure 13: Guidance model calculation of lateral error

The target speed at each distance (d_i) is calculated by interpolating the track speed profile at each of the crossing points, as seen in Figure 13.

The total lateral error (e_{total}) is calculated by summing the lateral errors (e_i) at each preview distance with a weighting (w_i) applied to each lateral error. The lateral errors are stored in vector e and the weightings are stored in vector w.

$$e_{total} = e \cdot w = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} \quad (18)$$

Initially only the lateral error at the CG was considered, but testing found that the model was too slow to react in tight corners.

The weightings have initially been determined empirically in conjunction with the lateral controller model (section 4.10).

4.10 Lateral Controller

The lateral controller (appendix A.7) uses a PI controller to control the steering system in an attempt to minimise the total lateral error (e_{total}). The steering wheel angle (θ) is calculated using a proportional gain (K_P) and integral gain (K_I):

$$\theta = K_p e_{total} + K_I \int e_{total} dt \quad (19)$$

Tuning of Kp and Ki was initially carried out using the Ziegler Nichols oscillation technique on a straight road with an initial offset. The gains were approximately Kp = 160 and Ki = 30, however it was found that the optical lever weightings from (18) and the preview distances effect the controller performance considerably.

Work is underway to select the optical lever weightings and gains using an optimisation routine (appendix E.2) based upon the response time, accumulated lateral error and over shoot characteristics. A slalom track was created which mimics a typical Formula Student track segment and the model was run at a constant speed through the slalom. Currently no useful results have been obtained from this.

4.11 Longitudinal Controller

The longitudinal driver model (appendix A.8) from MathWorks [27] is used to control the longitudinal velocity of the vehicle. It was decided to use this model mainly as it has the capability for a future gear shifting implementation.

The model calculates the longitudinal error between the current longitudinal velocity and the target speed from the guidance model. A PI controller generates normalised engine torque and braking commands to attempt to minimise the longitudinal error. Tracking windup is used to deal with saturation in the powertrain and braking models.

The longitudinal controller tuning was initially carried out using the Ziegler Nichols oscillation technique on a straight road with an initial offset (similar to the lateral

controller), which obtained Kp = 15 and Ki = 1. However, due to the added complexity of the tracking windup and feed-forward gains it was decided to attempt to find the gains using an optimisation technique similar to the lateral controller above. Currently this work is underway and the gains have not been finalised.

4.12 Optimisation

The optimisation implementation is designed to optimise the simulation parameters using the in-built MATLAB fmincon [28] function for constrained non-linear functions. The algorithm currently uses the gradient based interior-point algorithm. Currently the objective function is the lap time produced by the model, but this could be any number of performance parameters in the future.

Initial work has attempted to optimise the track target speed profile (see appendix E.1 for the code) to minimise the lap time. In the 'Track Model' implementation the initial target speed profile at each track waypoint was estimated by the user. The waypoint target speeds are the optimisation variables, and upper and lower speed bounds are declared as the inequality constraints:

$$\underline{V}^*_{waypoints} = \min LTS\left(\underline{V}_{waypoints}\right) \quad (20)$$
$$Subject\ to: V_{min} \leq \underline{V}_{waypoints} \leq V_{max}$$

Where:

$\underline{V}_{waypoints}$	= Waypoint speed profile vector, m/s
$\underline{V}^*_{waypoints}$	= Optimum waypoint speed profile vector, m/s
LTS	= Lap Time function, s
V_{min}	= Lower speed profile bound, m/s
V_{max}	= Upper speed profile bound, m/s

Currently only the waypoint speed profile has been optimised successfully, see section 5.3.1 for results. Work is underway to optimise the controller gains (see appendix E.2 for the code) to minimise the accumulated absolute lateral error.

In theory the optimisation tool can be used to optimise any selected input parameters. In most cases the objective function will be the Lap Time, however this can be any number of performance parameters – for example fuel efficiency.

5 Results & Discussion

It was decided to split the results up into three sections - vehicle model testing, lap time simulator testing and optimisation testing. This was to ensure that the model was verified at each major milestone – rather than at the end of development.

5.1 Vehicle Model Testing

The vehicle model testing is intended to test that the vehicle model responds to inputs in the same way that the real vehicle does. These tests were carried out before development began on the controller models to ensure that the model being used for lap simulation was actually representative of the Formula Student car.

Unfortunately only a small amount of subsystem testing data was available for the UGR18 car. No steering position, engine speed/torque or brake pressure data was available so individual subsystems could not be isolated in test harnesses. Only accelerometer, velocity and GPS data was available from the tests. It was decided to compare just peak acceleration values between the test data and the model as this gives a good indication of a vehicles maximum performance.

5.1.1 Vehicle Longitudinal Acceleration Comparison

The maximum longitudinal acceleration of the model was tested by commanding maximum engine torque on a straight track from zero m/s. This was compared to testing data of vehicle launch from standstill recorded from the UGR18 Forrestburn testing session [25]:

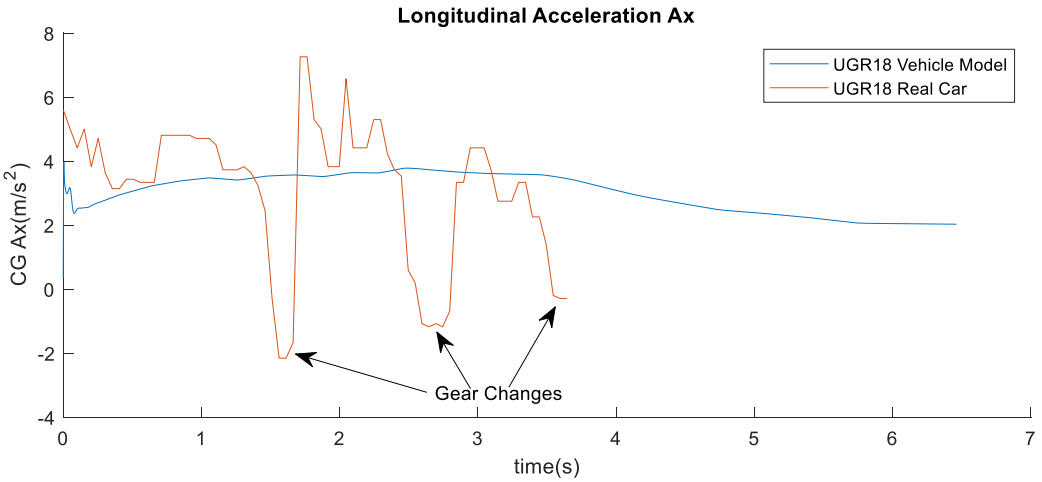


Figure 14: Longitudinal Acceleration comparison of vehicle model and real car

A peak longitudinal acceleration of 3.8m/s^2 was found for the vehicle model. The peak longitudinal acceleration of the real car was found to be around 7.6m/s^2 from a launching start – note that at around 1.2 seconds the real car begins shifting gears so the accelaeration drops.

The real vehicle was found to have a consistently higher longitudial acceleration in all of the comparisons. There are a number of known differences between the model which can be attributed to these dicrepencies. The vehicle model currently doesn't have any load transfer – in real life the rear wheels have load transferred onto them in acceleration, which them to gain more grip and accelerate harder.

Another difference is that the model does not have a clutch implemented, so the engine cannot launch – in reality the engine would be spun up to a high RPM and the clutch would be 'dropped', allowing a very high impulse to be used for launching the vehicle.

The vehicle model's longitudinal acceleration was deemed accurate enought to continue with the guidance and controller models, but will require the implementation of load transfer and a clutch in the future to improve accracy.

5.1.2 Vehicle Longitudinal Deceleration Comparison

The maximum longitudinal decceleration was measured by braking at peak pedal force from a constant speed of 20m/s. This was tested with the model and with the real life UGR18 vehicle at the Forrestburn test session [25]:

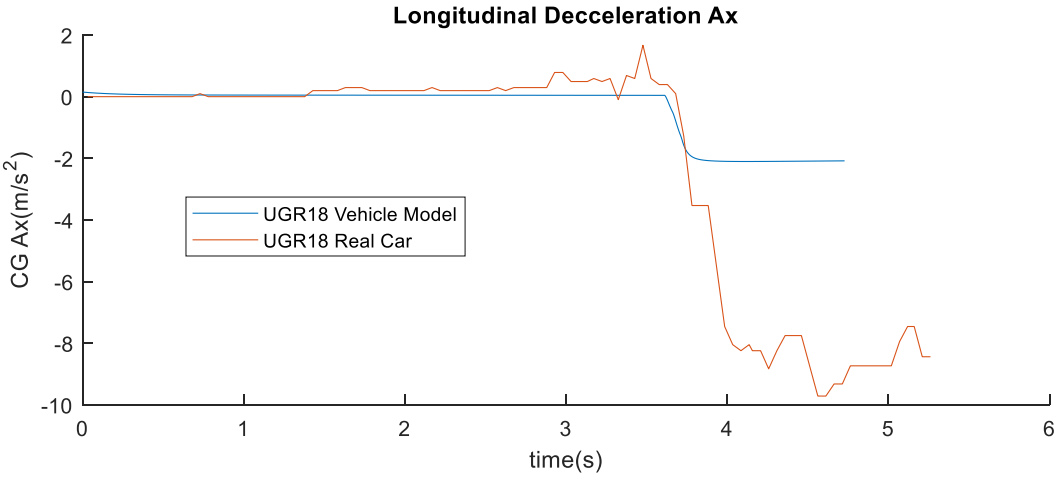


Figure 15: Longitudinal Deceleration comparison of vehicle model and real car

A peak longitudinal decceleration of -2.1m/s^2 was found for the vehicle model under braking. The peak longitudinal decceleration of the real car was found to be around -9.7m/s^2.

The real vehicle has a much higher braking decceleration than the model. This is thought to be mainly due to the lack of load transfer in the simulation. In reality the load is transferred onto the front wheels under braking, which allows them to get more grip. This means that on UGR18 the front braking system is biased to have a much higher pressure system and larger brakes and discs – which accounts for around 70% of the braking force. In the model the braking system is the same as UGR18 but with no load transfer, therefor the front braking system is only achieving a fraction of its potential real life braking force.

The braking decceleration of the model was rather unrepresentative when compared to the real car. It was however decided to proceed with assembling the vehicle model into the laptime simulator – with the knowledge that the braking will be lacking. This test shows that a load transfer model should be implemented as soon as faesible in the model development.

5.1.3 Vehicle Model Lateral Acceleration Comparison

The maximum lateral acceleration was measured by comparing the lateral acceleration which was obtained from that last corner on the Forrestburn test session [25]. The real vehicle entered this corner at 15.5m/s and exited at 12m/s. A track model of the same corner was developed using the Driving scenario designer (see section 4.8) which replicated the speed profile and track profile observed from the real car.

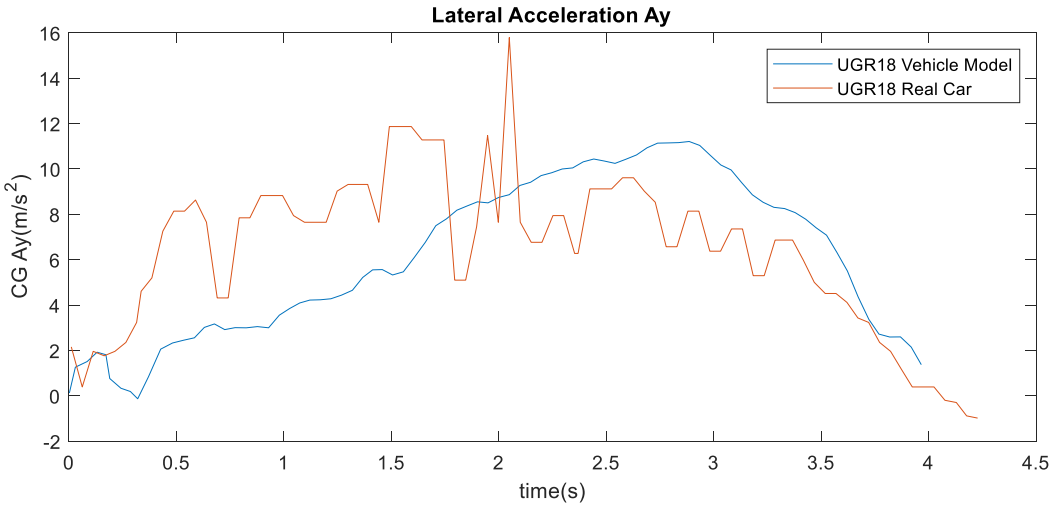


Figure 16: Lateral Acceleration comparison for last corner of Forrestburn hill climb [25]

A peak lateral acceleration of 11.2m/s^2 was found for the vehicle model at this cornering case. The peak lateral acceleration during testing was found to be around 15.8m/s^2 for the real car, although there is a large fluctuation when it reaches the peak (see Figure 16 above).

In general there is a good match between the real car and vehicle model in cornering. The high accuracy is likely due to an empirical based steering system and tyre model. This model could be improved in the future with the inclusion of a load transfer and suspension model in an attempt to simulate the effect of changing suspension geometry and a rolling chassis.

5.2 Lap Time Simulator Testing

The Lap Time Simulator tests are intended to check that the track, guidance and controller models are successfully allowing the vehicle model to follow the track in the same way that the real vehicle follows a track. As mentioned in the vehicle model testing (section 5.1) the testing results available for UGR18 is rather limited, so a full system comparison around a lap was not possible, only the lap times and controller performance was analysed.

5.2.1 Lap Time Simulator – Hill climb testing

The UGR18 Forrestburn hill climb test session (appendix C.4) was used to obtain a track and speed profile for the guidance model. The Lap Time Simulator was then used to try to follow these profiles to ensure that the combined vehicle model and controller models can closely follow the real vehicles performance.

The fastest lap time recorded by the real vehicle was 62.9 seconds. The speed profile for this lap was then input to the Lap Time Simulator as a target speed profile. The LTS was able to follow this trajectory successfully and obtained a lap time of 56.6 seconds, which is an accuracy of 10.2%.

To investigate the difference in lap times the trajectory of the real car and LTS were compared:

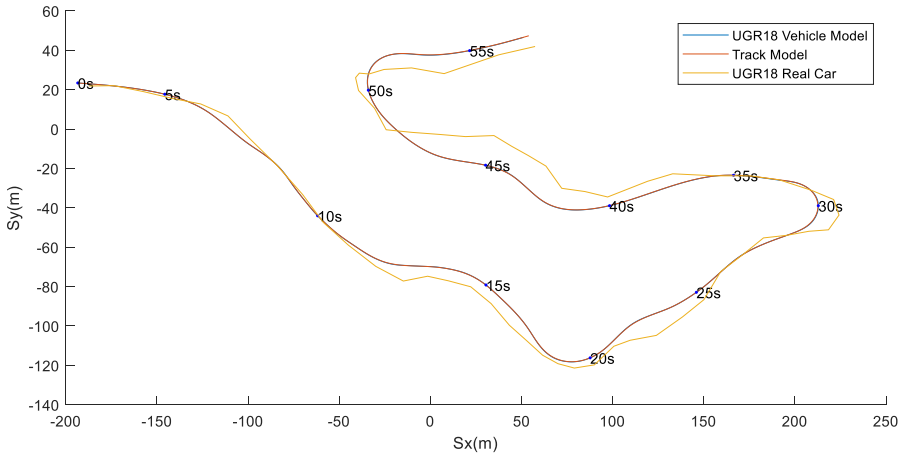


Figure 17: Forrestburn Test comparison between UGR18 real car and model

Note: The UGR18 Vehicle model and Track model are closely overlaid in the above graph.

The total distance covered by the real car was recorded as 979m whereas the total distance recorded by the LTS was 854m – this likely explains the difference in lap times obtained. The most likely cause may have been that in the track model the hill gradient was not accounted for. Another possible cause of the distance difference was that there was a discrepancy between the model trajectory and real recorded trajectory (as seen above), upon further inspection it was found that the GPS data collected was very inaccurate and deviated a lot over adjacent laps. Future addition of gradient to the track model and the use of more accurate GPS data may improve lap time correlation.

The Vehicle Model was observed to follow the track model very closely – with a maximum lateral error of 0.24m deviation (see Figure 18) from the track. The longitudinal controller was also observed to follow the speed profile closely. This suggests that both controllers are responsive and well tuned.

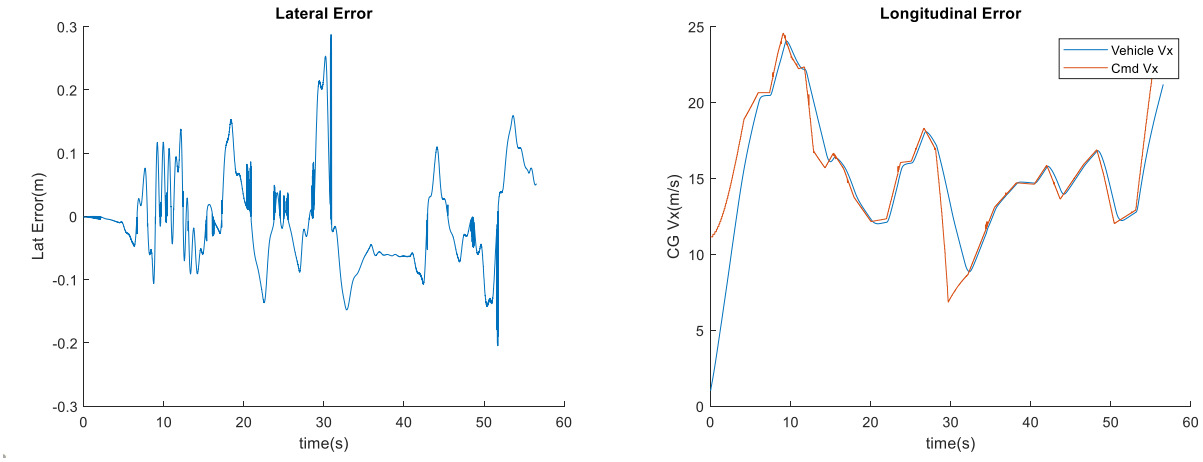


Figure 18: LTS Lateral and Longitudinal controller error recorded on Forrestburn track

5.2.2 Lap Time Simulator – FSUK18 Endurance event testing

The endurance event is the main dynamic event at the FSUK competition in Silverstone. It is a 1km technical track which must be circuited 22 times. Unfortunately at the 2018 competition the only data that was available was lap times (appendix C.3) as a software issue caused the vehicle dynamics data to be at too low resolution.

The real car, UGR18, achieved an average lap time in the endurance event of 81.35 seconds and average speed of 12.3m/s, as seen in the Official FSUK Competition Results [29].

The initial target speed profile for the LTS test was set as the average speed of 12.3m/s as obtained by the real car, from here the model was tested iteratively. The target speed profile was reduced at any waypoints of the simulated lap that exceeded the maximum lateral error limit of 3m. The target speed profile was increased at any waypoints which had a lateral error less than 0.2m.

The LTS achieved a lap time of 88.71 seconds (accuracy of 8.3% comparing to real car) when using the initial target speed profile for the track. The following trajectory was found for the LTS:

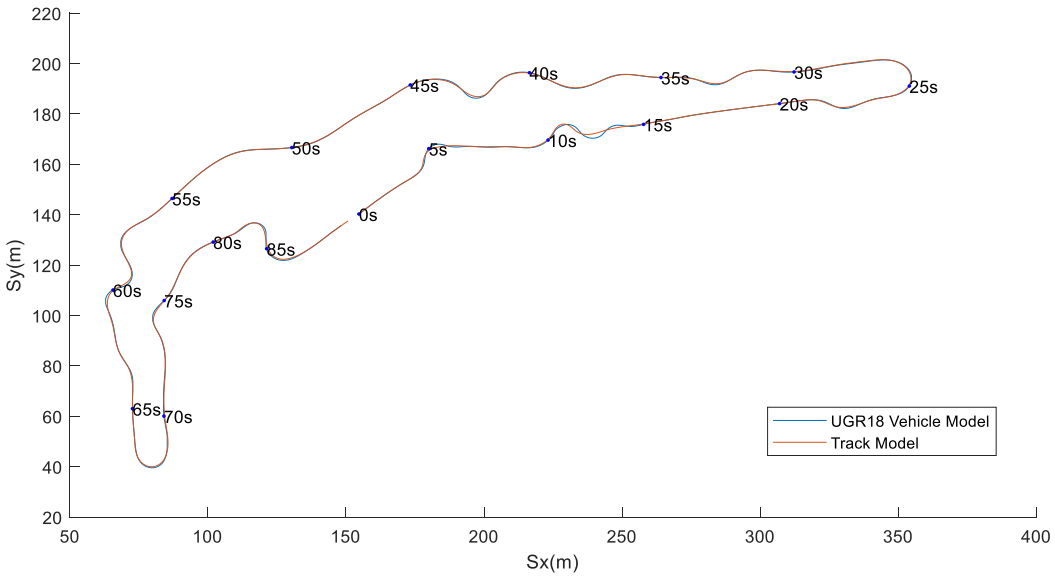


Figure 19: LTS simulation of FSUK 2018 endurance event

It can observed on Figure 19 that the vehicle model follows the track closely for the most part. The model deviates laterally from the track an average of 0.41m, with a peak of 2.03m. The largest lateral errors were found to occur at chicanes, which suggests that the steering controller will need some additional tuning for this track case.

The real car is currently around 7 seconds a lap quicker than the LTS. This is because the speed profile is an initial estimate of the vehicle speed at each track waypoint. This initial speed profile will be used by the optimisation routine (section 4.12), which will later be used to find the fastest target speed profile and in theory be faster than the real car as the vehicle model will be at its maximum performance.

The initial LTS target speed profile test shown above suggests that the LTS is performing roughly similar to the real car and serves as a good starting point in future speed profile optimisation.

5.3 Optimisation testing

The optimisation routine outlined in section 4.12 has only had limited testing. The only testing which has been carried out has been simple speed profile optimisation which minimises the lap time for a straight track. Controller gain optimisation to minimise the accumulated lateral error has also been carried out, but usable results are yet to be obtained.

5.3.1 Speed Profile Optimisation

The optimisation routine was used to find the optimum target speed profile for a straight track with 3 waypoints. A lower bound speed of 1.5m/s and upper bound speed of 20m/s were applied (see Appendix E.1). The following table summarises the optimisation results from that test:

Table 2: Speed profile optimisation results for straight track

Waypoint No.	1	2	3	Lap Time (s)
Initial Target Speed Profile(m/s)	2.00	5.00	10.00	15.42
Optimised Target Speed Profile(m/s)	4.83	15.7	19.04	6.68

The optimisation routine found a local minimum in 8 iterations.

It is observed that the optimisation routine doesn't return the maximum possible target speed profile from the model (upper bound of 20m/s), instead it finds a gradually increasing speed profile. The gradually increased speed profile makes sense when considering the tyre behaviour of the vehicle. If the speed demand is much higher than the current speed of the vehicle then a large amount of torque is demanded from the engine, this in turn would cause the tyre slip ratio to increase beyond its optimum force value (see Figure 9). In the real world this would be seen as a wheel spin.

The optimisation routine has successfully found the torque profile which would provide optimum tyre behaviour and prevent wheel spin. A similar logic to this is applied in launch control and anti-lock braking systems in the automotive industry.

It was attempted to optimise a simple curved track with the intention to expand it to a whole track model. However the results have not successfully converged yet, more work is required on this track case.

6 Conclusions

The main aims set out at the beginning of this project have been met. A transient vehicle model has been developed which represents the UGRacing Formula Student car. This vehicle model was then combined with a track model and controller to simulate lap times around existing racetracks. The individual vehicle subsystems and Lap Time Simulator were then validated against existing UGR18 test data. And finally an initial optimisation routine was developed which successfully optimises the vehicle inputs to minimise the lap time, which can be easily expanded for future analysis.

A literary study at the beginning of the design process revealed that the majority of Lap Time Simulator software available is either steady state or quasi-static simulators. This is due to their simplicity and fast computation time. However, steady state and quasi-static simulators are limited in accuracy and detail by their assumption of constant acceleration. Transient simulators are used less often in industry and often at a much higher technical level. Transient simulators use a more conventional modelling approach and can thus represent more detailed models such as suspension and load transfer, they can also be expanded more readily to include new subsystems or optimisation techniques. The increase in computing power available over recent years has made transient simulators a much more viable option for a LTS.

The vehicle model was designed to rely as much as possible on existing empirical data from the Formula Student car. The powertrain model was based upon data directly obtained from an engine dynamometer. The steering model uses a lookup table which was constructed using geometric test data. The tyre model was a semi-empirical model based upon tyre data collected from tyre bench testing.

The vehicle model was validated by comparing the longitudinal and lateral accelerations to UGR18 test cases. It was found that the lateral acceleration correlates closely to the test

results, which suggests that the steering and lateral tyre models are reasonably accurate. The longitudinal acceleration of the LTS was found to be lower than the UGR18 test results, which was thought to be due to a lack of clutch and load transfer model in the LTS. The longitudinal deceleration of the LTS braking system was found to be much lower than the UGR18 test results, which was thought to again be due to the lack of a load transfer model.

Future work to include suspension load transfer and a clutch model in the drivetrain subsystem would be required to address the shortcomings in the vehicle model. Additionally, the use of brake pressure sensors and steering sensors in the test phase of the Formula Student car would allow better side by side comparison of the LTS.

The Lap Time Simulator was validated by using the UGR18 speed profile from two test sessions as the target speed profile in the LTS. The LTS showed decent accuracy – achieving lap times of around 10% of the real vehicle. This accuracy is expected to increase with the addition of a suspension model and optimisation of the target speed profile.

Optimisation of the vehicle speed profile has successfully been carried out in the vehicle launching scenario; the optimisation routine was able to predict the optimum engine torque for the highest acceleration. Work is currently underway to optimise the speed profile for the curving track scenario, which will allow the fastest possible lap times to be found.

The Lap Time Simulator and optimisation routine has been designed such that any vehicle parameter can be investigated and optimised as the user desires. This will allow future concept studies to be carried out within the UGRacing team and allow the best vehicle set-ups for future events to be realised.

7 References

[1] SAE International, "Vehicle Dynamics Terminology J670_200801," *SAE Standards,* vol. J670, no. 200801, 2008.

[2] IMeche, "formula-student rules," 2018. [Online]. Available: http://www.imeche.org/docs/default-source/1-oscar/formula-student/fs-rules_2018_v1-1.pdf?sfvrsn=2.

[3] Formula 1®, "Testing," 2019. [Online]. Available: https://www.formula1.com/en/championship/inside-f1/understanding-f1-racing/Testing.html. [Accessed 2019].

[4] R. Roland, J. Douglas and C. Thelin, Computer Simulation of Watkins Glen Grand Prix Cicuit Performance, Calspan Report No. ZL-5002-K-1, 1971.

[5] D. W.F.Milliken, Race Car Vehicle Dynamics, SAE (ISBN 1-56091-526-9), 1995.

[6] Calspan, "Tire Performance Testing," [Online]. Available: https://www.calspan.com/services/transportation-testing-research-equipment/tire-performance-testing/. [Accessed Jan 2019].

[7] Formula SAE, "Formula SAE Tire Test Consortium," [Online]. Available: http://www.fsaettc.org/. [Accessed Jan 2019].

[8] H. Pacejka, Tire and Dehicle Dynamics, Elsevier (ISBN 978-0-08-097016-5), 2002.

[9] Optimum G, "Optimum Lap," 2018. [Online]. Available: http://www.optimumg.com/software/optimumlap/.

[10] B. Siegler, A. Deakin and D. Crolla, "Lap Time Simulation: Comparison of Steady State, Quasi- Static and Transient Racing Car Cornering Strategies," *SAE TECHNICAL PAPER SERIES,* no. 2000-01-3563, pp. 5-6, 2000.

[11] Milliken Research Associates Inc, "Lap Time Simulation, LTS," 2018. [Online]. Available: http://www.millikenresearch.com/lts.html. [Accessed Jan 2019].

[12] Tag Heuer, "Professional Timing," 2019. [Online]. Available: https://tagheuer-timing.co.uk/. [Accessed 2019].

[13] DATAS Ltd, "RaceSim," 2015. [Online]. Available: http://www.datas-ltd.com. [Accessed 2019].

[14] ADAMS, MSC Software, "Adams car," 2019. [Online]. Available: http://www.mscsoftware.com/product/adams-car. [Accessed 2019].

[15] J. Ellis, Vehicle Handling Dynamics (ISBN 0-85298-885-0), London: Mechanical Engineering Publications, 1994.

[16] C. Reynolds, "Steering Behaviors For Autonomous Characters," in *Game Developers Conference*, San Jose, California, 1999.

[17] D. Casanova, R. Sharp and P. Symonds, "A Mathematical Model for Driver Steering Control, with Design, Tuning and Performance Results," *Vehicle System Dynamics,* vol. 33, no. 5, pp. 289-326, 2010.

[18] C. MacAdam, "Application of an Optimal Preview Control for Simulation of Closed-Loop Automobile Driving," *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 11, no. 6, 1981.

[19] MathWorks, "MathWorks Support for Student Competitions," 2019. [Online]. Available: https://uk.mathworks.com/academia/student-competitions.html. [Accessed Jan 2019].

[20] MathWorks, "MATLAB and Simulink Racing Lounge," 2019. [Online]. Available: https://uk.mathworks.com/academia/student-competitions/racing-lounge.html.

[21] MathWorks, "Mapped SI Engine," 2019. [Online]. Available: https://uk.mathworks.com/help/autoblks/ref/mappedsiengine.html. [Accessed Jan 2019].

[22] Ricardo, "WAVE," 2019. [Online]. Available: https://software.ricardo.com/products/wave. [Accessed Jan 2019].

[23] MathWorks, "Limited Slip Differential," 2017. [Online]. Available: https://www.mathworks.com/help/releases/R2018b/autoblks/ref/limitedslipdifferential.html. [Accessed March 2019].

[24] B. Siegler, "Lap Time Simulation for Racing Car Design," The University of Leeds; School of Mechanical Engineering, Leeds, 2002.

[25] Monklands Sporting Car Club, "Hill Climbs," 2019. [Online]. Available: https://www.mscc.org.uk/forrestburn/hill-climbs. [Accessed March 2019].

[26] MathWorks, "Driving Scenario Designer," 2018. [Online]. Available: https://uk.mathworks.com/help/releases/R2018b/driving/ref/drivingscenariodesigner-app.html. [Accessed March 2019].

[27] MathWorks, "Longitudinal Driver," 2017. [Online]. Available: https://www.mathworks.com/help/releases/R2018b/autoblks/ref/longitudinaldriver.html. [Accessed 2019].

[28] MathWorks, "fmincon," 2006. [Online]. Available: https://www.mathworks.com/help/releases/R2018b/optim/ug/fmincon.html. [Accessed March 2019].

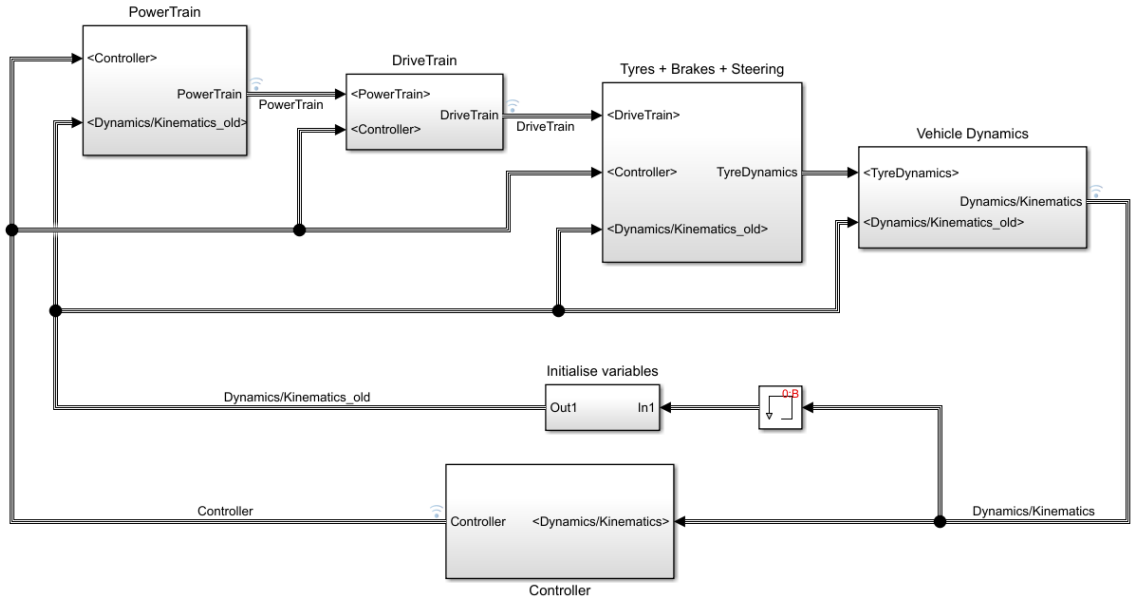
[29] IMeche, "FSUK Previous Events," 2018. [Online]. Available: http://www.imeche.org/docs/default-source/1-oscar/formula-student/2018/2018-results/fs_uk_2018---endurance.pdf?sfvrsn=2. [Accessed March 2019].

[30] MathWorks, "Modeling a Vehicle Dynamics System," 2018. [Online]. Available: https://uk.mathworks.com/help/ident/examples/modeling-a-vehicle-dynamics-system.html. [Accessed Jan 2019].

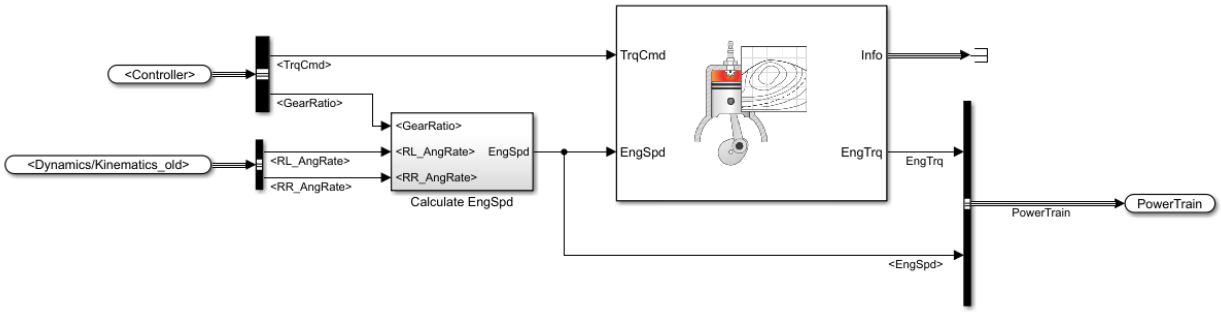
Appendix A

The UGR_LTS Simulink model subsystems are detailed below

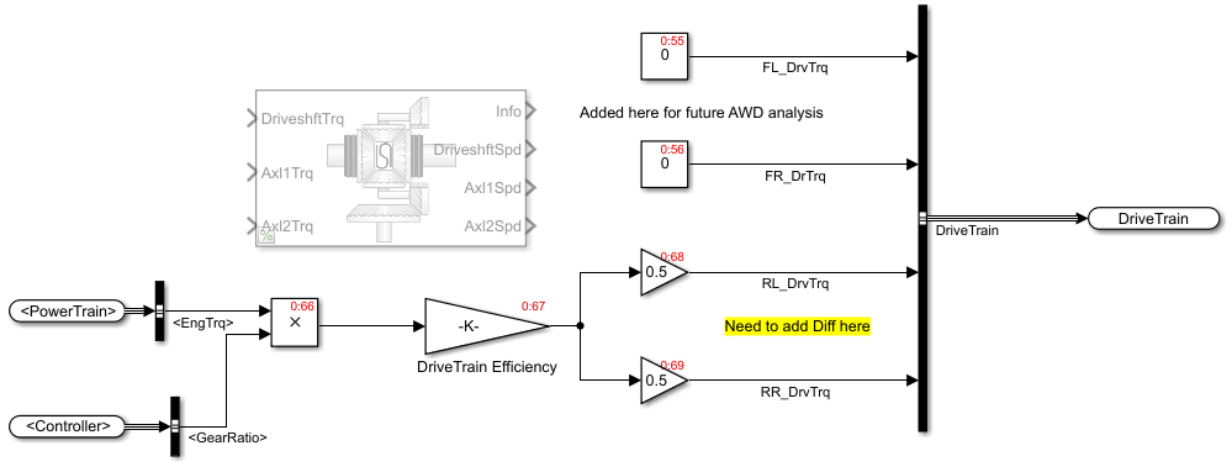
A.1 UGR_LTS Model



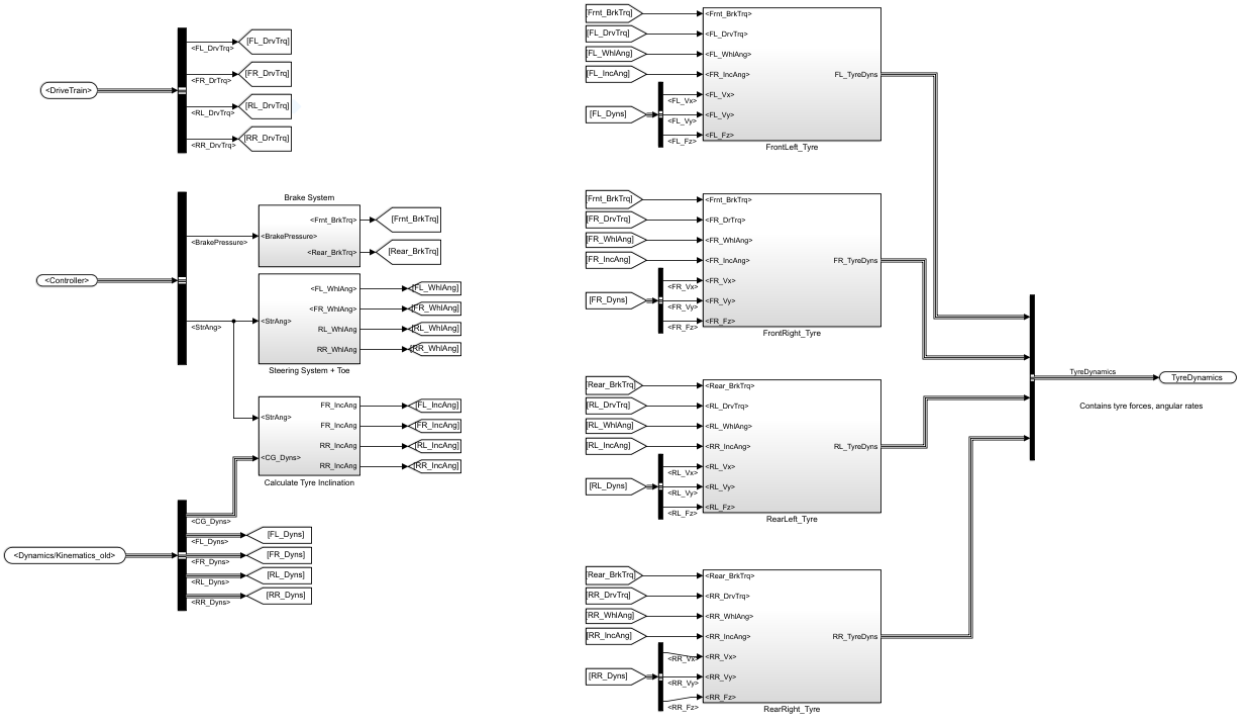
A.2 Powertrain Model



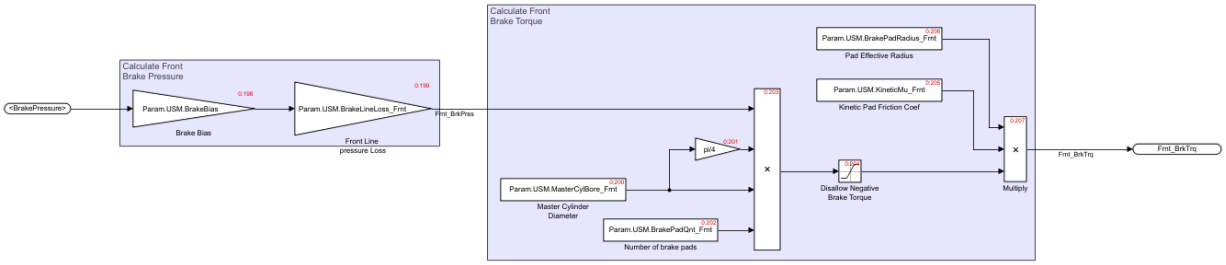
A.3 Drivetrain Model



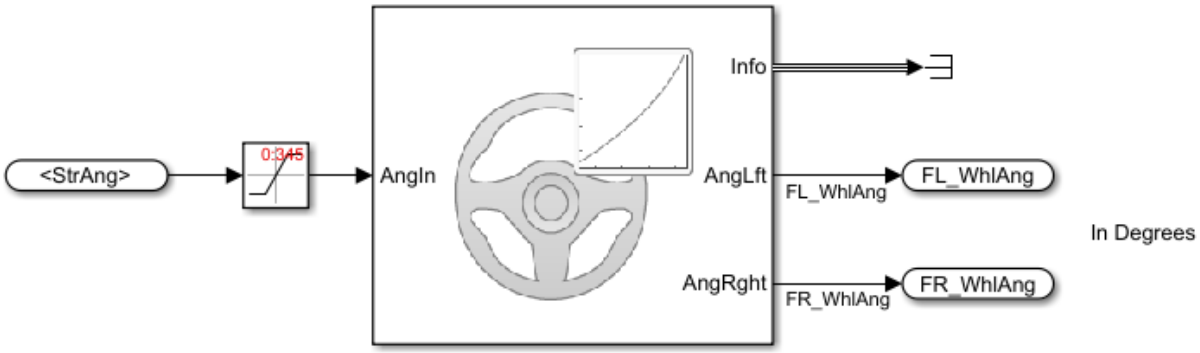
A.4 Tyre + Brakes + Steering Model



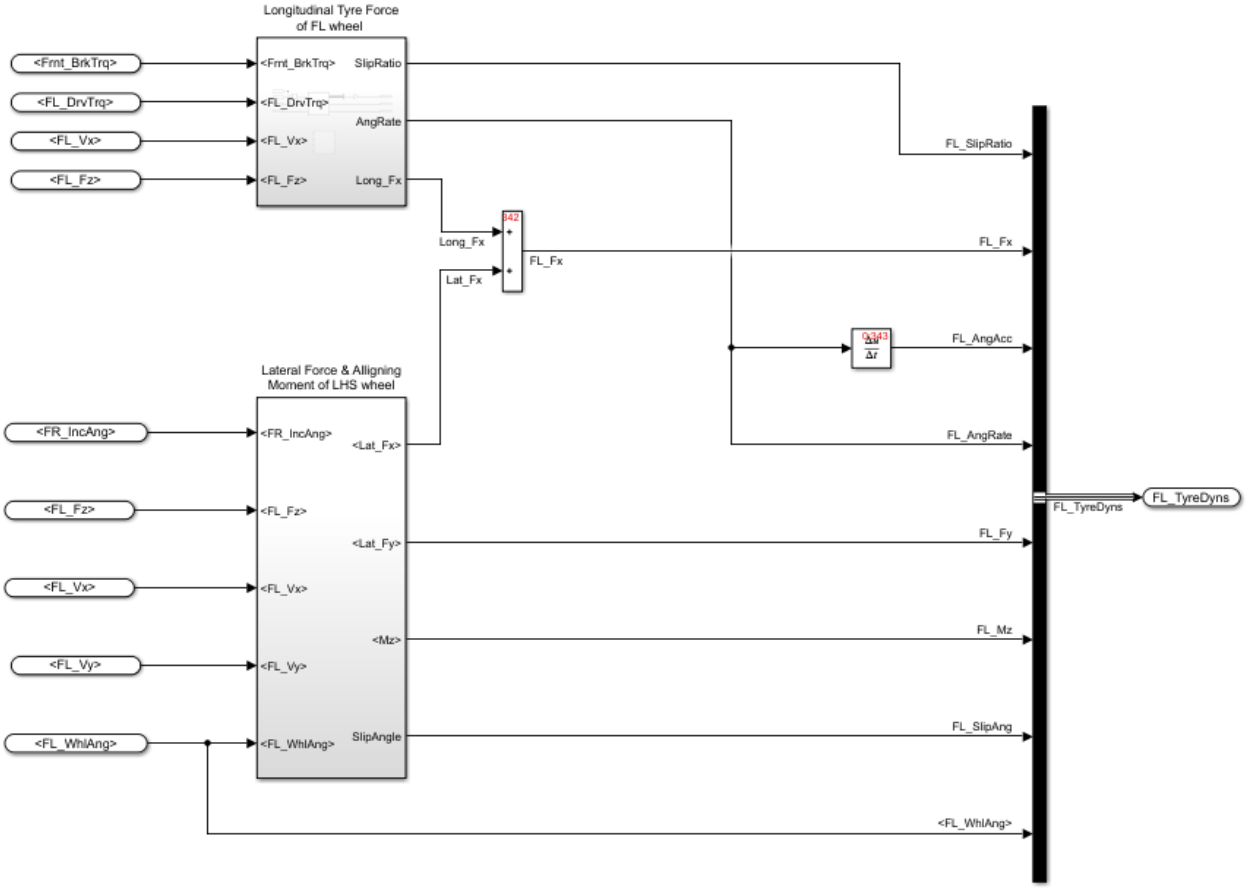
A.4.1 Brake System Model



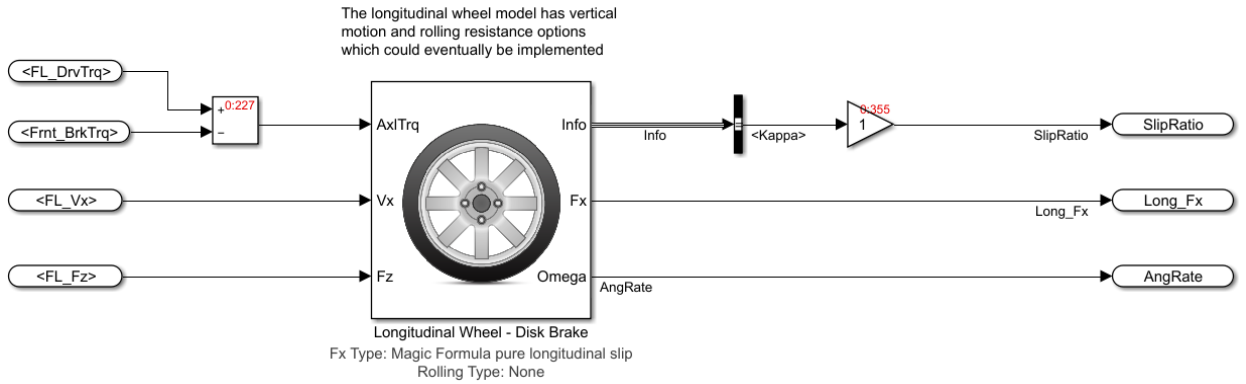
A.4.2 Steering Model



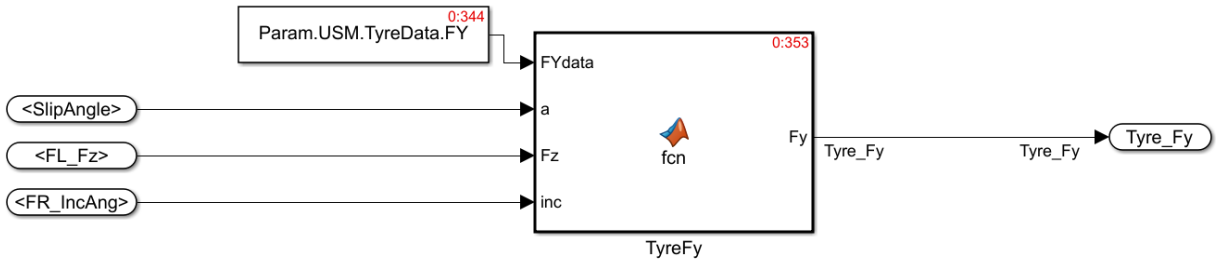
A.4.3 Tyre Model



A.4.3.1 Longitudinal Tyre Model

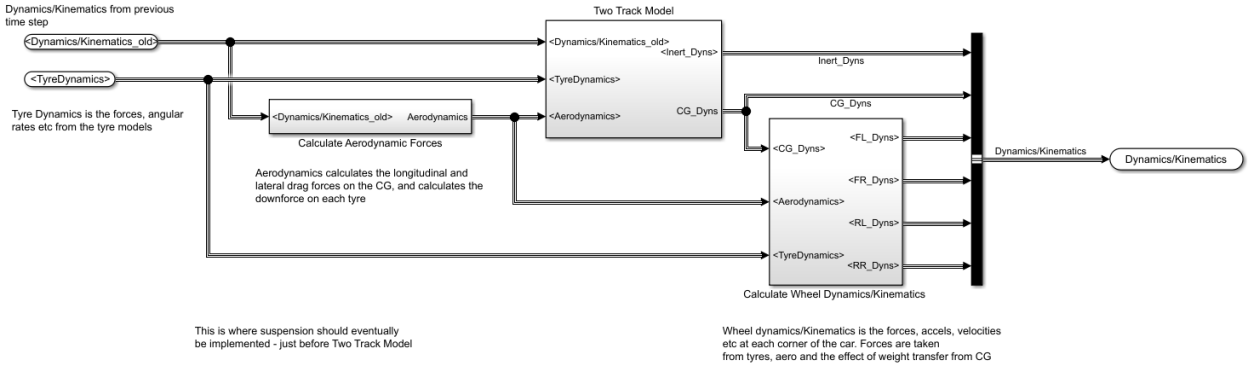


A.4.3.2 Lateral Simulink Tyre Model

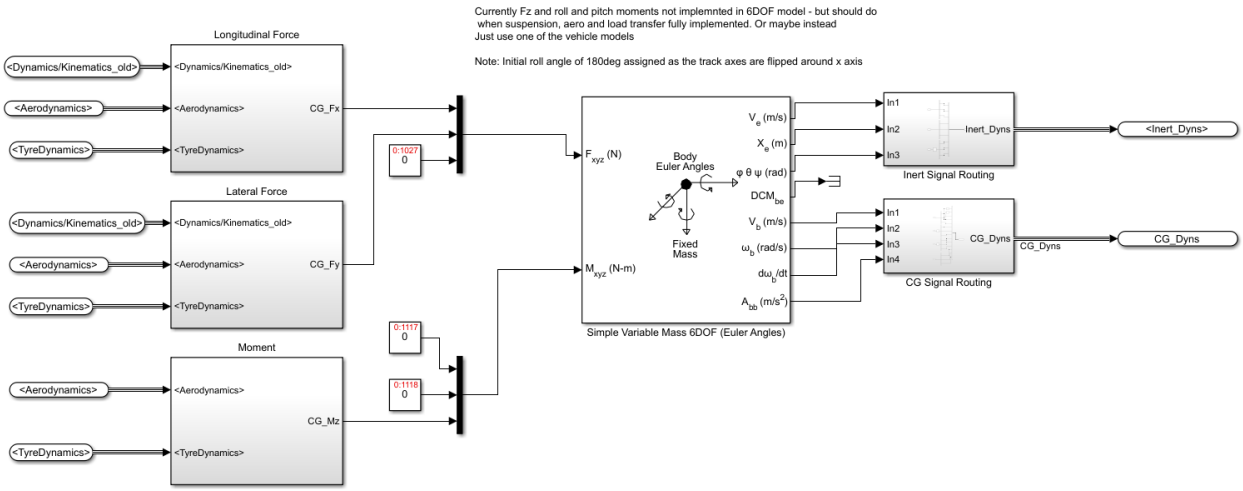


Pacejka 96 equation for lateral force
Find inside ..\Lap Time Simulator\Functions\Tyres

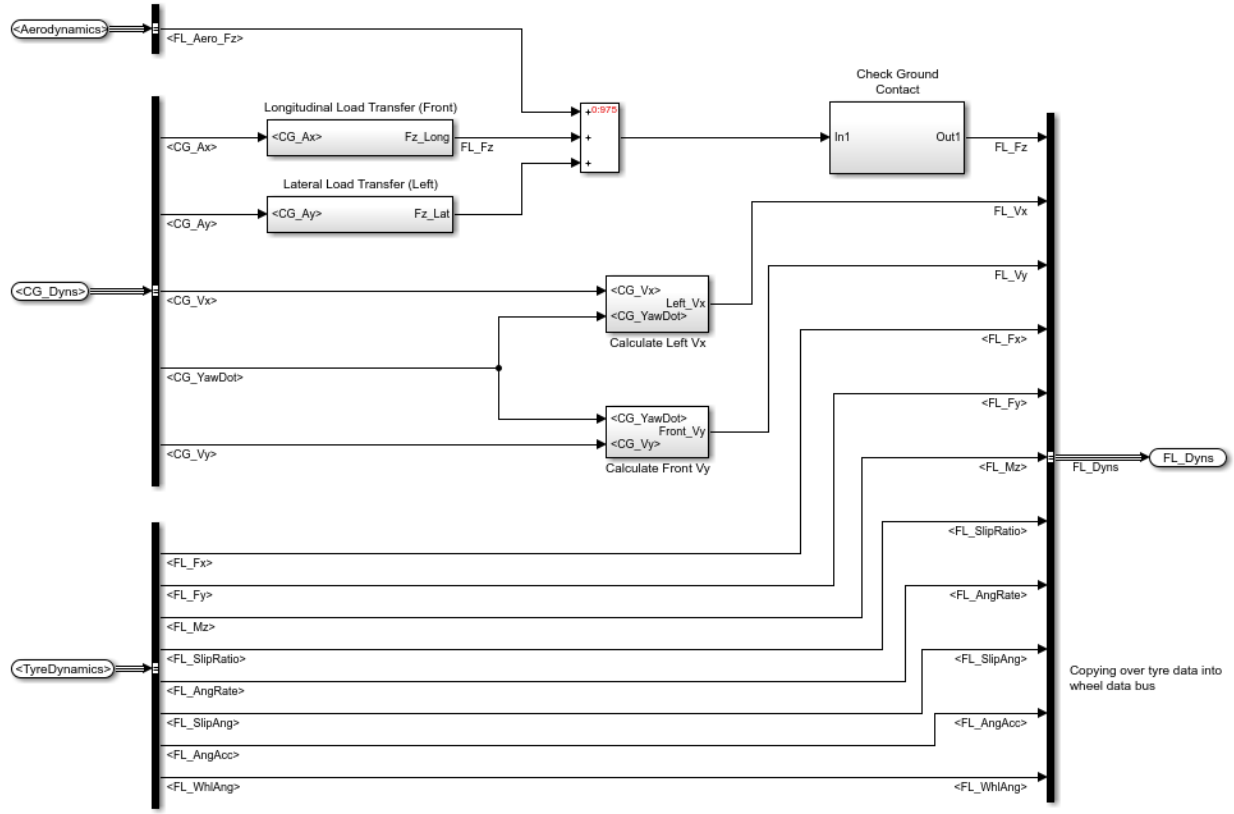
A.5 Vehicle Dynamics Model



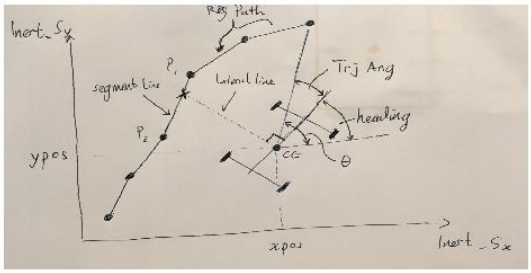
A.5.1 Two Track Model

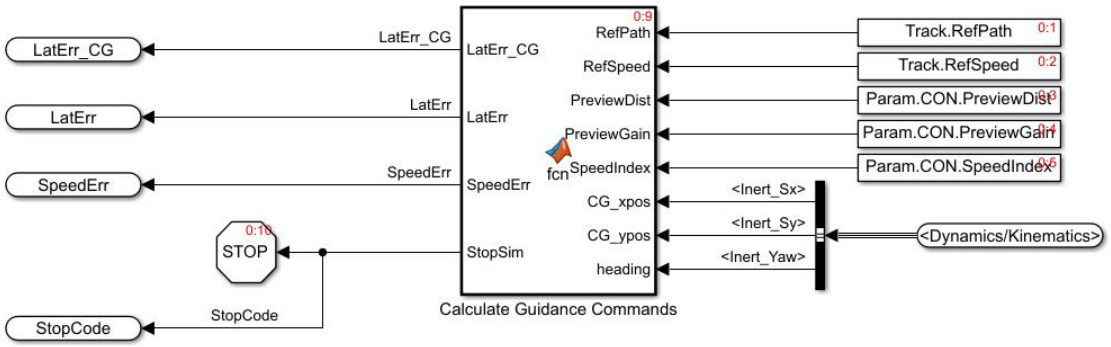


A.5.2 Wheel Dynamics Model



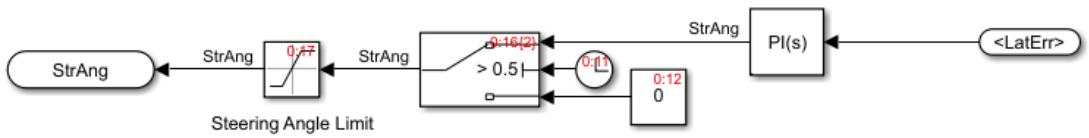
A.6 Guidance Model





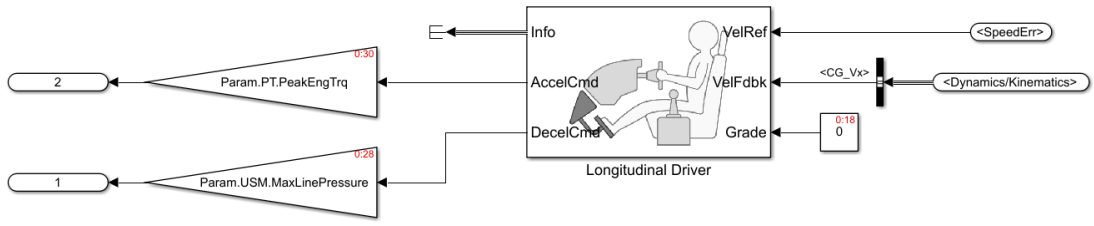
LatErr is a function of all preview points, LatErr_CG is geometric lateral error from CG

A.7 Lateral Controller Model



0.5second delay to remove initial flucatuations
in steering angle - this needs looked at

A.8 Longitudinal Controller Model



University of Glasgow

Appendix B

B.1 Vehicle Parameter Setup script – Param_UGR18.m

This script specifies the 'Param' structure values which contain all of the parameters required by the various LTS subsystems:

```
%% CHASSIS
Param.CH.Mass = 300; % Total vehicle mass, kg
Param.CH.Ixx = 1; % Roll moment of inertia, kg*m^2
Param.CH.Iyy = 1; % Yaw moment of inertia, kg*m^2
Param.CH.Izz = 90; % Yaw moment of inertia, kg*m^2 (NEEDS UPDATED - approximat)
Param.CH.I = eye(3).*[Param.CH.Ixx;Param.CH.Iyy;Param.CH.Izz]; % Moment of inertia matrix
Param.CH.CG_h = 0.3; % CG height above ground, m
Param.CH.Fnt_WDist = 0.45; % Stationary front weight distribution, eg 0.4 = 40% of mass sits on
front wheel
Param.CH.WheelBase = 1.55; % Wheel Base, m
Param.CH.Fnt_CGDist = (1 - Param.CH.Fnt_WDist)*Param.CH.WheelBase; % Distance from CG to front
axle (m)
Param.CH.Rear_CGDist = Param.CH.Fnt_WDist*Param.CH.WheelBase; % Distance from CG to front axle
(m)
Param.CH.FntTrack = 1.19; %Front Track (m) (NEEDS UPDATED)
Param.CH.RearTrack = 1.1; % Rear Track (m) (NEEDS UPDATED)

%% SUSPENSION
Param.SUS.FrntToe = 0; % Front static Toe angle (deg), +ve toe in, -ve toe out
Param.SUS.RearToe = 0; % Rear static Toe angle (deg), +ve toe in, -ve toe out
Param.SUS.FrntIncl = 0; % Front static tyre incliniation [camber] angle (deg), usually -ve
Param.SUS.RearIncl = 0; % Rear static tyre incliniation [camber] angle (deg), usually -ve

%% STEERING
% For details steering block details see
https://www.mathworks.com/help/releases/R2018b/vdynblks/ref/kinematicsteering.html
% Steering data obtained from UGR18 testing
Param.STR.StrAng_Bpts = [-135 -90 -45 0 45 90 135]; % Steering wheel breakpoints (deg)
Param.STR.WhlAngLft_Bpts =[-20.6 -12.8 -6.2 0 5.9 12.0 19.2]; % Left wheel angle breakpoints
(deg)
Param.STR.WhlAngRgt_Bpts = [-19.2 -12.0 -5.9 0 6.2 12.8 20.6]; % Right wheel angle breakpoints
(deg)
Param.STR.Ratio = mean(Param.STR.StrAng_Bpts./Param.STR.WhlAngLft_Bpts,'omitnan'); % Average
steering ratio
Param.STR.StrRange = 150; %steering wheel range (deg)
Param.STR.WhlRange = Param.STR.StrRange/Param.STR.Ratio; % Wheel angle range (deg)

%% POWERTRAIN
Param.PT.EngTrqRPM = load('CBR600RR_TrqRpm_WAVE.mat'); % Engine data and RPM breakpoints and map
% See the Mapped SI Engine simulink block for further data to add into Mat
% file above - eg  Need to eventually add air, fuel, Temp etc data to this
Param.PT.EngMap =   CreateEngineMap(Param.PT.EngTrqRPM.RPM,Param.PT.EngTrqRPM.TorqueNm);
Param.PT.PeakEngTrq = max(Param.PT.EngTrqRPM.TorqueNm);

%% DRIVETRAIN
Param.DT.DT_Efficiency = 0.8; %Drive Train efficiency
% Drive ratios obtained from: http://www.aperaceparts.com/tech/2008hondacbr600rr.html
Param.DT.PrimaryRatio = 2.111;      % Primary drive gear ratio: between engine and clutch
Param.DT.TransRatio_1st = 2.750;    % 1st gear transmission ratio: between clutch and sprocket
Param.DT.TransRatio_2nd = 2.000;    % 2nd gear transmission ratio
Param.DT.TransRatio_3rd = 1.666;    % 3rd gear transmission ratio
Param.DT.TransRatio_4th = 1.444;    % 4th gear transmission ratio
Param.DT.TransRatio_5th = 1.304;    % 5th gear transmission ratio
Param.DT.TransRatio_6th = 1.208;    % 6th gear transmission ratio
Param.DT.FinalRatio = 3;            % Final drive ratio: between sprocket and wheel
```

```
% Gear ratios calculated from drive ratios above:
Param.DT.GearRatio_1st = Param.DT.PrimaryRatio*Param.DT.TransRatio_1st*Param.DT.FinalRatio;
Param.DT.GearRatio_2nd = Param.DT.PrimaryRatio*Param.DT.TransRatio_2nd*Param.DT.FinalRatio;
Param.DT.GearRatio_3rd = Param.DT.PrimaryRatio*Param.DT.TransRatio_3rd*Param.DT.FinalRatio;
Param.DT.GearRatio_4th = Param.DT.PrimaryRatio*Param.DT.TransRatio_4th*Param.DT.FinalRatio;
Param.DT.GearRatio_5th = Param.DT.PrimaryRatio*Param.DT.TransRatio_5th*Param.DT.FinalRatio;
Param.DT.GearRatio_6th = Param.DT.PrimaryRatio*Param.DT.TransRatio_6th*Param.DT.FinalRatio;

%% UNSPRUNG MASS
% Brake System
Param.USM.BrakeBias = 0.7; % ratio of front to rear, eg 0.6 = 60% of pressure at front
Param.USM.KineticMu_Frnt = 0.65; % Coefficient of kinetic friction for front pads
Param.USM.KineticMu_Rear = 0.4;
Param.USM.StaticMu_Frnt = Param.USM.KineticMu_Frnt*1.1;  % Coefficient of static friction for
front pads (assuming 10% over kinetic mu)
Param.USM.StaticMu_Rear = Param.USM.KineticMu_Rear*1.1;
Param.USM.MasterCylBore_Frnt = 0.015; % Master cylinder diameter for front system,m
Param.USM.MasterCylBore_Rear = 0.015;
Param.USM.BrakePadRadius_Frnt = 0.08375; % Radius to the centre of pad, m
Param.USM.BrakePadRadius_Rear = 0.07375;
Param.USM.BrakePadQnt_Frnt = 4; % Quantity of pads of per wheel on front
Param.USM.BrakePadQnt_Rear = 2;
Param.USM.BrakeLineLoss_Frnt = 1; % Line pressure losses, eg 0.9 = 90%
Param.USM.BrakeLineLoss_Rear = 1;
Param.USM.MaxLinePressure = 7e+6; % Peak brake line pressure, Pa

% Wheels
Param.USM.WheelRadius = 0.254; % Loaded Wheel Radius,m (NEEDS UPDATED)
Param.USM.I_FrntWhl = 100*0.0897; % Moment of interia of UGR19 front wheel assembly(wheel and
tyre) - from CAD, kg*m^2
Param.USM.I_RearWhl = 100*0.0897; % Moment of interia of UGR19 rear wheel assembly(wheel and
tyre) - from CAD, kg*m^2

% Tyres
Param.USM.TyreData.FX = Pacejka_avon_70_20_13_FX_LongWheel; % Longitudinal Tyre data, for
'Longitudinal Wheel' model
%Param.USM.TyreData.FX = Pacejka_avon_70_20_13_FX; % Longitudinal Tyre data
Param.USM.TyreData.FY = Pacejka_avon_70_20_13_FY; % Lateral Tyre data
Param.USM.TyrePres_Frnt = 82737; % Tyre Pressure of front tyres, Pa
Param.USM.TyrePres_Rear = 82737; % Tyre Pressure of rear tyres, Pa
Param.USM.TyreData.RollRes = 0; % Tyre Rolling resistance, N/rads

%% AERO
Param.AE.CD = -0.7; % Overall drag coefficient (NEEDS UPDATED)
Param.AE.CL = 0.0; % Overall lift coefficient (NEEDS UPDATED, front and rear values, or possible
a distribution?)
Param.AE.CL_FntDist = 0.45; % Lift force front distribution, 0.4 = 40% of downforce on front
wheels
Param.AE.FrntArea = 1.0; % Frontal surface area (m^2)
Param.AE.AirDensity = 1.225; %kg/m^3 @15 degC

%% CONTROLLER
Param.CON.PreviewDist = [0 2 4]'; % Preview distance of track, m (increasing array from 0)
Param.CON.PreviewGain = [0.5 0.75 0.1]'; % Gain applied to previewed error
% Param.CON.PreviewGain = (1/sum(Param.CON.PreviewGain)).*Param.CON.PreviewGain; % Normalised
preview gain
Param.CON.SpeedIndex = length(Param.CON.PreviewGain); % Index of preview point to take speed
error from
Param.CON.Lat_Kp = -100; % Lateral Controller Proportional Gain
Param.CON.Lat_Ki = -20; % Lateral Controller Integral gain
Param.CON.Long_Kp = 15; % Longitudinal Controller Proportional Gain
Param.CON.Long_Ki = 1; % Longitudinal Controller Integral gain
Param.CON.Long_vnom = 15; %Longitudinal Controller Nominal velocity, m/s - (NEEDS UPDATED)
```

B.2 Track Parameter Setup function – LoadTrack.m

This function loads the mat file created by the 'Driving Scenerio Designer' app and outputs the 'Track' structure ready for use by the LTS:

```
function [Track] = LoadTrack(TrackDataFile)
% Function to gather and process track data from a driving scenario mat
% file 'TrackDataFile' - outputs a structure called Track

% The track coordinates and velocity profile are found by running and recording the
% driving scenario

Track = load(TrackDataFile);

% Construct a drivingScenario object.
scenario = drivingScenario;

% Add the ego car
egoCar = vehicle(scenario, ...
    'ClassID', 1);
waypoints = Track.data.ActorSpecifications.Waypoints;
speed = Track.data.ActorSpecifications.Speed;
trajectory(egoCar, waypoints, speed);

% Setup simulation
scenario.SampleTime = 0.05; % THIS CAN BE UPDATED TO INCREASE/DECREASE RESOLUTION

% Extract ego pose information
restart(scenario);
poses = record(scenario);

% Driver path is a subsampled version of ego poses
numPoints = numel(poses);
Track.RefPath = zeros(numPoints,2);
Track.RefYaw = zeros(numPoints,1);
Track.RefVel = zeros(numPoints,2);
for n = 1:numPoints
    Track.RefPath(n,:) = poses(n).ActorPoses(1).Position(1:2);
    Track.RefYaw(n,:) = poses(n).ActorPoses(1).Yaw;
    Track.RefVel(n,:) = poses(n).ActorPoses(1).Velocity(1:2);
end
Track.RefSpeed = sqrt((Track.RefVel(:,1).^2)+(Track.RefVel(:,2).^2));
Track.simStopTime = poses(end).SimulationTime;
```

B.3 Initial Condition Setup script – IntCon.m

This script specifies the 'IntCon' structure values which contain all of the initial conditions required by the various LTS subsystems:

```
% CG Dynamics
IntCon.CG_Vx = 1; % Velocity of CG in x direction, m/s DONT START WITH ZERO

% Inertial Dynamics
IntCon.Inert_Yaw = 180; % Inertial Heading angle, deg (0deg parrallel with inertial x axis, 90deg
parrallel w inertial y axis)
IntCon.Inert_Sx = 1; % Inertial x position, m
IntCon.Inert_Sy = 0; % Inertial y position, m
if exist('Track') % assigns initial inertial dynamics based on track start poition and angle
    IntCon.Inert_Sx = Track.RefPath(1,1);
    IntCon.Inert_Sy = Track.RefPath(1,2);
    IntCon.Inert_Yaw = atan2d((Track.RefPath(2,2)-Track.RefPath(1,2)),...
```

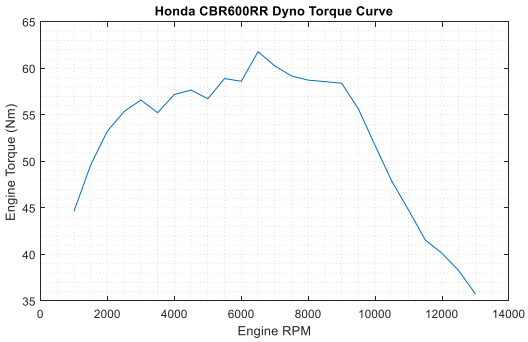
```
                                (Track.RefPath(2,1)-Track.RefPath(1,1)));
end

% Fz
IntCon.FL_Fz = 0.5*Param.CH.Mass*9.81*Param.CH.Fnt_WDist; % Front Left normal force
IntCon.FR_Fz = 0.5*Param.CH.Mass*9.81*Param.CH.Fnt_WDist; % Front Right normal force
IntCon.RL_Fz = 0.5*Param.CH.Mass*9.81*(1-Param.CH.Fnt_WDist); % Rear Left normal force
IntCon.RR_Fz = 0.5*Param.CH.Mass*9.81*(1-Param.CH.Fnt_WDist); % Rear Right normal force

% Vx
IntCon.FL_Vx = IntCon.CG_Vx; % Front Left Velocity in X direction (m/s)
IntCon.FR_Vx = IntCon.CG_Vx;
IntCon.RL_Vx = IntCon.CG_Vx;
IntCon.RR_Vx = IntCon.CG_Vx;

% Ang Rate
IntCon.FL_AngRate = IntCon.FL_Vx/Param.USM.WheelRadius; % Front Left Wheel angular rate (rad/s)
IntCon.FR_AngRate = IntCon.FR_Vx/Param.USM.WheelRadius;
IntCon.RL_AngRate = IntCon.RL_Vx/Param.USM.WheelRadius;
IntCon.RR_AngRate = IntCon.RR_Vx/Param.USM.WheelRadius;
```

Appendix C

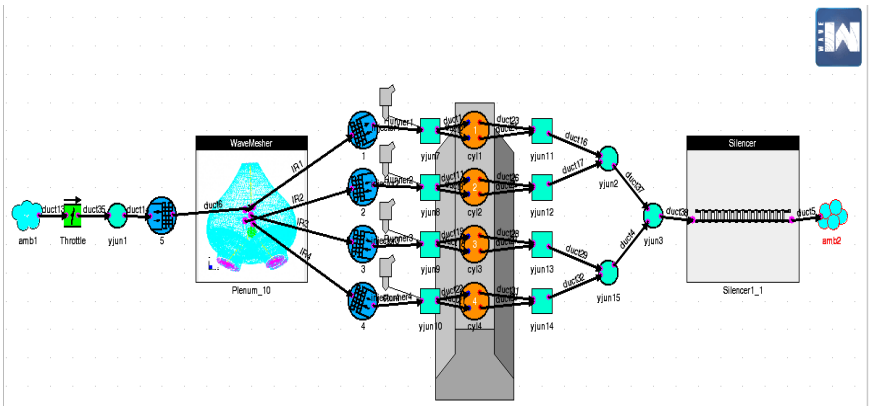
Shown below are the various tests and simulations which were carried out external to this project which were used to validate the LTS.

C.1 Engine Testing

UGR18 Honda CBR600RR Dynamometer testing:



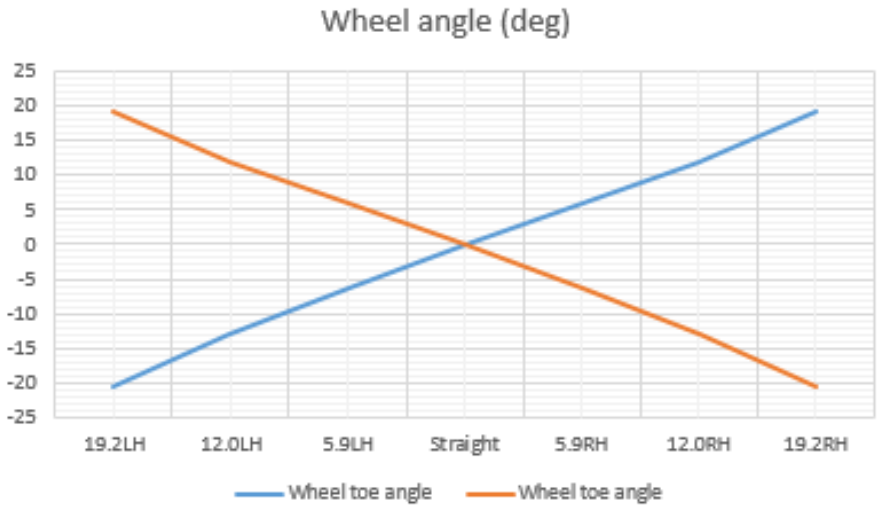
UGR18 Honda CBR600RR Ricardo Wave setup:



C.2 Steering System Testing

UGR18 Steering system testing setup:

UGR18 Steering angle test results:

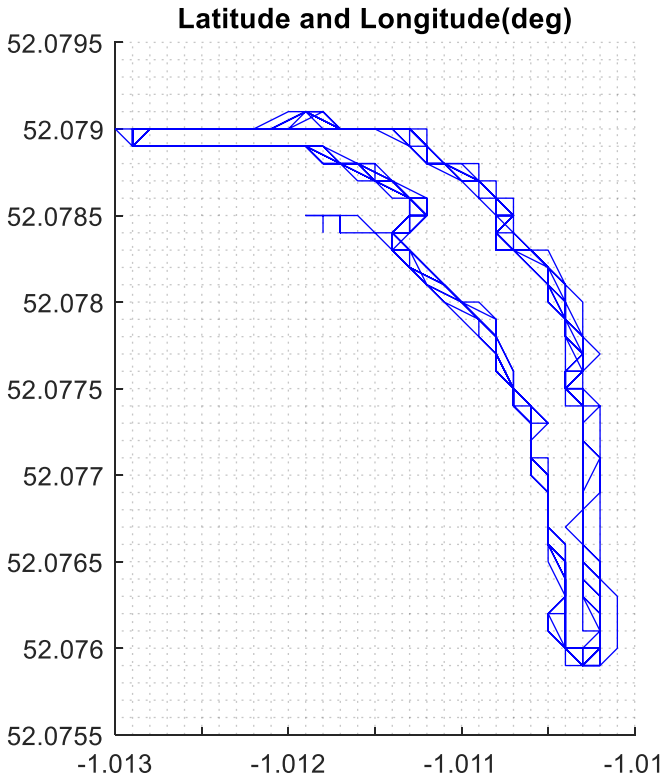


C.3 FSUK 2018 Endurance Results

FSUK 18 Endurance results:

Endurance Place	Endurance Car Num	Team Endurance	Endurance Time	Endurance Laps	Endurance Cones	Endurance Adjusted Time	Endurance Score
15	52	University of Glasgow	1789.6	22	14	1817.6	115.7

FSUK 18 GPS data collected for 22 laps:



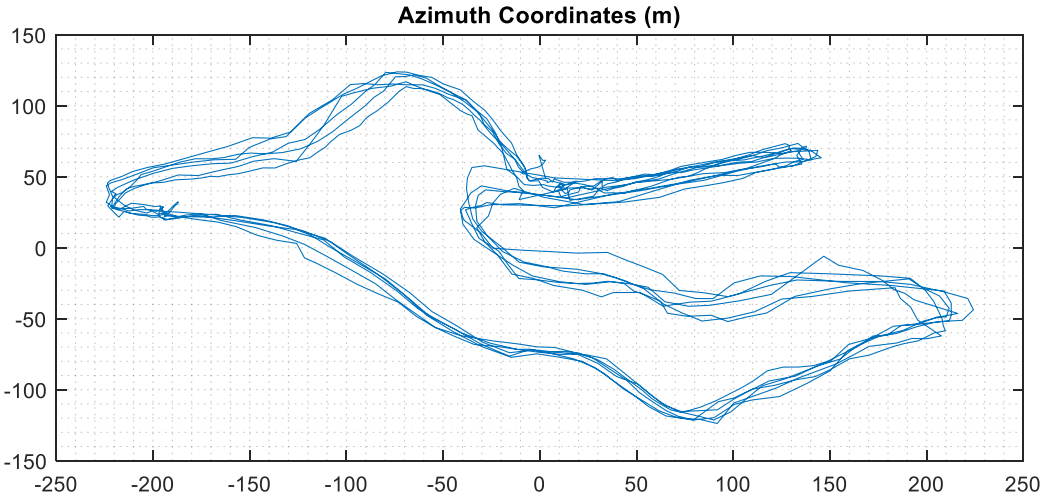
C.4 Forrestburn Hillclimb 2018 Testing

All testing data from the Forrestburn Hill climb was collected using the 'Track Addict' Data collection tool - http://racerender.com/TrackAddict/Features.html

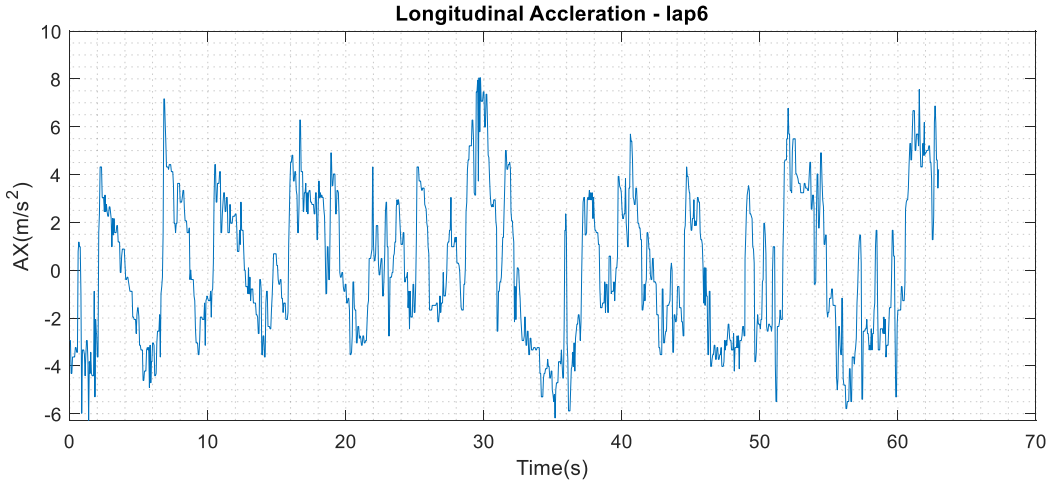
The Track Addict data was post processed using the 'Race Render' tool - http://racerender.com/RR3/Features.html

Forrestburn Hill climb testing footage:

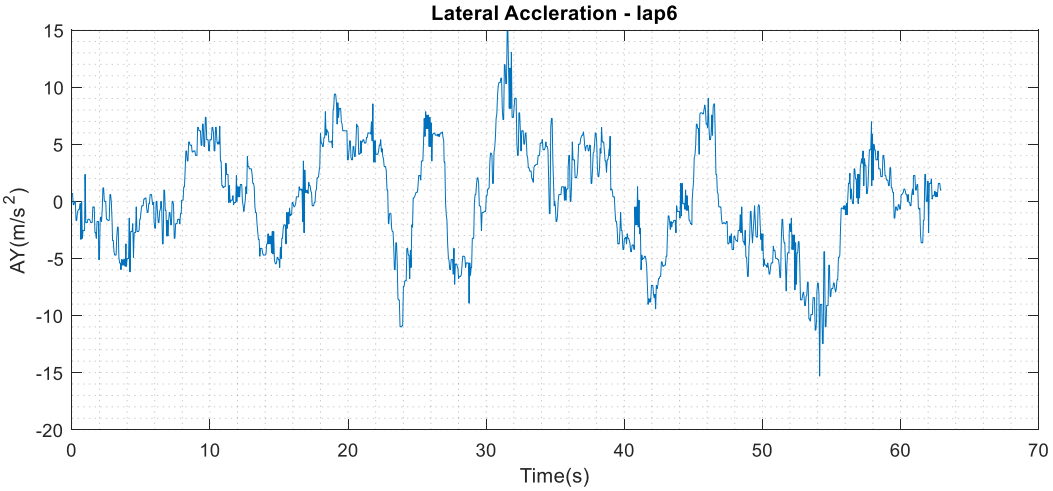
GPS Data recorded for all 6 laps:



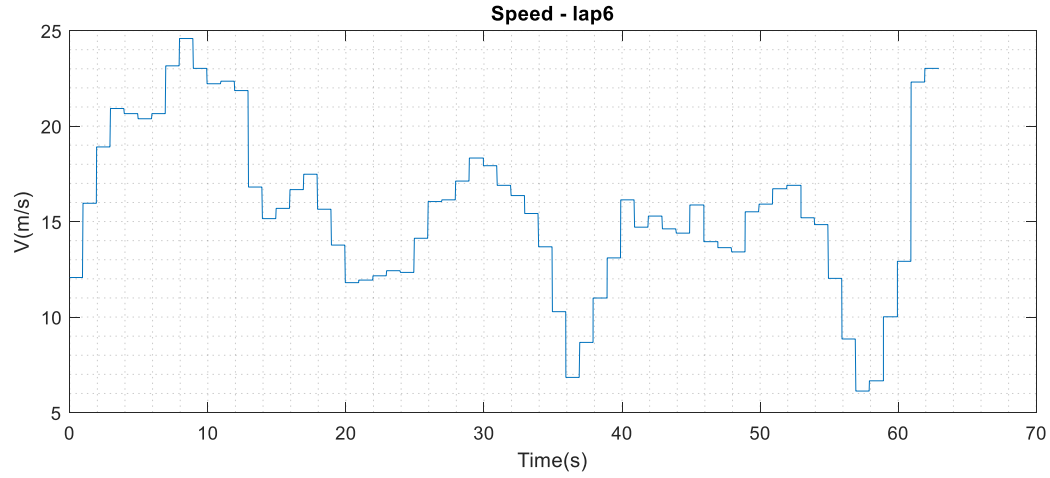
Longitudinal Acceleration recorded for lap 6:



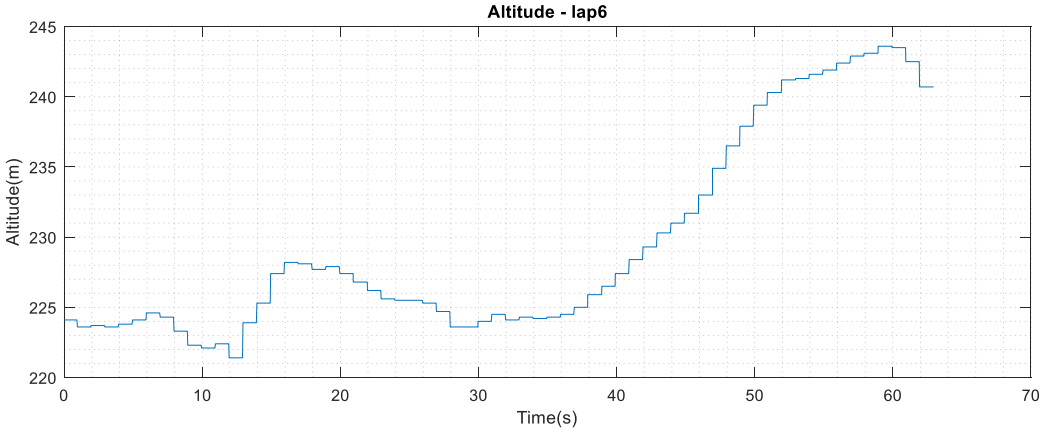
Lateral Acceleration recorded for lap6:



Speed recorded for lap 6:



Altitude recorded for lap 6:



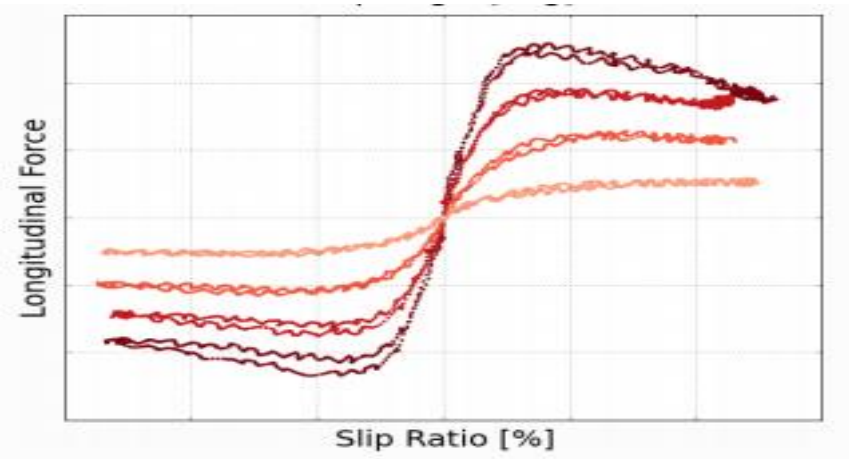
Appendix D

Detailed below is the method by which the tyres were tested and processed to produce the LTS tyre models.

D.1 Tyre Testing Consortium

The Tyres are tested by Calspan using a test belt:

The raw tyre forces are recorded as:



D.2 Pacejka Tyre Coefficients

The 1996 Pacejka curve fitting equations are used to process the raw tyre data.

Note that:

Physical Characteristics		
F_{z0}	Nominal Wheel Load, N	1125.93000
R_0	Unloaded Tyre Radius, m	0.25400

D.2.1 Longitudinal Tyre Force Pacejka Equations

$$F_{x0} = D_x \sin\left[C_x \arctan\left\{B_x\kappa_x - E_x\left(B_x\kappa_x - \arctan\left(B_x\kappa_x\right)\right)\right\}\right] + S_{Vx} \quad (1.1)$$

$$\kappa_x = \kappa + S_{Hx} \quad (1.2)$$

$$C_x = p_{Cx1}\lambda_{Cx} \quad (1.3)$$

$$D_x = \mu_x F_z \quad (1.4)$$

$$\mu_x = \left(p_{Dx1} + p_{Dx2}df_z\right)\lambda_{\mu x} \quad (1.5)$$

$$E_x = \left(p_{Ex1} + p_{Ex2}df_z + p_{Ex3}df_z^2\right)\left\{1 - p_{Ex4}\,\mathrm{sgn}\left(\kappa_x\right)\right\}\lambda_{Ex} \quad (1.6)$$

$$K_{x\kappa} = F_z\left(p_{Kx1} + p_{Kx2}df_z\right)\exp\left(-p_{Kx3}df_z\right)\lambda_{Kx\kappa} \quad (1.7)$$

$$B_x = \frac{K_{x\kappa}}{C_x D_x} \quad (1.8)$$

$$S_{Hx} = \left(p_{Hx1} + p_{Hx2}df_z\right)\lambda_{Hx} \quad (1.9)$$

$$S_{Vx} = F_z\left(p_{Vx1} + p_{Vx2}df_z\right)\lambda_{Vx}\lambda_{\mu x} \quad (1.10)$$

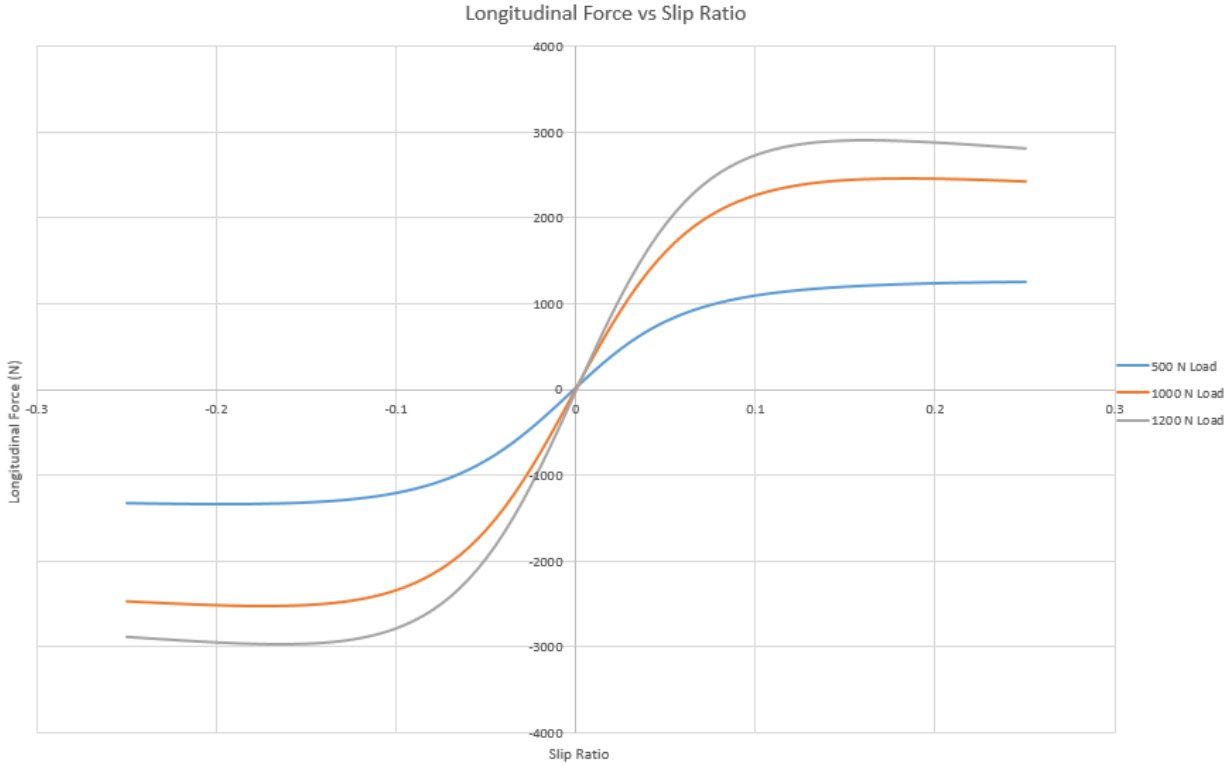
- All of the "λ" terms in Equations 1.1-1.10 are user scaling factors, which have a default value of 1. The use of each scaling factor is described in Table 1.

Table 1: Longitudinal Force User Scaling Factors

User Scaling Factors used in '96 Pacejka Tire Model	
Subscript to λ Term	**Definition**
μx	Peak friction coefficient
Kxk	Slip stiffness
Cx	Shape factor
Ex	Curvature factor
Hx	Horizontal shift
Vx	Vertical shift

Where for the UGR18 tyres:

Longitudinal Force		
p_{Cx1}	Shape factor C_x for longitudinal forces	1.50000
p_{Dx1}	Longitudinal friction μ_x at F_{z0}	2.46280
p_{Dx2}	Variation of friction μ_x with load	-0.25113
p_{Ex1}	Longitudinal curvature E_x at F_{z0}	0.03154
p_{Ex2}	Variation of curvature E_x with load	-1.06770
p_{Ex3}	Variation of curvature E_x with load^2	-0.32284
p_{Ex4}	Factor in curvature E_x while driving	-0.51161
p_{Kx1}	Longitudinal slip stiffness K_x/F_z at F_{z0}	38.54360
p_{Kx2}	Variation of slip stiffness K_x/F_z with load	-0.00003
p_{Kx3}	Exponent in slip stiffness K_x/F_z with load	0.03567
p_{Hx1}	Horizontal shift S_{Hx} at F_{z0}	0.00079
p_{Hx2}	Variation of shift S_{Hx} with load	-0.00418
p_{Vx1}	Vertical shift in S_{Vx}/F_z at F_{z0}	-0.02721
p_{Vx2}	Variation of shift S_{Vx}/F_z with load	0.06023



D.2.2 Lateral Tyre Force Pacejka Equations

$$F_{y0} = D_y \sin\left[C_y \arctan\left\{B_y\alpha_y - E_y\left(B_y\alpha_y - \arctan\left(B_y\alpha_y\right)\right)\right\}\right] + S_{vy} \quad (1.11)$$

$$\alpha_y = \alpha + S_{Hy} \quad (1.12)$$

$$\gamma_y = \gamma\lambda_{\gamma y} \quad (1.13)$$

$$C_y = p_{Cy1}\lambda_{Cy} \quad (1.14)$$

$$D_y = \mu_y F_z \quad (1.15)$$

$$\mu_y = \left(p_{Dy1} + p_{Dy2}df_z\right)\left(1 - p_{Dy3}\gamma_y^{\,2}\right)\lambda_{\mu y} \quad (1.16)$$

$$E_y = \left(p_{Ey1} + p_{Ey2}df_z\right)\left\{1 - \left(p_{Ey3} + p_{Ey4}\gamma_y\right)\mathrm{sgn}\left(\alpha_y\right)\right\}\lambda_{Ey} \quad (1.17)$$

$$K_{y\alpha} = p_{Ky1}F_{z0}\sin\left[2\arctan\left\{\frac{F_z}{p_{Ky2}F_{z0}\lambda_{Fzo}}\right\}\right]\left(1 - p_{Ky3}\,|\,\gamma\,|\right)\lambda_{Fzo}\lambda_{Ky\alpha} \quad (1.18)$$

$$B_y = \frac{K_{y\alpha}}{C_y D_y} \quad (1.19)$$

$$S_{Hy} = \left(p_{Hy1} + p_{Hy2}df_z + p_{Hy3}\gamma_y\right)\lambda_{Hy} \quad (1.20)$$

$$S_{vy} = F_z\left\{p_{vy1} + p_{vy2}df_z + \left(p_{vy3} + p_{vy4}df_z\right)\gamma_y\right\}\lambda_{vy}\lambda_{\mu y} \quad (1.21)$$

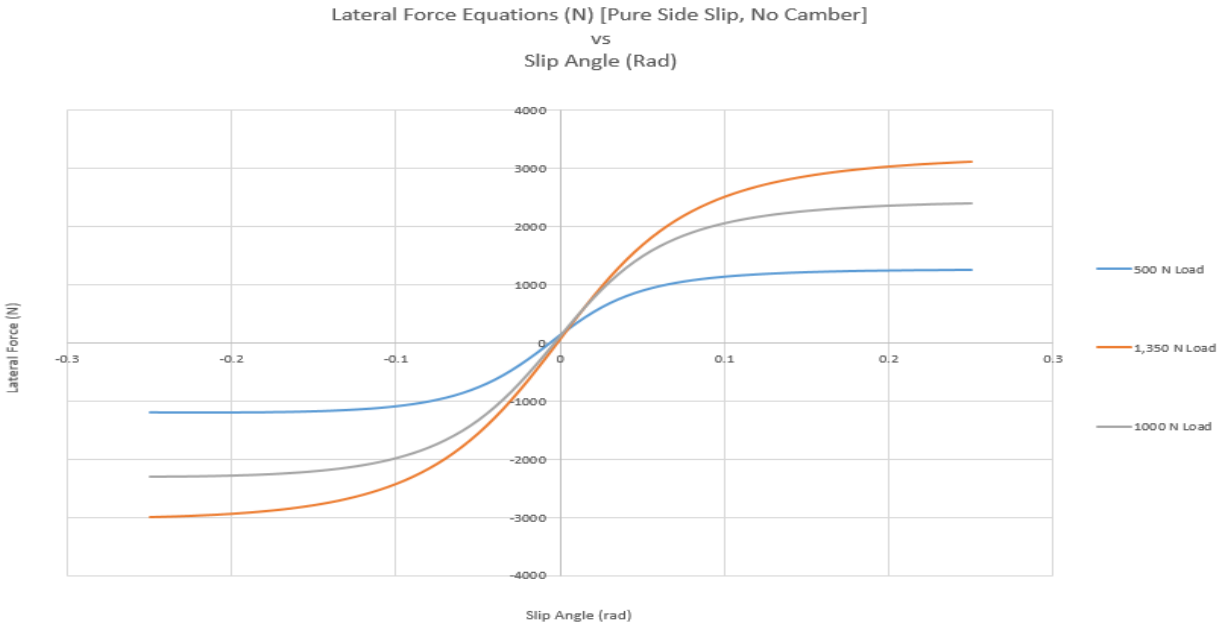
- All of the "λ" terms in Equations 1.11-1.21 are user scaling factors, which have a default value of 1. The use of each scaling factor is described in Table 2.

Table 2: Lateral Force User Scaling Factors

User Scaling Factors used in '96 Pacejka Tire Model	
Subscript to λ Term	**Definition**
Fz0	Nominal (rated) load
μy	Peak friction coefficient
Ky	Cornering stiffness
Cy	Shape factor
Ey	Curvature factor
Hy	Horizontal shift
Vy	Vertical shift
γy	Camber shift
Kyγ	Camber force stiffness
Kyα	Slip angle force stiffness

Where for UGR18 Tyres:

Lateral Force		
p_{Cy1}	Shape factor C_y for lateral forces	1.33935
p_{Dy1}	Lateral friction μ_y	2.33427
p_{Dy2}	Variation of friction with μ_y load	-0.19002
p_{Dy3}	Variation of friction μ_y with camber^2	4.63415
p_{Ey1}	Lateral curvature E_y at F_{z0}	0.44000
p_{Ey2}	Variation of curvature E_y with load	-0.00361
p_{Ey3}	Zero order camber depemdemcy of curvature E_y	-0.37561
p_{Ey4}	Variation of curvature E_y with camber	0.03212
p_{Ky1}	Maximum value of stiffness K_y/F_{z0}	-33.35031
p_{Ky2}	Load at which K_y reaches maximum value	-1.40482
p_{Ky3}	Variation of K_y/F_{z0} with camber	3.01935
p_{Hy1}	Horizontal shift S_{Hy} at F_{z0}	0.00113
p_{Hy2}	Variation of shift S_{Hy} with load	-0.00720
p_{Hy3}	Horizontal shift S_{Hy} with camber	-0.06086
p_{Vy1}	Vertical shift in S_{Vy}/F_z at F_{z0}	0.06543
p_{Vy2}	Variation of shift S_{Vy}/F_z with load	0.00211
p_{Vy3}	Variation of shift S_{Vy}/F_z with camber	-1.68755
p_{Vy4}	Variation of shift S_{Vy}/F_z with camber and load	2.93830



University of Glasgow

Appendix E

Detailed below are the optimisation routines which were used to optimise various of the LTS parameters:

E.1 Speed Profile Optimisation Program

```
%% Optimisation
% x0 = array of start point(s)
x0 = Track.data.ActorSpecifications.Speed';

% lb = array of lower bound(s)
lb = ones(size(x0))*1.5;

% ub = array of upper bound(s)
ub = ones(size(x0))*20;

%% Start with the default options
options = optimoptions('fmincon');
%% Modify options setting
% options = optimoptions(options,'Algorithm', 'active-set');
options = optimoptions(options,'Display', 'iter');
options = optimoptions(options,'Diagnostics', 'off');
options = optimoptions(options,'StepTolerance', 1e-10);
% options = optimoptions(options,'ConstraintTolerance', 1e-06);
options = optimoptions(options,'OptimalityTolerance', 1e-02);

[SpeedProfile_Opt,LapTime_Opt,exitflag,output,lambda,grad,hessian] = ...
fmincon(@SpeedProfile_ObjFun,x0,[],[],[],[],lb,ub,[],options);

disp('Initial Speed Profile:')
disp(x0);
disp('Optimum Speed Profile:');
disp(SpeedProfile_Opt);
fprintf('Optimum Lap Time: %0.3fs \n', LapTime_Opt);
```

E.2 Controller Gain Optimisation Program

```
%% Optimisation
% x0 = array of start point(s)
x0 = [Param.CON.Lat_Kp,Param.CON.Lat_Ki,Param.CON.PreviewDist];

% lb = array of lower bound(s)
lb = [-150,-100,0.1];

% ub = array of upper bound(s)
ub = [-30,-15,3];

%% Start with the default options
options = optimoptions('fmincon');
%% Modify options setting
% options = optimoptions(options,'Algorithm', 'active-set');
options = optimoptions(options,'Display', 'iter-detailed');
options = optimoptions(options,'Diagnostics', 'off');
% options = optimoptions(options,'StepTolerance', 1e-10);
% % options = optimoptions(options,'ConstraintTolerance', 1e-06);
% options = optimoptions(options,'OptimalityTolerance', 1e-02);
```

```
[Controller_Opt,LatErrTotal_Opt,exitflag,output,lambda,grad,hessian] = ...
fmincon(@Controller_ObjFun,x0,[],[],[],[],lb,ub,[],options);

disp('Initial Controller Gains:')
disp(x0);
disp('Optimum Controller Gains:');
disp(Controller_Opt);
fprintf('Optimum Lateral Error Total: %0.3fs \n', LatErrTotal_Opt);
```