Lap Time Simulation for Racing Car Design

by

Blake Siegler

Submitted in accordance with the requirements for the degree of PhD.

The University of Leeds

School of Mechanical Engineering

February 2002

The candidate confirms that the work submitted is his own work and that appropriate credit has been given where reference has been made to the work of authors.

Acknowledgements

I would like to take this opportunity to thank my supervisor Proffessor Crolla for his invaluable advice and support throughout my PhD and the other members of the Leeds Vehicle Dynamics Research Group, especially Richard King, Andrew Deakin and Warren Manning for their help and insight into all aspects of vehicle behaviour. Howard Ash, Adrian Simms, Scott Roberts and many other members of the Leeds Formula car team were invaluable in helping me prepare the car during the on track testing. Also worthy of a mention are the technical and support staff at the School, which especially includes Mick, Stan, Ian, both Pauls and Dave who assisted me with the logistics involved in my research and the construction of the data logging equipment. The vehicle testing was assisted by Leonardo from Entran, Koert and Friedes from TNO, John Grist from Datron, Chris Jones from Druck and Systron who provided equipment as well as Leyland Technical Centre, Finningly Airfield and Bruntingthorpe Airfield in the provision of suitable testing facilities. Lastly thanks to my father and Dani for all the proof reading.

Abstract

Racing teams use numerous computational tools (CAD, FEA, CFD) to aid in the design of racing cars and the development of their performance. Computer simulation of racing car handling through Lap Time Simulation (LTS) packages complements these tools. It also allows teams to examine the effect of different vehicle parameter setups to optimise vehicle performance. In similarity with the automotive industry, time is limited and rapid development of new ideas and technology is essential. Thus, the use of a more sophisticated computer simulation would allow a team to gain a significant advantage over their competitors.

As LTS are computationally intensive, previous packages have simulated a full lap using a quasi-static method which splits the path of the vehicle into segments. An analysis is then made of the vehicle at each segment point using the external forces acting on the vehicle. Due to the constant acceleration (i.e. steady state) assumption across each segment, this method does not take into account the effect of roll, pitch and yaw inertia as well as damping and tyre lag effects. Another aspect that is not accounted for is the variation in the fastest effective vehicle path along the circuit (i.e. racing line) due to change in driver control inputs or vehicle parameters.

The overall aim of this thesis is to develop a transient LTS methodology, which adopts a strategy to vary the racing line taken in order to address the problems found with the existing quasi-static LTS packages. In parallel an investigation of the accuracy of vehicle models in relationship to racing car performance has been developed.

The thesis begins with a study of racing car modelling techniques and a review of current LTS packages. A description is then given of the collection of vehicle

handling data from an actual racing car (along with attaining a vehicle parameter set) and the measured results displayed and discussed.

The creation of two vehicle models, a simple and sophisticated version, is detailed and the measured results are compared to the simulated results of each vehicle model. It was found that the simple vehicle model does not fully represent the actual vehicle's lateral dynamic behaviour, although its steady state response was deemed to be accurate. The sophisticated vehicle model was seen to not only accurately predict the full range of lateral dynamic behaviour of the actual vehicle, but also the actual vehicle's longitudinal and combined lateral and longitudinal dynamic behaviour.

To further investigate LTS techniques, a comparison study was made between various simulation approaches which indicated that the transient approach, although more complicated and time consuming, allows for more accurate tuning of a greater number of vehicle parameters.

Finally, the creation of two simulation packages has been detailed and case studies are presented to provide further insight into the look and feel of the packages. The first package is a quasi-static approach based LTS package, where a case study is made into the sensitivity of overall lap time to a range of vehicle parameters. The second is a transient approach based simulation package which optimises the driver controls, varying the racing line taken by the vehicle and ensuring the manoeuvre is completed in the quickest time for that vehicle parameter set. This final Manoeuvre Time Minimisation package fulfils the overall aim of the thesis and a case study is made into the effect of front damping value on manoeuvre completion time.

Table of Contents

List of Figures

List of Tables

Abbreviations

LTS	Lap Time Simulation
DOF	Degrees of Freedom
CPU	Central processing unit
IMechE	Institute of Mechanical Engineers
SAE	Society of Automotive Engineers
SQP	Sequential Quadratic Programming
QP	Quadratic Programming

Nomenclature

A: Acceleration, ms^{-2}

$A_{calliper}$: Total area of pistons in a calliper, m^2

A_{master}: Area of master cylinder, m^2

a: Centre of gravity distance from front axle, m (figure 3.2)

B: Stiffness factor for tyre magic formula equation

Balbar: Brake system balance bar ratio

$B(j\omega)$: Butterworth filter response

b: Centre of gravity distance from rear axle, m (figure 3.2)

C: Shape factor for tyre magic formula equation

C_{damp}: Damping value, Nsm^{-1}

C_l: Aerodynamic coefficient of lift

C_d: Aerodynamic coefficient of drag

D: Peak value for tyre magic formula equation

d: Preview point distance from centre of gravity, m (figure 3.11)

E: Curvature factor for tyre magic formula equation

e_n: Path error, m (figure 3.11)

e_ψ: Path heading error, rad (figure 3.11)

F: Force, N

FA: Frontal area of vehicle, m^2

g: Gravitational constant, ms^{-2}

h: Sprung mass centre of gravity distance from roll axis, m (figure 3.7)

I_{zz}: Yaw inertia of whole vehicle about a3 axis, kgm^2

I_{yyb}: Pitch inertia of sprung mass about a2 axis, kgm^2

I_{xxb}: Roll inertia of sprung mass about a1 axis, kgm^2

I_w: Spin inertia of wheel about its spin axis (axle), kgm^2

k: Spring stiffness, Nm^{-1}

k_ϕ: Sprung mass roll stiffness, $Nmrad^{-1}$

k_θ: Sprung mass pitch stiffness, $Nmrad^{-1}$

LLT: Lateral load transfer, N

l: Wheelbase of car, m (figure 3.2)

M: Moment, Nm

m: mass, kg

N: Normal force at tyre contact patch, N

n: Filter order

Pedal: Brake pedal force ratio

R_{track}: Path radius, m

r: Rotational velocity about g3 axis, $rads^{-1}$ (figure 3.2)

$r_{calliper}$: Radius of calliper, m

r_w: Radius of tyre, m

S_H: Horizontal shift for tyre magic formula equation

S_v: Vertical shift for tyre magic formula equation

s: Path distance, m

T: Torque, Nm

t: Half the value of track of axle, m (figure 3.2)

u: Velocity of vehicle in a1 axis direction, ms^{-1} (figure 3.2)

v: Velocity of vehicle in a2 axis direction, ms^{-1} (figure 3.2)

x: Variable

z_r: Roll centre height, m

α: Tyre slip angle, rad

δ_f: Steered angle of front axle, rad (figure 3.2)

ζ: Damping ratio

θ: Pitch angle of sprung mass, rad (figure 3.8)

κ: Longitudinal slip ratio of tyre

μ_{pad}: Coefficient of friction between brake pad and disc

μ_{rr}: Coefficient of rolling resistance for a tyre

ρ: Air mass density, kgm^{-3}

σ: Tyre lateral slip angle lag coefficient

ϕ: Roll angle of sprung mass, rad (figure 3.7)

Ψ: Vehicle heading angle, rad (figure 3.11)

ω_r Rotational velocity of wheel about axle, $rads^{-1}$

ω_c: Signal frequency

Subscripts :

0: Nominal value

b: At sprung mass

brake: Produced by brakes

drag: Tyre drag force

e: Produced by engine

f: At front axle

fl: Front left wheel

fr: Front right wheel

foot: Produced by drivers foot

inner: Inner wheel in a corner

n: Number of iterations

outer: Outer wheel in a corner

r: At rear axle

rl: Rear Left wheel

rr: Rear right wheel

t: Produced by tyre

x: In a1 axis (longitudinal) direction

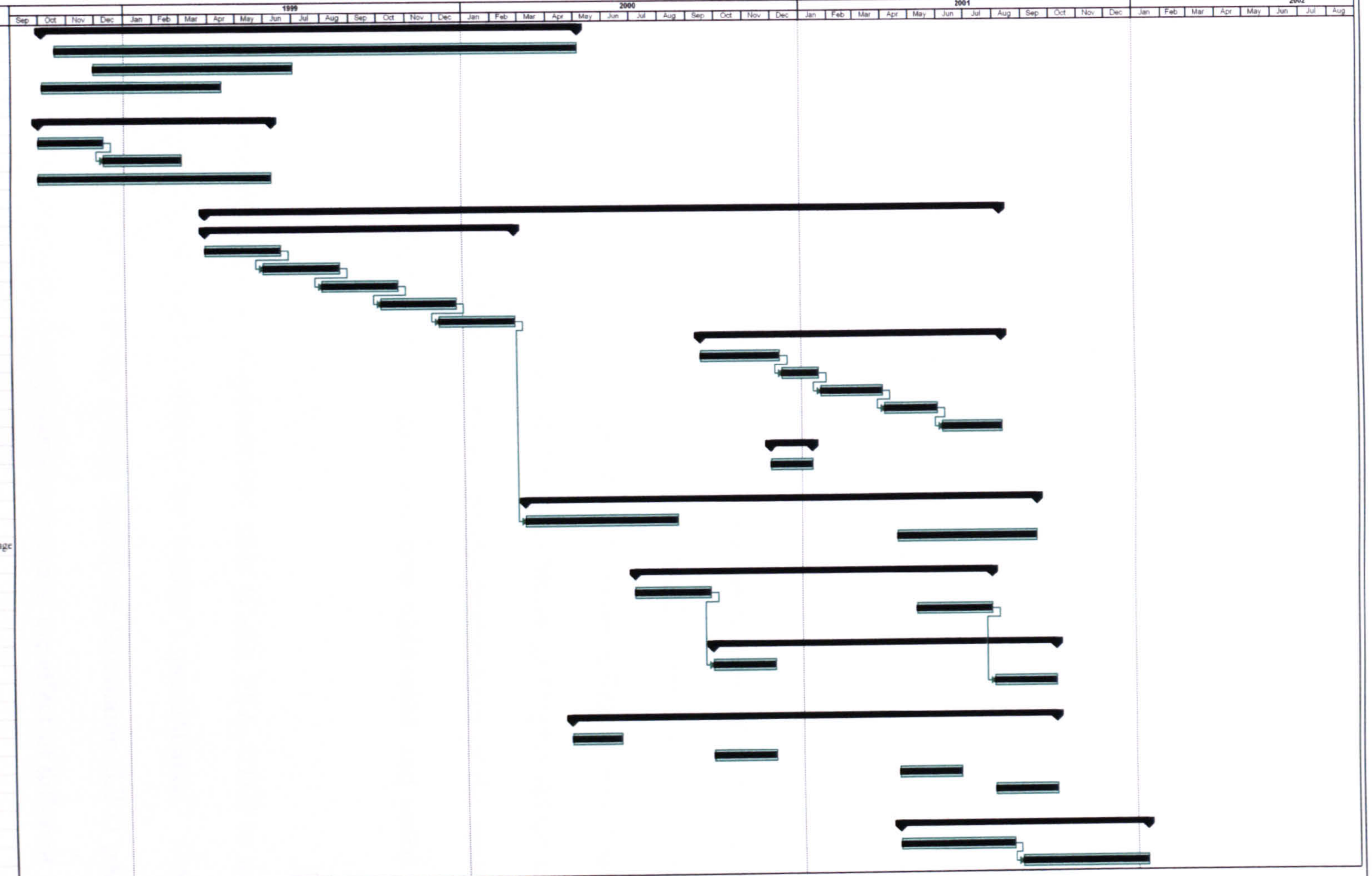
y: In a2 axis (lateral) direction

z: In g3 axis (vertical) direction

Axis System - the axis system used is the SAE standard vehicle and tyre axis system [1].

Project Plan

ID	Task Name
1	**Vehicle Dynamics Review**
2	Learn Techniques and Theories
3	Learn About Current Packages
4	Review Current Literature
5	
6	**Tools**
7	Look at Programming Tools Available
8	Learn Matlab and Simulink
9	Review Simulation Methods
10	
11	**Vehicle Models**
12	**Simple Vehicle Model**
13	Derive Equations of Motion
14	Create Tyre Model
15	Create Load Transfer Model
16	Create Drivetrain Model
17	Create Brake Model
18	**Sophisticated Vehicle Model**
19	Derive Equations of Motion
20	Improve Existing Tyre Model
21	Create Load Transfer Model
22	Improve Existing Drivetrain Model
23	Improve Existing Brake Model
24	**Path Following Preview Controller**
25	Develop Model
26	
27	**Lap Time Simulation**
28	Develop Quasi-Static LTS Package
29	Develop Transient Manoeuvre Time Minimisation Package
30	
31	**Vehicle Testing**
32	Phase One Testing
33	Phase Two Testing
34	
35	**Model Validation**
36	Simple Vehicle Model
37	Sophisticated Vehicle Model
38	
39	**Prepare Papers and Presentations**
40	FISITA Youth Conference
41	SAE Motorsports Conference
42	EAEC Congress
43	SAE World Congress
44	
45	**Final Review**
46	Update Literature Review
47	Write Up Thesis



Task Milestone Rolled Up Task Rolled Up Progress Project Summary Rolled Up Split
Progress Summary Rolled Up Milestone External Tasks Split

1 Introduction

1.1 Application of Vehicle Modelling to Racing Cars

The nature of motorsport is to test the performance of a vehicle and driver combination on different types of courses, be they circuits or short runs. The best combination is the one that produces the shortest overall time over a number of laps around each course. Motorsport is highly competitive and races are won on margins of hundredths of a second.

Racing cars are designed so that their components can be adjusted relatively easily in order to optimise the vehicle's setup on each individual circuit. There are normally a vast number of combinations that a team can use for each different vehicle setup and as such, getting the most out of the vehicle is a difficult and highly skilled task. In the past, this had been predominantly because the performance of the vehicle was only measured after the physical change had been undertaken and tested on the circuit.

A racing team uses numerous computational tools (CAD, FEA, CFD) to aid with construction of the car and to develop the vehicle's performance. Computer simulation of racing car handling, through Lap Time Simulation (LTS) packages, complements these tools and allows teams to examine the effect of different vehicle parameter setups to optimise vehicle performance. As with the automotive industry, time is limited and rapid development of new ideas and technology is essential. Thus, the use of a more sophisticated computer simulation would allow a team to gain a significant advantage over their competitors.

In similarity to other computational tools, LTS is useful in both the initial design stage and the development stage. In the initial design phase, fundamental vehicle parameters can be optimised which cannot be changed later, e.g. centre of gravity position (centre of gravity may only be moved by a few percent using ballast). During the development stage, however, it can be used to optimise changeable vehicle parameters, e.g. spring rates.

The use of handling simulations allows an expensive vehicle to be modelled at its limit of adhesion without risk of the vehicle being damaged or injury to the driver, as is the case with track testing. This does not mean that LTS completely replaces track testing, but complements it and can reduce the amount of time spent at the circuit.

1.2 Use of Lap Time Simulation (LTS) Packages

Typical use of vehicle handling and multi-body simulations involves studying the vehicle's behaviour when undertaking simple manoeuvres which may be well within the vehicle's limit of performance. LTS packages are similar to these, but they connect many manoeuvres together (to form a complete lap) and the vehicle is always travelling at its limit of performance. LTS is therefore an extension to the use of vehicle handling models and its main aim is to find the shortest possible time a vehicle with a certain parameter set can complete a given circuit.

LTS packages have to simulate many manoeuvres for each lap of the circuit and as such they are highly processor intensive. As the level of sophistication of the model increases, a greater amount of processing power and run time is needed. One of the limitations to increasing model sophistication has been the CPU power available in the portable computers used to run simulations at the circuit. Computing power is

increasing and its costs are decreasing at an exponential rate as predicted by Moor's Law [2]. As a result, the use of highly sophisticated simulation on portable computers has become possible and simulation run times have dropped significantly.

As these simulations are computationally intensive, previous packages simulate a full lap by splitting the path of the vehicle (found from recorded vehicle data) into segments (e.g., every 1m) and an analysis is made of the vehicle at each segment point using the external forces acting on the vehicle. This simple simulation approach, where the circuit is idealised as a series of straights and constant radius turns, is a quasi-static method.

Milliken et al [3] describe a commonly used method of finding the fastest lap time, which involves using the corners as limiting factors for the simulation. The maximum speed at which the vehicle can negotiate all the corners is found (which is independent of the straight speeds), therefore the speed the vehicle enters and leaves all the straights is known. From this, the vehicle's performance along the straights can then be found.

Due to the constant acceleration (i.e. steady state) assumption across each segment, previous studies [4] have shown that this method does not take into account the effect of the vehicle's roll, pitch and yaw inertia, as well as damping effects and tyre lag. Another aspect that is not accounted for is the variation in the fastest effective vehicle path along the circuit (i.e. racing line) due to change in driver control inputs or vehicle parameters. The method basically assumes that the racing line found from the actual vehicle data is the fastest in all cases.

1.3 Aim of Research

The overall aim of this research is to develop a transient lap time simulation methodology, which adopts a strategy to vary the racing line taken in order to address the problems found with the existing packages described above. In parallel an investigation of the accuracy of vehicle models in relationship to racing car performance will be developed. The main project objectives are listed below:

- Review current approaches to racing car modelling and LTS.
- Develop a vehicle model of a racing car and confirm its accuracy by comparison with measured results.
- Measure detailed performance data from an actual vehicle, which will allow vehicle models to be validated for all vehicle operating conditions.
- Create a LTS package that uses a quasi-static simulation approach.
- Identify key areas in which current packages could be improved.
- Develop the quasi-static package to incorporate transient effects and to include the ability to optimise the fastest line taken by the vehicle.

2 Literature Review

2.1 Historical Development

LTS techniques involve the analysis of a racing car's behaviour whilst accelerating, braking and cornering. This allows the optimisation of its performance to minimise the overall time taken to complete a lap on a given circuit. From the beginnings of motorsport, racing car teams have used measured sector and overall lap times to gauge vehicle performance. By 1954, Mercedes-Benz [5] created a simple hand calculation based LTS using sector times which were then aggregated to predict overall lap time. This produced a basic lap simulation with which vehicle performance could be predicted over the whole circuit for a range of different vehicle parameters.

Until the advent of accessible digital computing in the 1980's, racing car performance prediction, for the most part, was based around simple hand derived equations of motion which were increasingly being solved by digital computers. An example of this was published in 1971 [6] and is the first published example of the use of g-g diagrams [3] and the quasi-static simulation approach.

The quasi-static approach is defined as idealising the circuit as a series of segments consisting of straights and constant radius turns. The steady state solution to the vehicle's equations of motion is found at each of these track segments using the external forces acting on the vehicle. The quasi-static approach is normally used in conjunction with a method to find the fastest lap time, which involves using the corners as limiting factors for the simulation [3]. The maximum speed at which the vehicle can negotiate each corner's minimum path radius (i.e. the apex of each

corner) is found and thus the speed at which the vehicle enters and exits all the straights is known. The vehicle's performance along the straights can then be calculated.

In 1982, the first commercially available digital LTS 'RCSIM' was unveiled by Milliken Associates [3] which improved on earlier quasi-static packages by the use of a more sophisticated bicycle type vehicle model to estimate the vehicle's performance in each track segment. This package has seen continual development to the present day [3, 7].

Over the last decade, computer technology and software has improved rapidly. This has been followed by an increasing availability of vehicle dynamics, mathematical modelling and multi-body software packages, which can be used to automatically generate a vehicle's equations of motion and simulate its performance. Additionally, tyre manufacturers have been able to produce more detailed and accurate tyre performance data across a greater range of operating conditions. To make this data more readily accessible, robust empirical tyre models have been developed (e.g. the Pacejka Magic Tyre Formula [8]). All of these factors have allowed teams to produce highly sophisticated vehicle models with greater ease than was previously possible.

Over the last five years there has been a major increase in the number of LTS packages appearing. All but a few of these, however, have made use of the quasi-static simulation approach, which still allows a reasonably realistic prediction of vehicle performance. The most commercially successful and widely used, is the PiSim LTS package [2] which uses a generic vehicle model with only a few degrees of freedom (DOF). The main aim of the PiSim package is to enable teams to

minimise lap times and establish the sensitivity of the vehicle's performance to parameters changes.

A major improvement in LTS techniques has only come about in 2001 with Casanova et al [9, 10, 11] developing a LTS package based on a transient simulation approach. The transient simulation approach is defined as simulating the vehicle's performance using a continuous time history and dynamic solution to the vehicle's equations of motion. The transient approaches found in the literature optimise the vehicle's control inputs to find the minimum lap time achievable with a given vehicle parameter set. This allows a more realistic prediction of vehicle performance and further details are reviewed in section 2.4.

Development of LTS packages must be seen in the context of the more general area of vehicle performance analysis by computer simulation. It might be viewed as a subset, specialised to meet the needs of the racing car industry, whereas the majority of development has occurred in the broader automotive industry. Consequently, some of the general issues involved in the modelling of vehicle handling performance are first reviewed. These issues are then followed by an overview of the types of computer package used to generate the equations of motion in the vehicle model. This overview is then followed by a review of the development of packages specifically aimed at lap time simulation.

2.2 Review of Vehicle Modelling Issues

Vehicle modelling involves idealising the actual vehicle system as a set of equations of motion. These equations can then be solved in a simulation to generate a prediction of the vehicle's performance. The models used in LTS packages and the racing car industry are based on those developed for the modelling of general road vehicles. This section (2.2) has been split into five sub-sections discussing:

2.2.1 The vehicle model itself.

2.2.2 The external forces affecting the system – tyre models.

2.2.3 The external forces affecting the system – aerodynamic models.

2.2.4 The sub-systems creating the longitudinal forces – powertrain models.

2.2.5 The sub-systems creating the longitudinal forces – brake models.

A compromise has to be reached between model sophistication and accuracy because an over-sophisticated model can significantly decrease productivity [12]. Accurate and comprehensive parameter sets also need to be determined. This can become complicated, because as the sophistication of the model increases, the detail and number of parameters needed is also increased [13].

2.2.1 Vehicle Models

Dixon [14] describes a basic vehicle model where the vehicle is represented as a point mass with a single wheel below it. This model has two DOF which represent lateral and longitudinal acceleration. Unfortunately this does not represent the under/neutral/over steer behaviour of the vehicle as it yaws.

Milliken et al [3] illustrates that the yaw DOF can be added by extending the model to have two wheels, this is commonly known as a bicycle model as seen in figure 2.1. Both wheels on the front and rear axle are effectively combined into one. The vehicle's mass is assumed to be concentrated at its centre of gravity and a suitable value of inertia is used to represent the vehicle's yaw inertia.

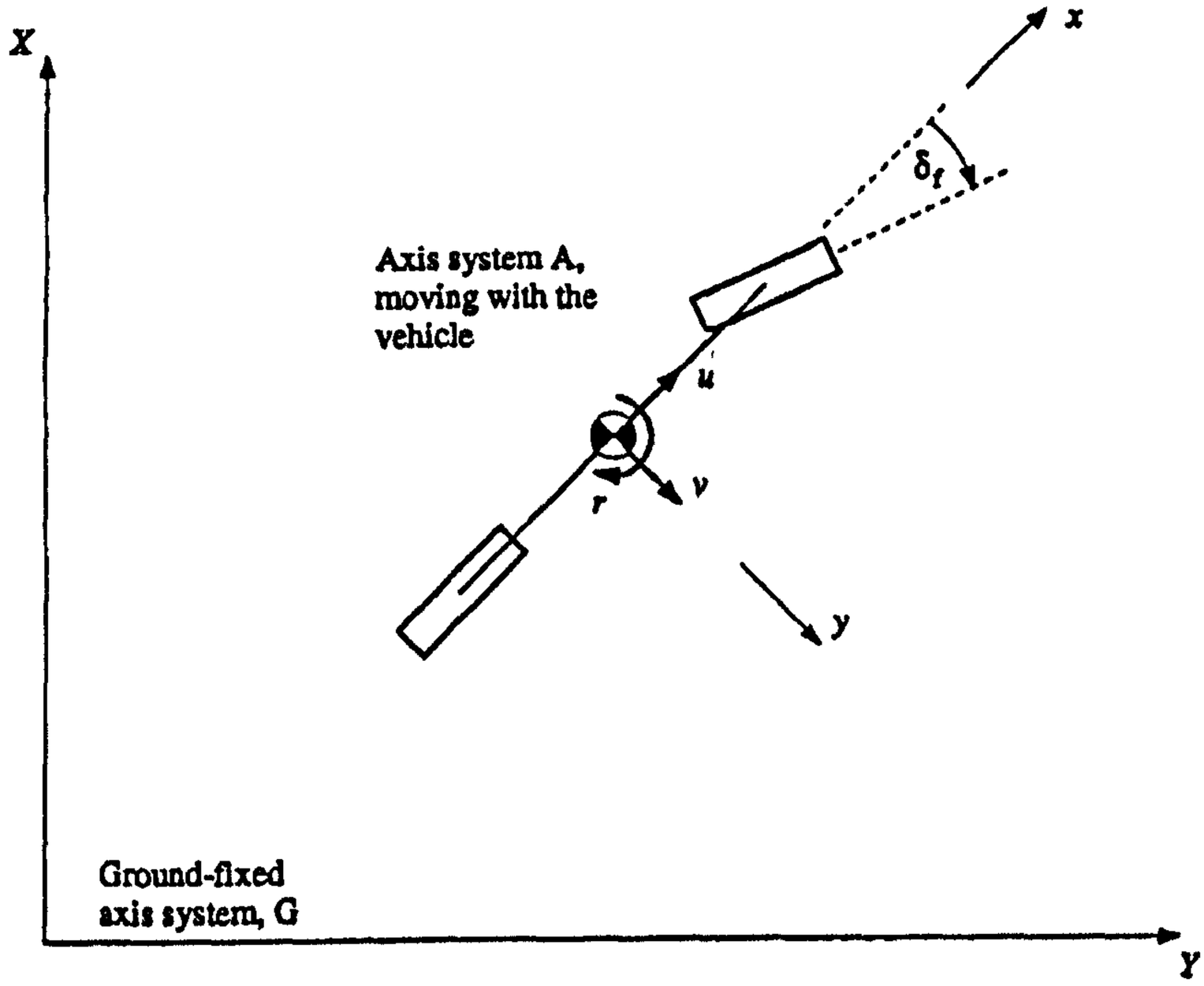


Figure 2.1 – Bicycle vehicle model

To improve its accuracy and give a better estimation of the slip angles the tyres encounter, the bicycle model can then be extended to a four wheel model by adding a front and a rear axle, each with two wheels. An approximation can then be made for the load transfer in lateral and longitudinal directions by using a quasi-static approximation [3].

An even better approximation of load transfer can be made by modelling the dynamics of the vehicle's sprung mass. This is carried out by adding roll and pitch DOF to the model. Ellis [15] demonstrates this by splitting the car into three bodies: the front unsprung mass, the rear unsprung mass and the sprung mass. The sprung

mass is pin jointed and free to rotate about the vehicle roll and pitch axis. Its motion is opposed by torsional springs and dampers on both axes, which relate to the vehicle's actual suspension components. Suitable values of roll, pitch and yaw inertia are then assigned to the sprung mass body, with the body masses again assumed to be concentrated at the centre of gravity position of each body. A vertical DOF may also be given to the sprung mass body to model its ride behaviour but, this is not normally an important factor when modelling racing cars running on smooth surfaces.

Crolla [16] shows that by making each wheel into a separate body with an extra DOF each for spin, camber and/or steer (once more with suitable inertia values), the modelling of powertrain/brake and suspension kinematics effects is possible. Using this approach, other effects may also be modelled such as steering system, axle and chassis compliance.

2.2.2 Tyre Models

To accurately predict the behaviour of a racing car, it is required to model the external forces acting on the vehicle as accurately as possible. At low speeds the main external forces acting on the vehicle are generated by the tyres. Racing cars are designed to operate at the peak of the tyre force curves. The tyre forces at limit handling, therefore, need to be modelled as accurately as possible.

There are two main types of tyre model employed in vehicle handling simulations. The first is the physical model, where the tyre's response is derived from first principles. These range in complexity from a single longitudinal and lateral spring, to springs mounted radial around a hub, to the most complex finite element models that include details of the air, rubber, tread pattern and fabric and steel cords used in

the tyres construction. A physical model is normally used in a vehicle ride model as the vertical response of the vehicle is of primary concern.

The second is the empirical model, where actual physical test data is used to describe the tyre's response. They are basically either a look up table or a function fitted to the measured data (which reduces the amount of data required). An empirical model is normally used in a vehicle handling model where the resultant lateral or longitudinal tyre force is of primary concern.

Considering that tyres are made from several different materials, with highly anisotropic material properties, their responses are very non-linear and it is difficult to derive an accurate and simple physical model [8]. The use of an empirical model, therefore, is the most efficient way to describe the tyre's response [13] in a vehicle handling model and will reduce simulation times.

One of the most commonly used empirical tyre models for vehicle simulation is the Pacejka Magic Formula method [8]. The model uses measured tyre data to find the coefficients of a mathematical function that matches the actual tyre's response as given in equation (1).

$$F = f\{\alpha, \kappa, N\} \qquad (1)$$

where:

α: Tyre slip angle, rad

κ: Longitudinal slip ratio of tyre

N: Normal force at tyre contact patch, N

F: Force, N

From equation (1) it is evident that the forces produced by the tyre are primarily due to three main external inputs:

- The slip angle at each tyre contact patch, which is dependent on the geometry of the vehicle and its longitudinal, lateral and yaw velocities.

- The normal forces at the tyre contact patch, which are equal to the sum of the static force on the tyre, the aerodynamic downforce on the axle and the lateral and longitudinal load transfer due to the sprung mass rolling and/or pitching.

- The longitudinal slip ratio of the tyre which is defined [8] as being the ratio of the velocity of the tyre contact patch to the velocity of the vehicle at that point and is dependent on the torque being applied to the tyre by the brakes or powertrain.

The camber angle of the tyre is another less significant factor and is the inclination of the longitudinal plane of the tyre to the road surface. This inclination produces a camber thrust [8] that offsets the lateral force produced by the tyre. The camber angle changes due to vehicle body rolling and/or bump and is controlled by the vehicle's suspension kinematics.

There are other factors that affect the tyre's response to these inputs and these include change in static inflation pressure, temperature and wear. These factors only have a minor influence on the tyre's performance and there will be uncertainty about their exact values at any given time [3].

The last factor that is important in making the model realistic is the time response of the tyre. Pacejka et al have extended the model to take this into account [8] and it

takes the form of a first order lag applied to the slip angle or slip ratio to represent the time taken to build up the forces in the tyre.

Other empirical models have been used to simulate tyre behaviour but these tend to contain large amounts of data [7] or are based on simplistic mathematical models [17]. Some empirical models have failed to take into account large slip angles and drivers have produced quicker lap times by 'sliding' the vehicle at high slip angles [7]. Other authors have produced tyre lag models but these are either the same as the Pacejka's model [18] or unnecessarily over-sophisticated [19].

2.2.3 Aerodynamic Models

The other set of external forces applied to the vehicle arises from the aerodynamic effects on its body as it moves through the air. This is normally modelled as a simple drag force acting at the centre of gravity and opposing longitudinal motion and two normal forces each acting at the centre of the front and rear axles (either upwards for lift or downwards for downforce). These forces are given by equation (2) and are dependent on the vehicle's frontal area and coefficient of lift or drag, which is measured in a wind tunnel or found using a CFD package [3].

$$F = \frac{1}{2}\rho FAC.u^2 \qquad (2)$$

where:

u: Velocity of vehicle in a1 axis direction, ms^{-1}

FA: Frontal area of vehicle, m^2

C: Aerodynamic coefficient of lift or Aerodynamic coefficient of drag

ρ: Air mass density, kgm^{-3}

An increase in sophistication is possible by using a parameter map to vary the aerodynamic coefficient depending on vehicle attitude or ride height. This method more accurately models the effects of the vehicle's aerodynamic devices which are sensitive to changes in these factors [20].

LTS packages have added separate models of individual aerodynamic devices to study the result of changing their design and orientation [21]. Unfortunately, modelling each device separately leads to inaccuracies when simulating the performance change to the whole vehicle. This is because the performance of an individual aerodynamic device is sensitive to the aerodynamic attributes of the entire vehicle [22].

2.2.4 Powertrain Models

To generate longitudinal force from a tyre, a slip ratio is created at the contact patch by applying a torque about the tyres axis of spin, causing it to spin at a speed different to that dictated by the vehicle's velocity [8]. This torque is produced by either the braking or the powertrain systems.

The simplest representation of the powertrain is a fixed maximum power value applied to the driven wheels [23]. To create a more detailed representation of the range of torques created by the powertrain, the engine can be modelled as a parameter map of torque against engine speed (found from experimental data) and the drivetrain modelled as a set of gear ratios of this torque. This model gives a high level of accuracy but with moderate sophistication to ensure efficient model running when compared to a full thermodynamic model of engine performance [24].

Additionally, the amount of torque that is applied from the engine is dependent on the driver throttle input. It is rare that the engine parameter map has been extended to include throttle position, as race engines are designed to produce the largest torque possible and thus it is assumed the driver will demand 100% throttle. Therefore the throttle value is normally modelled as a percentage of the maximum torque available at a given engine speed. Even though this is a linear approximation, it still makes a reasonable approximation of the actual non-linear engine response.

If large values of torque are produced by the engine or the powertrain rotational speeds are high, it may become necessary to model the powertrain's inertia or stiffness [25]. A differential (or differentials for more than one driven axle) can also be added into the model to account for the distribution of engine torque on an axle. Various types of differentials [3] have been modelled including locked, open or limited slip types (e.g. Salisbury, Torsen, etc.).

Many vehicles use control systems such as traction control, automatic gear selection, CVT and active differentials that limit the torque applied by the engine or vary the drivetrain ratios. These can also be modelled [26] as necessary.

2.2.5 Brake Models

All racing car braking systems use a hydraulic circuit and friction devices to transform the driver's control input of pedal force into a torque applied to each wheel. The brake pedal force is distributed between two master cylinders (one each for the front and rear hydraulic circuits) using a variable balance bar [3]. Brake discs with callipers or drum type systems can be simply modelled as friction devices [23] which derive axle torque from front or rear circuit pressures. Control systems (e.g. ABS) can also be modelled to limit torque applied to the wheel [27].

2.3 Computer Simulation Packages

All computer simulation packages relevant to multibody system dynamics problems use one of four different techniques to generate equations of motion to define the vehicle model. These techniques are described in table 2.1 along with an example of the main packages available [16]. As any LTS will involve the vehicle negotiating a considerable number of different straights and corners in each lap, the most applicable method will be the one that will allow the generation of not only an accurate but, efficient vehicle model that describes the vehicle's behaviour.

Technique	**Leading Software using this method**	**Method of Equation Generation**
Numeric Multibody codes	ADAMS	Equations are generated in numerical form.
Symbolic Multibody codes	Autosim	Equations are generated in symbolic form.
Purpose Built codes	Carsim	One generic model only available, as motion equations already defined.
Simulation Toolkits	MatLab	Equations of motion pre-defined by the user.

Table 2.1 – Methods of model generation.

Generating the equations using numeric multibody codes creates large and inefficient models of the system [13]. Suitable models have been produced using software packages such as ADAMS [28]. This study highlighted the fact that accurate models could be produced but demanded a considerable investment of time and money to develop a sophisticated model. Furthermore, a significant amount of computer processing power would be necessary, because ADAMS has to reconstruct the equations of motion for each parameter change [29].

Symbolic multibody codes are more efficient in generating the equations of motion used to define the vehicle model, which greatly reduces simulation times. The use of these software packages is limited and they have not really found widespread use throughout the industry. There are also no suitable purpose built codes available with a suitable racing car model, apart from inefficient numeric multibody based models.

As most racing cars are of similar design, a set of pre-defined equations of motion can therefore be defined by the author and utilised in a simulation toolkit. MatLab is at present an industry standard and it is relatively simple to create a vehicle model using user generated equations of motion. This software package has been used throughout the work described in the thesis. There are a number of toolboxes available for MatLab, such as symbolic programming, data acquisition and optimisation routines, which will help with the development of the vehicle models and LTS package.

2.4 Lap Time Simulation

LTS packages tend to fall into two different categories, either those that are commercially available or those that have been developed by academic institutions or individuals for personal use (non-commercial). All but a few of these packages use a quasi-static simulation approach. In addition to these packages there are others that are either not well publicised or are solely the property of certain racing teams [30]. These private packages are not available for use by outside parties due to the confidential information they contain and the secrecy associated with the competitive nature of motorsport. The vehicle dynamic models used in commercially available packages are also 'locked' away in the source code of the software. The implications

of this are that in order to ascertain the level of sophistication of existing LTS packages, review has been restricted to the information contained in the literature.

The main aim of any LTS package is to find the quickest possible time taken for a vehicle with a given parameter set to complete a lap of a circuit. The parameter set can then be varied to try to reduce this time. A LTS package works in the same way as many other engineering packages, involving three stages:

1. Firstly the package initialises by gathering vehicle parameters from the user and attaining circuit data (trackmap) from the user and/or measured vehicle data (i.e. downloaded from a data-logging system).
2. The lap simulation then takes place with the vehicle equations of motion being solved for the given vehicle parameter set and, using the quasi-static or the transient simulation approach, the minimum lap time is found.
3. Finally, the performance of the vehicle around the circuit is displayed to the user and compared with measured data, if required, as shown in figure 2.2.

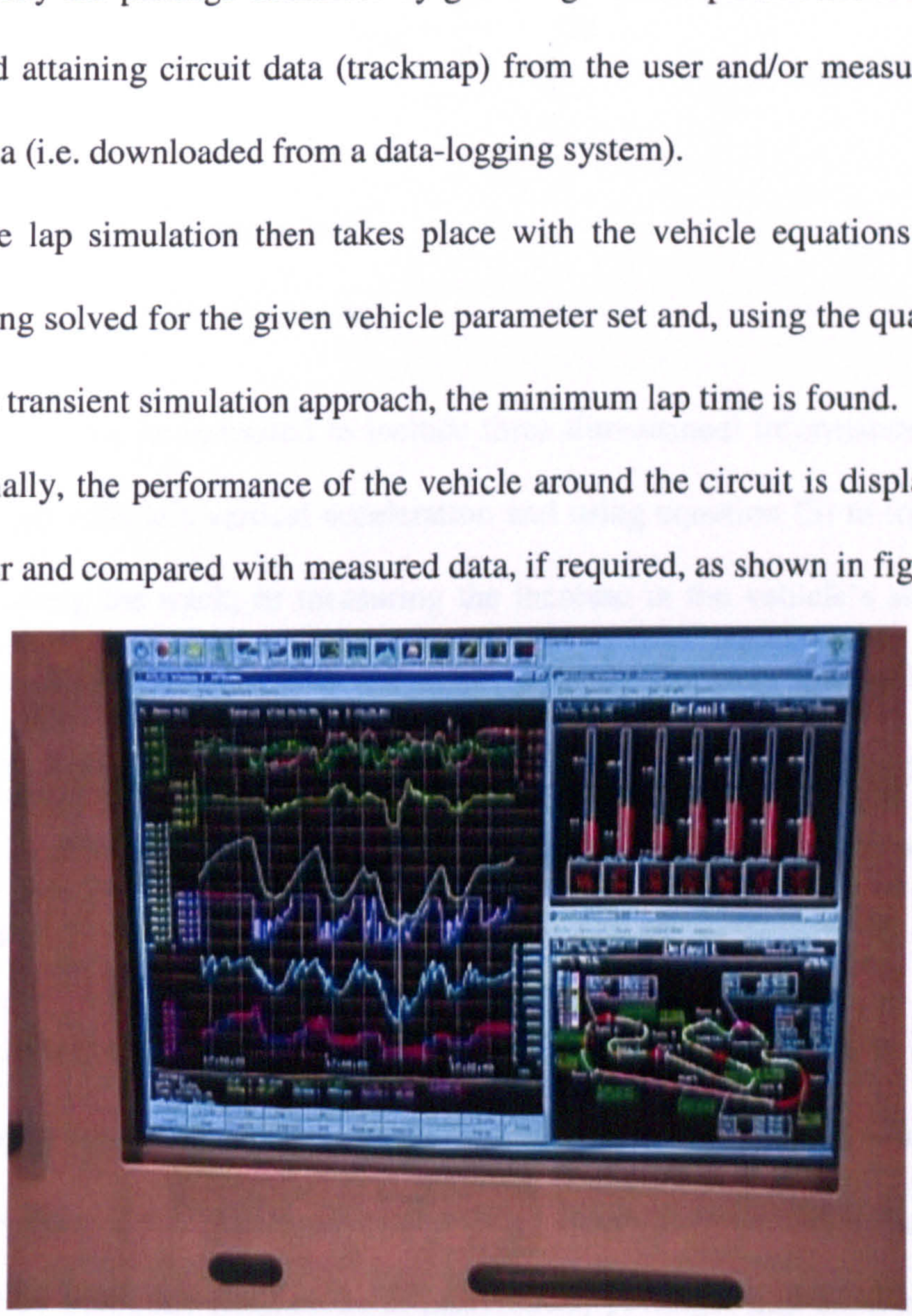


Figure 2.2 – TAG Heuer vehicle data package.

The circuit trackmap consists of a set of path radii at fixed distances around the circuit and is normally found from lateral acceleration and forward velocity measured data. As the data logging equipment records data at a fixed time step, each fixed distance circuit segment point is found by linear interpolation between fixed time steps using the forward velocity data. Each segment path radius can then be found again, involving linear interpolation, using the lateral acceleration data and equation (3).

$$R_{track} = \frac{u^2}{A_y} \qquad (3)$$

where:

A: Acceleration, ms^{-2}

R_{track}*:* Path radius, m

The track map can be extended to include three dimensional information, either by measuring the vehicle's vertical acceleration and using equation (3) to measure dips and crests along the track, or measuring the increase in the vehicle's static weight using pushrod load cells to give the track camber. To keep the track map two dimensional, these effects can be accounted for by increasing the tyre friction coefficient in this area of the track. Data can also entered by the user and may be found from other sources, e.g. survey data.

A detailed description of the quasi-static simulation approach is given in sub-section 2.4.1 as background to the LTS packages that are reviewed in the following two sub-sections 2.4.2 and 2.4.3. These have been split into commercial and non-commercial (developed by academic institutions or individuals) packages and are listed in order of increasing sophistication for each section. A detailed description of the transient

simulation approach that is used by the most sophisticated non-commercial LTS package will be made in the last sub-section.

2.4.1 The Quasi-Static Simulation Approach

Most LTS packages begin by finding the minimum path radius at each corner in the trackmap. This defines the corner apex (and minimum speed) points [3]. Steady state acceleration is assumed across each segment. At this corner apex point, therefore, it is assumed that the vehicle is maximising its lateral acceleration (i.e. not using any tyre grip to produce longitudinal acceleration). Its forward velocity at that point is then found by rearranging equation (3) and using the vehicle model.

Care is taken to check whether the vehicle's performance in the corner is not power limited. This occurs when the engine power available is not great enough to overcome aerodynamic and tyre losses [14] and if this is the case the corner apex performance is limited by engine power.

From this apex point the simulation increases forward velocity backwards down the previous straight (backwards marching) and forwards along the next straight (forwards marching) in steady state segments [31]. The time taken for the vehicle to complete each segment is minimised (i.e. greatest longitudinal acceleration) using a friction circle approach [3].

The friction circle approach assumes that any potential tyre grip not utilised in producing lateral acceleration can be used to produce longitudinal acceleration. This potential grip is found by drawing a plot of lateral force against longitudinal force for each tyre (which tends to be an ellipse) to find the available longitudinal force given lateral force being produced. So as path radius decreases away from the corner apex,

increasing amounts of tyre grip can be utilised in braking or engine acceleration. As in real life, the maximum amount of longitudinal force produced can be limited by the brake system design or available engine power.

The steady state solution of the dynamic vehicle model equations for each segment is then found before moving on to the previous or next track segment. To save solving the same steady state equations more than once, parameter maps of vehicle performance (e.g. g-g diagrams) under all possible operating conditions can be employed [32]. These parameter maps are normally found before the main lap analysis, in the initialisation phase.

The distance point on the straight where the backwards marching from the next corner meets the forwards marching from the previous corner is the straight's maximum velocity point. This is found for all the straights in the circuit and the data collated into a continuous matrix which is graphically displayed to the user. The sum of all segment times is equivalent to the total minimum lap time for that vehicle parameter set.

Due to the constant acceleration assumption across each segment, previous studies [4] have shown that this method does not take into account the effect of roll, pitch and yaw inertia as well as damping and tyre lag effects. Another aspect that is not accounted for is the variation in the fastest effective vehicle path along the track (i.e. racing line) due to change in driver control inputs or vehicle parameters. The method basically assumes that the racing line found from the actual vehicle data is the fastest in all cases.

2.4.2 Review of Commercially Available LTS Packages

Listed below, in order of increasing sophistication, are LTS packages that are presently available commercially. Most packages have seen widespread use in the lower levels of motorsport, with only PiSim and RaceSim being used in the top echelons of motorsport, where PiSim is, by far, the most widely used. All of the packages, apart from the ADAMS package, detailed in this section use the same quasi-static simulation approach detailed above to find the minimum lap time.

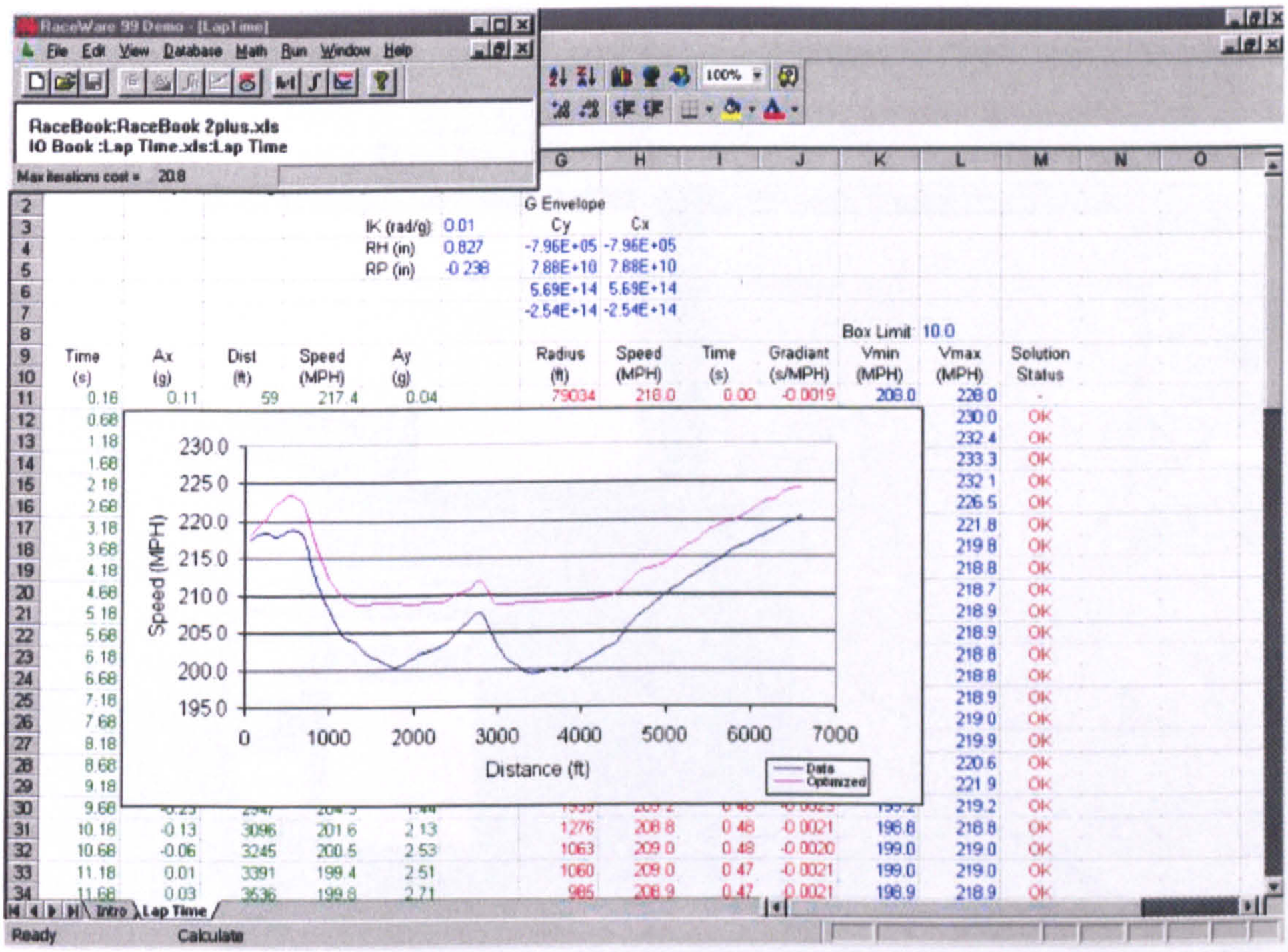


Figure 2.3 – RaceWare LTS package.

RaceWare [33] produced by Vehicle Dynamics Performance Ltd – This package uses Microsoft Excel to handle all the data with a separate Visual Basic executable to control the simulation as shown in figure 2.3. The vehicle model used includes:

- Simple bicycle model, with quasi-static load transfer approximation.

- Tyre model is based on Pacejka Magic Formula.
- Single aerodynamic coefficients for front/rear lift and overall drag.
- Powertrain in form of engine parameter map and gear ratios.
- Brake system is idealised to use maximum traction available.
- Two dimensional trackmap.

Unfortunately, no attempt is made by the literature to judge the accuracy of the results.

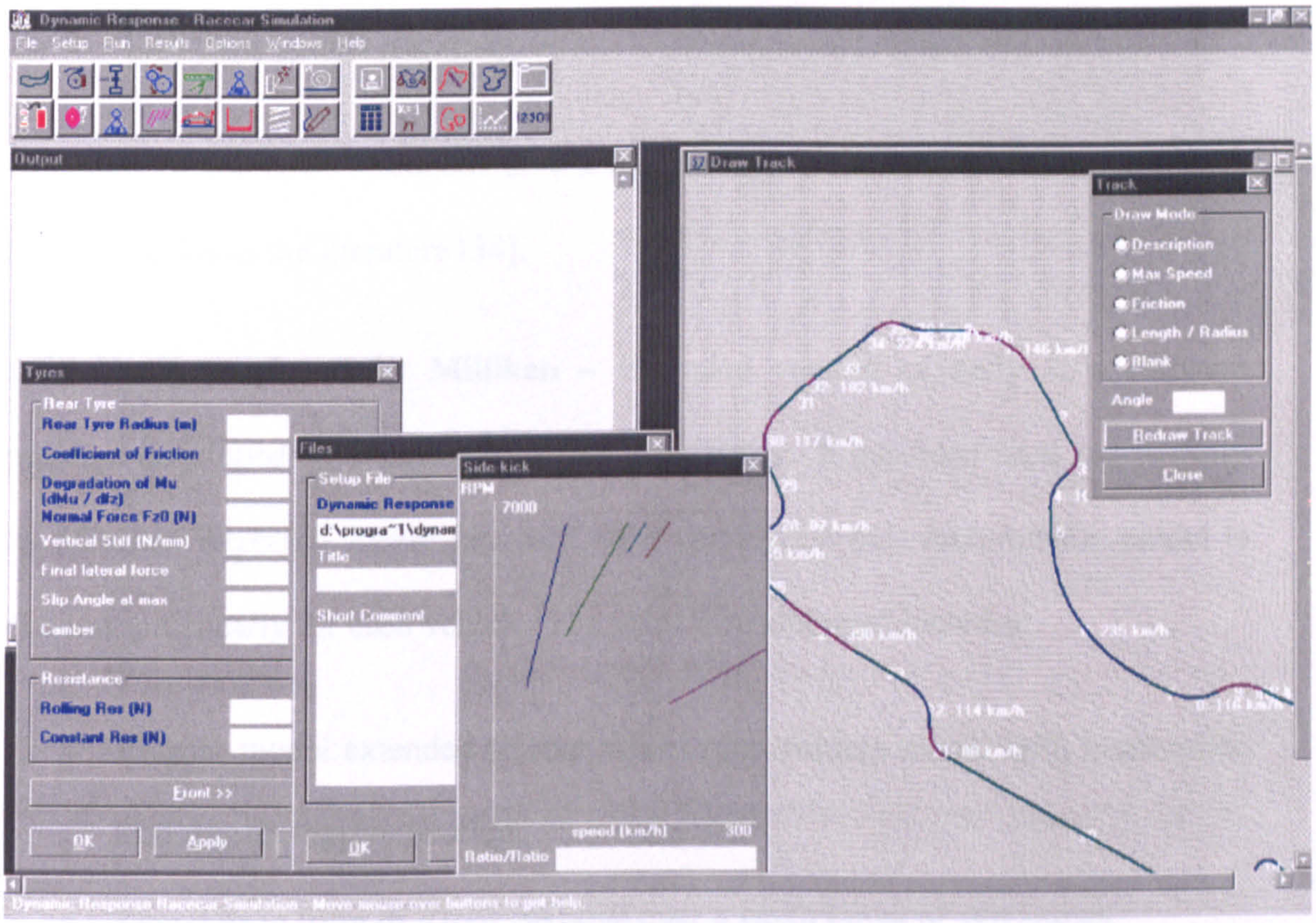


Figure 2.4 – Dynamic Response LTS package.

Dynamic Response [34, 35] produced by Pressplay Ltd – Used successfully by both Formula Ford and Touring Car teams around the world, see figure 2.4. The vehicle model used includes:

- Bicycle model extended to four wheel type vehicle model with quasi-static load transfer approximation.
- Tyre model is based on Pacejka Magic Formula.
- Aerodynamic coefficients can vary with change in ride height.
- Powertrain in form of engine parameter map and gear ratios. Differential model is included along with front or rear wheel drive models.
- Brake system is modelled in detail and takes into account brake heating and fade.
- Two dimensional trackmap.

Accuracy of 2% (2 or 3 seconds on a 2 minute lap) is claimed, which is backed up with examples in the literature [34].

LTS [3, 7] produced by Milliken – Extended version of early 80's software package developed by Milliken Research Associates. It has been used successfully by Milliken Research Associates with good correlation but, each vehicle model is created specifically for each vehicle. The vehicle model used includes:

- Bicycle model extended to four wheel type vehicle model with quasi-static load transfer approximation.
- Tyre data in form of a look up table over a small range of slip angles.
- Aerodynamic coefficients vary with change in ride height.
- Powertrain in form of engine parameter map and gear ratios. Differential model is included along with front or rear wheel drive models.
- Brake system is modelled in detail.
- Suspension kinematic effects included in the model.
- Three dimensional trackmap.

The literature shows a good correlation between measured and simulated results but, the simulation results were found to be lacking in some cases due to the limited range of slip angles employed in the tyre data [7]. No scales are given on the figures or values given, so it is assumed, due to the model sophistication, that the accuracy is of a similar order to the Dynamic Response package [34].

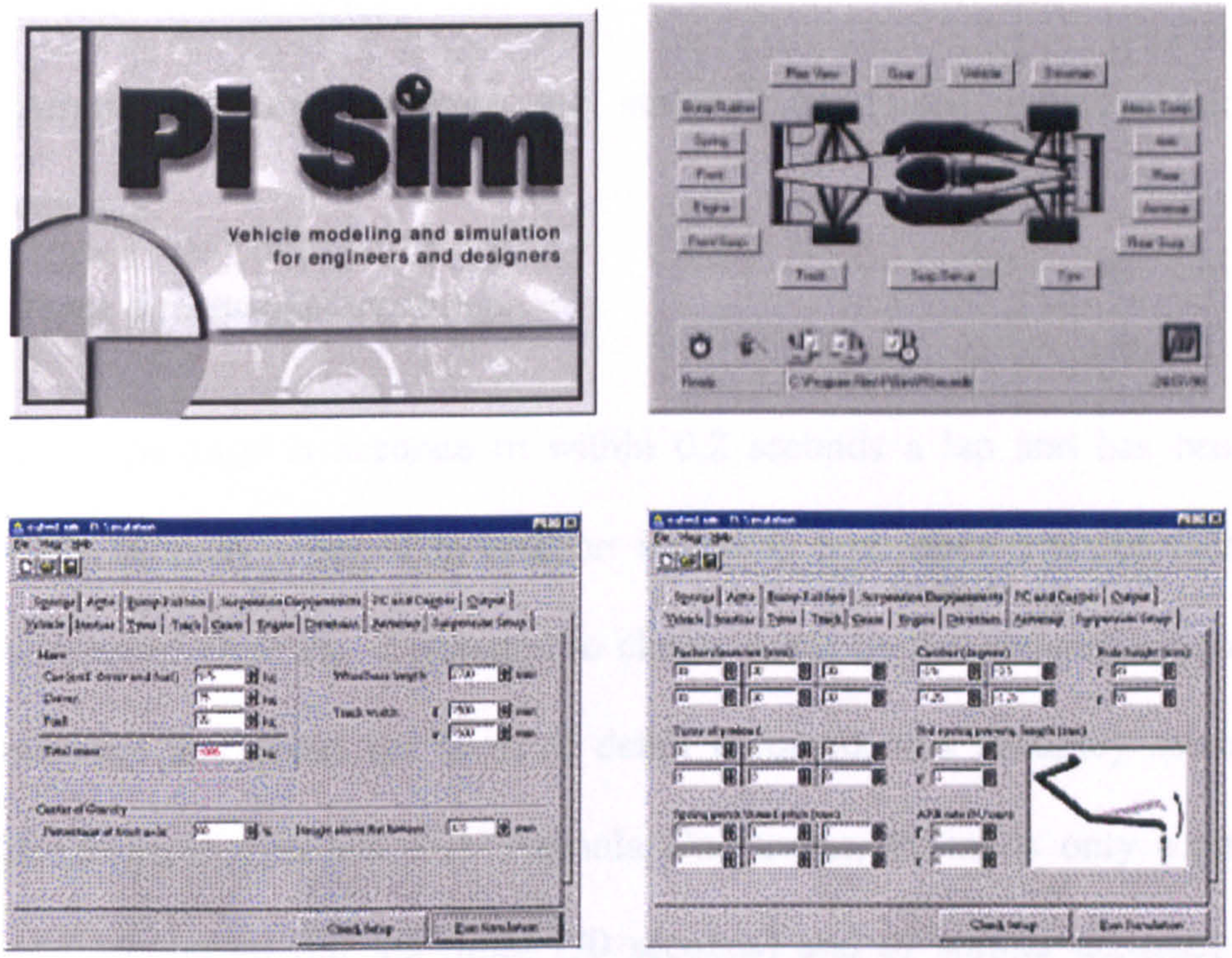


Figure 2.5 – PiSim LTS package.

PiSim [2, 36, 37, 38, 39] produced by Pi Corporation – Original version of PiSim, was released in early 1997. Pi has spent a significant amount of time on the Visual Basic user interface, see figure 2.5, and has kept simulation times short (under a minute) by encoding the main solution processing with the C programming language.

The vehicle model used includes:

- Bicycle model extended to four wheel type vehicle model with quasi-static load transfer approximation.
- Tyre model is based on Pacejka Magic Formula.

- Aerodynamic coefficients vary with change in ride height.
- Powertrain in form of engine parameter map and gear ratios. Differential model is included along with front or rear wheel drive models.
- Brake system is modelled in detail.
- Suspension kinematic effects included in the model, as well as asymmetric chassis characteristics.
- Atmospheric compensation for ambient conditions (aerodynamics and engine).
- Three dimensional trackmap.

Pi claim the package is accurate to within 0.2 seconds a lap and has been used successfully in many types of motorsport including both open wheeled racers and production based vehicles. Compared to claims made by the companies producing other packages and given the level of detail involved, this accuracy seems very doubtful. From discussions with Formula One teams, PiSim is only accurate to within one second on full lap times (90 seconds) and of similar accuracy to the Dynamic Response package [34].

RaceSim [31, 40] produced by DATAS – Two simulations are possible, a standard quasi-static simulation approach employed for lap time minimisation whilst investigating parameter sensitivities and a transient simulation option that uses measured driver control inputs to 'drive' the vehicle around the circuit, in a dynamic simulation. Although this approach calculates the transient vehicle response to driver inputs, its main shortcoming is that it does not allow effective vehicle parameter sensitivity studies in relation to minimum lap time, as vehicle performance is not optimised in relation to the parameter change due to vehicle control inputs

being fixed. A sophisticated user interface is provided [41], with short simulation times and the vehicle model used includes:

- Seven DOF, four wheel vehicle model, which includes a vehicle lateral, longitudinal and yaw DOF and vehicle body ride, pitch, roll and chassis compliance in roll (gives front and rear chassis roll) DOF.
- Tyre model is based on Pacejka Magic Formula, with first order tyre lag included.
- Aerodynamic coefficients vary with change in ride height.
- Powertrain in form of engine parameter map and gear ratios. Several differential models available, which include front/rear/all wheel drive models. An ABS, traction control, active differential and automatic gear selection models are also available.
- Brake system is modelled as maximum torque that can be applied to wheel.
- Suspension kinematic effects included in the model, as well as asymmetric chassis characteristics. A non-linear bump rubber can also be applied to any wheel.
- Atmospheric compensation for ambient conditions (aerodynamics and engine).
- Three dimensional trackmap.

The package attempts to address the failings of the quasi-static simulation approach by approximating the effect of damping on vehicle performance. The approximation is carried out by assuming that on an actual vehicle, the ideal damping value would hold the tyre contact patch against the ground allowing the tyre to produce the maximum force possible (as measured using steady state tests, which produce the

empirical tyre data [8]). An estimation of the vehicle's damping performance against an idealised damping value is made using a very simple quarter car model, which is found in an initial simulation. This 'grip modifier' is then used in the main quasi-static approach based LTS to degrade the force the tyre produces. This makes some attempt to address the influence of the dampers on vehicle performance but, still does not model the more important affect of sprung mass dynamics (i.e. the affect on corner entry or exit performance of dynamic load transfer). Unfortunately, no actual values of accuracy are given in the literature, but judging from its widespread use and high price tag, it probably attains a level of accuracy similar to the Dynamic Response package [34].

ADAMS Racecar Module [28, 29] produced by Mechanical Dynamics Corporation – Mechanical Dynamics have produced a generic racing car vehicle model for their numeric multi-body dynamics package, ADAMS. This has been used by several teams successfully for LTS. A quasi-static simulation approach is not used, but instead, a driver model is used to estimate the optimum racing line around a circuit. This is done using the trackmap found from circuit survey data and geometrically finding the racing line that produces the smallest change in path radii. The driver model then attempts to follow the optimum racing line at the limit of the vehicle's performance. The generic ADAMS racing car vehicle model includes:

- Detailed multi-body vehicle model including all suspension components and some compliances, 53 DOF in total.
- Tyre model is based on Pacejka Magic Formula.
- All suspension kinematics and non-linear effects are included in the model, as well as asymmetric chassis characteristics.

Features that are possible to add by the user, using the ADAMS simulation package:

- Aerodynamic coefficients vary with change in ride height.
- Powertrain in form of engine parameter map and gear ratios, including open or locked differential models.
- Brake system is modelled as maximum torque that can be applied to wheel.
- Two dimensional trackmap.

These additional features are not straight forward enough to include in the vehicle model and require substantial experience to produce. ADAMS is also highly processor intensive and any simulation can take a significant amount of time. Although, a close correlation with actual data has been seen in the literature, no specific details were supplied or claims made [28].

2.4.3 Review of Non-Commercial LTS Packages

The LTS packages described below have been developed by academic institutions and individuals as an investigation of a special case of vehicle dynamics. Most of the packages do not contain a graphical based user interface and all but three (HP-VEHSAP, North Carolina State and Cranfield package) use the quasi-static simulation approach. The same vehicle model detailed below is used in each case with exceptions detailed:

- Simple bicycle model extended to a four wheel model, with quasi-static load transfer approximation.
- Tyre model is based on Pacejka Magic Formula.
- Single aerodynamic coefficients for front/rear lift and overall drag.
- Powertrain in form of engine parameter map and gear ratios.

- Brake system is idealised to use maximum traction available.
- Two dimensional trackmap.

Michigan University, USA [17] – A simple package created in Fortran and specifically for an Indy racing car. It is a simplistic package designed for a simple oval course and makes some observations about comparing parameter changes. The literature makes no attempt to judge the accuracy of the results.

Dominy et al [21] – A simple LTS package developed in the early 1980's to study aerodynamic effects. Some effort is made to validate the simulation results, which shows a reasonable level of accuracy, even with a linear tyre model.

La Joie [42] – A simulation created using Fortran and, again, a detailed simulation is undertaken using a simplistic vehicle model. The literature makes no attempt is to judge the accuracy of the results.

HP-VEHSAP [43] – In 1996, a comprehensive package was produced by Lugus Vehicle Technologies. Front/rear and all wheel drive powertrain models and a differential model are included. The LTS does not use the quasi-static simulation approach but finds the transient solution by employing a driver model to follow an optimum racing line at the limit of the vehicle's performance. Regrettably the driver model is not very effective in doing this and some oscillation in the vehicle path is seen. Once again the literature makes no attempt to judge the accuracy of the results.

LapSim [44] by Reynard – An example of an 'in house' developed package developed for use mainly with Indy car teams. Although the package is not commercially available it has seen use with many teams and has a sophisticated user interface and short simulation times. Vehicle model also includes:

- Locked differential model is included.
- Suspension kinematic effects included in the model, as well as asymmetric chassis characteristics.
- Three dimensional trackmap.

No example results are published.

University of North Carolina State, USA [45, 46] – The vehicle model is specifically for a NCSU Legends racing car, which is raced on a short oval circuit. This does not use the quasi-static simulation approach instead, a path and speed following driver model is used to follow a racing line and speed profile defined by the user. An optimisation routine is included, which attempts to vary the driver longitudinal control inputs to minimise the lap time whilst not deviating from the prescribed path by too much. The vehicle model also includes:

- Vehicle body roll, pitch and ride DOF.
- Suspension kinematic effects included in the model, as well as asymmetric chassis characteristics.
- Three dimensional trackmap.

The package is not very accurate as details of overall lap time indicate a difference of 3 seconds on a 20 second lap. An unsuccessful attempt was made in the literature at vehicle parameter optimisation (see section 2.5) and produced little change in lap time or parameter values.

University of Brescia, Italy [32, 47] – The latest package, detailed in reference [32], is similar in sophistication to the PiSim package with an advanced user interface and short simulation times. The vehicle model also includes:

- Aerodynamic coefficients vary with change in ride height.
- Differential model is included along with front or rear wheel drive models.
- Brake system is modelled in detail.
- Suspension kinematic effects included in the model, as well as asymmetric chassis characteristics.
- Three dimensional trackmap.

The vehicle's cornering performance is found before the main lap simulation, to reduce run times, by forming parameter maps of the vehicle's performance in the form of g-g diagrams. No attempt is made in the literature to judge the accuracy of the results, but due to its sophistication, it must be of a similar level of accuracy as the Dynamic Response package.

Cranfield [9, 10, 11] – The most significant progress so far has been made by Casanova et al who, in 2000, produced a series of papers describing work with the Benetton Formula One team in the production of a fully transient LTS package. The package used an optimisation routine which varied the vehicle's control inputs to minimise the time taken around the track. A more in-depth description of this simulation approach is given in sub-section 2.4.4. The vehicle model also includes:

- Limited slip differential model is included.
- Brake system is modelled in detail.

Results are given in the literature [11], which show a close correlation to measured data. At present this package seems to offer the most accurate approximation to the actual vehicle's performance and addresses some of the problems of previous simulation approaches.

2.4.4 The Transient Simulation Approach

This approach uses a non-linear numerical optimisation routine to adjust a driver control matrix. Whilst trying to minimise the manoeuvre time, the optimisation routine also ensures that the vehicle remains inside the track boundaries and limits the control inputs to realistic values [9, 10, 11].

The driver control matrix consists of a steer wheel angle control value and a longitudinal control value that is initiated at regular distances along the manoeuvre (this is effectively a look-up table against distance of control values). The longitudinal value controls either throttle position or brake pedal force (as the driver does not usually press both pedals at the same time), i.e. It ranges from a value of +1 = 100% throttle and no brake pedal force to 0 = 0% throttle and no brake pedal force to –1 = 0% throttle and full brake pedal force. Both control values are limited to within realistic boundaries (maximum steer angle, maximum throttle, driver pedal force). The simulation approach method of operation is detailed in table 2.2.

Even though a set of control points is used to optimise the performance of the vehicle that are at discrete distance points along the track, these points are then described in the form of a look up table and so form a continuous control time history for the vehicle. This means that the vehicle is continuously accelerating and its performance at any one point along the track is dependent on its performance in the previous point on the track due to damping, inertia and tyre lag effects. The vehicle racing line is also adjusted (as long as it remains inside the track boundaries) to the optimum line for the driver control inputs used for that vehicle parameter set. Therefore, this addresses many of the shortcomings of the quasi-static simulation approach.

1. User specifies initial velocity, control point density, vehicle parameters, vehicle control limits and track layout. Track layout is defined as a look up table against distance of centre line and track width at regular intervals.
2. Circuit and control matrix are split into series of shorter manoeuvres (up to two or three corners in each manoeuvre) and each one is solved in turn.
3. Optimisation routine begins, which is a constrained non-linear optimisation routine [48]. For each manoeuvre:
   a. User provides initial racing line and a path following, non-linear preview controller is used [9] to produce an initial guess for the driver control matrix (either line or initial control inputs can be taken from actual vehicle data).
   b. Run vehicle simulation with initial guess, vehicle simulation stops when reaches end of manoeuvre and returns time taken to reach this point.
   c. Adjust control matrix and re-run vehicle simulation and check whether time taken to complete manoeuvre is reduced.
   d. Run constraint routine to check whether the previous solution remains within the track boundaries, if not solution is disregarded.
   e. Repeat (c) and (d) until no further improvement in time taken to complete manoeuvre is found and optimisation routine ends.
   f. Vehicle parameters at end of manoeuvre (e.g. forward velocity, etc.) are used for initial values at beginning of next manoeuvre.
4. Collate all manoeuvres into a single matrix and display final simulation results. Total lap time is the sum of individual manoeuvre times.

load data
initialise
run
end
Next Manoeuvre
Next Iteration

Table 2.2 – Transient simulation approach method of operation.

2.5 Parameter Optimisation

A further level of sophistication is possible with the optimisation of the vehicle parameters used in the LTS package. These are varied by a parameter optimisation routine which reruns the LTS for each combination, instead of leaving it for the user to do by trial and error. As there are many definable parameters with each having a large set of possibilities for a given vehicle setup, it is only practical to find the optimum combination of a few parameters for any given run.

There are several methods by which parameter optimisation can be conducted. The most basic involves the user defining a set of parameter combinations for the routine to run through [42] (i.e. a batch file approach). A simple extension to this is in the application of design of experiment [49] or Taguchi matrix [50] techniques to these batch files.

Other authors have utilised Pareto-minimum analysis techniques to find the optimum vehicle setup. This was done by Kasprzak et al [51] for a range of longitudinal centre of gravity positions and roll stiffness distributions. The minimum time was found for the vehicle to complete a small and a large steady state circle for every possible combination of these two parameters. The Pareto-minimum parameter set was found to be the set which gave the smallest total time for both circles. This was extended by Hacker et al [52] to include a range of aerodynamic downforce distributions as well. In addition, Miano et al [53] approximated an eighteen DOF vehicle model using a Neural Network (to save computation time) and optimised a range of parameters using standard ISO test manoeuvres. All these studies reflect the large time scales involved in searching through every possible parameter

combination and consequently this method is highly inefficient when compared to numerical optimisation routines [48].

Yearstretch [54] has applied a genetic algorithm based numerical optimisation routine to complete the task of gear ratio parameter optimisation for an entire lap of a circuit. This was achieved with the use of a separate Excel program which executes the Dynamic Response LTS package [34, 35] with the various parameter sets. It was seen that this technique efficiently found an optimum solution but was still highly time consuming.

2.6 Conclusions

This chapter has not only summarised the literature for the existing level of technology available in LTS packages, but also reviewed the state of racing car modelling and simulation. The following conclusions have been reached:

- The use of LTS packages has rapidly expanded in the last five years.
- Vehicle performance can be estimated with many levels of model sophistication but, as LTS packages are computationally intensive, it is necessary to find the minimum sophistication of vehicle model that will provide a suitably accurate solution.
- The most suitable tyre model is the empirical Pacejka Magic formula model.
- Simple aerodynamic models based on measured coefficients provide appropriate levels of accuracy.
- Powertrain models normally consist of maps of engine torque produced and the driveline as a series of ratios. If necessary, additional systems can be modelled, including differentials and control systems.

- Braking systems can be modelled as hydraulic systems using friction devices.
- MatLab is the most effective software package to base a LTS package on due to its flexibility, extra toolboxes and author familiarisation with the package.
- The quasi-static approach of using steady state assumptions across each track segment implies that the effect of roll, pitch and yaw inertia as well as damping and tyre lag is not taken into account. In addition, the use of only the measured vehicle line causes inaccuracies.
- The transient approach accounts for some of the deficiencies of previous LTS packages even with the use of discrete control points.
- It is possible to use parameter optimisation routines in addition to an LTS package but it greatly increases the time for each investigation.

3 Racing Car Modelling

3.1 Introduction

As shown in the previous chapter, many levels of model sophistication are possible, however, as LTS packages are computationally intensive, it is necessary to find a suitable trade off between model sophistication and the quest for accuracy. As a consequence it was decided to begin with the creation of a simple seven DOF (longitudinal, lateral, yaw and four wheel spin DOF), four wheel model. This utilises a Pacejka combined slip tyre model and a quasi-static approximation for normal force at the contact patch. It also accounts for aerodynamic lift and drag and tyre rolling resistance.

Whilst comparing the simple model against actual data, it was found that inaccuracies occurred [55]. Therefore, this simple model has been extended to a more sophisticated thirteen DOF (longitudinal, lateral, yaw, roll, pitch, four wheel spin, four tyre lag DOF). Again it is a four wheel model and utilises a Pacejka combined slip tyre model (but with lateral tyre lag) and also accounts for aerodynamic lift and drag and tyre rolling resistance. Both vehicle models are fully described below.

The vehicle models described below have been created using MatLab and Simulink software for rapid model development, short simulation runtimes and efficient display of results. The vehicle models are created in Simulink and figure 3.1 shows how their modular design allows for the development of individual components (e.g. engine, tyres, etc.).

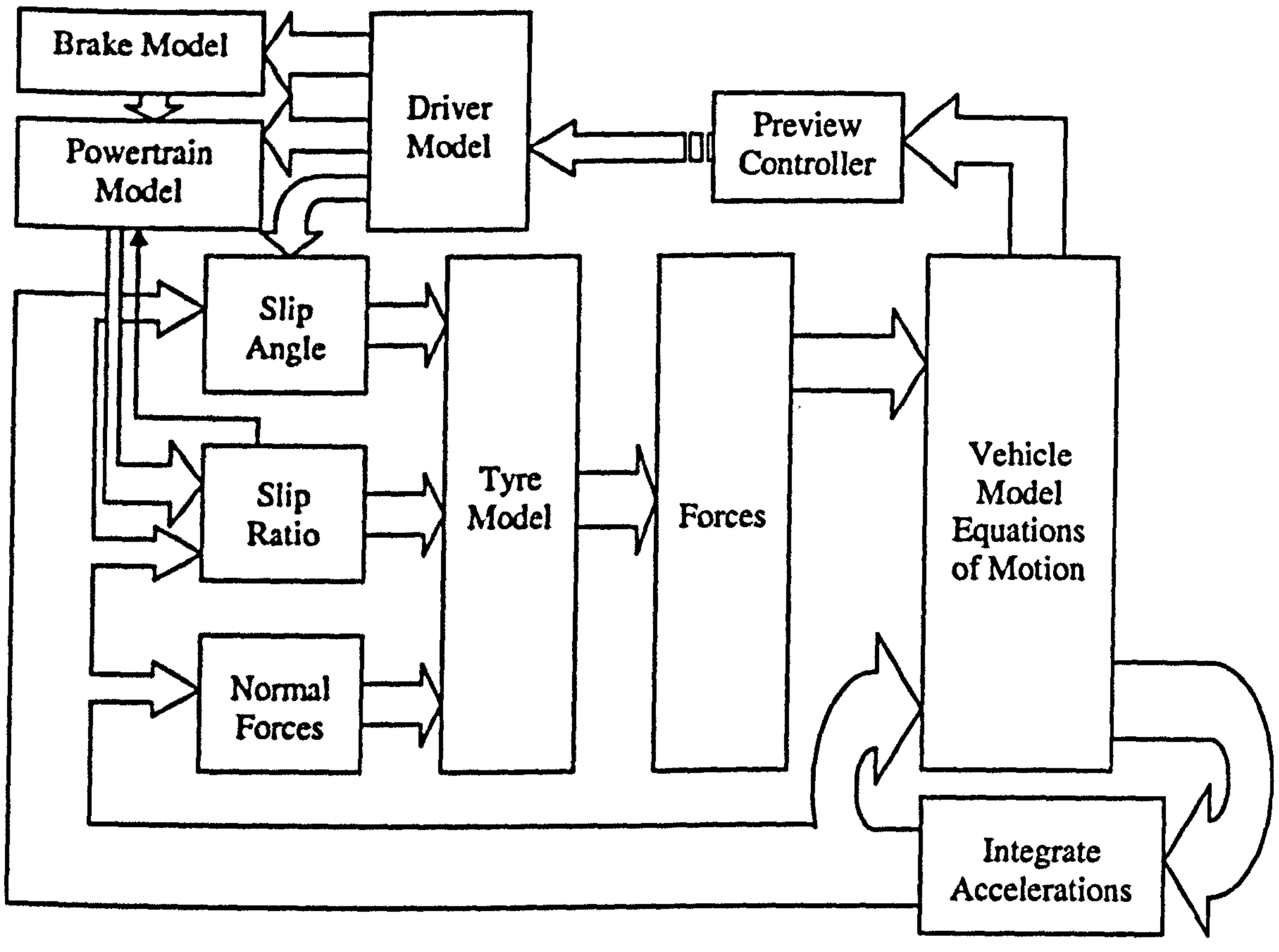


Figure 3.1 – Modular vehicle model design.

To begin a simulation, a MatLab script based program loads the vehicle parameters, a driver control time history matrix and an initial velocity. The Simulink vehicle model is then called from the MatLab workspace, the simulation run and results returned to the workspace. The MatLab script then displays the simulation results or passes them to other MatLab scripts for post processing (e.g. frequency response, path error). Examples are given of the user interface routines in Chapter Seven.

3.2 Simple model

3.2.1 Vehicle Model

From reviewing the literature, an initial minimum level of sophistication was determined, which allowed all the major factors affecting vehicle performance to be adequately modelled. This simple vehicle model consisted of seven DOF (longitudinal, lateral, yaw and four wheel spin DOF) and has all four wheels modelled. The model utilised a Pacejka combined slip tyre model and a quasi-static approximation for normal force at the contact patch. It also accounted for aerodynamic lift and drag and tyre rolling resistance.

The performance of the vehicle is modelled by assuming the mass of the vehicle is concentrated at the vehicle's centre of gravity and a suitable estimate of the vehicle's yaw inertia is used, as shown in figure 3.2. Appendix A shows the full derivation of the equations of motion of the vehicle model from first principles, using the inertial axis system in figure 3.2. The full model created in Simulink is shown in Appendix D.

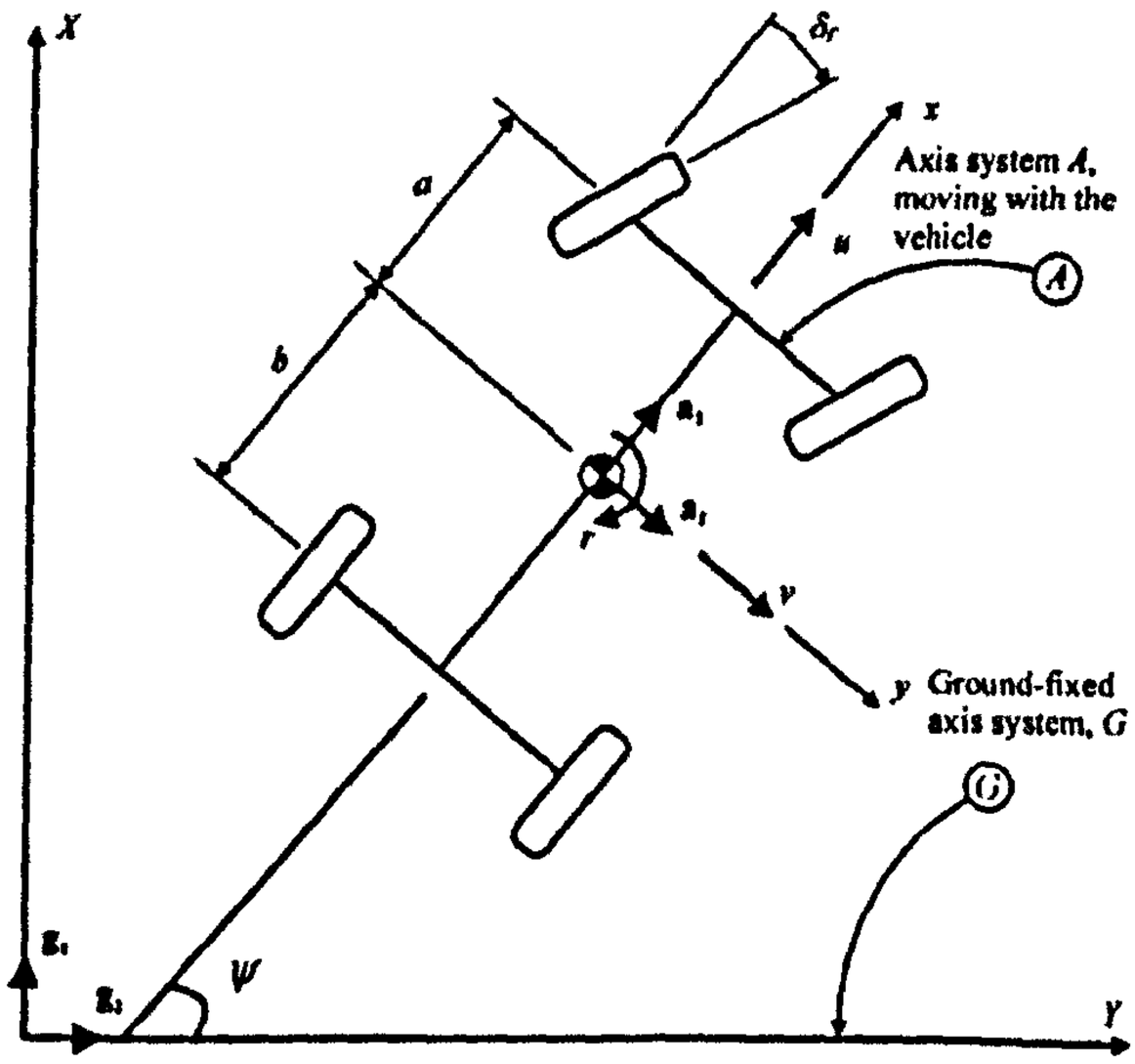


Figure 3.2 – The moving axis system, A, for a simple four wheel model.

The equations used to represent the vehicle's longitudinal, lateral and yaw response are given in equations (4), (5) and (6) respectively. These can be seen to be in the form of Newton's second law and due to the fact that the front wheels are steered the front wheel lateral force terms also appear in equation (4) along with the longitudinal tyre force terms. A similar thing happens in equation (5) (subscripts are defined in the nomenclature and shown in figure 3.2).

$$\sum F_x = \cos\delta_f F_{xfl} + \cos\delta_f F_{xfr} - \sin\delta_f F_{yfl} - \sin\delta_f F_{yfl} + F_{xrl} + F_{xfr} - 1/2\left(\rho FAC_d u^2\right) - \sum_{n=1}^{4} F_{drag} \quad (4)$$
$$= m(\dot{u} - vr)$$

$$\text{where} \qquad \sum_{n=1}^{4} F_{drag} = \mu_{rr} N_n \cos\alpha_n$$

$$\sum F_y = \cos\delta_f F_{yfl} + \cos\delta_f F_{yfr} + \sin\delta_f F_{xfr} + \sin\delta_f F_{xfr} + F_{yrl} + F_{yrr} = m(\dot{v} + ur) \quad (5)$$

$$\sum M_z = \left(F_{yfl} + F_{yfr}\right)\cos\partial_f a - \left(F_{yrl} + F_{yrr}\right)b = I_{zz}\ddot{r} \quad (6)$$

where:

a: Centre of gravity distance from front axle, m (figure 3.2)

b: Centre of gravity distance from rear axle, m (figure 3.2)

I_{zz}: Yaw inertia of whole vehicle about a3 axis, kgm^2

m: mass, kg

r: Rotational velocity about g3 axis, $rads^{-1}$ (figure 3.2)

u: Velocity of vehicle in a1 axis direction, ms^{-1} (figure 3.2)

v: Velocity of vehicle in a2 axis direction, ms^{-1} (figure 3.2)

δ_f: Steered angle of front axle, rad (figure 3.2)

μ_{rr}: Coefficient of rolling resistance for a tyre

The equation used to represent the response of each wheel system is shown in equations (7) to (10). Figure 3.3 shows the axis system used for each wheel.

$$F_{xfl} r_{wfl} = I_{wfl} \dot{\omega}_{fl} \qquad (7)$$

$$F_{xfr} r_{wfr} = I_{wfr} \dot{\omega}_{fr} \qquad (8)$$

$$F_{xrl} r_{wrl} = I_{wrl} \dot{\omega}_{rl} \qquad (9)$$

$$F_{xrr} r_{wrr} = I_{wrr} \dot{\omega}_{rr} \qquad (10)$$

where:

I_w: Spin inertia of wheel about its spin axis (axle), kgm^2

r_w: Radius of tyre, m

ω: Rotational velocity of wheel about axle, $rads^{-1}$

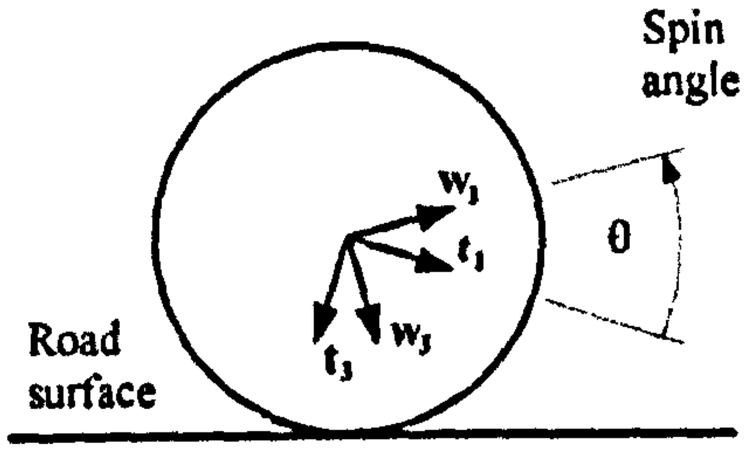


Figure 3.3 – Wheel axis system

3.2.2 Tyre Model

The slip angles at each tyre contact patch are dependent on the geometry of the vehicle and its longitudinal, lateral and yaw velocities. Using figure 3.4 the value of the slip angle at each tyre contact patch can be found and is defined as the longitudinal velocity of the tyre contact patch divided by the lateral velocity of the tyre contact patch, see equations (11) to (14).

$$\alpha_{fl} = \left(\frac{u + t_f r}{v + ar} - \delta_f \right) \quad (11)$$

$$\alpha_{fr} = \left(\frac{u - t_f r}{v + ar} - \delta_f \right) \quad (12)$$

$$\alpha_{rl} = \left(\frac{u + t_r r}{v - br} \right) \quad (13)$$

$$\alpha_{rr} = \left(\frac{u - t_r r}{v - br} \right) \quad (14)$$

where:

t: Half the value of track of axle, m (figure 3.2)

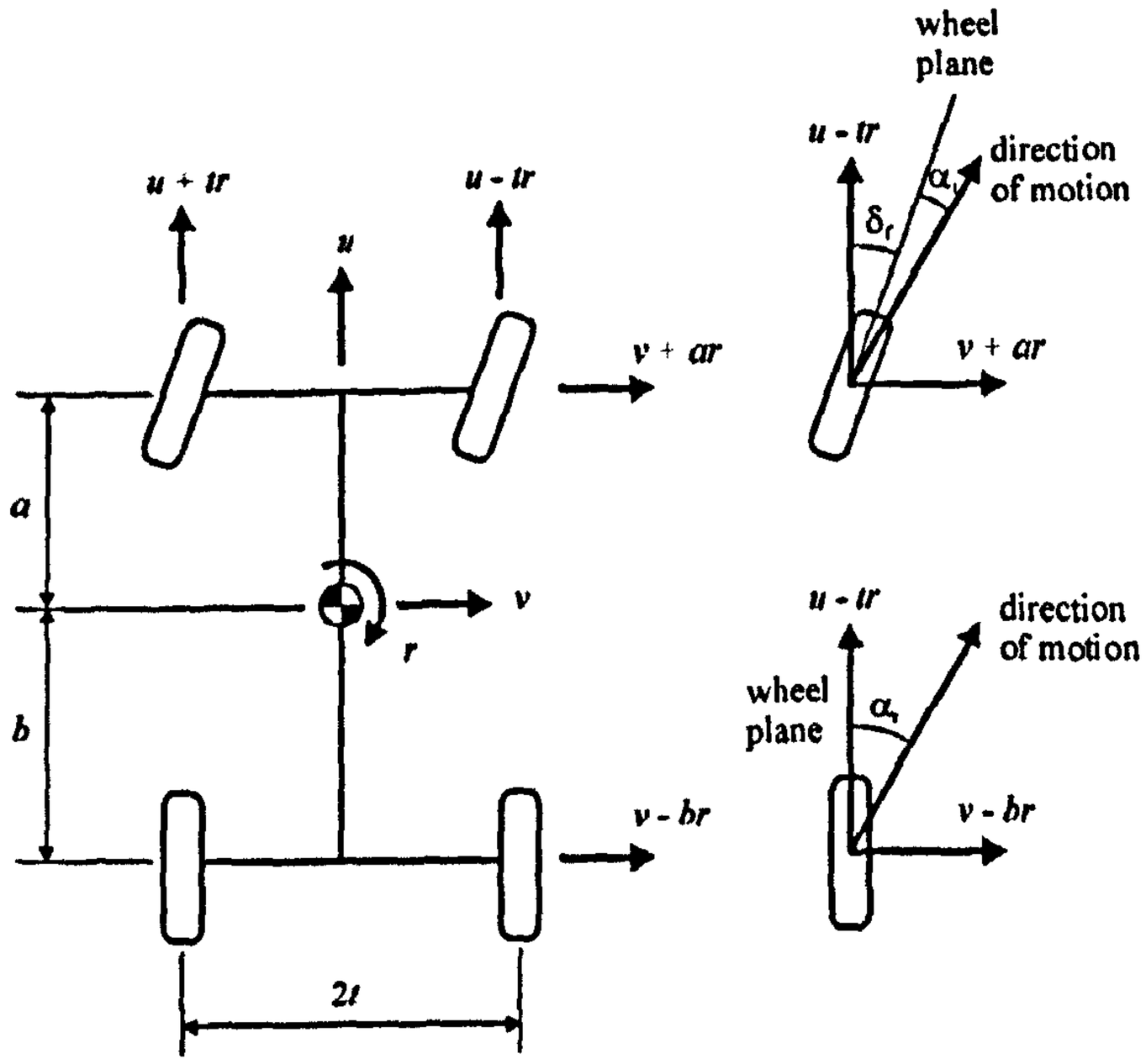


Figure 3.4 – Tyre slip angles.

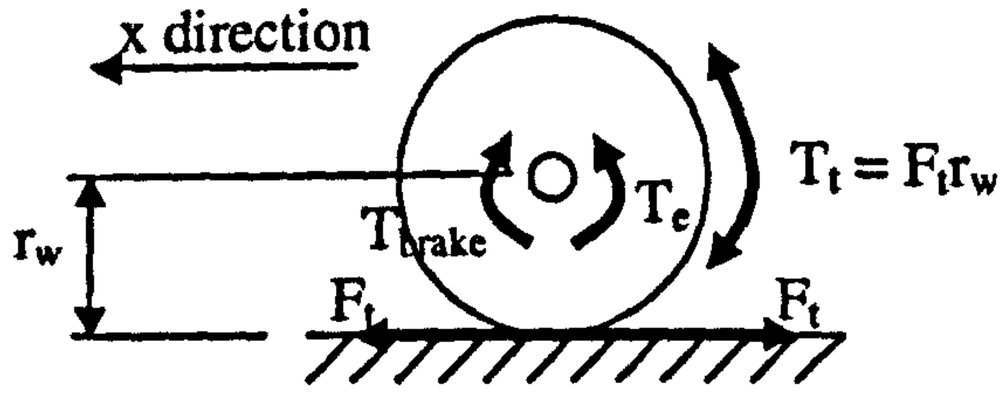


Figure 3.5 – Torques producing wheel longitudinal slip.

The longitudinal slip ratio of a tyre is defined [8] as being the ratio of the velocity of the tyre contact patch to the velocity of the vehicle at that point. The rotational velocity of the wheel is found by integration of the sum of the torque being applied to the tyre by the powertrain or braking system and the torque being produced by the tyre at that time due to the slip ratio generated, see figure 3.5. The tyre's contact patch velocity can then be divided by the ground velocity at that point to give the slip ratio, see equations (15) to (18).

$$\kappa_{fl} = \left(r_w \int \frac{-T_{brakef} - T_{tfl}}{I_w} .dt \right) \left(\frac{1}{u + t_f r} \right) \tag{15}$$

$$\kappa_{fr} = \left(r_w \int \frac{-T_{brakef} - T_{tfr}}{I_w} .dt \right) \left(\frac{1}{u - t_f r} \right) \tag{16}$$

$$\kappa_{rl} = \left(r_w \int \frac{T_{erl} - T_{braker} - T_{trl}}{I_w} .dt \right) \left(\frac{1}{u + t_r r} \right) \tag{17}$$

$$\kappa_{rr} = \left(r_w \int \frac{T_{err} - T_{braker} - T_{trr}}{I_w} .dt \right) \left(\frac{1}{u - t_r r} \right) \tag{18}$$

where:

T_{brake}: Torque produced by the brake at that wheel, Nm

T_e: Torque produced by the engine at that wheel, Nm

T_t: Torque produced by the tyre itself at that wheel, Nm

The tyre is modelled using the Pacejka Magic Formula [8], which is based on an empirical approach, described in the previous chapter. The resultant forces created by the tyre are derived from a set of coefficients which are obtained by direct measurement of the tyre. These coefficients (see Appendix C) define the tyres response to external factors which generate forces in the tyre (i.e. α, κ and N, in equation (1)). Pacejka [8] has shown that the response of the tyre is given by equations (19) and (20). The coefficients used in these equations are found from the measured tyre coefficients and the applied external inputs.

$$F = D\sin[C\arctan\{Bx - E(Bx - \arctan Bx)\}] - S_v \qquad (19)$$

where for lateral force $x = \alpha + S_H$ and for longitudinal force $x = \kappa + S_H$ (20)

B: Stiffness factor for tyre magic formula equation

C: Shape factor for tyre magic formula equation

D: Peak value for tyre magic formula equation

E: Curvature factor for tyre magic formula equation

S_H: Horizontal shift for tyre magic formula equation

S_v: Vertical shift for tyre magic formula equation

x: Variable

3.2.3 Load Transfer Model

A quasi-static approximation of lateral load transfer [3] has been used to calculate the normal force at the tyre contact patch and is dependent on static weight distribution, suspension roll stiffness distribution and aerodynamic downforce, see equations (21) to (26).

$$N_{fouter} = \left(\frac{mgb}{2l}\right) + LLT_f + \frac{1}{4}\rho.FA.C_{lf}.u^2 \qquad (21)$$

$$N_{finner} = \left(\frac{mgb}{2l}\right) - LLT_f + \frac{1}{4}\rho.FA.C_{lf}.u^2 \qquad (22)$$

$$N_{router} = \left(\frac{mga}{2l}\right) + LLT_r + \frac{1}{4}\rho.FA.C_{lr}.u^2 \qquad (23)$$

$$N_{rinner} = \left(\frac{mga}{2l}\right) - LLT_r + \frac{1}{4}\rho.FA.C_{lr}.u^2 \qquad (24)$$

where:

$$LLT_f = \frac{(\dot{v}+ur)m}{2t_f}\left(\frac{hk_{\phi f}}{k_{\phi f}+k_{\phi r}} + \frac{b}{l}z_{rf}\right) \qquad (25)$$

and $$LLT_r = \frac{(\dot{v}+ur)m}{2t_r}\left(\frac{hk_{\phi r}}{k_{\phi f}+k_{\phi r}} + \frac{a}{l}z_{rr}\right) \qquad (26)$$

and also:

g: Gravitational constant, ms^{-2}

h: Sprung mass centre of gravity distance from roll axis, m

l: Wheelbase of car, m (figure 3.2)

LLT: Lateral load transfer, N

k_ϕ: Sprung mass roll stiffness, $Nmrad^{-1}$

z_r: Roll centre height, m

3.2.4 Powertrain Model

The engine has been modelled as a parameter map of engine speed against torque produced (this is found from experimental data and is at 100% throttle), see figure 3.6. As the clutch is not modelled and the engine is modelled as linked directly to the road wheels (through the drivetrain), the parameter map is adjusted to give a

reasonable level of torque at low engine speeds. This accounts for the driver slipping the clutch from a stationary start to allow the engine speed to be high and give a reasonable level of torque. The simulation controls the level of torque applied using a throttle value that limits the engine torque value taken from the parameter map.

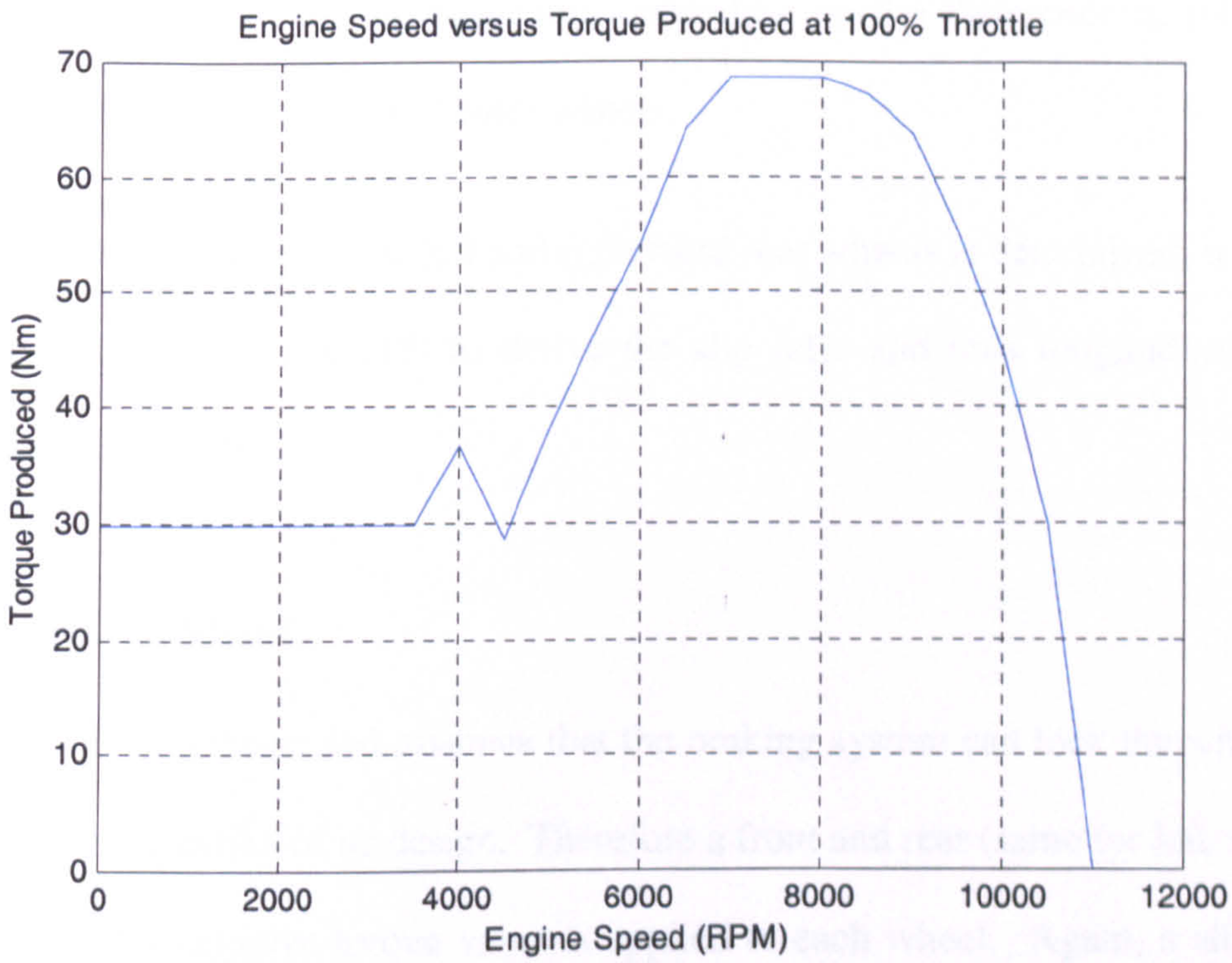


Figure 3.6 – Engine parameter map.

The drivetrain is modelled as a set of six different ratios which increase the torque (and decrease the rotational velocity) from the engine. These ratios take into account the internal engine gear reductions and vehicle driveline ratios and a control function automatically changes up gear at 10,000 RPM or down gear at 5,000 RPM (stopping at 1^{st} or 6^{th} gear). These values are definable by the user and keeps the model in the optimum gear that will maximise the torque produced by the engine, any tighter a 'rev range' was found to be unrealistic compared to the actual driver rev range utilised.

The torque is applied to the rear axle through a torque biasing (Torsen type) limited slip differential, which splits the torque between the rear wheels (biases) depending on the difference in rear wheel rotational velocities [56]. This initial simple model is quite basic and linearly takes torque from the inner wheel and applies it to the outer wheel up to the value of the bias ratio, depending on the difference in rotational velocities between the inner and outer wheels.

Once the torque applied to the left and right hand rear wheels is determined, it can be used in equations (15) to (18) to derive the slip ratio and thus longitudinal force produced by the tyre.

3.2.5 Braking Model

The model is very basic and assumes that the braking system can lock the wheels at any speed, irrespective of its design. Therefore a front and rear (same for left or right side of vehicle) negative torque value is applied to each wheel. Again, a slip ratio can then be derived from equations (15) to (18) and thus longitudinal force produced by the tyre is found.

3.3 Sophisticated model

3.3.1 Vehicle Model

The vehicle is assumed to consist of three parts; a sprung mass body, a front unsprung mass body and a rear unsprung mass body. A suitable value for the second moment of area is assigned for the vehicle's yaw inertia and the sprung mass roll and pitch inertia values (the products of inertia are assumed small and negligible). Crolla [16] has shown that the equations of motion of the vehicle can be derived from first principles using a Lagrangian approach. This involves using a set of partial derivatives (Lagrange's equations) which contain the total kinetic, potential and dissipative energies of each of the components of the system and can be evaluated to give the equations of motion.

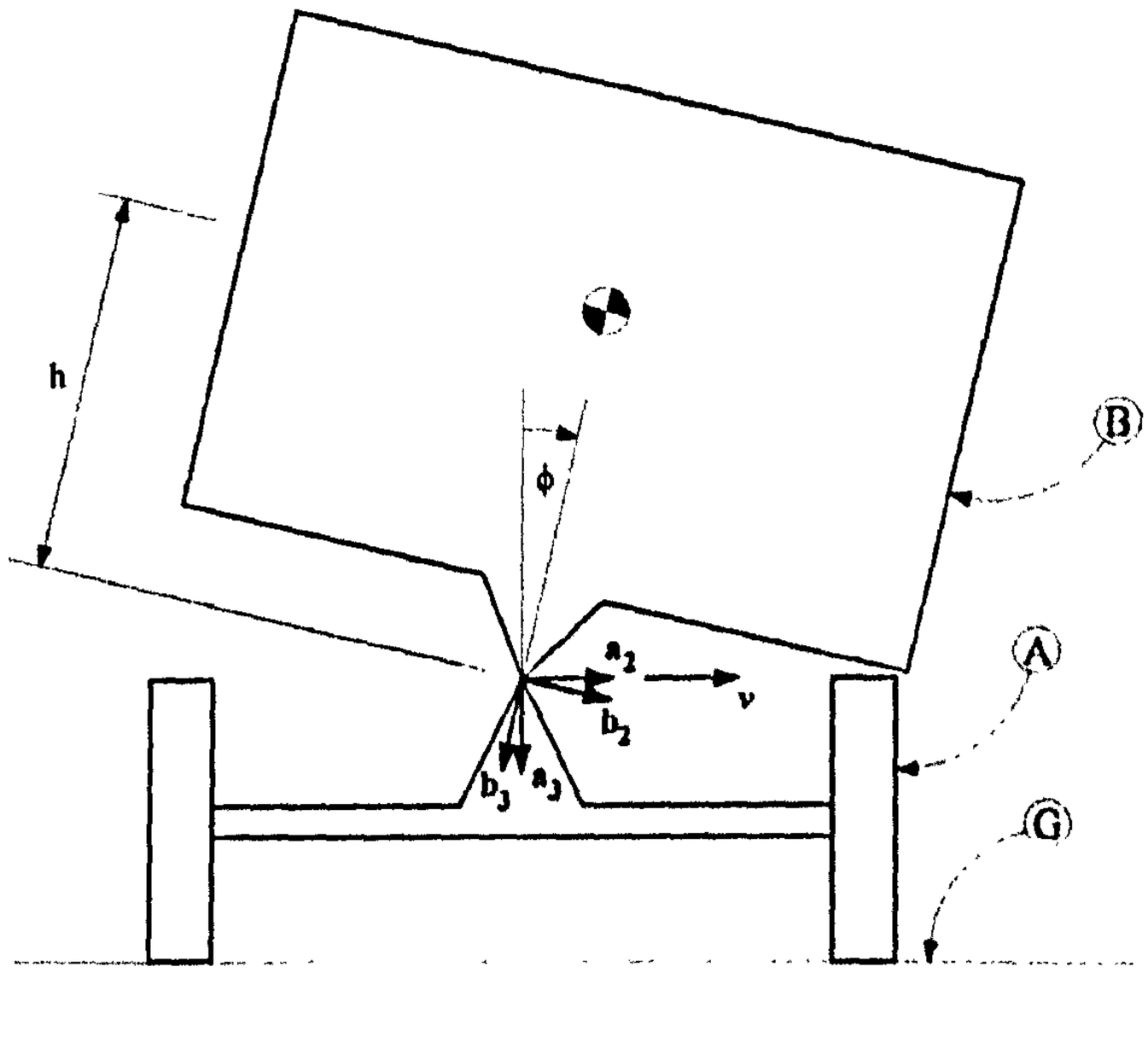


Figure 3.7 – The moving axis system, B, for roll DOF.

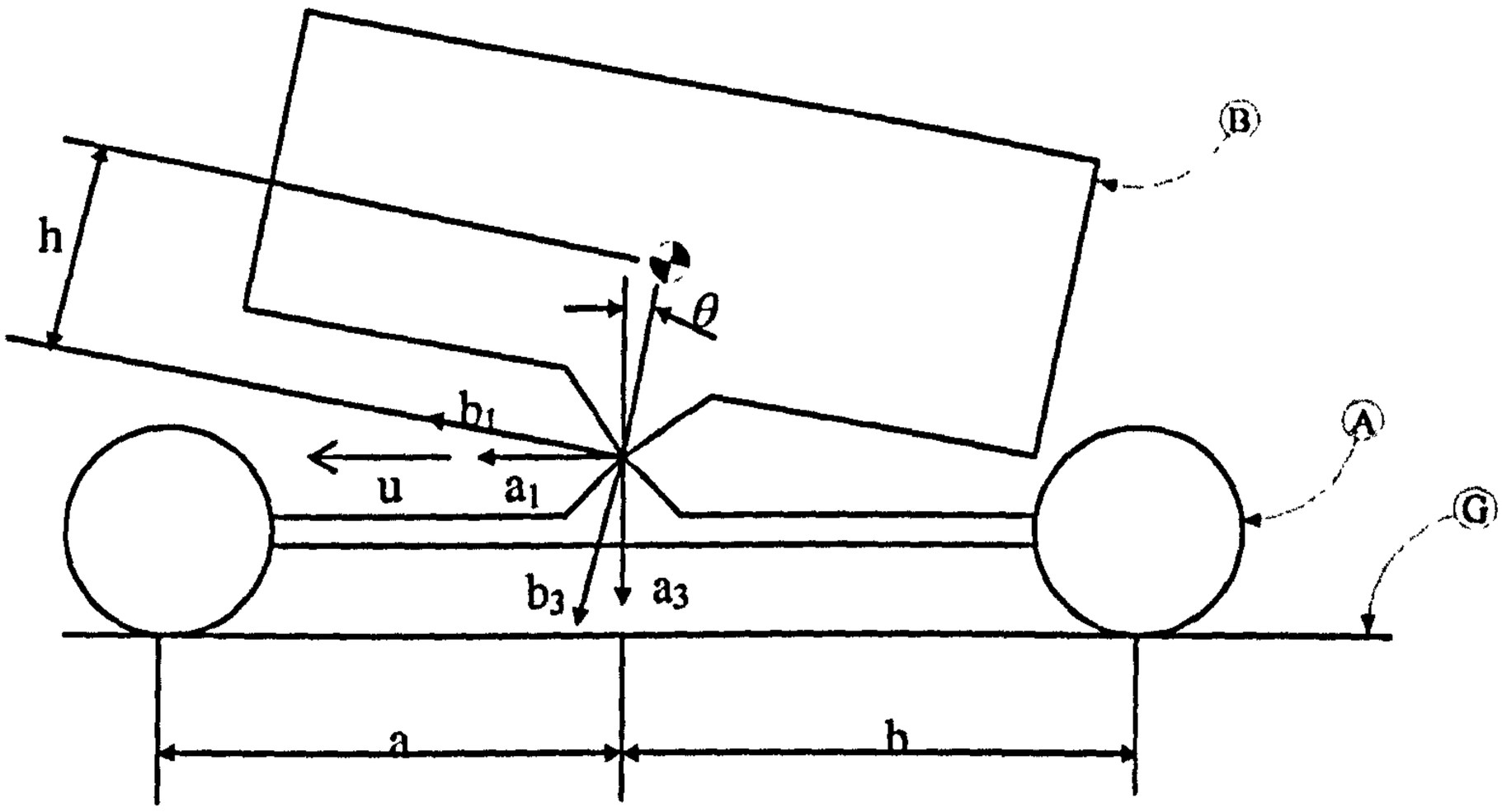


Figure 3.8 – The moving axis system, B, for pitch DOF.

Figures 3.2, 3.7 and 3.8 shows the axis systems used to describe the vehicle. Two inertial frames of reference (axis system A and B) have been set up in a non inertial axis system G. The vehicle fixed reference system A moves in the ground fixed reference system G, whilst the unsprung mass fixed reference system B is free to rotate around a_1 or a_2 (i.e. roll about the vehicle fixed axis system a_1 axis or pitch around the vehicle fixed axis system a_2 axis respectively). The equations that are used to represent the vehicle's performance are given below in equations (27) to (35) (longitudinal, lateral, yaw, roll, pitch and four wheel spin DOF) and are in a similar form to those for the simple model. The full derivation of these equations can be seen in Appendix B, with Appendix E displaying the Simulink model.

$$\sum F_x = \cos\delta_f F_{xfl} + \cos\delta_f F_{xfr} - \sin\delta_f F_{xfl} - \sin\delta_f F_{yfl} + F_{xrl} + F_{xrr} - 1/2\left(\rho FAC_d u^2\right) - \sum_{n=1}^{4} F_{drag} = \quad (27)$$
$$m_f\left(\dot{u} - vr - ar^2\right) + m_r\left(\dot{u} - vr + br^2\right) + m_b\left(\dot{u} - vr - \ddot{\theta}h - \dot{\phi}hr\right)$$

where $\sum_{n=1}^{4} F_{drag} = \mu_{rr} N_n \cos\alpha_n$

$$\sum F_y = \cos\delta_f F_{yfl} + \cos\delta_f F_{yfr} + \sin\delta_f F_{xfr} + \sin\delta_f F_{xfr} + F_{yrl} + F_{yrr} = \quad (28)$$
$$m_f\left(\dot{v} + ur - a\dot{r}\right) + m_r\left(\dot{v} + ur - b\dot{r}\right) + m_b\left(\dot{v} + ur - \dot{\theta}hr + \ddot{\phi}h\right)$$

$$\sum M_z = \left(F_{yfl} + F_{yfr}\right)\cos\partial_f a - \left(F_{yrl} + F_{yrr}\right)b = m_f\left(uar + \dot{v}a - a^2\dot{r}\right) \quad (29)$$
$$+ m_r\left(b^2\dot{r} - ubr - \dot{v}b\right) + m_b\left(u\dot{\phi}h - v\dot{\theta}h\right) + I_{zz}\dot{r}$$

$$\sum M_x = 0 = m_b\left(urh - \dot{v}h - \dot{\theta}h^2 r - \phi gh + \ddot{\phi}h^2\right) \quad (30)$$
$$+ I_{yyb}\dot{\theta}r + I_{xxb}\ddot{\phi} + k_\phi\phi + C_\phi\phi$$

$$\sum M_y = 0 = m_b\left(vrh - \dot{u}h + \dot{\phi}h^2 r - \theta gh + \ddot{\theta}h^2\right) \quad (31)$$
$$- I_{xxb}\dot{\phi}r + I_{yyb}\ddot{\theta} + k_\theta\theta + C_\theta\theta$$

$$F_{xfl} r_{wfl} = I_{wfl}\dot{\omega}_{fl} \quad (32)$$

$$F_{xfr} r_{wfr} = I_{wfr}\dot{\omega}_{fr} \quad (33)$$

$$F_{xrl} r_{wrl} = I_{wrl}\dot{\omega}_{rl} \quad (34)$$

$$F_{xrr} r_{wrr} = I_{wrr}\dot{\omega}_{rr} \quad (35)$$

where:

I_{yyb}: Pitch inertia of sprung mass about a2 axis, kgm^2

I_{xxb}: Roll inertia of sprung mass about a1 axis, kgm^2

M: Moment, Nm

θ: Pitch angle of sprung mass, rad (figure 3.8)

ϕ: Roll angle of sprung mass, rad (figure 3.7)

3.3.2 Tyre Model

The tyre model used is the same as in the simple model, except that the lag in lateral force produced by the tyre has been included in the model. This was introduced using a first order system [8] to lag the slip angle being fed into the tyre model and includes the result of normal force and longitudinal speed on the lagged response of the tyre (see equations (37) to (40)).

$$\alpha_{fl} = \left(\frac{u + t_f r}{v + ar} - \delta_f\right) - \left(\frac{\sigma_f N_{fl}}{N_0 (u + t_f r)}\right)\dot{\alpha}_{fl} \qquad (37)$$

$$\alpha_{fr} = \left(\frac{u - t_f r}{v + ar} - \delta_f\right) - \left(\frac{\sigma_f N_{fr}}{N_0 (u - t_f r)}\right)\dot{\alpha}_{fr} \qquad (38)$$

$$\alpha_{rl} = \left(\frac{u + t_r r}{v - br}\right) - \left(\frac{\sigma_r N_{rl}}{N_0 (u + t_r r)}\right)\dot{\alpha}_{rl} \qquad (39)$$

$$\alpha_{rr} = \left(\frac{u - t_r r}{v - br}\right) - \left(\frac{\sigma_r N_{rr}}{N_0 (u - t_r r)}\right)\dot{\alpha}_{rr} \qquad (40)$$

where:

σ: Tyre lateral slip angle lag coefficient

3.3.3 Load Transfer Model

The normal forces at the tyre contact patch are proportional to the static weight distribution, the aerodynamic downforce on the axle and the lateral and longitudinal load transfer due to the sprung mass rolling and/or pitching (see equations (41) to (44)).

$$N_{fl} = \left(\frac{mgb}{2l}\right) + \frac{1}{4}\rho.FA.C_{lf}.u^2 - \theta\frac{k_{\theta f}a}{2} + \phi k_{\phi f}t_f \qquad (41)$$

$$N_{fr} = \left(\frac{mgb}{2l}\right) + \frac{1}{4}\rho.FA.C_{lf}.u^2 - \theta\frac{k_{\theta f}a}{2} - \phi k_{\phi f}t_f \qquad (42)$$

$$N_{rl} = \left(\frac{mga}{2l}\right) + \frac{1}{4}\rho.FA.C_{lr}.u^2 + \theta\frac{k_{\theta r}b}{2} + \phi k_{\phi r}t_r \qquad (43)$$

$$N_{rr} = \left(\frac{mga}{2l}\right) + \frac{1}{4}\rho.FA.C_{lr}.u^2 + \theta\frac{k_{\theta}b}{2} - \phi k_{\phi}t_r \qquad (44)$$

where:

k_ϕ: Sprung mass roll stiffness, $Nmrad^{-1}$

k_θ: Sprung mass pitch stiffness, $Nmrad^{-1}$

3.3.4 Powertrain Model

The main difference between the simple and sophisticated models is the differential model used. The differential model takes the form of an empirical model which uses the same equation form as the Pacejka Tyre Formula (equation (15)) and varies the torque biasing value, up to a saturation value, given the difference in left and right wheel rotational velocities. The relationship is shown in figure 3.9 and the initial slope and final saturation value is controlled by a torque sensitivity and bias ratio value respectively. Not only is this model used for driving, but also braking, as the vehicle is modelled as having only one rear brake disc mounted on the differential housing (this design is peculiar to the Leeds Formula SAE car). The only other difference between the simple and sophisticated vehicle models is a gear change time, which is a user specified delay in applying engine torque after a gear change.

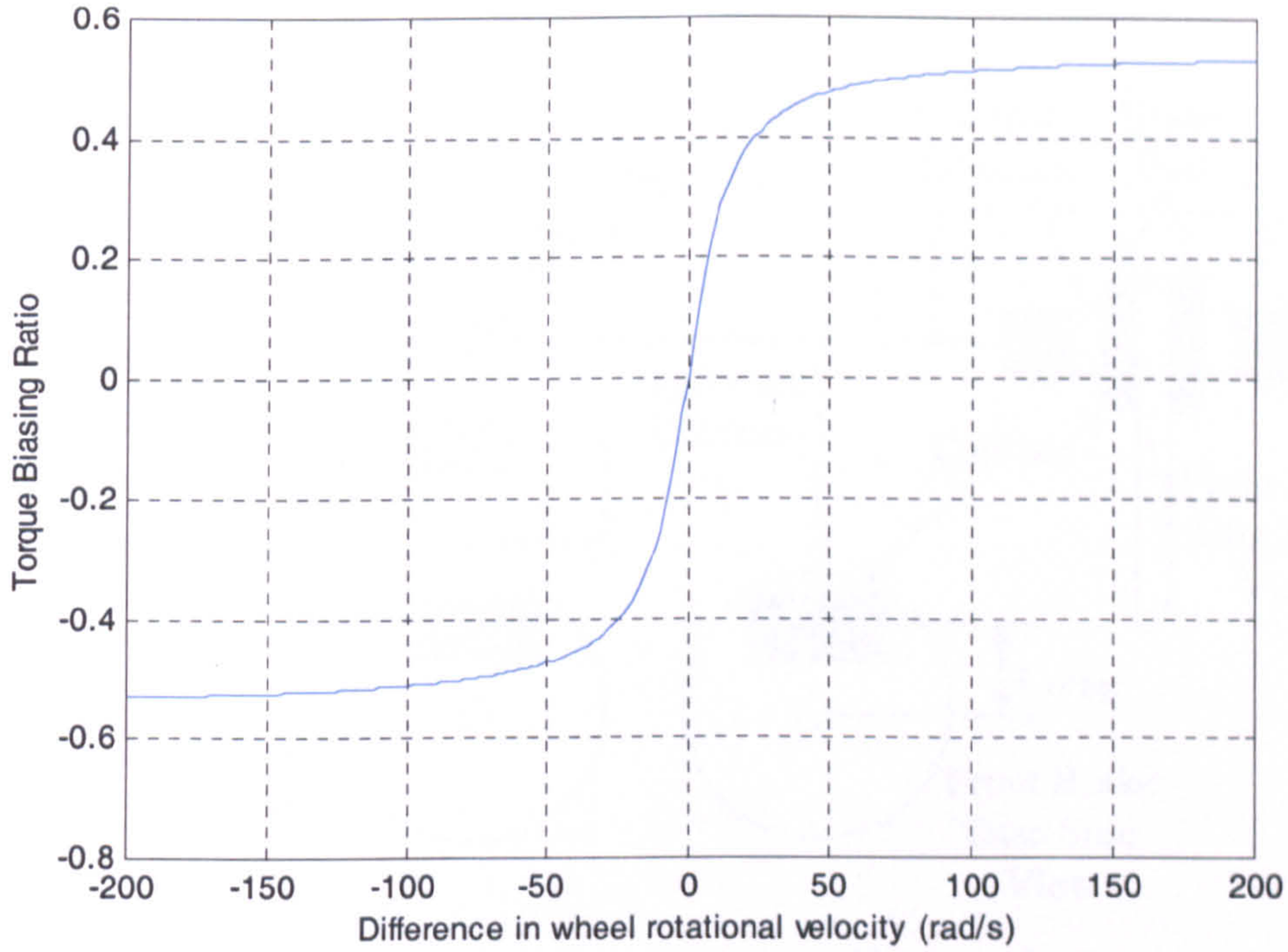


Figure 3.9 – Response of empirical torque biasing differential model.

3.3.5 Braking Model

The braking system schematic shown in figure 3.10 shows that the braking torques, T_{brake}, are generated using a driver control input of pedal force. This force is distributed between the two master cylinders by the balance bar and each generates a pressure in the front and rear hydraulic circuits. The brake discs and callipers on the vehicle are modelled as friction devices which generate a torque by applying a force resisting wheel rotation at the calliper radius, see equation (45). Once the torque is found a slip ratio can then be derived using equations (15) to (18) and thus longitudinal force produced by the tyre.

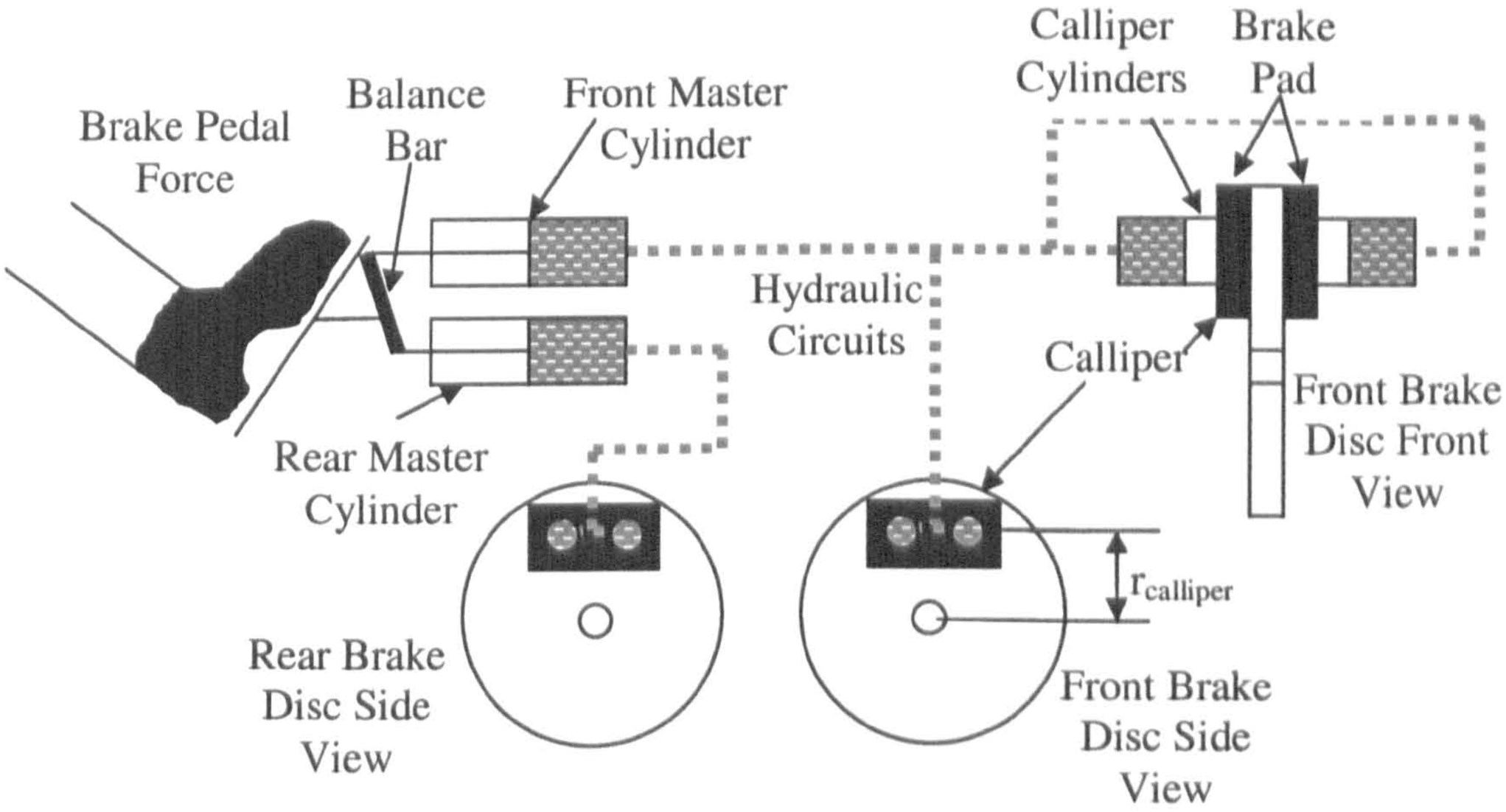


Figure 3.10 – Schematic of a braking system using three brake discs, each with a four-pot calliper.

$$T_{brake} = \left(\frac{Pedal(\mu_{pad}) Balbar(A_{calliper}) r_{calliper}}{A_{master}} \right) F_{foot} \qquad (45)$$

where:

$A_{calliper}$: Total area of pistons in a calliper, m^2

A_{master}: Area of master cylinder, m^2

Balbar: Brake system balance bar ratio

Pedal: Brake pedal force ratio

$r_{calliper}$: Radius of calliper, m

μ_{pad}: Coefficient of friction between brake pad and disc

As mentioned above, this particular model involves only one brake disc on the rear of the vehicle mounted to the differential casing. The negative bias ratio of the differential, therefore, can play an important roll in combined cornering and braking manoeuvres. The model is easily expandable to a more conventional 4 brake system.

3.4 Non-linear Path Following Preview Controller

A non-linear path following preview controller has been created to enable closed loop steer angle control input estimations for following a given path, see figure 3.1. These estimations have then been employed in the transient approach for LTS as an initial guess for the vehicle control inputs used to initialise the optimisation routine.

The scheme employed has been detailed by Casanova et al [9] and involves using a path previewing optical lever to find the correct steer angle needed to follow a given path. The optical lever can be seen in figure 3.11 and involves estimating the path error between the ideal path and preview points on a line drawn out, forward from the vehicle and along its longitudinal axis (i.e. an optical lever). These errors are then passed through a series of control gains and saturation functions (see figure 3.12) to estimate the driver's steer angle input given by the current deviation from the ideal path and the approaching ideal path the vehicle is being required to follow.

A more complete description of this method is given below.

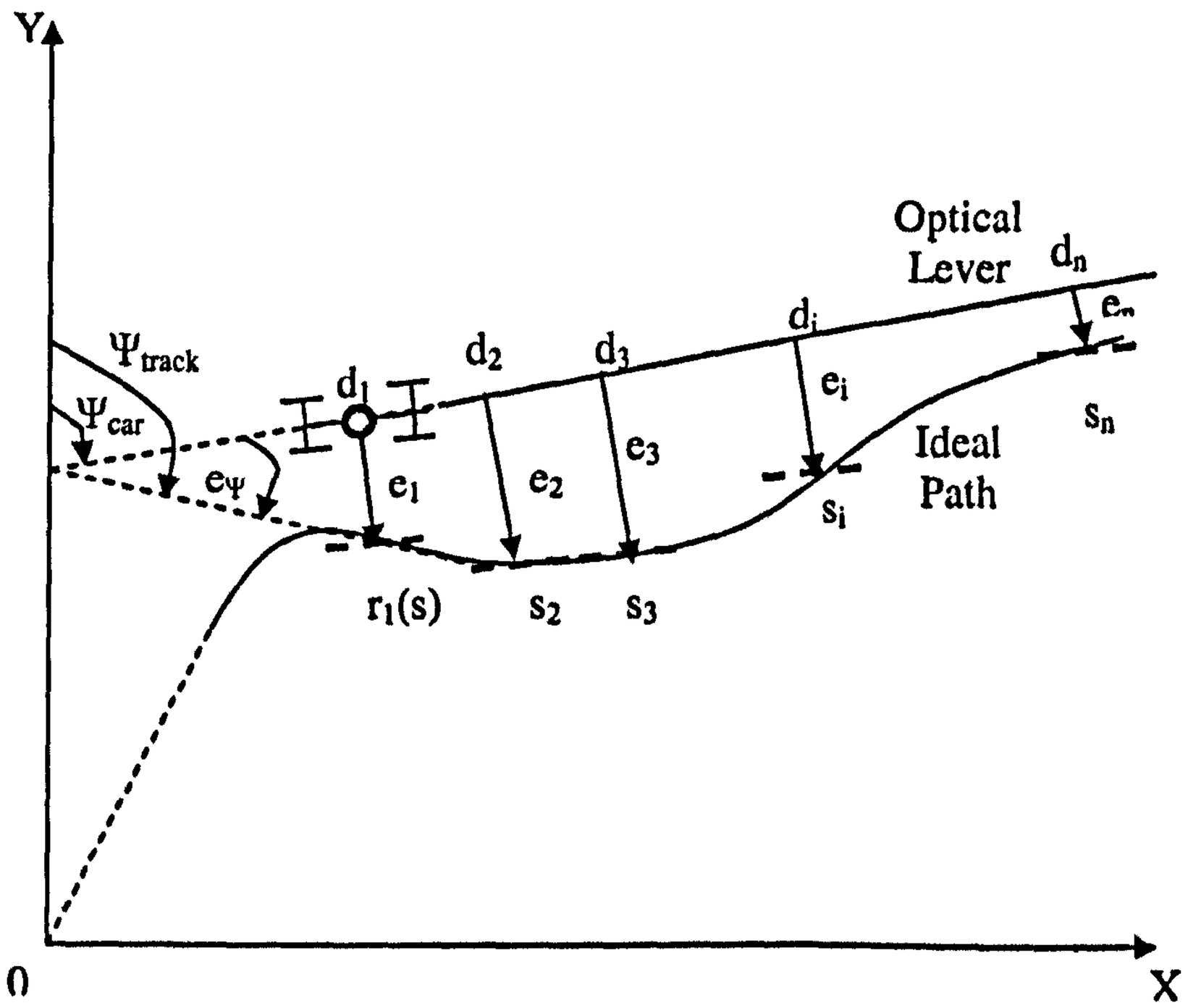


Figure 3.11 – Optical lever and path error techniques.

The simulation is run with the vehicle undergoing constant forward velocity and an outline of the controller's method of operation is given below:

1. Path description defined by user (or found from data logging measured data) and consists of matrix of path curvature, tangent angle to path, x position and y position (measured from a fixed axis) at fixed distance steps along the track's path.
2. An optical lever is extended forward from the vehicle parallel to the vehicle's longitudinal axis. Preview points are placed at fixed distances, d, forward from the vehicle's centre of gravity.
3. At any given point in the simulation, the vehicle's position and yaw angle is used to estimate the position of each preview point relative to the track. Using linear interpolation between the discrete points in the path description matrix, the path error, e_n and e_ψ (perpendicular distance between preview point and path and error angle), is found for each preview point on the optical lever, see figure 3.11.
4. A non-linear control scheme with saturation functions then uses these path errors to estimate the required steer angle. It does this by passing the errors through a gain and saturation matrix, see figure 3.12. The saturation matrix ensures the steer response for each path error value keeps within certain limits and prevents the vehicle oscillating about the correct path.

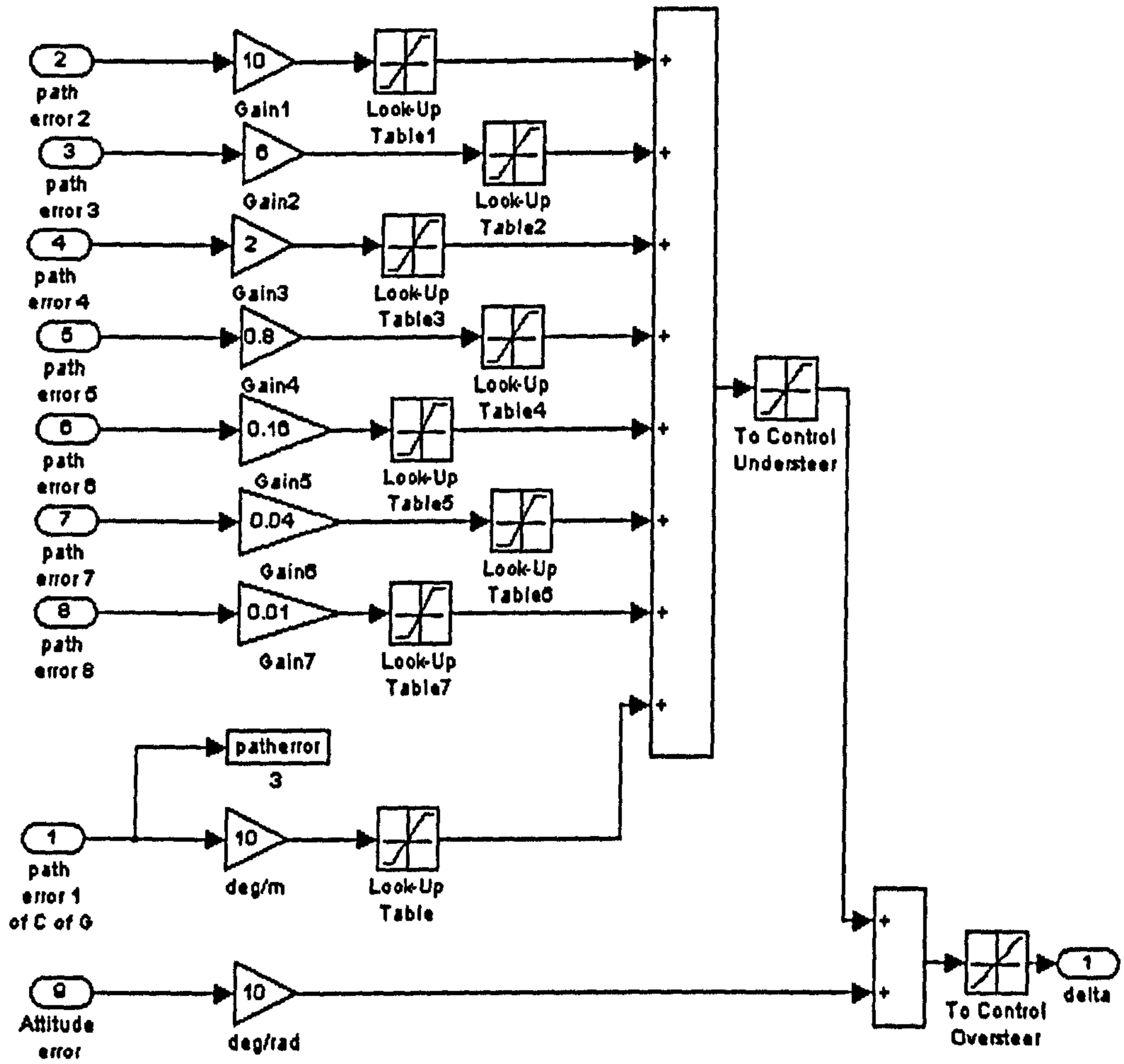


Figure 3.12 – Non-linear control scheme for path following.

Empirical tuning of the gains and saturation limits was carried out. The preview controller was found to follow a given path with only a small error (below 0.5m lateral offset), even at high levels of lateral acceleration where the tyre response is highly non-linear. It has been used in the Manoeuvre Time Minimisation package to create an initial guess for the driver control matrix.

3.5 Conclusions

The chapter details the development of two vehicle models, a simple seven DOF and a more sophisticated thirteen DOF model. The simple model contains a four wheel, single body system with lateral, longitudinal and yaw DOF. It uses a Pacejka Magic tyre formula combined slip model, with each wheel having a spin DOF and a quasi-static approximation of weight transfer. The effects of aerodynamic and tyre rolling resistance loads are also accounted for. Powertrain and basic braking and differential models have been included.

The sophisticated vehicle model is a development of the simple model and includes a three body (front and rear unsprung masses and vehicle body sprung mass) system, which adds a roll and pitch DOF to the model. A tyre lag model, giving each tyre an extra degree of freedom, has been included with an improved differential and brake system models.

Both models have been created in MatLab and Simulink and full derivations and model details are displayed in the Appendices. In addition to the vehicle models, a path following non-linear preview controller has been produced to enable an initial guess to be produced of vehicle control inputs.

4 Experimental Results

4.1 Introduction

Actual racing car handling performance data is essential in the assessment of the accuracy of the vehicle models. As an in-depth study of racing car handling has not previously been published, it was decided by the author to carry out this task as part of the research undertaken. This chapter details the collection of this racing car handling performance data, which was gathered in two separate phases. In each phase a different vehicle was used and care was taken to ensure the data collected was accurate by applying well-established vehicle testing procedures [3].

Phase One involved the collection of data detailing lateral dynamic handling behaviour [55, 57]. This was used to validate the lateral, roll and yaw DOF, as well as lateral tyre, load transfer and aerodynamic areas of the vehicle models. The data logging equipment was developed in co-operation with Delft University [58] and the testing conducted in September 2000 using the University of Leeds F4 racing car.

Phase Two involved the collection of longitudinal and combined lateral and longitudinal dynamic handling behaviour. This was used to not only validate the longitudinal and pitch DOF of the vehicle models, but also the complete model including powertrain, braking, tyre, load transfer and aerodynamic models. The data logging equipment was developed by the author and the testing conducted in July 2001 using the University of Leeds F5 racing car.

The chapter also details the methods used for obtaining the parameter set for each vehicle. The measured results are summarised and discussed and details are given of any post-processing of data that occurred.

4.2 Data Logging System

The University of Leeds F4 (see Figure 4.1) and F5 racing cars were built for the Formula SAE/Student competitions. Each vehicle has a similar design and both are rear wheel drive and raced on tight, twisty circuits involving moderately low speeds (below 40 ms^{-1}). Both use a restricted Honda CBR 600cc motorcycle engine, which drives a limited slip differential mounted on the rear axle.

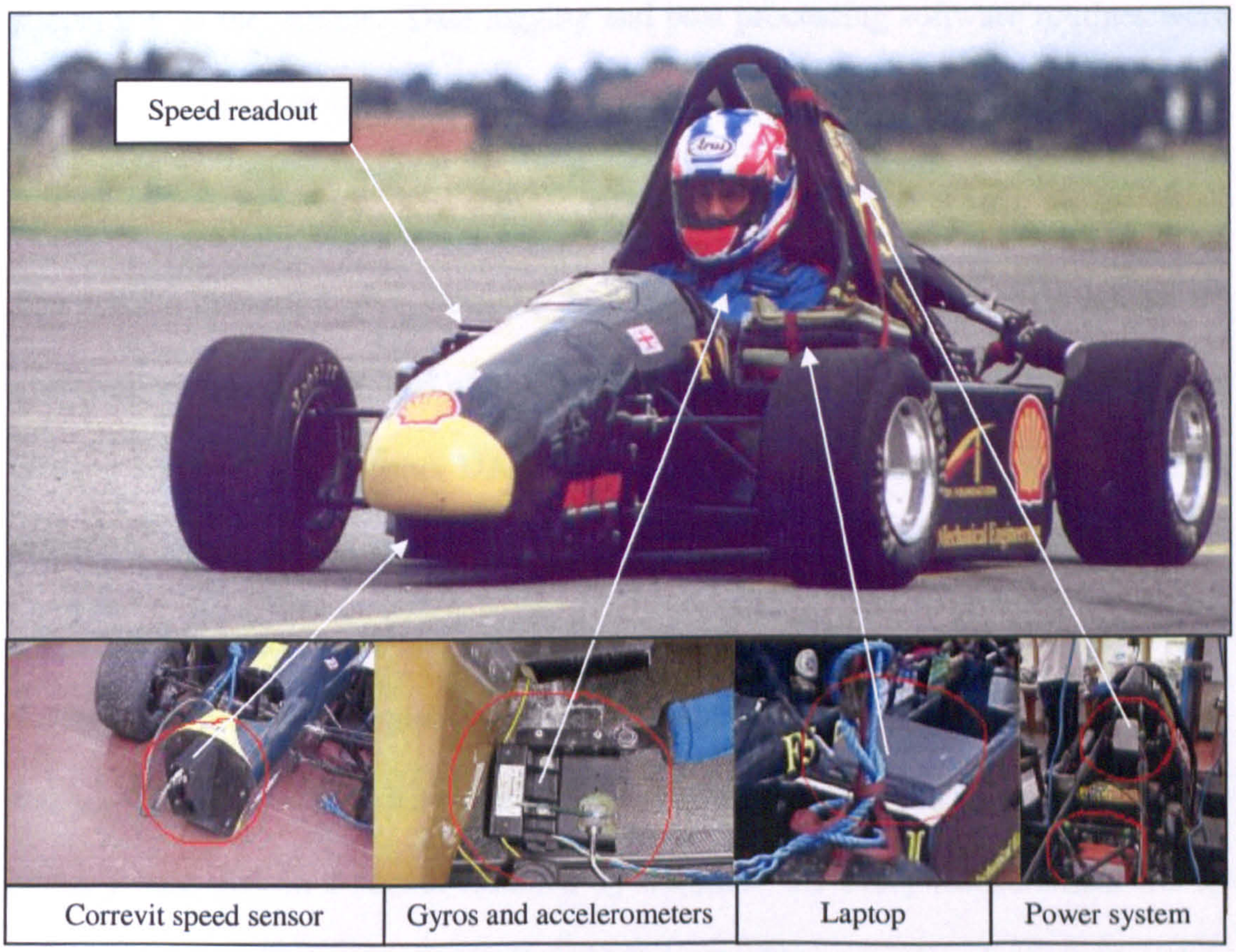


Figure 4.1 – Leeds University F4 racing car and data logging equipment locations.

In Phase One, the data logging system was constructed in cooperation with Koert De Kok [58] from Delft University and all of the equipment used was loaned by Delft University. Because of this, a time span of only a week was available to fit the system to the vehicle and conduct the tests. This fact and the reliance on the weather meant that there was no time for test repeats.

Due to the short testing time frame, considerable effort went into the systems construction, observations of the vehicle model completing planned test manoeuvres were used to specify the parameters that were to be measured and the sensor ranges required. Other considerations made in regard to the sensor specifications and the data acquisition system design itself were cost, durability, weight, accuracy and attachment to the vehicle. Data logging and post processing software routines were developed in advance for rapid evaluation of the data, once measured, and the system took two days of the available week to fit to the vehicle.

The system consisted of a Compaq Armada 1700 laptop running a Windows 95 operating system with 96 MB of RAM. Eight analogue data channels were digitised using a National Instruments DAQCard-AI-16XE-50 data acquisition card. This PCMCIA card digitised the eight analogue channels using a 16 bit A/D converter. The sampling rate was set at 200 Hz (see below) and the channels were measured differentially (i.e. each signal was measured with respect to its measured reference signal). The digital data was recorded using a program written in Labview on the laptop's hard disk in a format readable by MatLab. To every measurement file an increment number was automatically added to the filename to make sure that data was not overwritten. The entire system weighed approximately 30 kg, which was acceptable in relation to the overall weight of the vehicle (300 kg). Finally, a readout

displaying the vehicle's forward velocity was used to allow the driver to try to maintain fixed forward velocities. The vehicle performance parameters measured and the types of sensor used are shown in table 4.1.

Channel	Parameter Measured	Sensor and Range	Make and Model
1	Lateral acceleration	Piezoelectric accelerometer -5g to +5g	Druck 3140
2	Rack displacement	Linear potentiometer 0 to 150mm	Penny and Giles 129-56A
3	Yaw rate	Piezoelectric gyro -64 to +64 $degs^{-1}$	Systron Gyrochip AQRS
4	Roll rate	Piezoelectric gyro -64 to +64 $degs^{-1}$	Systron Gyrochip AQRS
5	Longitudinal speed	Correvit speed sensor 150 ms^{-1}	Datron Correvit S-CE
6	Lateral speed	Correvit speed sensor 20 ms^{-1}	Datron Correvit S-CE
7	Steering wheel angle	Absolute encoder	MIRA
8	Steering wheel torque	Strain Gauges	MIRA

Table 4.1 – Phase One sensors.

In Phase Two, the data logging system was constructed with no outside assistance, using the experience gained in Phase One. Some of the sensors, in particularly the correvit, were loaned for only a short period of time (a week). This fact and the reliance on the weather again meant that there was no time for test repeats.

The same techniques described above were employed to specify the parameters that were measured, the data acquisition system and the sensors used. Once again, limited time was available to fit the system to the vehicle and repair the vehicle faults

(see sub-section 4.5.2) as the car had to be shipped back from competition in the USA.

The system consisted of a Pentium II, 600 MHz laptop running Windows NT with 128 Mb RAM. Thirteen channels were digitised using a National Instruments DAQCard-6062E card. This PCMCIA card digitised the thirteen analogue channels using a 12 bit A/D converter. The sampling rate was set at 200 Hz (see below) and the channels were measured as referenced single ended signals (i.e. each signal was measured with respect to the card ground reference signal). The digital data was recorded on the laptop's hard disk using a program written in MatLab, which allowed the files to be created in the same format as Phase One (program listings are given in Appendix F) and again, file numbers were automatically incremented to avoid overwriting. This system weighed approximately 20 kg, which, being lighter, was even more suitable for use with the vehicle (F5 weight was 300 kg). The vehicle performance parameters measured and the types of sensor are shown in table 4.2.

A case example of the development involved with the data acquisition system throughout the testing is the measurement of lateral and longitudinal velocity by the correvit sensor. In Phase One it was mounted at the rear of the vehicle, which was found to be flexible, changing the measured lateral and longitudinal velocity signals, during harsh manoeuvring. This is because the sensor is sensitive to its direction of travel and any angular offset produced an error in the measured data. Stiffening the mount during Phase One failed to completely eliminate the problem, so for Phase Two the sensor was solidly mounted to the front of the vehicle directly along its longitudinal access (see Figure 4.1). This significantly improved the data produced by the correvit.

Channel	Parameter Measured	Sensor Type Used	Make and Model
1	Lateral acceleration	Piezoelectric accelerometer -2g to +2g	Entran EGCS
2	Longitudinal acceleration	Piezoelectric accelerometer -2g to +2g	Entran EGCS
3	Yaw rate	Piezoelectric gyro -64 to +64 $degs^{-1}$	Systron Gyrochip AQRS
4	Roll rate	Piezoelectric gyro -64 to +64 $degs^{-1}$	Systron Gyrochip AQRS
5	Pitch rate	Piezoelectric gyro -64 to +64 $degs^{-1}$	Systron Gyrochip AQRS
6	Longitudinal speed	Correvit speed sensor 150 ms^{-1}	Datron Correvit S-CE
7	Lateral speed	Correvit speed sensor 20 ms^{-1}	Datron Correvit S-CE
8	Steering wheel angle	Rotational potentiometer 0 to 720 degrees	RS
9	Front brake line pressure	Piezoelectric pressure transducer 0 to 50 bar	Druck PMP 317
10	Rear brake line pressure	Piezoelectric pressure transducer 0 to 50 bar	Druck PMP 317
11	Front wheel rotational speed	Hall effect sensor 0 to 180 $rads^{-1}$	RS
12	Engine speed	Hall effect sensor 0 to 12000 rpm	RS
13	Throttle position	Rotational potentiometer 0 to 50 degrees	Webber

Table 4.2 – Phase Two sensors.

Due to the limited space and low weight of the vehicles, both data acquisition systems were small and compact and housed the laptop, the filters, the sensors and the power system in separate units which were distributed throughout each vehicle, see figure 4.1. The acceleration and gyro sensors were placed as close as possible to the centre of gravity, whilst the linear potentiometer was mounted to investigate the response of the steering system. In addition, a start/stop recording button and a data collection status L.E.D. were attached within easy reach and view of the driver.

A high frequency filter was applied to the analogue sensor signals during measurement on the vehicle to eliminate high frequency noise. Crolla reports [16] that the maximum input frequency generated by the driver is approximately 5 Hz and that the maximum handling response of the vehicle system is also around 5 Hz. To make sure no vehicle response data was filtered out, the width of the pass band was set to 16 Hz and all frequencies higher than the 16 Hz boundary were filtered digitally using an identical lowpass filter for each channel. This kept all the signals at the same band width and introduced equal phase shifts for each channel. The filter type is always a trade-off between the width of the transition band (a frequency domain characteristic) and the settling time of the step response (a time domain characteristic) and so it was decided to use a Butterworth filter [59] for efficiency. The magnitude of the frequency response of a Butterworth filter is given in equation (46).

$$B(j\omega) = \frac{1}{1 + \frac{\omega}{\omega_c}^{2n}} \qquad (46)$$

where:

ω: Signal frequency

$B(j\omega)$: Butterworth filter response

n: Filter order

The stop band was set to filter out 99 % of all the frequency content of the signal above 16 Hz. Using these criteria the cut off frequency was found to be at least 26 Hz using a sample rate of 165 Hz. This value was rounded up to 30 Hz using a sample rate of 200 Hz. The characteristic behaviour of the filter was then found to be 99.67 % pass at 16 Hz and 0.81 % pass at 200 Hz.

4.3 Obtaining the Vehicle Parameter Sets

Before each testing phase, a vehicle parameter set, with data logging system attached, was measured for each vehicle. To ensure consistency in each phase of testing, vehicle parameters were kept constant throughout. The parameters measured and the method of collection are listed below:

- Wheelbase, track, wheel radius, frontal area – measured.
- Vehicle overall mass, centre of gravity lateral, longitudinal and vertical position – found using electronic balances [3], see figure 4.2 a).
- Yaw inertia – found using a three point pendulum experiment and yaw gyro, see figure 4.2 b). Frequency of pendulum was found using measured yaw velocity given by gyro and from this the inertia of the pendulum was derived [14], i.e. the vehicle's yaw inertia.
- Front/rear sprung mass roll stiffness and roll centre heights – calculated using a suspension kinematics package.
- Coefficients of front/rear lift and overall vehicle drag – from wind tunnel measurement, see figure 4.2 c). The coefficient of drag of the F5 vehicle was found in Phase Two using the coast down manoeuvre (see the end of sub section 4.5.2).

- Tyre rolling resistance – also found using the coast down manoeuvre in Phase Two.
- Pacejka tyre model parameters – provided by Delft University [58].
- Engine torque curve – measured on an engine dynamometer, see figure 4.2 d).
- Engine gear ratios – engine workshop manual.
- Differential parameters – from Torsen literature [56].
- Brake system – system component specification detailed in company catalogue. Coefficient of friction of brake pad/disc interface provided by manufacturer.

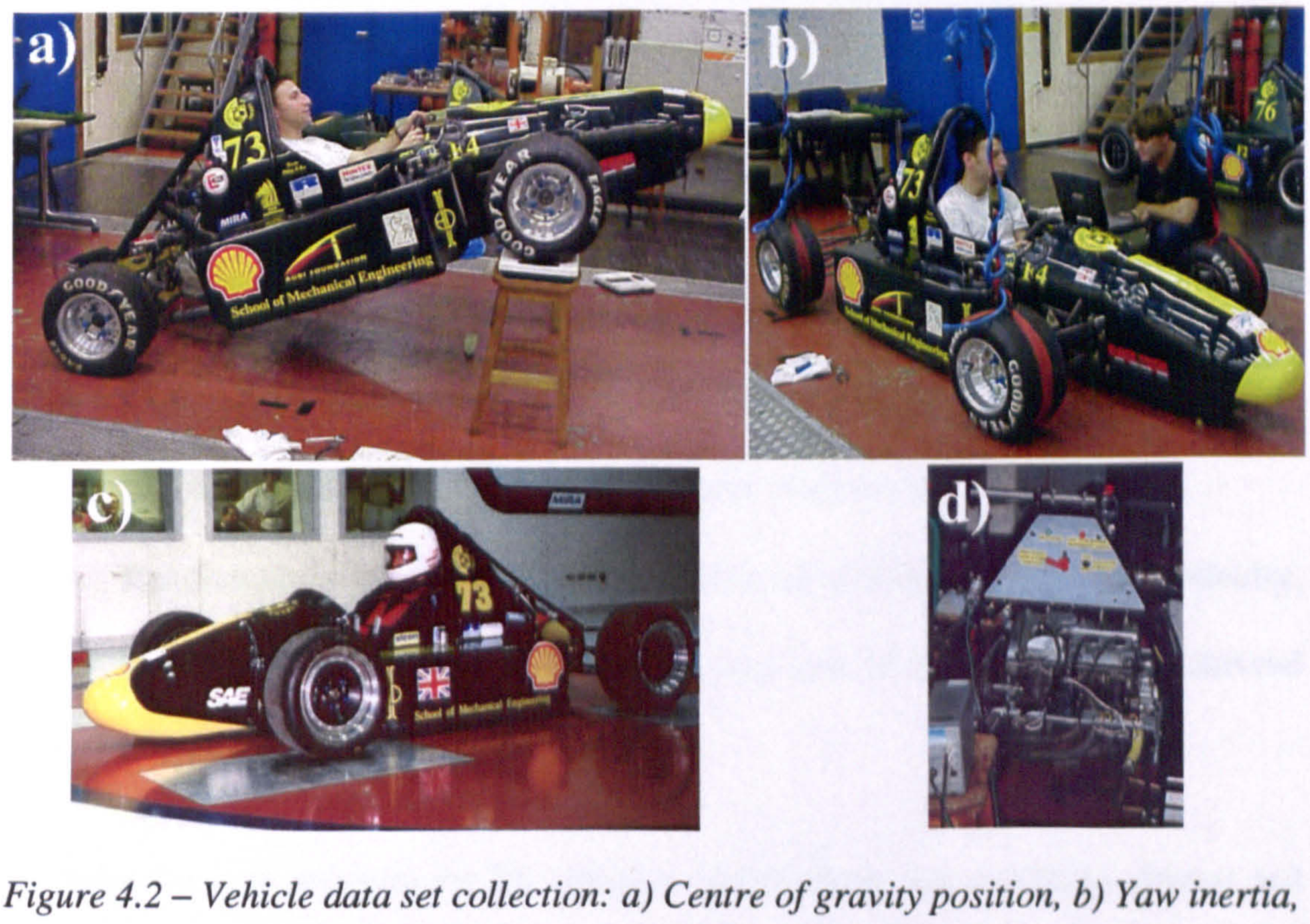


Figure 4.2 – Vehicle data set collection: a) Centre of gravity position, b) Yaw inertia, c) Aerodynamic coefficients, d) Engine torque curve.

4.4 Manoeuvres Undertaken

During Phase One, when measuring the F4 vehicle's lateral dynamic behaviour, several manoeuvres were conducted at constant forward velocities. The manoeuvres not only measured the steady state performance of the vehicle, but also its transient response. To check for consistency each manoeuvre was repeated several times. The manoeuvres conducted were:

- Steady state circle (constant path radius) manoeuvre – conducted for both left and right hand turns, from stationary up to the lateral acceleration limit of the vehicle by increasing the vehicle speed at a fixed steer angle. A complete recording (a sweep) made from start to finish and steady state 'snapshots' were taken at fixed forward velocities up to the limit of the vehicle's performance.
- J-turn (step steer input) manoeuvre – conducted for both left and right hand turns, at various constant forward velocities and degrees of steer input.
- Double lane-change manoeuvre – at various constant forward velocities using ISO test standard [60] to describe the path boundaries.
- Random steer input manoeuvre – conducted at a constant forward velocity, which enabled the frequency domain response of the vehicle to be derived from its time domain data.

In Phase Two, to measure the F5 vehicle's longitudinal and combined (lateral and longitudinal) dynamic behaviour, several further manoeuvres were conducted. Again, each manoeuvre was repeated several times for consistency and the manoeuvres conducted were:

- Acceleration manoeuvre – standing start acceleration to maximum forward velocity in straight line.
- Braking manoeuvre – from maximum forward velocity to a stop in straight line.
- Coast down manoeuvre – from maximum forward velocity to stop in straight line, without any driver inputs. Carried out twice in opposite directions.
- Hairpin manoeuvre – involved the vehicle negotiating a 180° hairpin type corner at the limit of its performance (braking into a corner and engine acceleration out of it). Conducted for both left and right hand corners.
- Slalom manoeuvre – weaving in and out of cones, with defined entry and exit gates and conducted at the limit of the vehicle's performance.

4.5 Results of Testing

Once the data had been recorded, further software filtering was required to attenuate high frequency noise that was superimposed on the low frequency vehicle responses that were being measured [16]. This noise was caused by high frequency vibrations in the vehicle body created by the engine and the tyre and road surface interaction. These filtering routines [59] can be seen in Appendix F and an example is given in figure 4.3 of the effect filtering has when applied to measured j-turn data. A Butterworth filter with a stop band of 7 Hz and a cut off frequency of 15 Hz was used. It can be seen that the filtering routines successfully remove the high frequency noise and allows the low frequency response of the vehicle to be examined.

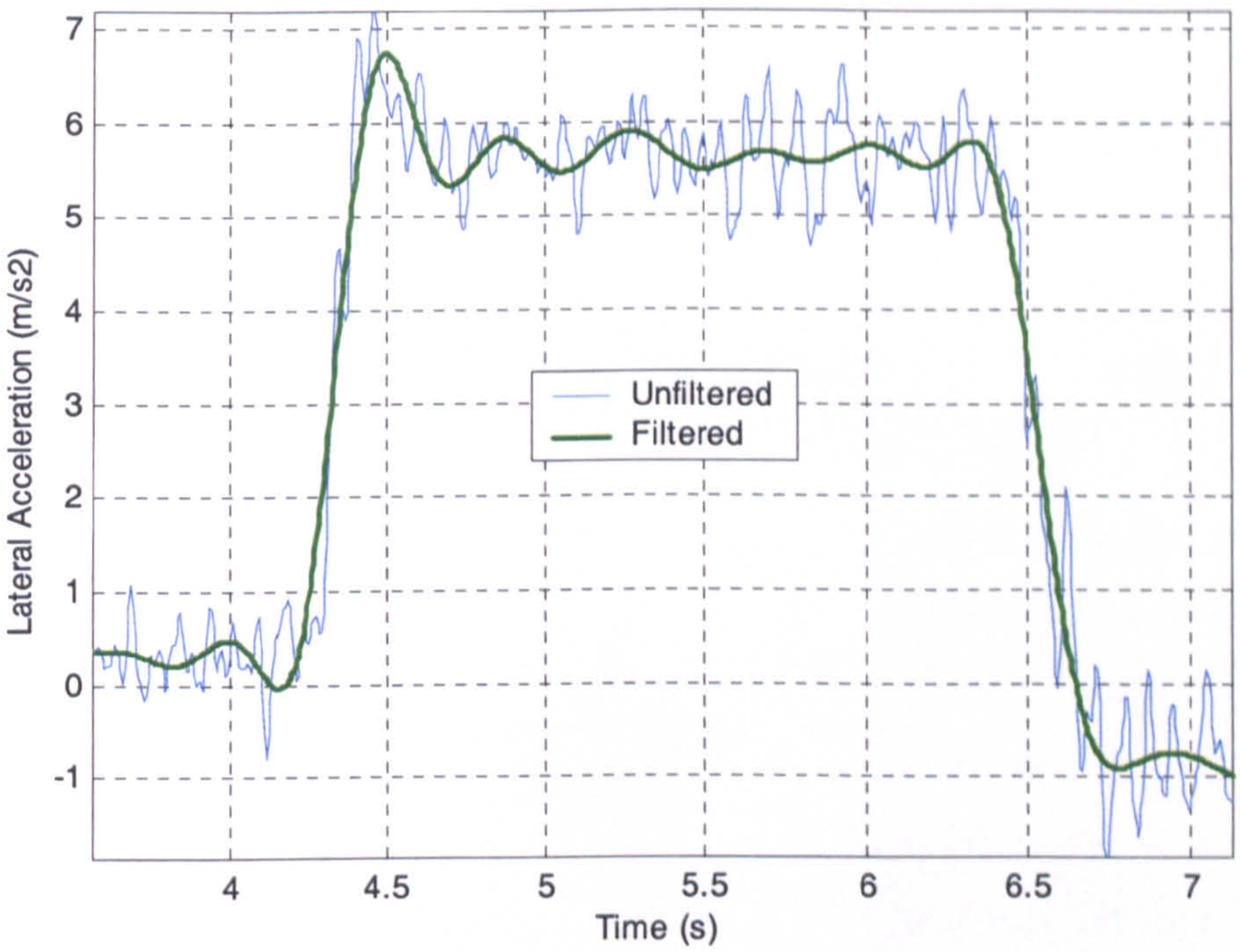


Figure 4.3 – Effect of filtering on lateral acceleration response in a j-turn manoeuvre.

Software filtering, however, was unable to attenuate the noise seen in the roll and pitch gyro measured data without affecting the underlying low frequency signal. The reason for this was that the gyros were measuring the small angular velocities seen in the stiffly sprung vehicle body (the maximum roll or pitch angles of the vehicles are approximately 1.5 degrees) and the noise generated in the vehicle body was of a similar magnitude to the angular velocities being measured. This noise also included low frequency signals, which masked the vehicle's body rotational responses that were being measured. Finally, as angular velocities were being measured, no steady state values could be extracted from the random noise.

4.5.1 Phase One Results

Table 4.3 details the data collected in Phase One and gives an indication of the variations made between runs.

Manoeuvre Undertaken	Number of Runs Measured	Details
Calibration	30	Stationary (with/without engine running) and before each set of manoeuvre runs.
Steady state circle manoeuvre	64 snapshots and 15 sweeps	For both left and right hand turns up to the limit of performance of the vehicle.
J-turn manoeuvre	40	For both left and right hand turns at 10, 11, 12, 13, 15 and 18 ms^{-1}.
Double lane-change manoeuvre	15	At 15, 26, 27, 28, 29 ms^{-1}.
Random steer input manoeuvre	15 minutes of data collected	At constant forward velocity.

Table 4.3 – Phase One testing results.

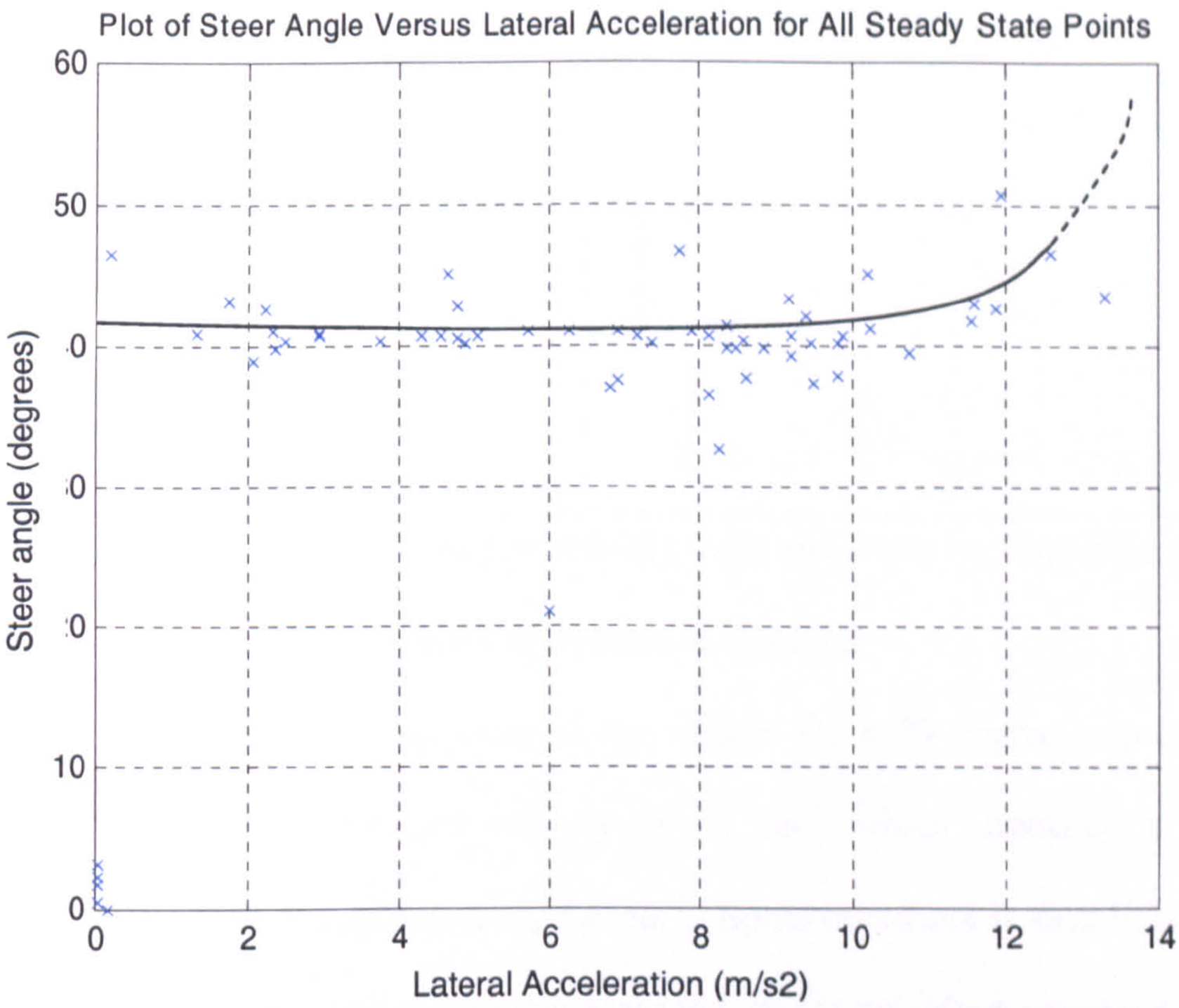


Figure 4.4 – Phase One steady state circle manoeuvre results.

Figure 4.4 shows the steady state snapshot results taken in the first testing phase. Sixty four snapshots were taken over the full range of the vehicle's lateral acceleration response and both left and right hand circles have been combined on the same graph. A best fit line has been placed on the plot, as well as the snapshots (denoted by a cross), and the theoretical continuation of the data is given by the broken line. Maximum lateral acceleration achievable was 13.4 ms^{-2} at a forward velocity of 15 ms^{-1} on a 16.8 m path radius. It can be seen that the vehicle's handling balance is neutral up to its limit of performance but, exhibits terminal understeer at the limit, due to the increase in steer angle needed to increase lateral acceleration [16]. This trend is similar to the results published by Miano et al [53] for another unspecified racing car.

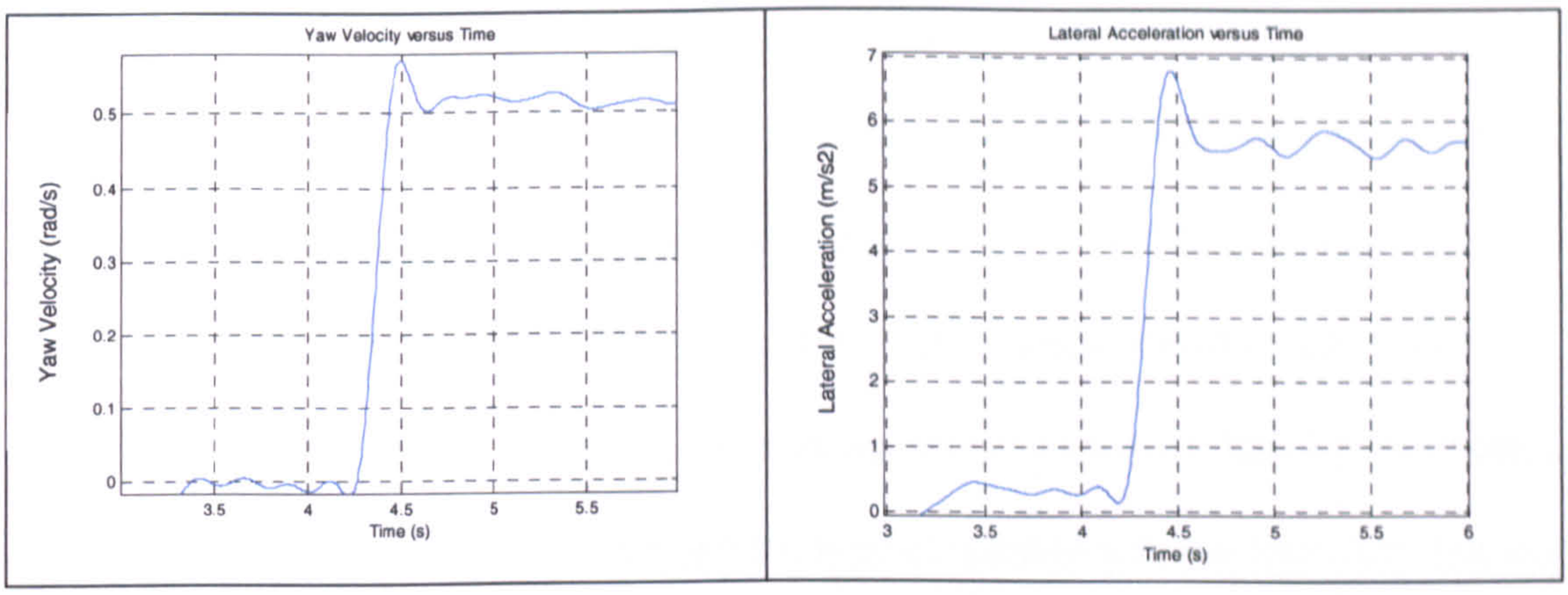


Figure 4.5 – Phase One measured yaw velocity (left) and lateral acceleration (right) responses in a j-turn manoeuvre.

Figure 4.5 shows the j-turn response of the vehicle for a 28 degree step steering wheel angle input at a forward velocity of 12 ms^{-1}, which produced a lateral acceleration response of approximately 5.6 ms^{-2}. Some overshoot is seen in both the yaw velocity and lateral acceleration responses but, this is quickly damped out in less than one oscillation. This compares favourably to the response given for a Ferrari by Crolla [16] (the Ferrari reached a much lower lateral acceleration of 3 ms^{-1}) and the results again show similar properties to the results published by Miano et al [53] for another unspecified racing car, undergoing a different j-turn manoeuvre.

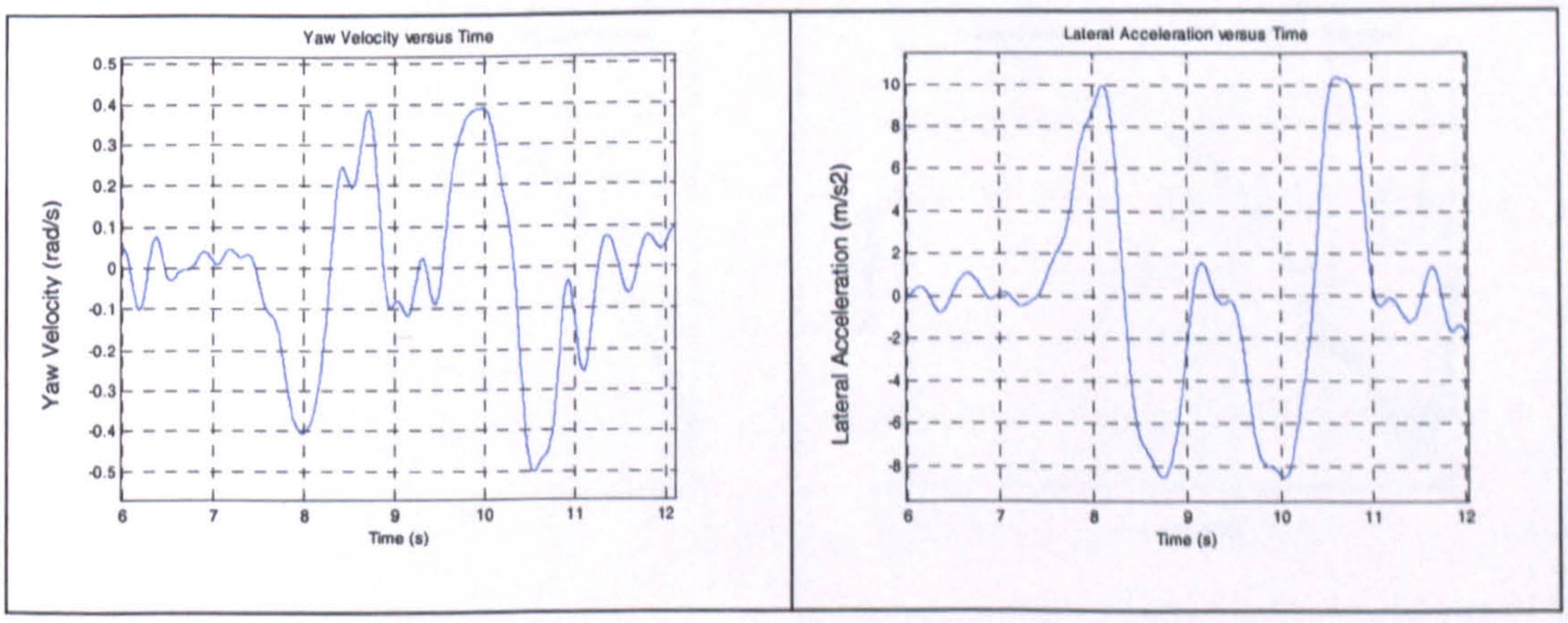


Figure 4.6 – Phase One measured yaw velocity (left) and lateral acceleration (right) responses in a lane-change manoeuvre.

Figure 4.6 shows the double lane-change manoeuvre response of the vehicle at a forward velocity of 28 ms^{-1}, giving a peak lateral acceleration response of approximately 10 ms^{-2}. As this manoeuvre is a standard ISO defined test it can be compared to the results presented by Miano et al [53] and shows that the vehicle has a similarly shaped response. Again, no analysis can be made on relative performance as non-dimensional axes were used and the type of vehicle was not specified, but the trends seen in the responses are similar.

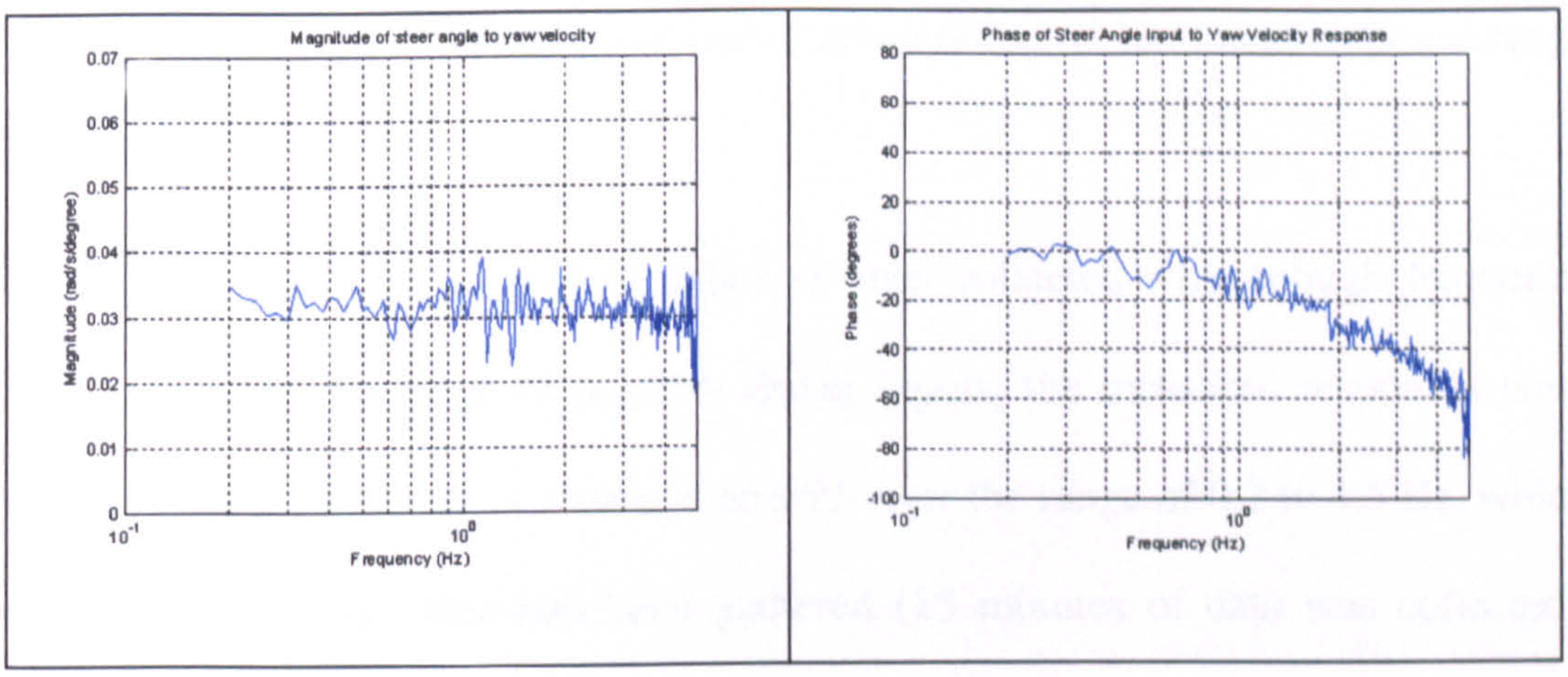


Figure 4.7 – Phase One magnitude (left) and phase (right) frequency domain responses taken from measured yaw velocity response of the F4 vehicle.

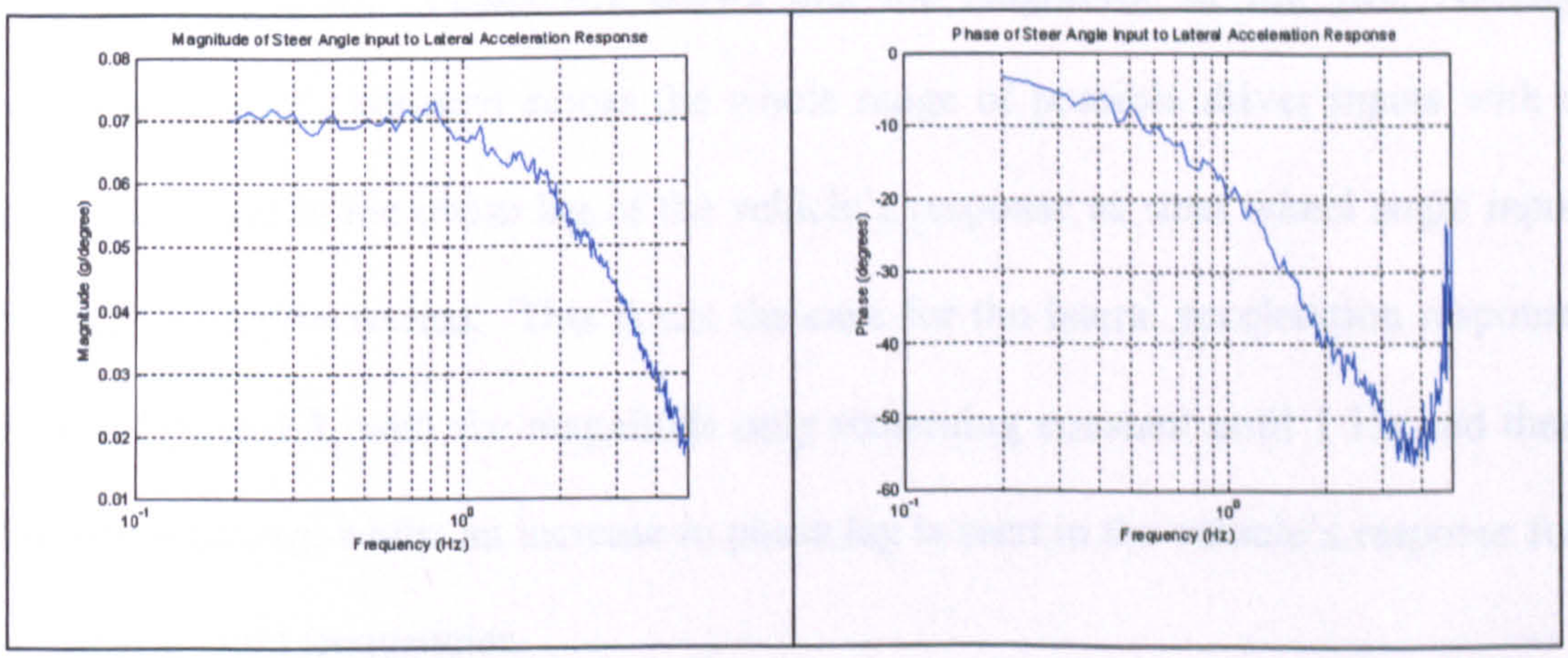


Figure 4.8 – Phase One magnitude (left) and phase (right) frequency domain responses taken from measured lateral acceleration response of the F4 vehicle.

Figures 4.7 and 4.8 show the frequency domain response of the vehicle, which were found from the random steer input time domain measured data by applying the Welch spectral analysis method [59] to find the complex transfer function between the steer angle input and the yaw velocity and steer angle input and lateral acceleration responses. The transfer functions show the response of the vehicle for a given driver input and are split into gain and phase for simplicity. To reiterate the test procedure, the driver applies a varying frequency steer input, at a constant forward velocity and a relatively low lateral acceleration (in the linear response range of the car).

To ensure an adequate amount of data had been collected with enough frequency content to cover the range of possible driver inputs, the measured responses were found to have a coherence of greater than 95% over the range of 0.2 to 4.5 Hz, which indicated that enough data had been gathered (15 minutes of data was collected). These routines can be seen in Appendix F.

The frequency domain responses allow a further examination of the vehicle's response properties. Figure 4.7 shows that the magnitude of the yaw velocity response is nearly constant across the whole range of possible driver inputs with a steady increase in the phase lag of the vehicle's response as steer wheel angle input frequencies are increased. This is not the case for the lateral acceleration response seen in figure 4.8, with the magnitude only remaining constant until 1 Hz and then linearly reducing, whilst an increase in phase lag is seen in the vehicle's response for increasing input frequencies.

Frequency response results are accepted as somewhat difficult to interpret, but flat gains are linked to consistency and small phase lags are linked to responsiveness [16]. Thus, the vehicle's response may be deemed to be desirable as the magnitudes

remain constant up until 1 Hz and the phase lags do not grow to greater than 70 degrees.

4.5.2 Phase Two Results

Table 4.4 details the data collected in Phase Two and gives an indication of the variations made between runs.

Manoeuvre Undertaken	Number of Runs Measured	Details
Calibration	20	Stationary (with/without engine running) and before each set of manoeuvre runs.
Acceleration manoeuvre	40	Times measured over 90m from standing start.
Braking manoeuvre	20	To a stand still.
Coast down manoeuvre	2	One made uphill against wind and the other downhill with wind.
Hairpin manoeuvre	30	Both left and right handed.
Slalom manoeuvre	15	Some runs made at constant forward velocity.

Table 4.4 – Phase Two testing results.

During Phase Two, it was found that all data had not been completely captured. This was due to errors made in the specification of some sensors and vehicle faults that occurred during testing. The problems encountered are listed below and the solutions in respect to the vehicle validation process given in the following chapter:

- The vehicle suffered from electrical problems causing it to be underpowered compared to the engine parameter map that had been measured earlier.

- The measured engine and front wheel rotational speeds were found to be saturating at low speeds due to the frequency to voltage conversion circuit design.
- The front and rear brake line pressure sensors saturated below, but close to, the maximum pressures attained in the braking system.

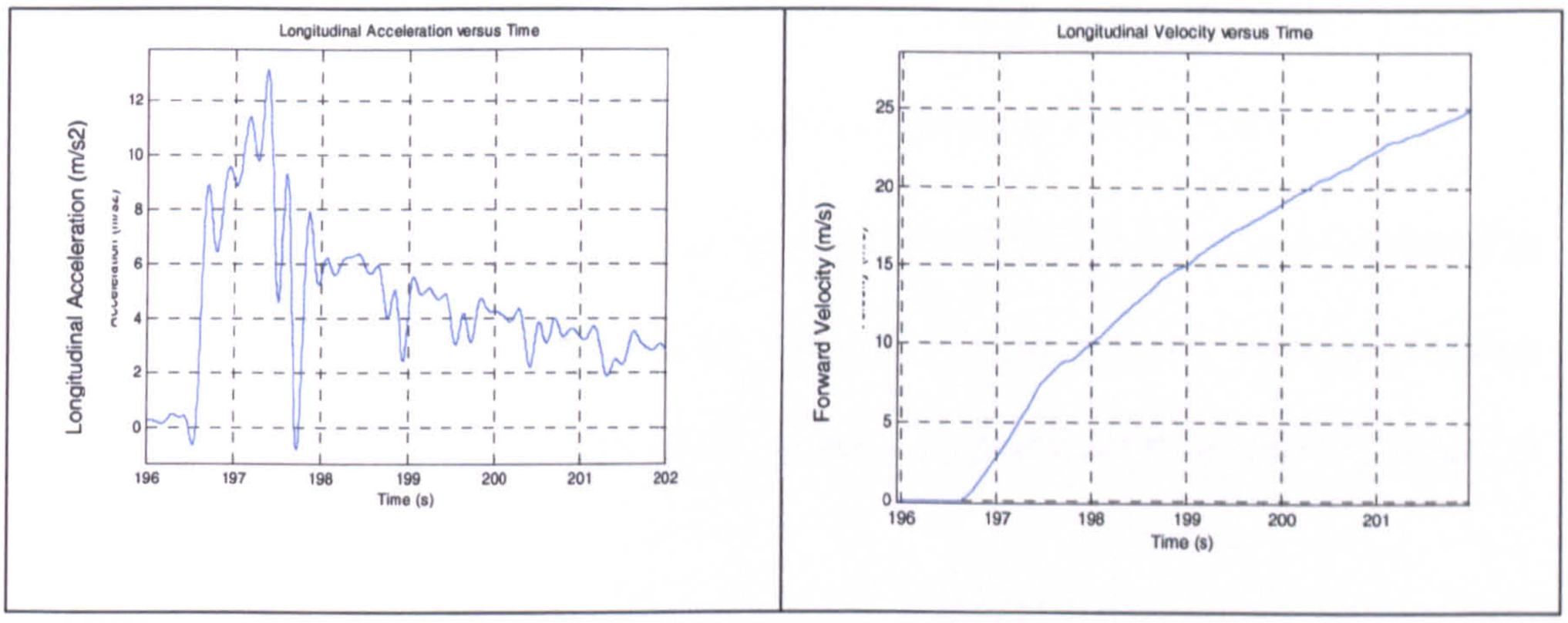


Figure 4.9 – Measured longitudinal acceleration (left) and longitudinal velocity (right) responses in a standing start acceleration manoeuvre.

Figure 4.9 shows an acceleration manoeuvre, where the vehicle accelerates from a standstill to 25 ms^{-1}. The sharp dips in longitudinal acceleration correspond to the gear change lag time and the vehicle reaches 6^{th} gear. The 0 to 60 mph time is seen to be 5.7 seconds which was due to the electrical fault mentioned above. Without the fault, 0 to 60 mph times of 3.5 seconds have been recorded by stopwatch measurements of the vehicle's performance whilst undergoing the same test.

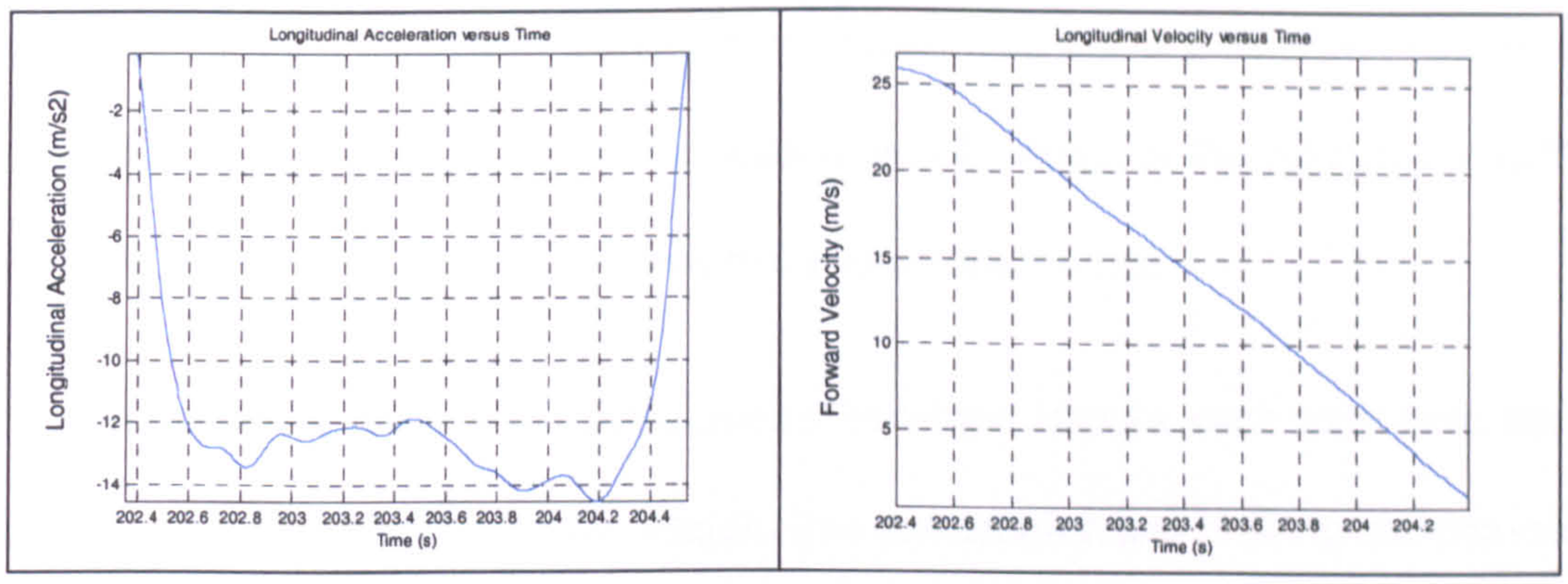


Figure 4.10 – Measured longitudinal acceleration (left) and longitudinal velocity (right) responses in a straight line braking manoeuvre.

Figure 4.10 shows a braking manoeuvre from just over 26 ms^{-1} to a standstill at approximately -13 ms^{-2}. This shows that the vehicle has an excellent braking response that allowed it to stop from 26 ms^{-1} to a standstill in approximately 2 seconds.

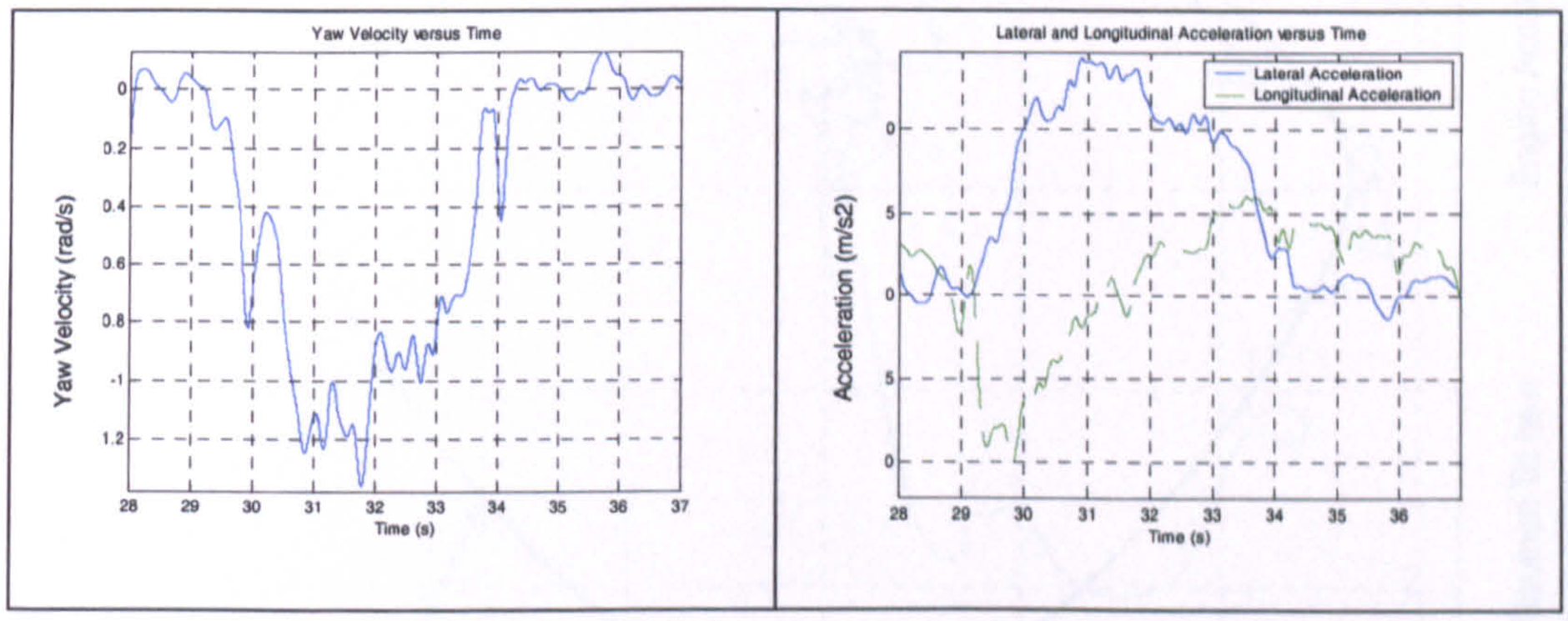


Figure 4.11 – Measured yaw velocity (left) and acceleration (right) responses for a hairpin manoeuvre.

Figure 4.11 shows the vehicle negotiating a 180 degree hairpin at its limit of performance. The hairpin has a 12.5m path radius at its centre-line and 5m track width. Figure 4.11 demonstrates how the driver decreases the vehicle's longitudinal deceleration to zero, whilst building up its lateral acceleration to a maximum at the apex, the driver is also seen to increase longitudinal acceleration away from the apex,

whilst diminishing the lateral acceleration. Examining the g-g acceleration plot in figure 4.12 shows how the driver takes maximum advantage of the vehicle's possible performance envelope by staying close to the envelope's edge.

The points where the vehicle performance envelope crosses each axis have been found from the steady state circle, straight line acceleration and braking manoeuvres. As such, these results are effectively steady state (or as close to steady state as possible with the longitudinal manoeuvring) values and so, whilst undergoing the transient hairpin manoeuvre the vehicle crosses this steady state performance limit line [3].

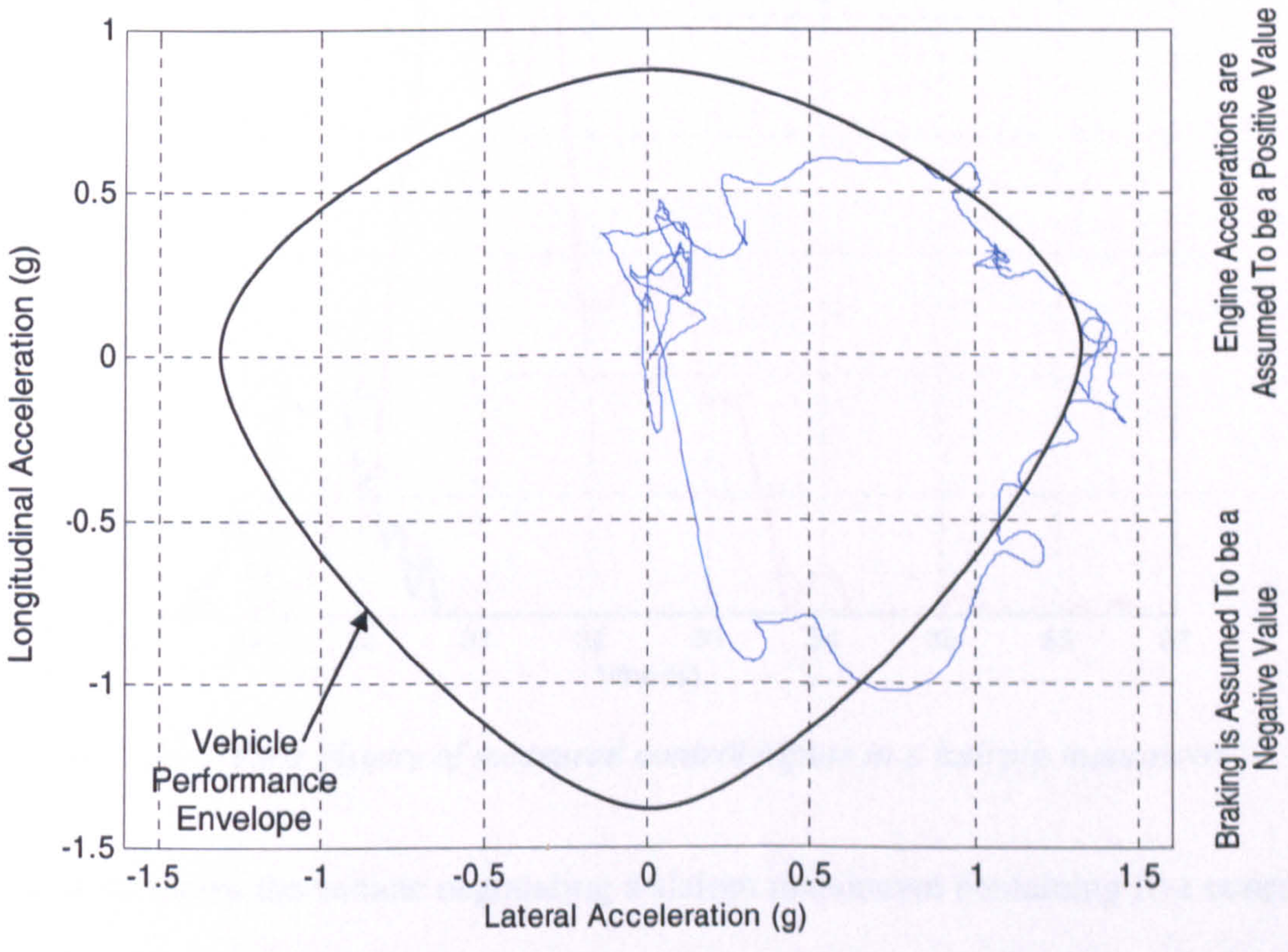


Figure 4.12 – g-g diagram for hairpin manoeuvre.

From observations and feedback from the driver, it was seen that the driver followed a racing line through the corner. The racing line is produced by a driving technique commonly referred to as 'apexing', where the driver ensures that the vehicle's

minimum speed, maximum lateral acceleration and thus minimum path radius occurs at the apex of a corner [3]. Along with the subjective observations of the vehicle's path, the driver control inputs seen in figure 4.13, further implies the use of this driving technique and the corner apex is assumed to be reached at just under 31 seconds, where the longitudinal controls are zero and lateral acceleration is at a maximum (see figure 4.11).

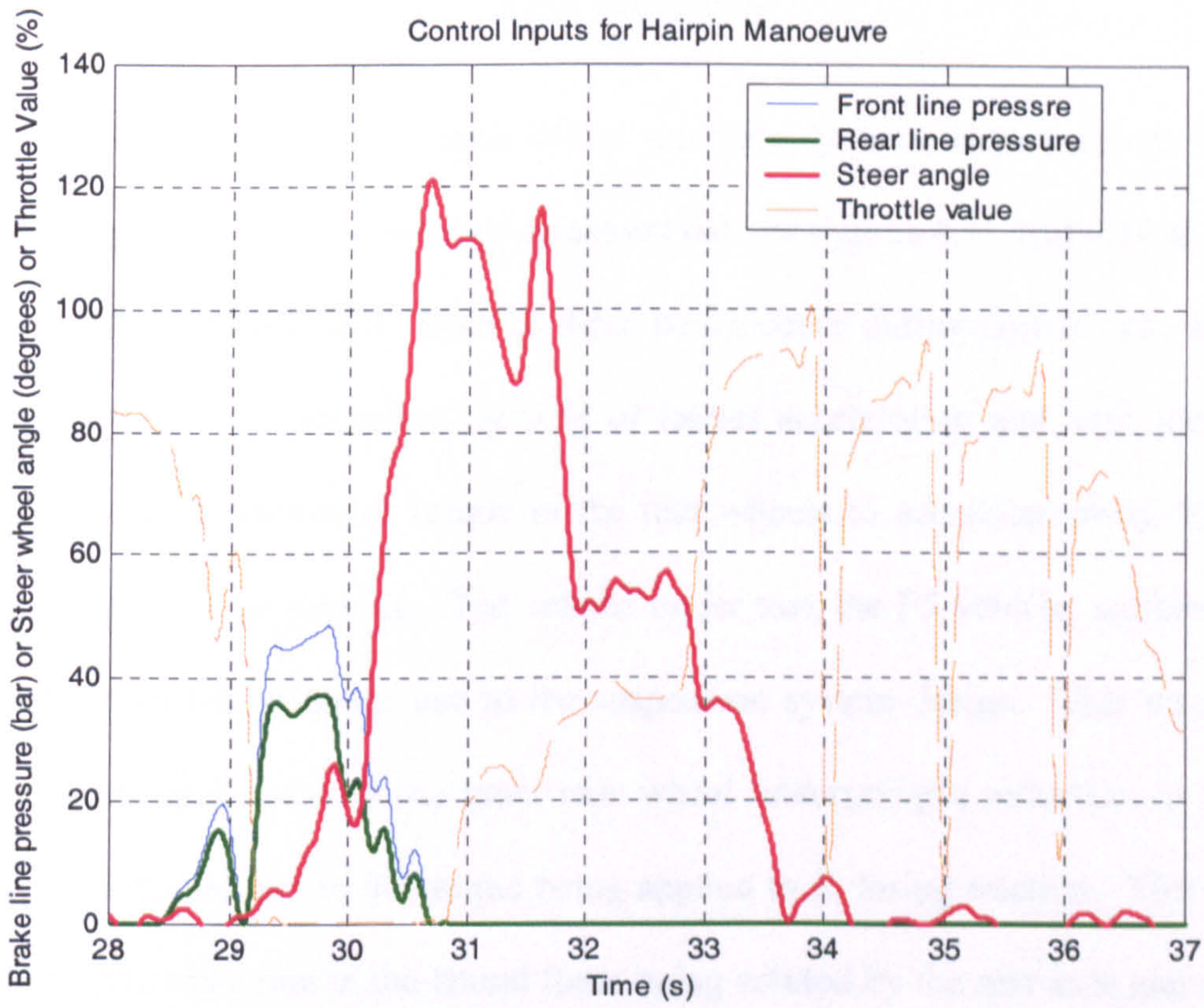


Figure 4.13 – Time history of measured control inputs in a hairpin manoeuvre.

Figure 4.14 shows the vehicle negotiating a slalom manoeuvre containing five cones and an entry and exit gate. As expected, the driver is seen to wait until the final cone before accelerating out of the manoeuvre, at the limit of the vehicle's performance. The cones are also negotiated very close to the limit of the vehicle's performance with lateral acceleration peaking at approximately 13 ms^{-2}.

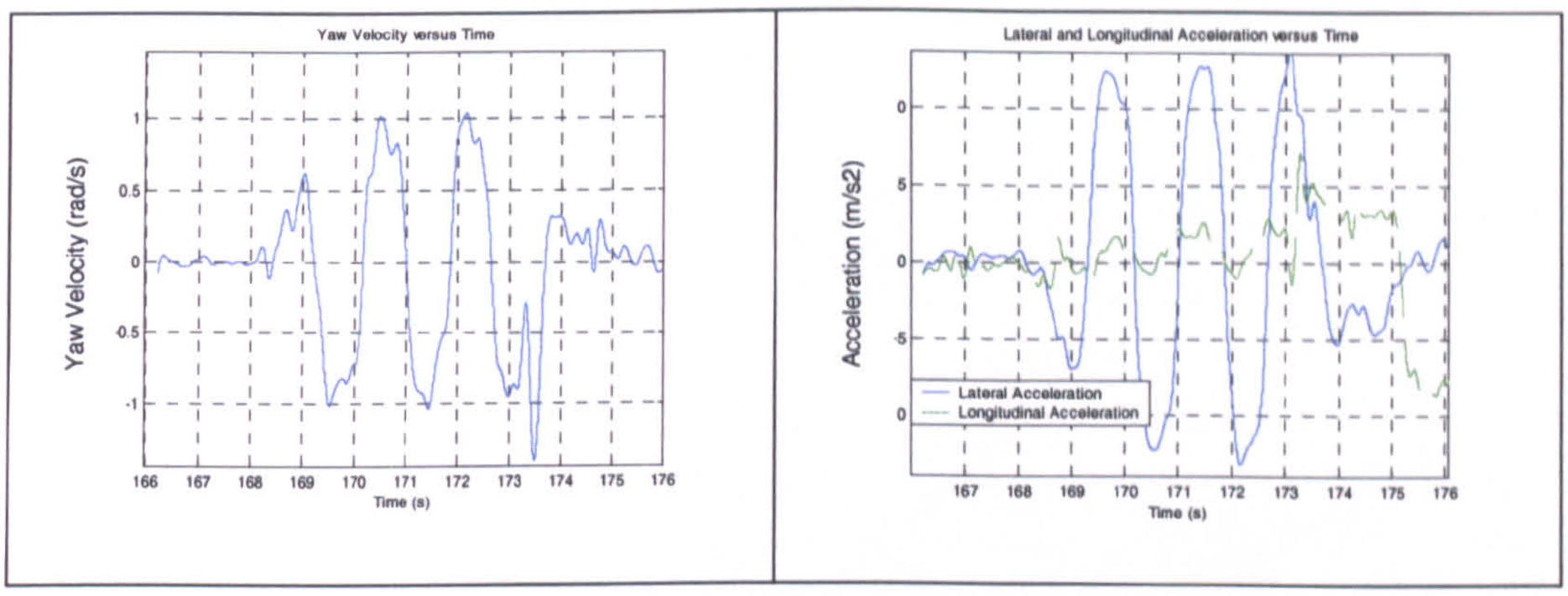


Figure 4.14 – Measured yaw velocity (left) and acceleration (right) responses in a slalom manoeuvre.

The only undesirable characteristic of the vehicle's dynamic behaviour can be seen as a short peak in the yaw velocity measured data of figures 4.11 and 4.14 at 34 and 173.5 seconds respectively. Both of these peaks occur during corner exit, with the vehicle undergoing approximately 0.5g of lateral acceleration and with the driver applying a large amount of torque to the rear wheels to accelerate away from the corner as quickly as possible. The vehicle under test, the F5 vehicle, suffered from rear wheel lift off problems due to the suspension system design. This meant that these spikes are caused by the inner rear wheel undergoing a reduction in normal force (lift off) and due to the torque being applied to it, losing traction. This causes an immediate reduction in the lateral force being created by the rear axle and a sharp increase in the yaw velocity (i.e. the back end stepping out in an oversteer effect). This only happened instantaneously and did not lead to the vehicle going into an uncontrolled spin because of the driver's control inputs. Previous subjective observations had made the problem apparent but, the measured data now proves that there is wheel lift off and quantifies the reduction in performance that occurs, as a result.

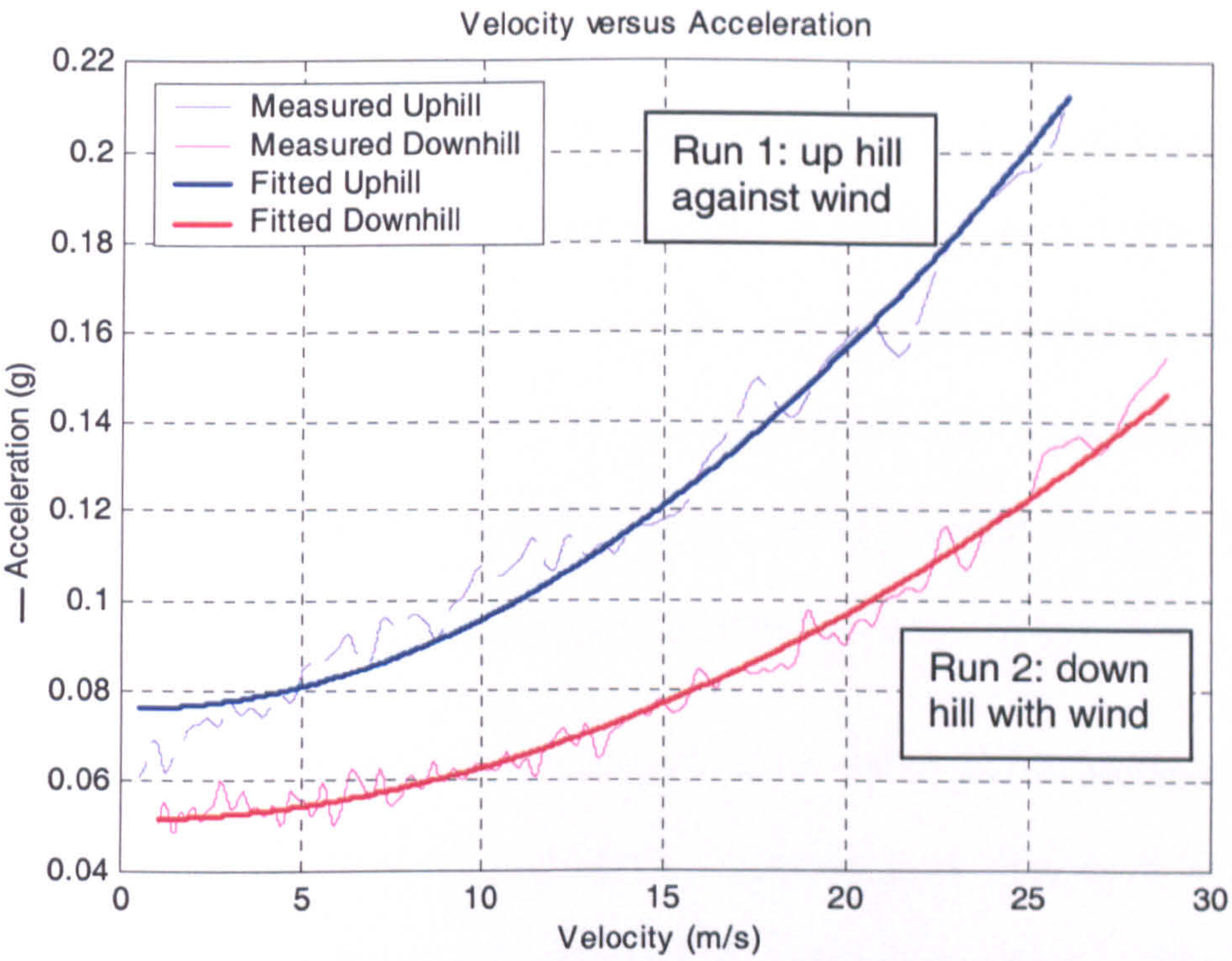


Figure 4.15 – Coast down manoeuvre curve fitting results.

The coast down manoeuvre allowed the identification of the F5 vehicle's aerodynamic drag and tyre rolling resistance coefficients. An optimisation routine was used to vary these parameters in an equation of motion of the vehicle coasting (equation (47)). The equation's response was then compared against measured data and the coefficients varied until the closest match was found, see figure 4.15. This was done for both Run One and Run Two and the results averaged to eliminate the effect of ambient wind conditions and track slope. The aerodynamic drag coefficient of 1.05 that was derived closely matches that found in the wind tunnel for the F4 vehicle of 0.98 (which is nearly identical in design) and the tyre rolling resistance coefficient of 0.016 is identical to that quoted in literature for a similar tyre [14].

$$-A_x = \frac{\left(0.5.\rho.FA.C_d.u^2\right)+\left(4.\mu_{rr}.m.g\right)}{m} \qquad (47)$$

4.6 Conclusions

It can be seen that a large amount of vehicle handling data has successfully been recorded for a racing car in two testing phases, using purpose built data acquisition systems. The data not only describes the vehicle's individual lateral or longitudinal dynamic handling behaviour, but also combined lateral and longitudinal dynamic handling behaviour. These results represent a new contribution to published knowledge in the area of practical racing car handling measured data [55].

Overall, the data confirms that the vehicles have stable and responsive handling properties, which can reach a peak lateral acceleration of 13.4 ms^{-2}, longitudinal engine acceleration of 10 ms^{-1} and a braking deceleration of -14 ms^{-2}. The use of the driver applying the 'apexing' technique to negotiate the hairpin manoeuvre at the limit of the vehicle's performance has also been shown.

In addition, comprehensive parameter sets for the vehicles have been found and data filtering and handing routines produced. These results are critical in allowing a full validation of the vehicle models described in Chapter Three, by comparison of measured and simulated responses.

5 Model Validation

5.1 Introduction

The initial objective of the chapter is to judge the accuracy of the lateral dynamic behaviour of two different vehicle models. This is undertaken using a comparison study between the simple and sophisticated vehicle models that were detailed in Chapter Three and the measured data collected in the testing Phase One and detailed in Chapter Four. The second objective is to validate the sophisticated vehicle model's longitudinal and combined lateral and longitudinal dynamic responses using the data collected in the testing Phase Two, also detailed in Chapter Four.

The test manoeuvres were simulated by the MatLab vehicle models and the vehicle parameter sets collected were used to represent the F4 and F5 vehicles. The Phase One manoeuvres were simulated using the measured steer angle as a time history input to control the vehicle model at a constant forward velocity (which was also measured on the vehicle).

An exception to this was the steady state simulated results, which were created by applying a constant steer angle and forward velocity. The lateral acceleration values were only recorded when the yaw acceleration had reduced to zero and this was done for increasing forward velocities up to the limit of the vehicle's lateral acceleration performance.

The simulated frequency domain responses were found from the random steer input manoeuvre time domain responses using the Welch spectral analysis method [59] to find the complex transfer function between the steer input and the yaw velocity and

lateral acceleration responses. Phase Two manoeuvres were simulated using the measured control inputs (steer angle, front and rear brake line pressures and throttle position) as time history control inputs to the vehicle model, after specifying an initial forward velocity.

5.2 Lateral Dynamic Behaviour Comparison Study

This section details a comparison study between the Phase One measured results and the simple and sophisticated vehicle model simulated results. As detailed in the previous chapter, the steady state lateral acceleration response of the actual vehicle was measured using the steady state circle manoeuvre. The transient response of the actual vehicle was measured using the j-turn, lane-change and random steer manoeuvres. The j-turn and lane-change manoeuvres allow a comparison of the time domain responses, whilst the random steer manoeuvre allows a comparison of the frequency domain responses.

5.2.1 Simple Vehicle Model

The sub-section shows a comparison study between the Phase One measured results and the simple vehicle model simulated results. The study is used to find the accuracy of the vehicle model in predicting the actual vehicle's lateral performance and includes comparisons for the steady state circle, j-turn, lane-change and random steer manoeuvres.

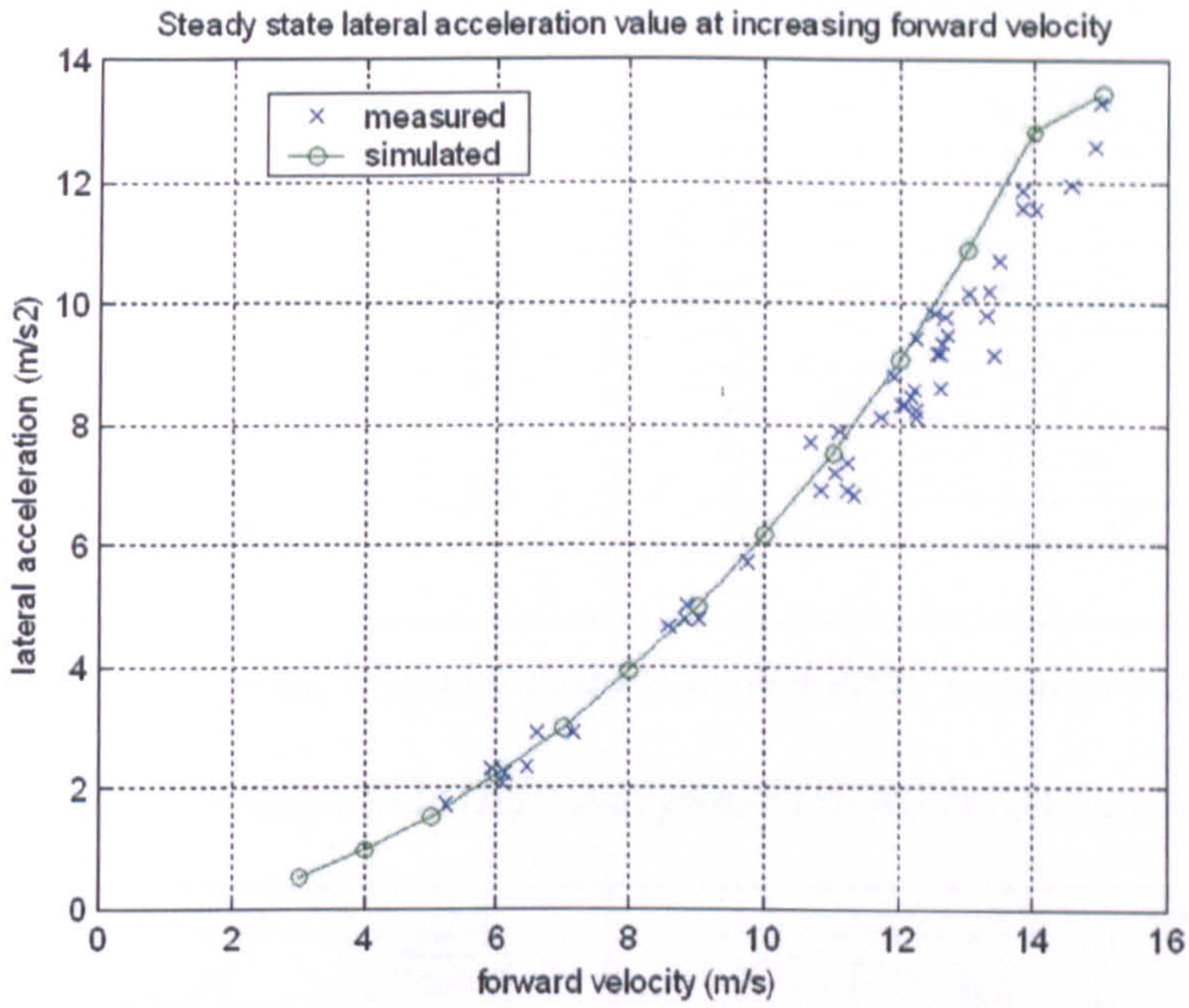


Figure 5.1 – Simple vehicle model and measured data, steady state responses.

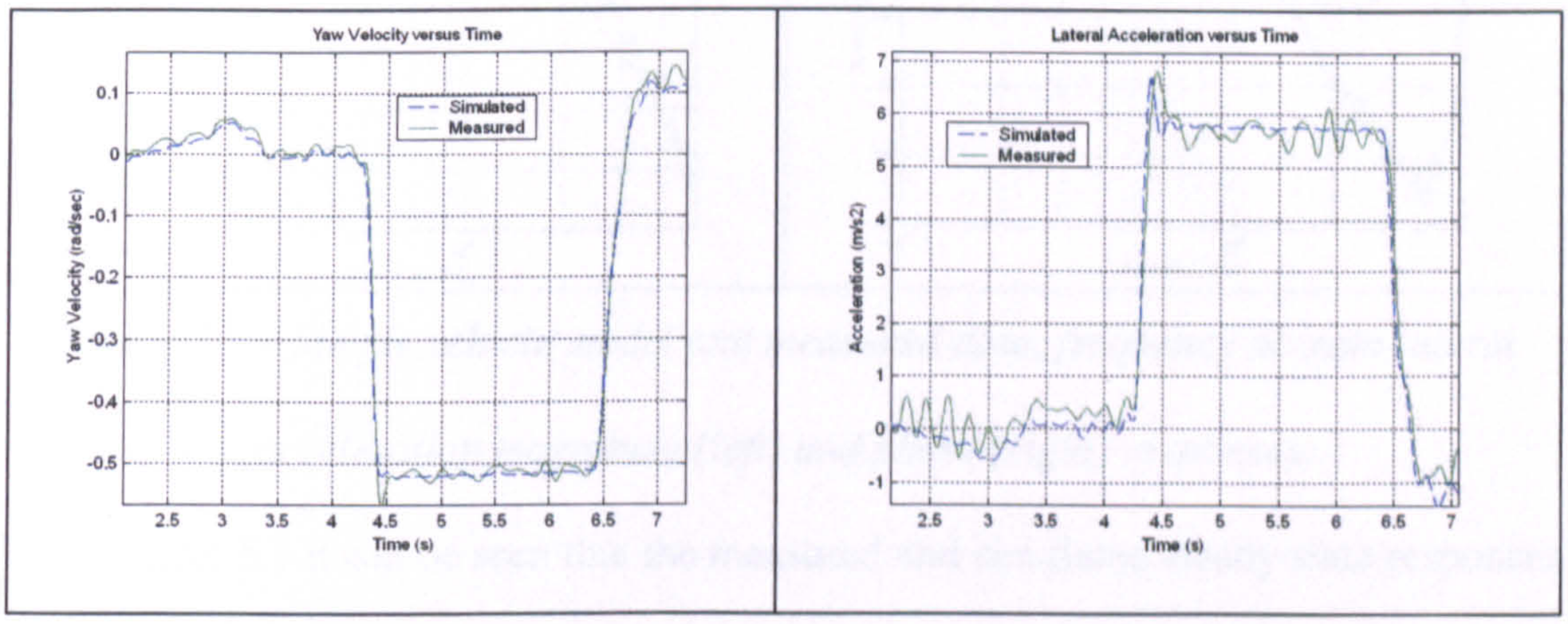


Figure 5.2 – Simple vehicle model and measured data, j-turn manoeuvre yaw velocity (left) and lateral acceleration (right) responses.

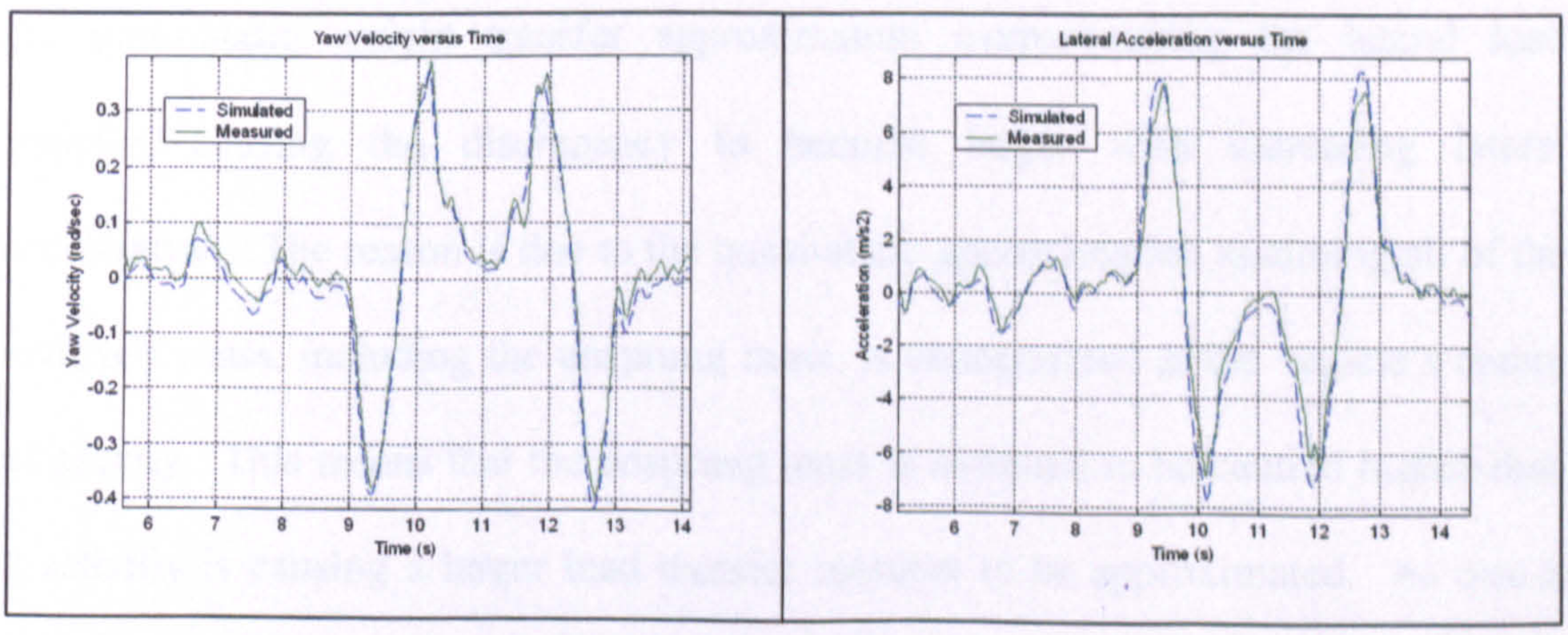


Figure 5.3 – Simple vehicle model and measured data, lane-change manoeuvre yaw velocity (left) and lateral acceleration (right) responses.

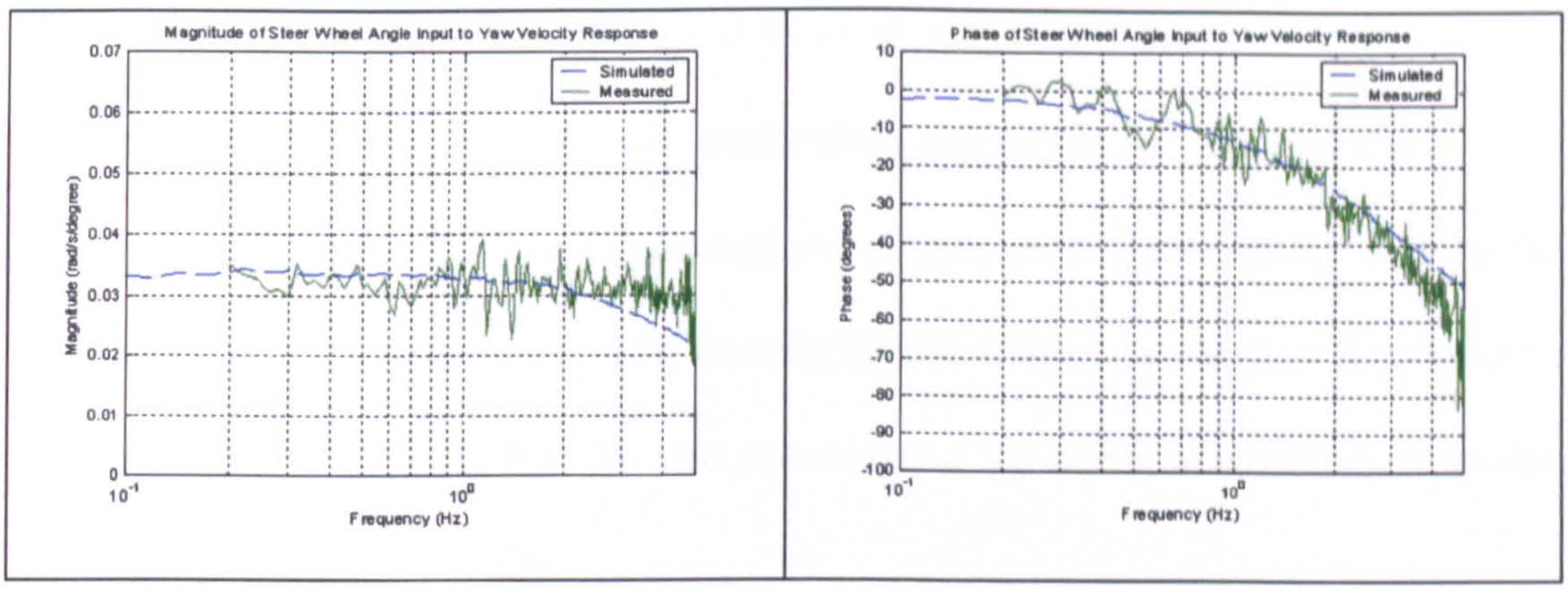


Figure 5.4 – Simple vehicle model and measured data, frequency domain yaw velocity magnitude (left) and phase (right) responses.

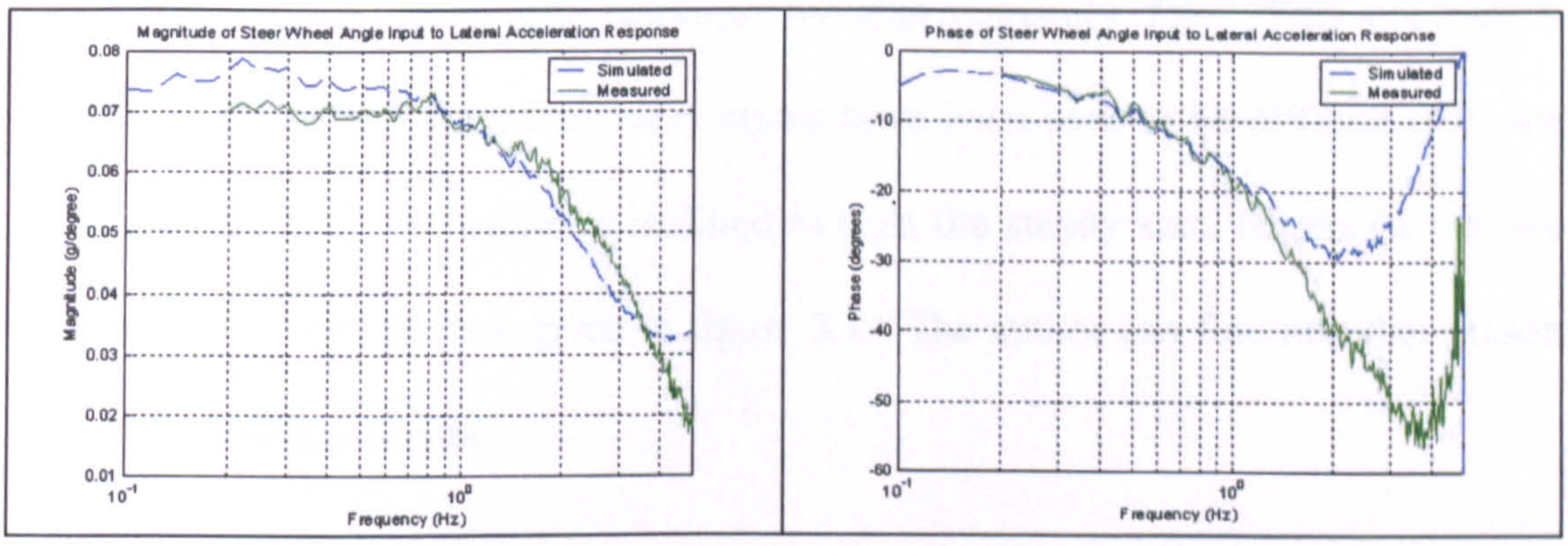


Figure 5.5 – Simple vehicle model and measured data, frequency domain lateral acceleration magnitude (left) and phase (right) responses.

From figure 5.1 it can be seen that the measured and simulated steady state responses closely match, except that the simulation makes a slight overestimation, which becomes greater at higher lateral accelerations. This inaccuracy is probably due to the quasi-static weight transfer approximation overestimating the lateral load transfer, causing the discrepancy to become larger with increasing lateral acceleration. The reason is due to the quasi-static approximation assuming all of the vehicle's mass, including the unsprung mass, is concentrated at the vehicle's centre of gravity. This means that the unsprung mass is assumed to be centred higher than it actually is causing a larger load transfer moment to be approximated. As this is directly proportional to lateral acceleration, the discrepancy increases with lateral acceleration [3].

Figures 5.2 and 5.3 show a very good level of agreement between measured and simulated yaw velocity and lateral acceleration responses for the j-turn and lane-change manoeuvres. This indicates that the simple model accurately predicts the vehicle's transient performance but, examining the frequency domain responses in figures 5.4 and 5.5, shows that the yaw velocity and the lateral acceleration response agreement is poor above 1 Hz.

Below 0.7 Hz, however, the vehicle model does slightly overestimate the measured lateral acceleration magnitude response by approximately 7%. This contradicts figure 5.1, but as low frequency steer inputs have been seen to be difficult to create using a driver [61], the author is inclined to trust the steady state results (which are simpler to produce) and are given in figure 5.1. The author can find no other reason why the difference occurs.

The factors that influence the frequency domain response of the vehicle were previously investigated by Heydinger et al [62] where it was found that the low frequency range up to 0.5 Hz is influenced primarily by the vehicle's steady state response. The medium frequency range from 0.5 to 1.5 Hz is influenced by the sprung mass dynamics and some low frequency steering system effects. The high frequency response above 1.5 Hz is influenced by the tyre lag dynamics and the steering system response.

The vehicle's steering system was designed to be sufficiently stiff so that it did not have an effect on the vehicle's handling behaviour. Torsional stiffness analysis and static experimentation reaffirmed the system's stiffness. To verify its dynamic response, the frequency domain response of the rack displacement in relation to the

steer wheel angle input was found and is shown in figure 5.6. The coherence of this measured data was found to be greater than 95% over the range of 0.2 to 6 Hz.

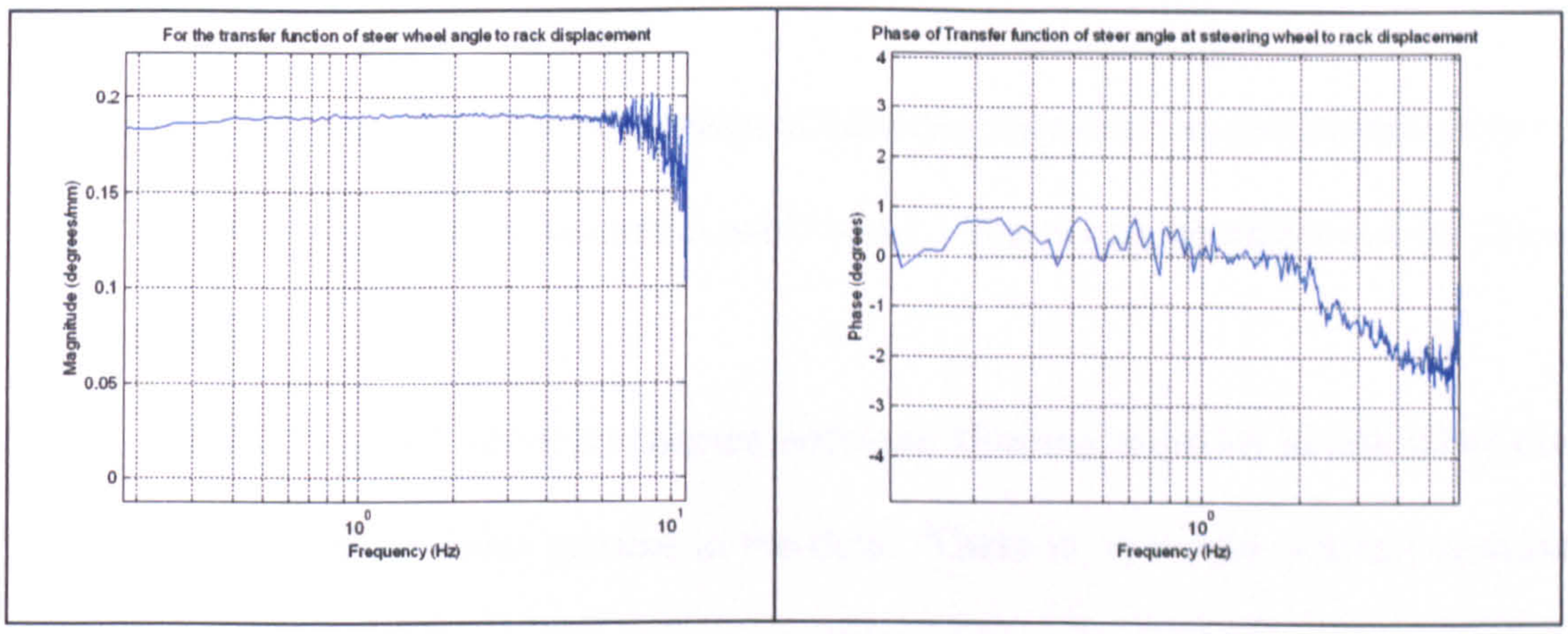


Figure 5.6 – Measured steering system frequency domain magnitude (left) and phase (right) responses.

The near constant response of the system across the full range of inputs shown in figure 5.6 implies that the steering system is sufficiently stiff and only has a minor effect on the response of the vehicle's handling behaviour, therefore it has correctly been modelled as a simple gain on steer wheel angle input. This is also true of the relationship between rack displacement and wheel steer angle which was found to be a nearly linear relationship using a suspension kinematics package [63] and has again found, through torsional stiffness analysis and static measurement, to be very stiff and modelled as a simple gain.

Consequently, as the sprung mass roll and tyre lag dynamics are not accounted for in the vehicle model, these are likely to be the factors affecting the frequency domain lateral acceleration response and causing the inaccuracy. This simple model, therefore, may be used with confidence to simulate the vehicle's steady state response and transient manoeuvres below 1 Hz. Accordingly, it is suitable for use in a lap time simulation package that uses a quasi-static approximation of the vehicle's

performance, because only the steady state response of the vehicle model is found. In addition its simplicity relative to a more sophisticated vehicle model will give reduced simulation times.

Although there is a high level of agreement between measured and simulated results, it can be seen that small inaccuracies still occur. These can be explained by three phenomena:

1. As shown in the previous chapter, software filtering removes nearly all of the high frequency noise present in the data. There is, however, a small amount of low frequency noise present. This cannot be completely attenuated without affecting the responses that were being measured [59].
2. The software filtering conducted resulted in a small phase lag, which became more apparent at higher frequencies [59]. An attempt was made to minimise this phase lag by design of the filter.
3. The actual vehicle's forward velocity did vary slightly during the testing, whereas it was kept constant in the simulations. The variation was, in each case, less than 2% and so will not have a significant impact on the measured results but caused minor discrepancies.

5.2.2 Sophisticated Vehicle Model

The sub-section shows a comparison study between the Phase One measured results and the sophisticated vehicle model simulated results. As with the previous sub-section the study is used to find the accuracy of the vehicle model in predicting the actual vehicle's lateral performance and includes comparisons for the steady state circle, j-turn, lane-change and random steer manoeuvres.

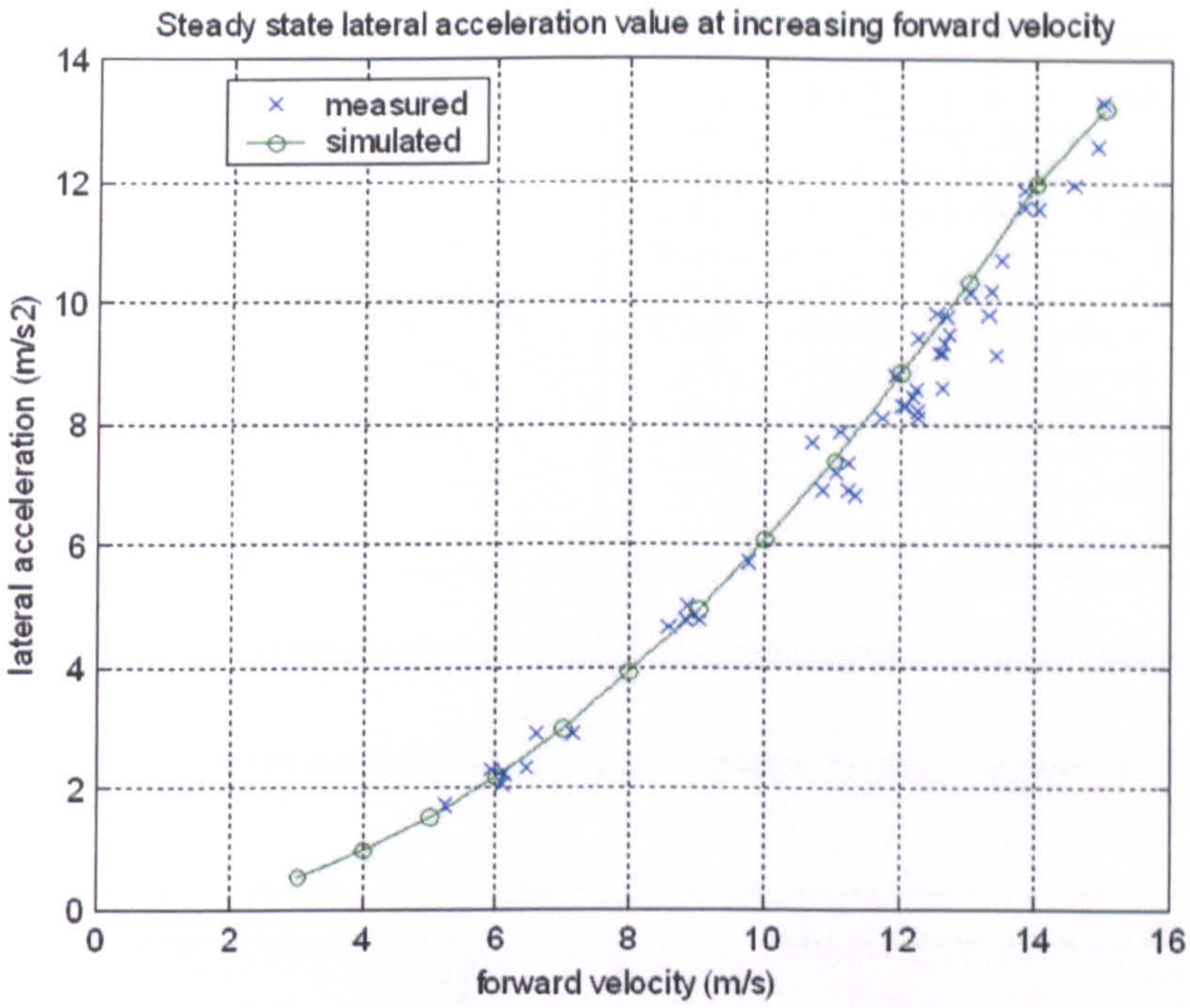


Figure 5.7 – Sophisticated vehicle model and measured data, steady state response.

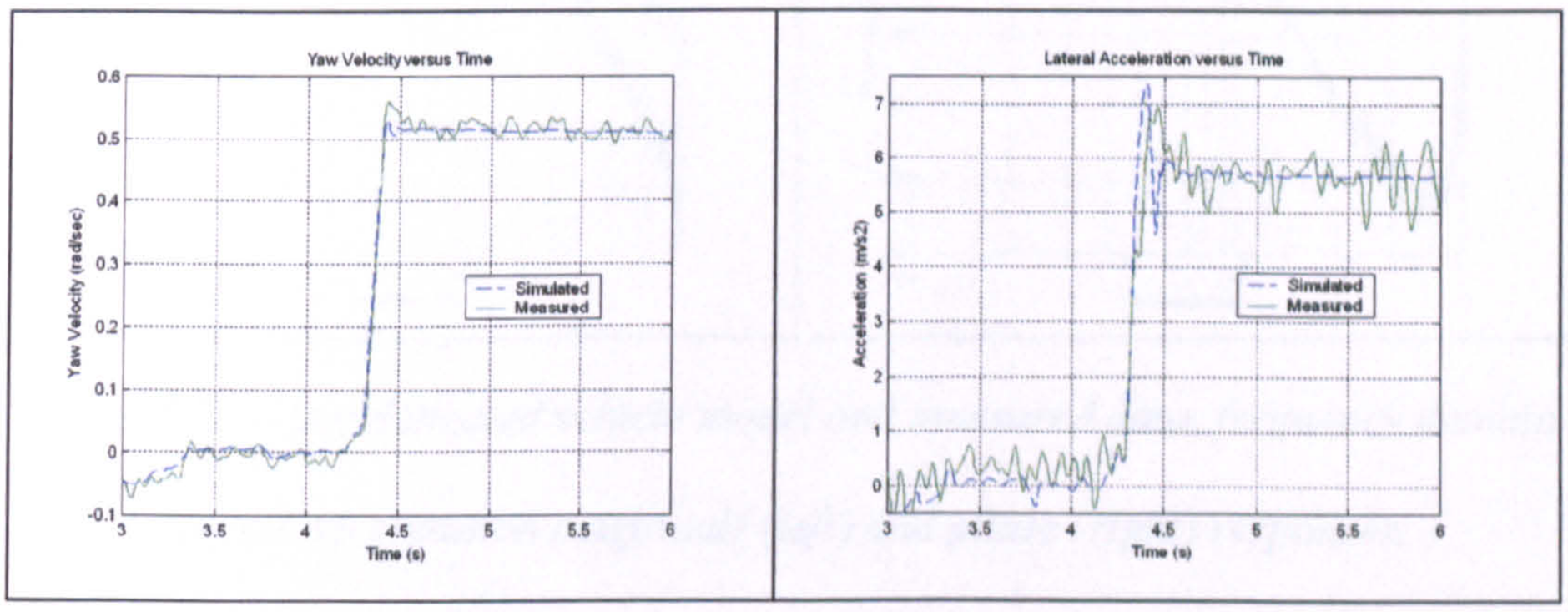


Figure 5.8 – Sophisticated vehicle model and measured data, j-turn manoeuvre yaw velocity (left) and lateral acceleration (right) responses.

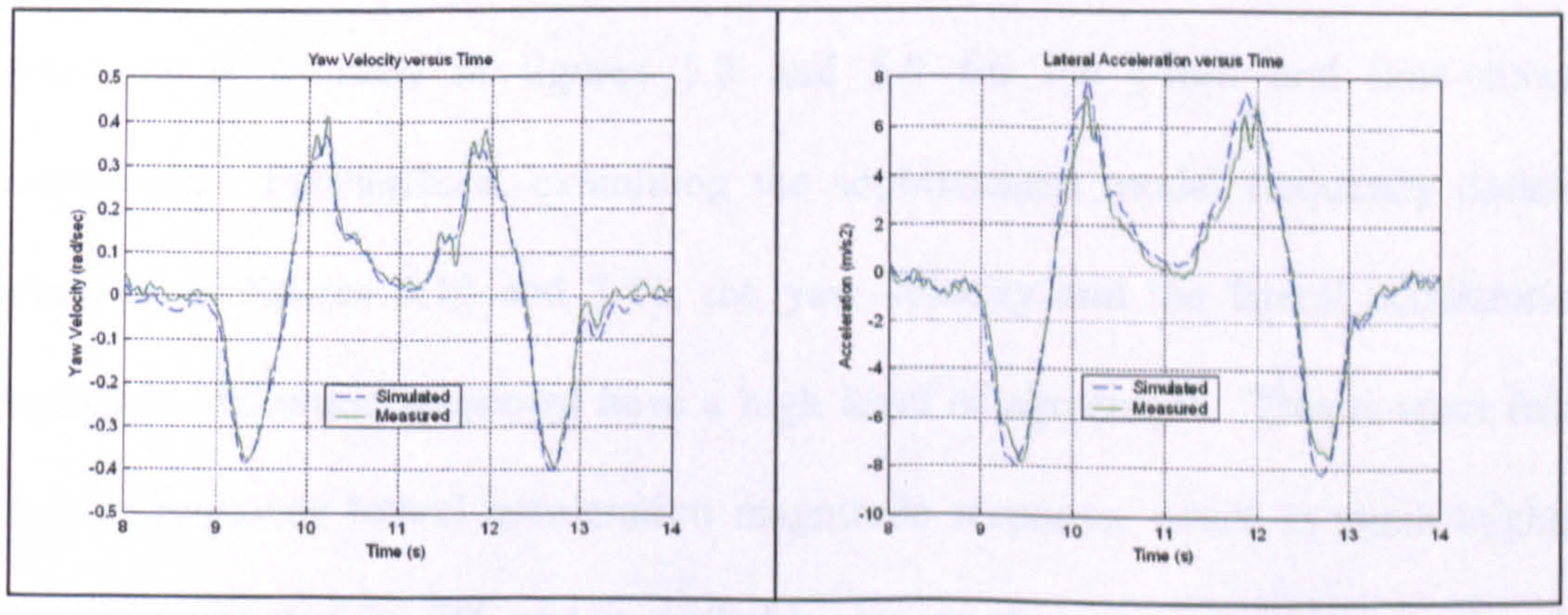


Figure 5.9 – Sophisticated vehicle model and measured data, lane-change manoeuvre yaw velocity (left) and lateral acceleration (right) responses.

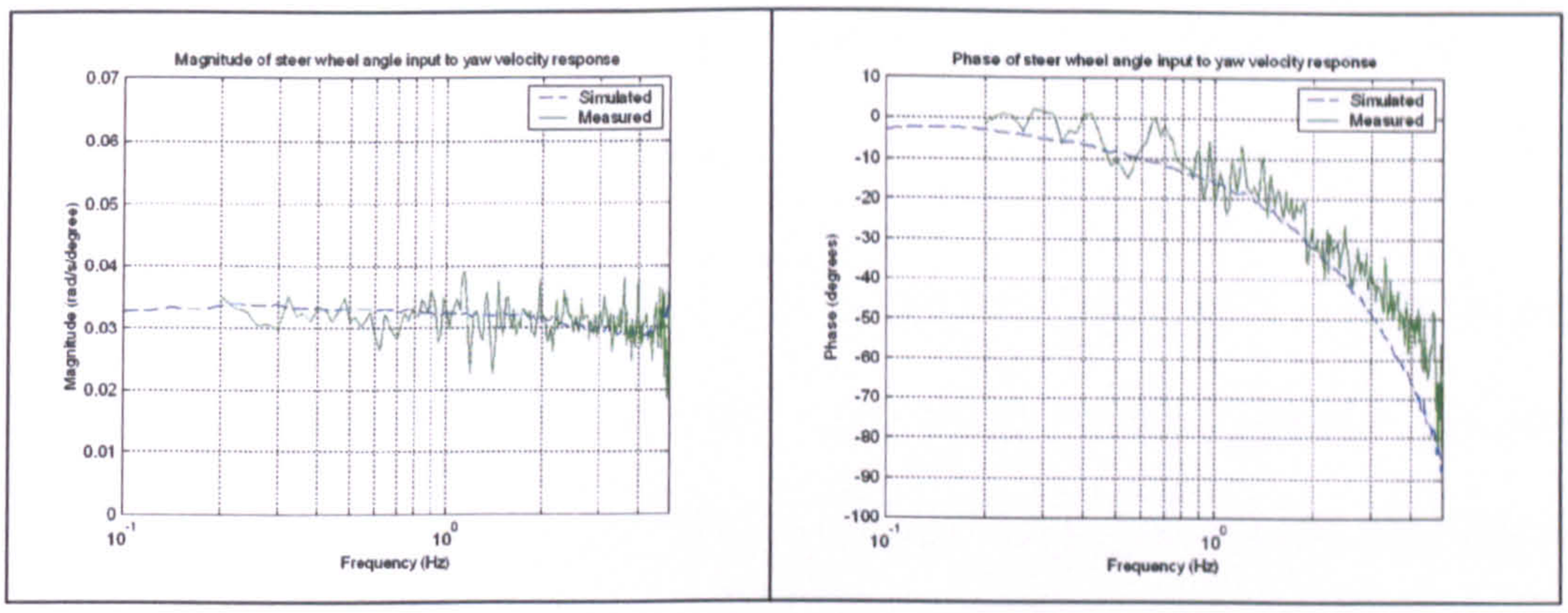


Figure 5.10 – Sophisticated vehicle model and measured data, frequency domain yaw velocity magnitude (left) and phase (right) responses.

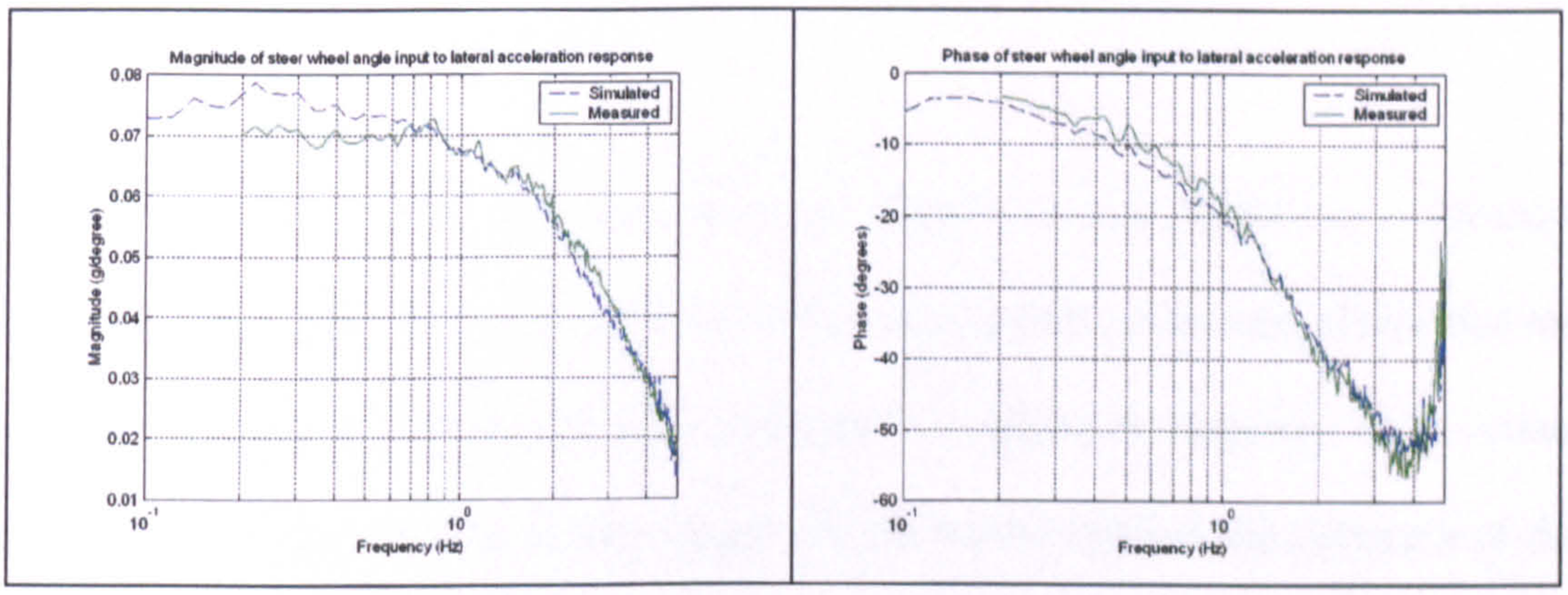


Figure 5.11 – Sophisticated vehicle model and measured data, frequency domain lateral acceleration magnitude (left) and phase (right) responses.

Figure 5.7 shows that the sophisticated vehicle model accurately predicts the steady state lateral acceleration response of the vehicle and an impressive level of agreement is in seen in figures 5.8 and 5.9 for the j-turn and lane-change manoeuvres. Furthermore, examining the sophisticated model frequency domain responses in figures 5.10 and 5.11, the yaw velocity and the lateral acceleration magnitude and phase responses have a high level of agreement. This is apart from the low frequency lateral acceleration magnitude response, which is again slightly over approximated by 7% and is probably due to the same reason given for the simple vehicle model. From these results, the sophisticated model can be seen to

accurately predict the vehicle's steady state and transient response for the full range of its lateral vehicle dynamic behaviour.

This result is not the same as the result found for the simple model where the lateral acceleration response above 1 Hz was seen not to closely simulate the actual vehicle's response. It can be seen from Heydinger et al [62] that the sophisticated vehicle model has a larger range of agreement with the measured data due to the inclusion of the roll degree of freedom and the tyre lag model. The sophisticated vehicle model may therefore be used with confidence in a transient simulation package.

Previous studies [9, 10 ,11] have used the simple vehicle model in a transient simulation package. It can be seen from the comparisons presented above that the simple vehicle model does not fully represent the transient response of an actual vehicle. Therefore the use of the simple vehicle model reduces the accuracy of the transient simulation package presented by Casanova et al.. If the more sophisticated vehicle model was used, then the approximations made of racing car performance using this transient simulation package would be more accurate.

5.3 Longitudinal Dynamic Behaviour

This section details a study between the Phase Two measured results and the sophisticated vehicle model simulated results. The study is used to find the accuracy of the vehicle model in predicting longitudinal dynamic behaviour. Only the sophisticated vehicle model's performance has been studied due to the more detailed powertrain and braking models used.

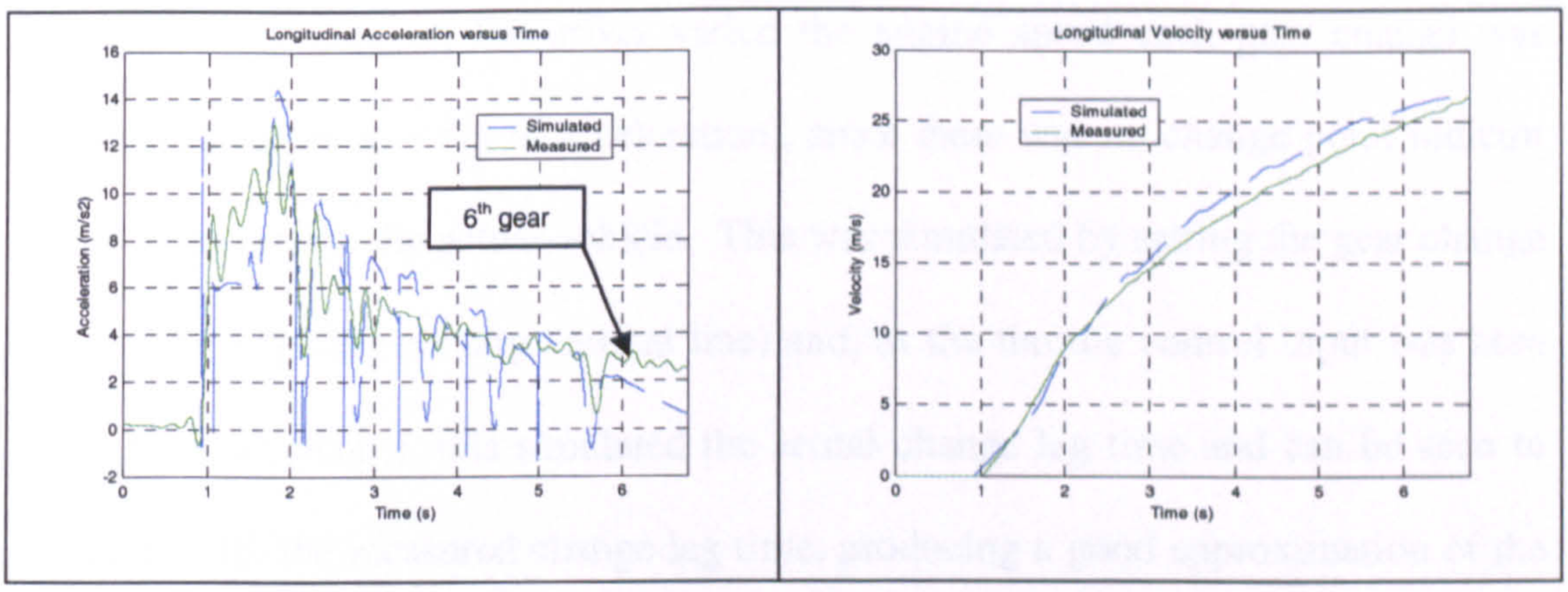


Figure 5.12 – Sophisticated vehicle model and measured data, standing start acceleration manoeuvre longitudinal acceleration (left) and longitudinal velocity (right) responses.

Figure 5.12 shows the comparison between measured and simulated standing start acceleration runs. It can be seen that there is a close resemblance in both the acceleration and speed plots. There are differences, especially around the 6th gear area, which was due to an engine fault on the actual vehicle causing it to be underpowered.

As there was only a limited time to take the measurements, the author was unable to solve the electrical problem with the vehicle's fuel system which altered the performance of the engine from the parameter map found previously on the engine

dynamometer. To account for this, the simulated engine output given by the parameter map was adjusted by applying a constant reduction factor but, this obviously does not fully account for the actual fault which caused a non-uniform decrease in engine performance. The reduction factor, however, was found to be constant for all the results taken, implying that a good estimation of measured vehicle performance was still being made by the vehicle model.

During the manoeuvre, the driver varied the engine speed each gear change was made at (denoted as a dip in acceleration), since there was no change point indictor (or tachometer) on the actual vehicle. This was simulated by setting the gear change time to zero (giving a sharp vertical line) and, as the throttle control input was zero during a gear change, this simulated the actual change lag time and can be seen to coincide with the measured change lag time, producing a good approximation of the actual vehicle's gear changes.

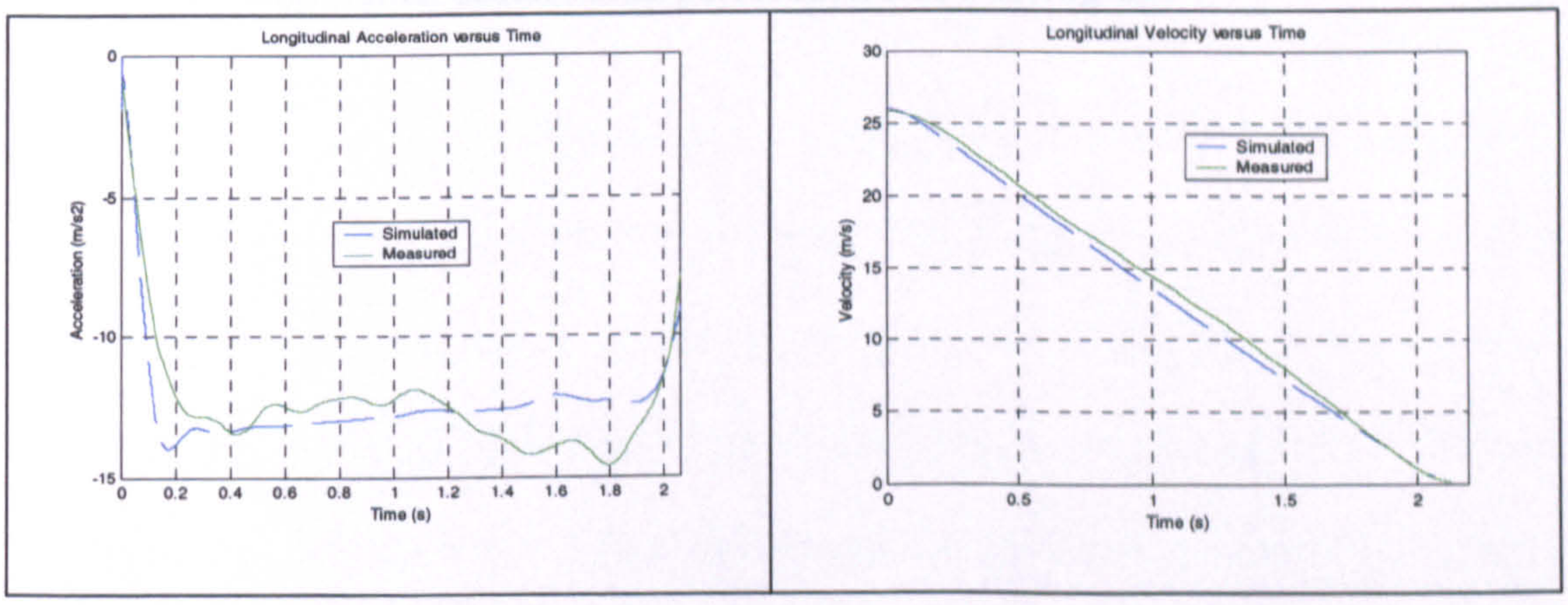


Figure 5.13 – Sophisticated vehicle model and measured data, braking manoeuvre longitudinal acceleration (left) and longitudinal velocity (right) responses.

Figure 5.13 shows a straight line braking manoeuvre to a stop and it can be seen that the measured and simulated responses closely match. A slight difference is seen and this was due to the brake line pressure sensors saturating below the maximum values

reached in the hydraulic circuits. To account for this, a constant increase factor was applied to the brake line pressure control inputs, this however did not quite model the non-uniform changes made by the driver whilst above the sensor saturation point. This factor was quantified using the original Excel spreadsheet used to design the vehicle's braking system (the spreadsheet has been verified to be accurate by the team). Thus, by assuming the hydraulic system saturated at the same pressure that the pressure sensors had, the author found that the same increase factor was needed to produce the same stop shown above.

When studying the use of the brakes in other manoeuvres (low deceleration straight line braking and hairpin manoeuvres, see figure 5.14), where the pressure sensors have not saturated and the increase factor was not needed, the vehicle model still made by an accurate prediction of the vehicle's longitudinal performance. From these results, therefore, it is seen that the sophisticated vehicle model can accurately predict the longitudinal acceleration performance of a racing car.

5.4 Combined Lateral and Longitudinal Dynamic Behaviour

This section details a comparison study between the Phase Two measured results and the sophisticated vehicle model simulated results. The study is used to find the accuracy of the vehicle model in predicting combined lateral and longitudinal performance.

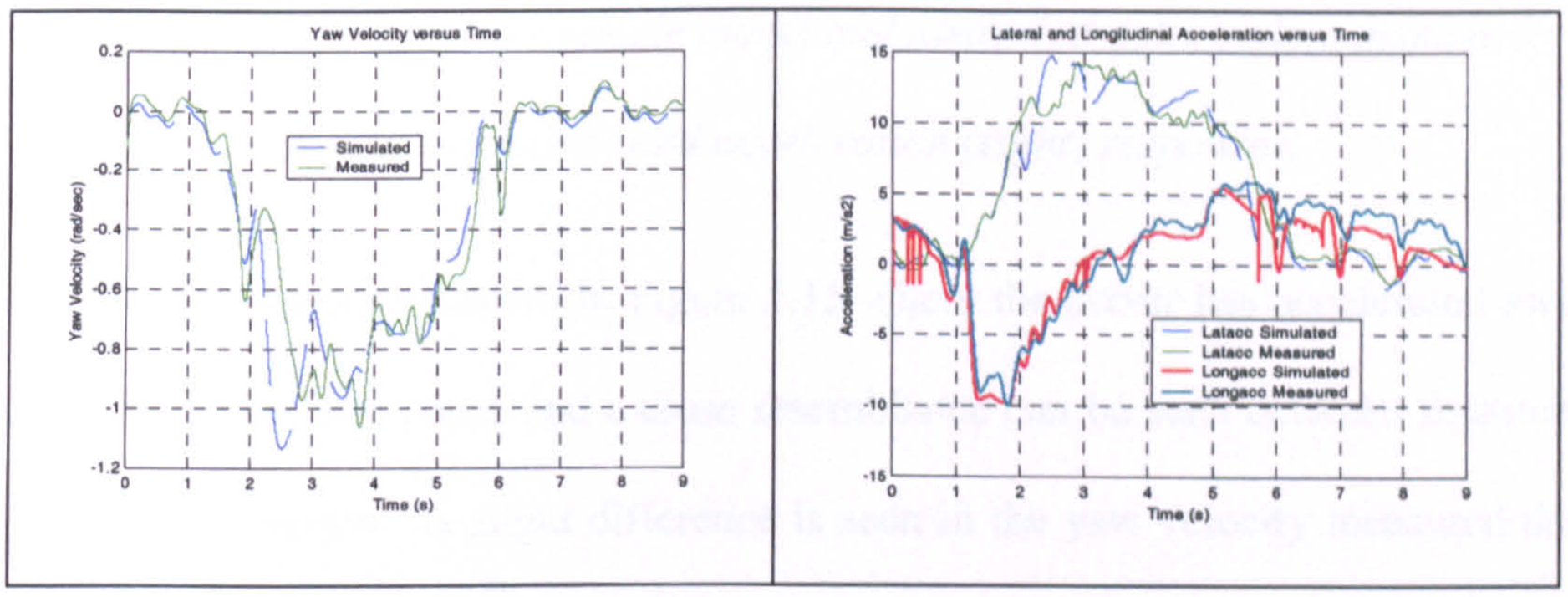


Figure 5.14 – Sophisticated vehicle model and measured data, hairpin manoeuvre yaw velocity (left) and acceleration (right) responses.

The hairpin manoeuvre is shown in Figure 5.14, for the control inputs displayed in Figure 4.4. A close resemblance can be seen between measured and simulated values but a difference again occurs is apparent for longitudinal acceleration whilst accelerating out of the corner (from six seconds onwards). This difference was caused by the electrical fault that reduced engine power during Phase Two and was discussed in sub-section 5.3. Below six seconds, the longitudinal acceleration match is good as the vehicle is traction limited and the driver is not applying the reduced maximum engine power.

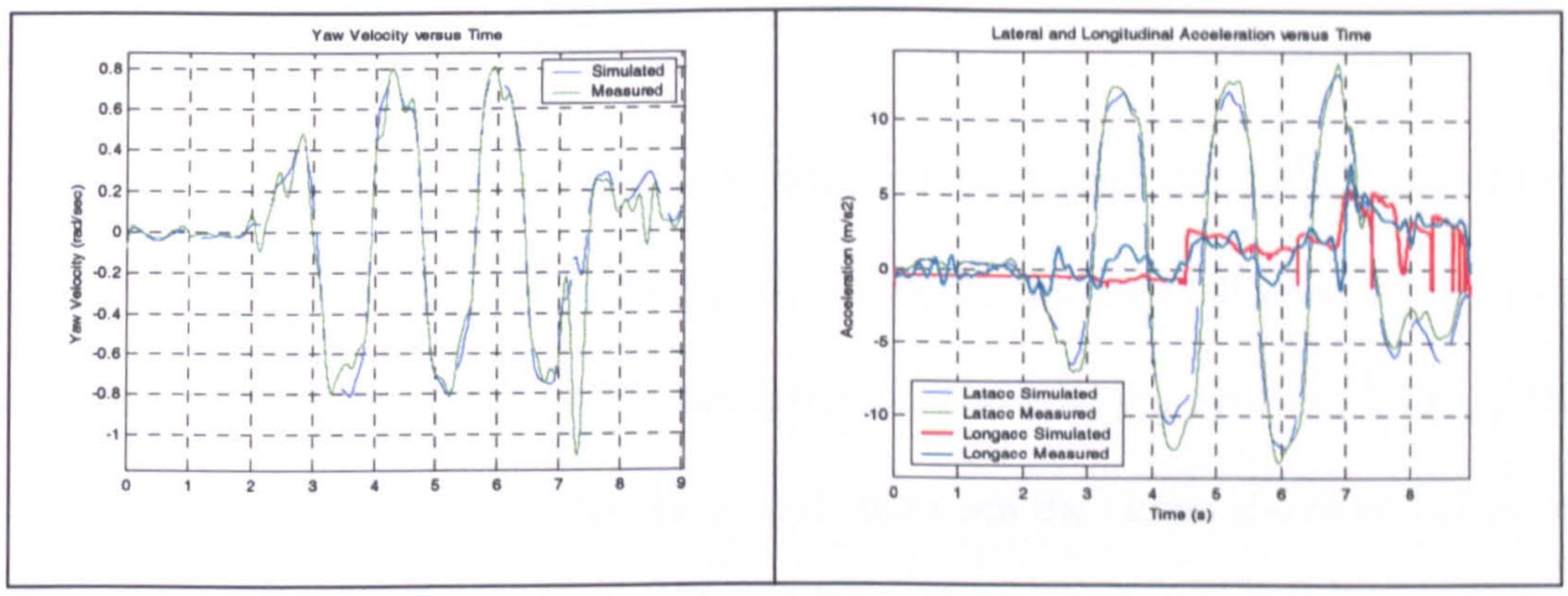


Figure 5.15 – Sophisticated vehicle model and measured data, slalom manoeuvre yaw velocity (left) and acceleration (right) responses.

A slalom manoeuvre is shown in Figure 5.15 where the driver has accelerated away through the final two cones and a close resemblance can be seen between measured and simulated values. A slight difference is seen in the yaw velocity measured data of figure 5.14 and 5.15, a short peak is seen at 6 and 7.25 seconds respectively, which is not seen in the simulated data and is due to the F5 vehicle wheel lift off problem discussed in Chapter Four. This is not reflected in the simulated data as the vehicle model does not account for the effect of suspension kinematics and so does not simulate this effect. In producing the vehicle model, it has been assumed that the suspension system does not have a significant effect on the vehicle's performance and keeps the wheels at fixed camber angles in all situations. This is normally a good approximation when studying vehicles such as racing cars which only undergo small amounts of suspension movement but, as seen in this study can allow potential problems to be overlooked.

From these results it is seen that the sophisticated vehicle model can be used with some confidence to predict combined lateral and longitudinal acceleration performance of a racing car.

5.5 Conclusions

The chapter has shown that the simple vehicle model does not fully represent the lateral dynamic behaviour of a racing car, whereas the sophisticated model more accurately represents it due to the inclusion of the roll and tyre lag DOF. This conclusion is achieved by a comparison study between the lateral dynamic behaviour of the simple and sophisticated vehicle models that were detailed in Chapter Three and the data collected in the testing Phase One. The sophisticated vehicle model's longitudinal and combined lateral and longitudinal dynamic responses are also validated using the data collected in the testing Phase Two. Comparisons are given between measured and simulated data for the following manoeuvres:

- Steady state circle manoeuvre.
- J-turn manoeuvre.
- Double lane-change manoeuvre.
- Random steer manoeuvre.
- Standing start acceleration manoeuvre.
- Braking to a stand still manoeuvre.
- Hairpin manoeuvre.
- Slalom manoeuvre.

It was deduced that the sophisticated vehicle model may be used with confidence in a LTS package using a transient approach, as it is accurate for the full range of lateral dynamic behaviour. The simple model, which is only accurate for low frequency and steady state responses, may be used in a LTS package using a quasi-static approach where the poor representation of the actual vehicle's transient response will not affect the solution results. Previous studies [9, 10, 11] have used the simple vehicle

model in a transient approach based LTS package. It can be seen, therefore, that the approximations that were made of racing car performance are not as accurate as when using the more sophisticated vehicle model.

Now that the sophisticated vehicle model has been validated, it can be used with confidence to optimise, not only 'tuneable' vehicle parameters such as the vehicle's sprung mass roll stiffness distribution, but also, fundamental design parameters such as the wheelbase, track, etc.. The range of comparisons shown in this chapter have contributed to the published body of knowledge on racing car performance prediction [55, 57].

6 Comparison of Simulation Approaches

6.1 Introduction

A comparison study has been conducted to evaluate the differences between the steady state, quasi-static and transient simulation approaches. In this study each approach has been used to simulate a vehicle negotiating two different simple manoeuvres in order to focus on the issues involved with each simulation approach.

Each manoeuvre consists of straight with a 180° corner at the end in (the simulations ends at the apex point, mid way through the corner). The first manoeuvre is at a constant forward velocity, the second is the vehicle braking down from 30 ms^{-1} until it is travelling slow enough to allow it to negotiate the corner and arrive at the apex at its maximum lateral acceleration. Both manoeuvres are split into two different sections, the first section is where the vehicle is travelling in a straight line and the second, whilst the vehicle is cornering. All three approaches use the sophisticated vehicle model (detailed in Chapter Three) with the University of Leeds F4 vehicle parameter set.

6.2 Steady State Simulation Approach

The most basic approach is a steady state simulation approach where the vehicle's longitudinal and lateral acceleration performance is modelled separately (i.e. the vehicle brakes and then turns in) and kept at constant values. Thus, only the lateral acceleration performance of the vehicle is taken into account during cornering and remains constant for the duration of the section.

The steady state solution of a simulation occurs when the system is in equilibrium and time dependent variables are zero [14]. This can be found by calculation, for example, by removing all the time dependent variables. It can also be found through simulation by applying a step steer input at a fixed forward velocity to a dynamic model and leaving the simulation to settle down to its steady state values.

The maximum forward velocity that the vehicle can negotiate the corner minimum path radius is found using a routine based on the Newton-Raphson iteration method [64]. The routine finds the steer angle which gives the vehicle's maximum steady state lateral acceleration for a given constant forward velocity. This simple optimisation routine is shown in equation (48), where x_{n+1} is the new solution value based on the previous value x_n. This method proved to be very efficient, finding the solution to 2 decimal places in only four iterations. The forward velocity is then increased and the iteration routine run again until no solution can be found for that path radius or the lateral acceleration limit drops. As the simulation is steady state, the vehicle is modelled negotiating the corner at this fixed forward velocity, steer angle, path radius and maximum lateral acceleration.

$$x_{n+1} = x_n + \frac{f(x_n)}{f'(x_n)} \qquad (48)$$

6.3 Quasi-Static Simulation Approach

Current LTS packages use the quasi-static simulation approach [36], where the corner is split into a series of constant radius turns [34] each with a decreasing path radius (simulating an increase of steer angle towards the corner apex). There are approximately fifty segments which ensures that the time step is small and the simulation makes a closer approximation to real life. The vehicle's acceleration

across each path segment is found in the same way as with the steady state approach by allowing the simulation to settle down to its steady state values.

The lateral tyre force needed to maintain this lateral acceleration and the remaining tyre force available is found using a friction circle approach [3]. The remaining tyre force is then used to derive the longitudinal acceleration of the vehicle. The minimum path radius and speed at the apex is found using the Newton-Raphson iteration routine described above.

6.4 Transient Simulation Approach

This approach is the closest approximation to what occurs in reality where the vehicle undergoes continuous non-steady linear or rotational accelerations [14]. In negotiating the corner in the first manoeuvre, a continuous control time history turns the dynamic vehicle model into the corner until it reaches its maximum steer angle and lateral acceleration at the corner apex. The maximum value at the apex is found using the Newton-Raphson iteration technique described above.

In the second manoeuvre, the vehicle is braking and cornering at the same time. Again a continuous control time history turns the dynamic vehicle model into the corner until it reaches its maximum steer angle and lateral acceleration at the corner apex but, at the same time, a diminishing brake force is used to move the vehicle around the edge of the tyre friction circles, reducing its longitudinal deceleration to zero close to the apex of the corner. Both the quasi-static and transient manoeuvres use the same input values for brake pedal force and steer angle, which are shown in figure 6.1.

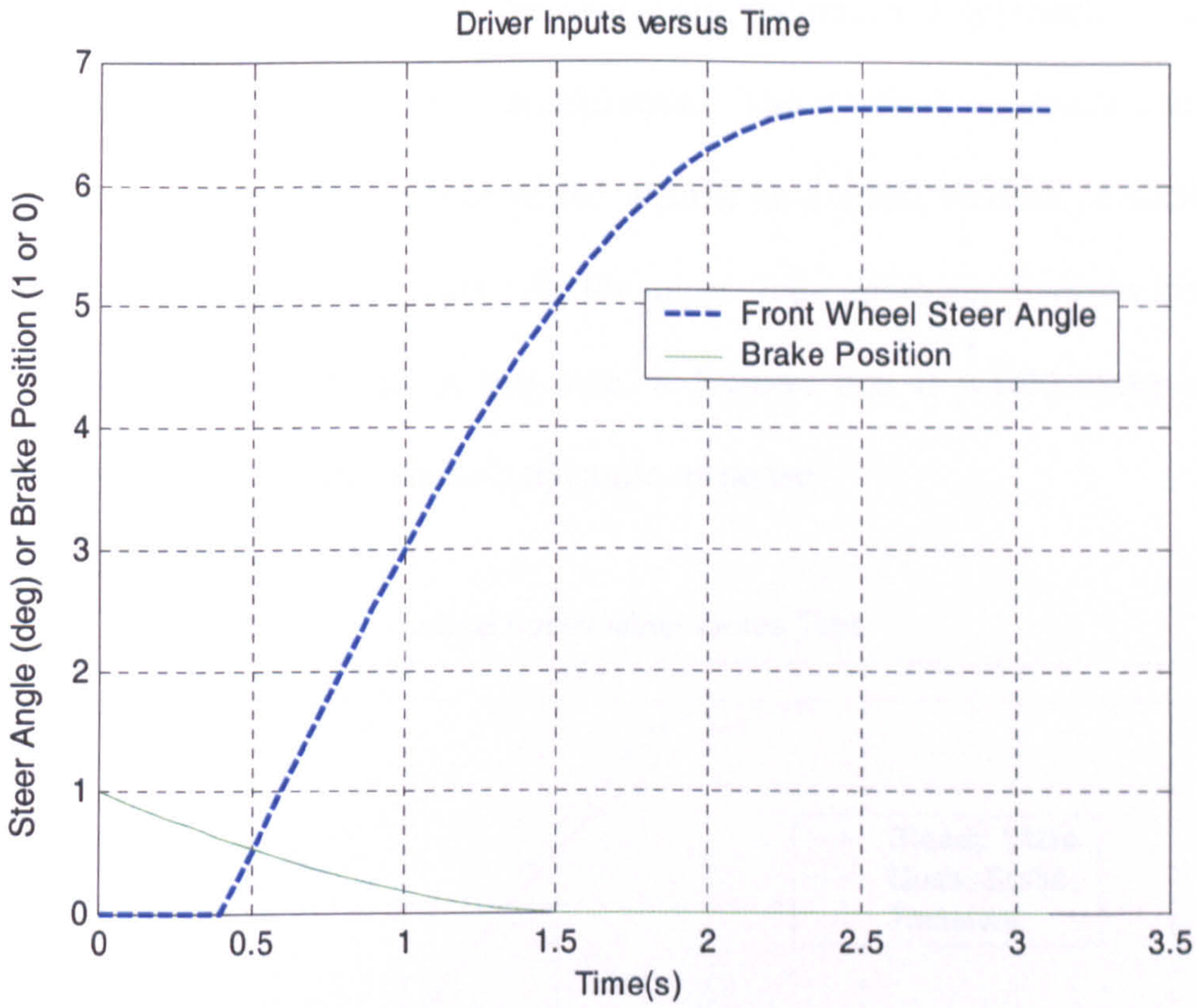


Figure 6.1 – Driver input values for quasi-static and transient simulation approaches in combined braking and cornering manoeuvre.

6.5 Results

The Newton-Raphson iteration routine found the maximum forward velocity at the corner apex to be 15 ms^{-1} with a steer angle of 6.6° and 13.4 ms^{-2} steady state lateral acceleration, which corresponded to a 16m minimum path radius.

Manoeuvre One (no braking) – Figure 6.2 shows a graph of lateral acceleration against time during the cornering section for each simulation approach. The steady state approach shows the vehicle being instantaneously at its peak constant lateral acceleration and minimum path radius. In contrast, the quasi-static and transient simulation gently builds up the lateral acceleration as the steer angle is increased to its optimum value, which is closer to what occurs in reality. It can be seen how the quasi-static approach makes discontinuous constant steady state approximations of

the vehicle's performance and diverges away from the transient approach's results as the vehicle reaches its peak lateral acceleration. The transient approach also takes into account the dynamic response of the vehicle as a small amount of oscillation about the steady state value occurs. As the quasi-static approach diverges from the transient approach's response in this area, it follows that it would make a poor approximation of the actual vehicle's dynamic response.

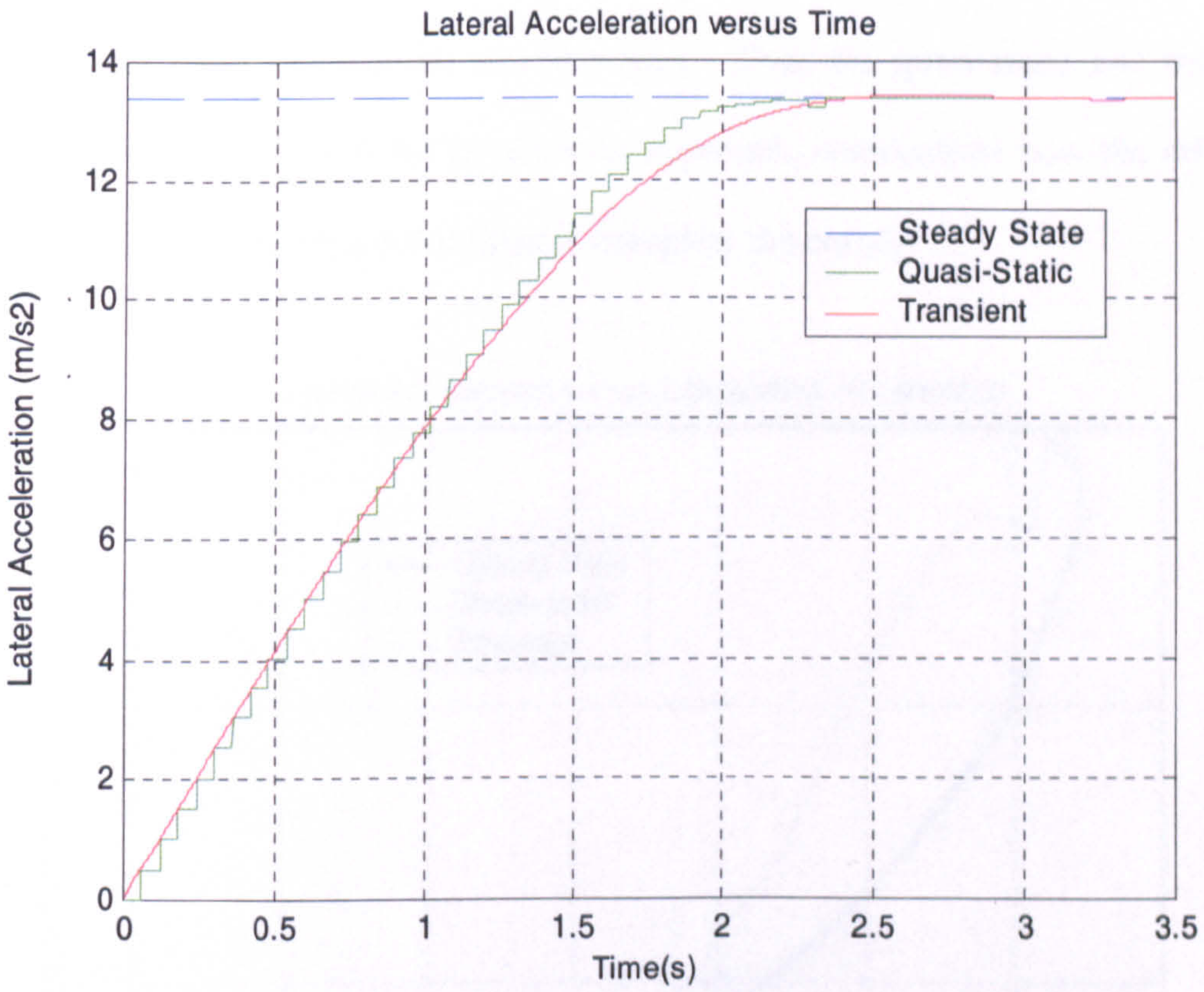


Figure 6.2 – Manoeuvre One: Graph of lateral acceleration versus time for all three simulation approaches.

Manoeuvre Two (Braking) – Figures 6.3 to 6.5 show graphs of the vehicle's lateral versus longitudinal acceleration, track position and forward velocity versus time in each simulation approach solution, during the cornering section. Again the steady state approach gives separate constant values for longitudinal and lateral acceleration as it is assumed to corner at constant speed. Due to the use of the same control inputs in each case, the quasi-static and transient approaches show similar results as the vehicle speed is decreased towards the corner apex and the lateral acceleration is built up, reaching a maximum near the apex. Thus the quasi-static and transient approaches, as opposed to the steady state approach, demonstrate how the driver in actuality uses combined accelerations to complete the corner.

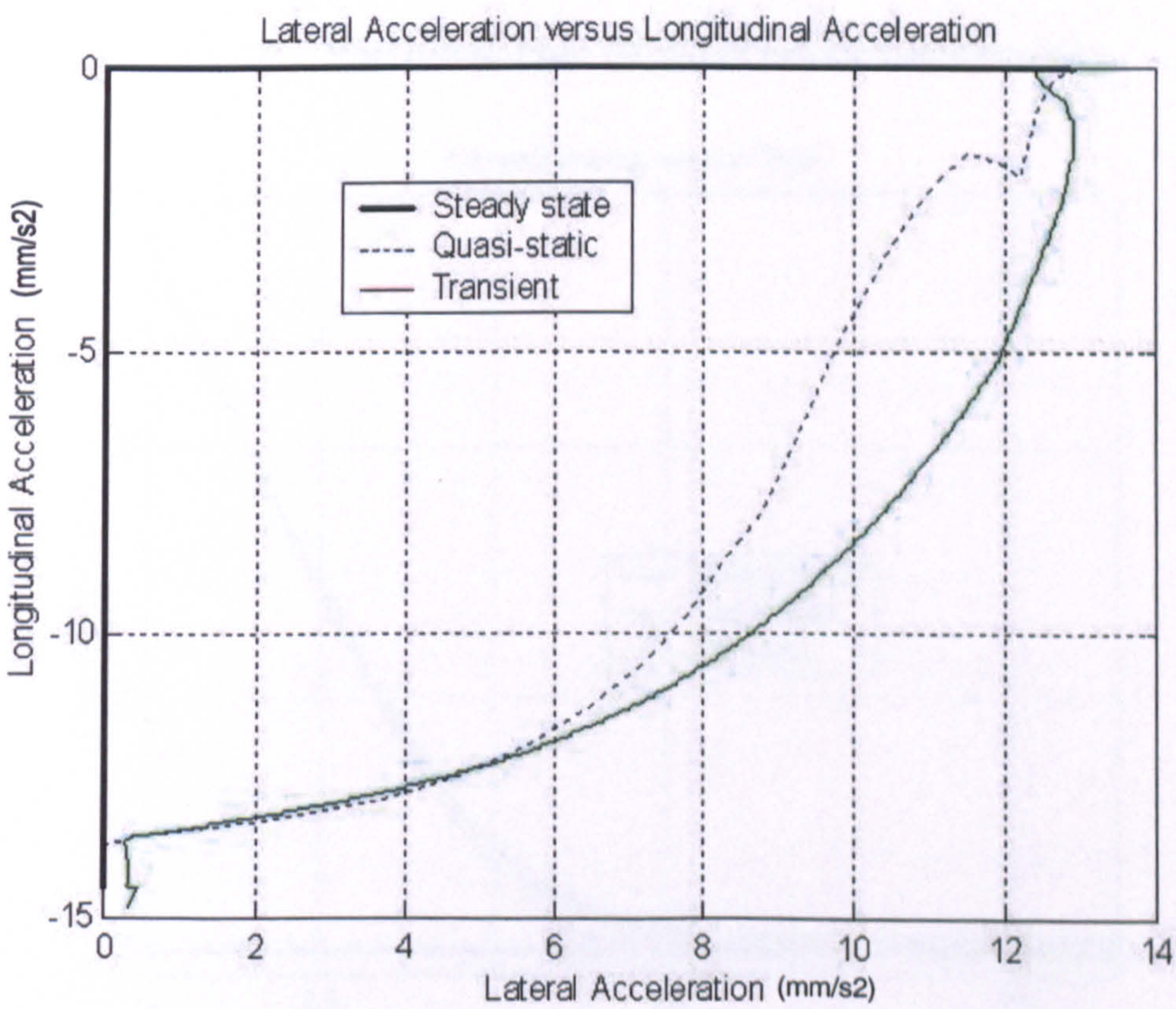


Figure 6.3 – Manoeuvre Two: Graph of longitudinal versus lateral acceleration.

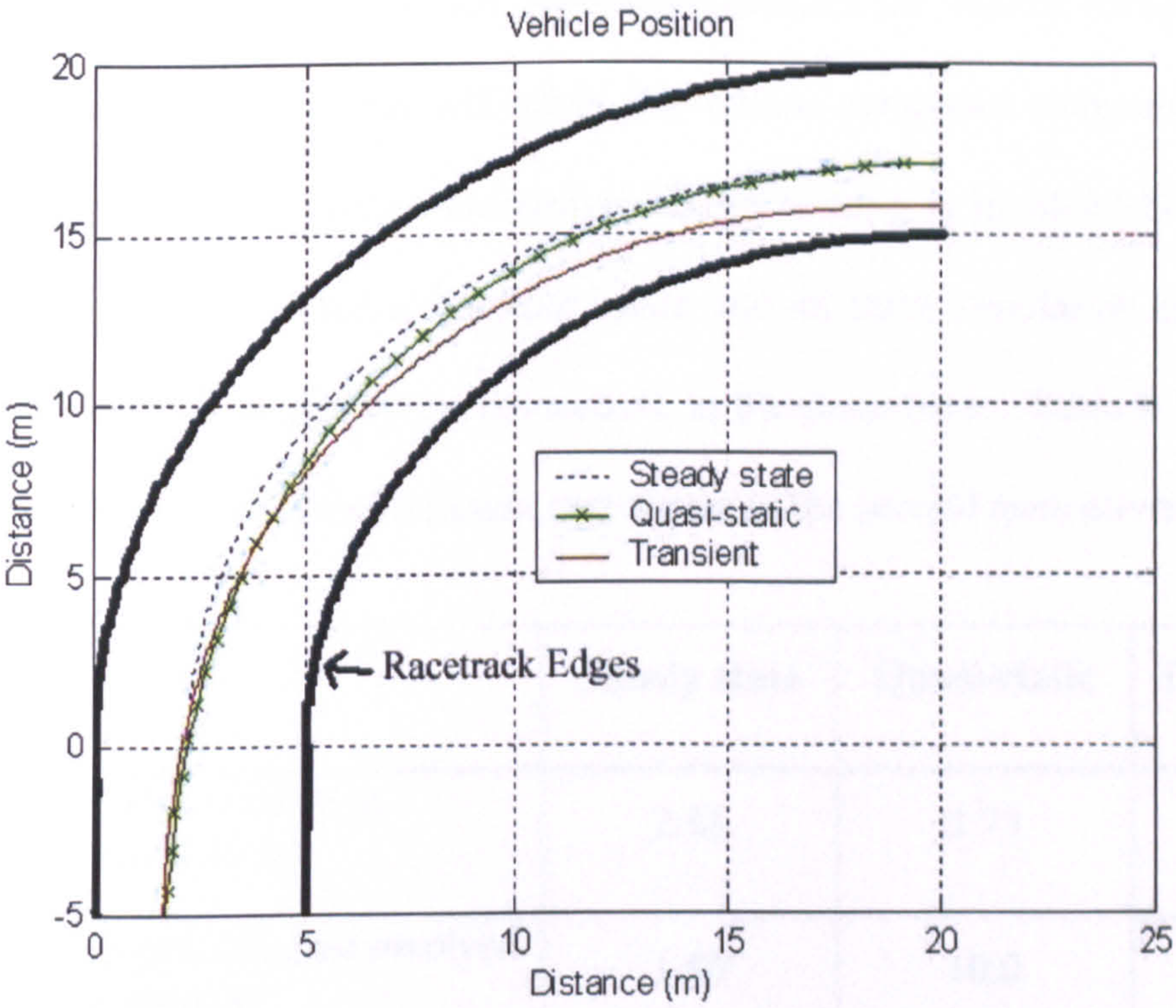


Figure 6.4 – Manoeuvre Two: Graph of vehicle track position.

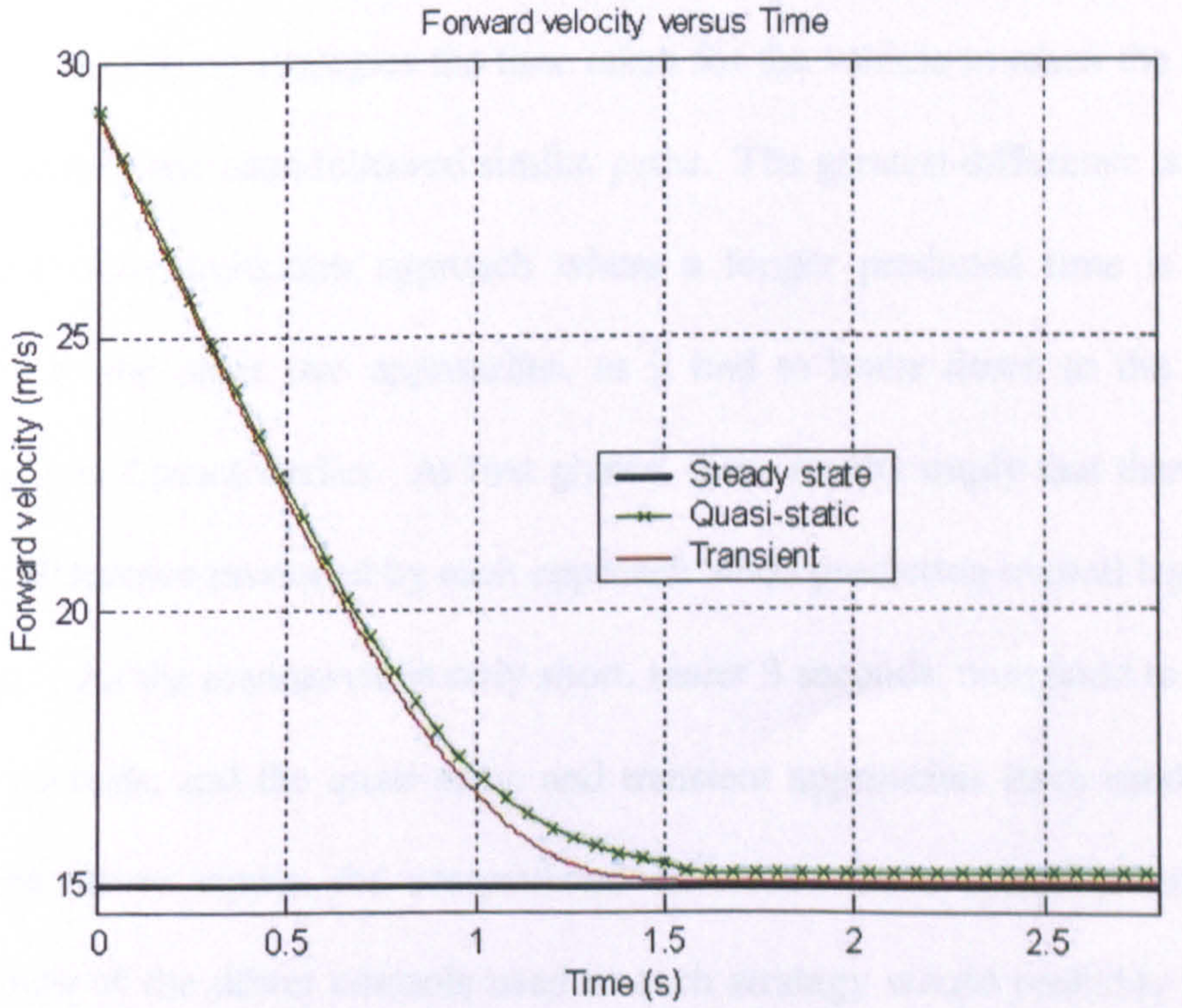


Figure 6.5 – Manoeuvre Two: Graph of forward velocity versus time.

The overall predicted time that each approach simulates the vehicle completing the manoeuvre is important, as this will affect any vehicle parameter comparison study. This only applies to the second manoeuvre where braking is involved because the first manoeuvre is simulated at constant speed and all three simulation approaches predict the vehicle completing the manoeuvre in the same time. Table 6.1 shows a comparison between the three different techniques in the second manoeuvre.

Approach	**Steady state**	**Quasi-static**	**Transient**
Total predicted time taken to complete manoeuvre (s)	2.85	2.73	2.71
Overall computational time involved in finding solution (s)	1.50	10.0	15.0

Table 6.1 – Manoeuvre Two: predicted and computation time for each approach.

In all three modelling strategies the time taken for the vehicle to reach the apex does not vary greatly and each followed similar paths. The greatest difference is seen with the steady state simulation approach where a longer predicted time is produced compared to the other two approaches, as it had to brake down to the minimum cornering speed much earlier. At first glance, these results imply that there may not be much difference produced by each approach when predicting overall lap times for a full lap. As the manoeuvre is only short, under 3 seconds, compared to a full lap, over 80 seconds, and the quasi-static and transient approaches have used the same predefined driver inputs, the accumulated difference for a complete lap and the optimisation of the driver controls used in each strategy would probably produce a larger difference between approaches. Further to this the first parameter sensitivity study seen in the next chapter, where parameter sets only produced small differences

between predicted lap times. Therefore, these relatively small predicted time differences between approaches are more significant than they may first appear.

The overall computational time involved with each approach varies greatly and can also be seen in table 6.1. The study has been conducted on a Pentium II 450 MHz processor with 128 Mb of RAM and it can be seen that the steady state simulation approach involves the shortest computation time, with the transient being the longest. The steady state approach, therefore, in producing shorter simulation times, would allow the race engineer to undertake a greater number of parameter investigations in a shorter overall computation time, but these results would not represent the vehicle as closely as those produced by other simulation approaches.

Nevertheless, it should be noted, that reducing the number of segments in the quasi-static approach, would decrease its computation time but may not greatly reduce the accuracy of the simulation (a large number of segments have been used in this case to allow it to be directly compared with the transient approach). Finally, no attempt has been made to optimise the driver controls (a predefined set of driver controls has been used) as this would play an important role in the length of time taken to compute the minimum time solution.

6.6 Conclusions

A comparison study has been conducted to evaluate the differences between the steady state, quasi-static and transient simulation approaches. Each approach has been used to simulate a vehicle negotiating a straight with a 180° corner at the end in two different manoeuvres. The first manoeuvre is at a constant forward velocity, the second is the vehicle braking down from 30 ms^{-1} until it is travelling slow enough to allow it to negotiate the corner and arrive at the apex at its maximum lateral acceleration.

The results demonstrated that when compared to the transient approach, the quasi-static and steady state approaches make a poor approximation of the vehicle's dynamic response. In the second manoeuvre, the small differences seen in predicted manoeuvre completion times produced by each approach would probably produce a cumulative difference which would be significant over a full lap.

It was also noted that the overall computation time between each approach varied greatly, with the transient approach taking the longest. The transient approach however, allows for more accurate tuning of a greater number of vehicle parameters as it also takes into account the dynamic response of the vehicle, which includes parameters that are not accounted for in the other simulation approaches.

7 Vehicle Parameter Sensitivity Studies

7.1 Introduction

Two different simulation packages have been developed by the author using the simulation approaches described in the literature review in Chapter Two. A quasi-static simulation approach based LTS package and a transient simulation approach based Manoeuvre Time Minimisation package.

In this chapter, two examples are used as case studies to illustrate how parameter sensitivity studies can be carried out using these simulation packages. The case studies also give further insights into the use of these types of simulation packages and the relevance of the simulation approaches used in them to actual racing car performance prediction.

The first case study, using the quasi-static simulation approach based LTS package, is the effect of a number of vehicle parameters on overall lap time. The second case study, using the transient simulation approach based Manoeuvre Time Minimisation package, is the effect of the front damping value on the time taken to complete a simple manoeuvre. Descriptions of the operation of the packages, including the graphical user interfaces used, are given and conclusions have been drawn on the relevance of the results to actual racing car performance prediction.

7.2 Quasi-Static Simulation

The LTS package uses the simple vehicle model described in Chapter Three and the fastest time it takes to negotiate the corner and straight sections of the track was found using two separate methods.

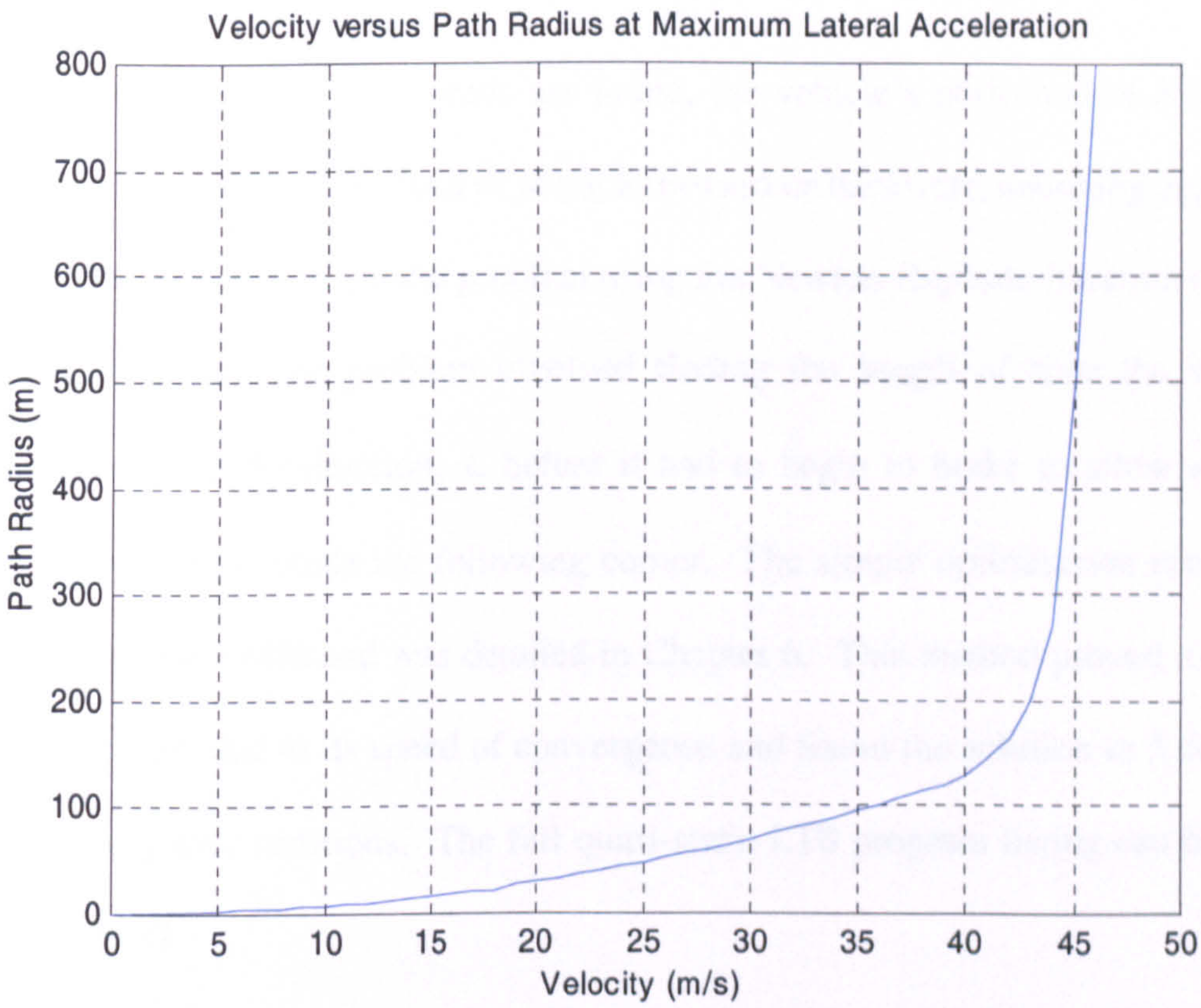


Figure 7.1 – Forward velocity against simulated path radius for maximum vehicle steady state lateral accelerations.

The vehicle's cornering performance was found from a performance map which is found before the main lap simulation, which reduced simulation computation times by stopping repeat simulations occurring [32]. Each corner section is treated as a steady state segment and the vehicle's performance was found by applying a step steer input to the dynamic vehicle model at a constant forward velocity, and waiting for the yaw acceleration to reduce to zero [65]. As the forward velocity is increased, the path radii followed at the maximum steady state lateral acceleration achievable is found. This corresponding path radius that the vehicle is negotiating then forms a

performance map against forward velocity, at maximum steady state lateral acceleration and is shown in figure 7.1 where it is used in the main lap simulation. Engine power limiting effects are also accounted for in the calculation, because at high speeds tyre drag greatly increases the minimum path radii the vehicle can negotiate.

Once the limiting cornering speeds are found, the vehicle's performance along the straights was calculated. Instead of using a forward or backward marching approach [31], it was decided to solve the problem using the Newton-Raphson iteration routine [64]. In this case, the problem involved finding the length of time the vehicle underwent engine acceleration, τ, before it had to begin to brake to allow it slow down enough to negotiate the following corner. The simple optimisation routine is shown in equation (48) and was detailed in Chapter 6. This method proved to again be very efficient due to its speed of convergence and found the solution to 2 decimal places in only four iterations. The full quasi-static LTS program listing can be seen in Appendix G.

The parameter sensitivity study involved finding the sensitivity to overall lap time of various vehicle parameters. This is achieved by varying fourteen vehicle parameters by -10% and +10% from their baseline values and ranking them in order of effectiveness in minimising overall lap time. The study is conducted using the parameter set found for the Leeds University F4 racing car and it is simulated completing the circuit shown displayed on the graphical user interface in figure 7.2. The circuit is the one the vehicle is competed on in the USA except that a slalom section on straight thirteen involving five cones has been removed for simplicity and the fact that the track map is created by scaling from a circuit diagram.

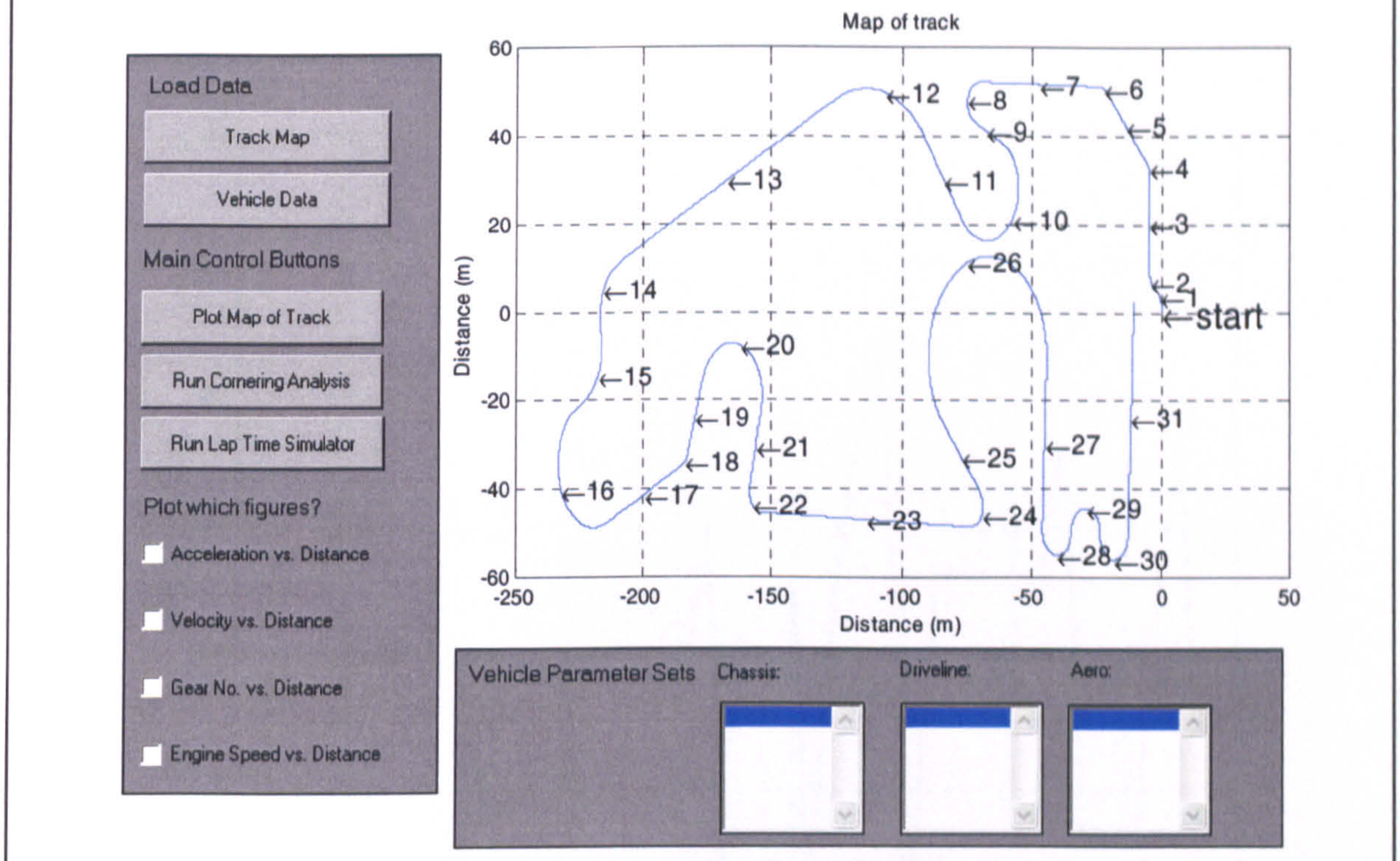


Figure 7.2 – LTS graphical user interface.

Figures 7.3 to 7.6 show the performance of the vehicle around the circuit for the baseline parameter set. The horizontal straight line areas of the plots below indicate the steady state cornering approximation used. The vehicle is simulated never undergoing combined lateral and longitudinal accelerations only purely lateral or longitudinally and this is shown in figure 7.4). As no data was recorded from the circuit, the trackmap has been created manually and the values found through measurement of a scale course map. Due to this, the corners are idealised as a constant path radius, rather than the 'real' racing line, where the path radius is changing as the vehicle moves towards and away from the apex.

The baseline simulation produced an overall lap time of 54.98 seconds, whereas the actual measured lap time was measured to be on average, around 60 seconds, which is 8% slower than those predicted in the simulation. The non-inclusion of the slalom

section and the steady state assumption idealising vehicle cornering performance are the main reasons for the difference.

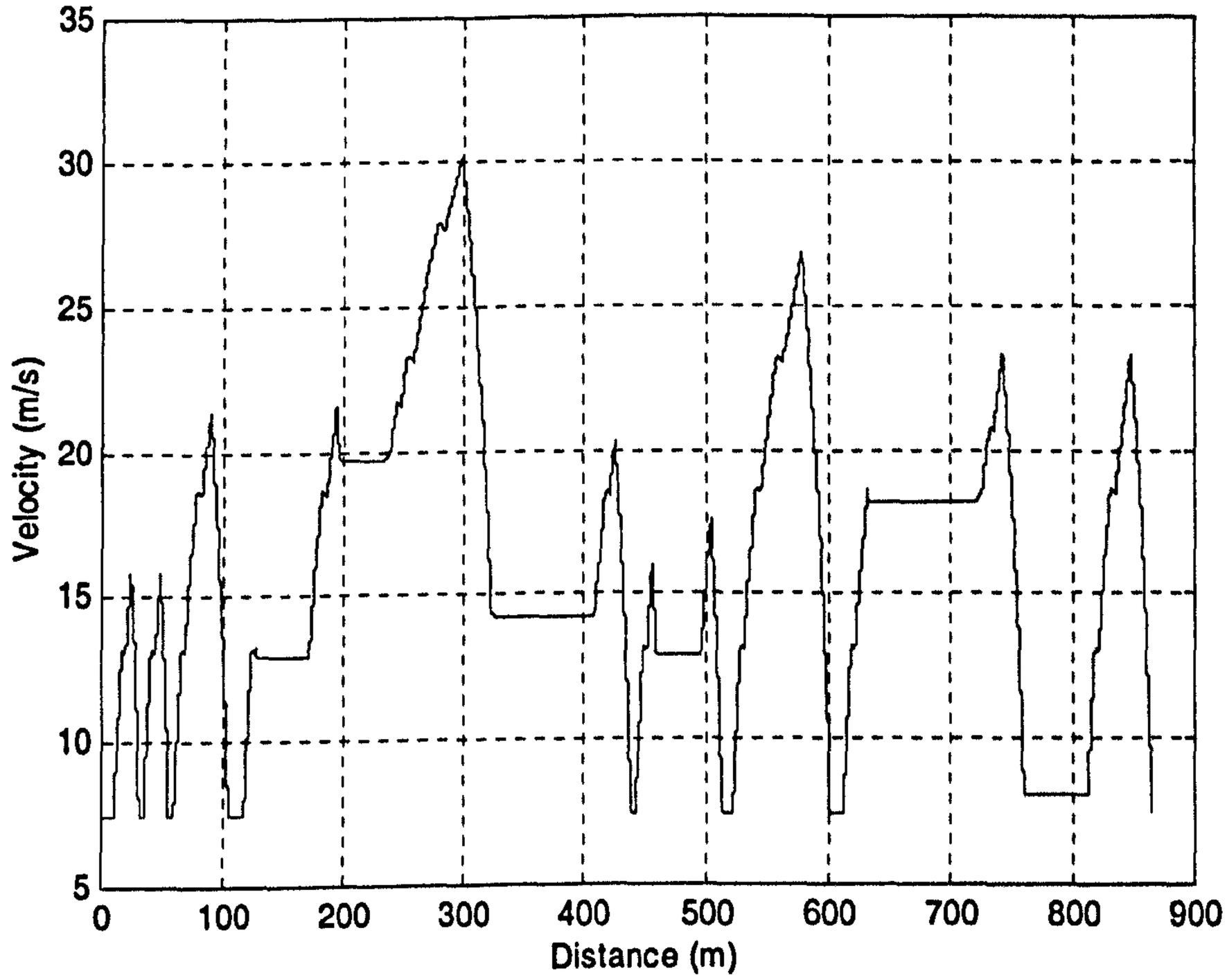


Figure 7.3 – Velocity versus distance.

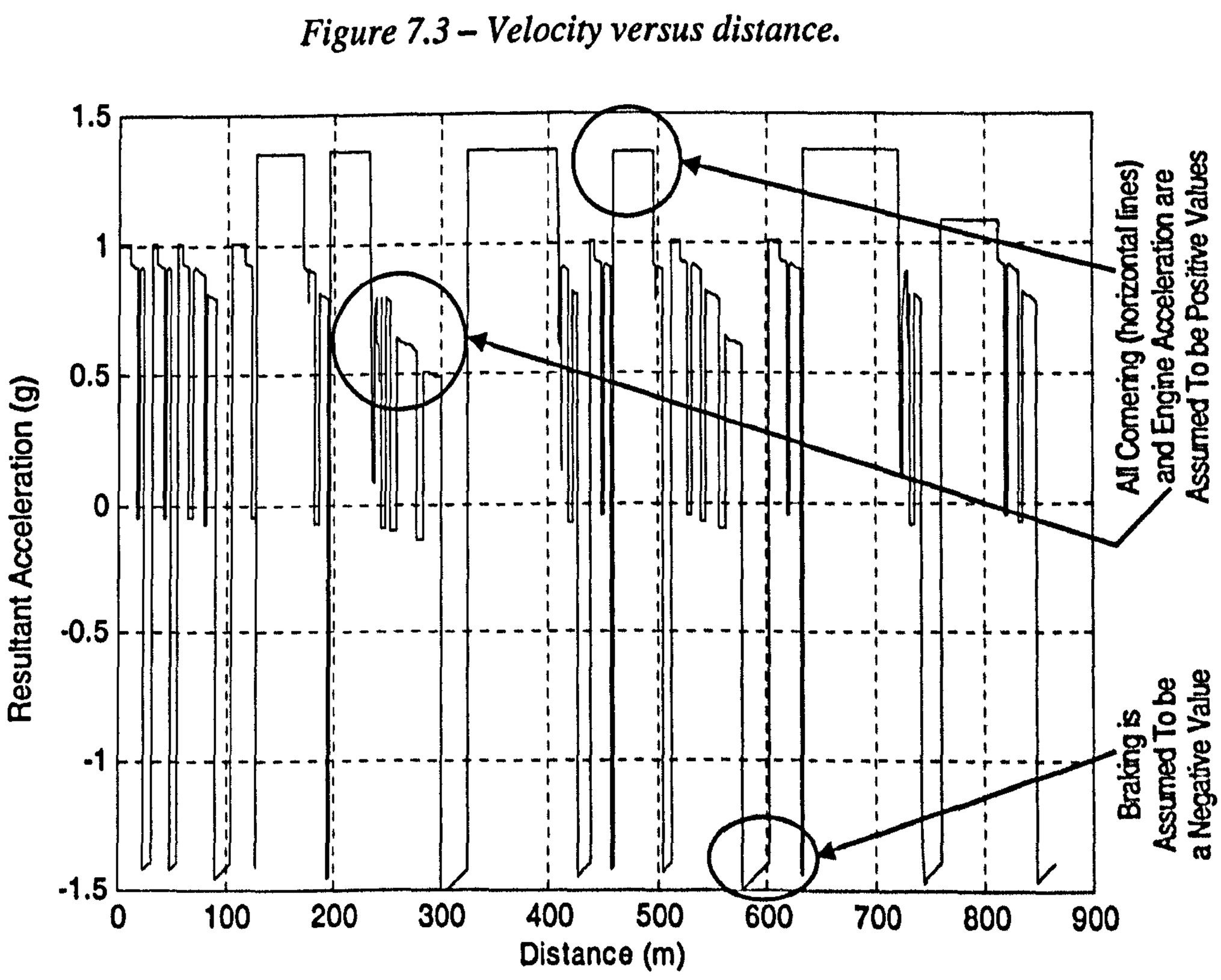


Figure 7.4 – Acceleration versus distance.

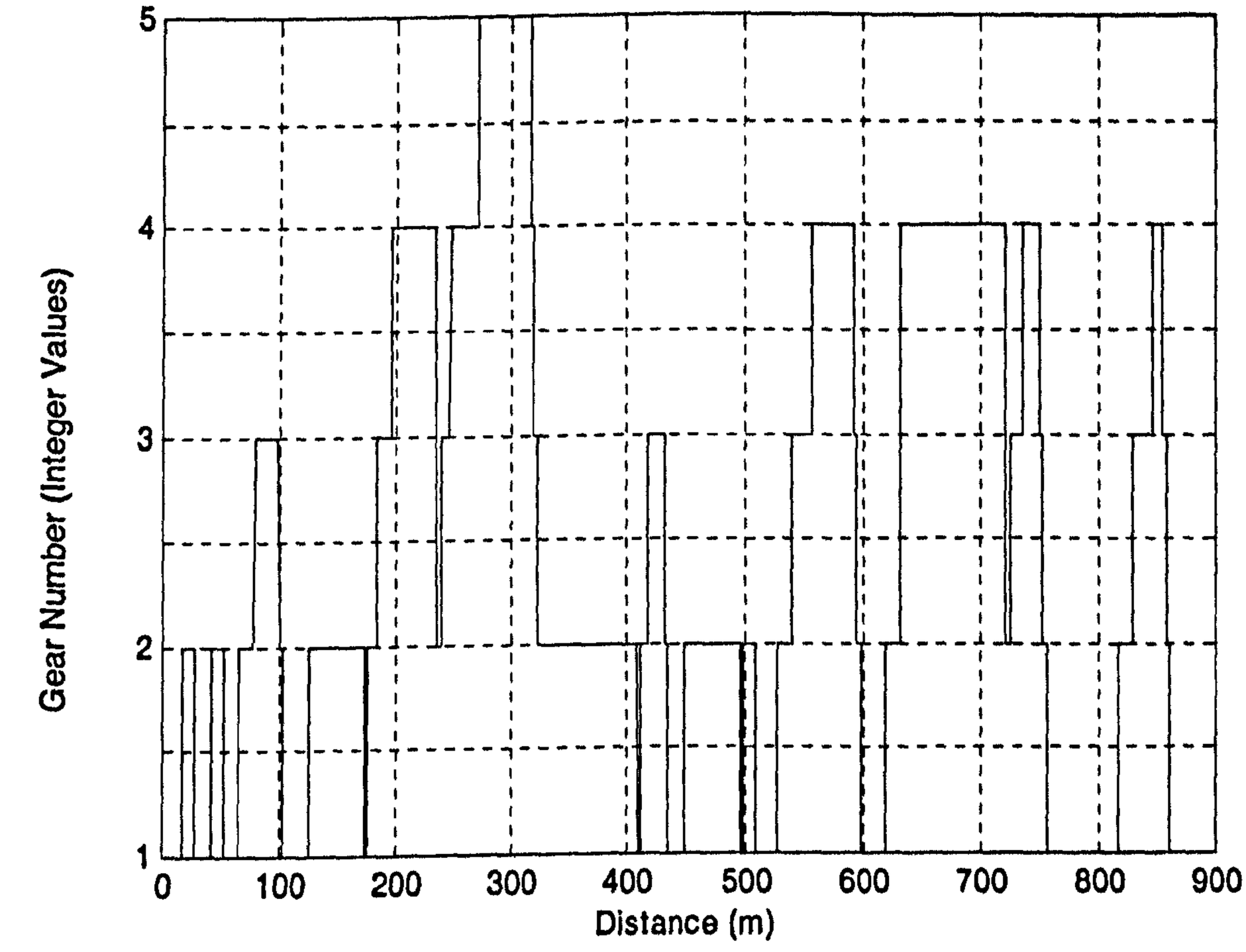


Figure 7.5 – Gear number versus distance.

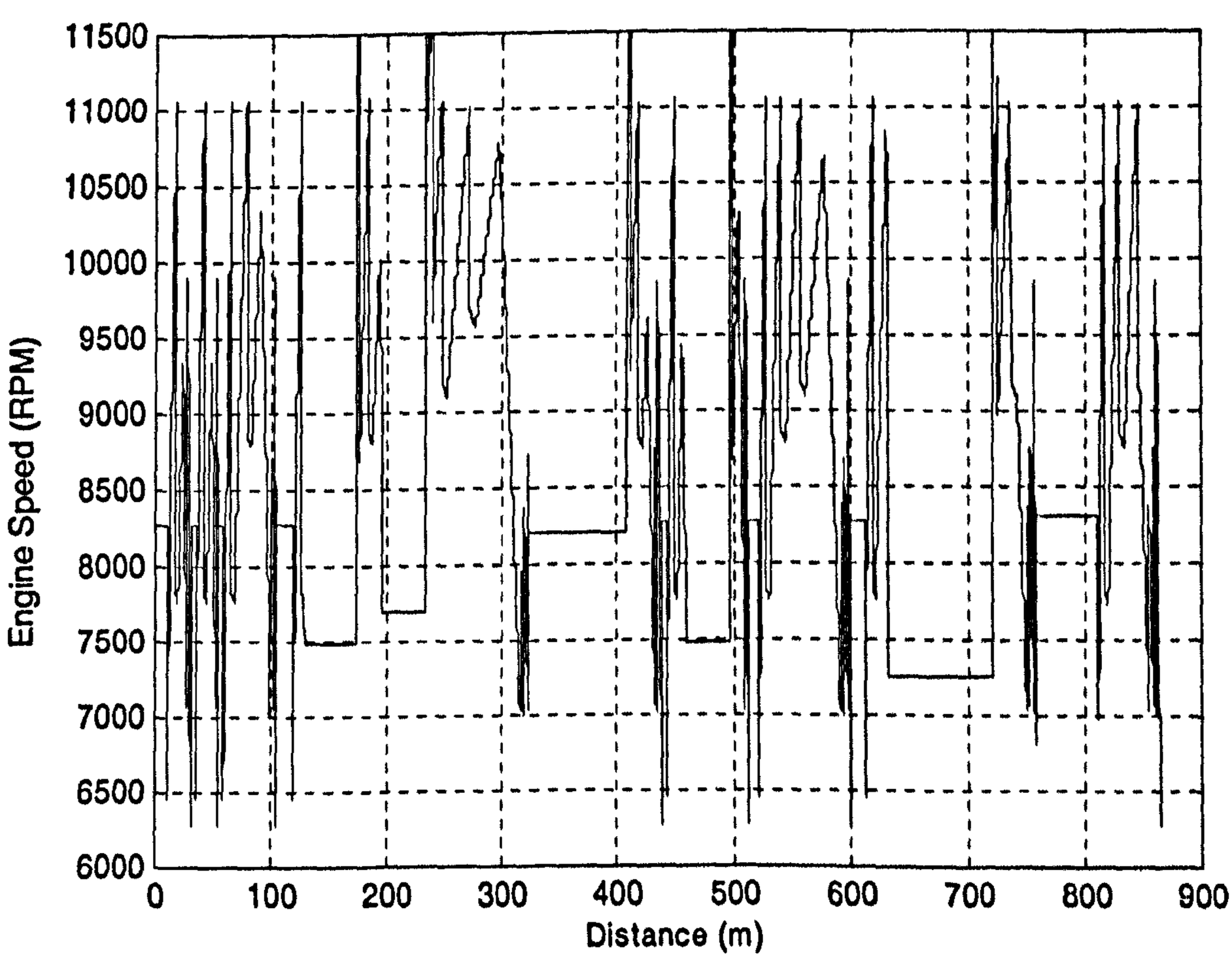


Figure 7.6 – Engine speed versus distance.

Table 7.1 shows the results of the parameter sensitivity study, ranked in order from 1 (the quickest) to 29 (the slowest), with the unchanged baseline values producing the 5th fastest time. As a total of 14 parameters are each varied twice by ± 10% and the ranking includes the baseline vehicle parameter set, there are 29 ranking positions. The parameters used in the study are the main ones an engineer would use to tune the vehicle's performance and their baseline values and units can be found in Appendix C. As well, a ± 10% change was found to be within the realistic design limits of the actual car for these parameters.

Parameter Varied	**Rank with 10% decrease**	**Rank with 10% increase**
Lateral tyre friction coefficient	29	1
Longitudinal tyre friction coefficient	9	2
Engine torque	14	3
Aerodynamic drag coefficient	4	10
Baseline vehicle parameter set	5	
Centre of gravity height	8	6
Front track	7	19
Mass	23	11
Wheelbase	12	28
Aerodynamic front axle lift coefficient	20	13
Aerodynamic rear axle lift coefficient	15	18
Front roll rate	26	16
Rear roll rate	17	24
Rear track	25	21
Centre of gravity distance from front axle	27	22

Table 7.1 – Overall lap time parameter sensitivity study results.

As expected, improvements were made by increasing the tyre friction coefficients or increasing torque produced by the engine or decreasing aerodynamic drag. It is seen that the lap time is most sensitive to the lateral tyre friction coefficient which produces not only the quickest but, slowest time as well. The overall time differences were small and this shows that small differences in lap time are important when conducting vehicle parameter investigations.

Any variation in the other parameters produces a reduction in lap time and so their values are deemed to be close to their optimum values. There are three reasons why no performance improvement is seen when the parameters are increased or decreased and these are listed below along with the corresponding parameters. For all of these parameters there is a compromise value where the performance of the vehicle is maximised and the baseline F4 vehicle parameter set is seen, from the results, to contain values close to these optimums as it has already been 'optimised' by the Leeds team by repetitive subjective testing.

1. Any variation in load transfer between left and right (lateral acceleration) or front and rear (longitudinal acceleration) wheels during acceleration may reduce the overall force produced by the tyres due to the tyre's load sensitivity [3]. The tyre's load sensitivity implies that for increasing normal force on a tyre a greater amount of longitudinal and/or lateral force is produced by the tyre, but this gain diminishes as normal force increases. Therefore there is an optimum value for front track, rear track, wheelbase and centre of gravity height, where the forces produced by the left and right or front and rear tyres combine to produce a maximum value.

2. The handling balance of the baseline vehicle parameter set was seen to be close to neutral in Chapter Four. Any change in front roll rate, rear roll rate, front aerodynamic lift coefficient, rear aerodynamic lift coefficient and centre of gravity distance from front axle would change this, making the vehicle more over or under steer balanced and reducing the maximum lateral acceleration possible. This is because both the front and rear axles will not be maximising their lateral force potential at the same time. Moreover, lateral load transfer distribution has an effect on the vehicle's handling balance so the parameters mentioned in the first reason above may also affect the vehicle's handling balance.
3. Reduction in overall mass will mean that there is less normal force on the tyres and so they produce less force, whereas increasing mass means there is more mass to be accelerated by the overall force that is produced. Therefore an optimum value exists where these two factors balance out.

7.3 Transient Simulation

To conduct the second parameter sensitivity study the author has used the transient simulation approach described in Chapter Two in conjunction with the sophisticated vehicle model described in Chapter Three, to create a Manoeuvre Time Minimisation package. The package finds the minimum time taken for a vehicle to complete a simple manoeuvre.

It has been demonstrated previously [9, 10, 11] that this package could easily be expanded to find the minimum time for the vehicle to complete a full lap of a circuit by adding several of these simple manoeuvres together. The package has not been extended in this case as the author has not been able to obtain a trackmap of an actual

circuit. Instead the author has concentrated on the accuracy of the vehicle model used in the simulation and minimising the speed of finding the solution.

As described in Chapter Two, the approach uses a constrained nonlinear optimisation routine, in this case, a sequential quadratic programming (SQP) method implemented through a Matlab sub-routine [48]. The routine minimises the manoeuvre completion time by adjusting the driver control matrix, within defined boundaries, whilst also having the constraint of keeping the vehicle inside the track boundaries. Full program listings are given in Appendix H.

The method is referred to as SQP because at each major iteration step, a quadratic programming (QP) sub-problem is solved. The SQP is implemented in three stages at each major iteration step:

- Update a Hessian matrix containing the Lagrangian function of the problem (used to solve QP sub problem).
- Converge on solution to general problem by finding the quadratic programming problem solution.
- Direction and distance away of new iterate solution found using line search and merit function calculations.

A number of tests have been undertaken by the author to check the robustness of the results given by the Manoeuvre Time Minimisation package and these are detailed below and the studies are only summarised here due to the simplicity of the results:

- To ensure that the package was able to optimise the vehicle's lateral dynamic behaviour, a steady state circle manoeuvre was used. The package successfully found the optimum steer angle, which maximised lateral acceleration and minimised the time taken to complete the circle.

- To ensure that the package was able to optimise the vehicle's longitudinal dynamic behaviour, a straight line acceleration manoeuvre was used. The package successfully found the optimum throttle position, which maximised longitudinal acceleration and minimised the time taken to complete the straight.
- From these two studies where the solution was intuitive, it was found that the solution given is the actual solution minimum, i.e. does not find a local minimum solution, but the global minimum.
- By re-running the same problem with the same input variables twice, it was found that the optimisation routine is stable as the same solution is reached each time.
- Observing the results, it was found that the final solution was always within the vehicle control limits and track boundaries.
- By varying the initial guess given for each manoeuvre it was found that a poor initial guess does not affect the final solution, only the time taken to reach it, as the same solution is reached each time.

There are a number of internal parameters used in the optimisation routine which affect not only the speed of solution but also the accuracy of the results. The most appropriate values for these internal parameters was found by running simulations with various parameter values and studying the time to find the solution and comparing results. An example is given in figure 7.7, where the best compromise value for control point distance has been found to be 4m. Control point distance (i.e. the distance between the driver control matrix points on the vehicle path) is critical in the simulation as it is important to find a value that would ensure reasonable solution times without missing too much of the detail of the continuous optimum vehicle

control history. Figure 7.7 shows that the 4m setting is able to achieve this, whilst also producing reasonable simulation times, as it has half the number of variables to be optimised compared to the 2m setting. In addition, 6m was tried but this was seen to miss too much detail in the control history. Control point distance is obviously speed dependent and during higher speed manoeuvres a larger value would be used.

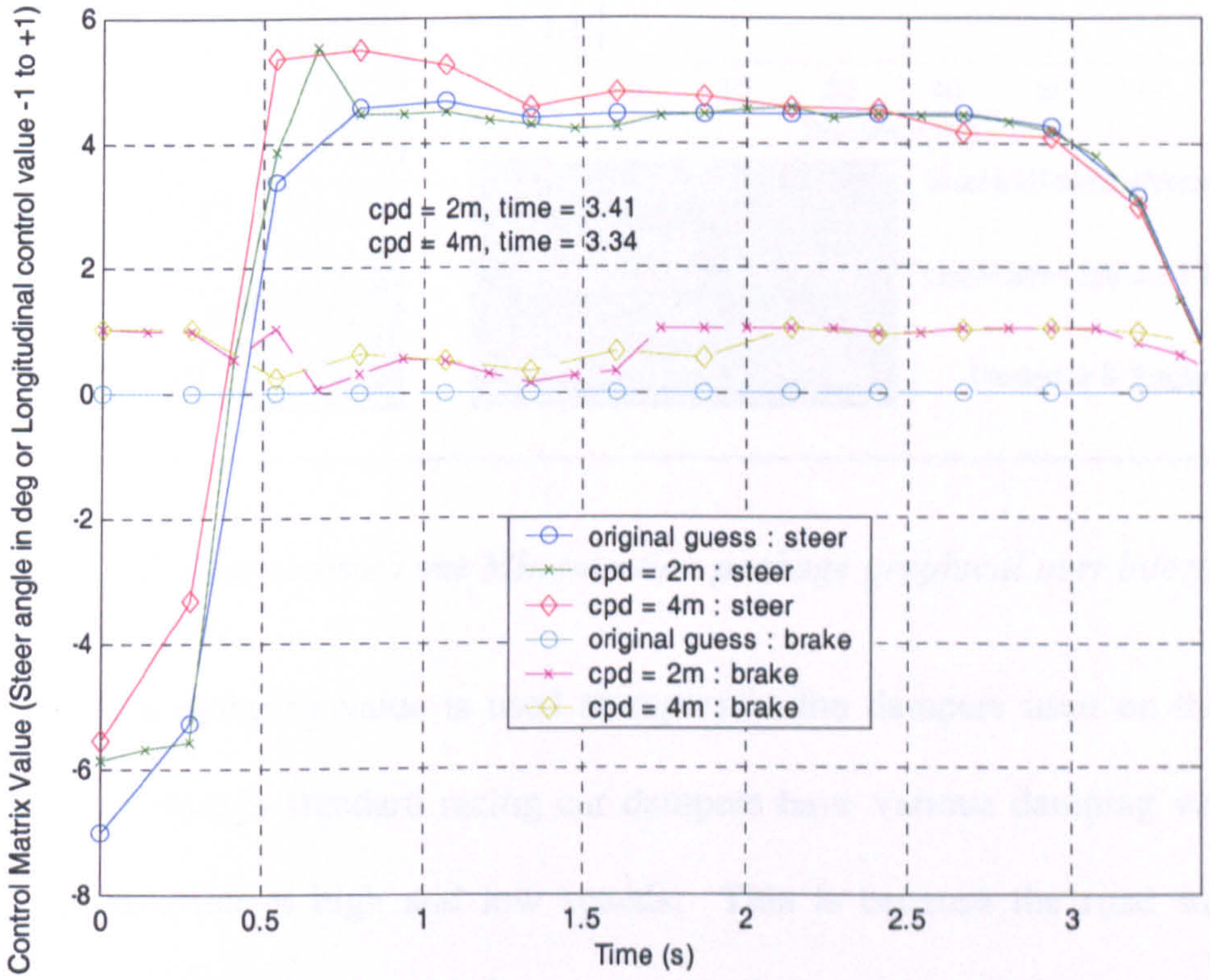


Figure 7.7 – Comparison of control point densities.

A parameter sensitivity study of the effect the vehicle's front damping value has on the time to complete a simple manoeuvre has been conducted. The simple manoeuvre involved the Leeds University F4 car completing the right hand corner shown on the Manoeuvre Time Minimisation package's graphical user interface in figure 7.8. The corner has a 17.5m path radius along its centre line and 5m track width with a control point specified every 4m along the vehicle path. The package took 4 hours to find each parameter set solution on a Pentium PII 400 MHz computer with 128 Mb of RAM and running Windows NT operating system.

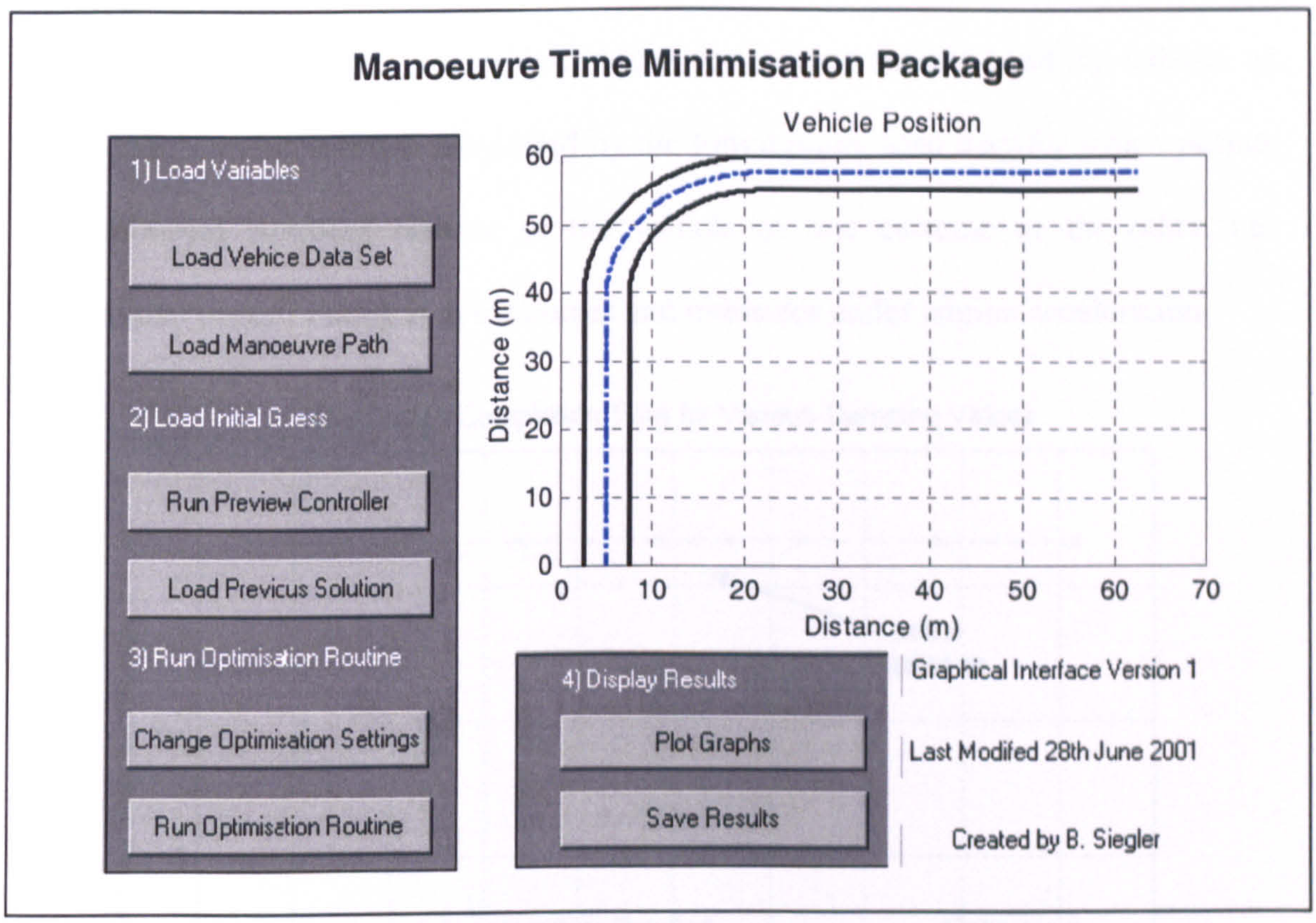


Figure 7.8 –Manoeuvre Time Minimisation package graphical user interface.

A fixed linear damping value is used to represent the dampers used on the Leeds vehicle, even though standard racing car dampers have various damping values for bump and rebound at high and low speeds. This is because the road surface is smooth and so only low speed damping will play a part and the mountain bike dampers peculiar to the Leeds vehicle have similar rebound and bump characteristics, unlike the more commonly used racing car dampers.

As the road surface has been assumed to be perfectly smooth in this example, the front damping value affects the rate of change of the lateral and longitudinal load transfer and will alter the transient handling balance of the vehicle as it enters or exits the corner [3]. As sprung mass roll and pitch velocities will be occurring simultaneously, the effect of the damper is to change the load (and load distribution) on the tyres during the transient corner entry and exit phases. This changes the

forces produced by the tyres, which in turn varies the transient handling balance of the vehicle. This effect is magnified by the longitudinal load transfer which pushes the transient handling balance of the vehicle to one extreme or the other. i.e. understeer under braking into the corner and oversteer under engine acceleration.

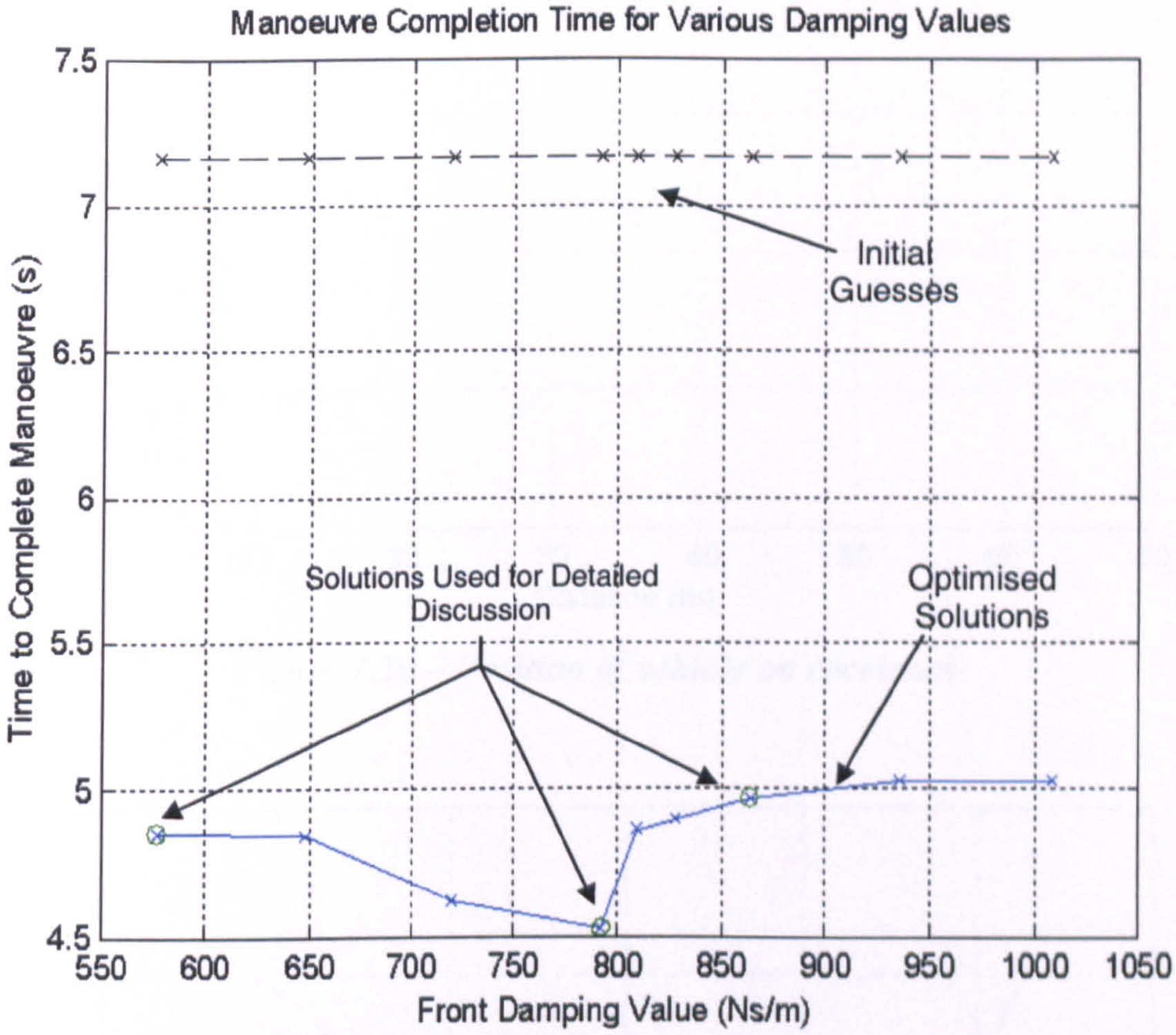


Figure 7.9 – Results of front damping sensitivity study.

The initial guess had a solution of 7.17 seconds at a constant forward velocity of 15 ms^{-1}. In each case the package has significantly reduced the time taken to complete the manoeuvre. To make an examination of the results simpler, three of the solutions have been chosen to be discussed in detail and are circled on figure 7.9 (the damping values correspond to values measured on a dyno-mometer). The middle solution is the minimum solution, whilst the other two are either side, this gives a better idea of the trends that are occurring. Figures 7.10 and 7.11 shows how each of these solutions remain within the track boundary, whilst taking the racing line.

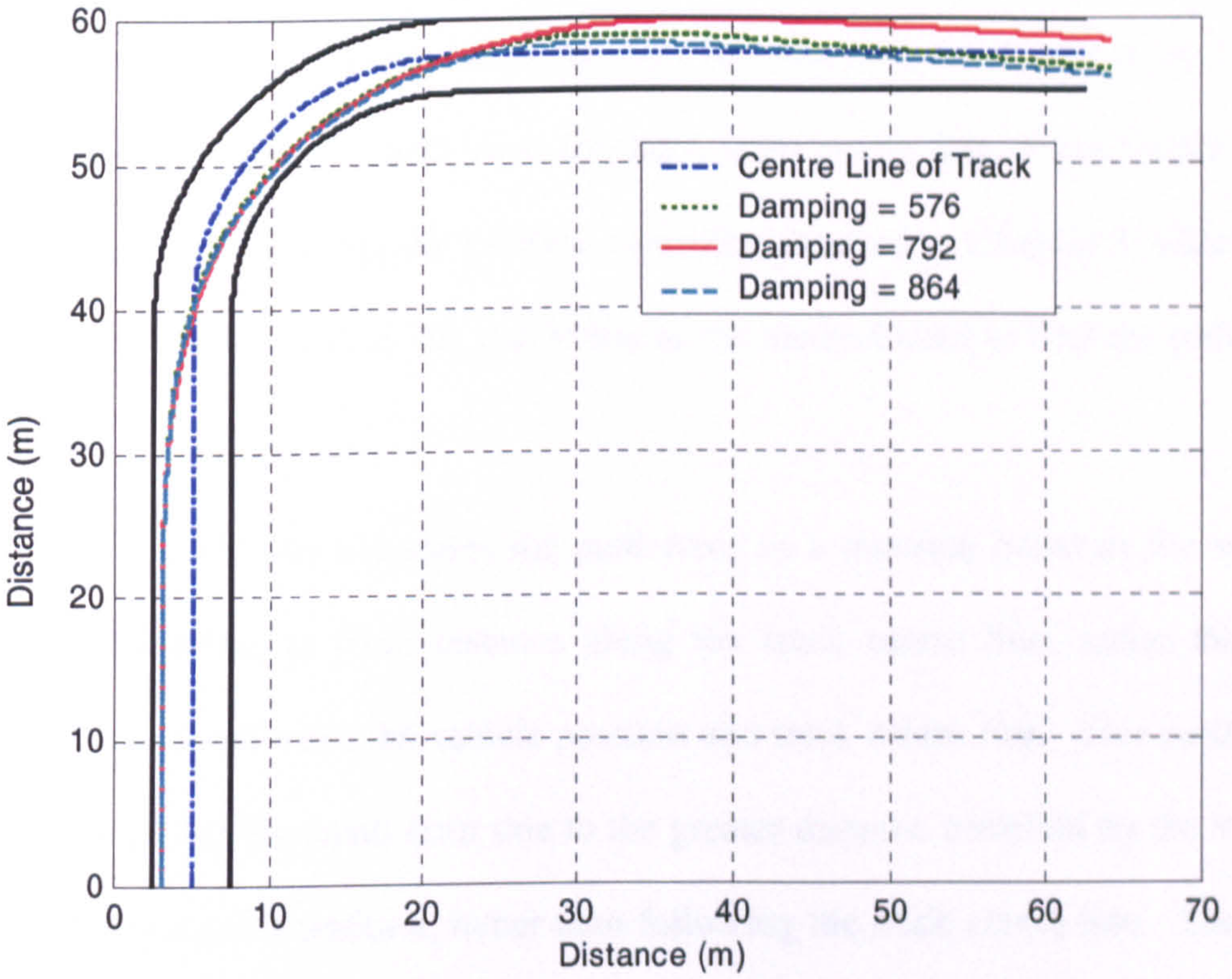


Figure 7.10 – Position of vehicle on racetrack.

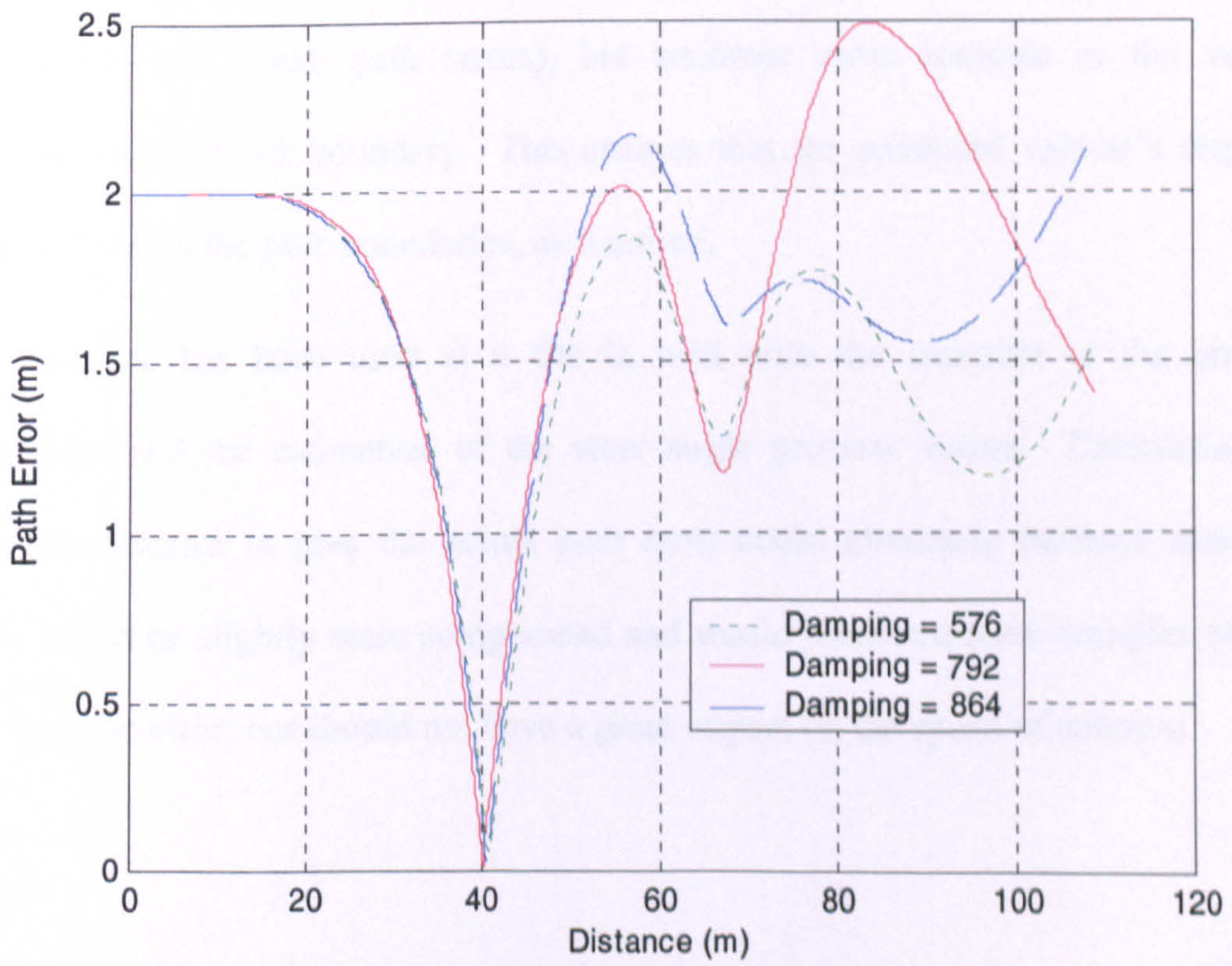


Figure 7.11 – Distance of vehicle from track centre line.

If a closer comparison is made between Figure 7.10 and 7.11, it can be seen that although the solutions remain within the path boundary, there appears to be a small error in the calculation of path error (distance from centre line of the track). This distance is calculated using the preview controller detailed in Chapter 3, which was produced by Casanova et al. [9] and is due to the method used to find the path error distance.

The preview controller calculates the path error as a distance between the vehicle position and points at fixed distance along the track centre line, rather than the shortest distance between the vehicle position and track centre line. This method of calculation causes the small error due to the greater distance travelled by the vehicle in reaching that track position, rather than following the track centre line. The error begins to arise as the vehicle yaws and as long as the yaw angle remains relatively small. It means that the predicted path error is insensitive and not accurate close to the centre line (small path errors), but becomes more accurate as the vehicle approaches the track boundary. This ensures that the predicted vehicle's response remains within the path boundaries, as required.

The method has been used at it fits in well with the structure of the preview controller and the estimation of the steer angle preview values. Calculating the shortest distance to give the actual path error could obviously increase accuracy. This would be slightly more complicated and would involve a more complex routine to find path error, but should not have a great impact on the speed of solution.

Figure 7.12 and 7.13 show how, for each solution, the package optimises the driver control matrix using combined lateral and longitudinal acceleration to maximise the vehicle's performance, moving the vehicle around the edge of its performance envelope. This mimics the technique demonstrated by an actual driver as earlier seen in Section 4.5.2. This method of balancing lateral and longitudinal acceleration as the vehicle approaches the apex has been accepted as the fastest method of negotiating a corner [3].

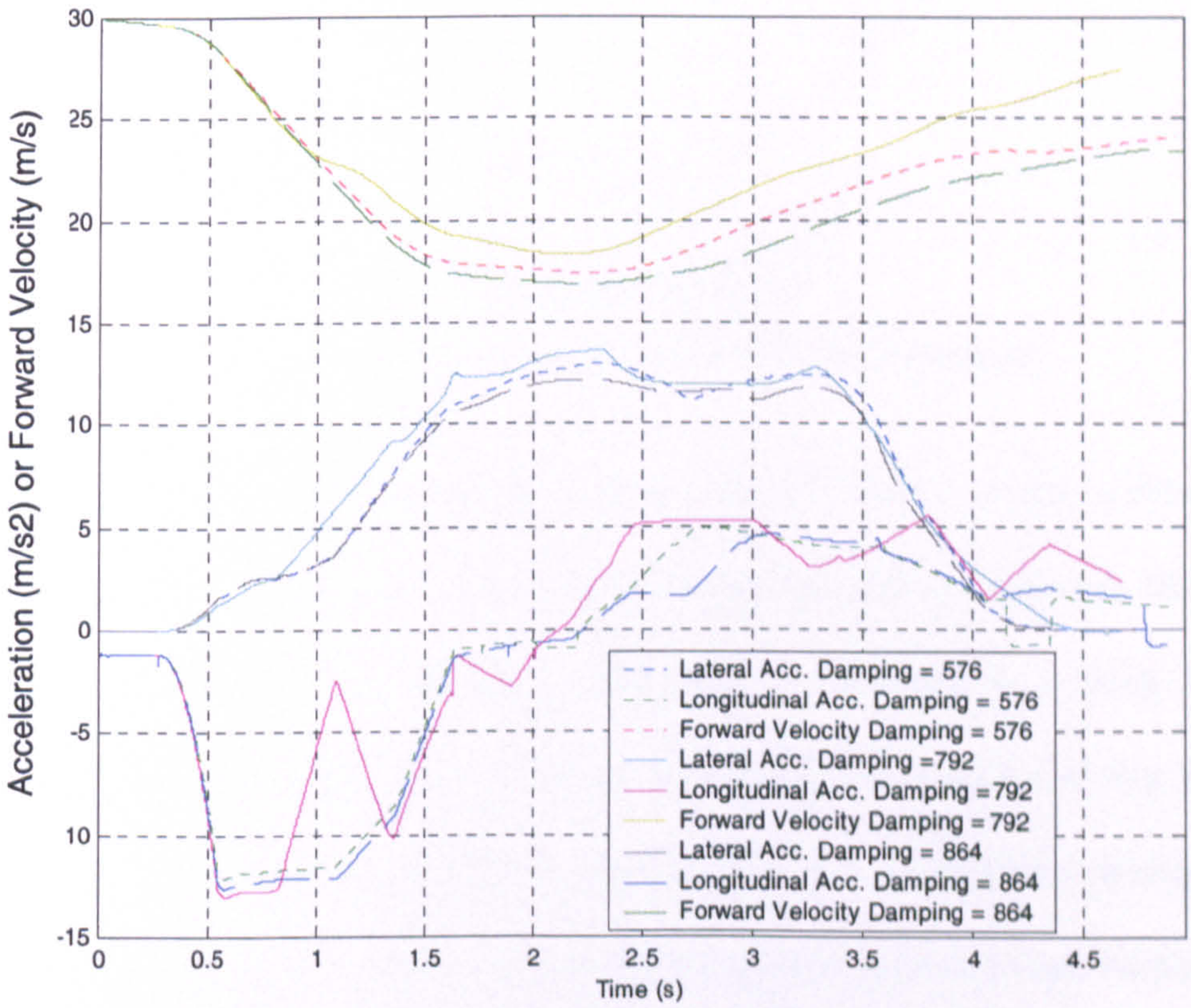


Figure 7.12 – Velocity, lateral and longitudinal acceleration of vehicle.

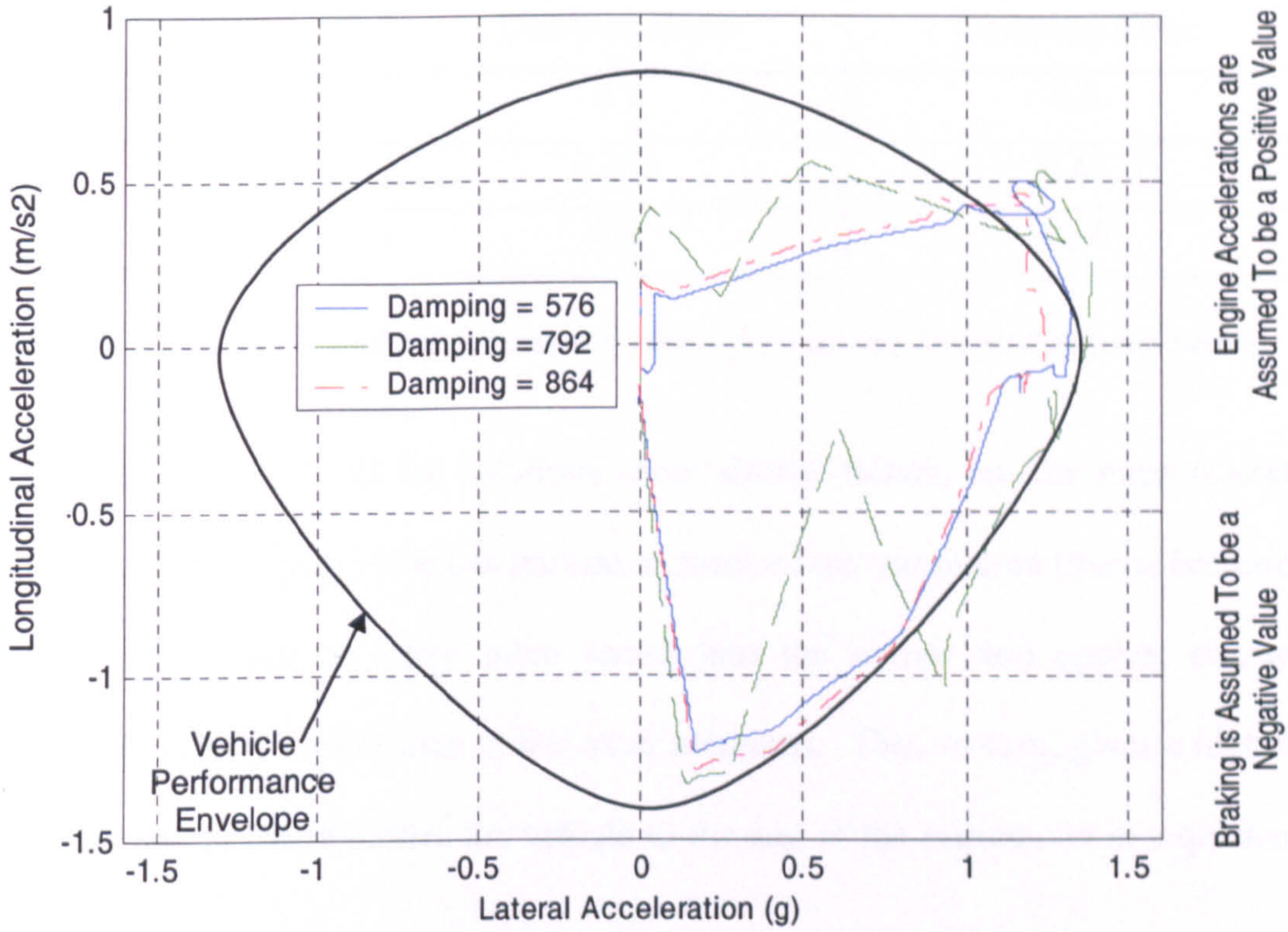


Figure 7.13 – g-g diagram for vehicle in manoeuvre.

The results indicate that a damping value of 792 Nsm^{-1} is the optimum for completing the manoeuvre. In the vehicle model, the vehicle body has no vertical degree of freedom and the vehicle's suspension is modelled as a front and rear torsional spring and damper affecting body roll motion and a front and rear torsional spring and damper affecting body pitch motion. To try to quantify the relationship of the front linear damping value to the vehicle, the damping ratio, found from equation (49) for the front roll and pitch torsional spring stiffness and damping values, are shown in table 7.2 for the three solution values shown in the figures.

$$\zeta = \frac{C_{damp}}{2\sqrt{km}} \qquad (49)$$

where:

C_{damp}: Damping value, Nsm^{-1}

k: Spring stiffness, Nm^{-1}

ζ: Damping ratio

Linear Front Damping Value (Nsm^{-1})	Front Roll Damping Ratio	Front Pitch Damping Ratio
576	0.2	0.3
792	0.275	0.42
864	0.3	0.46

Table 7.2 – Front roll and pitch damping ratios for various linear damping ratios.

Examining figure 7.12, all the solutions show similar trends, but the main reason why the middle damping value has the fastest manoeuvre completion time is because it allows the vehicle to carry more speed into the corner and applies engine acceleration slightly sooner than in the other solutions. This, in turn, gives a higher corner exit speed, which carries the vehicle to the end of the manoeuvre in a quicker time.

It can be seen from figure 7.12, that to allow the vehicle to enter the corner at a higher speed, the longitudinal acceleration plot for the middle damping value is quite uneven, which implies uneven longitudinal driver control input values. With the front pitch damping ratio at 0.42, it appears that this value is better at controlling the change in the longitudinal load transfer due to these uneven longitudinal control values. This ability to allow the uneven longitudinal control values is in contrast to the other damping values, which are not able to complete the manoeuvre with the optimum control matrix solution found for the middle damping value.

The higher and lower damping values do not allow these uneven control values, because the lower damping value causes too much longitudinal load transfer to occur too quickly (i.e. it is under-damped) and thus pushing the transient handling balance of the vehicle more quickly to one extreme or the other which may cause the vehicle to go out of control or leave the track boundaries. On the other hand, the higher

damping value may cause the longitudinal load transfer response to be over-damped and this does not allow the tyres to build up lateral or longitudinal force fast enough to stop the vehicle from going out of control or leaving the racetrack. The higher damping value solution also produces the slowest time to complete the manoeuvre and so, the lower end of the damping range seems to be more desirable as it has to brake less and can apply the throttle sooner.

Furthermore, referring to table 7.2, the front roll damping ratio does not seem to have as great an effect on the vehicle's performance. This is because, the front roll damping ratio does not change as drastically between the different linear damping settings, as seen with the pitch ratios. It probably does however, determine the transient handling balance of the vehicle and so the middle value is assumed to be the most appropriate value as it allows higher lateral accelerations to occur during corner entry and exit.

Finally, the middle damping value solution undergoes the manoeuvre at a slightly higher lateral acceleration and takes slightly longer to reduce this higher lateral acceleration to zero. This means that it takes a slightly wider path to the other solutions by taking longer to make the transition from cornering to straight line running. This also allows the solution to maintain a slightly higher longitudinal acceleration, whilst the lateral acceleration is reduced.

7.4 Conclusions

This chapter gives examples of the use of LTS packages by detailing two parameter sensitivity studies. In both cases the simulation approaches and packages used to conduct the parameter sensitivity studies have been fully detailed, together with user input/output features of the packages.

The first parameter sensitivity study was to study the effect of various vehicle parameters on overall lap time and was conducted using a quasi-static simulation approach based LTS package. It was found that the vehicle was most sensitive to the lateral tyre friction coefficient and that most of the parameters in the baseline vehicle parameter set were already at their optimum values.

The second parameter sensitivity study was on the effect of the front damping value on the time taken to complete a 90 degree right hand corner using a transient simulation approach based Manoeuvre Time Minimisation package. In developing the package, the internal parameters used in the package have been optimised and the robustness of the results examined. It was found that the results were realistic and the package optimised the vehicle control history in a similar manner to an actual driver [3]. It also found the minimum time taken to complete the manoeuvre in each case. A damping value of 792 Nsm^{-1} was found to produce the shortest time to complete the manoeuvre as it allowed more uneven longitudinal driver controls to occur, which meant more speed could be carried into the corner and the throttle applied sooner to accelerate the vehicle out of the corner at a higher speed.

These case studies demonstrate the effectiveness of LTS packages in being able to optimise vehicle performance through varying the vehicle's parameters and showing their potential worth to racing teams.

8 Conclusions

Overall the aims of this research project have been met; a simulation package based on a transient solution has been developed. The key features of this package are that it incorporates a strategy to optimise the racing line through a corner and the solution includes detailed transient effects in contrast to most previous LTS packages which use a quasi-static solution. The work has been presented to the automotive industry in several publications [4, 55, 57, 65] during the course of the research project. Conclusions regarding the details of each stage of the research are outlined below.

A comprehensive study of the literature has shown that computer simulation of racing car handling, through LTS packages, complements the numerous computational tools used by racing teams. These packages allow teams to examine the effect of different vehicle parameter setups to optimise vehicle performance. In similarity with the automotive industry, time is limited and rapid development of new ideas and technology is essential. Thus, the use of a more sophisticated computer simulation allows the team to gain a significant advantage over their competitors. LTS packages, however, are computationally intensive and the correct balance between simulation accuracy and computation time is crucial.

The literature has also shown that nearly all existing packages have used the quasi-static simulation approach. It has been shown by the author that this method does not take into account the effect of roll, pitch and yaw inertia as well as damping and tyre lag effects, due to the constant acceleration assumption across each segment. Another aspect that is not accounted for is the variation in the fastest effective vehicle path along the track, i.e. racing line, due to change in driver control inputs or

vehicle parameters. The approach basically assumes that the racing line found from the actual vehicle data is the fastest in all cases. The only attempts to include transient effects have used a rather simple vehicle model [9, 10, 11].

The author in an attempt to investigate the accuracy of vehicle models in relationship to racing car performance has created two vehicle models, a simple seven DOF and a more sophisticated thirteen DOF model. The simple model contains a four wheel, single body system with lateral, longitudinal and yaw DOF. It uses a Pacejka Magic tyre formula combined slip model, with each wheel having a spin DOF and a quasi-static approximation of weight transfer. The effects of aerodynamic and tyre rolling resistance loads are also accounted for. Powertrain and basic braking and differential models have been included.

The sophisticated vehicle model is a development of the simple model and includes a three body (front and rear unsprung masses and vehicle body sprung mass) system, which adds a roll and pitch DOF to the model. A tyre lag model, giving each tyre an extra degree of freedom, has been included with improved differential and brake system models. In addition to the vehicle models, a path following non-linear preview controller has been produced to enable an initial guess to be found of vehicle control inputs, for use in the transient simulation approach.

To fully valide the vehicle models, a large amount of vehicle handling data has been successfully recorded for a racing car in two testing phases, using purpose built data acquisition systems. The data not only describes the vehicle's individual lateral or longitudinal dynamic handling behaviour, but also its combined lateral and longitudinal dynamic handling behaviour. Comprehensive parameter sets for the vehicles have also been found and data filtering and handling routines produced.

Judgements have been made on the vehicle's performance using the measured responses taken.

The validation study has shown that the simple vehicle model does not fully represent the lateral dynamic behaviour of a racing car. The sophisticated model, however, does represent this due to the inclusion of the roll and tyre lag DOF. Additionally, the sophisticated vehicle model's longitudinal and combined lateral and longitudinal dynamic responses were found to closely correlate with the measured responses.

It was concluded that the sophisticated vehicle model, therefore, may be used with confidence in a LTS package using a transient approach as it is accurate for the full range of lateral dynamic behaviour. The simple model, which is only accurate for low frequency and steady state responses, may be used in a LTS package using a quasi-static approach where the poor representation of the actual vehicle's transient response will not affect the solution results.

To further investigate the various simulation approaches, a comparison study is made between the approaches which indicated that the transient approach, although more complicated and time consuming, would allow for more accurate tuning of a greater number of vehicle parameters. Examples of the use of LTS packages were then detailed in two parameter sensitivity studies. In both cases the simulation approaches and packages used to conduct the parameter sensitivity studies have been fully detailed. As well, the user input/output features of the packages have been shown.

The first parameter sensitivity study was to investigate the effect of various vehicle parameters on overall lap time and was conducted using a quasi-static simulation approach based LTS package. It was found that the vehicle was most sensitive to

lateral tyre coefficient and that most of the parameters in the baseline vehicle parameter set were already at their optimum values.

The second parameter sensitivity study was on the effect of the front damping value on the time taken to complete a 90 degree right hand corner using a transient simulation approach based Manoeuvre Time Minimisation package. In developing the package, the internal parameters used in the package have been optimised and the robustness of the results examined. It was found that the package produced a vehicle control history in a similar manner to an actual driver [3]. It also found the minimum time taken to complete the manoeuvre in each case. A damping value of 792 Nsm^{-1} was found to produce the shortest time to complete the manoeuvre as it allowed more uneven longitudinal driver controls to occur. This meant that more speed could be carried into the corner and the throttle applied sooner to accelerate the vehicle out of the corner giving a shorter overall manoeuvre time.

The research work presented in this thesis has extended previous work on LTS in three areas:

1. Development of a package involving the transient solution using a relatively sophisticated vehicle model.
2. Development of a technique to optimise the driver's racing line through a corner.
3. Execution of two sets of experimental tests to obtain data against which to validate the predicted results from the models.

Finally case studies have been carried out in order to demonstrate the potential use of the developed simulation packages for vehicle parameter selection and performance optimisation.

References

[1] **SAE,** *Vehicle Dynamics Terminology,* SAE Standards J670e, July 1976.

[2] **Purnell A. J.,** *Innovative Computer Technology in Professional Motorsports,* SAE Technical Paper Series 983089, 1998.

[3] **Milliken W. F., and Milliken D. L.,** *Race Car Vehicle Dynamics,* Published by SAE, ISBN = 1-56091-526-9, 1995.

[4] **Siegler B. P., Deakin A., Crolla D.,** *Comparison of Steady State and Transient Racing Car Corner Time Simulation,* SAE Technical Paper Series 2000-01-3563, 2000.

[5] **Moss S. and Pomeroy L.,** *Design and Behaviour of the Racing Car,* Published by William Kimber, London, 1963.

[6] **Roland R., Douglas Jnr. And Thelin C. F.,** *Computer Simulation of Watkins Glen Grand Prix Circuit Performance,* Calspan Report No. ZL-5002-K-1, August 1971.

[7] **Thomas D. W., Segal, Milliken and Michalowicz,** *Analysis and Correlation Using Lap Time Simulation- Dodge Stratus for the North American Touring car,* SAE Technical Paper Series 962528, 1996.

[8] **Pacejka H. B. and Besselink I. J. M.,** *Magic Formula Tyre Model with Transient Properties, Tyre models for Vehicle Dynamics Simulation,* Vehicle system dynamics supplement 27, pp 234-249, 1997.

[9] **Casanova D., Sharp R. S., and Symonds P.,** *A Mathematical Model for Driver Steering Control, with Design, Tuning and Performance Results,* Vehicle System Dynamics 33, pp 289-326, 2000.

[10] **Casanova D., Sharp R. S., and Symonds P.,** *Minimum Time Manoeuvring: the Significance of Yaw Inertia,* Vehicle System Dynamics 34, pp 77-115, 2000.

[11] **Casanova D., Sharp R. S., and Symonds P.,** *On Minimum Time Optimisation of Formula One Cars: the Influence of Vehicle Mass,* Proceedings 5th International symposium on Advanced Vehicle Control, Michigan, USA, 2000.

[12] **Harty D.,** *The Myth of Accuracy,* Journal of the Engineering Integrity Society January 2001 pp 22-28, UK.

[13] **King R.,** *Flexible, User-Friendly and Efficient Model Development and Analysis for Ground Vehicle System Dynamics,* PhD Report, The University of Leeds, 2001.

[14] **Dixon J. C.,** *Tyres, Suspension and Handling 2nd edition,* Published by SAE, ISBN = 1-56091-831-4, 1996.

[15] **Ellis J. R.,** *Vehicle Handling Dynamics,* Mechanical Engineering Publications, London, ISBN = 0-85298-885-0, 1994.

[16] **Crolla D.A.,** *An Introduction to Vehicle Dynamics,* Vehicle Dynamics Group, Department of Mechanical Engineering, University of Leeds, 1991.

[17] **McNay G., Southwick,** *An Approximate Lap Time Minimisation Based on Indy Style Racing Car Geometry,* SAE Technical Paper Series 910011, 1991.

[18] **Heydinger G. J., Garrot W. R. and Chrstos J. P.,** *The Importance of Tire Lag on Simulated Transient Vehicle Response,* SAE Technical Paper Series 910235, 1991.

[19] **Guo K., Ren L. and Lu D.,** *On the Non-steady Comprehensive Tire Model Associated with Lateral Slip Turn-slip and Camber,* Proceedings of AVEC 5th International Symposium on Advanced Vehicle Control, Ann Arbor, Michigan, Aug 22-24 2000.

[20] **Worrall W.,** *Racecar Aerodynamic Development using 'Newpan' CFD Software,* Racecar Magazine Vol. 9 No 2 pp 54 - 64, 1998.

[21] **Dominy R. G.,** *Aerodynamics of Grand Prix cars,* IMechE Part D: Journal of automotive engineering, Vol. 206, No 4 pp 267-274,1992.

[22] **Katz J.,** *New Directions in Race Car Aerodynamics,* Published by SAE, ISBN = 0-8376-0142-8, 1995.

[23] **Gillespie T. D.,** *Fundamentals of Vehicle Dynamics,* Published by SAE, ISBN = 1-56091-199-9, 1992.

[24] **Huang R. W. and Velinsky S. A.,** *Spark Ignition Engine Modelling for Vehicle Dynamic Simulation,* DSC-Vol 52, Advanced automotive technologies, Published by ASME, 1993.

[25] **Faix Louis J.,** *Vehicle Performance Predictions - a PC method,* SAE Technical Paper Series 983076, 1998.

[26] **Swaroop D., Hedrick J. K., Yip P. P.,** *Dynamic Surface Control for a Class of Nonlinear Systems,* IEEE Transactions on Automatic Control, Volume 45 Part 10 pp 1893-1898, 2000.

[27] **Maciuca D. B., Gerdes C., Hedrick J. K.,** *Automatic Braking for IVHS,* AVEC 1994 Proceedings pp 396-401, 1994.

[28] **Tempest J., Guttilla M. and Wurster U.,** *Full Lap Time Simulation of a Formula One Car,* lecture given to 11th European ADAMS user conference, 1996.

[29] **Mühlmeier A.,** *Simulation of a Prototype Car for a Le Mans Race,* lecture given European ADAMS user conference, Berlin November 17 1999.

[30] **TAG Heuer** internet site, http://www.tagheuer-timing.com/pages/index.cfm, 1999.

[31] **McDermott M.,** *Unimagined Heights?,* Racetech (#29) pp 45 to 50, April/May 2000.

[32] **Candelpergher A., Gadola M. and Vetturi D.,** *Developments of a Method for Lap Time Simulation,* SAE Technical Paper Series 2000-01-3562, 2000.

[33] **Vehicle Dynamics Performance** internet site and demo software, http://www.fc.net/~vdp/Contents.htm, 1999.

[34] **Fuller M.,** *Lap time simulation analysis,* Final year engineering thesis, Swinburne University of Technology, published on Pressplay internet site, 1999.

[35] **Pressplay** internet site and demo software, http://www.presspley.com, 1999.

[36] **Carter D.,** *A Track Day with a Lap Time Simulator,* lecture given to engineering in motorsport conference, Williams Grand Prix November 1999.

[37] **Pi Group** internet site and demo software, http://www.pi-group.com/, 1999.

[38] **McDermott M.,** *Cap In Hand,* Racetech Magazine (#22) pp 26 – 29, Feb 1999.

[39] **Mcdermott M.,** *Innovations – PiSim,* Racetech Magazine (#21) pp 19 –20, Dec/Jan 1998/1999.

[40] **Murphy C.,** *Simulation Made Simple,* Racecar Vol 10 No 2 pp 25 to 27, 2000.

[41] **DATAS Trade Stand,** conversation at Autosport Show, NEC, Birmingham, January 2001.

[42] **La Joie J. C.,** *Race Car Performance Optimization,* SAE Technical Paper Series 942492, 1994.

[43] **Jennings M. J.,** *Dynamic Simulation of Race Car Performance,* SAE Technical Paper Series 962539, 1996.

[44] **Anon.,** *Reynard CVD – Lap Simulation,* Reynard LapSim instruction manual, 1999.

[45] **Lowndes E. M.,** *Development of an Intermediate Degree of Freedom Vehicle Dynamics Model for Optimal Design Studies*, PhD Thesis, Department of Mechanical and Aerospace Engineering, North Carolina State University, 1998.

[46] **Lowndes E. M., David J.W.,** *Development of an Intermediate Degree of Freedom Vehicle Dynamics Model for Optimal Design Studies*, Advanced Vehicle Technologies ASME 2000 DE-Vol 106 pp 19 to 24, 2000.

[47] **Gadola M., Vetturi D., Cambiaghi D. and Manzo,** *A tool for Lap Time Simulation,* SAE Technical Paper Series 962529, 1996.

[48] **Coleman T., Branch M. A., and Grace A.,** *MatLab Optimisation Toolbox version 2 user guide,* Published by Mathworks, MatLab software product documentation, 1999.

[49] **Hill J.,** *Test of Time,* Racecar Engineering pp 57 - 62, Oct 2001.

[50] **Orszulik S.,** *Experimental Design and Taguchi,* Orzulik internet site, http://www.orszulik.free-online.co.uk, 2001.

[51] **Kasprzak, E. M., Lewis, K. E. and Milliken, D. L.,** *Steady state Vehicle Optimization Using Pareto Minimum Analysis,* SAE Technical Paper Series 983083, 1998.

[52] **Hacker K., Lewis K. and Kasprzak E. M.,** *Racecar Optimization and Tradeoff Analysis in a Parallel Computing Environment,* SAE Technical Paper Series 2000-01-3564, 2000.

[53] **Miano C. M., Gobbi M., Mastinu G. and Cesarini R.,** *On the Integrated Design of the Tyre-Suspension System of a Racing Car,* Advanced Vehicle Technologies Symposium, DE-Vol 106 pp 25 - 35, Published by ASME , 2000.

[54] **Weighell P.,** *Genetic Software,* Racetech (#33) pp 33 - 44, Dec/Jan 2000/01.

[55] **Siegler B. P., and Crolla D.,** *Comparison of Simulated and Measured Vehicle Handling Performance for a High Performance Racing Car,* Proceedings 8th European Automotive Congress, Conference B: Vehicles for the next decade, Bratislavia, Slovak republic, 2001.

[56] **Chocholek S. E.,** *The Development of a Differential for the Improvement of Traction Control,* Torsen intrnet site, http://www.torsen.com, 2000.

[57] **Siegler B. P., and Crolla D.,** *Lap Time Simulation for Racing Car Design,* Proceedings SAE World Congress, Computer and Optimisation Session, Detroit, USA, March 2002.

[58] **Kok D. O.,** *Optimal Performance of a Racing Car On a Circuit,* Delft University Masters Thesis, 2001.

[59] **Anon.,** *MatLab Signal Processing Toolbox User's Guide Version 4,* Published by Mathworks Inc., 1998.

[60] **International Standards Organisation,** *Passenger cars - Test track for a severe lane-change manoeuvre Part 1: Double-lane change,* ISO/FDIS 3888 - 1:1999(E), 1999.

[61] **Barter N. F., Little J.,** *The Handling and Stability of Motor Vehicles Part 7: Frequency response Measurements and Their Analysis,* MIRA Report No. 1970/10, April 1970.

[62] **Heydinger G. J., Garrott W. R., Chrstos J. P., Guenther D. A.,** *A Methodology for Validating Vehicle Dynamics Simulations,* SAE Technical Paper Series 900128, 1990.

[63] **King R.,** *Development of the Suspension Kinematics Package of a Formula Type Racing Car,* MSc Report, The University of Leeds, 1997.

[64] **James G.,** *Modern Engineering Mathematics,* Published by Addison-Welsey ISBN = 0-201-18054-5, 1994.

[65] **Siegler B. P.,** *Racing Car Lap Time Simulation,* 2000 FISITA World Automotive Congress Youth Conference, Seoul, Korea, 2000.

Appendix A

Derivation of 7 DOF vehicle model equations of motion (before generalised forces are substituted) using Newtonian Approach. Final equation forms and the figures detailing the frames of reference are given in Section 3.2.1.

Transformation table between vehicle fixed reference frame A and ground fixed frame G:

	$\mathbf{a}_1$	$\mathbf{a}_2$	$\mathbf{a}_3$
$\mathbf{g}_1$	$\cos\Psi$	$-\sin\Psi$	0
$\mathbf{g}_2$	$\sin\Psi$	$\cos\Psi$	0
$\mathbf{g}_3$	0	0	1

Linear momentum of vehicle expressed as:

$$\mathbf{L} = mu\mathbf{a}_1 + mv\mathbf{a}_2$$

Angular momentum expressed as:

$$\mathbf{H} = Ir\mathbf{a}_3$$

Their rates of change in A are:

$$\frac{{}^{A}d\mathbf{L}}{dt} = mu\dot{\mathbf{a}}_1 + mv\dot{\mathbf{a}}_2$$

And:

$$\frac{{}^{A}d\mathbf{H}}{dt} = Ir\dot{\mathbf{a}}_3$$

The rate of change of linear momentum in G is:

$$\frac{{}^{G}d\mathbf{L}}{dt} = \frac{{}^{G}d\mathbf{L}}{dt} + {}^{G}\Omega^{A} \times \mathbf{L}$$

Where the last term is the cross product between the angular velocity of A relative to G and the linear momentum.

If:

$$^{G}\Omega^{A} = r\mathbf{a}_3$$

This leads to:

$$\frac{^{G}d\mathbf{L}}{dt} = m(\dot{u} - vr)\mathbf{a}_1 + m(\dot{v} + ur)\mathbf{a}_2$$

The rate of change of angular momentum in G is:

$$\frac{^{G}d\mathbf{H}}{dt} = \frac{^{G}d\mathbf{H}}{dt} + {}^{G}\Omega^{A} \times \mathbf{H}$$

Thus:

$$\frac{^{G}d\mathbf{H}}{dt} = Ir\mathbf{a}_3$$

Therefore, if the sums of the external forces in the $\mathbf{a}_1$ and $\mathbf{a}_2$ directions are denoted by $\sum F_x$ and $\sum F_y$, and the sum of external torques about the $\mathbf{a}_3$ direction is denoted by $\sum M_z$ then the equations of motion can be written as:

$$\sum F_x = m(\dot{u} - vr)$$

$$\sum F_y = m(\dot{v} + ur)$$

$$\sum M_z = I_{zz}\dot{r}$$

For each wheel system, the equation of motion can be written directly:

$$F_{xfl} r_{wfl} = I_{wfl} \dot{\omega}_{fl}$$

$$F_{xfr} r_{wfr} = I_{wfr} \dot{\omega}_{fr}$$

$$F_{xrl} r_{wrl} = I_{wrl} \dot{\omega}_{rl}$$

$$F_{xrr} r_{wrr} = I_{wrr} \dot{\omega}_{rr}$$

Appendix B

Derivation of 13 DOF vehicle model equations of motion (before generalised forces are substituted) using MatLab Symbolic Toolbox based program. The listing below details the derivation of the results by first giving the symbolic toolbox commands, which are in bold, these are followed by the MatLab workspace output. Notes are also given which are preceded by % and are in italics. Final equation forms and the figures detailing the frames of reference are given in Section 3.3.1.

% File creates Lagrangian 9DOF model in algebraic format
% to find equations of motion for car
% DOF are pitch, roll, yaw, lateral, longitudinal and 4x wheel spins
% The 4 tyre lag DOF are given in Section 3.3.2
% Uses symbolic toolbox
% By B Siegler Created 12/12/00
% Last modified 21/12/00

% 1) Find rotation matrices between frames of reference
% 2) Derive energies for each body
% 3) Find general forces
% 4) Find partial derivatives
% 5) Form Lagrangian equation for each DOF
% 6) Thus equations of motion are given before substitution of generalised
% forces

% Notes on dynamics:
% Do not have to take into account height of unsprung mass due to being at
% ground plane
% Long and lat scrub derivatives are zero due to no suspension
% Roll centre heights only affects scrub derivatives

% Notes on algebra:
% All reference frames are in capitals
% Partial derivatives denoted by b
% See end for main simplification of results
% See nomenclature for full listing of variable meanings
% Greek letters indicated by English name, i.e. Φ = psi
% dot *addition to variable denotes time derivative of variable*
% Psidot *replaced with* r
% Subscript: front = fr, *rear =* re, *followed by: right =* ri, *left =* le

==========

```
% 1) Find rotation matrices between frames of reference

syms phi theta psi real;

% Syms command specifies variables phi, theta and psi as real numbers

% Create transformation (by rotation) matrix for sprung mass and unsprung
% mass reference frames.

GtransA = [cos(psi) -sin(psi) 0;...
           sin(psi)  cos(psi) 0;...
              0         0     1];

% Due to inversed z-axis

AtransG = simple(inv(GtransA))

AtransG =

[  cos(psi),  sin(psi),       0]
[ -sin(psi),  cos(psi),       0]
[        0,        0,       1]


BtransApitch = [cos(theta)  0   sin(theta);...
                     0      1        0    ;...
                -sin(theta) 0   cos(theta)];

AtransBpitch = simple(inv(BtransApitch))

AtransBpitch =

[  cos(theta),          0, -sin(theta)]
[          0,          1,          0]
[  sin(theta),          0,  cos(theta)]


BtransAroll =  [1      0          0   ;...
                0 cos(phi) -sin(phi);...
                0 sin(phi)  cos(phi)];

AtransBroll = simple(inv(BtransAroll))

AtransBroll =

[        1,        0,        0]
[        0,  cos(phi),  sin(phi)]
[        0, -sin(phi),  cos(phi)]
```

AtransB = AtransBpitch*AtransBroll

AtransB =

[cos(theta), sin(theta)*sin(phi), -sin(theta)*cos(phi)]
[0, cos(phi), sin(phi)]
[sin(theta), -cos(theta)*sin(phi), cos(theta)*cos(phi)]

BtransA = BtransApitch*BtransAroll

BtransA =

[cos(theta), sin(theta)*sin(phi), sin(theta)*cos(phi)]
[0, cos(phi), -sin(phi)]
[-sin(theta), cos(theta)*sin(phi), cos(theta)*cos(phi)]

syms phidot thetadot r real;

% Velocities in ref frame A

GrotBinA = [0 0 r] + [phidot 0 0]*BtransApitch +[0 thetadot 0]*BtransAroll

GrotBinA =

[phidot*cos(theta), thetadot*cos(phi), r+phidot*sin(theta)-thetadot*sin(phi)]

% Velocities in ref frame B

GrotBinB = [0 0 r]*AtransB + [phidot 0 0] + [0 thetadot 0]

GrotBinB =

[r*sin(theta)+phidot, -r*cos(theta)*sin(phi)+thetadot, r*cos(theta)*cos(phi)]

GrotBinB = GrotBinB*[1 0 0;0 -1 0;0 0 1]

GrotBinB =

[r*sin(theta)+phidot, r*cos(theta)*sin(phi)-thetadot, r*cos(theta)*cos(phi)]

% Finding the position and velocity of P = Position of C of G

% h is height of c of g above roll axis

syms h u v real;

```
PinB = [0 0 -h]';          % Position of P in ref frame B

PinA = BtransA*PinB        % Position of P in ref frame A

PinA =

[ -sin(theta)*cos(phi)*h ]
[           sin(phi)*h   ]
[ -cos(theta)*cos(phi)*h]

velPinG = 0 + cross(GrotBinA,PinA) + [u v 0];

% Cross product for unsprung body fixed axis

velPinG = simple(simple(simple(velPinG)))

% Simple command simplifies results (collects like terms together, etc.)

velPinG =

[-thetadot*cos(phi)^2*cos(theta)*h-(r+phidot*sin(theta)-
thetadot*sin(phi))*sin(phi)*h+u,  -(r+phidot*sin(theta)-
thetadot*sin(phi))*sin(theta)*cos(phi)*h+phidot*cos(theta)^2*cos(phi)*h+v,
phidot*cos(theta)*sin(phi)*h+thetadot*cos(phi)^2*sin(theta)*h]


ub = velPinG*[1 0 0]'     % Sprung mass velocities

ub =

-thetadot*cos(phi)^2*cos(theta)*h-(r+phidot*sin(theta)-
thetadot*sin(phi))*sin(phi)*h+u


vb = velPinG*[0 1 0]'     % Sprung mass velocities

vb =

-(r+phidot*sin(theta)-
thetadot*sin(phi))*sin(theta)*cos(phi)*h+phidot*cos(theta)^2*cos(phi)*h+v


wb = velPinG*[0 0 1]'     % Sprung mass velocities

wb =

phidot*cos(theta)*sin(phi)*h+thetadot*cos(phi)^2*sin(theta)*h

==========================================================================
```

% 2) Derive energies for each body

% Parameters for sprung mass

syms mb Ixxb Ixyb Ixzb Ixyb Iyyb Iyzb Ixzb Iyzb Izzb rollstiff rolldamp pitchstiff pitchdamp g real;

% Parameters for unsprung mass

syms mf mr Izzf Izzr a b real;

% Parameters for wheel – spindot *is wheel rotational velocity,* Ispin *is wheel*
% inertia about axle

syms Ispin spindotfrri spindotfrle spindotreri spindotrele spinfrri spinfrle spinreri spinrele real;

% Kinetic Energy
% ==========

% For Sprung Mass
% ------------------

% Rotational

% Inertia matrix to work out energies

```
I =  [Ixxb -Ixyb -Ixzb;...
     -Ixyb  Iyyb -Iyzb;...
     -Ixzb -Iyzb  Izzb];
```

% Ixyb = Ixzb = Iyzb = 0 due to small compared to main inertia values
% (and symmetry of the vehicle about the x-axis)

```
KEbodyrot = 0.5*GrotBInB*(I*GrotBInB');

KEbodyrot = simple(KEbodyrot)

KEbodyrot =

(1/2*r*sin(theta)+1/2*phidot)*(Ixxb*(r*sin(theta)+phidot)-
Ixyb*(r*cos(theta)*sin(phi)-thetadot)-
Ixzb*r*cos(theta)*cos(phi))+(1/2*r*cos(theta)*sin(phi)-1/2*thetadot)*(-
Ixyb*(r*sin(theta)+phidot)+Iyyb*(r*cos(theta)*sin(phi)-thetadot)-
Iyzb*r*cos(theta)*cos(phi))+1/2*r*cos(theta)*cos(phi)*(-
Ixzb*(r*sin(theta)+phidot)-Iyzb*(r*cos(theta)*sin(phi)-
thetadot)+Izzb*r*cos(theta)*cos(phi))
```

% Translational

KEbodytrans = 0.5*mb*((ub^2)+(vb^2)+(wb^2));

KEbodytrans = simple(KEbodytrans)

KEbodytrans =

1/2*mb*((-thetadot*cos(phi)^2*cos(theta)*h-(r+phidot*sin(theta)-
thetadot*sin(phi))*sin(phi)*h+u)^2+(-(r+phidot*sin(theta)-
thetadot*sin(phi))*sin(theta)*cos(phi)*h+phidot*cos(theta)^2*cos(phi)*h+v)^2+
(phidot*cos(theta)*sin(phi)*h+thetadot*cos(phi)^2*sin(theta)*h)^2)

% For Unsprung Mass
% ----------------------

uf = u;
ur = u;
vf = v + a*r;
vr = v - b*r;

KEaxlef = 0.5*mf*((uf^2)+(vf^2)) +0.5*Izzf*(r^2);

KEaxler = 0.5*mr*((ur^2)+(vr^2)) +0.5*Izzr*(r^2);

% For wheels
% --------------

% Position and velocities

Pfrri = [a (tf/2) 0];
Pfrle = [a (tf/2) 0];
Preri = [-b (-tr/2) 0];
Prele = [-b (-tr/2) 0];

% Therefore KE's

KEfrri = 1/2*Ispin*(spindotfrri^2);
KEfrle = 1/2*Ispin*(spindotfrle^2);
KEreri = 1/2*Ispin*(spindotreri^2);
KErele = 1/2*Ispin*(spindotrele^2);

% Total KE
% ----------

KE = simple(simple(KEfrri + KEfrle + KEreri + KErele + KEaxlef +
KEaxler + KEbodyrot + KEbodytrans));

KE = simple(simple(KE))

KE =

1/2*Ispin*spindotfrri^2+1/2*Ispin*spindotfrle^2+1/2*Ispin*spindotreri^2+1/2*Is
pin*spindotrele^2+1/2*mf*(u^2+(v+a*r)^2)+1/2*Izzf*r^2+1/2*mr*(u^2+(v-
b*r)^2)+1/2*Izzr*r^2+(1/2*r*sin(theta)+1/2*phidot)*(Ixxb*(r*sin(theta)+phidot)-
Ixyb*(r*cos(theta)*sin(phi)-thetadot)-
Ixzb*r*cos(theta)*cos(phi))+(1/2*r*cos(theta)*sin(phi)-1/2*thetadot)*(-
Ixyb*(r*sin(theta)+phidot)+Iyyb*(r*cos(theta)*sin(phi)-thetadot)-
Iyzb*r*cos(theta)*cos(phi))+1/2*r*cos(theta)*cos(phi)*(-
Ixzb*(r*sin(theta)+phidot)-Iyzb*(r*cos(theta)*sin(phi)-
thetadot)+Izzb*r*cos(theta)*cos(phi))+1/2*mb*((-
thetadot*cos(phi)^2*cos(theta)*h-(r+phidot*sin(theta)-
thetadot*sin(phi))*sin(phi)*h+u)^2+(-(r+phidot*sin(theta)-
thetadot*sin(phi))*sin(theta)*cos(phi)*h+phidot*cos(theta)^2*cos(phi)*h+v)^2+
(phidot*cos(theta)*sin(phi)*h+thetadot*cos(phi)^2*sin(theta)*h)^2)

% Potential Energy
% ============

% For Sprung Mass
% ---------------------

PEgravity = mb*g*h*((1-cos(phi))+(1-cos(theta)))

PEgravity =

mb*g*h*(2-cos(phi)-cos(theta))

PEbody = 0.5*rollstiff*(phi^2) + 0.5*pitchstiff*(theta^2) - PEgravity

PEbody =

1/2*rollstiff*phi^2+1/2*pitchstiff*theta^2-mb*g*h*(2-cos(phi)-cos(theta))

% Dissipative Energy
% ==============

% For Sprung Mass
% ---------------------

DEbody = 0.5*rolldamp*(phidot^2) + 0.5*pitchdamp*(thetadot^2)

DEbody =

1/2*rolldamp*phidot^2+1/2*pitchdamp*thetadot^2

% None for Unsprung Mass and Wheels
% ----------

PE = PEbody;
DE = DEbody;

==========

% 3) Find general forces

syms Fxfrri Fxfrle Fxreri Fxrele Fyfrri Fyfrle Fyreri Fyrele rwheel real;

Qlongf = Fxfrri + Fxfrle + Fxreri + Fxrele;

Qlatf = Fyfrri + Fyfrle + Fyreri + Fyrele;

Qyawf = (Fyfrri + Fyfrle)*a - (Fyreri + Fyrele)*b;

Qrollf = 0;

Qpitchf = 0;

Qwheelfrrif = Fxfrri*rwheel;

Qwheelfrlef = Fxfrle*rwheel;

Qwheelrerif = Fxreri*rwheel;

Qwheelrerif = Fxrele*rwheel;

==========

% 4) Find partial derivatives

% Lateral, longitudinal and yaw derivatives

bKEbu = diff(KE,u) *% i.e.* $= \frac{\partial KE}{\partial u}$ *partial derivative*

% Diff command differentiates KE *equation with respect to* u

bKEbu =

mf*u+mr*u+1/2*mb*(-2*thetadot*cos(phi)^2*cos(theta)*h-
2*(r+phidot*sin(theta)-thetadot*sin(phi))*sin(phi)*h+2*u)

bKEbv = diff(KE,v)

bKEbv =

1/2*mf*(2*v+2*a*r)+1/2*mr*(2*v-2*b*r)+1/2*mb*(-2*(r+phidot*sin(theta)-
thetadot*sin(phi))*sin(theta)*cos(phi)*h+2*phidot*cos(theta)^2*cos(phi)*h+2*v
)

bKEbr = diff(KE,r)

bKEbr =

mf*(v+a*r)*a+Izzf*r-mr*(v-
b*r)*b+Izzr*r+1/2*sin(theta)*(Ixxb*(r*sin(theta)+phidot)-
Ixyb*(r*cos(theta)*sin(phi)-thetadot)-
Ixzb*r*cos(theta)*cos(phi))+(1/2*r*sin(theta)+1/2*phidot)*(Ixxb*sin(theta)-
Ixyb*cos(theta)*sin(phi)-Ixzb*cos(theta)*cos(phi))+1/2*cos(theta)*sin(phi)*(-
Ixyb*(r*sin(theta)+phidot)+Iyyb*(r*cos(theta)*sin(phi)-thetadot)-
Iyzb*r*cos(theta)*cos(phi))+(1/2*r*cos(theta)*sin(phi)-1/2*thetadot)*(-
Ixyb*sin(theta)+Iyyb*cos(theta)*sin(phi)-
Iyzb*cos(theta)*cos(phi))+1/2*cos(theta)*cos(phi)*(-Ixzb*(r*sin(theta)+phidot)-
Iyzb*(r*cos(theta)*sin(phi)-
thetadot)+Izzb*r*cos(theta)*cos(phi))+1/2*r*cos(theta)*cos(phi)*(-
Ixzb*sin(theta)-Iyzb*cos(theta)*sin(phi)+Izzb*cos(theta)*cos(phi))+1/2*mb*(-
2*(-thetadot*cos(phi)^2*cos(theta)*h-(r+phidot*sin(theta)-
thetadot*sin(phi))*sin(phi)*h+u)*sin(phi)*h-2*(-(r+phidot*sin(theta)-
thetadot*sin(phi))*sin(theta)*cos(phi)*h+phidot*cos(theta)^2*cos(phi)*h+v)*sin
(theta)*cos(phi)*h)

% Roll derivatives

bKEbphidot = diff(KE,phidot)

bKEbphidot =

1/2*Ixxb*(r*sin(theta)+phidot)-1/2*Ixyb*(r*cos(theta)*sin(phi)-thetadot)-
Ixzb*r*cos(theta)*cos(phi)+(1/2*r*sin(theta)+1/2*phidot)*Ixxb-
(1/2*r*cos(theta)*sin(phi)-1/2*thetadot)*Ixyb+1/2*mb*(-2*(-
thetadot*cos(phi)^2*cos(theta)*h-(r+phidot*sin(theta)-
thetadot*sin(phi))*sin(phi)*h+u)*sin(theta)*sin(phi)*h+2*(-(r+phidot*sin(theta)-
thetadot*sin(phi))*sin(theta)*cos(phi)*h+phidot*cos(theta)^2*cos(phi)*h+v)*(-
sin(theta)^2*cos(phi)*h+cos(theta)^2*cos(phi)*h)+2*(phidot*cos(theta)*sin(phi
)*h+thetadot*cos(phi)^2*sin(theta)*h)*cos(theta)*sin(phi)*h)

bKEbphi = diff(KE,phi)

bKEbphi =

(1/2*r*sin(theta)+1/2*phidot)*(-
Ixyb*r*cos(theta)*cos(phi)+Ixzb*r*cos(theta)*sin(phi))+1/2*r*cos(theta)*cos(ph
i)*(-Ixyb*(r*sin(theta)+phidot)+Iyyb*(r*cos(theta)*sin(phi)-thetadot)-
Iyzb*r*cos(theta)*cos(phi))+(1/2*r*cos(theta)*sin(phi)-
1/2*thetadot)*(Iyyb*r*cos(theta)*cos(phi)+Iyzb*r*cos(theta)*sin(phi))-
1/2*r*cos(theta)*sin(phi)*(-Ixzb*(r*sin(theta)+phidot)-
Iyzb*(r*cos(theta)*sin(phi)-
thetadot)+Izzb*r*cos(theta)*cos(phi))+1/2*r*cos(theta)*cos(phi)*(-
Iyzb*r*cos(theta)*cos(phi)-Izzb*r*cos(theta)*sin(phi))+1/2*mb*(2*(-
thetadot*cos(phi)^2*cos(theta)*h-(r+phidot*sin(theta)-
thetadot*sin(phi))*sin(phi)*h+u)*(2*thetadot*cos(phi)*cos(theta)*h*sin(phi)+th
etadot*cos(phi)*sin(phi)*h-(r+phidot*sin(theta)-
thetadot*sin(phi))*cos(phi)*h)+2*(-(r+phidot*sin(theta)-
thetadot*sin(phi))*sin(theta)*cos(phi)*h+phidot*cos(theta)^2*cos(phi)*h+v)*(th
etadot*cos(phi)^2*sin(theta)*h+(r+phidot*sin(theta)-
thetadot*sin(phi))*sin(theta)*sin(phi)*h-
phidot*cos(theta)^2*sin(phi)*h)+2*(phidot*cos(theta)*sin(phi)*h+thetadot*cos(
phi)^2*sin(theta)*h)*(phidot*cos(theta)*cos(phi)*h-
2*thetadot*cos(phi)*sin(theta)*h*sin(phi)))

bPEbphi = diff(PE,phi)

bPEbphi =

rollstiff*phi-mb*g*h*sin(phi)

bDEbphidot = diff(DE,phidot)

bDEbphidot =

rolldamp*phidot

% Pitch derivatives

bKEbthetadot = diff(KE,thetadot)

bKEbthetadot =

(1/2*r*sin(theta)+1/2*phidot)*Ixyb+1/2*Ixyb*(r*sin(theta)+phidot)-
1/2*Iyyb*(r*cos(theta)*sin(phi)-thetadot)+Iyzb*r*cos(theta)*cos(phi)-
(1/2*r*cos(theta)*sin(phi)-1/2*thetadot)*Iyyb+1/2*mb*(2*(-
thetadot*cos(phi)^2*cos(theta)*h-(r+phidot*sin(theta)-
thetadot*sin(phi))*sin(phi)*h+u)*(-cos(phi)^2*cos(theta)*h+sin(phi)^2*h)+2*(-
(r+phidot*sin(theta)-

thetadot*sin(phi))*sin(theta)*cos(phi)*h+phidot*cos(theta)^2*cos(phi)*h+v)*sin
(phi)*sin(theta)*cos(phi)*h+2*(phidot*cos(theta)*sin(phi)*h+thetadot*cos(phi)^
2*sin(theta)*h)*cos(phi)^2*sin(theta)*h)

bKEbtheta = diff(KE,theta)

bKEbtheta =

1/2*r*cos(theta)*(Ixxb*(r*sin(theta)+phidot)-Ixyb*(r*cos(theta)*sin(phi)-
thetadot)-
Ixzb*r*cos(theta)*cos(phi))+(1/2*r*sin(theta)+1/2*phidot)*(Ixxb*r*cos(theta)+Ix
yb*r*sin(theta)*sin(phi)+Ixzb*r*sin(theta)*cos(phi))-1/2*r*sin(theta)*sin(phi)*(-
Ixyb*(r*sin(theta)+phidot)+Iyyb*(r*cos(theta)*sin(phi)-thetadot)-
Iyzb*r*cos(theta)*cos(phi))+(1/2*r*cos(theta)*sin(phi)-1/2*thetadot)*(-
Ixyb*r*cos(theta)-Iyyb*r*sin(theta)*sin(phi)+Iyzb*r*sin(theta)*cos(phi))-
1/2*r*sin(theta)*cos(phi)*(-Ixzb*(r*sin(theta)+phidot)-
Iyzb*(r*cos(theta)*sin(phi)-
thetadot)+Izzb*r*cos(theta)*cos(phi))+1/2*r*cos(theta)*cos(phi)*(-
Ixzb*r*cos(theta)+Iyzb*r*sin(theta)*sin(phi)-
Izzb*r*sin(theta)*cos(phi))+1/2*mb*(2*(-thetadot*cos(phi)^2*cos(theta)*h-
(r+phidot*sin(theta)-
thetadot*sin(phi))*sin(phi)*h+u)*(thetadot*cos(phi)^2*sin(theta)*h-
phidot*cos(theta)*sin(phi)*h)+2*(-(r+phidot*sin(theta)-
thetadot*sin(phi))*sin(theta)*cos(phi)*h+phidot*cos(theta)^2*cos(phi)*h+v)*(-
3*phidot*cos(theta)*sin(theta)*cos(phi)*h-(r+phidot*sin(theta)-
thetadot*sin(phi))*cos(theta)*cos(phi)*h)+2*(phidot*cos(theta)*sin(phi)*h+thet
adot*cos(phi)^2*sin(theta)*h)*(-
phidot*sin(theta)*sin(phi)*h+thetadot*cos(phi)^2*cos(theta)*h))

bPEbtheta = diff(PE,theta)

bPEbtheta =

pitchstiff*theta-mb*g*h*sin(theta)

bDEbthetadot = diff(DE,thetadot)

bDEbthetadot =

pitchdamp*thetadot

% Wheel spin derivatives

bKEbspinfrrl = diff(KE,spinfrrl)

bKEbspinfrri =

0

bKEbspinfrle = diff(KE,spinfrle)

bKEbspinfrle =

0

bKEbspinreri = diff(KE,spinreri)

bKEbspinreri =

0

bKEbspinrele = diff(KE,spinrele)

bKEbspinrele =

0

bKEbspindotfrri = diff(KE,spindotfrri)

bKEbspindotfrri =

Ispin*spindotfrri

bKEbspindotfrle = diff(KE,spindotfrle)

bKEbspindotfrle =

Ispin*spindotfrle

bKEbspindotreri = diff(KE,spindotreri)

bKEbspindotreri =

Ispin*spindotreri

bKEbspindotrele = diff(KE,spindotrele)

bKEbspindotrele =

Ispin*spindotrele

% Find time derivatives

% Differentiation by time done by substituting u for udot, v for vdot, r for rdot,
% etc. using subs command (sin, cos functions updated as necessary)

syms bKEbutime bKEbvtime bKEbrtime bKEbphidottime bKEbthetadottime bKEbspindotfrritime bKEbspindotfrletime bKEbspindotreletime bKEbspindotreritime real;

% Time derivatives of velocities

syms udot vdot rdot phiddot thetaddot real

bKEbutime = subs(bKEbu,{u,v,r,thetadot,phidot},{udot,vdot,rdot,thetaddot,phiddot})

bKEbutime =

mf*udot+mr*udot+1/2*mb*(-2*thetaddot*cos(phi)^2*cos(theta)*h-(2*rdot+2*phiddot*sin(theta)-2*thetaddot*sin(phi))*sin(phi)*h+2*udot)

bKEbvtime = subs(bKEbv,{u,v,r,thetadot,phidot},{udot,vdot,rdot,thetaddot,phiddot})

bKEbvtime =

1/2*mf*(2*vdot+2*a*rdot)+1/2*mr*(2*vdot-2*b*rdot)+1/2*mb*((-2*rdot-2*phiddot*sin(theta)+2*thetaddot*sin(phi))*sin(theta)*cos(phi)*h+2*phiddot*cos(theta)^2*cos(phi)*h+2*vdot)

bKEbrtime = subs(bKEbr,{u,v,r,thetadot,phidot},{udot,vdot,rdot,thetaddot,phiddot})

bKEbrtime =

mf*(vdot+a*rdot)*a+Izzf*rdot-mr*(vdot-b*rdot)*b+Izzr*rdot+1/2*sin(theta)*(Ixxb*(rdot*sin(theta)+phiddot)-Ixyb*(rdot*cos(theta)*sin(phi)-thetaddot)-Ixzb*rdot*cos(theta)*cos(phi))+(1/2*rdot*sin(theta)+1/2*phiddot)*(Ixxb*sin(theta)-Ixyb*cos(theta)*sin(phi)-Ixzb*cos(theta)*cos(phi))+1/2*cos(theta)*sin(phi)*(-Ixyb*(rdot*sin(theta)+phiddot)+Iyyb*(rdot*cos(theta)*sin(phi)-thetaddot)-Iyzb*rdot*cos(theta)*cos(phi))+(1/2*rdot*cos(theta)*sin(phi)-1/2*thetaddot)*(-Ixyb*sin(theta)+Iyyb*cos(theta)*sin(phi)-Iyzb*cos(theta)*cos(phi))+1/2*cos(theta)*cos(phi)*(-Ixzb*(rdot*sin(theta)+phiddot)-Iyzb*(rdot*cos(theta)*sin(phi)-thetaddot)+Izzb*rdot*cos(theta)*cos(phi))+1/2*rdot*cos(theta)*cos(phi)*(-Ixzb*sin(theta)-Iyzb*cos(theta)*sin(phi)+Izzb*cos(theta)*cos(phi))+1/2*mb*((2*thetaddot*cos(phi)^2*cos(theta)*h+2*(rdot+phiddot*sin(theta)-thetaddot*sin(phi))*sin(phi)*h-2*udot)*sin(phi)*h-(2*(-rdot-phiddot*sin(theta)+thetaddot*sin(phi))*sin(theta)*cos(phi)*h+2*phiddot*cos(theta)^2*cos(phi)*h+2*vdot)*sin(theta)*cos(phi)*h)

bKEbphidottime =
subs(bKEbphidot,{u,v,r,thetadot,phidot},{udot,vdot,rdot,thetaddot,phiddot})

bKEbphidottime =

1/2*Ixxb*(rdot*sin(theta)+phiddot)-1/2*Ixyb*(rdot*cos(theta)*sin(phi)-
thetaddot)-
Ixzb*rdot*cos(theta)*cos(phi)+(1/2*rdot*sin(theta)+1/2*phiddot)*Ixxb-
(1/2*rdot*cos(theta)*sin(phi)-
1/2*thetaddot)*Ixyb+1/2*mb*((2*thetaddot*cos(phi)^2*cos(theta)*h+2*(rdot+p
hiddot*sin(theta)-thetaddot*sin(phi))*sin(phi)*h-
2*udot)*sin(theta)*sin(phi)*h+(2*(-rdot-
phiddot*sin(theta)+thetaddot*sin(phi))*sin(theta)*cos(phi)*h+2*phiddot*cos(th
eta)^2*cos(phi)*h+2*vdot)*(-
sin(theta)^2*cos(phi)*h+cos(theta)^2*cos(phi)*h)+(2*phiddot*cos(theta)*sin(p
hi)*h+2*thetaddot*cos(phi)^2*sin(theta)*h)*cos(theta)*sin(phi)*h)

bKEbthetadottime =
subs(bKEbthetadot,{u,v,r,thetadot,phidot},{udot,vdot,rdot,thetaddot,phiddot})

bKEbthetadottime =

(1/2*rdot*sin(theta)+1/2*phiddot)*Ixyb+1/2*Ixyb*(rdot*sin(theta)+phiddot)-
1/2*Iyyb*(rdot*cos(theta)*sin(phi)-thetaddot)+Iyzb*rdot*cos(theta)*cos(phi)-
(1/2*rdot*cos(theta)*sin(phi)-1/2*thetaddot)*Iyyb+1/2*mb*((-
2*thetaddot*cos(phi)^2*cos(theta)*h-2*(rdot+phiddot*sin(theta)-
thetaddot*sin(phi))*sin(phi)*h+2*udot)*(-
cos(phi)^2*cos(theta)*h+sin(phi)^2*h)+(2*(-rdot-
phiddot*sin(theta)+thetaddot*sin(phi))*sin(theta)*cos(phi)*h+2*phiddot*cos(th
eta)^2*cos(phi)*h+2*vdot)*sin(phi)*sin(theta)*cos(phi)*h+(2*phiddot*cos(theta
)*sin(phi)*h+2*thetaddot*cos(phi)^2*sin(theta)*h)*cos(phi)^2*sin(theta)*h)

% Wheel spin derivatives

syms spinddotfrri spinddotfrie spinddotreri spinddotreie real;

bKEbspindotfrritime = subs(bKEbspindotfrri,spindotfrri,spinddotfrri)

bKEbspindotfrritime =

Ispin*spinddotfrri

bKEbspindotfrietime = subs(bKEbspindotfrie,spindotfrie,spinddotfrie)

bKEbspindotfrietime =

Ispin*spinddotfrie

bKEbspindotreritime = subs(bKEbspindotreri,spindotreri,spinddotreri)

bKEbspindotreritime =

Ispin*spinddotreri

bKEbspindotreletime = subs(bKEbspindotrele,spindotrele,spinddotrele)

bKEbspindotreletime =

Ispin*spinddotrele

% 5) Form Lagrangian equation for each DOF

Qlong = bKEbutime - r*bKEbv

Qlong =

mf*udot+mr*udot+1/2*mb*(-2*thetaddot*cos(phi)^2*cos(theta)*h-
(2*rdot+2*phiddot*sin(theta)-2*thetaddot*sin(phi))*sin(phi)*h+2*udot)-
r*(1/2*mf*(2*v+2*a*r)+1/2*mr*(2*v-2*b*r)+1/2*mb*(-2*(r+phidot*sin(theta)-
thetadot*sin(phi))*sin(theta)*cos(phi)*h+2*phidot*cos(theta)^2*cos(phi)*h+2*v))

Qlat = bKEbvtime + r*bKEbu

Qlat =

1/2*mf*(2*vdot+2*a*rdot)+1/2*mr*(2*vdot-2*b*rdot)+1/2*mb*((-2*rdot-
2*phiddot*sin(theta)+2*thetaddot*sin(phi))*sin(theta)*cos(phi)*h+2*phiddot*co
s(theta)^2*cos(phi)*h+2*vdot)+r*(mf*u+mr*u+1/2*mb*(-
2*thetadot*cos(phi)^2*cos(theta)*h-2*(r+phidot*sin(theta)-
thetadot*sin(phi))*sin(phi)*h+2*u))

Qyaw = bKEbrtime + u*bKEbv - v*bKEbu

Qyaw =

mf*(vdot+a*rdot)*a+Izzf*rdot-mr*(vdot-
b*rdot)*b+Izzr*rdot+1/2*sin(theta)*(Ixxb*(rdot*sin(theta)+phiddot)-
Ixyb*(rdot*cos(theta)*sin(phi)-thetaddot)-
Ixzb*rdot*cos(theta)*cos(phi))+(1/2*rdot*sin(theta)+1/2*phiddot)*(Ixxb*sin(thet
a)-Ixyb*cos(theta)*sin(phi)-
Ixzb*cos(theta)*cos(phi))+1/2*cos(theta)*sin(phi)*(-
Ixyb*(rdot*sin(theta)+phiddot)+Iyyb*(rdot*cos(theta)*sin(phi)-thetaddot)-

Iyzb*rdot*cos(theta)*cos(phi))+(1/2*rdot*cos(theta)*sin(phi)-1/2*thetaddot)*(-
Ixyb*sin(theta)+Iyyb*cos(theta)*sin(phi)-
Iyzb*cos(theta)*cos(phi))+1/2*cos(theta)*cos(phi)*(-
Ixzb*(rdot*sin(theta)+phiddot)-Iyzb*(rdot*cos(theta)*sin(phi)-
thetaddot)+Izzb*rdot*cos(theta)*cos(phi))+1/2*rdot*cos(theta)*cos(phi)*(-
Ixzb*sin(theta)-
Iyzb*cos(theta)*sin(phi)+Izzb*cos(theta)*cos(phi))+1/2*mb*((2*thetaddot*cos(
phi)^2*cos(theta)*h+2*(rdot+phiddot*sin(theta)-thetaddot*sin(phi))*sin(phi)*h-
2*udot)*sin(phi)*h-(2*(-rdot-
phiddot*sin(theta)+thetaddot*sin(phi))*sin(theta)*cos(phi)*h+2*phiddot*cos(th
eta)^2*cos(phi)*h+2*vdot)*sin(theta)*cos(phi)*h)+u*(1/2*mf*(2*v+2*a*r)+1/2*
mr*(2*v-2*b*r)+1/2*mb*(-2*(r+phidot*sin(theta)-
thetadot*sin(phi))*sin(theta)*cos(phi)*h+2*phidot*cos(theta)^2*cos(phi)*h+2*v
))-v*(mf*u+mr*u+1/2*mb*(-2*thetadot*cos(phi)^2*cos(theta)*h-
2*(r+phidot*sin(theta)-thetadot*sin(phi))*sin(phi)*h+2*u))

Qroll = bKEbphidottime - bKEbphi + bPEbphi + bDEbphidot

Qroll =

1/2*Ixxb*(rdot*sin(theta)+phiddot)-1/2*Ixyb*(rdot*cos(theta)*sin(phi)-
thetaddot)-
Ixzb*rdot*cos(theta)*cos(phi)+(1/2*rdot*sin(theta)+1/2*phiddot)*Ixxb-
(1/2*rdot*cos(theta)*sin(phi)-
1/2*thetaddot)*Ixyb+1/2*mb*((2*thetaddot*cos(phi)^2*cos(theta)*h+2*(rdot+p
hiddot*sin(theta)-thetaddot*sin(phi))*sin(phi)*h-
2*udot)*sin(theta)*sin(phi)*h+(2*(-rdot-
phiddot*sin(theta)+thetaddot*sin(phi))*sin(theta)*cos(phi)*h+2*phiddot*cos(th
eta)^2*cos(phi)*h+2*vdot)*(-
sin(theta)^2*cos(phi)*h+cos(theta)^2*cos(phi)*h)+(2*phiddot*cos(theta)*sin(p
hi)*h+2*thetaddot*cos(phi)^2*sin(theta)*h)*cos(theta)*sin(phi)*h)-
(1/2*r*sin(theta)+1/2*phidot)*(-
Ixyb*r*cos(theta)*cos(phi)+Ixzb*r*cos(theta)*sin(phi))-
1/2*r*cos(theta)*cos(phi)*(-
Ixyb*(r*sin(theta)+phidot)+Iyyb*(r*cos(theta)*sin(phi)-thetadot)-
Iyzb*r*cos(theta)*cos(phi))-(1/2*r*cos(theta)*sin(phi)-
1/2*thetadot)*(Iyyb*r*cos(theta)*cos(phi)+Iyzb*r*cos(theta)*sin(phi))+1/2*r*co
s(theta)*sin(phi)*(-Ixzb*(r*sin(theta)+phidot)-Iyzb*(r*cos(theta)*sin(phi)-
thetadot)+Izzb*r*cos(theta)*cos(phi))-1/2*r*cos(theta)*cos(phi)*(-
Iyzb*r*cos(theta)*cos(phi)-Izzb*r*cos(theta)*sin(phi))-1/2*mb*(2*(-
thetadot*cos(phi)^2*cos(theta)*h-(r+phidot*sin(theta)-
thetadot*sin(phi))*sin(phi)*h+u)*(2*thetadot*cos(phi)*cos(theta)*h*sin(phi)+th
etadot*cos(phi)*sin(phi)*h-(r+phidot*sin(theta)-
thetadot*sin(phi))*cos(phi)*h)+2*(-(r+phidot*sin(theta)-
thetadot*sin(phi))*sin(theta)*cos(phi)*h+phidot*cos(theta)^2*cos(phi)*h+v)*(th
etadot*cos(phi)^2*sin(theta)*h+(r+phidot*sin(theta)-
thetadot*sin(phi))*sin(theta)*sin(phi)*h-
phidot*cos(theta)^2*sin(phi)*h)+2*(phidot*cos(theta)*sin(phi)*h+thetadot*cos(
phi)^2*sin(theta)*h)*(phidot*cos(theta)*cos(phi)*h-

2*thetadot*cos(phi)*sin(theta)*h*sin(phi)))+rollstiff*phi-
mb*g*h*sin(phi)+rolldamp*phidot

Qpitch = bKEbthetadottime - bKEbtheta + bPEbtheta + bDEbthetadot

Qpitch =

(1/2*rdot*sin(theta)+1/2*phiddot)*Ixyb+1/2*Ixyb*(rdot*sin(theta)+phiddot)-
1/2*Iyyb*(rdot*cos(theta)*sin(phi)-thetaddot)+Iyzb*rdot*cos(theta)*cos(phi)-
(1/2*rdot*cos(theta)*sin(phi)-1/2*thetaddot)*Iyyb+1/2*mb*((-
2*thetaddot*cos(phi)^2*cos(theta)*h-2*(rdot+phiddot*sin(theta)-
thetaddot*sin(phi))*sin(phi)*h+2*udot)*(-
cos(phi)^2*cos(theta)*h+sin(phi)^2*h)+(2*(-rdot-
phiddot*sin(theta)+thetaddot*sin(phi))*sin(theta)*cos(phi)*h+2*phiddot*cos(th
eta)^2*cos(phi)*h+2*vdot)*sin(phi)*sin(theta)*cos(phi)*h+(2*phiddot*cos(theta
)*sin(phi)*h+2*thetaddot*cos(phi)^2*sin(theta)*h)*cos(phi)^2*sin(theta)*h)-
1/2*r*cos(theta)*(Ixxb*(r*sin(theta)+phidot)-Ixyb*(r*cos(theta)*sin(phi)-
thetadot)-Ixzb*r*cos(theta)*cos(phi))-
(1/2*r*sin(theta)+1/2*phidot)*(Ixxb*r*cos(theta)+Ixyb*r*sin(theta)*sin(phi)+Ixz
b*r*sin(theta)*cos(phi))+1/2*r*sin(theta)*sin(phi)*(-
Ixyb*(r*sin(theta)+phidot)+Iyyb*(r*cos(theta)*sin(phi)-thetadot)-
Iyzb*r*cos(theta)*cos(phi))-(1/2*r*cos(theta)*sin(phi)-1/2*thetadot)*(-
Ixyb*r*cos(theta)-
Iyyb*r*sin(theta)*sin(phi)+Iyzb*r*sin(theta)*cos(phi))+1/2*r*sin(theta)*cos(phi)
(-Ixzb(r*sin(theta)+phidot)-Iyzb*(r*cos(theta)*sin(phi)-
thetadot)+Izzb*r*cos(theta)*cos(phi))-1/2*r*cos(theta)*cos(phi)*(-
Ixzb*r*cos(theta)+Iyzb*r*sin(theta)*sin(phi)-Izzb*r*sin(theta)*cos(phi))-
1/2*mb*(2*(-thetadot*cos(phi)^2*cos(theta)*h-(r+phidot*sin(theta)-
thetadot*sin(phi))*sin(phi)*h+u)*(thetadot*cos(phi)^2*sin(theta)*h-
phidot*cos(theta)*sin(phi)*h)+2*(-(r+phidot*sin(theta)-
thetadot*sin(phi))*sin(theta)*cos(phi)*h+phidot*cos(theta)^2*cos(phi)*h+v)*(-
3*phidot*cos(theta)*sin(theta)*cos(phi)*h-(r+phidot*sin(theta)-
thetadot*sin(phi))*cos(theta)*cos(phi)*h)+2*(phidot*cos(theta)*sin(phi)*h+thet
adot*cos(phi)^2*sin(theta)*h)*(-
phidot*sin(theta)*sin(phi)*h+thetadot*cos(phi)^2*cos(theta)*h))+pitchstiff*thet
a-mb*g*h*sin(theta)+pitchdamp*thetadot

Qwheelfrri = bKEbspindotfrritime - bKEbspinfrri

Qwheelfrri =

Ispin*spinddotfrri

Qwheelfrle = bKEbspindotfrletime - bKEbspinfrle

Qwheelfrle =

Ispin*spinddotfrle

Qwheelreri = bKEbspindotreritime - bKEbspinreri

Qwheelreri =

lspin*spinddotreri

Qwheelrele = bKEbspindotreletime - bKEbspinrele

Qwheelrele =

lspin*spinddotrele

==========

% 6) Thus equations of motion are given

% Further simplifications:

% 1) All 4th order (e.g. ^2^2) are negligible (go to zero)*
% 2) sin(angle) = 0 *and* cos(angle) = 1 *using subs command*
% 3) Products of inertia are zero

Qlongf = subs(Qlong,sin(theta),0);
Qlongf = subs(Qlongf,sin(phi),0);
Qlongf = subs(Qlongf,sin(psi),psi);
Qlongf = subs(Qlongf,cos(theta),1);
Qlongf = subs(Qlongf,cos(phi),1);
Qlongf = subs(Qlongf,cos(psi),1) *% Final simplified form*

Qlongf =

mf*udot+mr*udot+1/2*mb*(-2*thetaddot*h+2*udot)-
r*(1/2*mf*(2*v+2*a*r)+1/2*mr*(2*v-2*b*r)+1/2*mb*(2*phidot*h+2*v))

Qlatf = subs(Qlat,sin(theta),0);
Qlatf = subs(Qlatf,sin(phi),0);
Qlatf = subs(Qlatf,sin(psi),psi);
Qlatf = subs(Qlatf,cos(theta),1);
Qlatf = subs(Qlatf,cos(phi),1);
Qlatf = subs(Qlatf,cos(psi),1) *% Final simplified form*

Qlatf =

1/2*mf*(2*vdot+2*a*rdot)+1/2*mr*(2*vdot-
2*b*rdot)+1/2*mb*(2*phiddot*h+2*vdot)+r*(mf*u+mr*u+1/2*mb*(-
2*thetadot*h+2*u))

```
Qyawf = subs(Qyaw,sin(theta),0);
Qyawf = subs(Qyawf,sin(phi),0);
Qyawf = subs(Qyawf,sin(psi),psi);
Qyawf = subs(Qyawf,cos(theta),1);
Qyawf = subs(Qyawf,cos(phi),1);
Qyawf = subs(Qyawf,cos(psi),1) % Final simplified form
```

Qyawf =

mf*(vdot+a*rdot)*a+Izzf*rdot-mr*(vdot-b*rdot)*b+Izzr*rdot-Ixzb*phiddot+Iyzb*thetaddot+Izzb*rdot+u*(1/2*mf*(2*v+2*a*r)+1/2*mr*(2*v-2*b*r)+1/2*mb*(2*phidot*h+2*v))-v*(mf*u+mr*u+1/2*mb*(-2*thetadot*h+2*u))

```
Qrollf = subs(Qroll,sin(theta),0);
Qrollf = subs(Qrollf,sin(phi),0) - mb*g*h*phi;
% Simplification looses gravity term
Qrollf = subs(Qrollf,sin(psi),psi);
Qrollf = subs(Qrollf,cos(theta),1);
Qrollf = subs(Qrollf,cos(phi),1);
Qrollf = subs(Qrollf,cos(psi),1) % Final simplified form
```

Qrollf =

Ixxb*phiddot+Ixyb*thetaddot-Ixzb*rdot+1/2*mb*(2*phiddot*h+2*vdot)*h+1/2*phidot*Ixyb*r-1/2*r*(-Ixyb*phidot-Iyyb*thetadot-Iyzb*r)+1/2*thetadot*Iyyb*r+1/2*r^2*Iyzb+mb*(-thetadot*h+u)*r*h+rollstiff*phi+rolldamp*phidot-mb*g*h*phi

```
Qpitchf = subs(Qpitch,sin(theta),0) - mb*g*h*theta;
% Simplification looses gravity term
Qpitchf = subs(Qpitchf,sin(phi),0);
Qpitchf = subs(Qpitchf,sin(psi),psi);
Qpitchf = subs(Qpitchf,cos(theta),1);
Qpitchf = subs(Qpitchf,cos(phi),1);
Qpitchf = subs(Qpitchf,cos(psi),1) % Final simplified form
```

Qpitchf =

Ixyb*phiddot+Iyyb*thetaddot+Iyzb*rdot-1/2*mb*(-2*thetaddot*h+2*udot)*h-1/2*r*(Ixxb*phidot+Ixyb*thetadot-Ixzb*r)-1/2*phidot*Ixxb*r-1/2*thetadot*Ixyb*r+1/2*r^2*Ixzb+mb*(phidot*h+v)*r*h+pitchstiff*theta+pitchdamp*thetadot-mb*g*h*theta

```
Qwheelfrrif = Qwheelfrri % Final simplified form
```

Qwheelfrrif =

Ispin*spinddotfrri

Qwheelfrlef = Qwheelfrle *% Final simplified form*

Qwheelfrlef =

lspin*spinddotfrle

Qwheelrerif = Qwheelreri *% Final simplified form*

Qwheelrerif =

lspin*spinddotreri

Qwheelrelef = Qwheelrele *% Final simplified form*

Qwheelrelef =

lspin*spinddotrele

% Generalised forces may now be substituted to form the equations seen in
% section 3.3.1.

Appendix C

Vehicle parameter sets for use in handling models. Notes begin with %

F4 Vehicle Parameter Set

```
% Vehicle Data set for formula sae F4 car
% Created by B Siegler
% Last modified 12/7/01

g = 9.81;

muchange = 1;
% Decrease in coeff. of friction in certain part of circuit
accfail = 1;
% Driver ability to use all the available traction for engine
acceleration
brakefail = 1;
% Driver ability to use all the available traction for braking
driveeffic = 1;
% Drivetrain efficiency
latspin = 30;
% Stop simulation if vehicle is about to go into a spin in m/s2
brout = 0;
% Forward velocity in m/s model will not drop below
offground = 0;
% Normal force in N where wheel is about to loose contact with the
ground

% Tyre lag parameters

sigmamax = 0.027; % Maximum lag time of tyre slip angle response
sigmagain = 3.3;     % Gain on slip angle derivative factor

% Mass and inertia values

mf = 25;          % Mass of front unsprung mass in kg
mr = 25;          % Mass of front unsprung mass in kg
mb = 285.5;       % Mass of sprung mass in kg
m = mb + mf + mr; % Total mass of vehicle

Izz = 212;        % Yaw moment of inertia in Kgm2 of entire vehicle
                     = Izzf + Izzr + Izzb + a^2mf + a^2mr
Ixx = 35;         % Roll moment of inertia in Kgm2 of sprung mass
                     = Ixxb + h^2mb
Iyy = 65;         % Pitch moment of inertia in Kgm2 of sprung mass
                     = Iyyb + h^2mb
Iwheelfr = 0.21;% Moment of inertia for front wheel about spin axis
Iwheelre = 0.21;% Moment of inertia for front wheel about spin axis
Izzf = 8;         % Moment of inertia for front axle about yaw axis
Izzr = 8;         % Moment of inertia for rear axle about yaw axis
```

```
% Vehicle dimensions

l = 1.86;       % Wheelbase in m
a = 0.98;       % Centre of gravity distance from front wheels in m
b = l-a;        % Centre of gravity distance from rear wheels in m
h = 0.336;      % Centre of gravity height in m
tf= 1.26;       % Front track
tr = 1.15;      % Rear track

% Suspension parameters

% Springs

krf = 22070;    % Front ride rate at wheel centre in N/m
krr = 31500;    % Rear ride rate at wheel centre in N/m
antirollf = 10000;    % Front antiroll bar roll rate at wheel centre
                        in Nm/rad
antirollr = 5000;     % Rear antiroll bar roll rate at wheel centre
                        in Nm/rad
Ptyref = 0.827;       % Tyre gauge pressure above atmospheric in bar
ktyre = ((50*Ptyref)+140)*1000;      % Tyre vertical rate N/m
kphityref = 0.5*ktyre*(tf^2);        % Front tyre roll rate in Nm/rad
kphityrer = 0.5*ktyre*(tr^2);        % Rear tyre roll rate in Nm/rad
kthetatyref = ktyre*(a^2);        % Front tyre pitch rate in Nm/rad
kthetatyrer = ktyre*(b^2);           % Rear tyre pitch rate in Nm/rad

kwheelf = (ktyre*krf)/(ktyre - krf);    % Front wheel rate
kwheelr = (ktyre*krr)/(ktyre - krr);    % Rear wheel rate
springf = 0.5*kwheelf*(tf^2); % Front spring roll rate in Nm/rad
springr = 0.5*kwheelr*(tr^2); % Rear spring roll rate in Nm/rad
Pspringf = kwheelf*(a^2);     % Front spring pitch rate in Nm/rad
Pspringr = kwheelr*(b^2);     % Rear spring pitch rate in Nm/rad

kphif = ((springf + antirollf)*kphityref)/(kphityref + (springf +
antirollf));   % Front roll rate Nm/rad

kphir = ((springr + antirollr)*kphityrer)/(kphityrer + (springr +
antirollr));   % Rear roll rate Nm/rad

kthetaf = ((Pspringf*kthetatyref)/(kthetatyref + Pspringf);
% Front pitch rate in Nm/rad
kthetar = (Pspringr*kthetatyrer)/(kthetatyrer + Pspringr);
% Rear pitch rate in Nm/rad

rollstiff = (kphif + kphir);   % Roll stiffness of sprung mass due
                                 to suspension in Nm/rad
pitchstiff = (kthetaf + kthetar);   % Pitch stiffness of sprung mass
                                      due to suspension in Nm/rad

kf = kphif;                          % for quasi-static model
kr = kphir;                          % for quasi-static model

% Dampers

kdampf = 720;              % Front damper rate at wheel centre in Ns/m
kdampr = 720;              % Rear damper rate at wheel centre in Ns/m
dampf = 0.5*kdampf*(tf^2);     % Front damper roll rate in Nsm/rad
dampr = 0.5*kdampr*(tr^2);     % Rear damper roll rate in Nsm/rad
Pdampf = kdampf*(a^2);         % Front damper pitch rate in Nsm/rad
```

```
Pdampr = kdampr*(b^2);          % Rear damper pitch rate in Nsm/rad
pitchdamp = Pdampf + Pdampr + 500;  % Pitch damping of sprung mass
                                      due to suspension
rolldamp = dampf + dampr + 1000;    % Roll damping of sprung mass
                                      due to suspension
zf = 0.030;                         % Front roll centre height
zr = 0.050;                         % Rear roll centre height
deltamax = 30;          % Maximum vehicle steer angle in degrees
steeratio = 6;          % Ratio of steering wheel angle/steered angle
                            at road wheels

% Aerodynamic data

cd = 0.9;               % Aero coeff. of drag
fa = 0.8;               % Frontal area of vehicle in m2
clf = -0.14;            % Front lift coeff. (+ve is downforce)
clr = 0.17;             % Rear lift coeff. (+ve is downforce)
murres = 0.01;          % Rolling resistance friction value per wheel
mu = 1.3;               % Friction coeff.
p = 40000;              % Engine power in Watts
rho = 1.22;             % Air mass density in Kg/m3
brabi = 0.5;            % Brake bias ratio to rear

% Braking System

radcalf = 0.082;            % Effective radius of front caliper (m)
radcalr = 0.103;            % Effective radius of rear caliper (m)
Acalf = 0.001548;           % Area of caliper front (m2)
Acalr = 0.001548;           % Area of caliper rear (m2)
mupad = 0.5;                % Frictional coeff. at pad-disc interface
Amasterf = 0.0002375;       % Area of master cylinder front (m2)
Amasterr = 0.000306;        % Area of master cylinder rear (m2)
ratiopedal = 4.5;           % Ratio = foot to pedal pivot
                            distance/pushrod to pedal pivot distance
balbaratio = 0.42;          % Balance bar ratio front to rear

brakefr = (ratiopedal*balbaratio/Amasterf)*(radcalf*Acalf*mupad);
% Foot wheel torque gain produced by a foot force in N
brakere = (ratiopedal*balbaratio/Amasterf)*(radcalr*Acalr*mupad);
% Foot wheel torque gain produced by a foot force in N
maxbrake = 600;
% Maximum force on brake pedal (i.e. will lock up at any speed)

Ddiff = 0.35;  % Maximum perecentage of torque transferred between
rear wheels (bias ratio = 3.4 in this case)
Bdiff = 0.09;  % Differential sensitivity (controls difference in
slip ratio at which max transfer occurs)

% To make tyres be on other side of car

turnround = ones(57,1);
turnround(8,1) = -1;
turnround(13,1) = -1;
turnround(14,1) = -1;
turnround(16,1) = -1;
turnround(17,1) = -1;
turnround(40,1) = -1;
turnround(43,1) = -1;
turnround(44,1) = -1;
turnround(52,1) = -1;
```

```
% Goodyear 20x6.5-13inch at 15psi provided by Delft

pac=0;                  % Vector version of Pacejka coeffs.

% Pure lateral coeffs.

pac(1,1)   = 1445;                      %FNOMIN
pac(2,1)   = 1.676;                     %PCY1
pac(3,1)   = -2.587;                    %PDY1
pac(4,1)   = 0.59325;                   %PDY2
pac(5,1)   = -3.8474;                   %PDY3
pac(6,1)   = -0.14887;                  %PEY1
pac(7,1)   = 0.56009;                   %PEY2
pac(8,1)   = 0.023786;                  %PEY3
pac(9,1)   = 4.1175;                    %PEY4
pac(10,1)  = -34.238;                   %PKY1
pac(11,1)  = 1.0867;                    %PKY2
pac(12,1)  = 0.73877;                   %PKY3
pac(13,1)  = 0.0058088;                 %PHY1
pac(14,1)  = -0.0007589;                %PHY2
pac(15,1)  = 0.10852;                   %PHY3
pac(16,1)  = 0.041154;                  %PVY1
pac(17,1)  = -0.055694;                 %PVY2
pac(18,1)  = -0.72216;                  %PVY3
pac(19,1)  = 0.24275;                   %PVY4
pac(20,1)  = 0;                         % Camber angle
pac(21,1)  = 0.5385*muchange;
% LMuX - Value of mu at road surface change from measured data
(lambda MuY)

% Pure longitudinal coeffs.

pac(22,1)   = 4361                      ;%FNOMIN
pac(23,1)   = 1.6116                    ;%PCX1
pac(24,1)   = 1.1005                    ;%PDX1
pac(25,1)   = -0.0141                   ;%PDX2
pac(26,1)   = 0.02261                   ;%PEX1
pac(27,1)   = 0.16482                   ;%PEX2
pac(28,1)   = 0.21884                   ;%PEX3
pac(29,1)   = 0                         ;%PEX4
pac(30,1)   = 18.385                    ;%PKX1
pac(31,1)   = 1.5051                    ;%PKX2
pac(32,1)   = 0.29119                   ;%PKX3
pac(33,1)   = -0.000551                 ;%PHX1
pac(34,1)  = 0.0001                     ;%PHX2
pac(35,1)   = 0                         ;%PVX1
pac(36,1)   = 0                         ;%PVX2
pac(37,1)    = 1.25*muchange;
% LMuX - Value of mu at road surface change from measured data
(lambda MuY)

% Combined lateral coeff.

pac(38,1) = 6.1187;                     %rby1
pac(39,1) = 2.8069;                     %rby2
pac(40,1) = -0.0091738;                 %rby3
pac(41,1) = 1.004;                      %rcy1
pac(42,1) = -0.035516;                  %rhy1
pac(43,1) = 0.046621;                   %rvy1
pac(44,1) = 0.048196;                   %rvy2
pac(45,1) = 0.54064;                    %rvy3
```

```
pac(46,1) = 11.444;                    %rvy4
pac(47,1) = 1.9;                       %rvy5
pac(48,1) = -10.734;                   %rvy6

% Combined longitudinal coeff.

pac(49,1) = 10.395;                    %rbx1
pac(50,1) = -6.3236;                   %rbx2
pac(51,1) = 0.99326;                   %rcx1
pac(52,1) = -0.0029427;                %rhx1

% Radius and stiffness

pac(53,1) = 0.232;                 % Unloaded tyre radius
pac(54,1) = 1.9e5;%4;              % Vertical tyre stiffness
pac(55,1) = 500;                   % Vertical tyre damping
pac(56,1) = 2.5;                   % Mass of tyre belt
pac(57,1) = 700;                   % Fnominal normal load sigma (lag
parameter) found at
rrad = pac(53,1);              % Wheel rolling radius (old model)

% Engine Torque Curve (rpm & Nm)


revs =
[0;2000;2200;2400;2600;2800;3000;3200;3400;3600;3800;4000;4200;4400;
4600;4800;...
5000;5200;5400;5600;5800;6000;6200;6400;6600;6800;7000;7200;7400;760
0;7800;...
8000;8200;8400;8600;8800;9000;9200;9400;9600;9800;10000;10200;10400;
10600;...
10800;11000;11200;11400;11600;11800;12000;12200;12400;12600;12800;13
000;13200];

torque =
[35;35;35;35;35;35;35;35;35;35.89065817;36.42507543;36.90843949;37.4
322111;37.91053851;...
38.74605372;39.14371019;39.90764331;40.0066144;40.42108988;41.732825
3;42.9540962;44.0955414;...
45.28046024;46.39281449;47.43630573;48.41841514;49.344404;49.8115711
3;50.25348597;50.67088502;...
51.06810387;51.44426752;51.26611776;51.09531392;50.07702563;50.77699
768;50.62845011;50.64628912;...
50.66336902;50.67973726;50.69543741;50.70955414;50.72405395;50.73891
475;50.75141209;50.35651097;...
49.44093804;48.55806415;47.70616829;44.98270371;41.72892152;37.97133
758;34.03544951;30.22729608;...
26.24734607;22.96651075;19.7866242;5.56745802];

% Standard box

primred=1/1.863;

GR1=1/2.928;    % Gear ratios 1 to 6
GR2=1/2.062;
GR3=1/1.647;
GR4=1/1.368;
GR5=1/1.2;
GR6=1/1.086;
```

```
engsprocket=14;          % No. of teeth on engine sprocket
drisprocket=52;          % No. of teeth on final drive sprocket

redline=11300;           % rpm change-up engine speed
changedown=7000;         % rpm change down engine speed
changetime =0.2;         % Time to change gear

% Calculated values

over1=GR1*engsprocket*primred/drisprocket;
over2=GR2*engsprocket*primred/drisprocket;
over3=GR3*engsprocket*primred/drisprocket;
over4=GR4*engsprocket*primred/drisprocket;
over5=GR5*engsprocket*primred/drisprocket;
over6=GR6*engsprocket*primred/drisprocket;
```

```
hdiff = h - (zf + ((a/l)*(zr-zf))); % Roll moment arm

h = hdiff;
% Due to fact wheels are in ground plane therefore h should be hdiff
for 9dof model
Ixxb = Ixx - h*h*mb;
Iyyb = Iyy - 0.1*0.1*mb;   % As pitch axis must be close to C of G
```

F5 Vehicle Parameter Set

```
% Vehicle Data set for formula sae F5 car
% Created by B Siegler 19/7/01
% Last modified 1/8/01

g = 9.81;

muchange = 1;
% Decrease in coeff. of friction in certain part of circuit
accfail = 1;
% Driver ability to use all the available traction for engine
acceleration
brakefail = 1;
% Driver ability to use all the available traction for braking
driveeffic = 1;
% Drivetrain efficiency
latspin = 30;
% Stop simulation if vehicle is about to go into a spin in m/s2
brout = 0.5;
% Forward velocity in m/s model will not drop below
offground = 0;
% Normal force in N where wheel is about to loose contact with the
ground

% Tyre lag parameters

sigmamax = 0.027; % Maximum lag time of tyre slip angle response
sigmagain = 3.3;      % Gain on slip angle derivative factor
```

```
% Mass and inertia values

mf = 20.4;              % Mass of front unsprung mass in kg
mr = 20.4;              % Mass of front unsprung mass in kg
mb = 262.9;             % Mass of sprung mass in kg
m = mb + mf + mr;       % Total mass of vehicle
Izz = 200;      % Yaw moment of inertia in Kgm2 of entire vehicle =
                        Izzf + Izzr + Izzb + a^2mf + a^2mr
Ixx = 35;       % Roll moment of inertia in Kgm2 of sprung mass =
                        Ixxb + h^2mb
Iyy = 65;       % Pitch moment of inertia in Kgm2 of sprung mass =
                        Iyyb + h^2mb
Iwheelfr = 0.21;% Moment of inertia for front wheel about spin axis
Iwheelre = 0.21;% Moment of inertia for Rear wheel about spin axis
Izzf = 8;       % Moment of inertia for front axle about yaw axis
Izzr = 8;       % Moment of inertia for rear axle about yaw axis

% Vehicle dimensions

l = 1.8;        % Wheelbase in m
a = 0.98;       % Centre of gravity distance from front wheels in m
b = l-a;        % Centre of gravity distance from rear wheels in m
h = 0.336;      % Centre of gravity height in m
tf = 1.15;      % Front track
tr = 1.1;       % Rear track

% Suspension parameters

% Springs

kspringf = (350*9.81*1000)/(2.205*25.4);
% Front side spring rate in N/m from lb/inch
kspringr = (400*9.81*1000)/(2.205*25.4);
% Rear side spring rate in N/m from lb/inch
MRspringf = 0.68;                       % Front side spring motion ratio
MRspringr = 0.68;                       % Rear side spring motion ratio
krf = kspringf*(MRspringf^2);
% Front ride rate at wheel centre in N/m
krr = kspringr*(MRspringr^2);
% Rear ride rate at wheel centre in N/m
antirollf = 5000;
% Front antiroll bar roll rate at wheel centre in Nm/rad
antirollr = 0;
% Rear antiroll bar roll rate at wheel centre in Nm/rad

Ptyref = 0.827;       % Tyre gauge pressure above atmospheric in bar
ktyre = ((50*Ptyref)+140)*1000;  % Tyre vertical rate N/m
kphityref = 0.5*ktyre*(tf^2);    % Front tyre roll rate in Nm/rad
kphityrer = 0.5*ktyre*(tr^2);    % Rear tyre roll rate in Nm/rad
kthetatyref = ktyre*(a^2);       % Front tyre pitch rate in Nm/rad
kthetatyrer = ktyre*(b^2);       % Rear tyre pitch rate in Nm/rad

kwheelf = (ktyre*krf)/(ktyre - krf);    % Front wheel rate
kwheelr = (ktyre*krr)/(ktyre - krr);    % Rear wheel rate
springf = 0.5*kwheelf*(tf^2);    % Front spring roll rate in Nm/rad
springr = 0.5*kwheelr*(tr^2);    % Rear spring roll rate in Nm/rad
Pspringf = kwheelf*(a^2);        % Front spring pitch rate in Nm/rad
Pspringr = kwheelr*(b^2);        % Rear spring pitch rate in Nm/rad
```

```
kphif = ((springf + antirollf)*kphityref)/(kphityref + (springf +
antirollf));
% Front roll rate Nm/rad
kphir = ((springr + antirollr)*kphityrer)/(kphityrer + (springr +
antirollr));
% Rear roll rate Nm/rad
kthetaf = ((Pspringf*kthetatyref)/(kthetatyref + Pspringf);
% Front pitch rate in Nm/rad
kthetar = (Pspringr*kthetatyrer)/(kthetatyrer + Pspringr);
% Rear pitch rate in Nm/rad

rollstiff = (kphif + kphir);
% Roll stiffness of sprung mass due to suspension in Nm/rad
pitchstiff = (kthetaf + kthetar);
% Pitch stiffness of sprung mass due to suspension in Nm/rad

kf = kphif;     % for quasi-static model
kr = kphir;     % for quasi-static model

% Dampers

damperf = 2000;                         % Front damper rate in Ns/m
damperr = 2000;                         % Rear damper rate in Ns/m
kdampf = damperf*(MRspringf^2);
% Front ride rate at wheel centre in N/m
kdampr = damperr*(MRspringr^2);
% Rear ride rate at wheel centre in N/m
dampf = 0.5*kdampf*(tf^2);              % Front damper roll rate in Nsm/rad
dampr = 0.5*kdampr*(tr^2);              % Rear damper roll rate in Nsm/rad
Pdampf = kdampf*(a^2);                  % Front damper pitch rate in
Nsm/rad
Pdampr = kdampr*(b^2);                  % Rear damper pitch rate in Nsm/rad

pitchdamp = Pdampf + Pdampr;
% Pitch damping of sprung mass due to suspension
rolldamp = dampf + dampr;
% Roll damping of sprung mass due to suspension

zf = 0.025;                             % Front roll centre height
zr = 0.050;                             % Rear roll centre height
deltamax = 30;                          % Maximum vehicle steer angle in degrees
steeratio = 6;
% Ratio of steering wheel angle/steered angle at road wheels

% Aerodynamic data

cd = 1.05;          % Aero coeff. of drag
fa = 0.8;           % Frontal area of vehicle in m2
clf = -0.14;        % Front lift coeff. (+ve is downforce)
clr = 0.17;         % Rear lift coeff. (+ve is downforce)
murres = 0.016;     % Rolling resistance friction value per wheel
mu = 1.3;           % Friction coeff.
p = 40000;          % Engine power in Watts
rho = 1.22;         % Air mass density in Kg/m3
brabi = 0.5;        % Brake bias ratio to rear

% Braking System

radcalf = 0.082;            % Effective radius of front caliper (m)
radcalr = 0.103;            % Effective radius of rear caliper (m)
Acalf = 0.001548;           % Area of caliper front (m2)
```

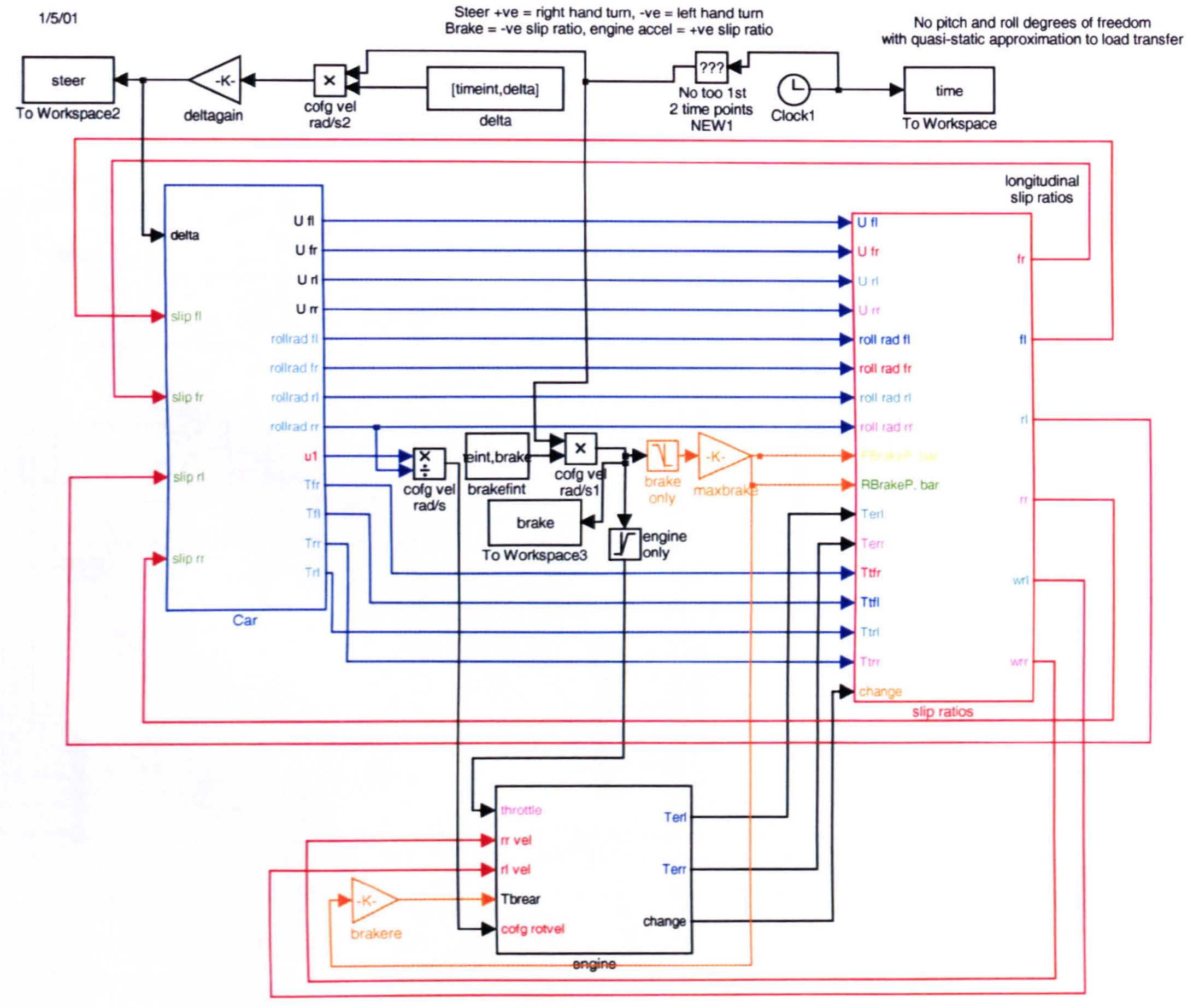
```
Acalr = 0.001548;          % Area of caliper rear (m2)
mupad = 0.35;              % Frictional coeff. at pad-disc interface
Amasterf = 0.0002375;      % Area of master cylinder front (m2)
Amasterr = 0.000306;       % Area of master cylinder rear (m2)
ratiopedal = 4.5;          % Ratio = foot to pedal pivot
                              distance/pushrod to pedal pivot distance
balbaratio = 0.42;         % Balance bar ratio front to rear

brakefr = (ratiopedal*balbaratio/Amasterf)*(radcalf*Acalf*mupad);
% Foot wheel torque gain produced by a foot force in N
brakere = (ratiopedal*balbaratio/Amasterf)*(radcalr*Acalr*mupad);
% Foot wheel torque gain produced by a foot force in N
maxbrake = 600;
% Maximum force on brake pedal (i.e. will lock up at any speed)

Ddiff = 0.35;
% Maximum perecentage of torque transferred between rear wheels
(bias ratio = 3.4 in this case)
Bdiff = 0.09;
% Differential sensitivity (controls difference in slip ratio at
which max transfer occurs)

% To make tyres be on other side of car

turnround = ones(57,1);
turnround(8,1) = -1;
turnround(13,1) = -1;
turnround(14,1) = -1;
turnround(16,1) = -1;
turnround(17,1) = -1;
turnround(40,1) = -1;
turnround(43,1) = -1;
turnround(44,1) = -1;
turnround(52,1) = -1;

% Goodyear 20x6.5-13inch at 15psi provided by Delft

pac=0;                     % Vector version of Pacejka coeffs.

% Pure lateral coeffs.

pac(1,1)  = 1445;                    %FNOMIN
pac(2,1)  = 1.676;                   %PCY1
pac(3,1)  = -2.587;                  %PDY1
pac(4,1)  = 0.59325;                 %PDY2
pac(5,1)  = -3.8474;                 %PDY3
pac(6,1)  = -0.14887;                %PEY1
pac(7,1)  = 0.56009;                 %PEY2
pac(8,1)  = 0.023786;                %PEY3
pac(9,1)  = 4.1175;                  %PEY4
pac(10,1) = -34.238;                 %PKY1
pac(11,1) = 1.0867;                  %PKY2
pac(12,1) = 0.73877;                 %PKY3
pac(13,1) = 0.0058088;               %PHY1
pac(14,1) = -0.0007589;              %PHY2
pac(15,1) = 0.10852;                 %PHY3
pac(16,1) = 0.041154;                %PVY1
pac(17,1) = -0.055694;               %PVY2
pac(18,1) = -0.72216;                %PVY3
pac(19,1) = 0.24275;                 %PVY4
pac(20,1) = 0;                       % Camber angle
```

```
pac(21,1) = 0.5385*muchange;
% LMuX - Value of mu at road surface change from measured data
(lambda MuY)

% Pure longitudinal coeffs.

pac(22,1)  = 4361                    ;%FNOMIN
pac(23,1)  = 1.6116                  ;%PCX1
pac(24,1)  = 1.1005                  ;%PDX1
pac(25,1)  = -0.0141                 ;%PDX2
pac(26,1)  = 0.02261                 ;%PEX1
pac(27,1)  = 0.16482                 ;%PEX2
pac(28,1)  = 0.21884                 ;%PEX3
pac(29,1)  = 0                       ;%PEX4
pac(30,1)  = 18.385                  ;%PKX1
pac(31,1)  = 1.5051                  ;%PKX2
pac(32,1)  = 0.29119                 ;%PKX3
pac(33,1)  = -0.000551               ;%PHX1
pac(34,1) = 0.0001                   ;%PHX2
pac(35,1)  = 0                       ;%PVX1
pac(36,1)  = 0                       ;%PVX2
pac(37,1)   = 1.25*muchange;          % LMuX - Value of mu at road
surface change from measured data (lambda MuY)

% Combined lateral coeff.

pac(38,1) = 6.1187;                  %rby1
pac(39,1) = 2.8069;                  %rby2
pac(40,1) = -0.0091738;              %rby3
pac(41,1) = 1.004;                   %rcy1
pac(42,1) = -0.035516;               %rhy1
pac(43,1) = 0.046621;                %rvy1
pac(44,1) = 0.048196;                %rvy2
pac(45,1) = 0.54064;                 %rvy3
pac(46,1) = 11.444;                  %rvy4
pac(47,1) = 1.9;                     %rvy5
pac(48,1) = -10.734;                 %rvy6

% Combined longitudinal coeff.

pac(49,1) = 10.395;                  %rbx1
pac(50,1) = -6.3236;                 %rbx2
pac(51,1) = 0.99326;                 %rcx1
pac(52,1) = -0.0029427;              %rhx1

% Radius and stiffness

pac(53,1) = 0.232;                   % Unloaded tyre radius
pac(54,1) = 1.9e5;                   % Vertical tyre stiffness
pac(55,1) = 500;                     % Vertical tyre damping
pac(56,1) = 2.5;                     % Mass of tyre belt
pac(57,1) = 700;
% Fnominal normal load sigma (lag parameter) found at
rrad = pac(53,1);                    % Wheel rolling radius (old model)
```

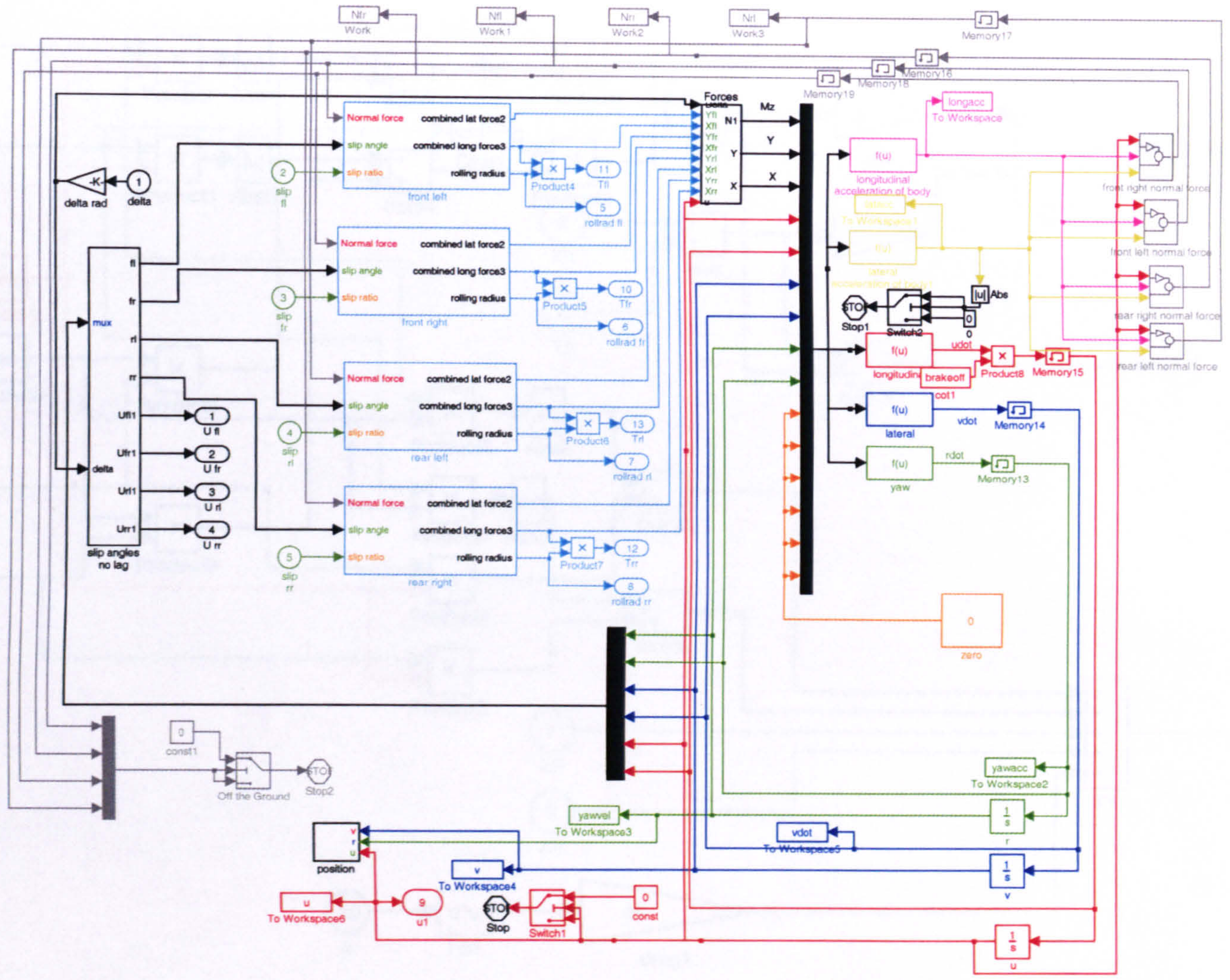
```
% Engine Torque Curve (rpm & Nm)

load f5engine.mat % See figure 3.5, section 3.3.4

% Standard box

primred=1/1.863;

GR1=1/2.928;    % Gear ratios 1 to 6
GR2=1/2.062;
GR3=1/1.647;
GR4=1/1.368;
GR5=1/1.2;
GR6=1/1.086;

engsprocket=13;        % =No. of teeth on engine sprocket
drisprocket=60;        % =No. of teeth on final drive sprocket

redline=10250;
% rpm change-up engine speed
changedown=6500;  % rpm change down engine speed
changetime =0.2;  % Time to change gear

% Calculated values

over1=GR1*engsprocket*primred/drisprocket;
over2=GR2*engsprocket*primred/drisprocket;
over3=GR3*engsprocket*primred/drisprocket;
over4=GR4*engsprocket*primred/drisprocket;
over5=GR5*engsprocket*primred/drisprocket;
over6=GR6*engsprocket*primred/drisprocket;

hdiff = h - (zf + ((a/l)*(zr-zf))); % Roll moment arm

h = hdiff;
% Due to fact wheels are in ground plane therefore h should be hdiff
for 9dof model
Ixxb = Ixx - h*h*mb;
Iyyb = Iyy - 0.1*0.1*mb;    % As pitch axis must be close to C of G
```

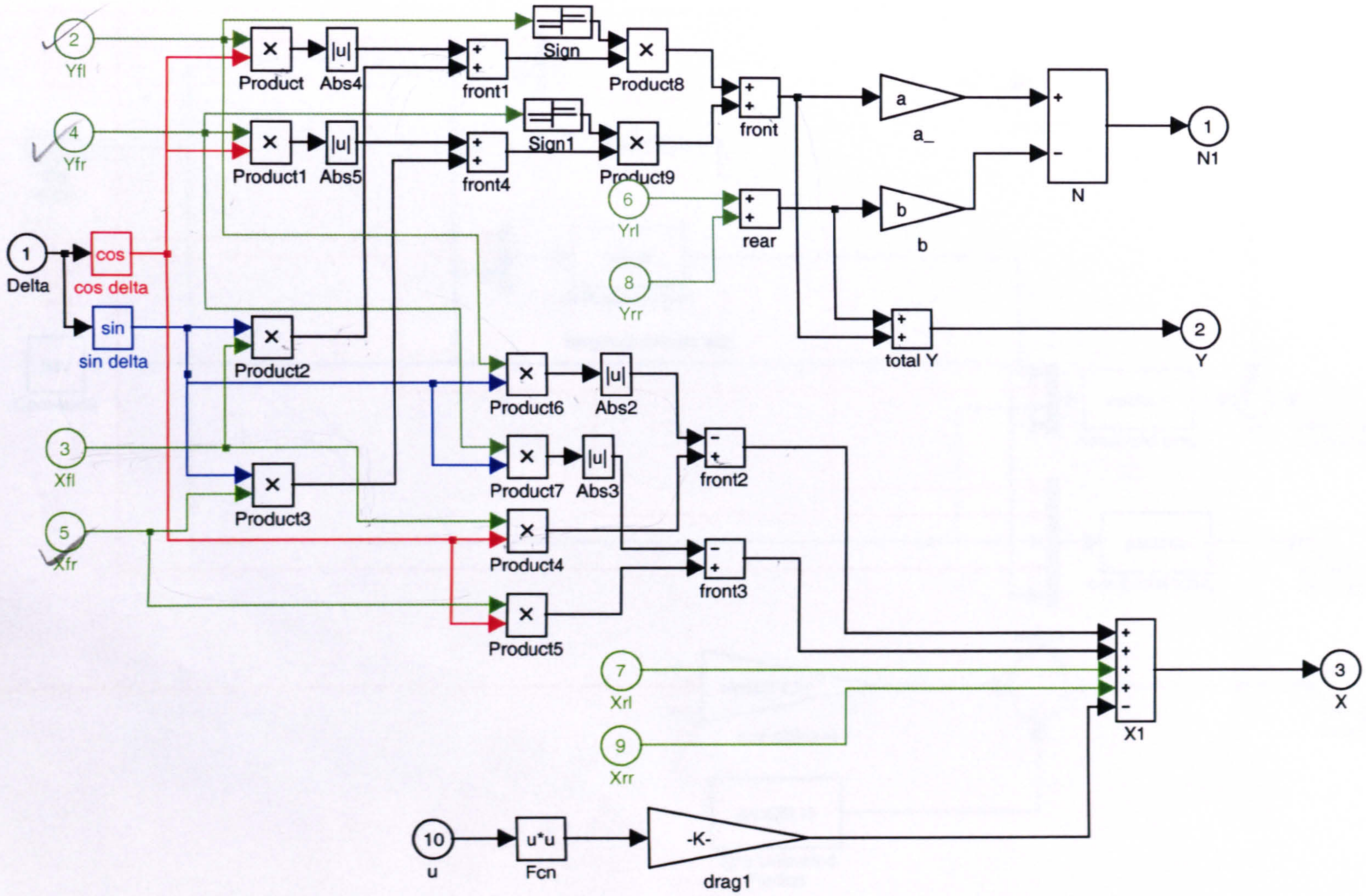
Appendix D

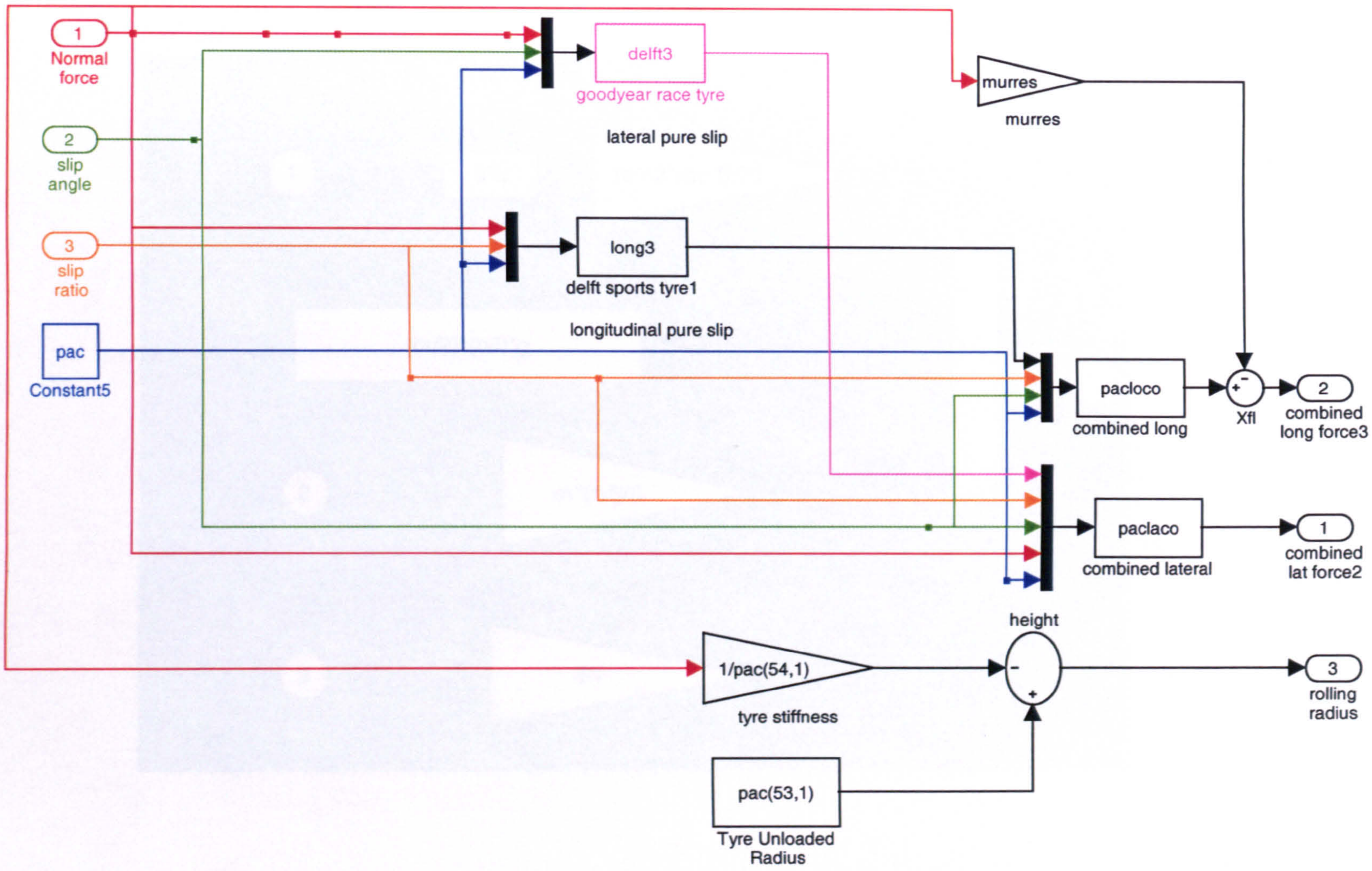
This appendix contains the 3 DOF Simulink model. The model is graphical based and is given in order or sub-system hierarchy, starting with the system overview. The colours do not have any particular meaning, but merely indicate groups of lines emanating from the same source.

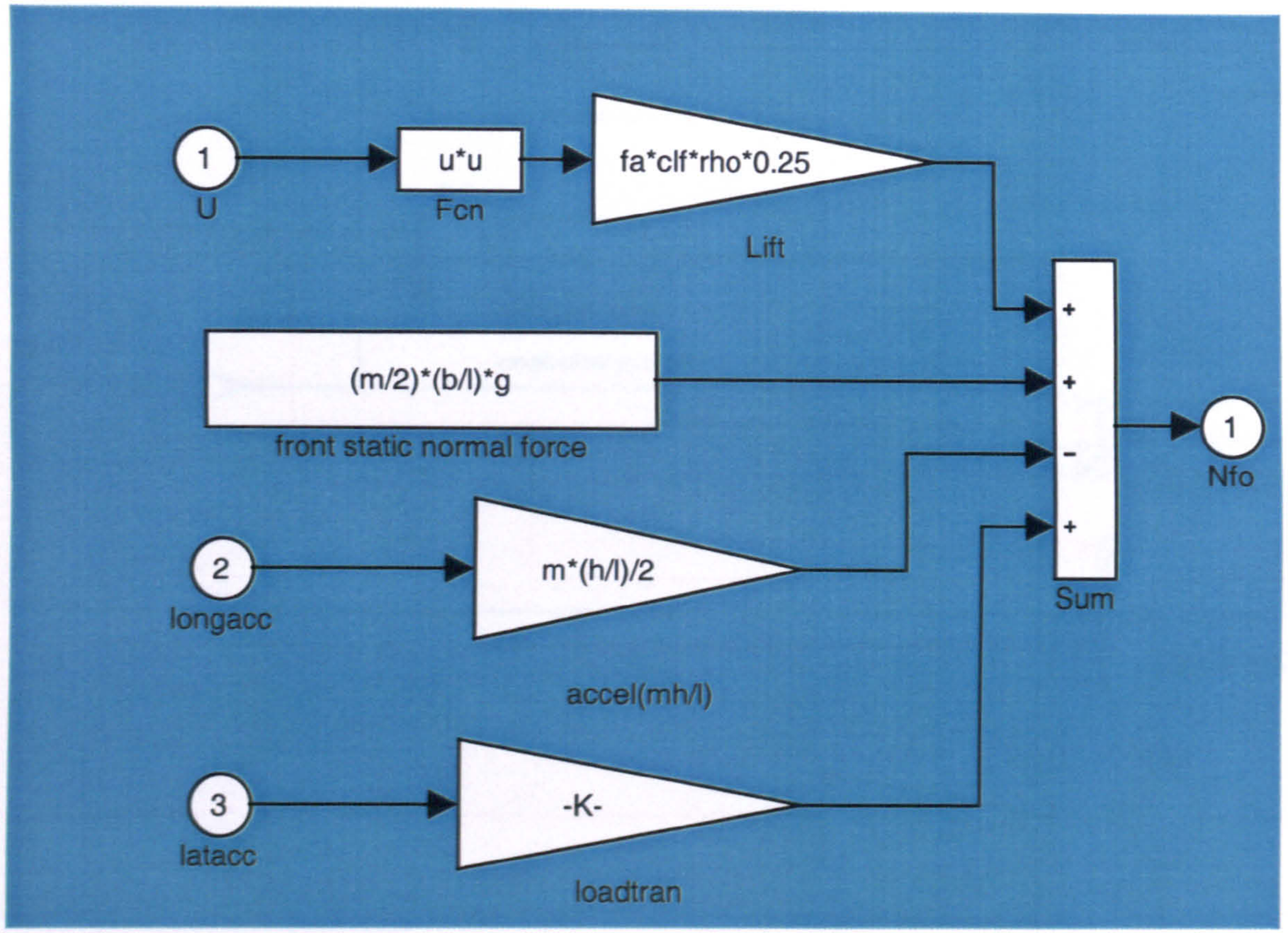


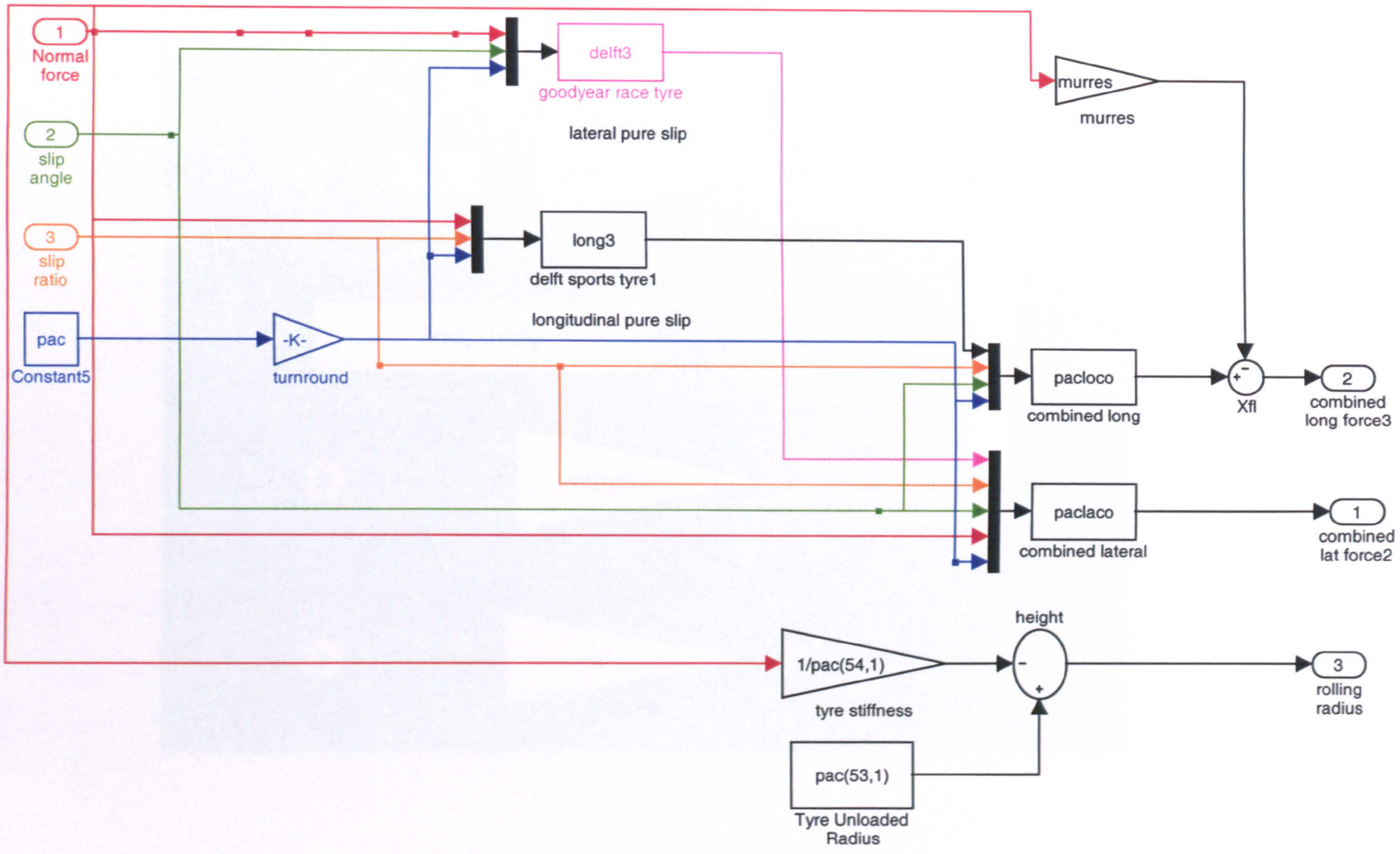
I:\Matlab\9dof\dof7ver8.mdl

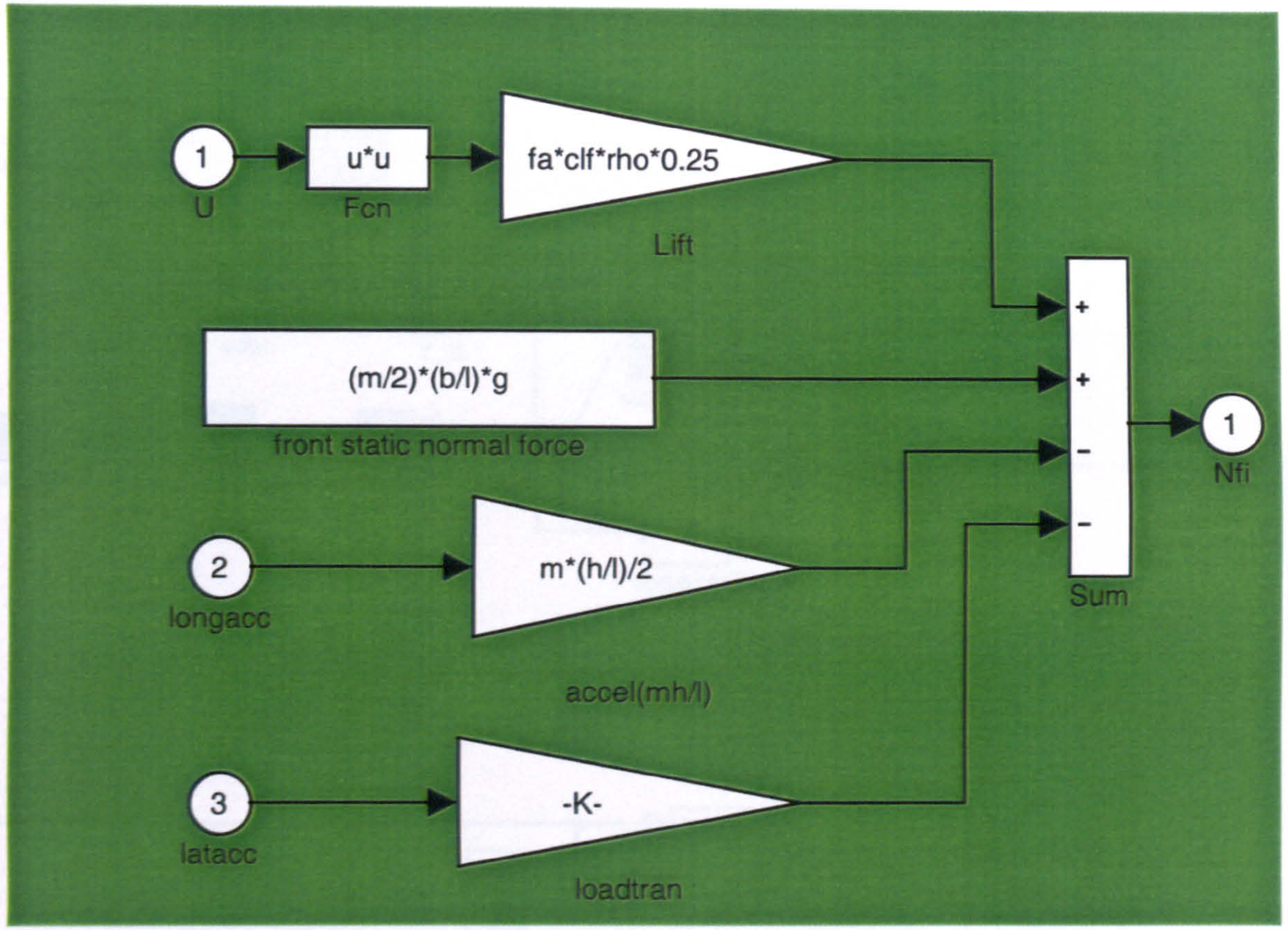

Ntr
Work
Nfl
Work1
Nrl
Work2
Nrl
Work3
Memory17
Memory16
Memory18
Memory19
Forces
Mz
N1
Y
X
longacc
To Workspace
f(u)
longitudinal acceleration of body
latacc
To Workspace1
lateral acceleration of body
front right normal force
front left normal force
rear right normal force
rear left normal force
Normal force
slip angle
slip ratio
combined lat force2
combined long force3
rolling radius
front left
front right
rear left
rear right
Product4
Product5
Product6
Product7
Tfl
Tfr
Trl
Trr
rollrad fl
rollrad fr
rollrad rl
rollrad rr
delta rad
delta
slip fl
slip fr
slip rl
slip rr
mux
fl
fr
rl
rr
Ufl1
Ufr1
Url1
Urr1
delta
slip angles no lag
U fl
U fr
U rl
U rr
Stop1
Switch2
Abs
udot
Product8
Memory15
brakeoff
cot1
lateral
vdot
Memory14
yaw
rdot
Memory13
zero
const1
Off the Ground
Stop2
yawacc
To Workspace2
yawvel
To Workspace3
vdot
To Workspace5
position
v
To Workspace4
u
To Workspace6
u1
Stop
Switch1
const
r
v
u

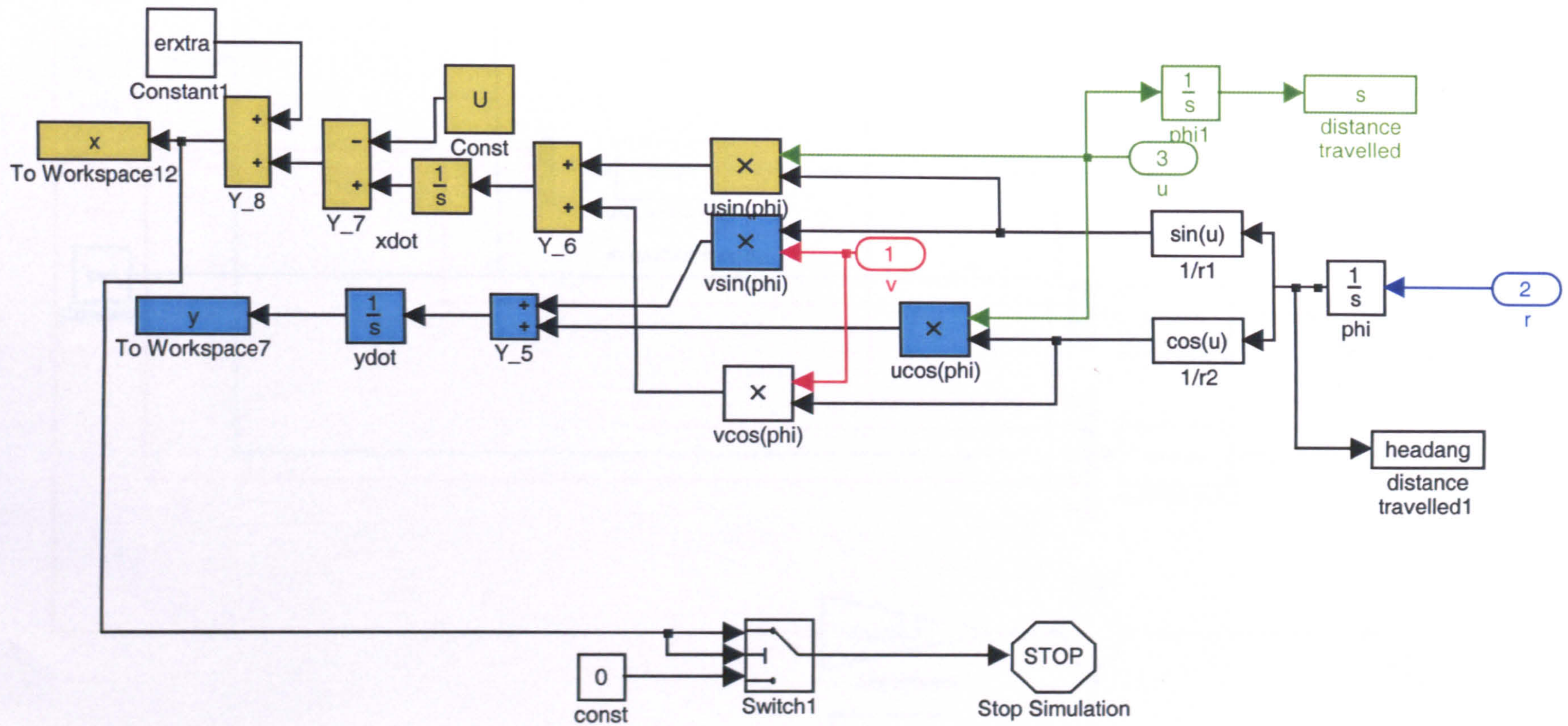

2
Yfl
4
Yfr
1
Delta
cos
cos delta
sin
sin delta
3
Xfl
5
Xfr
Product
Abs4
front1
Sign
Product8
Product1
Abs5
front4
Sign1
Product9
front
6
Yrl
rear
8
Yrr
a
a_
b
b
N
1
N1
total Y
2
Y
Product2
Product6
Abs2
Product7
Abs3
front2
Product3
Product4
front3
Product5
7
Xrl
9
Xrr
X1
3
X
10
u
u*u
Fcn
-K-
drag1

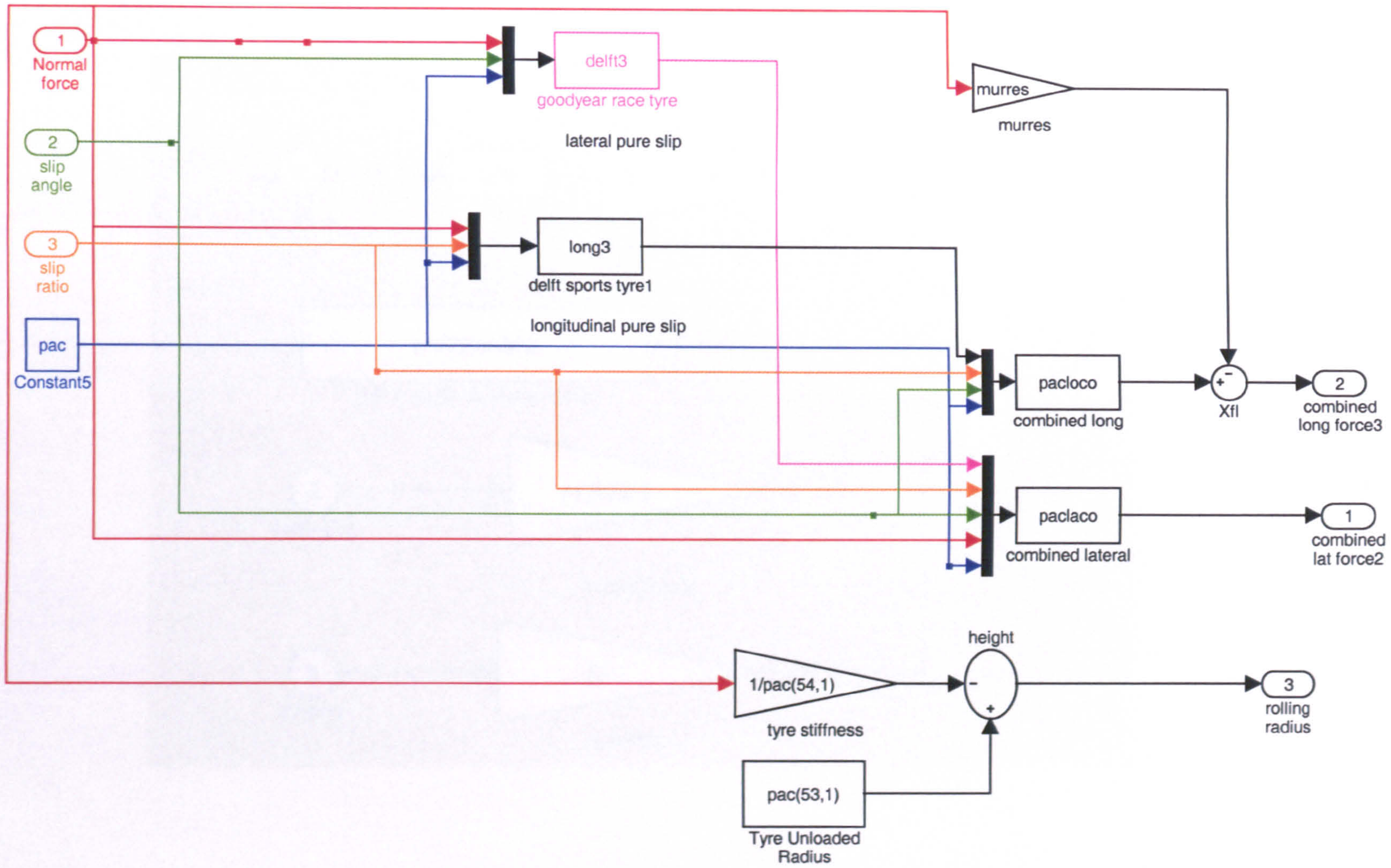

1
Normal force
2
slip angle
3
slip ratio
pac
Constant5
delft3
goodyear race tyre
lateral pure slip
long3
delft sports tyre1
longitudinal pure slip
murres
murres
pacloco
combined long
Xfl
2
combined long force3
paclaco
combined lateral
1
combined lat force2
height
1/pac(54,1)
tyre stiffness
3
rolling radius
pac(53,1)
Tyre Unloaded Radius

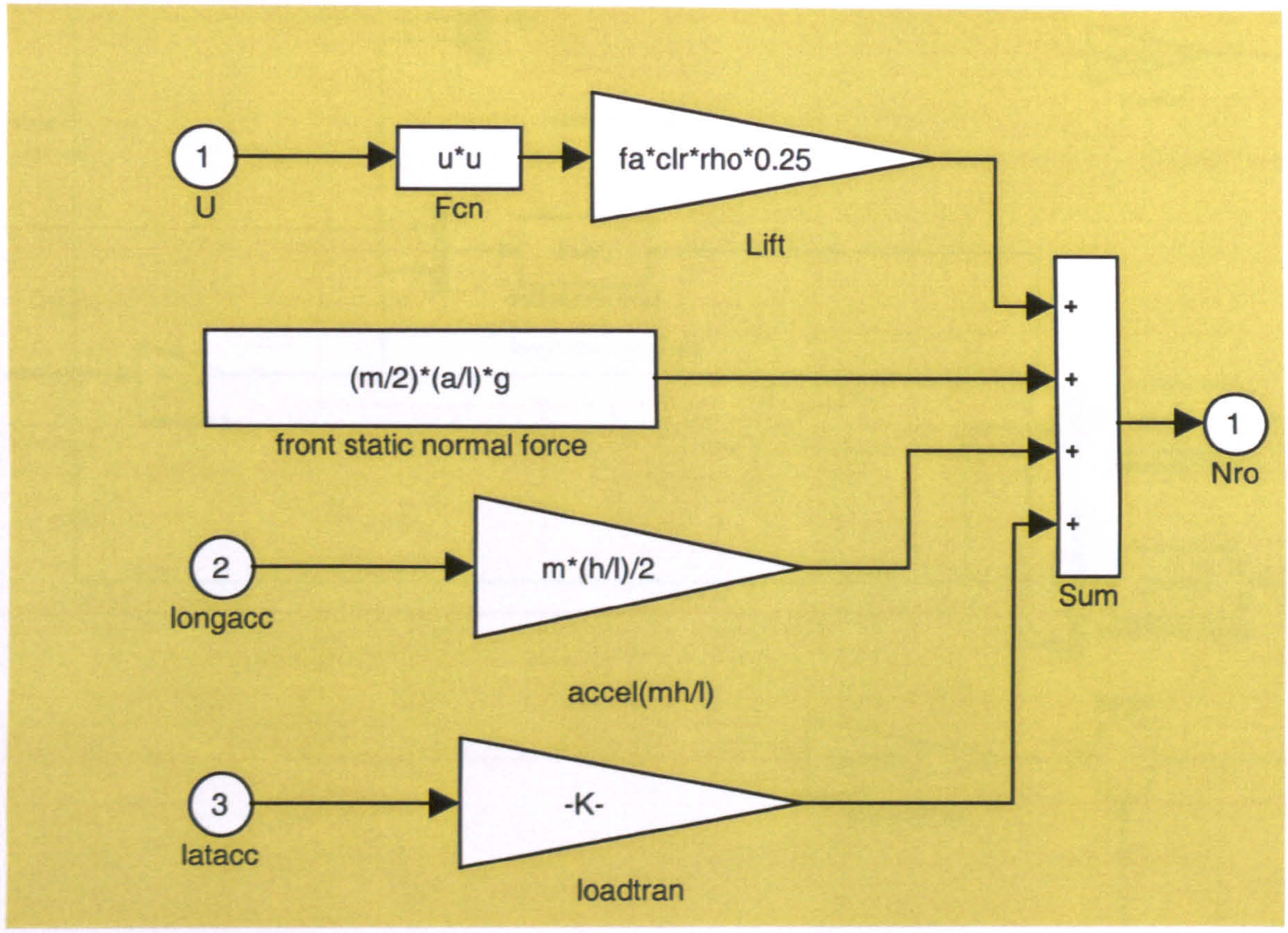

1
U
u*u
Fcn
fa*clf*rho*0.25
Lift
(m/2)*(b/l)*g
front static normal force
2
longacc
m*(h/l)/2
accel(mh/l)
3
latacc
-K-
loadtran
+
+
-
+
Sum
1
Nfo

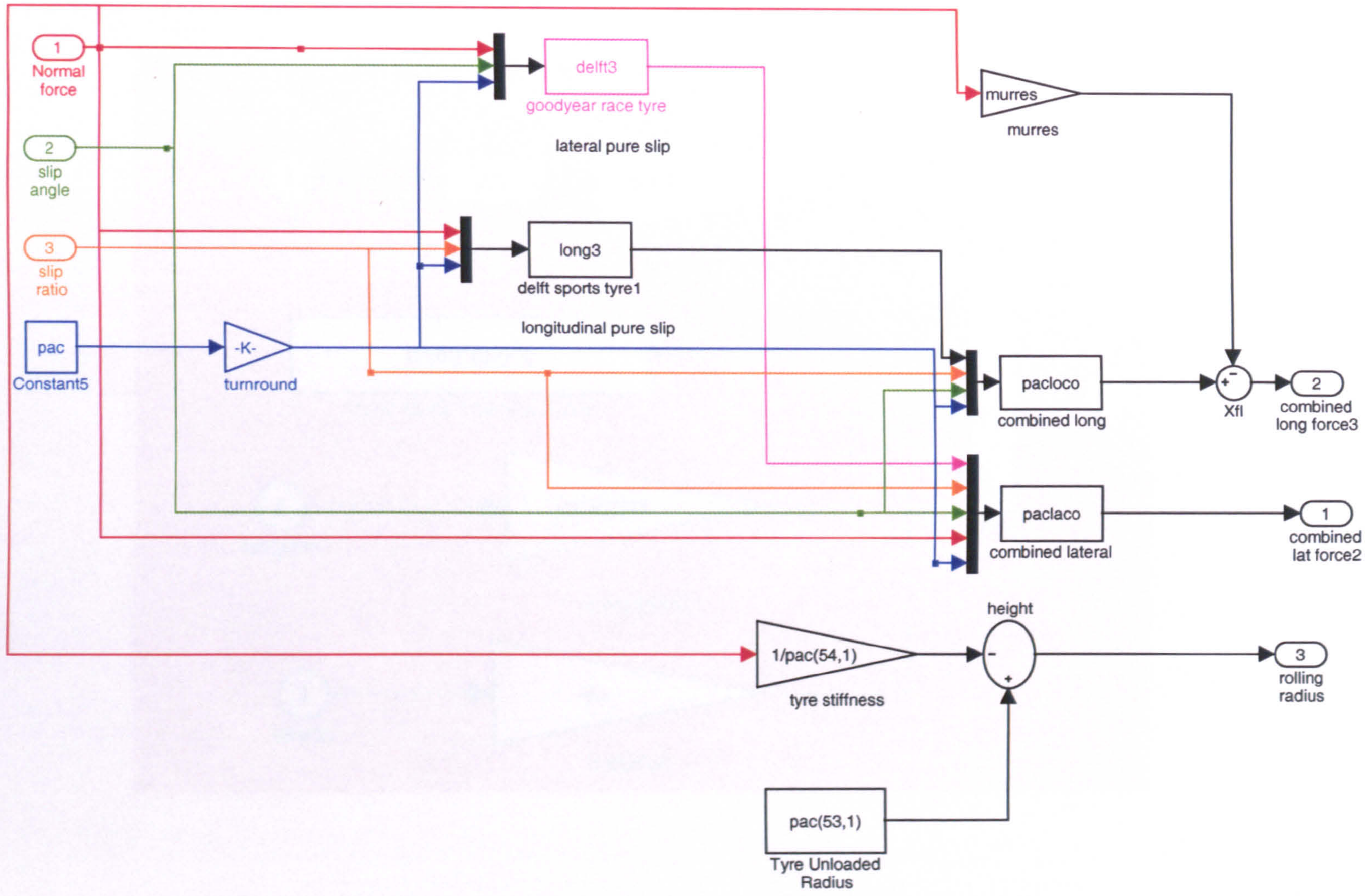

1
Normal force
2
slip angle
3
slip ratio
pac
Constant5
-K-
turnround
delft3
goodyear race tyre
lateral pure slip
long3
delft sports tyre1
longitudinal pure slip
murres
murres
pacloco
combined long
Xfl
2
combined long force3
paclaco
combined lateral
1
combined lat force2
height
1/pac(54,1)
tyre stiffness
pac(53,1)
Tyre Unloaded Radius
3
rolling radius

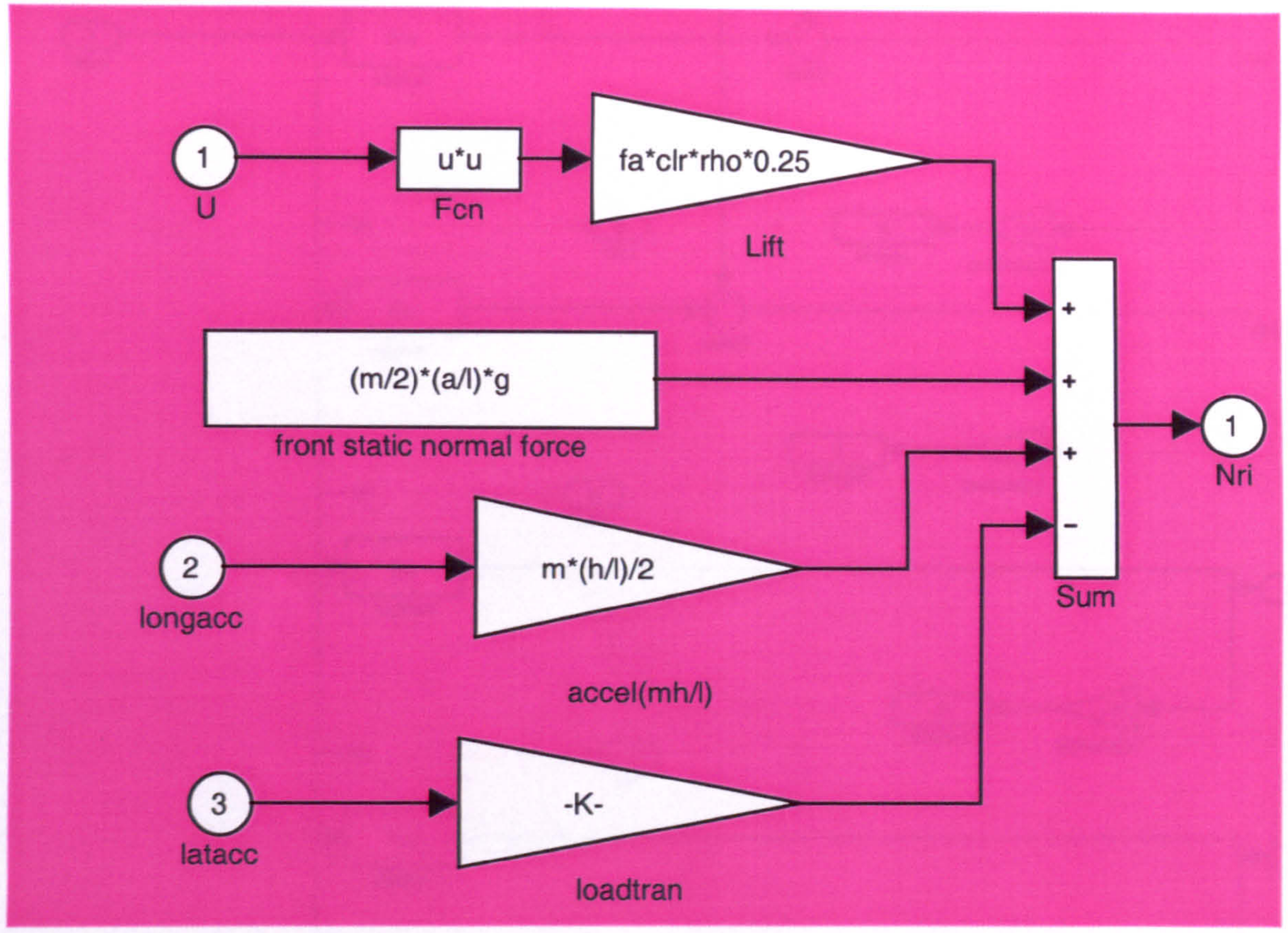

1
U
u*u
Fcn
fa*clf*rho*0.25
Lift
(m/2)*(b/l)*g
front static normal force
2
longacc
m*(h/l)/2
accel(mh/l)
3
latacc
-K-
loadtran
+
+
-
-
Sum
1
Nfi

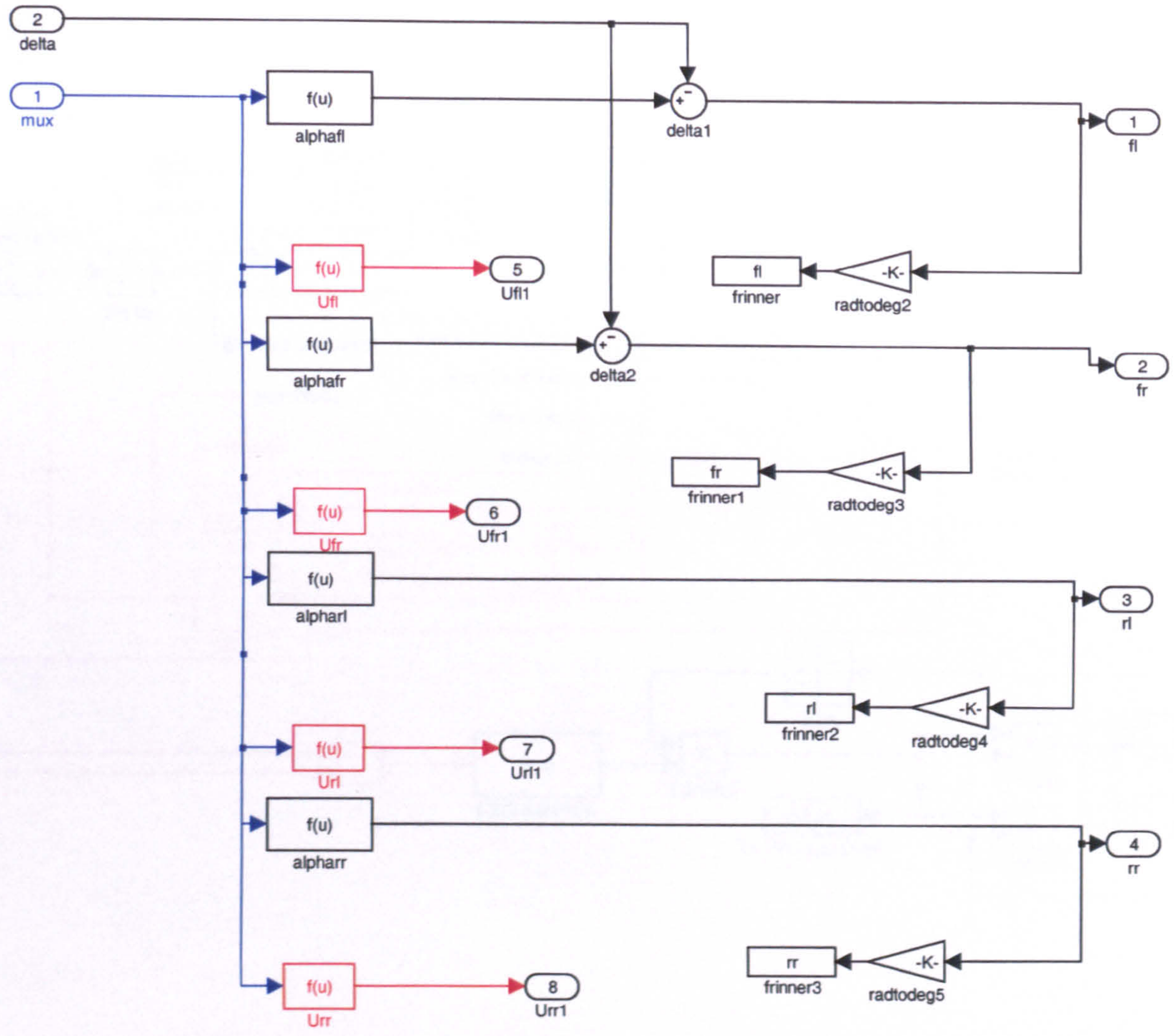

erxtra
Constant1
x
To Workspace12
Y_8
Y_7
U
Const
1/s
xdot
Y_6
usin(phi)
vsin(phi)
vcos(phi)
ucos(phi)
1
v
y
To Workspace7
1/s
ydot
Y_5
1/s
phi1
s
distance travelled
3
u
sin(u)
1/r1
cos(u)
1/r2
1/s
phi
2
r
headang
distance travelled1
0
const
Switch1
STOP
Stop Simulation

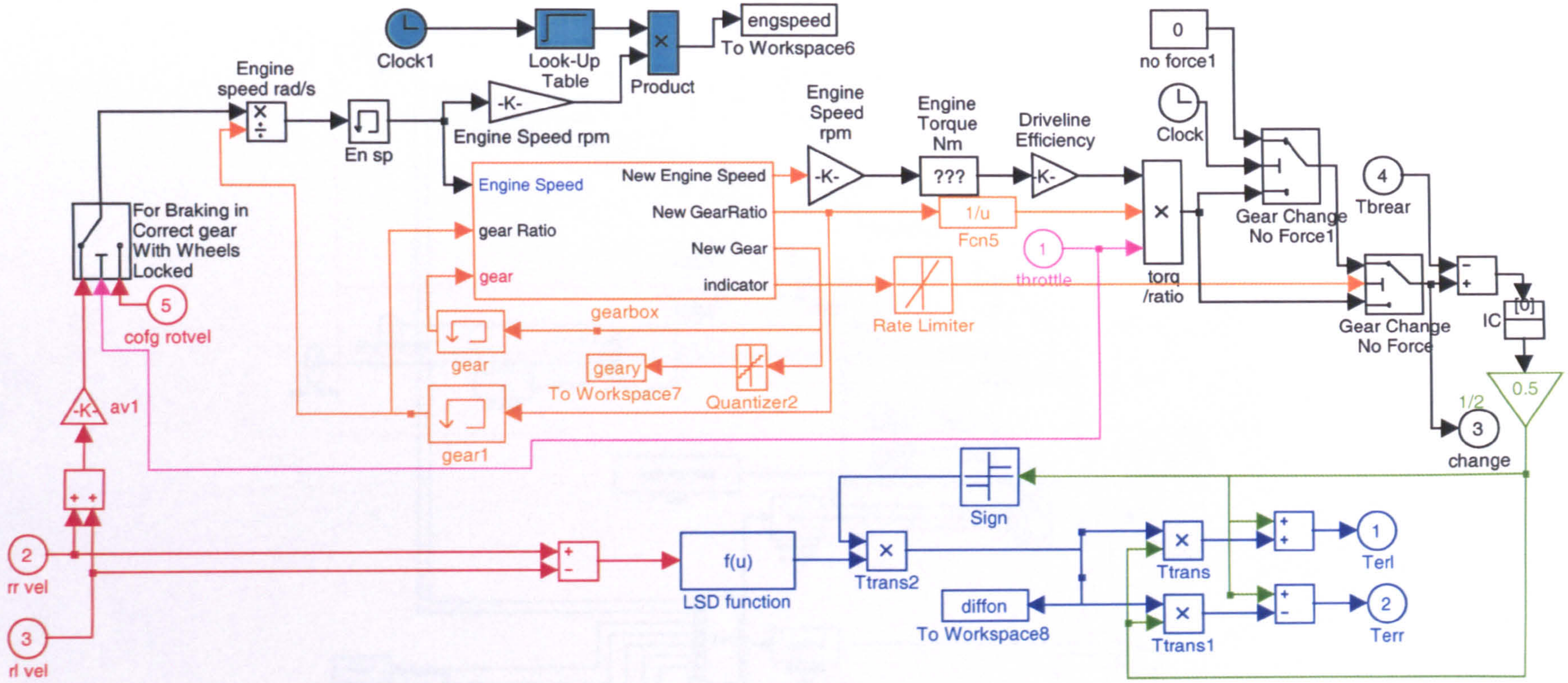

1
Normal force
2
slip angle
3
slip ratio
pac
Constant5
delft3
goodyear race tyre
lateral pure slip
long3
delft sports tyre1
longitudinal pure slip
murres
murres
pacloco
combined long
Xfl
2
combined long force3
paclaco
combined lateral
1
combined lat force2
height
1/pac(54,1)
tyre stiffness
pac(53,1)
Tyre Unloaded Radius
3
rolling radius

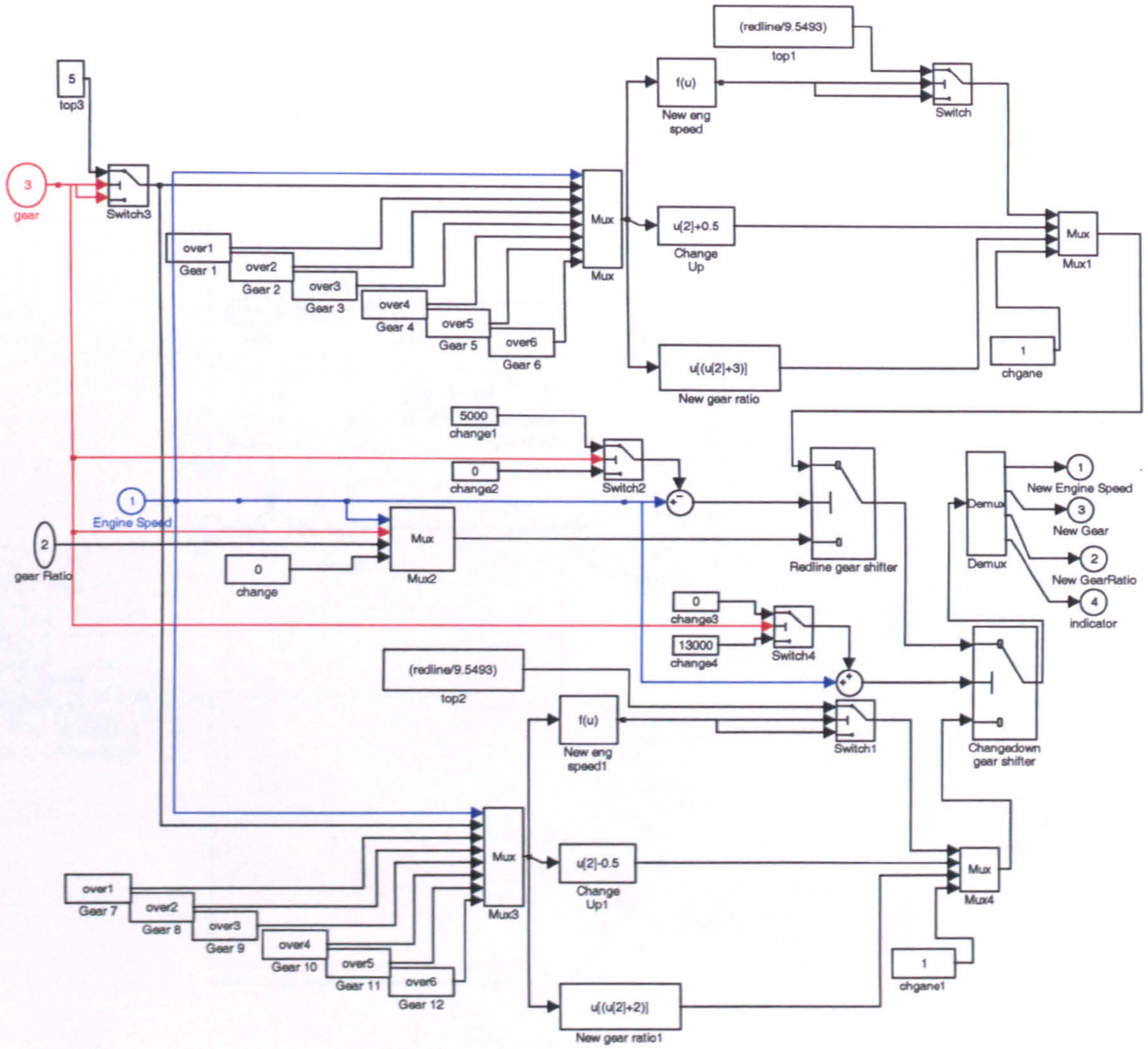

1
U
u*u
Fcn
fa*clr*rho*0.25
Lift
(m/2)*(a/l)*g
front static normal force
2
longacc
m*(h/l)/2
accel(mh/l)
3
latacc
-K-
loadtran
+
+
+
+
Sum
1
Nro

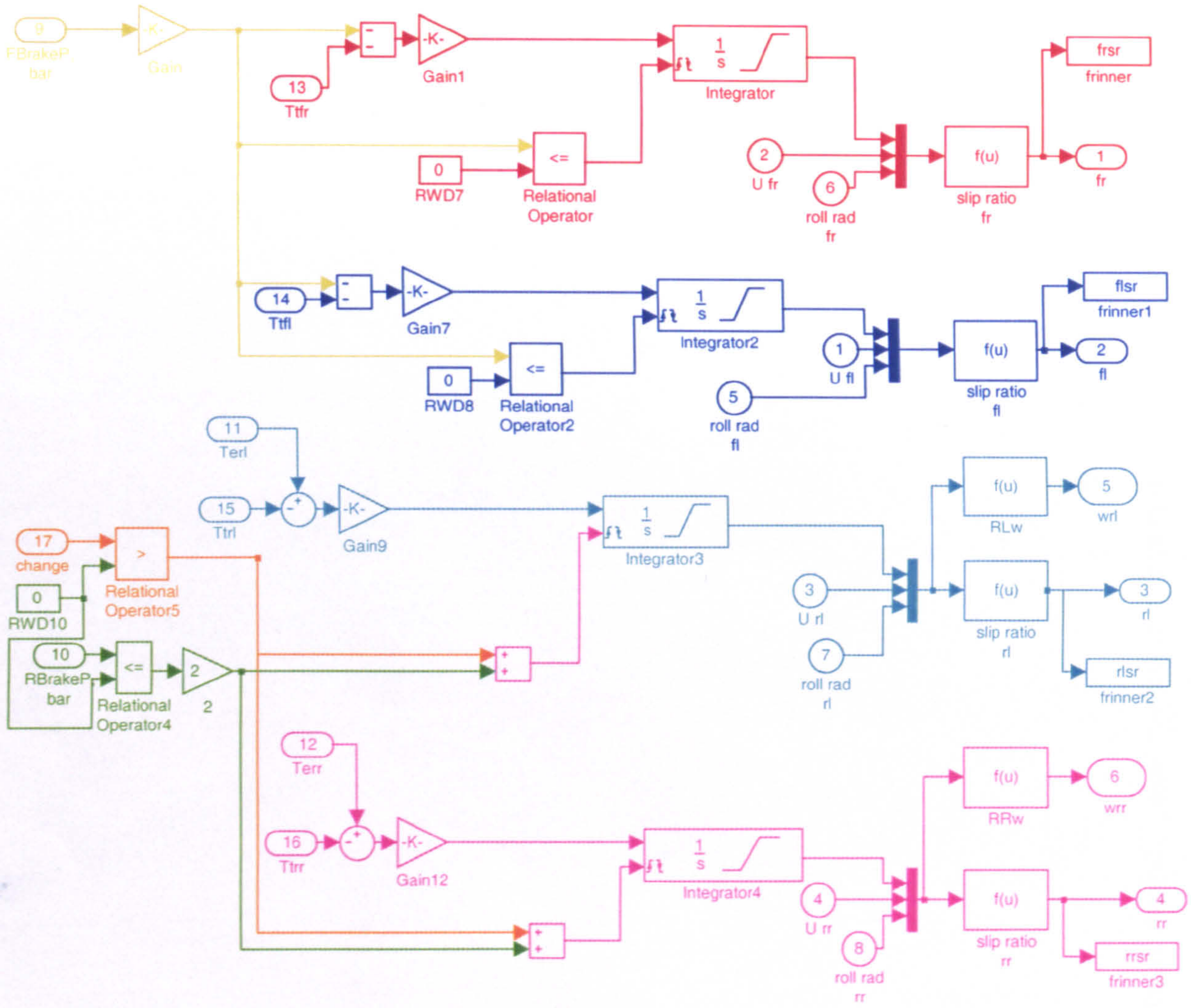

1
Normal force
2
slip angle
3
slip ratio
pac
Constant5
-K-
turnround
delft3
goodyear race tyre
lateral pure slip
long3
delft sports tyre1
longitudinal pure slip
murres
murres
pacloco
combined long
Xfl
2
combined long force3
paclaco
combined lateral
1
combined lat force2
height
1/pac(54,1)
tyre stiffness
3
rolling radius
pac(53,1)
Tyre Unloaded Radius


1
U
u*u
Fcn
fa*clr*rho*0.25
Lift
(m/2)*(a/l)*g
front static normal force
2
longacc
m*(h/l)/2
accel(mh/l)
3
latacc
-K-
loadtran
+
+
+
-
Sum
1
Nri


2
delta
1
mux
f(u)
alphafl
delta1
1
fl
f(u)
Ufl
5
Ufl1
fl
frinner
-K-
radtodeg2
f(u)
alphafr
delta2
2
fr
fr
frinner1
-K-
radtodeg3
f(u)
Ufr
6
Ufr1
f(u)
alpharl
3
rl
rl
frinner2
-K-
radtodeg4
f(u)
Url
7
Url1
f(u)
alpharr
4
rr
rr
frinner3
-K-
radtodeg5
f(u)
Urr
8
Urr1

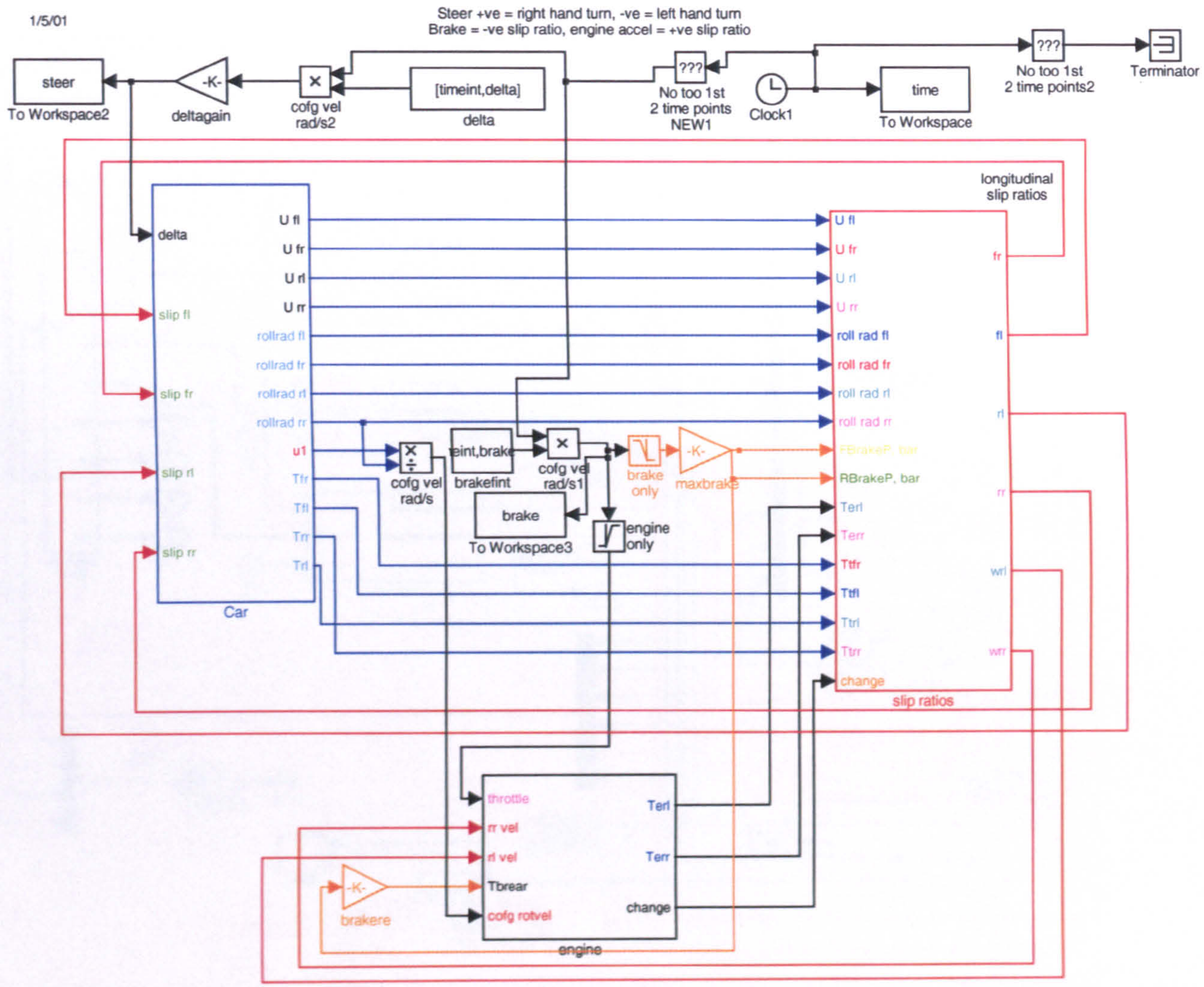

Clock1
Look-Up Table
Product
engspeed
To Workspace6
Engine speed rad/s
En sp
-K-
Engine Speed rpm
Engine Speed rpm
Engine Torque Nm
???
Driveline Efficiency
0
no force1
Clock
Engine Speed
gear Ratio
gear
New Engine Speed
New GearRatio
New Gear
indicator
1/u
Fcn5
1
throttle
torq /ratio
Gear Change No Force1
4
Tbrear
Gear Change No Force
IC
[0]
0.5
1/2
3
change
For Braking in Correct gear With Wheels Locked
5
cofg rotvel
gearbox
gear
geary
To Workspace7
Quantizer2
Rate Limiter
-K-
av1
gear1
Sign
2
rr vel
3
rl vel
f(u)
LSD function
Ttrans2
diffon
To Workspace8
Ttrans
Ttrans1
1
Terl
2
Terr

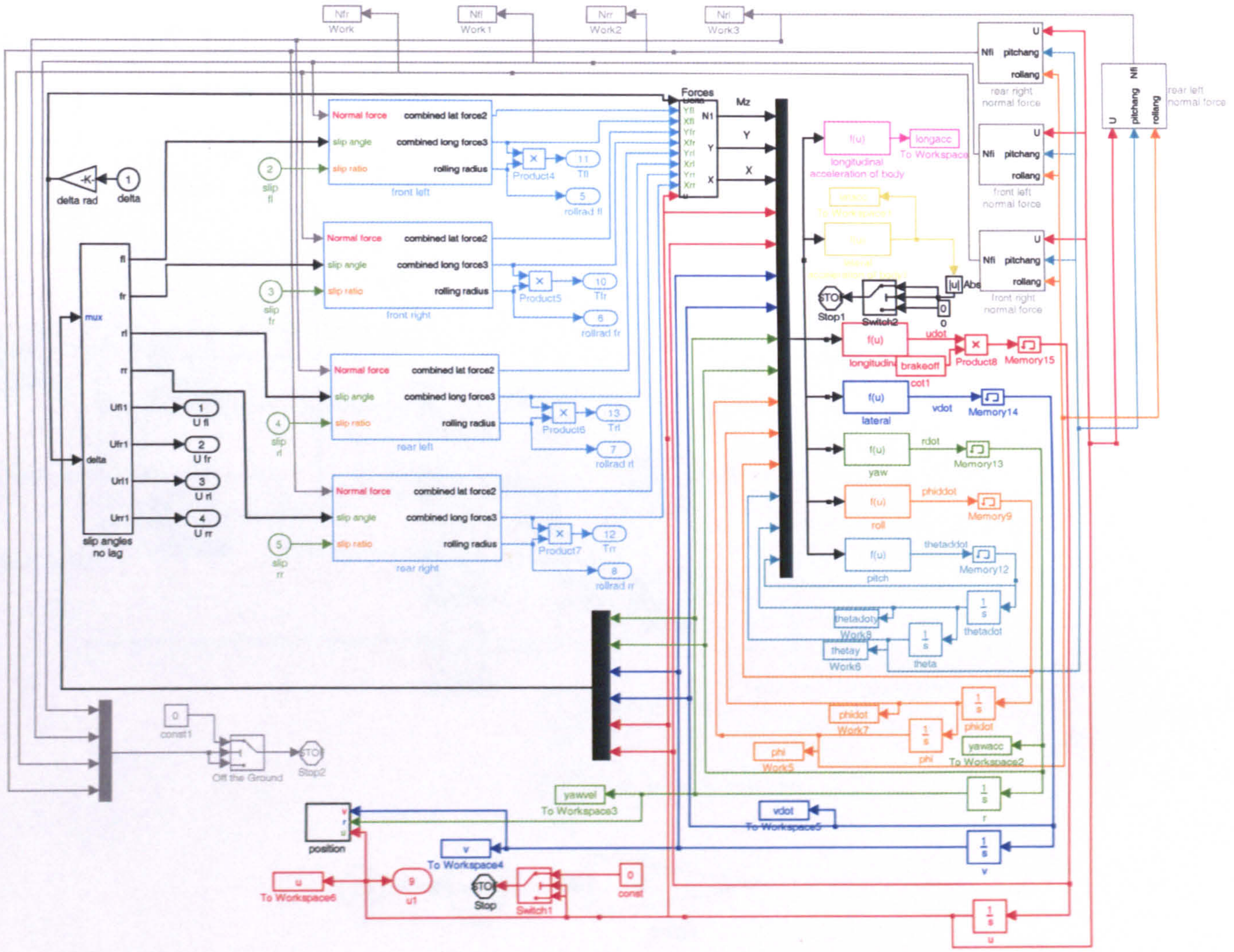

(redline/9.5493)
top1
5
top3
f(u)
New eng speed
Switch
3
gear
Switch3
Mux
over1
Gear 1
over2
Gear 2
over3
Gear 3
over4
Gear 4
over5
Gear 5
over6
Gear 6
Mux
u[2]+0.5
Change Up
Mux
Mux1
u[(u[2]+3)]
New gear ratio
1
chgane
5000
change1
0
change2
Switch2
1
Engine Speed
2
gear Ratio
Mux
Mux2
0
change
Redline gear shifter
Demux
1
New Engine Speed
3
New Gear
2
New GearRatio
4
indicator
Demux
0
change3
13000
change4
Switch4
(redline/9.5493)
top2
f(u)
New eng speed1
Switch1
Changedown gear shifter
Mux
Mux3
over1
Gear 7
over2
Gear 8
over3
Gear 9
over4
Gear 10
over5
Gear 11
over6
Gear 12
u[2]-0.5
Change Up1
Mux
Mux4
1
chgane1
u[(u[2]+2)]
New gear ratio1

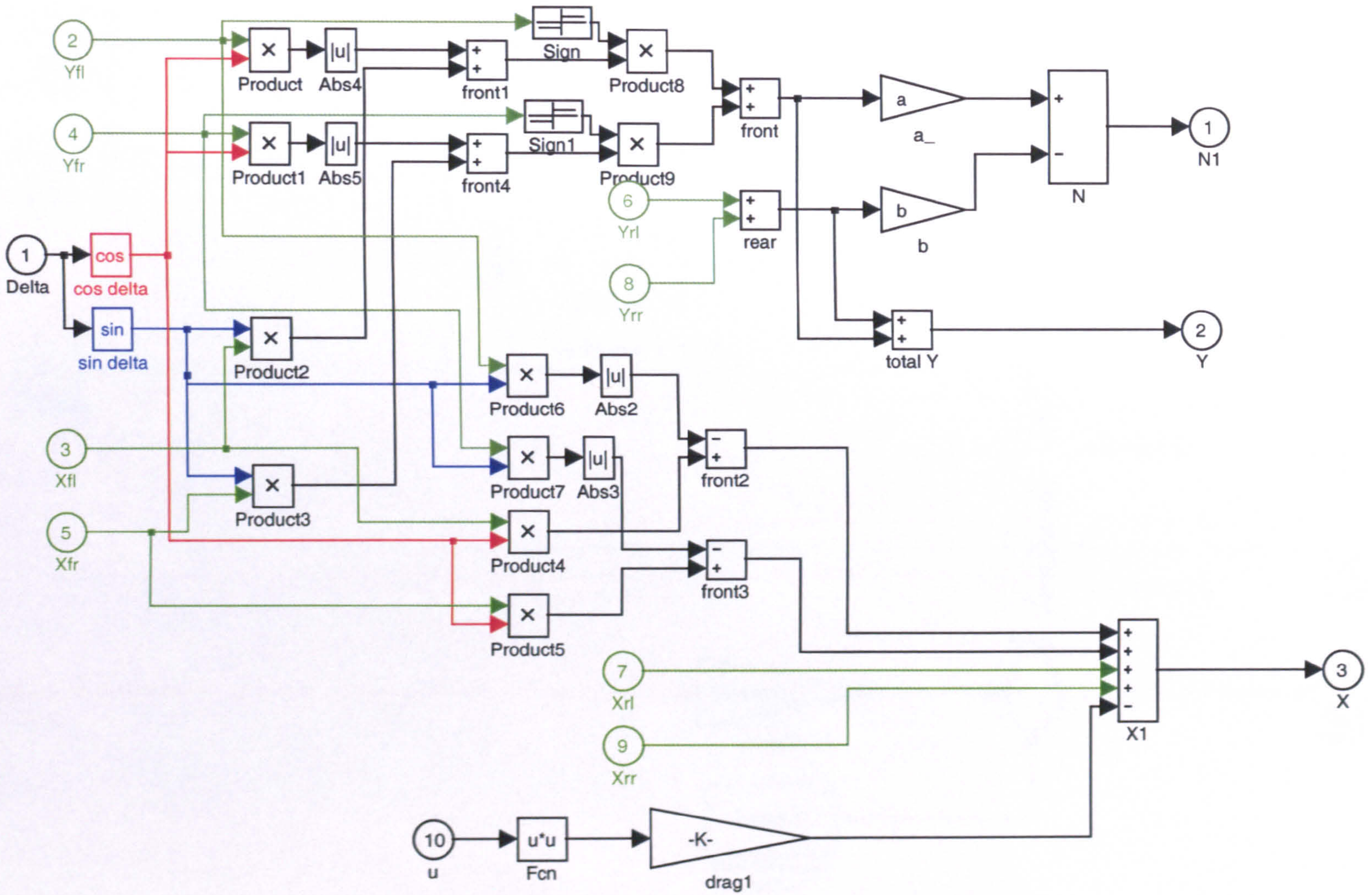

9
FBrakeP
bar
-K-
Gain
13
Ttfr
-K-
Gain1
1
s
Integrator
frsr
frinner
0
RWD7
<=
Relational
Operator
2
U fr
6
roll rad
fr
f(u)
slip ratio
fr
1
fr
14
Ttfl
-K-
Gain7
1
s
Integrator2
flsr
frinner1
0
RWD8
<=
Relational
Operator2
1
U fl
5
roll rad
fl
f(u)
slip ratio
fl
2
fl
11
Terl
15
Ttrl
-K-
Gain9
1
s
Integrator3
f(u)
RLw
5
wrl
17
change
>
Relational
Operator5
0
RWD10
10
RBrakeP
bar
<=
Relational
Operator4
2
2
3
U rl
7
roll rad
rl
f(u)
slip ratio
rl
3
rl
rlsr
frinner2
12
Terr
16
Ttrr
-K-
Gain12
1
s
Integrator4
f(u)
RRw
6
wrr
4
U rr
8
roll rad
rr
f(u)
slip ratio
rr
4
rr
rrsr
frinner3

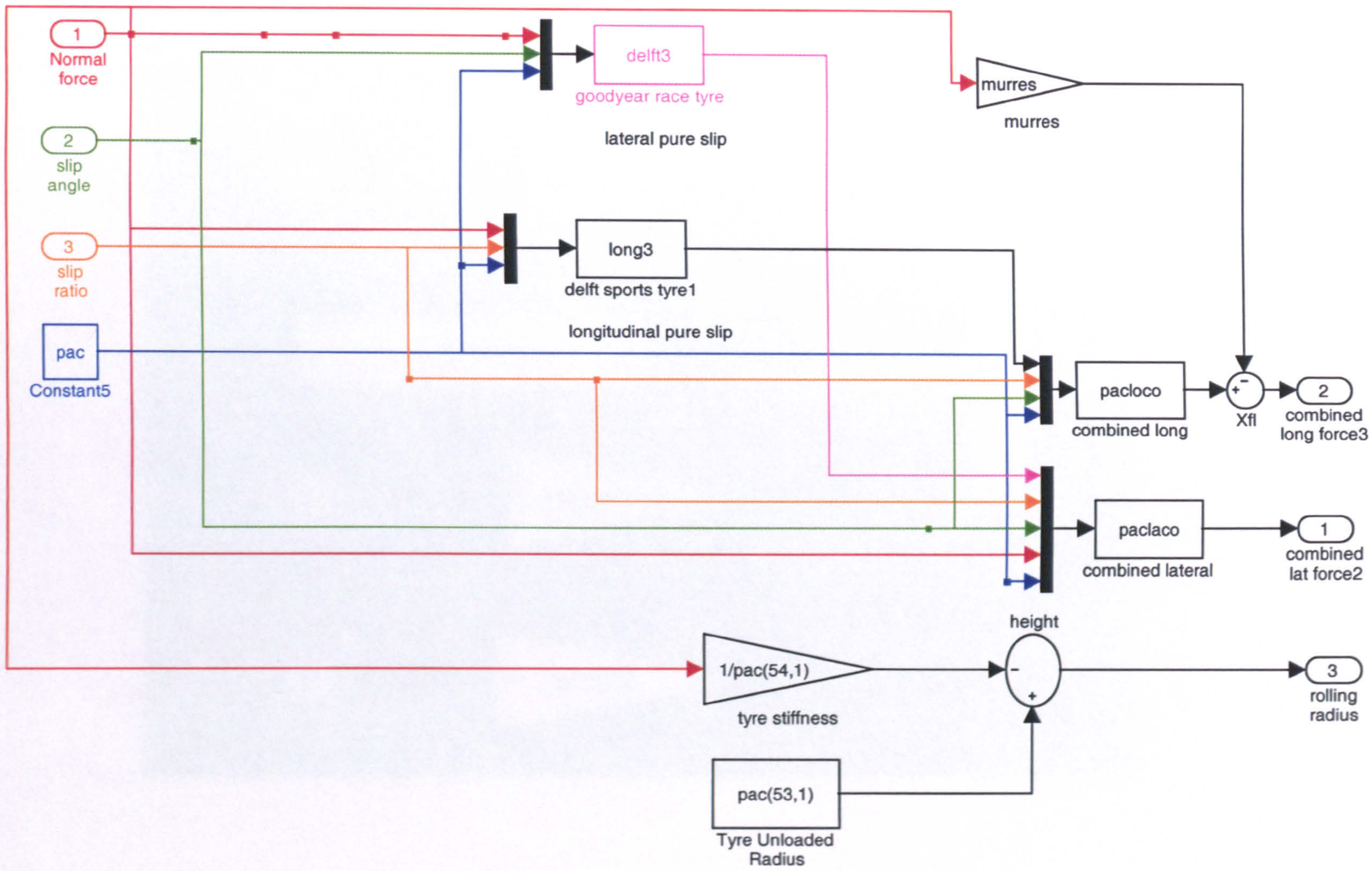
Appendix E

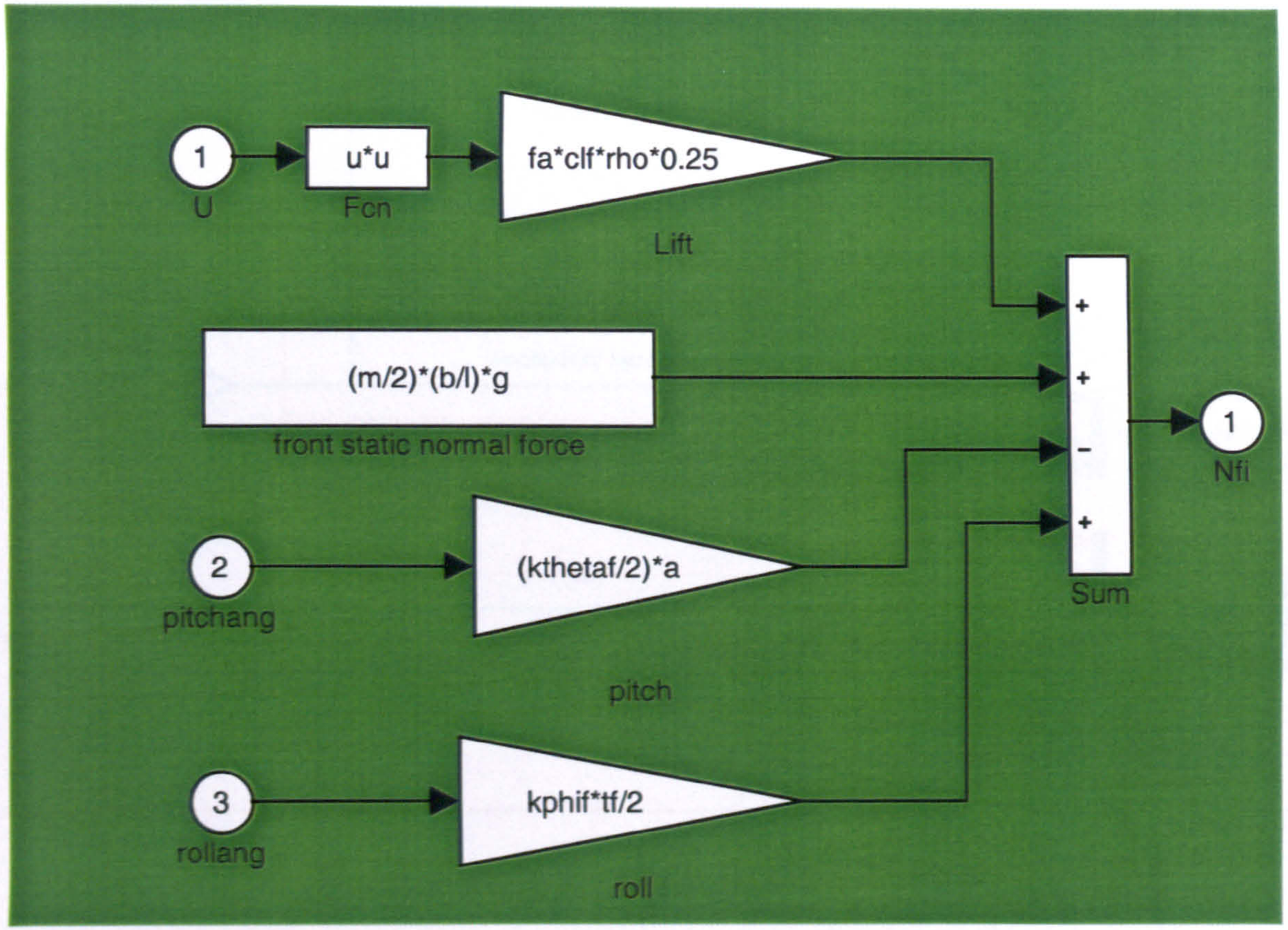
This appendix contains the 13 DOF Simulink model. The model is graphical based and is given in order or sub-system hierarchy, starting with the system overview. The colours do not have any particular meaning, but merely indicate groups of lines emanating from the same source.

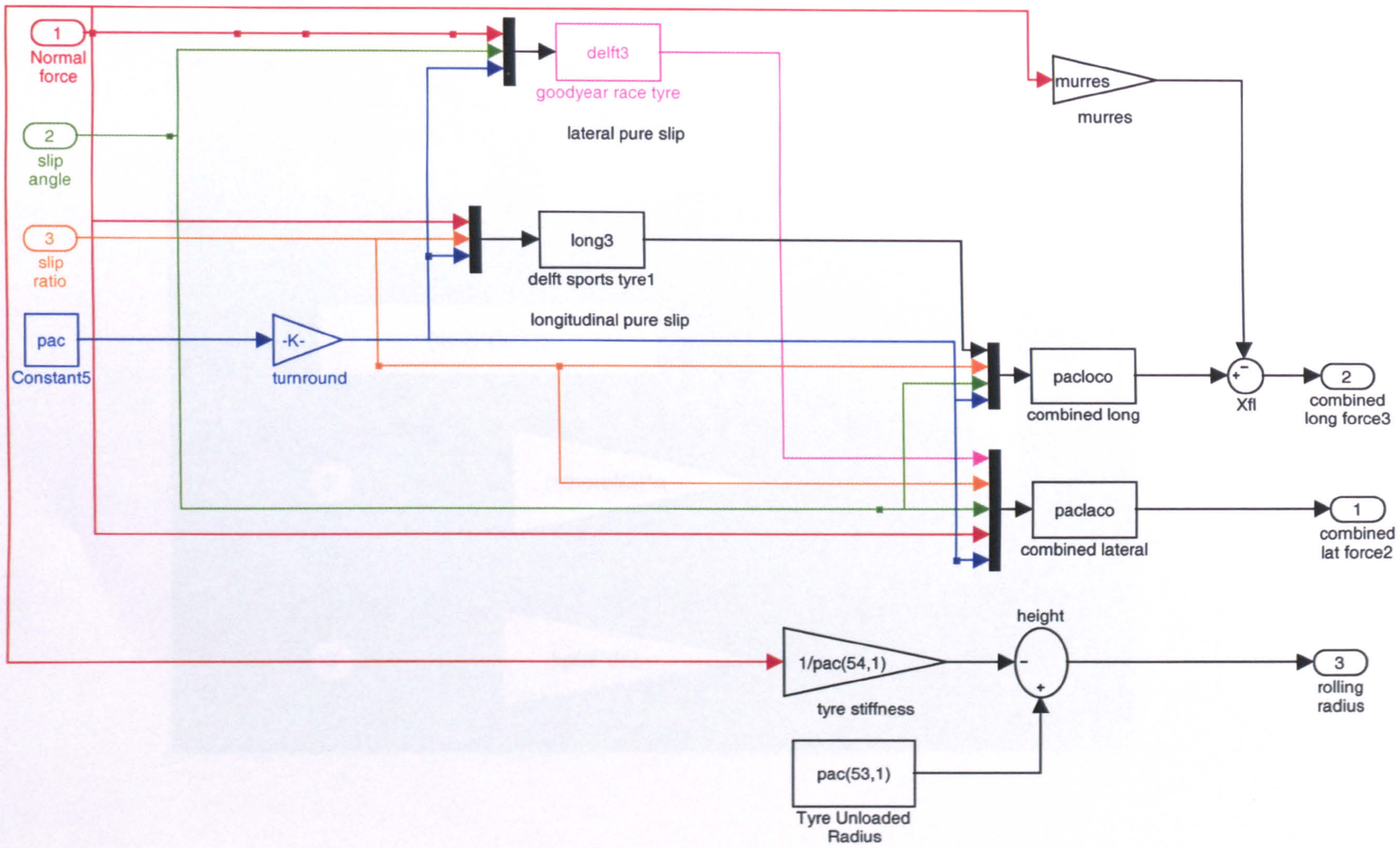

1/5/01
Steer +ve = right hand turn, -ve = left hand turn
Brake = -ve slip ratio, engine accel = +ve slip ratio
steer
To Workspace2
-K-
deltagain
cofg vel rad/s2
[timeint,delta]
delta
???
No too 1st 2 time points NEW1
Clock1
time
To Workspace
???
No too 1st 2 time points2
Terminator
longitudinal slip ratios
delta
slip fl
slip fr
slip rl
slip rr
U fl
U fr
U rl
U rr
rollrad fl
rollrad fr
rollrad rl
rollrad rr
u1
Tfr
Tfl
Trr
Trl
Car
cofg vel rad/s
timeint,brake
brakefint
cofg vel rad/s1
brake
To Workspace3
brake only
-K-
maxbrake
engine only
U fl
U fr
U rl
U rr
roll rad fl
roll rad fr
roll rad rl
roll rad rr
FBrakeP, bar
RBrakeP, bar
Terl
Terr
Ttfr
Ttfl
Ttrl
Ttrr
change
fr
fl
rl
rr
wrl
wrr
slip ratios
throttle
rr vel
rl vel
Tbrear
cofg rotvel
Terl
Terr
change
engine
-K-
brakere

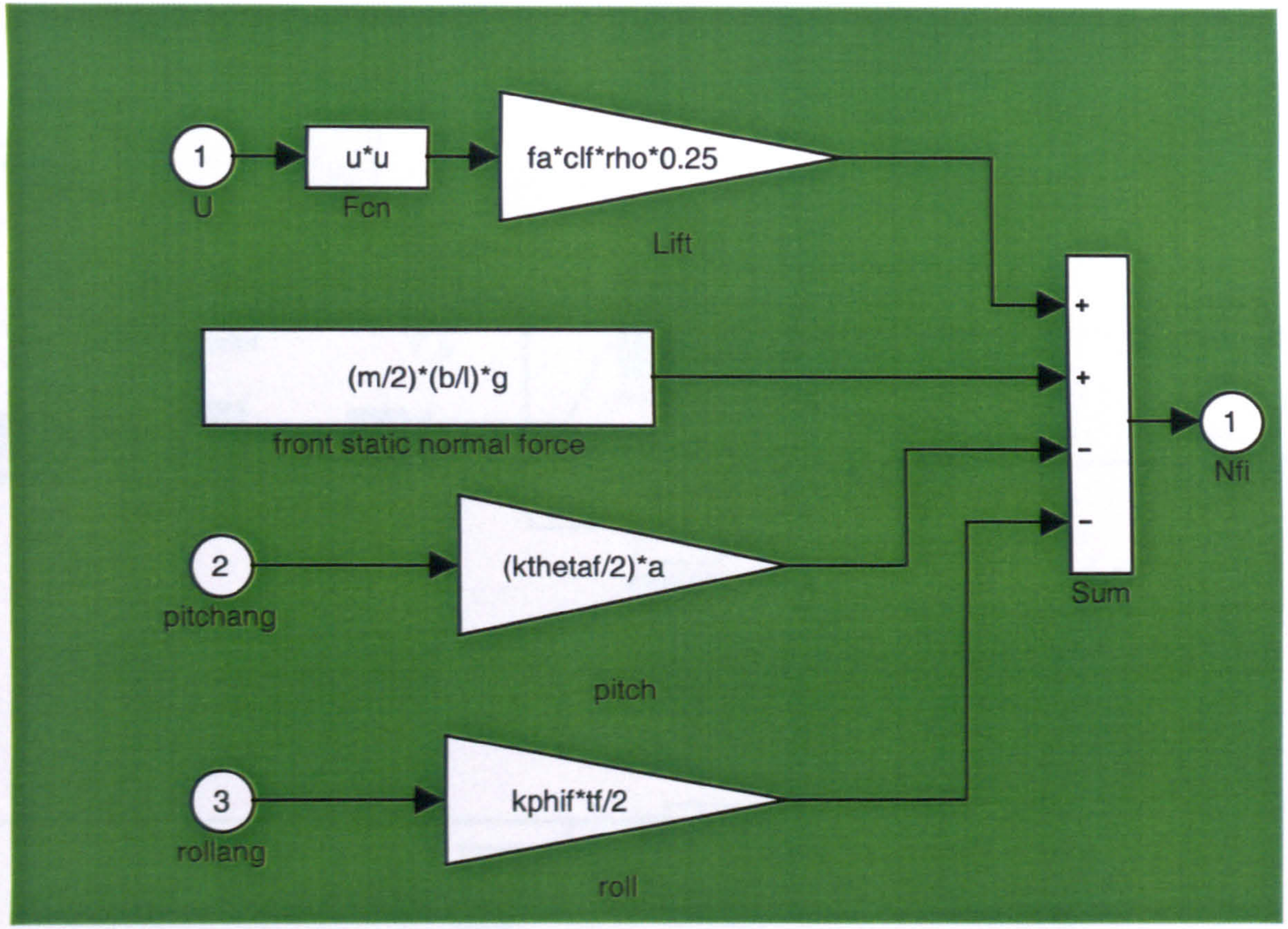

Nfr
Work
Nfl
Work1
Nrr
Work2
Nrl
Work3
Forces
Mz
N1
Y
X
delta rad
delta
slip fl
slip fr
slip rl
slip rr
Normal force
slip angle
slip ratio
combined lat force2
combined long force3
rolling radius
front left
front right
rear left
rear right
Product4
Product5
Product6
Product7
Tfl
Tfr
Trl
Trr
rollrad fl
rollrad fr
rollrad rl
rollrad rr
mux
fl
fr
rl
rr
delta
Ufl1
Ufr1
Url1
Urr1
slip angles no lag
U fl
U fr
U rl
U rr
U
Nfl
pitchang
rollang
rear right normal force
front left normal force
front right normal force
rear left normal force
f(u)
longitudinal acceleration of body
longacc
To Workspace
lateral
yaw
roll
pitch
Stop1
Switch2
Product8
Memory15
Memory14
Memory13
Memory9
Memory12
udot
vdot
rdot
phiddot
thetaddot
brakeoff
col1
thetadoty
Work8
thetay
Work6
theta
thetadot
phidot
Work7
phi
Work5
yawacc
To Workspace2
r
v
yawvel
To Workspace3
vdot
To Workspace5
const1
Off the Ground
Stop2
position
To Workspace4
To Workspace6
u1
Stop
Switch1
const

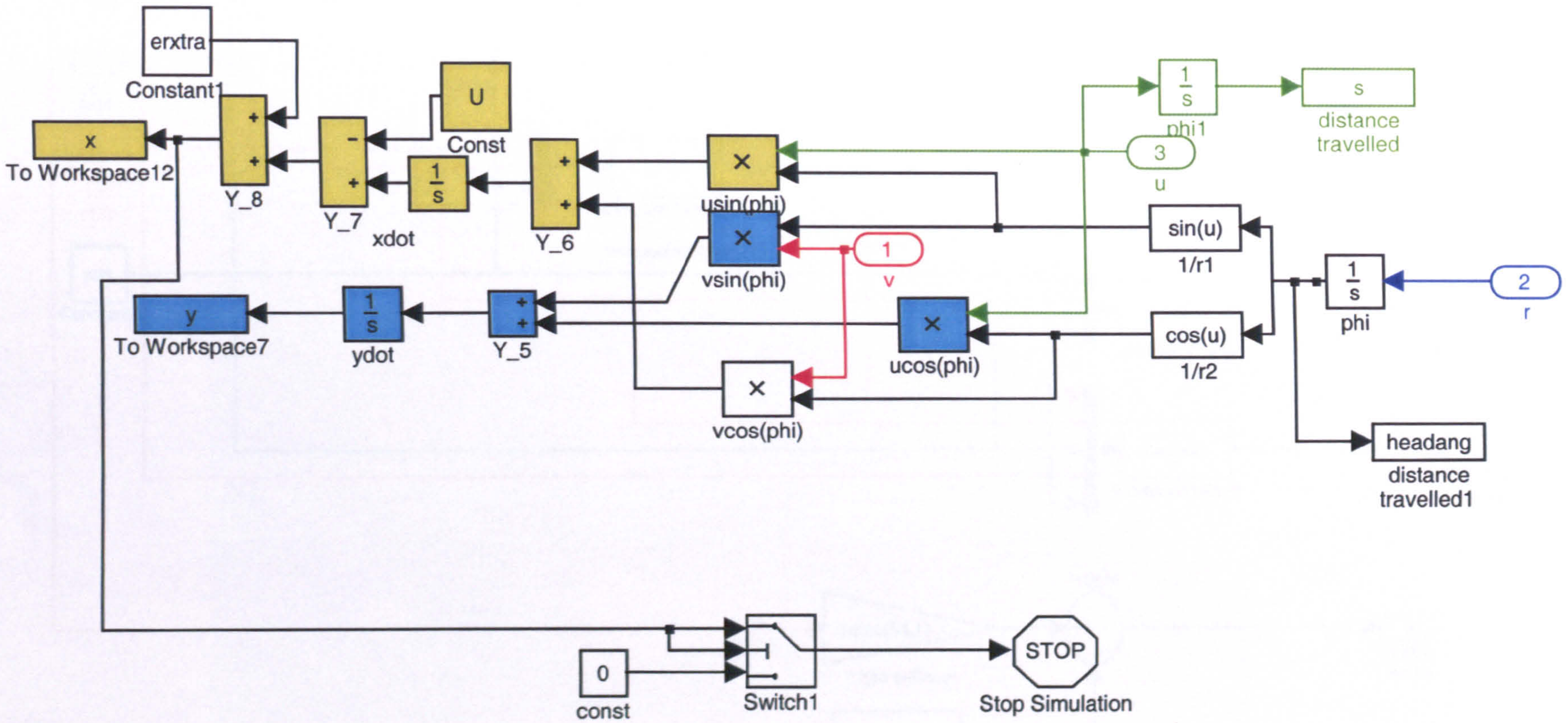

2
Yfl
4
Yfr
1
Delta
cos
cos delta
sin
sin delta
3
Xfl
5
Xfr
Product
Abs4
front1
Sign
Product8
Product1
Abs5
front4
Sign1
Product9
front
6
Yrl
8
Yrr
rear
a
a_
b
b
N
1
N1
total Y
2
Y
Product2
Product6
Abs2
Product3
Product7
Abs3
front2
Product4
front3
Product5
7
Xrl
9
Xrr
X1
3
X
10
u
u*u
Fcn
-K-
drag1

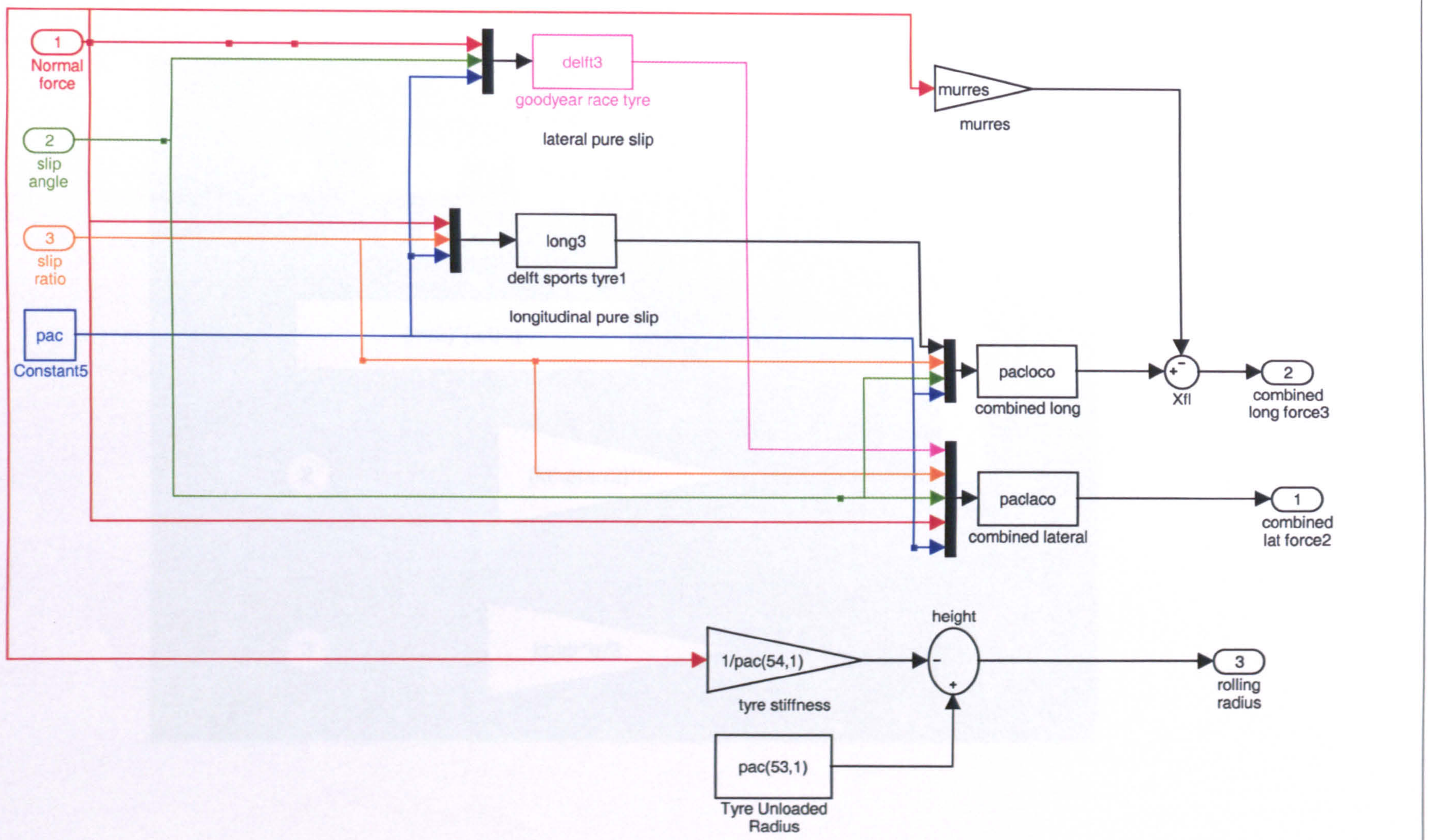

1
Normal force
2
slip angle
3
slip ratio
pac
Constant5
delft3
goodyear race tyre
lateral pure slip
long3
delft sports tyre1
longitudinal pure slip
murres
murres
pacloco
combined long
Xfl
2
combined long force3
paclaco
combined lateral
1
combined lat force2
height
1/pac(54,1)
tyre stiffness
3
rolling radius
pac(53,1)
Tyre Unloaded Radius

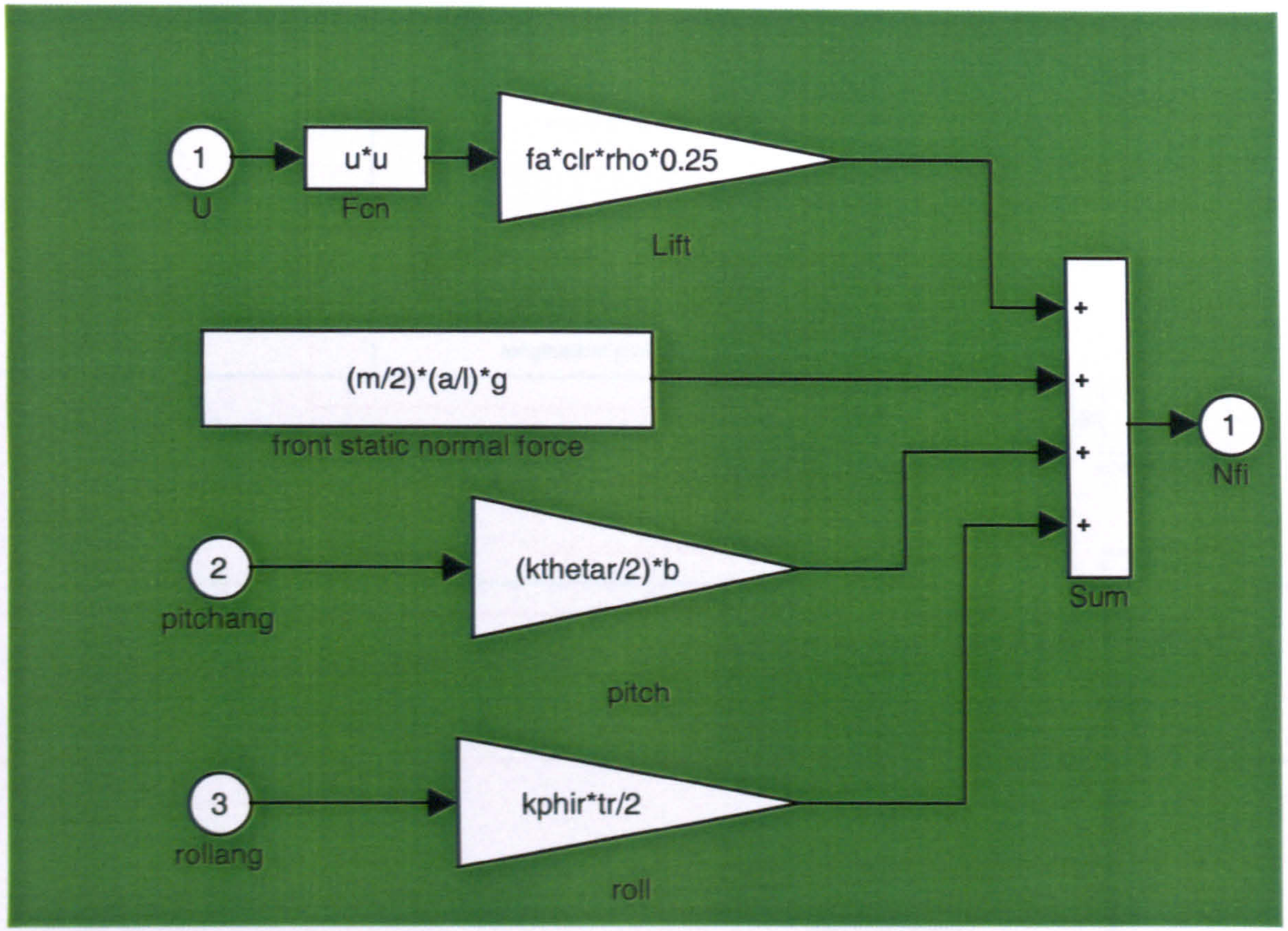

1
U
u*u
Fcn
fa*clf*rho*0.25
Lift
(m/2)*(b/l)*g
front static normal force
2
pitchang
(kthetaf/2)*a
pitch
3
rollang
kphif*tf/2
roll
+
+
-
+
1
Nfi
Sum

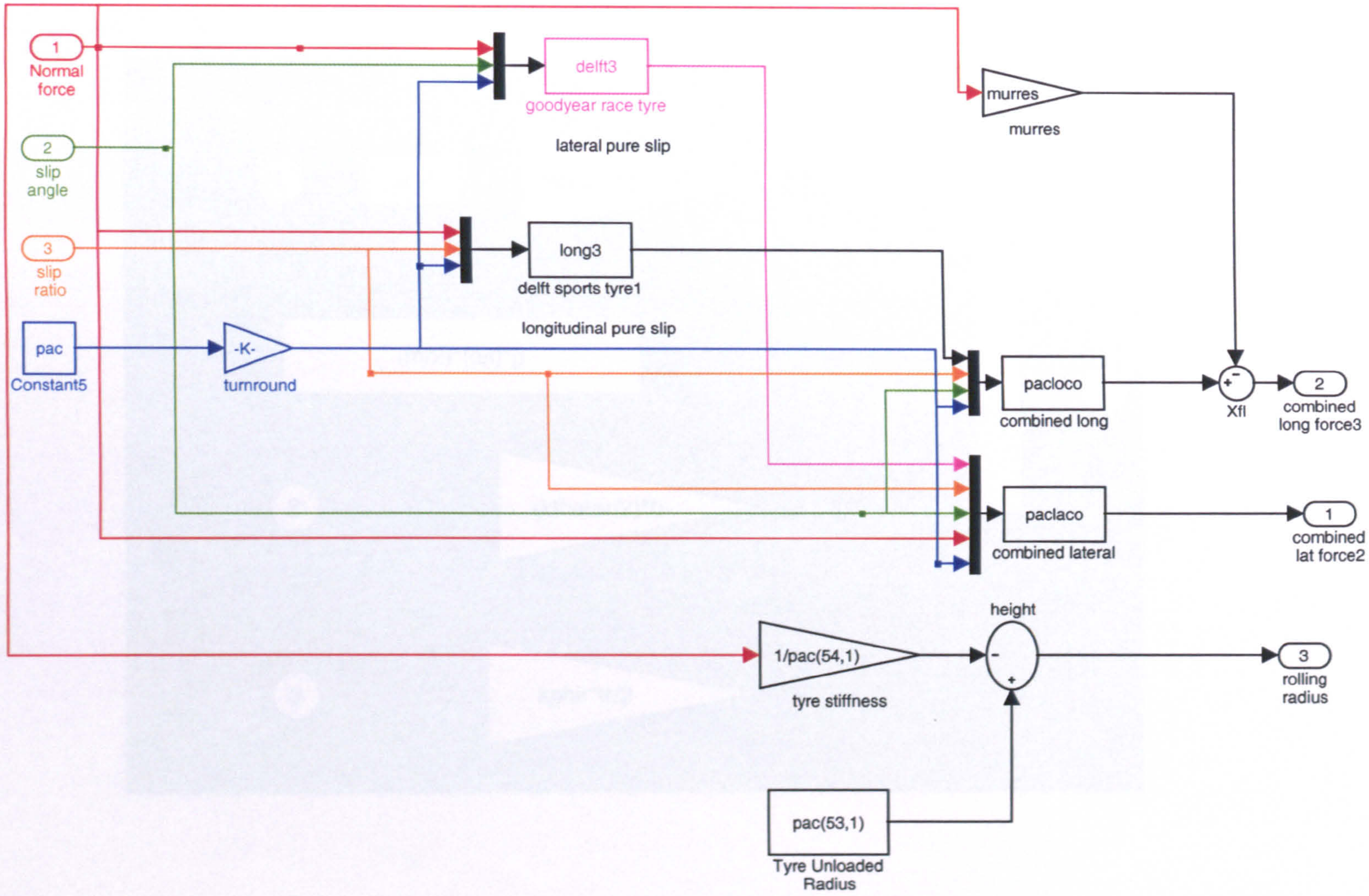

1
Normal force
2
slip angle
3
slip ratio
pac
Constant5
-K-
turnround
delft3
goodyear race tyre
lateral pure slip
long3
delft sports tyre1
longitudinal pure slip
murres
murres
pacloco
combined long
Xfl
2
combined long force3
paclaco
combined lateral
1
combined lat force2
height
1/pac(54,1)
tyre stiffness
3
rolling radius
pac(53,1)
Tyre Unloaded Radius

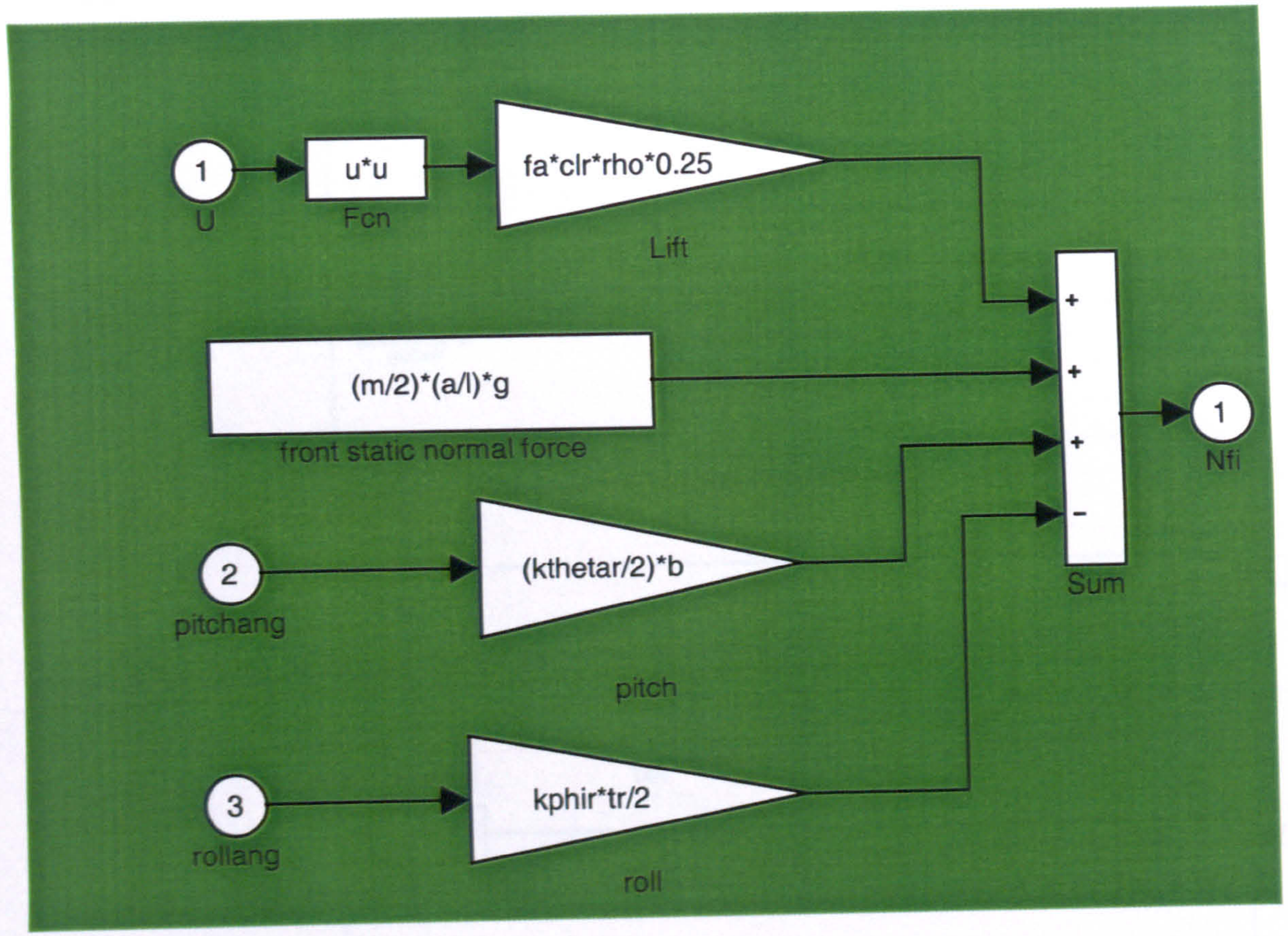

1
U
u*u
Fcn
fa*clf*rho*0.25
Lift
(m/2)*(b/l)*g
front static normal force
2
pitchang
(kthetaf/2)*a
pitch
3
rollang
kphif*tf/2
roll
+
+
-
-
1
Nfi
Sum

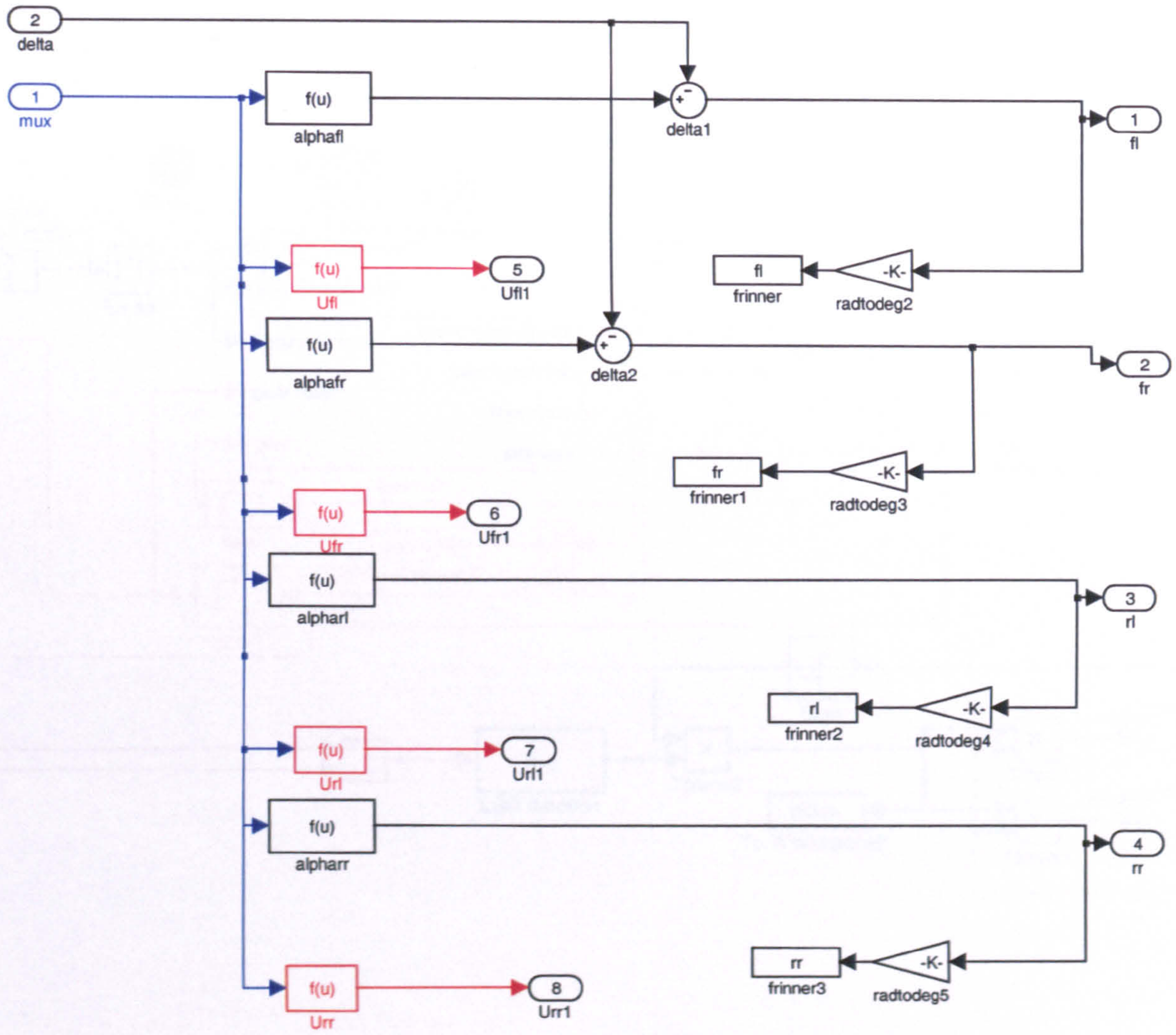

erxtra
Constant1
x
To Workspace12
Y_8
Y_7
U
Const
1/s
xdot
Y_6
usin(phi)
vsin(phi)
1
v
y
To Workspace7
1/s
ydot
Y_5
ucos(phi)
vcos(phi)
1/s
phi1
s
distance travelled
3
u
sin(u)
1/r1
cos(u)
1/r2
1/s
phi
2
r
headang
distance travelled1
0
const
Switch1
STOP
Stop Simulation

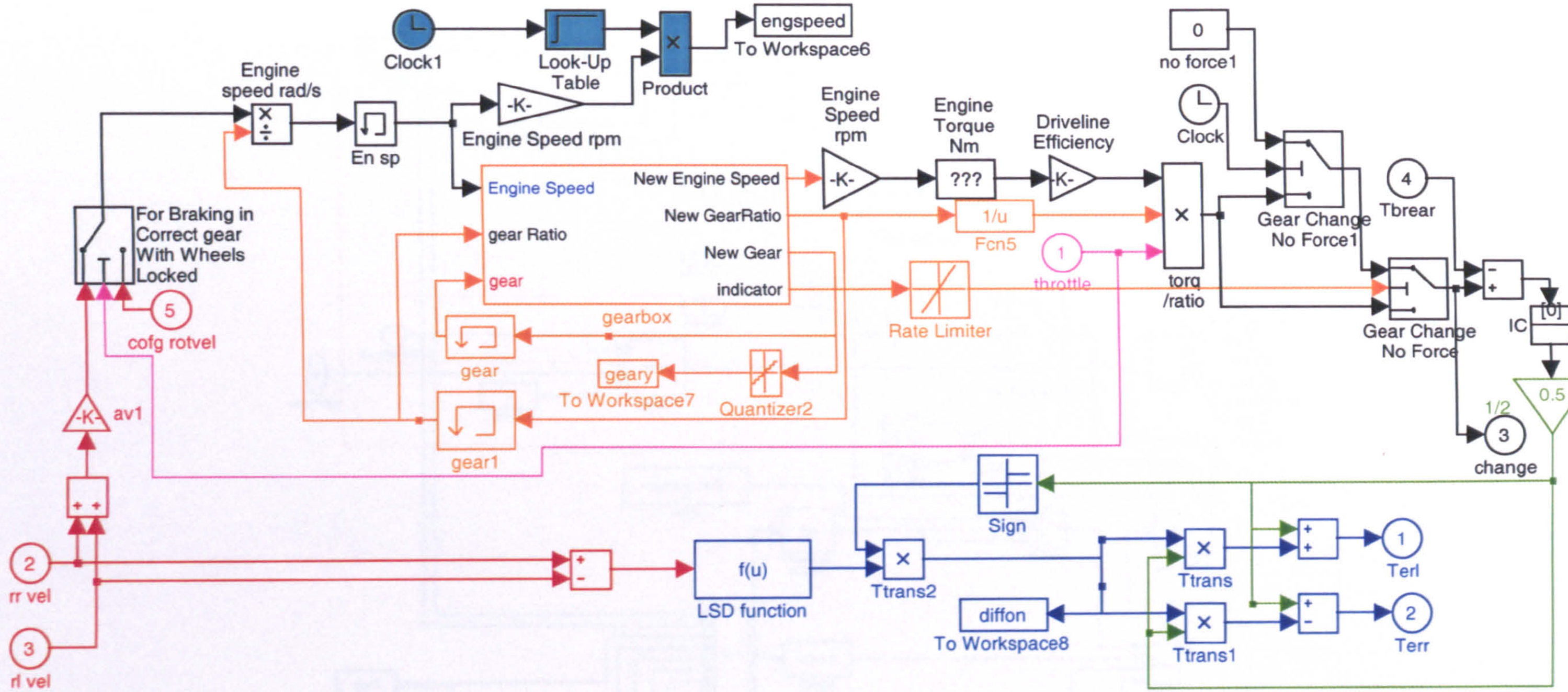

1
Normal force
2
slip angle
3
slip ratio
pac
Constant5
delft3
goodyear race tyre
lateral pure slip
long3
delft sports tyre1
longitudinal pure slip
murres
murres
pacloco
combined long
Xfl
2
combined long force3
paclaco
combined lateral
1
combined lat force2
height
1/pac(54,1)
tyre stiffness
3
rolling radius
pac(53,1)
Tyre Unloaded Radius

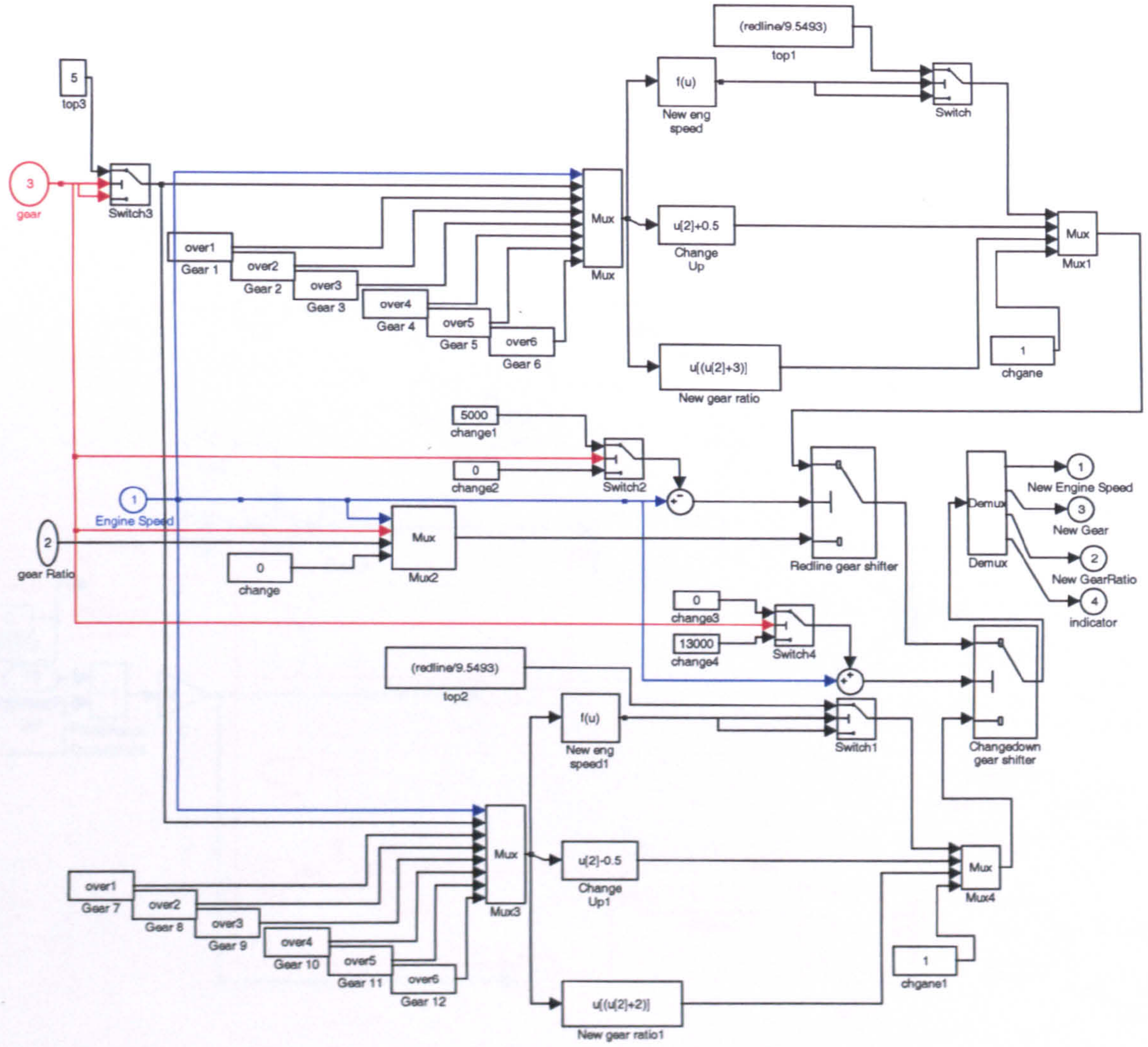

1
U
u*u
Fcn
fa*clr*rho*0.25
Lift
(m/2)*(a/l)*g
front static normal force
2
pitchang
(kthetar/2)*b
pitch
3
rollang
kphir*tr/2
roll
+
+
+
+
1
Nfi
Sum

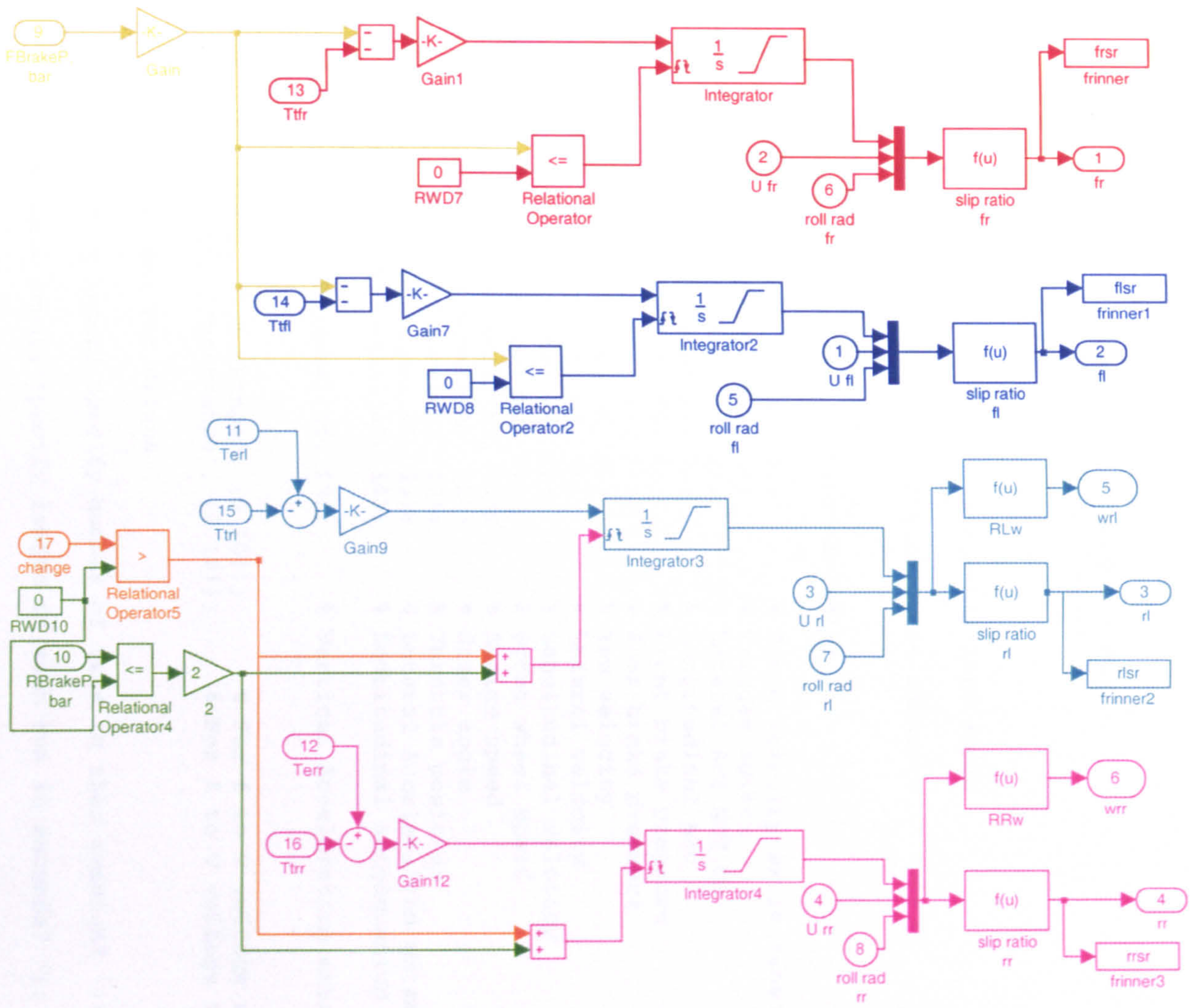

1
Normal force
2
slip angle
3
slip ratio
pac
Constant5
-K-
turnround
delft3
goodyear race tyre
lateral pure slip
long3
delft sports tyre1
longitudinal pure slip
murres
murres
pacloco
combined long
Xfl
2
combined long force3
paclaco
combined lateral
1
combined lat force2
height
1/pac(54,1)
tyre stiffness
3
rolling radius
pac(53,1)
Tyre Unloaded Radius


1
U
u*u
Fcn
fa*clr*rho*0.25
Lift
(m/2)*(a/l)*g
front static normal force
2
pitchang
(kthetar/2)*b
pitch
3
rollang
kphir*tr/2
roll
+
+
+
-
Sum
1
Nfi


2
delta
1
mux
f(u)
alphafl
delta1
1
fl
f(u)
Ufl
5
Ufl1
fl
frinner
-K-
radtodeg2
f(u)
alphafr
delta2
2
fr
fr
frinner1
-K-
radtodeg3
f(u)
Ufr
6
Ufr1
f(u)
alpharl
3
rl
rl
frinner2
-K-
radtodeg4
f(u)
Url
7
Url1
f(u)
alpharr
4
rr
rr
frinner3
-K-
radtodeg5
f(u)
Urr
8
Urr1


Clock1
Look-Up Table
Product
engspeed
To Workspace6
Engine speed rad/s
En sp
-K-
Engine Speed rpm
Engine Speed rpm
Engine Torque Nm
???
Driveline Efficiency
0
no force1
Clock
Gear Change No Force1
4
Tbrear
Engine Speed
gear Ratio
gear
New Engine Speed
New GearRatio
New Gear
indicator
1/u
Fcn5
1
throttle
torq /ratio
For Braking in Correct gear With Wheels Locked
5
cofg rotvel
gearbox
gear
geary
To Workspace7
Quantizer2
Rate Limiter
Gear Change No Force
IC
[0]
0.5
1/2
3
change
av1
gear1
Sign
2
rr vel
3
rl vel
f(u)
LSD function
Ttrans2
Ttrans
Ttrans1
diffon
To Workspace8
1
Terl
2
Terr


5
top3
3
gear
Switch3
over1
Gear 1
over2
Gear 2
over3
Gear 3
over4
Gear 4
over5
Gear 5
over6
Gear 6
Mux
(redline/9.5493)
top1
f(u)
New eng speed
Switch
u[2]+0.5
Change Up
Mux
Mux1
u[(u[2]+3)]
New gear ratio
1
chgane
5000
change1
0
change2
Switch2
1
Engine Speed
2
gear Ratio
Mux
Mux2
0
change
Redline gear shifter
Demux
Demux
1
New Engine Speed
3
New Gear
2
New GearRatio
4
indicator
0
change3
13000
change4
Switch4
(redline/9.5493)
top2
f(u)
New eng speed1
Switch1
Changedown gear shifter
Mux
Mux3
over1
Gear 7
over2
Gear 8
over3
Gear 9
over4
Gear 10
over5
Gear 11
over6
Gear 12
u[2]-0.5
Change Up1
Mux
Mux4
1
chgane1
u[(u[2]+2)]
New gear ratio1


9
FBrakeP
bar
K-
Gain
13
Ttfr
-K-
Gain1
1
s
Integrator
frsr
frinner
0
RWD7
<=
Relational
Operator
2
U fr
6
roll rad
fr
f(u)
slip ratio
fr
1
fr
14
Ttfl
-K-
Gain7
1
s
Integrator2
flsr
frinner1
0
RWD8
<=
Relational
Operator2
1
U fl
5
roll rad
fl
f(u)
slip ratio
fl
2
fl
11
Terl
15
Ttrl
-K-
Gain9
17
change
>
Relational
Operator5
0
RWD10
10
RBrakeP
bar
<=
Relational
Operator4
2
2
1
s
Integrator3
f(u)
RLw
5
wrl
3
U rl
7
roll rad
rl
f(u)
slip ratio
rl
3
rl
rlsr
frinner2
12
Terr
16
Ttrr
-K-
Gain12
1
s
Integrator4
f(u)
RRw
6
wrr
4
U rr
8
roll rad
rr
f(u)
slip ratio
rr
4
rr
rrsr
frinner3

Appendix F

Data acquisition and handling routines created in Matlab script. Notes begin with %

Data logging Program

```
% Version 1 of the data logging program
% for Phase 1+2 testing by B Siegler
% Created 17/7/01
% Last modified 9/8/01

% For use with national instruments card 6062E

% Initialise - create AI and allocate channels

AI = analoginput('nidaq', 1);
set(AI, 'InputType' , 'SingleEnded');

chan0 = addchannel(AI, 0);        % Pitch Velocity extra sensor
chan1 = addchannel(AI, 1);        % Trigger switch
chan2 = addchannel(AI, 2);        % Lateral Acc NOT ON
chan3 = addchannel(AI, 3);        % Longitudinal Acc
chan4 = addchannel(AI, 4);        % Front brake pressure
chan5 = addchannel(AI, 5);        % Rear brake pressure
chan6 = addchannel(AI, 6);        % Yaw velocity
chan7 = addchannel(AI, 7);        % Lateral velocity
chan8 = addchannel(AI, 8);        % Longitudinal velocity
chan9 = addchannel(AI, 9);        % Front wheel speed
chan10 = addchannel(AI, 10);      % Engine speed
chan11 = addchannel(AI, 11);      % Steer angle
chan12 = addchannel(AI, 12);      % Throttle position
chan13 = addchannel(AI, 13);      % Lateral Acceleration extra sensor
chan14 = addchannel(AI, 14);      % Longitudinal Acceleration extra
sensor
chan15 = addchannel(AI, 15);      % Vertical Acceleration extra
sensor

set(chan9, 'InputRange', [0 10]);       % For f to V voltage range
set(chan10, 'InputRange', [0 10]);      % For f to V voltage range

% Query user for values

numtrigs = input('Specify number of runs in this session? ');

duration = input('Specify length of each run in seconds? ');

if (numtrigs*duration) > 245
   disp(['']);
   disp(['====================== WARNING - maximum memory
allocation at 200 HZ and 16 channels exceeded ================']);
   disp(['']);
else
end
```

```
% Set values

set(AI, 'SampleRate', 200);
Actualrate = get(AI,'SampleRate');
set(AI, 'SamplesPerTrigger', Actualrate*duration);

set(AI, 'TriggerChannel' , chan1);
set(AI, 'TriggerType' , 'Software');%'HwAnalogPin');
set(AI, 'TriggerCondition' , 'Rising');
set(AI, 'TriggerConditionValue', 2.5);
set(AI, 'TriggerRepeat' , numtrigs);

% User inputs where to save data

disp(['']);
disp(['================================ Warning will over write
existing filenames ================================']);
disp(['']);

[newmatfile, newpath] = uiputfile('*.mat', 'Save As');

cd(num2str(newpath))

% Run session

start(AI)
while strcmp(AI.Running, 'On');
end

% Record data

for i = 1:1:numtrigs

   [data,time] = getdata(AI);

% Apply bias and gains to voltages to turn them into actual
vehicle parameters

% channel 2, 0 to 5V with 2.5V offset is -2g to +2g
channel1 = (data(:,3)-2.5)*(4/5);       % g

% channel 3, 0 to 5V with 2.5V offset is -2g to +2g
channel2 = (data(:,4)-2.5)*(4/5);       % g

% channel 4, 0 to 5V is 0 bar to 50 bar
channel3 = (data(:,5))*(50/5);          % bar

% channel 5, 0 to 5V 0 bar to 50 bar
channel4 = (data(:,6))*(50/5);          % bar

% channel 6, 0 to 5V with 2.5V offset is -64 deg/sec to +64
deg/sec
channel5 = (data(:,7)-2.5)*(128/5);     % degree / sec

% channel 7, 0 to 5V is 0 to 55.6 m/s (200 kph)
channel6 = (data(:,8))*(55.6/5);        % m/s

% channel 8, 0 to 5V with 2.5V offset is -6.94 m/s (25 kph) to
+6.94 m/s (25 kph)
channel7 = (data(:,9)-2.5)*(13.88/5);   % m/s
```

```
% channel 9, 0 to 12V  is 181.79 rad/s (93.65 mph)
channel8 = (data(:,10))*(181.79/12);      % rad/s - Saturates at
10 V which is 151.49 rad/s (78 MPH) NO at 5V = 75 rad/s CARD
SETUP PROBLEM

% channel 10, 0 to 12V  is 13000 rpm
channel9 = (data(:,11))*(13000/12);    % rpm - Saturates at 10 V
which is 10800 RPM No at 5V 5400 RPM CARD SETUP PROBLEM

% channel 11, 0 to 5V is -281.25 deg to +281.25 deg
channel10 = (data(:,12)-2.5)*(562.5/5);   % degrees

% channel 12, 2 to 5V is 0% to 100%
channel11 = (data(:,13)-2)*(100/3);    % percent throttle

    % channel 14, 0 to 5V with 2.5V offset is -5g to +5g
    channel12 = (data(:,15)-2.5)*(10/5);      % g

    % channel 13, 0 to 5V with 2.5V offset is -5g to +5g
    channel13 = (data(:,16)-2.5)*(10/5);      % g

    % channel 0, 0 to 5V with 2.5V offset is -5g to +5g
    channel14 = (data(:,1)-2.5)*(10/5);    % g

    % channel 14, 0 to 5V with 2.5V offset is -180 deg/sec to +180
    deg/sec
    channel15 = (data(:,14)-2.5)*(360/5);     % deg/s


save(num2str([newmatfile,num2str(i)]),'channel1','channel2','channel
3','channel4','channel5','channel6','channel7','channel8','channel9'
,'channel10','channel11','channel12','channel13','channel14','channe
115','time','Actualrate')  % save variables to file

% Storage rate is 15 channels at 200 Hz = 25.2 kbyte mat file size
per second of logging

end

% Termination: Plot the data and delete AI.

disp(['Actual sample rate was ',num2str(Actualrate),' Hz.']);
disp(['']);
disp(['Memory availability: ']);

daqmem(AI)      % Displays available memory

delete(AI)

y = 1;
n = 0;
runer = input('Do you wish to see results (y/n)? ');

if runer > 0.5;

   dataplot2

end
```

Data Handling Program

```
% Plotting program for phase 1+2 data logging
% Created by B Siegler 18/7/01
% Last Modified 30/7/01

clear;

disp(['Data viewing file created by B Siegler, please select *.mat
file to load:']);

[file dir]  = uigetfile('c:\matlab\testing\data2\*.mat');
load([dir file]);

N = [0:1:(length(channel1)-1)]';
T = time;
t_start = time(1,1);
t_end = time(length(time));

figure(1);

% Not connected

subplot(4,4,1);plot(T,channel1);grid on;
title('Latacc (g)');axis([t_start t_end -2 2]);

channel2 = channel2*1.24 + 0.04; % Long Acc sensor offset

subplot(4,4,2);plot(T,channel2);grid on;
title('Longacc (g)');axis([t_start t_end -2 2]);

subplot(4,4,3);plot(T,channel3);grid on;
title('Front line Pressure (bar)');axis([t_start t_end 0 50]);

subplot(4,4,4);plot(T,channel4);grid on;
title('Rear line Pressure (bar)');axis([t_start t_end 0 50]);

channel5 = (channel5 + 1.5)*1.18;       % Yaw vel sensor offset and
gain from not being at C of G

subplot(4,4,5);plot(T,channel5);grid on;
title('Yaw Velocity (deg/s)');axis([t_start t_end -64 64]);

% Swopping round lat and long velocity channels and adjustment for
angular offset of correvit

theta = -1.1;      % Angular offset of correvit
tempchan6 = channel6;

channel6 = (((channel7)/(13.88/5))+2.5)*(55.6/5);
% Correction from logv1
channel6 = ((channel6*cos(theta*pi/180)) +
(channel7*sin(theta*pi/180))) - 0.15;

subplot(4,4,6);plot(T,channel6);grid on;
title('Longitudinal Velocity (m/s)');axis([t_start t_end 0 56]);

channel7 = ((tempchan6)/(55.6/5))*(13.88/5)-2.5-4.445;
% Correction from logv1
```

```
channel7 = ((channel6*sin(theta*pi/180)) +
(channel7*cos(theta*pi/180)));

subplot(4,4,7);plot(T,channel7);grid on;
title('Lateral Velocity (m/s)');axis([t_start t_end -5.6 5.6]);

% f to V circuit not working

subplot(4,4,8);plot(T,channel8);grid on;
title('Front left wheel speed (rad/s)');axis([t_start t_end 0 195]);

% f to V circuit not working

subplot(4,4,9);plot(T,channel9);grid on;
title('Engine Speed (rpm)');axis([t_start t_end 0 13000]);

channel10 = channel10 + 66.8;     % Steer wheel angle offset

subplot(4,4,10);plot(T,channel10);grid on;
title('Steer Wheel Angle (deg)');axis([t_start t_end -180 180]);

subplot(4,4,11);plot(T,channel11);grid on;
title('Throttle (%)');axis([t_start t_end 0 100]);

%channel12 = channel12 - 0.045;  % Lat acc sensor offset

subplot(4,4,12);plot(T,channel12);grid on;
title('Latacc (g)');axis([t_start t_end -5 5]);

channel13 = (-1*channel13) + 0.115;     % Long acc sensor 2 offset

subplot(4,4,13);plot(T,channel13);grid on;
title('Longacc (g)');axis([t_start t_end -5 5]);

channel14 = channel14 + 1.02;     % Vert acc sensor offset

subplot(4,4,14);plot(T,channel14);grid on;
title('Vertacc (g)');axis([t_start t_end -5 5]);

channel15 = (channel15 - 2.5)*1.287;       % Roll ang sensor offset
and gain from not being at C of G

subplot(4,4,15);plot(T,channel15);grid on;
title('Roll Velocity (deg/s)');axis([t_start t_end -64 64]);

% Filtering bit - replots everything using butterworth filter
% at pass band 0 - 7 Hz and order 10

[B,A] = butter(5,5/100);

y = 1;
n = 0;
runer3 = input('Apply filter and replot results (y/n)? ');

if runer3 > 0.5;

   channel1f = filtfilt(B,A,channel1);
   channel2f = filtfilt(B,A,channel2);
   channel3f = filtfilt(B,A,channel3);
   channel4f = filtfilt(B,A,channel4);
   channel5f = filtfilt(B,A,channel5);
```

```
channel6f = filtfilt(B,A,channel6);
channel7f = filtfilt(B,A,channel7);
channel8f = filtfilt(B,A,channel8);
channel9f = filtfilt(B,A,channel9);
channel10f = filtfilt(B,A,channel10);
channel11f = filtfilt(B,A,channel11);
channel12f = filtfilt(B,A,channel12);
channel13f = filtfilt(B,A,channel13);
channel14f = filtfilt(B,A,channel14);
channel15f = filtfilt(B,A,channel15);

figure(2);

subplot(4,4,1);plot(T,channel1f);grid on;
title('Latacc (g)');axis([t_start t_end -2 2]);

subplot(4,4,2);plot(T,channel2f);grid on;
title('Longacc (g)');axis([t_start t_end -2 2]);

subplot(4,4,3);plot(T,channel3f);grid on;
title('Front line Pressure (bar)');axis([t_start t_end 0 50]);

subplot(4,4,4);plot(T,channel4f);grid on;
title('Rear line Pressure (bar)');axis([t_start t_end 0 50]);

subplot(4,4,5);plot(T,channel5f);grid on;
title('Yaw Velocity (deg/s)');axis([t_start t_end -64 64]);

subplot(4,4,6);plot(T,channel6f);grid on;
title('Longitudinal Velocity (m/s)');axis([t_start t_end 0 56]);

subplot(4,4,7);plot(T,channel7f);grid on;
title('Lateral Velocity (m/s)');axis([t_start t_end -5.6 5.6]);

subplot(4,4,8);plot(T,channel8f);grid on;
title('Front left wheel speed (rad/s)');axis([t_start t_end 0
195]);

subplot(4,4,9);plot(T,channel9f);grid on;
title('Engine Speed (rpm)');axis([t_start t_end 0 13000]);

subplot(4,4,10);plot(T,channel10f);grid on;
title('Steer Wheel Angle (deg)');axis([t_start t_end -180 180]);

subplot(4,4,11);plot(T,channel11f);grid on;
title('Throttle (%)');axis([t_start t_end 0 100]);

subplot(4,4,12);plot(T,channel12f);grid on;
title('Latacc (g)');axis([t_start t_end -2 2]);

subplot(4,4,13);plot(T,channel13f);grid on;
title('Longacc (g)');axis([t_start t_end -2 2]);

   subplot(4,4,14);plot(T,channel14f);grid on;
   title('Vertacc (g)');axis([t_start t_end -2 2]);

   subplot(4,4,15);plot(T,channel15f);grid on;
   title('Pitch Velocity (deg/s)');axis([t_start t_end -64 64]);

else
end
```

Frequency Domain Program

```
% Finds Measured data frequency domain response
% by Blake Siegler
% Created - 10/1/01
% Last Modified - 6/6/01
% random.mat to run!!!

clear;
tic

dir='c:\matlab\testing\data\wednesday\random\';
file=['random'];
load([dir file]);
N=[0:1:(length(channel_1r)-1)]';
T = N/200;
timeint = (N/200);
t_end=length(channel_1r)/200;

% Filters variables from actual data
[B,A] = butter(7,10/100);
channel_1fr = channel_1r; % filtfilt(B,A,channel_1r);
channel_2fr = channel_2r; % filtfilt(B,A,channel_2r);
channel_4fr = channel_4r; % filtfilt(B,A,channel_4r);

% Plotting bit


spectrum(channel_4r,channel_2fr*1.2,10000,0,hanning(10000),200,'line
ar');

figure(1)
semilogx(fy2,(abs(P_yaw2(:,4))));
title('For the transferfunction of steer angle to yaw velocity');
xlabel('Frequency (Hz)')
ylabel('Magnitude (rad/s/degree)')
grid

figure(2)
plot(fy2,P_yaw2(:,5));
title('For the coherence between input (steer angle) and output(yaw
velocity)')
xlabel('Frequency (Hz)')
ylabel('Ratio (1 is OK)')
grid

figure(3)
semilogx(fy2,((180/pi)*angle(P_yaw2(:,4))));
title('Phase For the transferfunction of steer angle to yaw
velocity');
xlabel('Frequency (Hz)')
ylabel('Phase (degrees)')
grid
```

```
[P_Fz2,f2] =
spectrum(channel_4r,channel_1fr,10000,0,hanning(10000),200,'linear')
;

figure(4)
semilogx(f2,(abs(P_Fz2(:,4))));
title('For the transferfunction of steer angle to latacc');
xlabel('Frequency (Hz)')
ylabel('Magnitude (g/degree)')
grid

figure(5)
plot(f2,P_Fz2(:,5));
title('For the coherence between input (steer angle) and
output(latacc)')
xlabel('Frequency (Hz)')
ylabel('Ratio (1 is OK)')
grid

figure(6)
semilogx(f2,((180/pi)*angle(P_Fz2(:,4)))-180);
title('Phase For the transferfunction of steer angle to latacc');
xlabel('Frequency (Hz)')
ylabel('Phase (degrees)')
grid

disp(['Time simulation took to run is ',num2str(toc/60),'
minutes.']);
```

Appendix G

Quasi-static simulation package MatLab based script program and Simulink models.

Notes begin with % in MatLab script.

LTS Version 4 graphical user interface

```
function fig = lts4fig_fig()
% This is the a Handle Graphics object
% and its children.  Note that handle values may change when these
objects
% are re-created. This may cause problems with any callbacks written
to
% depend on the value of the handle at the time the object was
saved.
% This problem is solved by saving the output as a FIG-file.
%
% To reopen this object, just type the name of the M-file at the
MATLAB
% prompt. The M-file and its associated MAT-file must be on your
path.
%
% NOTE: certain newer features in MATLAB may not have been saved in
this
% M-file due to limitations of this format, which has been
superseded by
% FIG-files.  Figures which have been annotated using the plot
editor tools
% are incompatible with the M-file/MAT-file format, and should be
saved as
% FIG-files.

load lts4fig_fig

h0 = figure('Color',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'Colormap',mat0, ...
    'FileName','C:\Matlab\ltsmk4\lts4fig.fig.m', ...
    'PaperPosition',[18 180 576 432], ...
    'PaperUnits','points', ...
    'Position',[172 102 802 625], ...
    'Tag','Fig1', ...
    'ToolBar','none');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.580392156862745 0.580392156862745
0.580392156862745], ...
    'ListboxTop',0, ...
    'Position',[165.1034482758621 40.3448275862069 327.7241379310345
91.241379310344841, ...
    'Style','frame', ...
    'Tag','Frame2');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
```

```
    'BackgroundColor',[0.580392156862745 0.580392156862745
0.580392156862745], ...
    'ListboxTop',0, ...
    'Position',mat1, ...
    'Style','frame', ...
    'Tag','Frame1');
h1 = axes('Parent',h0, ...
    'Units','pixels', ...
    'Box','on', ...
    'CameraUpVector',[0 1 0], ...
    'Color',[1 1 1], ...
    'ColorOrder',mat2, ...
    'Position',[258 197 476 324], ...
    'Tag','Axes1', ...
    'XColor',[0 0 0], ...
    'XGrid','on', ...
    'YColor',[0 0 0], ...
    'YGrid','on', ...
    'ZColor',[0 0 0], ...
    'ZGrid','on');
h2 = line('Parent',h1, ...
    'Color',[0 0 1], ...
    'Tag','Axes1Line1', ...
    'XData',mat3, ...
    'YData',mat4);
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Position',[-100.3157894736842 62.60061919504645
17.32050807568877], ...
    'String','Map of track', ...
    'Tag','Axes1Text36', ...
    'VerticalAlignment','bottom');
set(get(h2,'Parent'),'Title',h2);
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Position',[-100.3157894736842 -68.91640866873064
17.32050807568877], ...
    'String','Distance (m)', ...
    'Tag','Axes1Text35', ...
    'VerticalAlignment','cap');
set(get(h2,'Parent'),'XLabel',h2);
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','center', ...
    'Position',[-268.3157894736842 -0.557275541795647
17.32050807568877], ...
    'Rotation',90, ...
    'String','Distance (m)', ...
    'Tag','Axes1Text34', ...
    'VerticalAlignment','baseline');
set(get(h2,'Parent'),'YLabel',h2);
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'FontSize',16, ...
    'String','\leftarrowstart', ...
    'Tag','Axes1Text33');
```

```
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'FontSize',12, ...
    'Position',[-0.6450048174198725 3.39796120892083 0], ...
    'String','\leftarrow1', ...
    'Tag','Axes1Text32');
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'FontSize',12, ...
    'Position',[-4.144707647391422 6.920916466033344 0], ...
    'String','\leftarrow2', ...
    'Tag','Axes1Text31');
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'FontSize',12, ...
    'Position',[-4.761082619814922 20.12594299573779 0], ...
    'String','\leftarrow3', ...
    'Tag','Axes1Text30');
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'FontSize',12, ...
    'Position',[-5.130181235132099 32.76146554458742 0], ...
    'String','\leftarrow4', ...
    'Tag','Axes1Text29');
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'FontSize',12, ...
    'Position',[-13.73344958109397 42.02845098580508 0], ...
    'String','\leftarrow5', ...
    'Tag','Axes1Text28');
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'FontSize',12, ...
    'Position',[-22.83970368236792 50.79593637862712 0], ...
    'String','\leftarrow6', ...
    'Tag','Axes1Text27');
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'FontSize',12, ...
    'Position',[-47.26797534214111 51.63426853789934 0], ...
    'String','\leftarrow7', ...
    'Tag','Axes1Text26');
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'FontSize',12, ...
    'Position',[-74.84647242216116 48.26042655731247 0], ...
    'String','\leftarrow8', ...
    'Tag','Axes1Text25');
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'FontSize',12, ...
    'Position',[-67.29326681288625 40.97320817090993 0], ...
    'String','\leftarrow9', ...
    'Tag','Axes1Text24');
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'FontSize',12, ...
    'Position',[-57.67609275956805 20.81729136696969 0], ...
    'String','\leftarrow10', ...
    'Tag','Axes1Text23');
h2 = text('Parent',h1, ...
```

```
    'Color',[0 0 0], ...
    'FontSize',12, ...
    'Position',[-83.929241784347 71 30.060137005567 81 0], ...
    'String','\leftarrow11', ...
    'Tag','Axes1Text22');
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'FontSize',12, ...
    'Position',[-106.86822114500 78 49.740000947938 95 0], ...
    'String','\leftarrow12', ...
    'Tag','Axes1Text21');
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'FontSize',12, ...
    'Position',[-167.10251739246 05 29.898531884570 01 0], ...
    'String','\leftarrow13', ...
    'Tag','Axes1Text20');
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'FontSize',12, ...
    'Position',[-215.70939480889 87 4.9209019650962 25 0], ...
    'String','\leftarrow14', ...
    'Tag','Axes1Text19');
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'FontSize',12, ...
    'Position',[-217.93252865004 45 -14.519266616571 05 0], ...
    'String','\leftarrow15', ...
    'Tag','Axes1Text18');
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'FontSize',12, ...
    'Position',[-232.43296846238 89 -40.162095625289 33 0], ...
    'String','\leftarrow16', ...
    'Tag','Axes1Text17');
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'FontSize',12, ...
    'Position',[-199.37284757906 66 -41.361907609300 99 0], ...
    'String','\leftarrow17', ...
    'Tag','Axes1Text16');
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'FontSize',12, ...
    'Position',[-183.64585609273 95 -34.020963385508 86 0], ...
    'String','\leftarrow18', ...
    'Tag','Axes1Text15');
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'FontSize',12, ...
    'Position',[-179.72229073402 55 -23.783924095019 55 0], ...
    'String','\leftarrow19', ...
    'Tag','Axes1Text14');
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'FontSize',12, ...
    'Position',mat5, ...
    'String','\leftarrow20', ...
    'Tag','Axes1Text13');
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
```

```
    'FontSize',12, ...
    'Position',[-156.0638134302138 -30.81157450985256 0], ...
    'String','\leftarrow21', ...
    'Tag','Axes1Text12');
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'FontSize',12, ...
    'Position',mat6, ...
    'String','\leftarrow22', ...
    'Tag','Axes1Text11');
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'FontSize',12, ...
    'Position',[-113.7231480961271 -47.00832211412695 0], ...
    'String','\leftarrow23', ...
    'Tag','Axes1Text10');
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'FontSize',12, ...
    'Position',[-69.01796747542328 -45.75509775018616 0], ...
    'String','\leftarrow24', ...
    'Tag','Axes1Text9');
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'FontSize',12, ...
    'Position',[-76.78655683455878 -32.98326969309914 0], ...
    'String','\leftarrow25', ...
    'Tag','Axes1Text8');
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'FontSize',12, ...
    'Position',mat7, ...
    'String','\leftarrow26', ...
    'Tag','Axes1Text7');
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'FontSize',12, ...
    'Position',[-45.00173204685374 -30.05803018833477 0], ...
    'String','\leftarrow27', ...
    'Tag','Axes1Text6');
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'FontSize',12, ...
    'Position',[-40.45393433297453 -54.95686968557935 0], ...
    'String','\leftarrow28', ...
    'Tag','Axes1Text5');
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'FontSize',12, ...
    'Position',[-29.02244783803414 -44.35470864177537 0], ...
    'String','\leftarrow29', ...
    'Tag','Axes1Text4');
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'FontSize',12, ...
    'Position',[-18.46781038209989 -55.83847274555033 0], ...
    'String','\leftarrow30', ...
    'Tag','Axes1Text3');
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'FontSize',12, ...
```

```
    'Position',[-11.72136831485738 -24.12172064973028 0], ...
    'String','\leftarrow31', ...
    'Tag','Axes1Text2');
h2 = text('Parent',h1, ...
    'Color',[0 0 0], ...
    'HandleVisibility','off', ...
    'HorizontalAlignment','right', ...
    'Position',[-412.9473684210526 98.63777089783284
17.32050807568877], ...
    'Tag','Axes1Text1', ...
    'Visible','off');
set(get(h2,'Parent'),'ZLabel',h2);
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'Callback','trackmap', ...
    'ListboxTop',0, ...
    'Position',[18 245.7931034482759 111.7241379310345
24.827586206896561, ...
    'String','Plot Map of Track', ...
    'Tag','Pushbutton1', ...
    'Value',1);
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'Position',[275.5862068965518 49.0344827586207 65.79310344827587
59.586206896551741, ...
    'String',' ', ...
    'Style','listbox', ...
    'Tag','Listbox1', ...
    'Value',1);
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'Callback',mat8, ...
    'ListboxTop',0, ...
    'Position',[18 216.896551724138 111.7241379310345
24.827586206896561, ...
    'String','Run Cornering Analysis', ...
    'Tag','Pushbutton1');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'Callback','ltsmk4', ...
    'ListboxTop',0, ...
    'Position',[18 188.6896551724138 112.3448275862069
24.206896551724141, ...
    'String','Run Lap Time Simulator', ...
    'Tag','Pushbutton1');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.580392156862745 0.580392156862745
0.580392156862745], ...
    'Callback',mat9, ...
    'ListboxTop',0, ...
    'Position',mat10, ...
    'String','Acceleration vs. Distance', ...
    'Style','checkbox', ...
```

```
    'Tag','Checkbox1');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.580392156862745 0.580392156862745
0.580392156862745], ...
    'Callback',mat11, ...
    'ListboxTop',0, ...
    'Position',mat12, ...
    'String','Velocity vs. Distance', ...
    'Style','checkbox', ...
    'Tag','Checkbox1');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.580392156862745 0.580392156862745
0.580392156862745], ...
    'Callback',mat13, ...
    'ListboxTop',0, ...
    'Position',[18.620689655172 73.86206896551725 132.8275862068966
31.0344827586207], ...
    'String','Gear No. vs. Distance', ...
    'Style','checkbox', ...
    'Tag','Checkbox1');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.580392156862745 0.580392156862745
0.580392156862745], ...
    'Callback',mat14, ...
    'ListboxTop',0, ...
    'Position',[18.620689655172 43.44827586206898 133.448275862069
31.0344827586207], ...
    'String','Engine Speed vs. Distance', ...
    'Style','checkbox', ...
    'Tag','Checkbox1');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'FontSize',16, ...
    'FontWeight','bold', ...
    'ListboxTop',0, ...
    'Position',[180.75 432 355.5 25.5], ...
    'String','Lap Time Simulator v4.00 Control Window', ...
    'Style','text', ...
    'Tag','StaticText1');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.580392156862745 0.580392156862745
0.580392156862745], ...
    'FontSize',12, ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[18.620689655172 273.7241379310345 107.3793103448276
18], ...
    'String','Main Control Buttons', ...
    'Style','text', ...
    'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.580392156862745 0.580392156862745
0.580392156862745], ...
    'FontSize',12, ...
```

```
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[19.86206896551725 160.1379310344828 107.3793103448276
18], ...
    'String','Plot which figures?', ...
    'Style','text', ...
    'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.580392156862745 0.580392156862745
0.580392156862745], ...
    'FontSize',12, ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',mat15, ...
    'String','Vehicle Parameter Sets', ...
    'Style','text', ...
    'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.580392156862745 0.580392156862745
0.580392156862745], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',mat16, ...
    'String','Chassis:', ...
    'Style','text', ...
    'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.580392156862745 0.580392156862745
0.580392156862745], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',mat17, ...
    'String','Driveline:', ...
    'Style','text', ...
    'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.580392156862745 0.580392156862745
0.580392156862745], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',mat18, ...
    'String','Aero:', ...
    'Style','text', ...
    'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'Callback',mat19, ...
    'ListboxTop',0, ...
    'Position',mat20, ...
    'String','Vehicle Data', ...
    'Tag','Pushbutton1');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
```

```
    'Callback',mat21, ...
    'ListboxTop',0, ...
    'Position',mat22, ...
    'String','Track Map', ...
    'Tag','Pushbutton1');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.580392156862745 0.580392156862745
0.580392156862745], ...
    'Callback','                ', ...
    'FontSize',12, ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[21.1034482758620 355.0344827586208 68.27586206896554
16.137931034482 76], ...
    'String','Load Data', ...
    'Style','text', ...
    'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'Position',[347.5862068965518 49.0344827586207 65.17241379310346
59.586206896551741, ...
    'String',' ', ...
    'Style','listbox', ...
    'Tag','Listbox1', ...
    'Value',1);
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'Position',[420.2068965517242 49.65517241379311 65.79310344827587
58.965517241379321, ...
    'String',' ', ...
    'Style','listbox', ...
    'Tag','Listbox1', ...
    'Value',1);
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'FontSize',14, ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[14.25 9.75 562.5 25.5], ...
    'String','Lap Time Simulator v4.00, last modified December 2000
by Blake Siegler', ...
    'Style','text', ...
    'Tag','StaticText1');
if nargout > 0, fig = h0; end
```

LTS Version 4 Main Program

```
% Not So Simple Lap Time Simulator version 4
% by Blake Siegler
% Created - 24/7/99
% Last Modified - 29/8/00 for matlab version 5.3 and simulink 3
% Uses lts4fig.m to run program with ltsfig4.mat
% Needs files acbr2.m, acc3.mdl, brake3.mdl,
% runcorn.m, cornpow4.m, corn4.m, maxlat2.m, cb5solts.mdl and
% engpow.mdl, trackmap.m, curve3.mdl, straight.mdl, huntrat.m
% anging2.dll, delft3.dll, long3.dll,paclaco.dll,pacloco.dll
% lts4fig_fig.m, lts4fig_fig.mat
% lts4fig.m to run!!

% Features include:

% Graphical User interface
% Track editor
% Lateral and longitudinal weight transfer
% 2 DOF freedom type handling model, max lat acc found by quasi
steady state analysis
% Pacejka lateral force tyre model
% Aerodynamics at one ride-height with longitudinal and lateral
models
% Gear and drivetrain model
% Maximum cornering speed model
% Power limited cornering model
% Driver braking, acceleration and cornering ability modelled
% Can change value of coeff. of frcition on different parts of track
% Definable gear change time
% Driver model picks optimimum change up point for max torque at
wheels
% Pacejka longitudinal braking tyre model at optimum slip
% Combined tyre model in new cornering model

% Global Parameters

tic       % Used to start timer and find simulated time
tstep = 0.1;    % Maximum time step in simulink simulations (corn2.m
only!!)

% Matrix to define race track layout
% First item has to be a corner
% Straight/Corner Number...Path Radius (r=0 for striaght) in
m...Length in m...fraction reduction in mu
% All corners are taken as being in the same direction (right
handed)!!
% Can't handle straights that do not allow enough braking to next
corner.

% New Formula SAE endurance track without slalom

% Loop to run it around the track

siratr = size(racetrack,1);

% Loop to test if corner and creates matrix whole = [distance
velocity time accel gear revs]
% and import = [corner_number velocity first_row_data_starts]
```

```
blone = 0;
import = 0;
first = 0;
last = 0;
whole = [0 0 0 0 1 0];

for blone = 1:1:siratr
   first = size(whole,1);
      if abs(racetrack(blone,2)) > 0
         muchange = racetrack(blone,4);
         pr = abs(racetrack(blone,2));
         dist = racetrack(blone,3);
         [cory] = sim('runcorn',100); % ,[],[1e-3,0.01,tstep])
         last = last + size(corner,1);
      for slone=first:1:last; % Makes whole matrix
            whole(slone,1) = corner((slone-first+1),1);
            whole(slone,2) = corner((slone-first+1),2);
            whole(slone,3) = corner((slone-first+1),3);
            whole(slone,4) = corner((slone-first+1),4);
            whole(slone,5) = round(corner((slone-first+1),5));
            whole(slone,6) = corner((slone-first+1),6);
      end
      extra = last + 1;
      whole(extra,1) = 0;
      whole(extra,2) = 0;
      whole(extra,3) = 0;
      whole(extra,4) = 0;
      whole(extra,5) = 0;
      whole(extra,6) = 0;
import(blone,1) = blone;
import(blone,2) = corner(1,2);
import(blone,3) = last;
import(blone,4) = first;
      else
         import(blone,1) = blone;
         import(blone,2) = 0;
         import(blone,3) = 0;
         import(blone,4) = 0;
      end
end;


% Loop to test if straight and creates matrix wholestra = [distance
velocity time accel gear revs]

huntrat;     % Produces look up table of max slip ratio against
normal force for tyre

bltwo = 1;
extra = siratr + 1;
import(extra,1) = import(1,1);
import(extra,2) = import(1,2);
import(extra,3) = import(1,3);
first = 0;
last = 0;
wholestra = [0 0 0 0 1 0];
vert = 0;

for bltwo = 1:1:siratr
   first = size(wholestra,1);
      if abs(racetrack(bltwo,2)) == 0
```

```
length = racetrack(bltwo,3);
muchange = racetrack(bltwo,4);
before = bltwo -1;
after = bltwo +1;
accin = import(before,2);
brout = import(after,2);
acbr2;
last = last + size(total,1);
   for slone=first:1:last; % Makes wholestra matrix
         wholestra(slone,1) = total((slone-first+1),1);
         wholestra(slone,2) = total((slone-first+1),2);
         wholestra(slone,3) = total((slone-first+1),3);
         wholestra(slone,4) = total((slone-first+1),4);
         wholestra(slone,5) = total((slone-first+1),5);
         wholestra(slone,6) = total((slone-first+1),6);
      end
      extra = last + 1;
      wholestra(extra,1) = 0;
      wholestra(extra,2) = 0;
      wholestra(extra,3) = 0;
      wholestra(extra,4) = 0;
      wholestra(extra,5) = 0;
      wholestra(extra,6) = 0;
      import(bltwo,3) = last;
      import(bltwo,4) = first;
   else
   end
end;


% Loop to combine whole and wholestra using import

combined = size(whole,1) + size(wholestra,1);
grandtotal = [0 0 0 0];
blthree = 1;
amount = 0;
tother = 1;
first = 0;
last = 0;

for blthree = 1:1:siratr
      if import(blthree,2) > 0
         last = import(blthree,3) ;
         first = import(blthree,4) ;
            for slone = first:1:last
               amount = amount + 1;
               grandtotal(amount,1) = whole(slone,1) +
grandtotal(tother,1);
               grandtotal(amount,2) = whole(slone,2);
               grandtotal(amount,3) = whole(slone,3) +
grandtotal(tother,3);
               grandtotal(amount,4) = whole(slone,4);
               grandtotal(amount,5) = whole(slone,5);
               grandtotal(amount,6) = whole(slone,6);
         end
      tother = amount;
else
      last = import(blthree,3) ;
      first = import(blthree,4) ;
         for slone = first:1:last
            amount = amount + 1;
```

```
                grandtotal(amount,1) = wholestra(slone,1) +
grandtotal(tother,1);
                grandtotal(amount,2) = wholestra(slone,2);
                grandtotal(amount,3) = wholestra(slone,3) +
grandtotal(tother,3);
                grandtotal(amount,4) = wholestra(slone,4);
                grandtotal(amount,5) = fix(wholestra(slone,5));
                grandtotal(amount,6) = wholestra(slone,6);
            end
         tother = amount;
      end
end;

% Ploting bit

z = size(grandtotal,1);

other = [0 0];
for x = 1:1:z
   other(x,1) = grandtotal(x,4)/g;      % accel in g
   other(x,2) = grandtotal(x,2)*2.25;   % velocity in mph
end;
disp(['Time simulation took to run is ',num2str(toc),' seconds.']);
disp(['Total simulated lap time is ',num2str(grandtotal(z,3)),'
seconds.']);
disp(['Total distance covered in one lap is
',num2str(grandtotal(z,1)),' metres.']);
```

Corn4.m Program

```
% New cornering model for LTS MK 4
% Run before simulation to give look up table
% Cornering Model Using 2DOF to find maximum lateral accel at
particular forward velocity
% This implies a minimum path radius, a look up table of forward
% velocity against
% Path Radius is produced to give fastest time around a given path
% radius
% by Blake Siegler
% Created - 29/8/99
% Last Modified - 29/8/00 for matlab version 5.3 and simulink 3
% Needs files engpow.m, maxlat2.m cb5stolts.m and cornpow3.m to
% run!!!

% Cornering model loopy bit

tic
umin = 10;  % Minimum speed loop run at
umax = 40;  % Maximum speed loop run at 40
huntrat;    % Produces look up table of max slip ratio against
normal force for tyre
damp = 1;

pathrad = 0;
blone = 1;

for U = umin:2:umax          % Forward velocity
   get=2;                 % gear test loop to see which gear in and revs
   ygear(1,1) = (U/rrad/over1)*9.5493;
```

```
ygear(1,2) = (U/rrad/over2)*9.5493;
ygear(1,3) = (U/rrad/over3)*9.5493;
ygear(1,4) = (U/rrad/over4)*9.5493;
ygear(1,5) = (U/rrad/over5)*9.5493;
ygear(1,6) = (U/rrad/over6)*9.5493;
agear=1;
arev=ygear(1,1);
for get=2:1:6;
   if ygear(1,get) > changedown;
      agear=get;
      arev=ygear(1,get);
   else
      agear=agear;
      arev=arev;
   end;
end
delta = 1;
pathrad(1,1) = 0;
pathrad(1,2) = 0;
pathrad(1,3) = 0;
pathrad(1,4) = 0;
maxlat2;
pranswer((blone+2),1) = U;
pranswer((blone+2),2) = pathrad(1,1);
pranswer((blone+2),3) = pathrad(1,2);
pranswer((blone+2),4) = pathrad(1,3);
pranswer((blone+2),5) = pathrad(1,4);
pranswer((blone+2),6) = arev;
pranswer((blone+2),7) = agear;
blone = blone + 1;
end

% Loop to ensure pranswer can be used as a lookup table by
extrapolating pathrad if necessary

bighow = size(pranswer,1) - 1;

for cornbltwo = 2:1:bighow
   if pranswer(cornbltwo,3) <  pranswer((cornbltwo-1),3);
      pranswer(cornbltwo,3) = (pranswer((cornbltwo-1),3) +
pranswer((cornbltwo+1),3)) / 2;
   elseif pranswer((cornbltwo + 1),3) <  pranswer((cornbltwo),3);
      pranswer(cornbltwo,3) = (pranswer((cornbltwo-1),3) +
pranswer((cornbltwo+1),3)) / 2;
   else
   end
end

toc
```

Maxlat2.m Program

```
% maxlat2.m runs a sweep of steer angle
% and finds maximum lateral acceleration at particular speed.
% Created 29/8/00 by Blake Siegler
% Last Modified - 29/8/00 for matlab version 5.3 and simulink 3
% Needs files engpow.m, cb5stolts.m and cornpow4.m to run!!!

if delta > deltamax
      pathrad(2,1) = 0;
      pathrad(2,2) = 0;
      pathrad(2,3) = 0;
      pathrad(2,4) = 0;
   else
      [corn] = sim('cb5stolts',10);
      cornmax = size(pr,1);
      pathrad(2,1) = delta;
      pathrad(2,2) = pr(cornmax);
      pathrad(2,3) = latacc(cornmax);
      pathrad(2,4) = yawacc(cornmax);
end

if pathrad(2,3) > pathrad(1,3)
      pathrad(1,1) = delta;
      pathrad(1,2) = pr(cornmax);
      pathrad(1,3) = latacc(cornmax);
      pathrad(1,4) = yawacc(cornmax);
      delta = delta + 1;
      maxlat2;
   else
      delta = pathrad(1,1);
      cornpow4;
end
```

Cornpow4.m program

```
% Power Limited Cornering Program 4
% Created by Blake Siegler 29/8/00
% Last Modified - 29/8/00 for matlab version 5.3 and simulink 3
% Uses engpow.mdl and cornstolts.mdl to run!!!

[corn] = sim('cb5stolts',10);
cornmax = size(pr,1);
Dpower = U*(drag(cornmax));
[pow] = sim('engpow',1);

dfgd = [engpower(1,1) Dpower delta];

if engpower(1,1) < Dpower;
   delta = delta - 0.05;
   if delta > 0
      cornpow4;
   else
      delta = 0;
      pr(cornmax) = 10000;
      latacc(cornmax) = 0;
      yawacc(cornmax) = 0;
   end
end

pathrad(1,1) = delta;
pathrad(1,2) = pr(cornmax);
pathrad(1,3) = latacc(cornmax);
pathrad(1,4) = yawacc(cornmax);
```

ACBR2.m Program

```
% Linear acceleration and brake model
% This time using a Newton-Raphson iterative method
% by Blake Siegler
% Created - 5/8/99
% Last Modified - 16/8/00 for matlab version 5.3 and simulink 3
% Needs files huntrat.m, acc3.m and brake3.m to run!!
% To run on its own un-percentage accin,brout and length and ploting
bit and global parameters

% length = 300;          distance of straight in m
% accin = 15;          speed into straight in m/s
% brout = 10;          speed out of stright in m/s

% Iterative loop which accelerates for n seconds and then brakes to
brout

n = 0.3;            % Length of first acceleration run iteration in
seconds
can = -1;
total=[0 0 0];
lastot=1;
ygear=0;
loop =0;

while can < 0
      lastot = 1;
      total = 0;
      [accy] = sim('acc3',n);

         rubbish = 6;
      exit= size(accel,1);
      extra=1;
      accelf = [0 0 0 0 0 0];

         for filt = rubbish:1:exit;
            accelf(extra,1) = accel(filt,1);
            accelf(extra,2) = accel(filt,2);
            accelf(extra,3) = accel(filt,3);
            accelf(extra,4) = accel(filt,4);
            accelf(extra,5) = accel(filt,5);
            accelf(extra,6) = accel(filt,6);
            extra = extra + 1;
         end
lastacc = 1;
lastacc = size(accelf,1);
brain = accel(lastacc,2);
[bray] = sim('brake3',2*n);
j = 0;
z = 0;
k = 0;
lastbra = 1;
lastbra = size(brake,1);
z = lastacc + lastbra;
   for j=1:1:z;    % Makes total matrix
      if size(total,1) < lastacc
         total(j,1) = accelf(j,1);
         total(j,2) = accelf(j,2);
         total(j,3) = accelf(j,3);
```

```
                    total(j,4) = accelf(j,4);
                    total(j,5) = accelf(j,5);
                    total(j,6) = accelf(j,6);
                 else
                    k = j - lastacc;
                    total(j,1) = brake(k,1) + accelf(lastacc,1);
                    total(j,2) = brake(k,2);
                    total(j,3) = brake(k,3) + accelf(lastacc,3);
                    total(j,4) = -brake(k,4);
                          get=2;
% gear test loop to see which gear in and revs
                          agear=1;
                          arev=ygear(k,1);
                          previous=accelf(lastacc,5);
                          for get=2:1:previous;
                             if ygear(k,get) > changedown;
                                agear=get;
                                arev=ygear(k,get);
                             else
                                agear=agear;
                                arev=arev;
                             end;
                          end;
                       total(j,5) = agear;
                       total(j,6) = arev;
                    end
                 end
              n = n - ((total(z,1)-length)/accelf(lastacc,2));
% Newton-Rhapson iteration
              y=total(z,1);          % distance travelled so far...
              ej = fix((1*(total(z,1) - length - loop)));
% loop to find when iteration converges
                 if sign(ej) < 0
                    can = -1;
                 elseif sign(ej) > 0
                    can = -1;
                 else can = 1;
                 end;
                 %loop = loop + 0.5
end;
```

Huntrat.m Program

```
% Hunt finds the ratio which gives max long force.
% Using Pacejka formula
% by Blake Siegler
% Created - 11/8/00
% Last Modified - 16/8/00

% Global Parameters

% clear;

raty = [0 0];

% Program

for n = 6:1:14

   Z = 100*n;

   shx = pac(33,1);
   svx = Z*pac(35,1)*pac(37,1);
   cx = pac(23,1);
   mux = pac(24,1) * pac(37,1);
   dx = mux * Z;
   kx = pac(22,1) * pac(30,1);
   bx = kx/(cx*dx);
   ratio = 0;
   long = 0;

      while long < 0.999
         ratio = ratio + 0.001;
         ratio = ratio + shx;
         ex = pac(26,1)*(1 - (pac(29,1)*(sign(ratio))));
         long = sin(cx*atan((bx*ratio) - ex*((bx*ratio) -
atan(bx*ratio)))) + svx;
      end

   raty((n-5),1) = Z;
   raty((n-5),2) = ratio;

end
```

Trackmap.m Program

```
% Program to draw out track map in form of figure.
% Created by B Siegler 17/7/00
% Last modified 21/7/00
% Needs simulink program curve3.mdl, straight.mdl and anging.m to
run

% Creates local co-ords of x against y for all manoeuvres

howbig = size(racetrack,1);
tracmap = [0 0;0.0000001 1];
label = [0 0 0];

n=1;
oldx = tracmap(n,1);
oldy = tracmap(n,2);
newx = tracmap(n,1);
newy = tracmap(n,2);
extra = 0;

for loop1 = 1:1:howbig
   n = size(tracmap,1);
   shiftx = tracmap(n,1);
   shifty = tracmap(n,2);

   if abs(racetrack(loop1,2)) > 0
      pathrad = abs(racetrack(loop1,2));
      length = racetrack(loop1,3);
      [curvy] = sim('curve3',100);
% delivers matrix anti and clock [x y]
      howbig2 = size(tracmap);

% Rotation matrix to orientate next manoeuvre

oldx = tracmap(n-1,1);
oldy = tracmap(n-1,2);
newx = tracmap(n,1);
newy = tracmap(n,2);
extra = 0;

if and((newx-oldx<0),(newy-oldy>0))
   extra = pi/2;
elseif and((newx-oldx<0),(newy-oldy<0))
   extra = pi;
elseif and((newx-oldx>0),(newy-oldy<0))
   extra = 1.5*pi;
else
   extra = 0;
end

if eq(extra,1.5*pi)
   ang = (atan(abs(newx-oldx)/abs(newy-oldy)) + extra);
elseif eq(extra,pi/2)
   ang = (atan(abs(newx-oldx)/abs(newy-oldy)) + extra);
else
   ang = (atan(abs(newy-oldy)/abs(newx-oldx)) + extra);
end

rotate = [cos(ang) sin(ang);-sin(ang) cos(ang)];
```

```
clock = clock*rotate;
anti = anti*rotate;

if racetrack(loop1,2) > 0
   for sloop1 = 1:1:(size(clock))
      tracmap(n,1) = clock(sloop1,1) + shiftx;
      tracmap(n,2) = clock(sloop1,2) + shifty;
      n = n + 1;
   end

   label((loop1+1),1) = clock((round(sloop1/2)),1) + shiftx;
   label((loop1+1),2) = clock((round(sloop1/2)),2) + shifty;
   label((loop1+1),3) = loop1;

   else
      for sloop1 = 1:1:(size(anti))
         tracmap(n,1) = anti(sloop1,1) + shiftx;
         tracmap(n,2) = anti(sloop1,2) + shifty;
         n = n + 1;
      end

      label((loop1+1),1) = anti((round(sloop1/2)),1) + shiftx;
      label((loop1+1),2) = anti((round(sloop1/2)),2) + shifty;
      label((loop1+1),3) = loop1;

   end

else

   length = racetrack(loop1,3);
   gradient = (newy-oldy)/(newx-oldx);
   [starty] = sim('straight',100);
   % delivers matrix straight [x y]

   % Rotation matrix to orientate next manoeuvre

   oldx = tracmap(n-1,1);
   oldy = tracmap(n-1,2);
   newx = tracmap(n,1);
   newy = tracmap(n,2);
   extra = 0;

   if and((newx-oldx<0),(newy-oldy>0))
      extra = pi/2;
   elseif and((newx-oldx<0),(newy-oldy<0))
      extra = pi;
   elseif and((newx-oldx>0),(newy-oldy<0))
      extra = 1.5*pi;
   else
      extra = 0;
   end

   if eq(extra,1.5*pi)
      ang = (atan(abs(newx-oldx)/abs(newy-oldy)) + extra);
   elseif eq(extra,pi/2)
      ang = (atan(abs(newx-oldx)/abs(newy-oldy)) + extra);
   else
      ang = (atan(abs(newy-oldy)/abs(newx-oldx)) + extra);
   end

   rotate = [cos(ang) sin(ang);-sin(ang) cos(ang)];
```

```
            straigh = straigh*rotate;

            for sloop2 = 1:1:(size(straigh))
               tracmap(n,1) = straigh(sloop2,1) + shiftx;
               tracmap(n,2) = straigh(sloop2,2) + shifty;
               n = n + 1;
            end

            label((loop1+1),1) = straigh((round(sloop2/2)),1) + shiftx;
            label((loop1+1),2) = straigh((round(sloop2/2)),2) + shifty;
            label((loop1+1),3) = loop1;

      end
end


figure(1)
plot(tracmap(:,1),tracmap(:,2))
title('Map of track')
xlabel('Distance (m)')
ylabel('Distance (m)')
grid
zoom

text(0,0,'\leftarrowstart','fontsize',16)

for loop2 = 2:1:size(label)

text(label(loop2,1),label(loop2,2),['\leftarrow',num2str(label(loop2
,3))],'fontsize',12)
end

y = 1;
n = 0;
runer=input('Is the map of the track correct (y/n)? ');

if runer < 0.5;
   position = input('What is the number of the straight/corner that
is wrong? ');
   radius = input('What is the value of the path radius in m
(straight = 0, left handers = -ve)? ');
   long = input('What is the length of the path taken at this radius
in m? ');

   racetrack(position,2) = radius;
   racetrack(position,3) = long;

   disp(['Track map updated replotting (remember to update inputted
racetrack)...']);
   trackmap;
else
end;
```

Tracks.m Program

```
% List of tracks for use in LTS packages, Mark 1 , 2 and 2a
% Column 1  Column 2    Column 3    Column 4
% Number Path radius Length      Percentage drop in friction
% of maneuvre  0 for straight including circum

racetrack = [1 100   314   1;...
       2 0  800   1;...
       3 200   314   1;...
       4 0  300   1;...
       5 300   200   1;...
       6 0  100   1;...
       7 50 150   1;...
       8 0  800   1]

% A Newer Formula SAE endurance track without slalom

racetrack = [1 -4.78 5.00  1;...
             2  4.78  5.00  1;...
             3  0     21.51 1;...
             4  -4.78 3.8   1;...
             5  0     21.51 1;...
             6  -4.78 3.8   1;...
             7  0     45.41 1;...
             8  -4.78 13.34 1;...
             9  0     5.98  1;...
             10 11.95 43.78 1;...
             11 0     23.90 1;...
             12 -28.68   38.03 1;...
             13 0     90.77 1;...
             14 -14.34   20.01 1;...
             15 14.34 20.01 1;...
             16 -14.34   45.03 1;...
             17 0     30.85 1;...
             18 -4.78 4 1;...
             19 0     15.54 1;...
             20 11.95 36.52 1;...
             21 0     15.54 1;...
             22 -4.78 8.5   1;...
             23 0     80.12 1;...
             24 -4.78 11.26 1;...
             25 0     20.32 1;...
             26 22.71 89.14 1;...
             27 0     38.24 1;...
             28 -5.50 17.33 1;...
             29 5.50  17.33 1;...
             30 -5.50 17.33 1;...
             31 0     53.00 1]

% Steady state circle

racetrack = [1 8.55   53.72   1]

% Straight line run to 100m

racetrack = [1 0.5   0.3   1;...
       2 0  135   1]
```

Appendix H

Transient approach based manoeuvre time minimisation package, MatLab script. Notes begin with %.

Dof9opt2.m Program

```
% Optimisation routine that finds optimum longacc and steer controls
% for a given corner
% Uses optimisation toolbox!!
% Created 23/1/01 by Blake Siegler
% Last modified 28/6/01
% Needs files dof9ver8.mdl, dof9jturn2.m, dof9jturncon2.m,
% pateror3.mdl,
% drawcorn2.mdl, drawcorn.mdl, pathmaker4.mdl, prev2bet.mdl,
% pathmabet4.mdl and fsae01a.m to run!!

clear
tic

% Load Main Variables
================================================================

U = 15;  % Initial fwd. vel. in m/s
cpd = 4; % Control points ditance in m

% Load Vehicle Variables
================================================================

deltagain = 1;     % Too even out control matrix - also max value of
delta allowed
fsae01a;     % Loads all vehicle and tyre parameters
brakeoff = 1;  % Set to zero to remove rolling/aero resistence
effects, 1 to have on (maintains fwd vel)

% Load Path Variables
================================================================

% For Right Hand Turn starting at 5m

patrad = 17.5;          % path radius centre line in m
width = 5;              % track width in metres
clcourse = 5;              % X position of centre line of track start
position
stralen = 40;              % Length of Straights
length = patrad*pi/2;    % length of corner (180deg)

% Run Preview controller and produce path matrix
================================================

%y = 1;
%n = 0;
%runer = input('Load path and run initial conditions (y/n)? ');
```

```
%if runer > 0.5;

% Load circuit data - track centre line and width

[corn] = sim('drawcorn2',100);

nosteps = stralen/0.1253;

diststep = 0;
diststep2 = 0;
loop2 = 0;
timestep = 0;
inner =0;
for loop1 = 1:1:(2*nosteps+size(outer,1))
   timestep = 0.1253 + timestep;
   if loop1 <= nosteps
      diststep = diststep + 0.1253;
      inner(loop1,1) = 0;
      inner(loop1,2) = diststep;
      inner(loop1,3) = timestep;
      inner(loop1,4) = width;
   elseif and((loop1>nosteps),(loop1<=(nosteps+size(outer,1))))
      loop2 = loop2 + 1;
      inner(loop1,1) = outer(loop2,1);
      inner(loop1,2) = outer(loop2,2) + stralen;
      inner(loop1,3) = timestep;
      inner(loop1,4) = width;
   else
      diststep2 = diststep2 + 0.1253;
      inner(loop1,1) = patrad + diststep2;
      inner(loop1,2) = stralen + patrad;
      inner(loop1,3) = timestep;
      inner(loop1,4) = width;
   end
end

% Run preview controller to get inital guess for delta

huntrat;
prevdist = U/7;    % Distance between preview points

fulldist = stralen*2 + length;

ending = 100;      % time when end at apex
erxtra = 3;%5;     % stops it being in the negative x bit

% Gives centre line and track info.
[maker] = sim('pathmaker4',size(inner,1)*0.1253);  % Makes the ideal
path matrix to follow

% Change of inital guess
==========================================================================
% Changes initial guess to closer approx., rather than track centre
line
% Load circuit data - track centre line and width

patrad = 23;    % path radius centre line in m
width = 5;  % track width in metres
length = patrad*pi/2;    % length of corner (180deg)
```

```
[corn] = sim('drawcorn2',100);

stralen = 33;%13
nosteps = stralen/0.1253;
diststep = 0;
diststep2 = 0;
loop2 = 0;
timestep = 0;
inner =0;

for loop1 = 1:1:(2*nosteps+size(outer,1))
   timestep = 0.1253 + timestep;
   if loop1 <= nosteps
      diststep = diststep + 0.1253;
      inner(loop1,1) = 0;
      inner(loop1,2) = diststep;
      inner(loop1,3) = timestep;
      inner(loop1,4) = width;
   elseif and((loop1>nosteps),(loop1<=(nosteps+size(outer,1))))
      loop2 = loop2 + 1;
      inner(loop1,1) = outer(loop2,1);
      inner(loop1,2) = outer(loop2,2) + stralen;
      inner(loop1,3) = timestep;
      inner(loop1,4) = width;
   else
      diststep2 = diststep2 + 0.1253;
      inner(loop1,1) = patrad + diststep2;
      inner(loop1,2) = stralen + patrad;
      inner(loop1,3) = timestep;
      inner(loop1,4) = width;
   end
end

extrabet = 0;%-2;        % moves path across

% makes new path to follow betpath
[maker] = sim('pathmabet4',size(inner,1)*0.1253);  % Makes the ideal
path matrix to follow

OPTION = simset('AbsTol',1e-6,'RelTol',1e-6);

% now follows betpath
[corn] = sim('prev2bet',ending); % Uses 3DOF preview controller as
9DOF one doesn't work!!

disp(['Initial guess took ',num2str(time(size(time,1))),' seconds to
complete manoeuvre.']);

% Discretenise control matrix and set up communication grid (every
one metre)

nextone = 0;
nextone2 = 0;
deltax = 0;
timex = 0;

for loop3 = 1:1:size(s,1)
   if nextone2 <= s(loop3,1)
      nextone2 = cpd + nextone2;
      nextone = 1 + nextone;
      deltax(nextone,1) = delta(loop3,1)/deltagain;
```

```
        deltax(nextone,2) = 0;
        timex(nextone,1) = time(loop3,1);
    else
    end
end

% Return variables back to origional value

patrad = 17.5; % path radius centre line in m
stralen = 40;

%else
%end;

% Program

% Calls jturn function that runs car using steer time history
provided, stops when reaches
% certain x-position and returns time (end of manoeuvre).
% Jturn function needs: discrete time and steer history, fwd vel,
path, x-stop position

% Want to:
% Minimise time to complete the manoeuvre
% Keep to constraints on steer angle range
% and path taken

% options =
optimset('LargeScale','off','Display','iter','tolfun',0.001,'tolcon'
,0.05,'maxiter',35,'diffminchange',0.05,'diffmaxchange',1,'tolx',0.0
05);
% Standard options

% options =
optimset('LargeScale','off','Display','iter','tolfun',0.0001,'tolcon
',0.005,'maxiter',35,'diffminchange',0.001,'diffmaxchange',0.5,'tolx
',0.001);
% Options set tight

% options =
optimset('LargeScale','off','Display','iter','tolfun',0.1,'tolcon',0
.5,'maxiter',35,'diffminchange',0.05,'diffmaxchange',10,'tolx',0.05)
;

options =
optimset('LargeScale','off','Display','iter','tolfun',0.001,'tolcon'
,0.01,'maxiter',35,'diffminchange',0.001,'diffmaxchange',0.5,'tolx',
0.001);
% Options on hybrid
% tolfun = 0.01
%
================================================================
% options =
optimset('LargeScale','off','Display','iter','tolfun',0.001,'tolcon'
,0.01,'maxiter',45,'diffminchange',0.01,'diffmaxchange',1,'tolx',0.0
1);
%'tolfun',0.00001
% turn largescale on for other routines

% Minimisation routine which calls maxlat2 which delivers back cost
function
```

```
%dir='e:\matlab\9dof\resultopt\';
%file=['jturn3']; % To load the path
%load([dir file]);

% timex=timeint;
% deltax = xans;
%
==========================================================

% Set boundary conditions for contorl matrix
==========================================

lb = [(zeros(size(deltax,1),1) - 10) (zeros(size(deltax,1),1) - 1)];
ub = [(zeros(size(deltax,1),1) + 10) (zeros(size(deltax,1),1) + 1)];

lb(1,1) = 0;    % Just to initialise
lb(2,1) = 0;    % Just to initialise
lb(1,2) = 0;    % Just to initialise
lb(2,2) = 0;    % Just to initialise

ub(1,1) = 0;    % Just to initialise
ub(2,1) = 0;    % Just to initialise
ub(1,2) = 0;    % Just to initialise
ub(2,2) = 0;    % Just to initialise

==========================================================

[xans,fval,exitflag,output] =
fmincon('dof9jturn2',deltax,[],[],[],[],lb,ub,'dof9jturncon2',option
s,U,path,(stralen+patrad+clcourse),timex,erxtra,clcourse);

xans
fval
%xans = deltax;

% Plot final path

xend1 = stralen + patrad + clcourse;
timeint = timex;
delta = xans(:,1);
brakeint = xans(:,2);

fsae01a;            % Loads all vehicle and tyre parameters
deltagain = 1;      % Too even out control matrix
brakeoff = 1;
% Set to zero to remove rolling/aero resistence effects, 1 to have
on (maintains fwd vel)


 % Rerun results to display results
==========================================================

OPTIONS = simset('SrcWorkspace','current');%,'AbsTol',1e-
9,'RelTol',1e-9); % 'MaxStep',0.005,
[corn] = sim('dof9ver8',50,OPTIONS);
x1 = x;
y1 = y;

disp(['Final solution took ',num2str(time(size(time,1))),' seconds
to complete manoeuvre.']);
```

```
width = 5;  % track width in metres
patrad2 = patrad - width/2;    % path radius  in m
length = patrad2*(pi/2);   % length of corner (90deg)
[corn] = sim('drawcorn',100);

% Adds straights onto corner

inner(1,2) = inner(1,2) - stralen;
outer(1,2) = outer(1,2) - stralen;
endin = size(inner,1);
inner(endin+1,1) = inner(endin,1) + stralen;
inner(endin+1,2) = inner(endin,2);
outer(endin+1,1) = outer(endin,1) + stralen;
outer(endin+1,2) = outer(endin,2);

figure(1)
plot(x1(:,1),y1(:,1),path(:,1),path(:,2),'-
.',xpr2,ypr2,':',inner(:,1)+
width/2,inner(:,2)+stralen,'k',outer(:,1) +
width/2,outer(:,2)+stralen,'k','linewidth',2)
legend('optimised path','centre line of track','original guess')
title('Vehicle Position')
xlabel('Distance (m)')
ylabel('Distance (m)')
grid

figure(2)
plot(time,latacc,':',time,longacc,time,u,'-.')
legend('Lateral acc','Longitudinal acc','Fwd vel')
title('Acceleration and Velocity versus Time')
xlabel('Time (s)')
ylabel('Acceleration (m/s2) or Forward Velocity (m/s)')
grid

distdiff = abs(s(size(s,1),1) - path(size(path,1),3));
distdiff = distdiff * 0.8;

prevdist = 0;
paterr = 0;
OPTION = simset('MaxStep',0.1,'SrcWorkspace','current');
[corn3] =
sim('pateror3',2*s(size(s,1),1),OPTION);%(size(x1,1)/10),OPTION);

figure(3)
plot(paterrdist,paterr)
title('Vehicle Position')
ylabel('Path error (m)')
xlabel('Distance (m)')
grid

disp(['Time simulation took to run is ',num2str(toc/3600),'
hours.']);
```

Dof9jturn2.m Program

```
function F = dof9jturn2(delta1,U1,path,xend,time1,erxtra,clcourse)

% Jturn function used in conjunction with
% optimisation function to find steer angle time history
% to complete jturn manoeuvre in shortest time
% Created by B. Siegler 23/1/01
% Last modified 26/6/01
% Needs files dof9ver8.mdl, fsae01a.m to run!!!!

% Have to load variables into this current workspace

fsae01a;    % Loads all vehicle and tyre parameters
deltagain = 1;    % Too even out control matrix
brakeoff = 1;  % Set to zero to remove rolling/aero resistance
effects, 1 to have on (maintains fwd vel)
erxtra = 3; % stops it being in the negative x bit

% Control Values

xend1 = xend;
timeint = time1;
delta = delta1(:,1);
brakeint = delta1(:,2);
U = U1;
ending = 20;

% Program

OPTIONS = simset('SrcWorkspace','current');%,'AbsTol',1e-
9,'RelTol',1e-9,'MaxStep',0.001)
[corn] = sim('dof9ver8',ending,OPTIONS);
timend = time(size(time,1));

% Bit to check for completion of maneouvre

cheque = size(u,1);
cheque2 = -1;
endone = size(u,1);

cheque = endone;
   if u(cheque,1) <= (brout)
         timend = abs(timend + (path(size(path,1),3) + 5 -
s(size(s,1),1))); % better linear solution to time penalty problem
          cheque2 = 0;
      elseif or(or((Nfr(cheque,1) <= (offground)),(Nfl(cheque,1) <=
(offground))),or((Nrr(cheque,1) <= (offground)),(Nrl(cheque,1) <=
(offground))))
         timend = abs(timend + (path(size(path,1),3) + 5 -
s(size(s,1),1))); % better linear solution to time penalty problem
           cheque2 = 1;
      elseif abs(latacc(cheque,1)) >= (latspin)
          timend = abs(timend + (path(size(path,1),3) + 5 -
s(size(s,1),1))); % better linear solution to time penalty problem
             cheque2 = 2;
      else
   end
F = timend;
```

Dof9jturncon2.m Program

```
function [c,ceq] =
dof9jturncon2(delta1,U1,path,xend,time1,erxtra,clcourse)

% Constraints function to keep in track boundary
% Jturn function used in conjunction with
% optimisation function to find steer angle time history
% to complete jturn manoeuvre in shortest time
% Created by B. Siegler 23/1/01
% Last modified 8/5/01
% Needs files dof9ver8.mdl, fsae01a.m, pateror3.mdl to run!!!!

% Have to load variables into this current workspace

fsae01a;    % Loads all vehicle and tyre parameters
deltagain = 1;    % Too even out control matrix brakeoff = 1;  % Set
to zero to remove rolling/aero resistence effects, 1 to have on
(maintains fwd vel)
erxtra = 3; % stops it being in the negative x bit

% Control Values

xend1 = xend;
timeint = time1;
delta = delta1(:,1);
brakeint = delta1(:,2);
U = U1;
time2 = 0;

% Program

OPTIONS = simset('SrcWorkspace','current');
ending = 20;
[corn] = sim('dof9ver8',ending,OPTIONS);
timend = time(size(time,1));
x1 = x;
y1 = y;


if abs(timeint(size(timeint,1)) - timend) > 1;
   distdiff = 0;
else
   distdiff = abs(s(size(s,1),1) - path(size(path,1),3));
   distdiff = distdiff * 0.8
end

% Works out whether within track boundary

prevdist = 0;
paterr = 0;
OPTION = simset('MaxStep',0.1,'SrcWorkspace','current');
[corn3] =
sim('pateror3',2*s(size(s,1),1),OPTION);%(size(x1,1)/10),OPTION);

% Nonlinear inequlaity constraints

paterrr = 0;
nextone = 1;
```

```
% Makes discrete matrix of error distance at control points

for loop4 = 1:1:size(paterr,1)
   if nextone <= size(timeint,1)
         if timeint(nextone,1) <= paterrtime(loop4,1);
            paterrr(nextone,1) = paterr(loop4,1);
         nextone = 1 + nextone;
      else
       end
   else
   end
end

paterrrr = 0;

if eq(size(paterrr,1),size(timeint,1))
   paterrrr = paterrr;
else
   for loop5 = 1:1:size(timeint,1)
      if loop5 <= size(paterrr,1)
         paterrrr(loop5,1) = paterrr(loop5,1);
      else
         paterrrr(loop5,1) = 0;
      end
   end
end

c = [abs(paterrrr) - 2.5];

% No nonlinear equality constraints

ceq = [];
```