Model Based Wheel Slip Control via Constrained Optimal Algorithm

A thesis submitted in fulfilment of the requirements for the degree of Master of Engineering

Dae Keun Yoo

School of Electrical and Computer Engineering
RMIT University
April 2006

Contents

List of Figures

Abstract

In a near future, it is imminent that passenger vehicles will soon be introduced with a new revolutionary brake by wire system (BBW) which will replace all the mechanical linkages and the conventional hydraulic brake systems with complete 'dry' electrical components. One of many potential benefits of a brake by wire system is the increased brake dynamic performances due to a more accurate and continuous operation of the EMB actuators which leads to an increased amount of possibilities for controlling antilock brake system (ABS). The main focus of this thesis is on the application of a model predictive control (MPC) method to devise an ABS for a BBW vehicle. Unlike the traditional ABS control algorithms which are based on a trial and error method, the MPC based ABS algorithm aims to utilizes the behaviour of the model to optimize the wheel slip dynamics subject to system constraints. Performance of the proposed wheel slip controller is validated through Software-in-the-Loop (SiL) and Hardware-in-the-Loop (HiL) simulation. Furthermore, a novel multi processor real-time simulation system is developed using the reflective memory network and the off-the-shelf hardware components to meet the high demands of the computational power and the real time constraints of HiL simulation

Declaration

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program; and, any editorial work, paid or unpaid, carried out by a third party is acknowledged.

Dae Keun Yoo
April 2006

Acknowledgement

I would like to express my sincere gratitude to my supervisor, Prof. Liuping Wang for providing excellent guidance and assistance during the course of this work.

I also would like to acknowledge and thank the industry partners, PBR Automotive Pty Ltd and Research Centre for Advanced By-Wire Technologies (RABiT) for providing assistance and equipment for this research to be carried out.

On a personal note, I would like to thank my family for their encouragement. This research could not have been done without the relentless support of my family. Finally, I would like to thank the love of my life, Su Jin for her faith, love and support.

Chapter 1

Introduction

This chapter begins with the description of an Antilock Brake System (ABS) and a Brake-By-Wire (BBW) system. Following this, the motivation for this research work and the summary of contributions, as well as literature reviews are presented to give the reader an overall picture of the substances covered in this thesis. Finally, the chapter closes with an outline overview.

1.1 System Description

1.1.1 Anti-lock Braking System

Anti-lock Brake System (ABS) is designed to help alleviate the danger of vehicle instability during an emergency braking. Since it was first introduced to passenger vehicles in 1970s, ABS has been acclaimed as providing significant improvement to overall safety standard and the braking performance associated with road vehicles.

The primary objective of an ABS is to prevent the wheels from locking up due to an excessive brake torque applied by the driver during an emergency braking manoeuvre. Importance of avoiding the wheel lock-up is twofold. First, the directional stability of the vehicle is maintained and can be maximised to enable the avoidance of obstacles on the road during hard braking. Second, the significantly higher friction force can be attained between the tyre and the road, which in turn, minimises the braking distance, except for gravel condition, where it is found that the stopping distance is longer with ABS control.

In order to achieve these important objectives of driver safety, ABS system utilises on-borad embedded controllers, wheel speed sensors and auxiliary brake components

to recognise any impending wheel lock-ups by monitoring the level of longitudinal slip (λ). The term longitudinal slip (λ) is the normalised difference between the linear vehicle velocity (v) and the wheel angular speed (ωr), where it can be obtained by the Equation (1.1). Furthermore, the slip equation shows that wheel lock up (i.e. $\omega r = 0$) is indicated by the slip value of 1, and similarly the longitudinal slip value of $\lambda = 0$ indicates free motion of the wheel. Depending on the value of longitudinal slip, ABS system controls the amount of brake caliper clamp forces to achieve an optimal level of wheel slip, which in turn maximises the available friction force between the road and the tyre. A more detailed operational principle of ABS can be found in (Bosch, 1999).

$$\lambda = \frac{v - \omega r}{v} \tag{1.1}$$

Based on the above longitudinal slip (λ) definition, a further observation can be made about the correlation between the longitudinal slip and the tyre friction force, where it is illustrated by the curve shown in Figure 1.1. The horizontal axis of the curve represents the values of the longitudinal slip λ, and the vertical axis indicates the amount of attainable friction force between the road and the tyre. Analyzing the curve, the friction force increases with an increase in a slip value up to λ_o, where the slip λ_o is the optimum value for a given road condition. However for any value higher than λ_o, it is evident that less friction force (or the sliding force) is exerted on the wheel. Additionally, the longitudinal slip also affects the lateral controllability (i.e. capability of the tyre to generate the lateral force in response to steering commands) of a vehicle while undertaking a braking manoeuvre. The basic phenomena of this dependency is that when the tyre is generating a lateral force it reduces the absolute longitudinal force of a tyre. For instance, if the maximum longitudinal force of the tyre is generated 10% slip in a normal straight ahead case then this is also the value of slip which generates the maximum trade-off between longitudinal and lateral force. Based on this physical phenomena of the tyre and the fast changing nature of slip dynamics, it is impossible for an average driver to manually control the braking forces to avoid the wheel lock-up during an emergency braking manoeuvre, which explains the necessity of an antilock brake system.

1.1.2 Brake-By-Wire

The introduction of new active chassis control systems (e.g. Brake-Assistant (BA), Electronic Stability Program (ESP), and Traction Control System (TCS) etc.) in the past two decades has significantly increased the complexity of a conventional brake

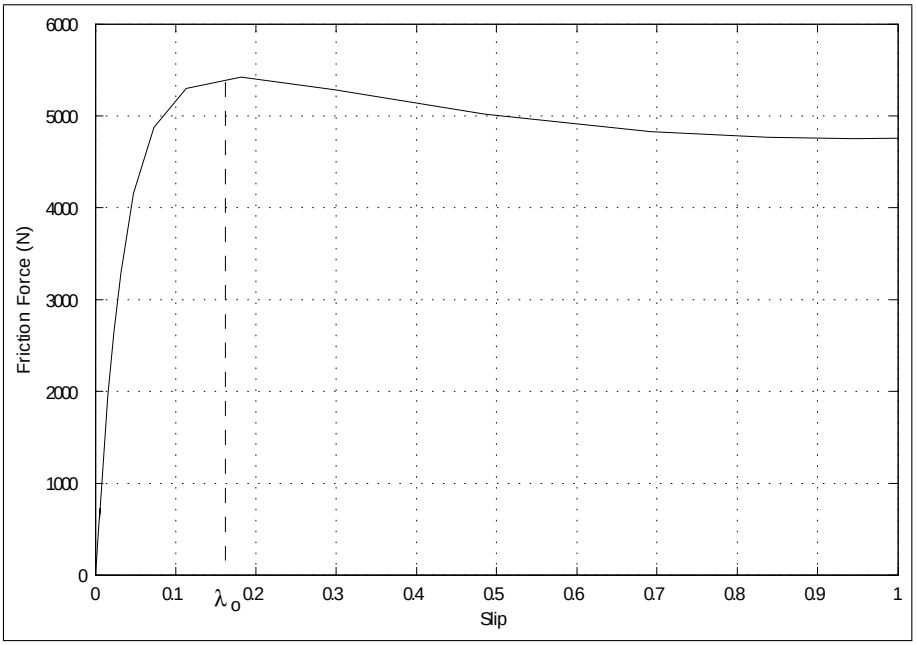


Figure 1.1: Tyre friction curve

system. This trend of introducing a more sophisticated active chassis control system is expected to be continued over the coming years. As a result, an emphasis on a driver safety, and possibly the importance of driver-vehicle interaction are also expected to rise. It is envisaged that the conventional hydraulic brake system will require a even more intricate system to support these features, which will eventuate in increasing number of brake-related components. To overcome the concerns of the system complexity and economical downfall, the "Brake-By-Wire" (BBW), also known as the Electromechanical brake system (EMB), has been proclaimed to be the next evolutionary step in automotive brake systems. In the latest development of a BBW system replaces all the mechanical linkages and the conventional hydraulic brake systems with complete 'dry' electrical components. The most common design of BBW system consists of distributed electronic brake controllers, central electronic control unit, electromechanical disc brake actuators, brake-by-wire pedal unit and 42V electrical systems, where similar set-ups can be found in (Solyom, 2004), (Hedenetz & Belschner, 1998) and (Petersen, 2003). Typical layout of a BBW system is shown in Figure 1.2 and further brief insights into the main components of the system are given below,

- **Wheel Brake Control Unit (WBCU)**

  The WBCU is the electronic control unit for providing basic braking function-

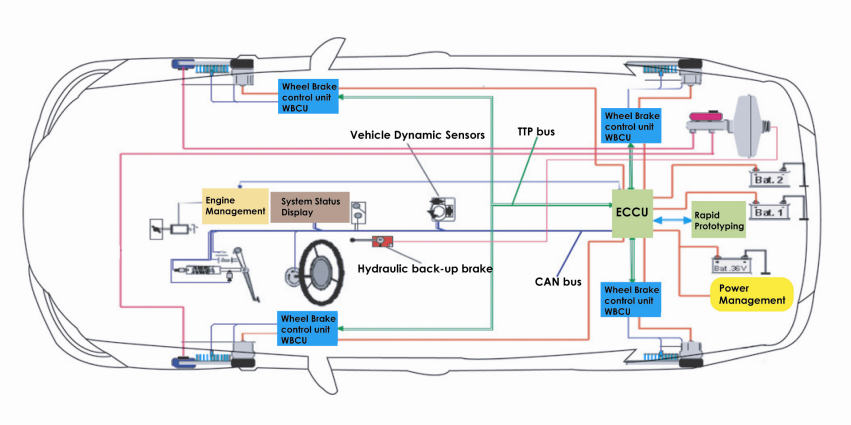


Figure 1.2: Typical BBW

ality such as delivering of requested clamp force from a driver or to follow the clamp force input trajectory from a vehicle dynamics controller. In order to ensure accurate and fast delivery of clamp force, closed loop clamp force control algorithm, based on controlling the movement of BLDC motor is implemented. WBCU also handles a fault tolerant communication bus to share the information with other distributed units.

- **Energy management / Central control unit (ECCU)**

  The ECCU is the core of the system. It contains the whole braking functionality such as start-up and shut-down control of the system, voting of pedal signals, diagnostic functions, failure detection and the driver information. The central control unit also operates the bus system and manages the interfaces to the outside of the brake system.

- **Brake-By-Wire Pedal Unit**

  Pedal unit consists of a mechanical pedal simulator to provide a comfortable characteristic between pedal force and pedal travel which is demanded by the driver to meter the braking force of the car properly. There are 3 analog sensors connected to the pedal to detect pedal travel, pedal force as well as the brake light switch where it is used for detecting the driver's wish for braking. Additionally there is a connection from one analog pedal sensor to the actuators to provide the actuator control units with the necessary pedal information in case of a double-fault of the bus system.

- **Communication Bus**

In a brake-by-wire system, there are two main communication buses to interconnect the various components. In order to ensure data consistency for the real-time operated EMB system, bidirectional and dual communication channel, Time Triggered Protocol (TTP) bus, are used to handle the exchange of informations such as commanded clamp forces from the central control unit and the delivered clamp forces from the local control unit. In a time-triggered architecture, the communication system decides when to transmit a message according to a predetermined schedule within each controller, hence provides a prompt transmission of messages with high data efficiency. Importance of the TTP bus and elaborated description of the protocol can be found in (Kopetz & Grunsteidl, 1994) . CAN (Controller Area Network) bus is also used in the system to provide the vehicle sensor informations and the brake pedal pressure values to the central control unit (ECCU). The rapid prototyping box is then connected via CAN bus for expanded brake functionality.

- **Actuators**Electromechanical disk brake (EMB) is the main actuator of the

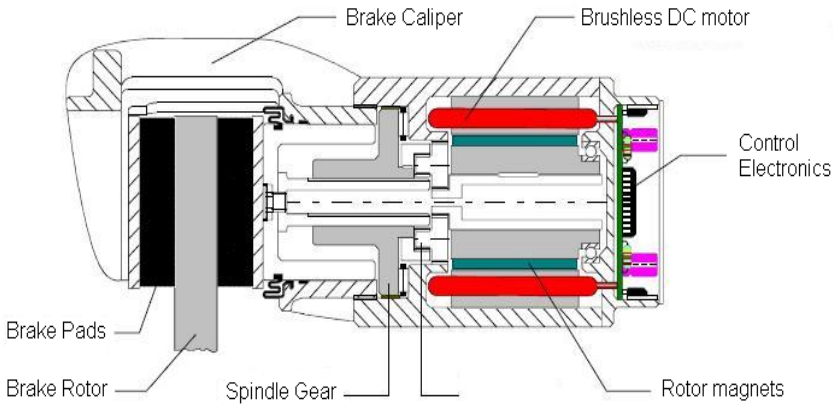


Figure 1.3: Cross-sectional view of EMB

brake-by-wire system for delivering the required clamp forces at the brakes. From the cross-sectional diagram of EMB shown in Figure 1.3, the design of EMB consists of brushless DC electric motor, planetary gear, spindle drive and a floating disk brake caliper housing. In order to apply the clamp forces at the brake, DC motor is actuated by the on-board electronics (WBCU), to generate the angular motion of the motor, hence spins the integrated gear which in turn, rotates the ball screw. As the ball screw rotates, spindle moves back and forth in horizontal motion and transforms the motion into the clamp forces at the brake through the floating caliper housing. Figure 1.4 shows the picture of the

actuator that has been fitted in the vehicle. For this work, complete modelling of the EMB is not considered, however a more complete description can be found in (Line *et al.* , 2004)

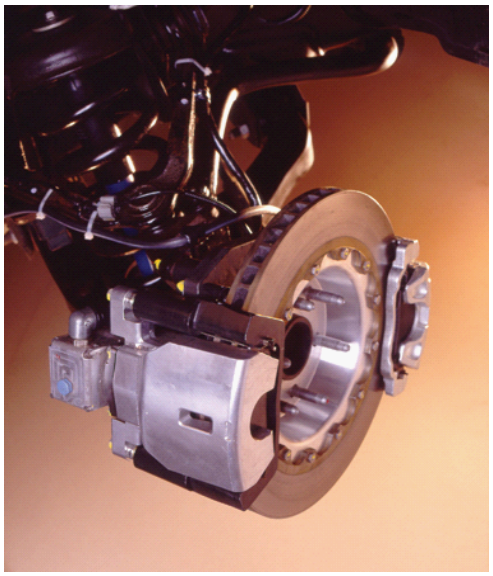


Figure 1.4: Photo of EMB fitted in the vehicle.

1.2 Motivation

The conventional ABS in hydraulic brake system utilises the hydraulic pump and solenoid control valve to modulate the brake fluid pressure in order to apply the brake torque at the wheel. The most widely implemented control strategy in this type of ABS, is the cyclic operation of the three discrete states (Build,Release,Hold). This type of discrete control promotes an on-off braking response which leads to several side-effects such as induced pulsating sensation from the brake pedal, which may affect the driver response and causes poor performance in regulating the wheel slip at the optimum level. With the introduction of a BBW system to passenger vehicles, a more accurate braking control can be achieved due to a continuous operation of electromechanical disk brake calipers (EMB). Hence it promises a new prospect in improving the performance of ABS brakes.(see (Bannatyne, 1998), (Ayoubi *et al.* , 2004) and (Kelling & Leteinturier, 2003) for a summary of conceivable benefits and challenges of brake-by-wire system and by-wire technology in general). An opportunity of this research project is found between RMIT university and Pacifica Group Technology Pty Ltd (PGT) to investigate and to devise a suitable control method of ABS,

which can utilise the techonological advantage of a brake-by-wire system and translate it into superior performance in ABS controlled braking. Moreover, the project further aims to develop an efficient simulation framework to support simulation validation and testing of a new control strategy under a realistic hardware environment

1.3 Literature Review

1.3.1 Antilock Brake System

Controlling the braking of a vehicle is inherently a very difficult control problem due to the highly nonlinear nature of the system and its environmental uncertainties such as road conditions, tyre wear and dynamic load transfer etc. To deal with inherent nonlinearity of the system, most of the production type ABS systems are table and rule based controllers which are tuned for the different road conditions by a trial and error. This type of design approach requires an extensive field testing and leads to serious limitation of further research due to a considerable time and a cost involved in a overall process. As a result, attention has been paid to unearthing a more advanced method of controlling the braking process, and numerous publications have appeared over the years.

(Drakunov *et al.* , 1995) recognizes the non-linear and uncertain nature involved with the design of ABS and devised an algorithm that enabled the maximum value of the tire/road friction force to be reached during emergency braking without a prior knowledge of the optimal slip. The algorithm allows for tracking of an unknown optimal value even if the optimal value changes in real-time. An inadequacy of this algorithm is that it requires friction force, since this quantity cannot be measured directly, an observer that estimates its magnitude is required.

The work conducted by (Unsal & Kachroo, 1999) is also approached in this section. They introduced a sliding mode controller using slip ratio as a control variable to maintain the wheel slip at a set value. This approach is limited by vehicle speed estimation. In overcoming this limitation an analytical non-linear observer based on both Kalman filter and sliding mode observer to estimate the vehicle speed investigated based on the works by (Semmler *et al.* , 2003). Semmler resolved this issue by applying a sliding mode control to track a reference input slip, this also utilized the continuous control advantage of a brake-by-wire actuator. (Hadri *et al.* , 2001) presented a nonlinear observer and controller based on passivity and sliding mode control. This control method is based on the on-line estimation of the tire force using

the concept of relaxation length. The controller only uses an angular wheel velocity measurement in its estimation process.

More recently published articles illustrate that optimal control method is gaining interest. (Petersen, 2003)apply a constrained Linear Quadratic Regulator (LQR) control method based on multi-parametric quadratic programming. The control method is specifically developed to capture the advantages of an electromechanical braking system. Validation of the controller performance is carried out on a full scale vehicle experiment. In a similar approach, (Anwar & Ashrafi, 2002) derive a generalized predictive control law for ABS control system.

1.3.2 Brake-by-wire

The first approach to brake-by-wire system, known as the electrohydraulic brake system (Jonner *et al.* , 1996) is based on the traditional hydraulic brake system where the by-wire function is implemented through hydraulic pumps and additional electric controller valves. Continued progression from its first implementation has led to the latest generation of BBW system, also known as Electromechanical brake system (EMB). In this system, brake forces and brake control are realised by electric components, whereby the actuation of the brakes are directed by electromechanical power without the use of brake fluid. (Schwarz *et al.* , 1998) and (Maron *et al.* , 1997) introduce an electromechanical brake system and include simulation models for both the brake and the vehicle. In particular (Schwarz *et al.* , 1990) investigates a clamp force sensor for its use in the control of the electromechanical disk brake.

In this form of BBW system, it requires a more time and effort concerning its improvements in safety requirements. (Hedenetz & Belschner, 1998) presents a brake-by-wire application without mechanical or hydraulic backup, with realisation by using a new architecture approach called time-triggered fault-tolerant communication. Comparison study of performance comparison between the conventional hydraulic and the electromechanical system by (Emereole, 2004) demonstrates that the electromechanical brake system offers improved braking performance in terms of fast and accurate brake torques application at the wheels.

1.3.3 Model Predictive Control

Model predictive control (MPC) is essentially a class of standard optimal control algorithm, except that it involves solving the constrained optimal control problem on-line

for the current state of the plant by defining a finite horizon rather than determining it off-line (Mayne *et al.* , 2000). Over the last few decades, MPC has been an active research area from both academic and industries.

A number of major publications have appeared in the literature. This includes the tutorial papers by (Rawlings, 2000) and (Wang, 2002) to give a wider background, an extensive theoretical review paper by (Garcia *et al.* , 1989), (Morari & Lee, 1999), as well as books by (Maciejowski, 2002) and (Rossiter, 2003).
One of the strong features of MPC is the on-line computation of the optimal control variable in the presence of constraints. Papers by (Rao *et al.* , 1998) and (Wright, 1997) discuss the application of quadratic programming to model predictive control.

Although a rich source of technical informations is available in the field of MPC, vast majority of the research have been heavily biased towards the discrete time systems and comparatively the continuous time system implementation received a less attention. Recent work on continuous time model predictive control includes (Gawthrop *et al.* , 1998), (Kouvaritakis *et al.* , 1999) and (Wang, 2001). In (Wang, 2001), addresses the apparent difficulties of a continuous time implementation and proposes a method of describing the control trajectory and formulation of constraints using a set of orthonormal basis functions.

1.3.4 Multi-Processor Simulation

In many application, a computer-aided simulation testing is widely acknowledged as means of optimising and observing the system's behavior under a variety of conditions. By no exception, computer-aided simulation testing is recognised as an indispensable tool in the development cycle of automotive components.

The increasing reliance of a simulation testing has escalated the complexity of the simulation model. Furthermore, the emergence of a real-time simulation and a hardware-in-the-loop simulation (HiL) require the guaranteed execution of tasks in fixed time interval. A traditional uniprocessor approach has often found to be inappropriate and computationally insufficient.

In order to accommodate these performance demands, the necessity of a distributed simulation is addressed by (Pollini & Innocenti, 2000) to achieve a simulation environment with flexibility and modularity/reusability of simulator components. A distributed simulation holds many advantages over the uniprocessor method and summary

of these foreseen benefits of parallel and distributed simulation from (Fujimoto, 1990) are given below:

- Execution times of analytic simulations can be reduced by subdividing a large simulation computation into many sub-computations that can execute concurrently. One can reduce the execution time by up to a factor equal to the number of processors that are used. This may be important simply because the simulation takes a long time to execute, e.g., simulations of communication networks containing tens of thousands of nodes may require days or weeks for a single run.
- Very fast executions are needed for on-line simulations because there is often very little time available to make important decisions. In many cases, simulation results must be produced in seconds in order for simulation results to be useful. Again, parallel simulation provides a means to reduce execution time.
- Simulations used for virtual environments must execute in real time, i.e., the simulator must be able to simulate a second of activity in a second of wallclock time so that the virtual environment appears realistic in that it evolves as rapidly as the actual system. Distributing the execution of the simulation across multiple processors can help to achieve this property.
- Distributed simulation techniques can be used to create virtual environments that are geographically distributed, enabling one to allow humans and/or devices to interact as if they were collocated. Such distributed virtual environments have obvious benefits in terms of convenience and reduced travel costs.
- Distributed simulation can simplify integrating simulators that execute on machines from different manufacturers. For example, flight simulators for different types of aircraft may have been developed on different architectures. Rather than porting these programs to a single computer, it may be more cost effective to "hook together" the existing simulators, each executing on a different computer, to create a new virtual environment.
- Another potential benefit of utilizing multiple processors is increased tolerance to failures. If one processor fails, it may be possible for other processors to continue the simulation provided critical elements do not reside on the failure processors.

From these advantages, a distributed simulation methodology has received a wide attention in the automotive industry, as a gateway to the real-time hardware in the

loop simulation, see (Nabi *et al.* , 2004), (Ploger *et al.* , 2004) and (Kohl & Jegminat, 2005).

1.4 Contributions

The main contributions of this thesis are summarised as below

- **Wheel slip controller design via continuous-time model predictive control.**

  A model based wheel slip control algorithm is developed using a generic continuous time model predictive control method. The subsidiary control logics are also developed for the purpose of supervisory state logic control. The final implementation of the wheel slip controller utilises a decentralised control architecture and the performance validation is carried out in a realistic hardware-in-the-loop simulation condition.

- **Real-time multiprocessor simulation system**

  Multi-processor simulation framework is developed based on the commercially available components. The proposed system uses a reflective memory network to combine the distributed simulation units. Both favorable attributes of a loosely coupled and a tightly coupled system, are combined to provide the flexibility and the real-time computational requirement. A group of associated software algorithms is also developed to provide the automated process of the model decomposition and the synchronisations of the distributed units.

- **Hardware-in-the-loop simulation validation testing.**

  Based on the real-time multiprocessor simulation concept, a hardware-in-the-loop simulation system is developed to incorporate a prototype brake by wire vehicle into the simulation loop. As a result, the proposed wheel slip predictive controllers are validated in a realistic simulation environment.

1.5 Thesis outline

The thesis consists of six chapters and organised chronologically in the order of development activities for the wheel slip control system. The outline of the thesis is as follow.

Chapter 2 presents the underlying physical properties of the vehicle and the wheel in a longitudinal motion. Mathematical description of a normal force variation caused by the pitch motion of the vehicle during braking is developed and a linearised wheel slip model is derived based on a quarter car model which contains the tyre deformation and the brake actuator dynamics.

Chapter 3 covers the design and implementation of wheel slip control system based on the continuous-time model predictive control algorithm. First, the control objective of a wheel slip control system is defined with a further discussion on the robustness requirements to the parameter uncertainties. Insights into the overall control structure including the supervisory state logic and flow chart of the algorithm are also presented.

Chapter 4 explains the hardware and the software aspects of the proposed distributed simulation framework. The concept of reflective memory network is introduced to provide an application of coupling multiple processors in a distributed environment. A synchronisation and a task allocation between distributed units are elaborated. Final section of the chapter presents the simulation validation of the wheel slip controller on a realistic vehicle simulation model. Firstly, the devised wheel slip control system is tuned for different road surfaces and the performance optimisation of the wheel slip control system is described in detail. Lastly, an antilock brake performance of the wheel slip controller is evaluated on the same set of road surfaces. In this case, the supervisory control logic is implemented to detect any impending wheel lock up.

Chapter 5 compares antilock brake performance between a PID system and the devised MPC wheel slip control system are compared.

Chapter 6 expands the scope of a software simulation testing to include a hardware in the loop (HiL) simulation. An overview of the proposed hardware-in-the-loop simulation framework is presented with a functional description of the real-time simulation cluster and the interconnections between the prototype brake-by-wire vehicle. This is followed with the performance evaluation of the tuned slip controller and the antilock wheel slip controller.

Chapter 7 summarises the main contributions of this dissertation and identifies the potential directions for future work.

Chapter 2

Mathematical Model Development

The aim of this chapter is to develop a simple analytical model of the longitudinal dynamics of the vehicle and the wheel, which possess most of the dynamic influential on the longitudinal braking process. (see also (Petersen, 2003), (Emereole, 2004) and (Hadri *et al.* , 2001)). Section 2.1 describes the variation of the normal force acting on the wheel while braking and develops a mathematical relationship between the normal forces and longitudinal acceleration (deceleration). Section 2.2 presents wheel slip dynamics of a single wheel in the longitudinal motion through a quarter car model. An analytical relationship between the change in slip and the brake torque are derived using a local linearisation. The transient dynamic behavior of the tyre is described and included in the final design of the linear model.

2.1 Longitudinal Vehicle Dynamic

A longitudinal model of a vehicle, as shown in Figure 2.1, is considered whereby each wheel remains on the same road surface and assumes the lateral tyre forces, the yaw moment and roll moment to be zero during a braking maneuver. Moreover, aerodynamic drag forces are assumed to have minimum effect, and more importantly the link between the vehicle mass and the ground (i.e. through vehicle suspension, wheel and tyre) is assumed to be rigid in the longitudinal direction, which allows the longitudinal braking forces at the wheels to be included in the vehicle pitch calculations.
Based on above assumptions and the simplified longitudinal model of a vehicle, it can be deduced that a weight transfer takes place about the centre of gravity due to longitudinal deceleration, which is known as pitch motion. This pitch motion causes

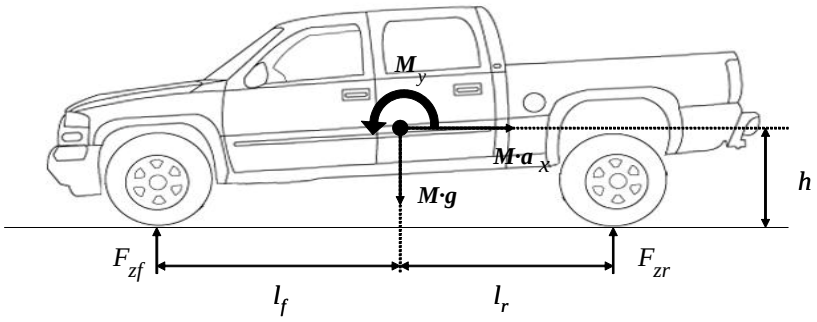


Figure 2.1: Loads acting on the longitudinal vehicle model during braking.

a variation in normal forces F_{zf} and F_{zr}, which can be described by constructing the equations of force and torque in respect of x and z axis.

$$\sum F_z = F_{zf} + F_{zr} - \frac{M}{2} \cdot g = 0 \tag{2.1}$$

$$\sum M_y = F_{zf} \cdot l_f - F_{zr} \cdot l_r - \frac{M}{2} \cdot a_x \cdot h = 0 \tag{2.2}$$

$$F_{zf} = \frac{M\left(l_r g - h a_x\right)}{l} \tag{2.3}$$

where M is vehicle mass at COG, a_x is the longitudinal acceleration of COG, h is the height of COG and $l = l_f + l_r$.A normal force of single front wheel can be derived as,

$$F_{zfl} = F_{zfr} = \frac{M\left(l_r g - h a_x\right)}{2l} \tag{2.4}$$

and similarly for a single rear wheel,

$$F_{zrl} = F_{zrr} = \frac{M\left(l_f g + h a_x\right)}{2l} \tag{2.5}$$

Figure 2.2 shows the simulated variation of the normal forces F_z during braking. As it shows, the difference in normal forces between the front and rear wheels are considerable and should be taken into an account when designing the controller.

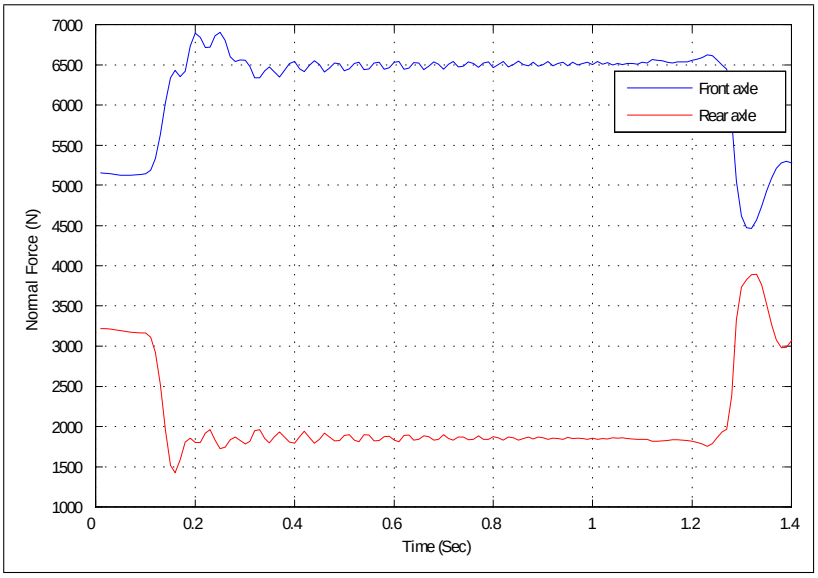


Figure 2.2: Variation of normal force (F_z) on front and rear wheels during step braking

2.2 Longitudinal Wheel Slip Dynamics

A model for longitudinal wheel slip can be derived from a simplified quarter car model shown in Figure. 2.3. Equation 2.6 describes a rotational motion of the quarter car model in the longitudinal direction with respect to vehicle velocity. A tire reaction force is generated between the tire contact patch and the road surface, which creates a torque that results in angular velocity . When the brake torque is applied to the wheel, it causes an angular deceleration to stop the wheel's rotation. In ideal conditions ABS system controls the value of slip (λ) to ensure maximum friction force between the tire and the road.

$$J\dot{\omega} = rF_x - T_b \tag{2.6}$$

$$m_q\dot{v} = -F_x \tag{2.7}$$

where

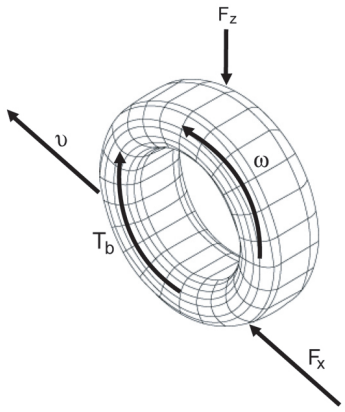


Figure 2.3: Quarter Car Model

m_q - mass of the quarter car
υ - vehicle speed
ω - wheel speed
Fz - vertical force
Fx - tyre friction force
Tb - brake torque
r - wheel radius
J - wheel inertia

The tyre friction force F_x in the above equation can not be evaluated analytically or be measured. The only way to evaluate it is through modelling and estimation, which holds one of the key aspect in ABS control development. There are vast range of tyre models from a simple model to understand the physics to a complex model that predicts the behavior precisely. Dynamic properties of the tyre is very complex, therefore, it is often not applicable in a vehicle control system. Empirical tyre models are frequently used in the area of vehicle dynamics control, as it provides a few parameters which can be determined from the testing.

Derivation of the longitudinal wheel slip dynamics is obtained by taking the derivative of the longitudinal slip Equation (1.1) with respect to time.

$$\lambda = \frac{\upsilon - \omega r}{\upsilon} \tag{2.8}$$

$$\frac{d\lambda}{dt} = \frac{\partial f}{\partial \upsilon}\frac{d\upsilon}{dt} + \frac{\partial f}{\partial \omega}\frac{d\omega}{dt} + \frac{\partial f}{\partial r}\frac{dr}{dt} \tag{2.9}$$

and assuming that the wheel radius r remains constant during braking, it can be simplified as below,

$$\dot{\lambda} = -\frac{r}{\upsilon}\dot{\omega} + \frac{\omega r}{\upsilon^2}\dot{\upsilon}$$

substituting the quarter car model into the previous Equation 1.1 yields,

$$\dot{\lambda} = -\frac{r}{\upsilon}\left(\frac{rF_x - T_b}{J}\right) - \frac{\omega r}{\upsilon^2}\left(\frac{F_x}{m_q}\right) \tag{2.10}$$

$$= \frac{1}{\upsilon}\left(\frac{1}{m_q}\cdot\frac{\omega}{\upsilon} + \frac{r^2}{J}\right)F_x + \frac{r}{J\upsilon}T_b \tag{2.11}$$

a further simplification is possible using the premise that $\frac{1}{m_q}\cdot\frac{\omega}{\upsilon} \ll \frac{r^2}{J}$,

$$\dot{\lambda} = \frac{r^2}{J\cdot\upsilon}F_x + \frac{r}{J\upsilon}T_b \tag{2.12}$$

As it was mentioned, tire friction force is nonlinear (see Figure 1.1) to simply our approach the tire friction force is linearized with respect to the change in slip (i.e. longitudinal slip stiffness) at a particular operating region. While this is not entirely representative of a real world environment it does simplify the problem domain and enable us to explore the potential benefits without the complexities of real life. Figure 2.4 shows the linearisation of the slip curve at the slip region of $\lambda_o = 0.1$ by using the slip stiffness. The slip stiffness K_x can be defined as the local gradient of the slip

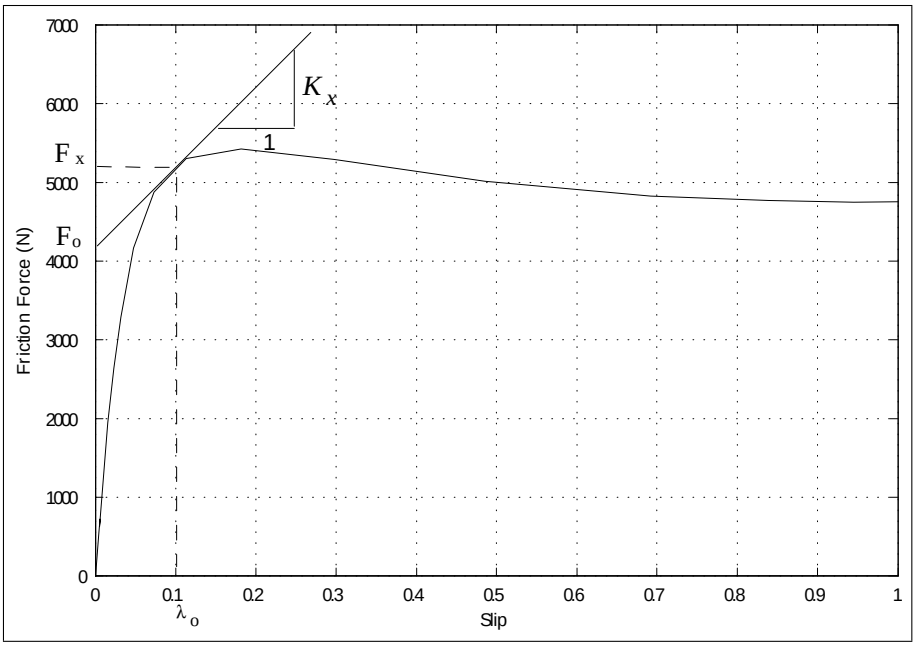


Figure 2.4: Linerisation of slip curve using slip stiffness

curve as below,

$$K_x = \frac{\partial F_x}{\partial \lambda}$$

The dynamics of the friction force F_x at an operating point can be expressed as,

$$F_x = K_x \tilde{\lambda}$$

where $\tilde{\lambda} = \lambda - \lambda_o$ representing the small deviation of the slip around the operating point λ_o. We can derive the following relationship

$$\begin{aligned} \dot{\tilde{\lambda}} &= \frac{r^2}{J \cdot \upsilon} K_x \tilde{\lambda} + \frac{r}{J\upsilon} T_b \\ &= \frac{1}{\upsilon} \left(\frac{r^2}{J} K_x \tilde{\lambda} + \frac{r}{J\upsilon} T_b \right) \end{aligned} \tag{2.13}$$

Step response of Equation (2.13) at fixed slip value of 0.1 with varying speed from 20 to 100 km/h, is shown in Figure 2.5 and also Figure 2.5 shows the step response of fixed speed with varying slip from 0.02 to 0.1. From the responses shown, response of the slip at a fixed slip value, has a tendency to be slower at high velocities and faster at lower velocities. At a given constant velocity, the response becomes slower and the steady state value increases, at the slip value is increased.

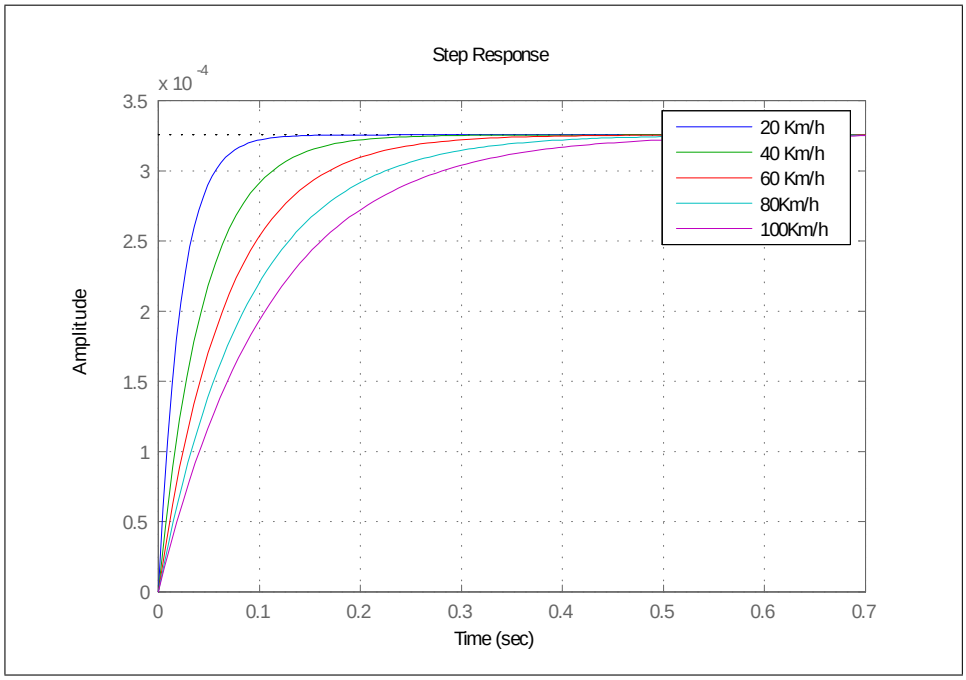


Figure 2.5: Step response of slip dynamic for speed from 20 Km/h to 100Km/h

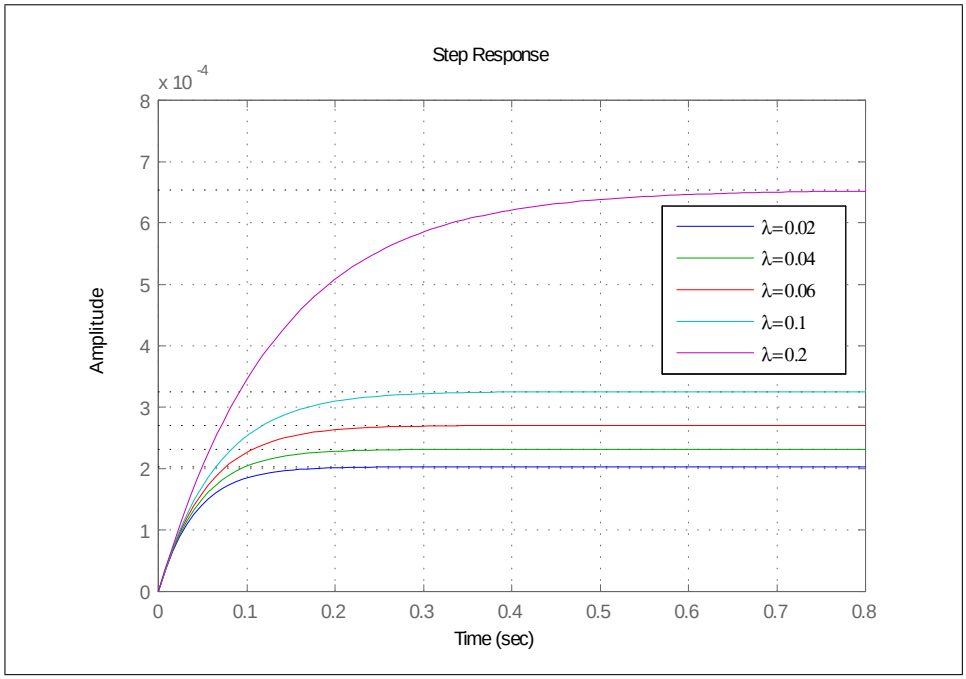


Figure 2.6: Step response of the slip dynamic at constant vehicle speed of 60 km/h for slip value from 0.02 to 0.2

The previous represents the general behavior of change in slip due to a brake torque variation, and it does not take into consideration tire deformation. Therefore, the friction force F_x indicated in the Equation (2.12) as instantaneous to the brake torque variation no longer apply, rather it is a function of time as it is depend on the deflection and the distance of the wheel traveled. The most widely known model to describe a transient response of the tire friction force is shown in Equation (2.14).

$$F_x = \frac{\sigma}{\upsilon_x}\dot{F}_x + \bar{F}_x \tag{2.14}$$

$$\dot{F}_x = \frac{\upsilon_x}{\sigma}F_x - \frac{\upsilon_x}{\sigma}\bar{F}_x \tag{2.15}$$

where

$$\bar{F}_x = \textit{Static Friction Force}$$
$$\sigma = \textit{Relaxation length}$$

The term relaxation length takes in the characteristics of the tire and is commonly described as the distance the tire must travel to build up the deflection necessary to transmit two thirds of the force. Including this dynamic behavior of the tyre friction force into the Equation (2.13) increases the overall model order which yields

the following form.

$$\ddot{\tilde{\lambda}} = \frac{r^2}{J \cdot \upsilon_x}\dot{F}_x - \frac{r}{J \cdot \upsilon_x}\dot{T}_b \tag{2.16}$$

$\dot{T}_b$ indicates that an actuator dynamics needs to be accounted for, which can be re-defined as follows.

$$\dot{T}_b = K_b\left(\tilde{T}_b - T_b\right) \tag{2.17}$$

where $\tilde{T}_b$ is the commanded brake torque, T_b is the measured torque and the K_b indicates the actuator dynamic. Substituting the above Equations (2.14) and (2.17) into the slip dynamic Equation (2.16) yields,

$$\begin{aligned}
\ddot{\tilde{\lambda}} &= \left(\frac{r^2}{J\upsilon}\right)\dot{F}_x - \frac{r}{J\upsilon}\dot{T}_b \\
&= \left(\frac{r^2}{J\upsilon}\right)\left(\frac{\upsilon_x}{\sigma}F_x - \frac{\upsilon_x}{\sigma}\bar{F}_x\right) - \frac{r}{J\upsilon}K_b\left(\tilde{T}_b - T_b\right) \\
&= \frac{r^2}{J\sigma}F_x - \frac{r^2}{J\sigma}\bar{F}_x - \frac{r}{J\upsilon}\cdot K_b\left(\tilde{T}_b - T_b\right)
\end{aligned} \tag{2.18}$$

Rarranging the Equation (2.12) to express the friction force in terms of a change in slip

$$F_x = \left(\frac{J\upsilon}{r^2}\right)\dot{\tilde{\lambda}} - \frac{T_b}{r} \tag{2.19}$$

and substituting above Equation into (2.18), it yields

$$\begin{aligned}
\ddot{\tilde{\lambda}} &= \frac{r^2}{J\sigma}\left(\left(\frac{J\upsilon}{r^2}\right)\dot{\tilde{\lambda}} - \frac{T_b}{r}\right) - \frac{r^2}{J\sigma}\bar{F}_x - \frac{r}{J\upsilon}K_b\left(\tilde{T}_b - T_b\right) \\
&= \frac{\upsilon}{\sigma}\dot{\tilde{\lambda}} - \frac{r^2}{J\sigma}\bar{F}_x - \frac{r}{J\sigma}T_b - \frac{r}{J\upsilon}\left(K_b\left(\tilde{T}_b - T_b\right)\right)
\end{aligned}$$

We can define static friction force $\bar{F}_x$ as $K_x\tilde{\lambda}$,then

$$\ddot{\tilde{\lambda}} = \frac{\upsilon}{\sigma}\dot{\tilde{\lambda}} - \frac{r^2}{J\sigma}K_x\tilde{\lambda} - \frac{r}{J\sigma}T_b - \frac{r}{J\upsilon}\left(K_b\left(\tilde{T}_b - T_b\right)\right)$$

Casting the above dynamic Equation into the state space representation yields the following form,

$$\begin{bmatrix} \dot{\tilde{\lambda}} \\ \ddot{\tilde{\lambda}} \\ \dot{T}_b \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -\frac{r^2}{J\sigma}K_x & \frac{\upsilon}{\sigma} & -\frac{r}{J\upsilon}K_b \\ 0 & 0 & K_b \end{bmatrix}\begin{bmatrix} \tilde{\lambda} \\ \dot{\tilde{\lambda}} \\ \left(\tilde{T}_b - T_b\right) \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{r}{J\sigma} \\ 0 \end{bmatrix}\begin{bmatrix} T_b \end{bmatrix} \tag{2.20}$$

Step response of the model is shown in Figure 2.7 and 2.8. Response shows that it is generally faster at lower speeds. This oscillatory behavior is more evident at low slip value and lower average speeds.

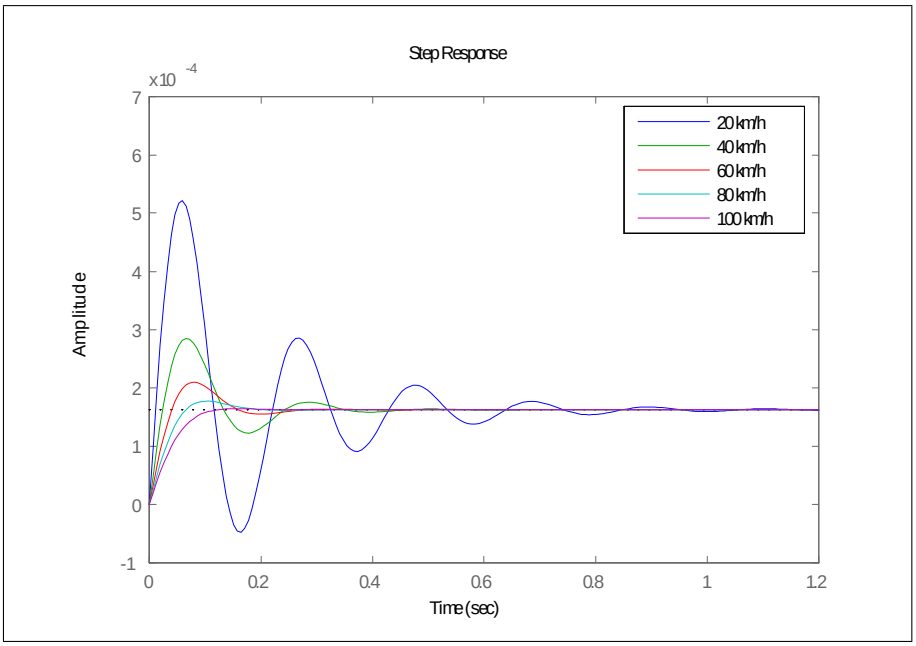


Figure 2.7: Step response of the dynamic slip at slip value of 0.1 for an vehicle speed from 20 to 100 km/h

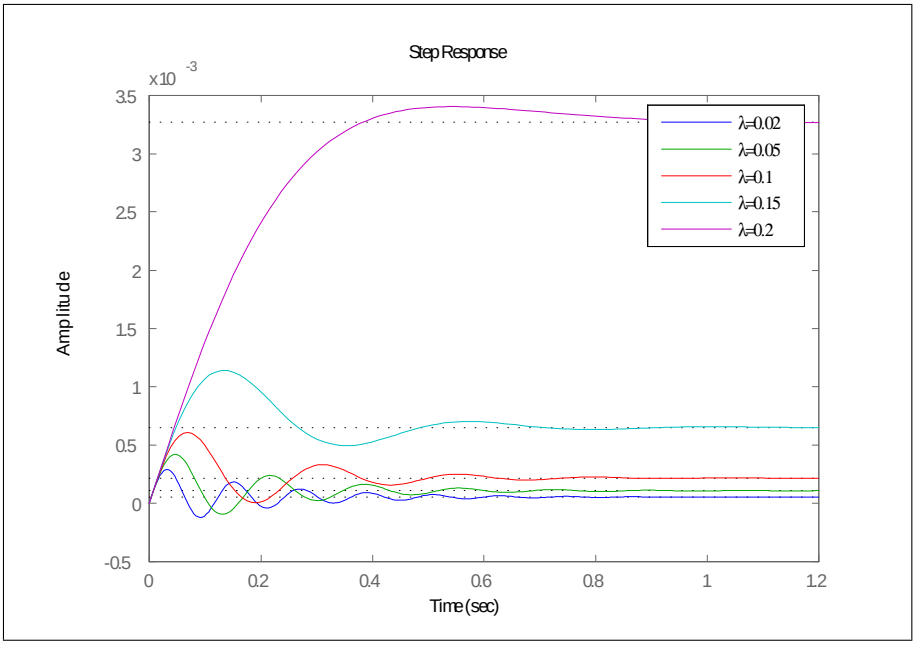


Figure 2.8: Step response of the slip dynamic at constant vehicle speed of 80 km/h for slip value from 0.02 to 0.1

Chapter 3

Wheel Slip Control System Design

An antilock brake controller must maintain the wheel slip at the optimal tyre-road friction characteristics to enhance the directional stability of a vehicle during an emergency braking manoeuvre. In this chapter, a more specific definition of the control problem and the statement of performance requirements for the wheel slip control system are presented. This chapter also covers several aspects of the wheel slip control system developed in this work, including the description of overall controller structure and the actuator constraint specification, as well as the stateflow of the control algorithm.

3.1 Overview

The primary control objective of a wheel slip control system is to keep the tyre's longitudinal slip trajectory at a given setpoint. Normally the setpoint is specified at the value close to the peak of tyre friction curve, so the steerability and lateral stability of a vehicle is maximised during hard braking. For a decentralized BBW system architecture, as shown in Figure 3.1, the wheel slip control system is designed to independently control the brake torque at each wheel. Each individual wheel slip controller (shown as WBCU) takes the measured state of the vehicle via central control unit (ECCU) and the sensed brake torque as an input to generate an appropriate control signal. The required brake torque is then transmitted through the electromechanical brake caliper (EMB), which takes the commanded control signal and converts it to a braking torque applied by the caliper. In order for the wheel control system to generate the optimum brake torque, an accurate measurement of a longitudinal slip value is fun-

damental which requires the knowledge of vehicle reference speed, as indicated by the Equation 1.1 in Chapter 1. Generally it can be assumed that a measured wheel speed value represents the vehicle reference speed when the vehicle is cruising or travelling at constant speed. However during an emergency braking, a vehicle speed can no longer be represented by the wheel speed value due to the inconsistent measurement of wheel speed values caused by the blocking of wheel. There are several approaches have appeared in the literature to estimate the vehicle speed and the implication of a real time implementation are given by (Basset *et al.* , 1997) and (Daiss & Kiencke, 1995). Since the optimal slip value changes very rapidly as the tyre contact patch translates onto ever-varying road surface, it is also vital to know the accurate estimates of the road friction coefficient which must converge rapidly despite the uncertain environment. Recent studies have investigated the application of observer/estimation schemes to obtain real-time data indirectly, the extended Kalman filter (EKF) has received increasing attention (Gustafsson, 1997). In a real application, mentioned feedback variables, such as vehicle speed (v), longitudinal wheel slip (λ) and friction coefficient (μ) must be estimated using an observer and can not be measured them directly. However, for the purpose of simulation study we assume feedback inputs of the wheel slip control system to be measurable and the optimal value of target slip to be known prior.

3.2 Model Predictive Wheel Slip Controller Design

A distributed wheel slip controller is designed (see Figure 3.2) using a generic model predictive control (MPC) method. There are three major aspects of model predictive control which make the design methodology attractive to both academics and engineers. The first aspect is the design formulation which has a complete multivariable system feature, specifically the performance parameters of the multivariable control system are related to the engineering aspects of the process, hence they can be understood and 'tuned' by engineers. This aspect is often overlooked by researchers. Nevertheless, the cost of commission and maintenance is a very important issue for a company to decide whether or not to use advanced control. The second aspect is the ability to handle both 'soft' constraints and hard constraints in the design. This is attractive to industry where tight profit margins and limits on the operation of the process are inevitably present. The third aspect is the ability to perform on-line optimization. The continuous time implementation of MPC is considered in this work, where we consider a continuous time state space model of the following form to predict the output behavior of the plant on the basis of the inputs and to compute the manipulated control signal that minimizes the objective function.

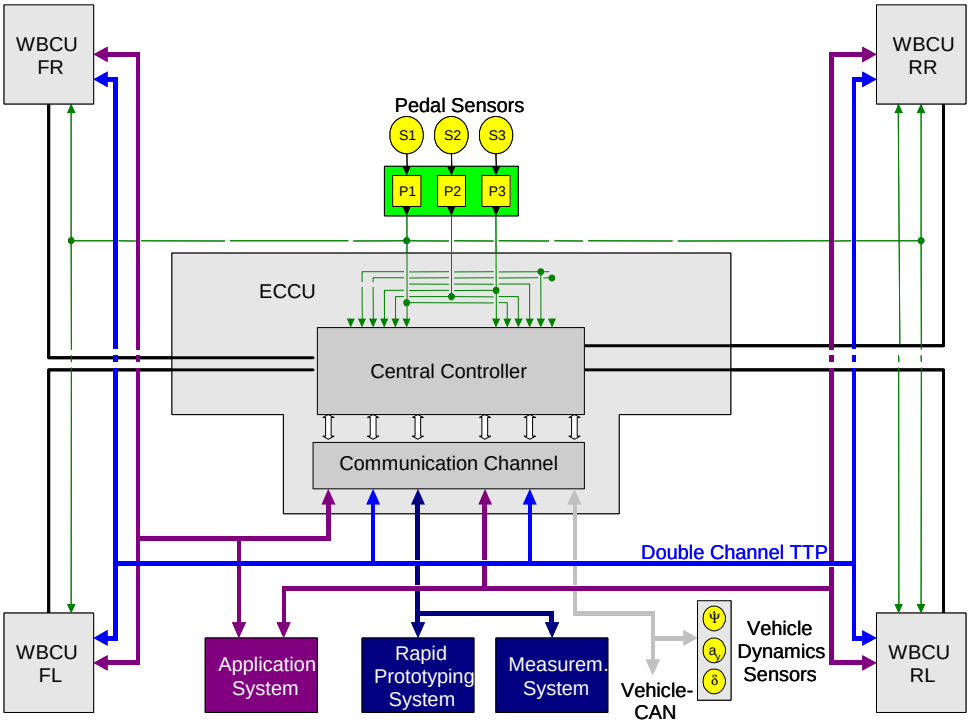


Figure 3.1: Control system architecture of BBW system

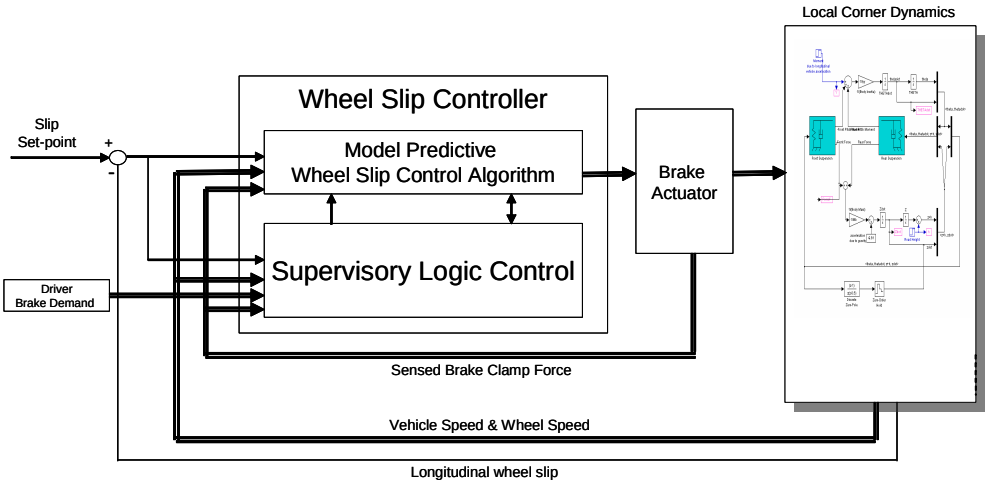


Figure 3.2: Local wheel slip controller configuration

$$\begin{aligned} \dot{X}_m(t) &= A_m X_m(t) + B_m u(t) \\ y(t) &= C_m X_m(t) \end{aligned}$$

where A_m, B_m and X_m are defined by the model (2.20) as below,

$$A_m = \begin{bmatrix} 0 & 1 & 0 \\ -\frac{r^2}{J\sigma}K_x & \frac{v}{\sigma} & -\frac{r}{Jv}K_b \\ 0 & 0 & K_b \end{bmatrix}$$

$$B_m = \begin{bmatrix} 0 \\ -\frac{r}{J\sigma} \\ 0 \end{bmatrix}, X_m = \begin{bmatrix} \tilde{\lambda} \\ \dot{\tilde{\lambda}} \\ \tilde{T}_b \end{bmatrix},$$

$$C_m = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

Furthermore, letting the auxiliary variable as $Z(t)$ as

$$Z(t) = \dot{X}_m(t)$$

State space model is rewritten in terms of $\dot{u}(t)$ into the augmented form as below.

$$\begin{bmatrix} \dot{Z}(t) \\ \dot{y}(t) \end{bmatrix} = \begin{bmatrix} A_m & 0 \\ C_m & 0 \end{bmatrix} \begin{bmatrix} Z(t) \\ y(t) \end{bmatrix} + \begin{bmatrix} B_m \\ 0 \end{bmatrix} \dot{u}(t)$$

$$y(t) = \begin{bmatrix} 0 & I \end{bmatrix} \begin{bmatrix} Z(t) \\ y(t) \end{bmatrix}$$

where I is the n x n unit matrix and $y(t)$ is the output of wheel slip. Assuming that the state variable vector $X(t_i)$, which in this case longitudinal wheel slips (λ) and brake clamp force (T_b) are available through measurement, then the prediction of future state variables $X(t_i + \tau)$ are given by the following equation.

$$\begin{aligned} X(t_i + \tau) &= e^{A\tau} X(t_i) + \int_{t_i}^{t_i+\tau} e^{A(t_i+\tau-\beta)} B\dot{u}(\beta)\, d\beta \\ &= e^{A\tau} X(t_i) + \int_0^{\tau} e^{A(\tau-\gamma)} B\dot{u}(t_i + \gamma)\, d\gamma \end{aligned}$$

Typically for a stable LTI system, the change in control variable decays to zero by the control horizon N_c (i.e. $\dot{u}(t_i + N_c) = 0$), which indicates that the plant has been successfully steered to steady state. Based on this assumption, the derivative of the control signal trajectories are described using a Laguerre function which has the Laplace transform of $l_i(t)$ as below.

$$\int_0^{\infty} l_i(t)\, e^{-st} dt = \sqrt{2p} \frac{(s-p)^{i-1}}{(s+p)^i}$$

where p is positive and often called scaling factor. From the equation above, we can derive a differential equation satisfied by the Laguerre functions. Specifically, we let $L(t) = [l_1(t)\ l_2(t) \dots l_N(t)]^T$ and $L(0) = \sqrt{2p}\,[1\ 1 \dots 1]^T$. Then the Laguerre functions satisfy the differential equation.

$$\dot{L}(t) = A_p L(t)$$

where

$$A_p = \begin{bmatrix} -p & 0 & \dots & 0 \\ -2p & -p & \dots & 0 \\ \vdots & & & \\ -2p & \dots & -2p & -p \end{bmatrix}$$

The solution of the differential equation leads to a representation of the Laguerre functions in terms of a matrix exponential function.

$$L(t) = e^{A_p t} L(0)$$

The derivative of the control signal can be described using a set of Laguerre functions as below.

$$\dot{u}(t) = \sum_{i=1}^{N} \xi_i l_i(t) = L(t)^T \eta \tag{3.1}$$

where $\eta = [\xi_1\ \xi_2\ \dots\ \xi_N]^T$ is the vector of coefficients. Hence the predicted future state at time $t_i + \tau$ can be rewritten as below.

$$X(t_i+\tau) = e^{A\tau} X(t_i) + \int_0^{\tau} e^{A(\tau-\gamma)} B_i L_i(\gamma)^T \ d\gamma\eta$$

where i corresponds to the each control input of four wheels. Finally, the wheel slip output can be expressed as.

$$y(t_i + \tau) = CX(t_i + \tau) \tag{3.2}$$

Based on the model prediction, the design objective is to find the control input $\dot{u}(t)$ to minimize the quadratic cost function.

$$J = \int_0^{T_p} [r(t_i + \tau) - y(t_i + \tau)]^T Q [r(t_i + \tau) - y(t_i + \tau)]\, d\tau + \int_0^{T_p} \dot{u}(\tau)^T R \dot{u}(\tau)\, d\tau$$

where the term $r(t_i + \tau)$ is the optimum wheel slip value and positive definite or semi-definite weighting matrices Q and R . In order to find the optimal solution to the above cost function, the term $y(t_i + \tau)$ is substituted by its model prediction (3.2),

and the input is described by the Laguerre function (3.1), which then leads to the following form.

$$J = \int_0^{T_p} \left[r\left(t_i+\tau\right) - Ce^{A\tau}X(t_i) - \varphi(\tau)\eta\right]^T \\ Q\left[r\left(t_i+\tau\right) - Ce^{A\tau}X(t_i) - \varphi(\tau)\eta\right]d\tau + \eta^T R\eta$$

where

$$\varphi(\tau) = \int_0^{\tau} e^{A(\tau-\gamma)}B_i L_i\left(\gamma\right)^T d\gamma$$

and by letting $\omega(t_i+\tau) = r(t_i+\tau) - Ce^{A\tau}X(t_i)$

$$J = \eta^T\{\int_0^{T_p}\varphi(\tau)Q\varphi(\tau)^T\eta + R\}\eta - 2\eta^T\int_0^{T_p}\varphi(\tau)Q\omega(t_i{+}\tau)d\tau + \rho \qquad (3.3)$$

where $\rho = \int_0^{T_p}\omega\left(t_i+\tau\right)^T Q\omega\left(t_i+\tau\right)d\tau$, which are not dependent on control input $u(t)$. Thus the optimal control input for unconstrained cases can be found by solving the least square problem as below where we assume that the optimum wheel slip is to be constant within the prediction horizon.

$$\eta = \Pi^{-1}\left\{\Psi_1 r\left(t_i\right) - \Psi_2 X\left(t_i\right)\right\}$$

where

$$\begin{aligned}\Pi &= \int_0^{T_p}\phi\left(\tau\right)Q\phi\left(\tau\right)^T d\tau + \bar{R}\\ \Psi_1 &= \int_0^{T_p}\phi\left(\tau\right)Qd\tau\\ \Psi_2 &= \int_0^{T_p}\phi\left(\tau\right)QCe^{A\tau}d\tau\end{aligned}$$

Applying the receding horizon control, the only the first control signal (i.e. the derivative control signal at $\tau = 0$) is used, hence the brake torque can be constructed as follows.

$$\begin{aligned}u\left(t_i\right) &= \int_0^{T_p}\dot{u}(t)dt\\ &\approx u(t_i{-}\Delta t) + L_1(0)^T\eta\Delta t\end{aligned}$$

In the presence of constraints, optimization problem requires to be formulated the problem into quadratic programming form. In this work, only the saturation limit of EMB calipers are considered which can be specified as below.

$$T_b^{\min} \le u\left(t\right) \le T_b^{\max}$$

where the maximum brake force ($T_b^{\max}$) generated by EMB is specified at $30kN$, and maximum brake torque applied to the rear axle is limited to half of the brake torque applied at the front axle (i.e. $15kN$). By using the definition of $\dot{u}(t)$, the above inequality constraints can be expressed in terms of parameter η as below.

$$u_{low}(t_i+\tau) < \int_0^{\tau_i} L_1(\gamma)^T d\gamma\eta + u(t_i-\Delta t) < u_{high}(t_i+\tau)$$

where $u(t_i-\Delta t)$ denotes the previous control signal. In the presence of constraints on brake torque actuation, the predictive control problem can be re-expressed as below.

$$\begin{aligned} \text{minimize } J &= \frac{1}{2}\eta^T\Pi\eta - \eta^T\{\Psi_1 r(t_i) - \Psi_2 X(t_i)\} \\ \textit{subject to} &: M_1\eta \le \gamma_1 \end{aligned} \tag{3.4}$$

where
$M_1 = \begin{bmatrix} -\Gamma \\ \Gamma \end{bmatrix}; \gamma_1 = \begin{bmatrix} -u_{low} + u(t_i-\Delta t) \\ u_{high} - u(t_i-\Delta t) \end{bmatrix}$

There are three categories of methods in solving the above equation. These are primal methods, dual methods and the primal-dual methods. In this work, primal-dual method of Hildreth's Quadratic Programming algorithm is chosen to provide the numerical solution to the constrained optimal problem, and the Equation 3.4 is written into the equivalent dual problem as below

$$\max_{\lambda\ge 0}\min_{\eta}\frac{1}{2}\eta^T\Pi\eta - \eta^T F + \lambda^T(M\eta-\gamma) \tag{3.5}$$

where $F = \{\Psi_1 r(t_i) - \Psi_2 X(t_i)\}$and the minimisation over η is given by

$$\eta = -\Pi^{-1}\left(F - M^T\lambda\right) \tag{3.6}$$

Substituting the above Equation into (3.5), the dual problem becomes

$$\max_{\lambda\ge 0}\frac{1}{2}\lambda^T H\lambda - \lambda^T K - \frac{1}{2}F^T E^{-1}F$$

where $H = M\Pi^{-1}M^T$ and $K = \gamma + ME^{-1}F$.Equivalently the above equation can be written in the form as below

$$\min_{\lambda\ge 0}\frac{1}{2}\lambda^T H\lambda + \lambda^T K + \frac{1}{2}F^T E^{-1}F$$

To solve above dual problem using Hildreth's Quadratic Programming algorithm, the direction vectors are selected to be equal to the basis vectors $e_i = (0,0,\ldots,1,\ldots 0,0)$. Then the λ vector can be varied one component at a time. At a given step in the process, having obtained a vector $\lambda \ge 0$, we fix our attention on a single component

λ_i. The objective function may be regarded as a quadratic function in this single component. We adjust λ_i to minimize the objective function. If that requires $\lambda_i < 0$, we set $\lambda_i = 0$.. In any case, the objective function is decreased. Then we consider the next component λ_{i+1}. If we consider one complete cycle through the components to be one iteration taking the vector λ_m to λ_{m+1}, the method can be expressed explicitly as

$$\lambda_i^{m+1} = \max\left(0, w_i^{m+1}\right) \tag{3.7}$$

where

$$w_i^{m+1} = -\frac{1}{h_{ii}}\left[k_i + \sum_{j=1}^{i-1} h_{ij}\lambda_j^{m+1} + \sum_{j=i+1}^{n} h_{ij}\lambda_j^m\right]$$

Hildreth's Quadratic Programming Procedure is easy for computer programming and the convergence of the algorithm can also be proved. However, a drawback of this extreme simple algorithm is that the convergence rate is too slow for the nonnegative λ_{iS}, depending on the distribution of the eigenvalues of H matrix, which is similar to other gradient based methods. It seems that for the negative and zero lies, the convergence rate is fast, in which several case studies have found that reliable estimation of the inactive constraints can be obtained with 10 iterations. Because Hildreth's Quadratic Programming algorithm converges, given a sufficient number of iterations, the algorithm will provide reliable results. This simple algorithm is used to identify the set of inactive constraints. By deleting the inactive constraints, the original inequality constraint problem becomes an equality constraint problem, and both η and the nonnegative λ_{iS} can be calculated in a closed form solution. Let the set of active constraints be represented by M_{act} and γ_{act}. The optimal solution to the constrained control problem is given by the solution of the linear equations

$$\begin{bmatrix} E & M_{act}^T \\ M_{act} & 0 \end{bmatrix}\begin{bmatrix} \eta \\ \lambda_{act} \end{bmatrix} = \begin{bmatrix} -F \\ \gamma_{act} \end{bmatrix} \tag{3.8}$$

Explicitly:

$$\lambda_{act} = -\left(M_{act}\Pi^{-1}M_{act}^T\right)^{-1}\left(\gamma_{act} + M_{act}\Pi^{-1}F\right) \tag{3.9}$$

$$\eta = \Pi^{-1}\left(F - M_{act}^T\lambda_{act}\right) \tag{3.10}$$

If the number of active constraints is greater than the number of decision variables, the matrix $M_{act}E^{-1}M_{act}^T$ contains zero eigenvalue(s), hence becomes singular. As a result, we can check whether the number of rows of M_{act} is less than the number of columns as the first step to determine whether the Hildreth's programming procedure has converged. If so, we proceed to calculate λ_{act} and η using Equations (3.9) and (3.10). However, if one or more elements in λ_{act} turn out to be negative in the closed

form solution, a remedy is to go back to the Hildreth programming for recalculating the set of inactive constraints.

3.3 Control State Logic

Activation of the wheel slip controller is only required when the wheel slip reaches a pre-defined threshold for a given road condition. It is crucial at this point that the controller takes a command in setting the brake torque that will prevent the wheel from locking up while ensuring optimal braking force. The proposed wheel slip control system utilizes instantaneous slip (λ^*), vehicle speed (υ), and driver brake demand (F_{demand}) to detect and control wheel blocking, while ensuring the correct transition between the controller states, as shown in Figure 3.3. Activation of the slip control logic ("On") can only be realized when the driver commands a brake force resulting in higher slip than the set-point (λ_r). Once activated, controller automatically steps in taking in charge of braking and applies corrected brake forces to prevent wheel block all while maintaining the optimal slip value. An in-built safety requirement for the controller is that it is only allowed to lower the brake force demanded by the driver partly to ensure that the driver intuitively feels as though they remain in control. The controller continue to remain active until either the vehicle speed falls below a minimum specified speed ($\upsilon_{\min}$) or the release / reduction of brake demand by the driver (F_{demand}).

3.4 Control Algorithm Structure

The state flow diagram of wheel slip controller is outlined in Figure 3.4 and Figure 3.5, where Figure 3.5 shows the summary of key steps involved in the constrained predictive control algorithm.

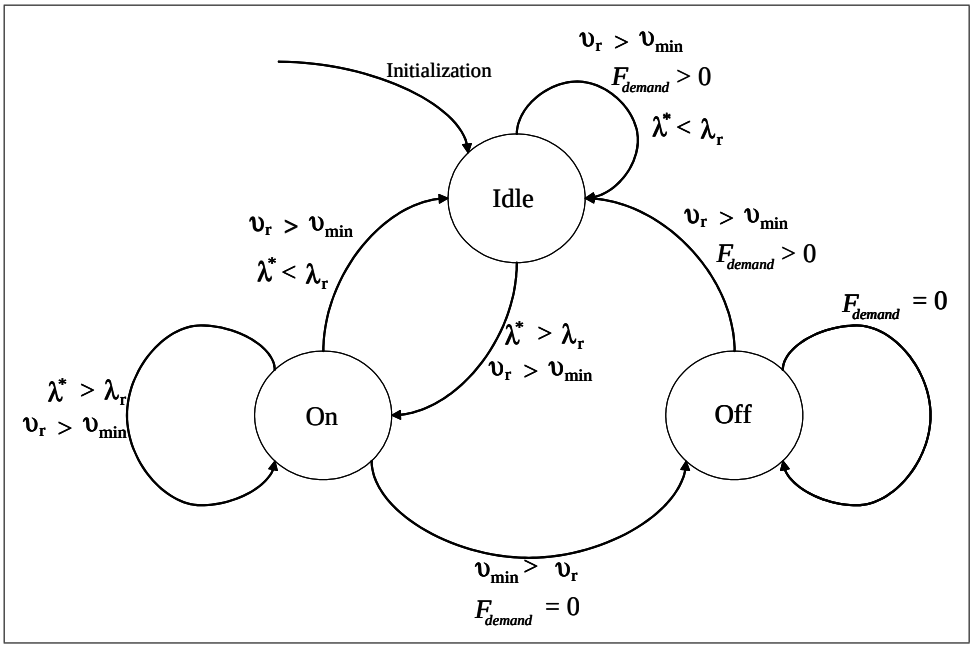


Figure 3.3: Wheel slip control state machine: Instantaneous slip (λ^*), Slip threshold (λ_r), Vehicle speed (υ_r), Minimum vehicle speed ($\upsilon_{\min}$), Driver force demand (F_{demand}).

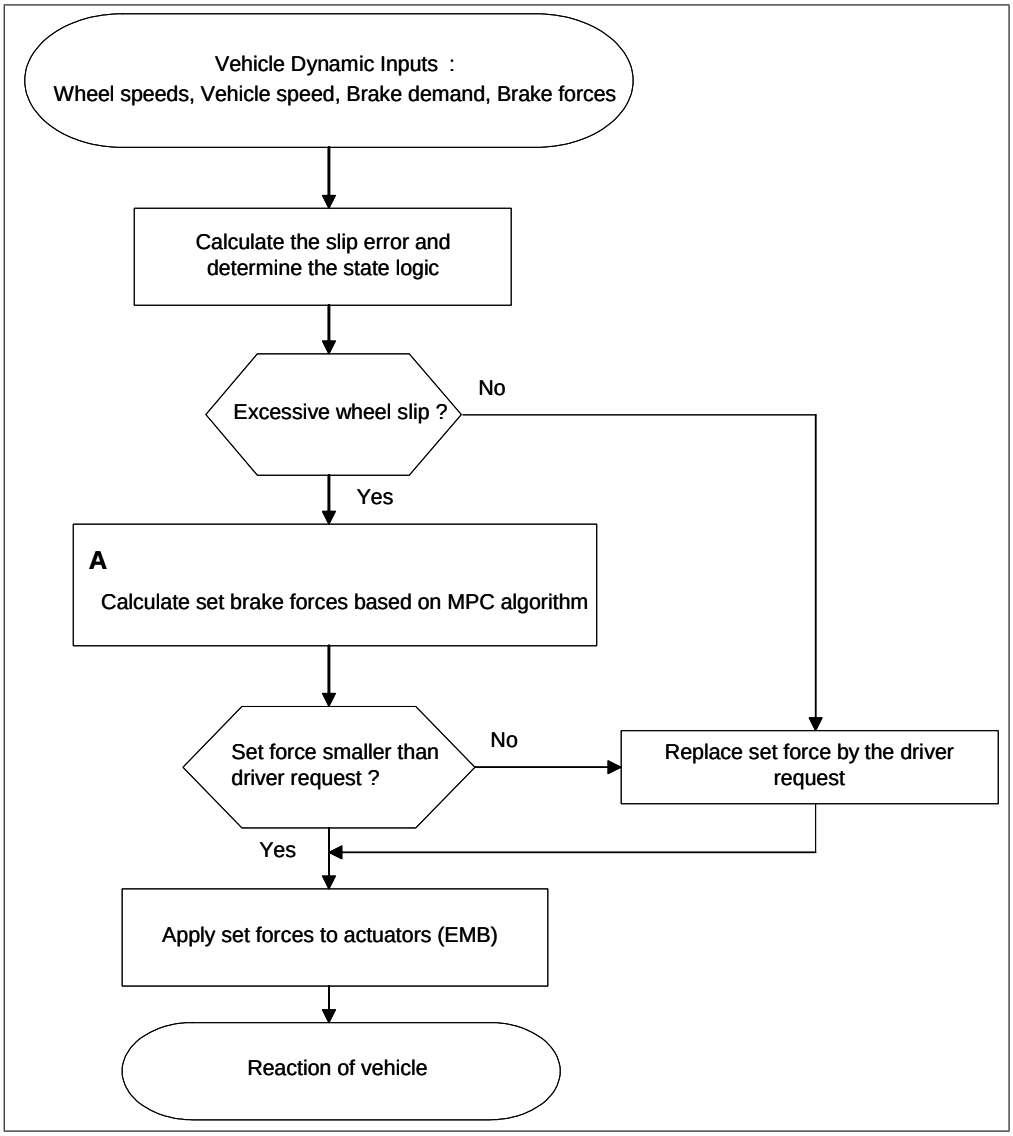


Figure 3.4: Wheel slip control algorithm structure

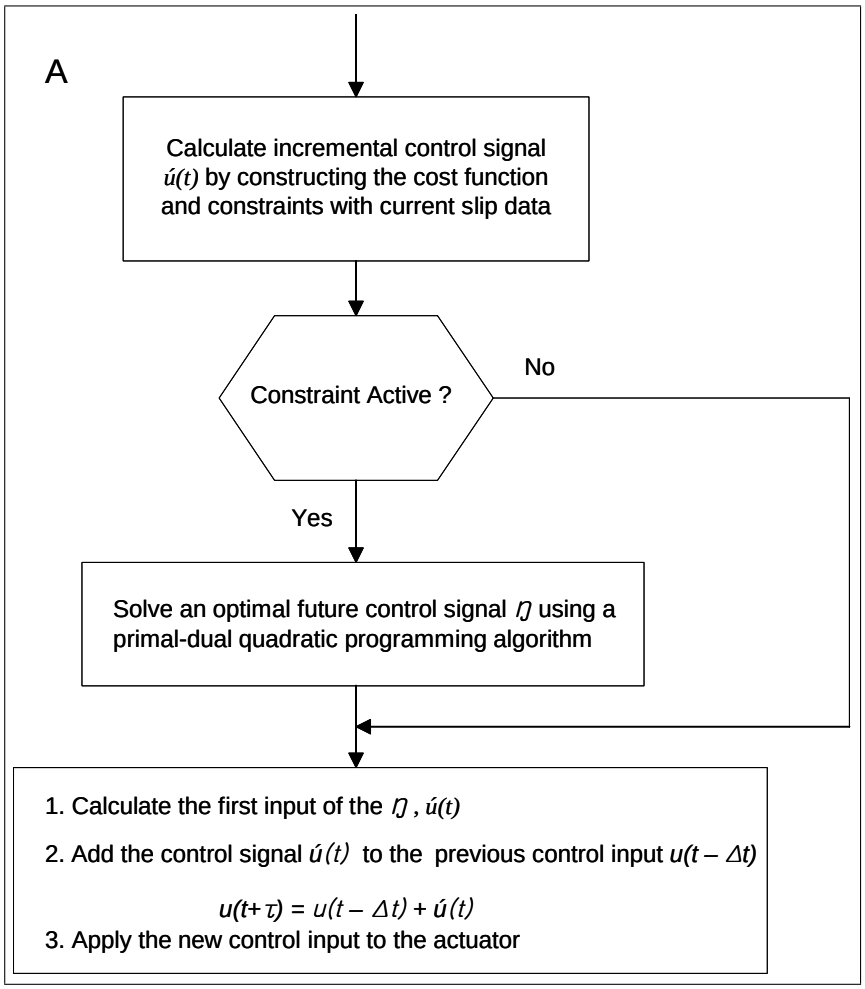


Figure 3.5: Model Predictive Control Strategy : incremental control signal $\dot{u}(t)$, future control signal η, control signal $u(t)$.

Chapter 4

Software-in-the-Loop Simulation

The first part of this chapter proposes the distributed simulation framework which combines the distributed processors in a tightly coupled configuration using the reflective memory network. The framework comprises of scalable and manageable hardware components using off-the-shelf products. Subsequent Sections 4.1.1 and 4.1.2, explain both hardware and software aspects of the proposed simulation methodology, including the parallelisation of the nonlinear vehicle dynamics model.

The final section of the chapter presents the validation of the wheel slip control which is carried out based on a commercially available nonlinear vehicle dynamics simulation model. In section 4.3, describes the tuning process of the wheel slip control algorithm for achieving the optimal brake performances. Section 4.4 presents the result of antilock brake performances of the tuned wheel slip predictive controller. A braking manoeuvre is performed on a high to low friction surface and the state logic described in Chapter 3 is implemented to prevent the wheel lock-up due to the sudden braking input.

4.1 Distributed Simulation Environment

4.1.1 Networking of Distributed Processors via Reflective Memory

The most challenging task in a distributed simulation system is the interconnections between the distributed computers. These interconnections have to be configured in a way that inter-processor communications should not affect the performance of the overall simulation.

Typically, a distributed system is configured in two ways; loosely coupled system and tightly coupled system.
In a loosely coupled system, the local area network (LAN) has been the most widely used topology to interconnect the distributed computers. Foreseen benefits of the LAN topology are summarised as below.

- A large number of distributed computers can be interconnected.
- It allows to couple the multiple computers in the distance of hundreds meters apart.

However, the speed of data transmission is slower and indeterministic when compared to a tightly coupled system. More importantly, a large software overhead is needed to ensure the synchronisation of transmitted messages. Based on this limitation, a loosely coupled system is often found to be unsuitable for the real time implementation.

As an alternative configuration, a tightly coupled system is available to overcome some of the real-time limitation in a loosely coupled system. In this configuration, processors are interconnected usually through physical shared memory architecture and utilise a high speed parallel memory bus in which all computers can access a common memory module. This architecture virtually requires no software overheads for data communication, hence guarantees requisite data communication speed and has a tighter control over the distributed system. However, the physical shared memory architecture also has its following limitation

- Memory bus contention
- Limited physical separation distance between computer.

As it can be seen, both tightly coupled and loosely coupled systems have the limitation in fulfilling the requirement of scalability and real time constraints. Hence for these reasons distributed real time system proposed in this work utilises the third alternative approach, namely reflective memory network.

This reflective memory network architecture offers the combined strength of both the physical shared memory and LAN. Concept of reflective memory network is rather simple. It operates by placing a reflective memory network card in each computer, The network card communicates over a serial ring architecture via fiber optic links

operating at very high data rates. Reflective memory networks obviate the need for time-consuming and non-deterministic protocols found in the message passing network. Data transfer is very quick and direct, within microseconds. All other CPUs on the network have the same value of the variable in their shared memory area, hence the reflective memory network provides a deterministic performance. Based on the above advantages of reflective memory networking, the distributed real-time simulation cluster is constructed using seven real time processors and reflective memory PCI cards with optical fibre link between the processors, (see Figure 4.1).

Figure 4.1: Picture of Real-time Simulation Cluster: comprises of 7 real-time simulation units constructed based on PC, reflective memory network, including reflective memory network switch and host PC

A detailed view of a local real-time simulation unit constructed based on a low cost standard PC with a reflective memory PCI card is shown in Figure 4.2.

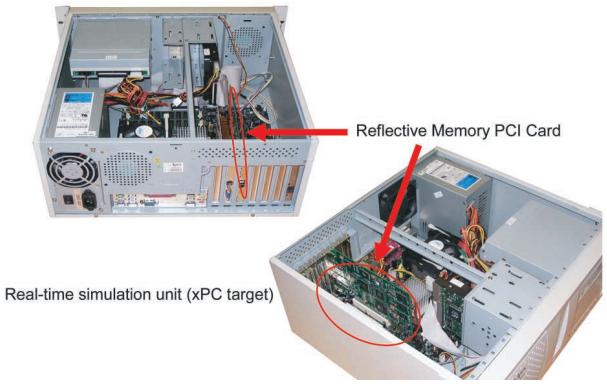


Figure 4.2: Fitted view of reflective memory card in real-time simulation unit

4.1.2 Synchronous distributed real-time simulation

Based on the concept of a reflective memory network and a distributed simulation platform, affiliated software components are designed to provide the computational methods and the administrative process for concurrent simulation, which features the simultaneous start and stop of the subsystem simulation and the alteration of distributed parameters online. Also, a partially automated process of model decomposition is implemented to allow an easy exchange of subsystems and coupling of different technical components.

In this work, we take a commercially available vehicle simulation model (ADVANCE) as an application example for the proposed distributed simulation framework. Overview of the vehicle model is shown in Figure 4.3.

Figure 4.4 shows the decomposed vehicle simulation model for the proposed distributed simulation platform. As shown in the Figure, the decomposed vehicle simulation model is distributed over six real-time simulation nodes. A master node is also fitted with a reflective memory card for overseeing the interactions between the distributed real-time simulation nodes. A host computer compiles the numerical simulation model to the real-time implementation format, which then downloads to the real-time simulation units via Ethernet (TCP/IP). In this work xPC Target which is a toolbox for MATLAB and Simulink, is used to compile the graphical Simulink models which contains blocks to interface with the specific hardware I/O such as reflective memory, and to download it to the real-time simulation node running the proprietary xPC operating system for real-time execution.

When the sequentially simulated model is decomposed in a distributed simulation en-

vironment, the execution list of an original model is no longer controlled by the central parser. This implies that the execution list must also be subdivided and implemented by the local parser in a distributed model. However, the sequence of model execution must be consistent with an original model to maintain causality and synchronisation between the distributed models.

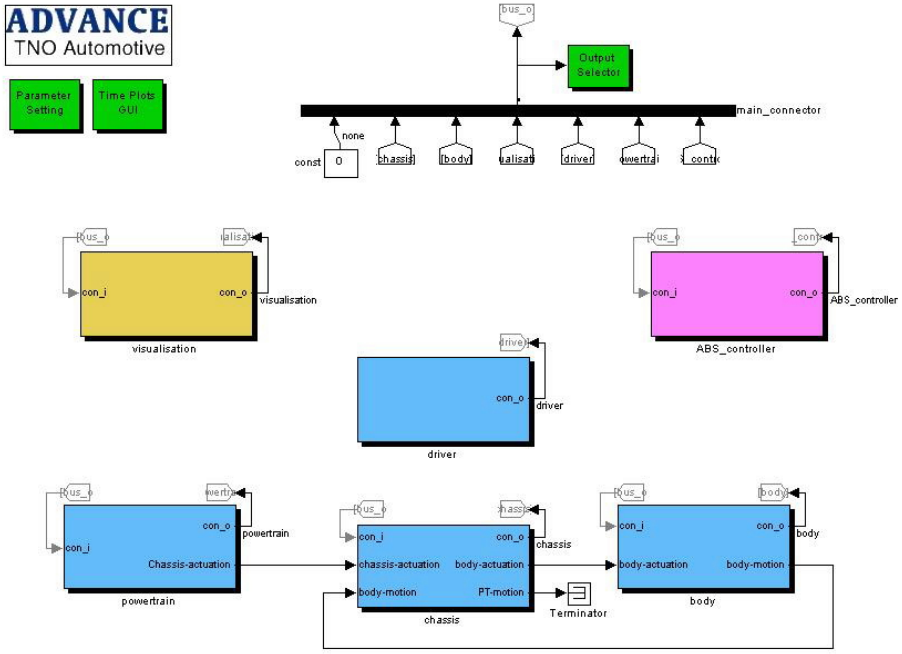


Figure 4.3: ADVANCE vehicle model

The sequential execution list of the ADVANCE vehicle model is shown in Figure 4.5 From the execution list above, a proposition can be made that the subsystems of each corner model can be regarded as concurrently executed subsystems. This closely represents the dynamics of a true vehicle. The order of execution of these subsystems do not affect the result of simulation. A revised execution list for the distributed simulation is shown in Figure 4.6

Based on the above execution list, the following procedure is implemented to maintain the correct execution of the distributed model and to ensure the synchronisation between the modules.

1. During the initialisation step, a global task manager in the master node sends the requests to check if all the node has been correctly initialised and a local

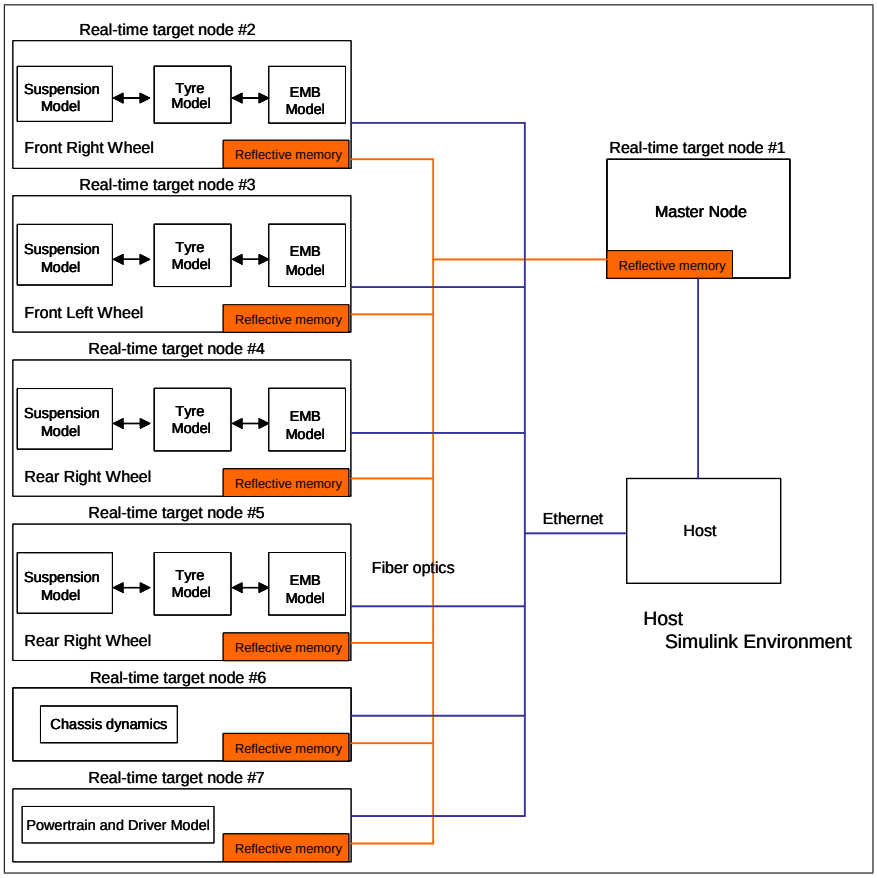


Figure 4.4: Overview of the distributed vehicle simulation model

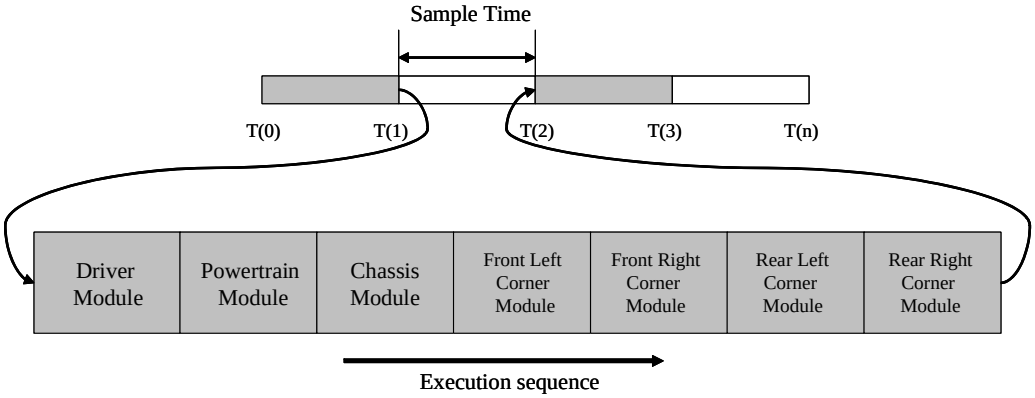


Figure 4.5: An original execution sequence of the vehicle model

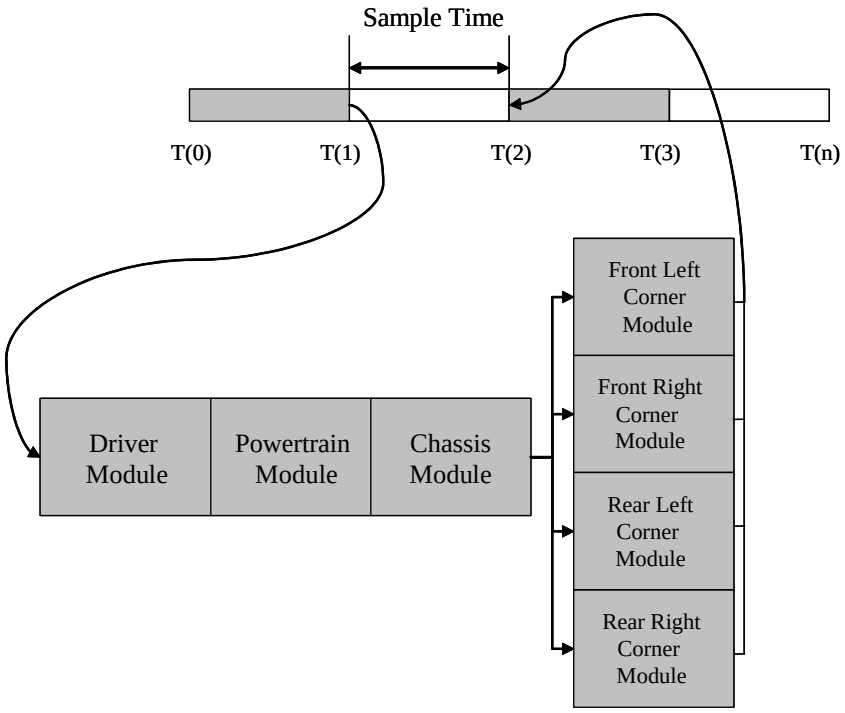


Figure 4.6: A revised execution list of the distributed vehicle model.

task manager in each local node initialises the distributed variables which contains the initial conditions for the dynamic components in the model. Once the initialisation is completed, the local task manager sends the reflective memory network message with designated node number. A master node checks the received node number, if there is an error then the master node resends the message to re-initailse. This step is necessary to validate the correct initialisation of the models.

2. After the initialisation, each simulator is placed in a "Triggered start wait" mode. In this mode, each local node waits for a triggered start message by the master node and only begins reading and writing I/O signals, but does not begin executing simulation code

3. A “Triggered start” message is used to begin all nodes simultaneously, and this messages is broadcasted to the reflective memory network by the master node according to the execution list. For instance, a global task manager broadcasts the network message which contains the node and the task number. It is then for the local event scheduler in a corresponding node to simulate the sub components of the distributed model and to update the dynamic variables.

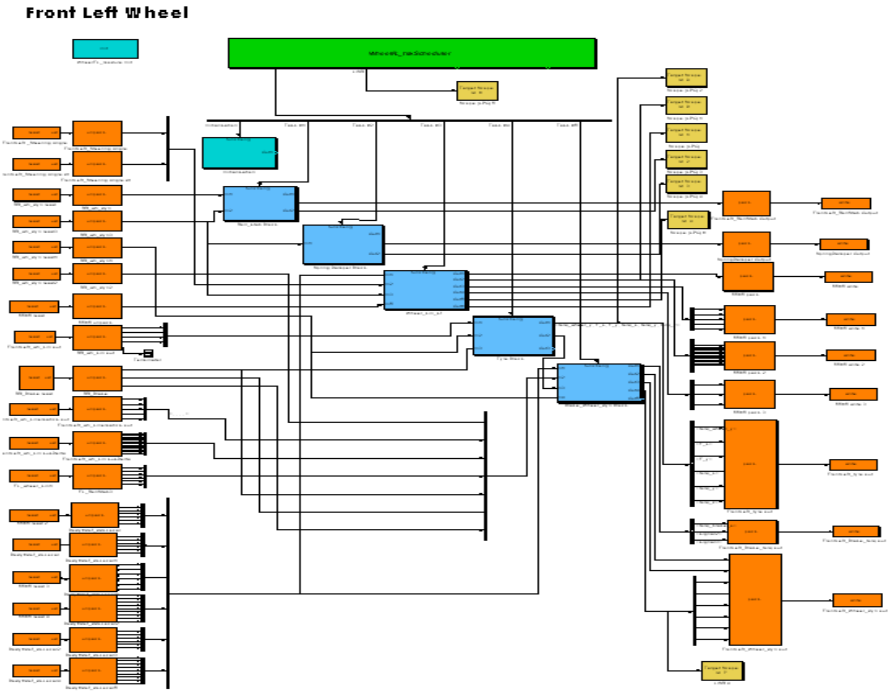


Figure 4.7: Decomposed simulation model : the front left corner dynamics

Figure 4.7 and 4.8 show the distributed model of front left corner and the chassis model. Reflective memory communication (orange blocks) runs in the background, and the local task manager (green block) which contains the local parser is connected to the triggered subsystems to execute the sub components of the model.

Lastly, to facilitate the easy data manipulation, data logging and to provide 3-d visualisation, reflective memory device driver is programmed to map the reflective memory into the standard windows operating system memory. A customised blockset of the reflective memory driver library is implemented and shown in the Appendix A.

4.2 Simulation testing conditions and scenarios

The performance of the wheel slip controller is evaluated through a nonlinear ADVANCE vehicle simulation model with validated parameter set of small passenger vehicle, including the Magic Formula tyre parameters set of "Continental Eco Contact 175/65/1". Nominal vehicle parameter values used in the simulation are shown in Table 4.1, and Figure 4.9 shows the detailed structure of the MPC based wheel slip controller. The slip controller contains three sub-blocks where the "MPC Wheel Slip Controller" block provides the formulation of the model prediction and quadratic cost

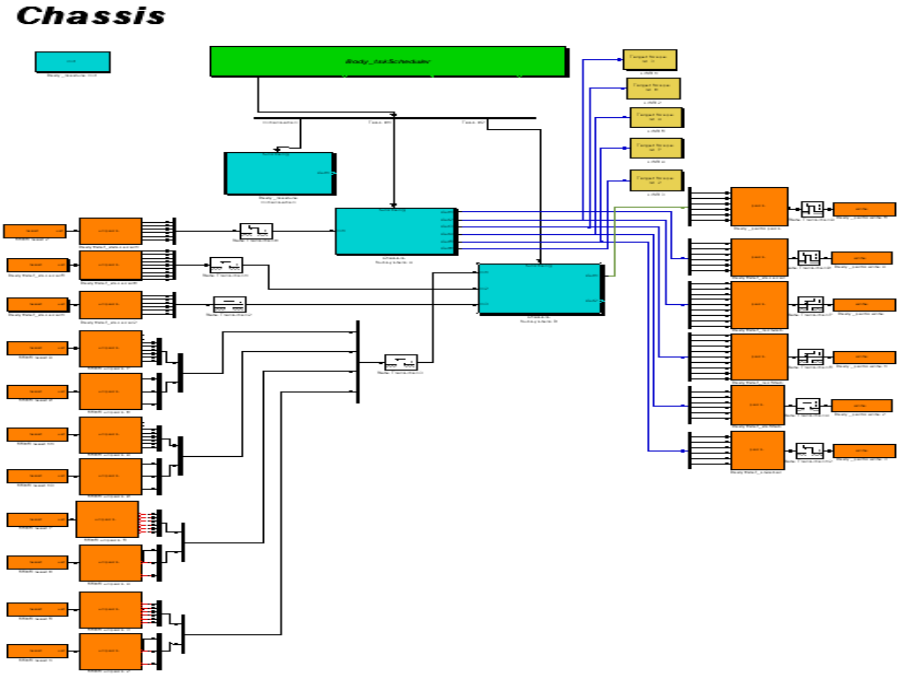


Figure 4.8: Decomposed simulation model : Chassis dyanmics

function based on given states, the "Hidreth's Quadratic Programming" block implements the constrained optimisation routine in Section 3.2 , and the "Supervisory logic" block implements the supervisory logic to ensure correct activation and transition of the states in the algorithm.

Unsprung vehicle mass	956 kg
Wheel mass	26 kg
Wheel inertia	0.78 kgm^2
Unloaded wheel radius	0.297 m

Table 4.1: Vehicle parameters used in the simulation

Simulation validation of the controller is carried out in two separate processes. First, the proposed wheel slip control algorithm is tuned for different road surfaces for tracking the reference slip trajectory, as shown in Table 4.2. Tuning procedure is carried out based on the guidelines mentioned in (Wang, 2001), and the algorithm has the following design parameters which affects the closed-loop performance of the controller.

- Pole location p : p is the pole location of the Laguerre model for the future control signal. Larger the value of p, the larger initial response of the future control

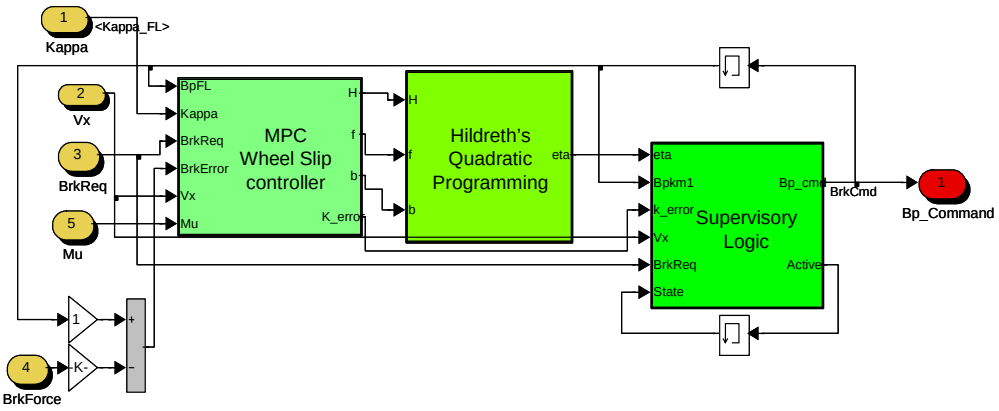


Figure 4.9: Simulink representation of the local wheel slip controller

signal, hence results in faster closed loop response speed. In the simulation studies p value is chosen in the range of 0.2..0.4.

- Prediction Horizon T_p: The prediction horizon is recommended to be chosen to be equal to the open loop settling time. In this case, T_p is set to 120 for a high and medium friction surface whereas T_p is set to 80 for a low friction surface.
- Parameter N : The parameter N is the number of terms used in capturing the control signal. For a relatively complex system, it i recommended to have a high value of N. High value of N results in more aggressive signal.

In order to ease the complexity of the tuning procedure, weighting matrices Q and R, as described in Chapter 3, are assumed to be identity matrix and the parameter N is set to a constant value of 3 ($N = 3$). The parameter p is only considered and adjusted. Lastly, based on the tuned parameters, performance validation of the controller is carried out on an emergency braking manoeuvre where a panic braking input is emulated by the driver model.
For both simulations, results are presented as follows.

1. Plot of vehicle speed and wheel speed of each wheel.
2. Plot of longitudinal slip and the set-point slip (dashed line).
3. Plot of clamp force generated by the EMB model.

4. Plot of friction curve based on the wheel slip and friction force data for each wheel.

Test No.	Surface condition (μ)	Slip setpoint (λ^*)
1	High friction surface ($\mu = 0.85$)	0.1
2	Medium friction surface ($\mu = 0.5$)	0.08
3	Low friction surface ($\mu = 0.2$)	0.04

Table 4.2 :Surface condition and the corresponding optimal slip value

4.3 Tuning Procedures for the Wheel Slip Control Algorithm

The controller tuning process for the low friction surface is deliberately omitted, due to the similar trend of responses that can be found in a medium friction surface.

4.3.1 Case A : High friction surface ($\mu = 0.85$)

For a high friction homogeneous surface ($\mu = 0.85$), the controller is tuned around the optimal slip region (λ^*). Figure 4.10 shows the front longitudinal slip responses and the commanded clamp forces. Figure clearly shows the difference in the control signal trajectory with varying values of the parameter p. It is seen that as p increases the closed loop response of the longitudinal slip also increases and becomes more oscillatory with a slight overshoot. Further observation of the vehicle responses is made in the following to study the effect of this tunable parameter p on a overall braking response. Plot of vehicle speed and wheel speeds values obtained from the nonlinear vehicle model is shown in Figures 4.11,4.12 and 4.13. It can be seen that a more aggressive control action with higher p value slightly reduces the total braking time, however it also induces the significant variability of slip responses in a lower speed region which may affect the stability of the vehicle. This may viewed as larger p corresponds to higher gain in control and less robust with respect to model uncertainty. Figure 4.14 shows the plot of longitudinal slip against the braking force between the tyre and the road. This plot graphically illustrates the slip performance with respect to the optimal slip region for three different control actions. Subplot (a) shows the overall slips performance with $p = 0.2$ and as we expected it corresponds to a slow initial response and conservative control action. Subplot (b) shows the response of $p = 0.4$ and a better slip performance of tracking maximum friction force can be observed. Subplot (c) shows the slip performance from aggressive control action ($p = 0.8$), as it

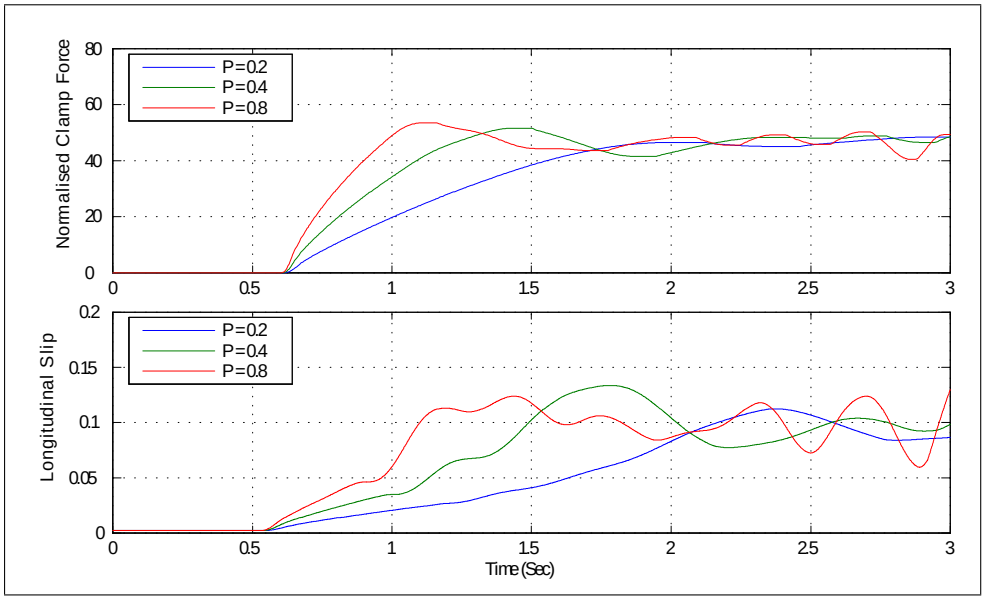


Figure 4.10: Commanded clamp forces (top) and longitudinal slip response of a front left wheel (bottom) on a high friction surface ($\mu = 0.85$) with varying value of parameter p

indicated before, output response is more oscillatory and response of rear wheel slip becomes unstable.

4.3.2 Case B : Medium friction surface ($\mu = 0.5$)

Similar response in longitudinal slip and the trajectory of the control signal is observed with higher value of parameter p, invoking a more aggressive control action and a faster slip response, (see Figure 4.15). However, the control signal trajectory and the slip responses are more oscillatory with a larger overshoot for all parameters of p. This is mainly due to the significantly less friction forces is available and the higher slip stiffness value, as we have demonstrated in the step response of wheel slip dynamic model in Chapter 2. The above observations are highlighted in the vehicle and wheel speed responses, shown in Figure 4.11, 4.12 and 4.13. Even with the modest value of p, a large oscillation of the wheel speed is observed and for the highest value of p response becomes more oscillatory and leads to a longer braking time. Figure 4.19 summarises the above findings where the conservative control action by the controller ($p = 0.2$), (see subplot (a)), achieves a more desirable result of regulating the slip value within the optimal region than the case with a higher value of p.

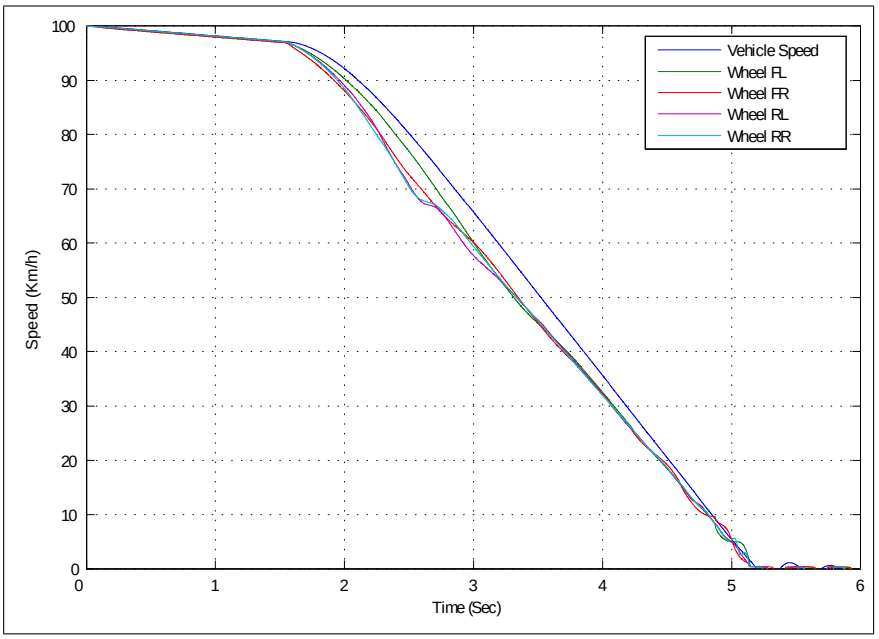


Figure 4.11: Plot of vehicle and wheel speeds on a high friction surface ($\mu = 0.85$) with with p value of 0.2

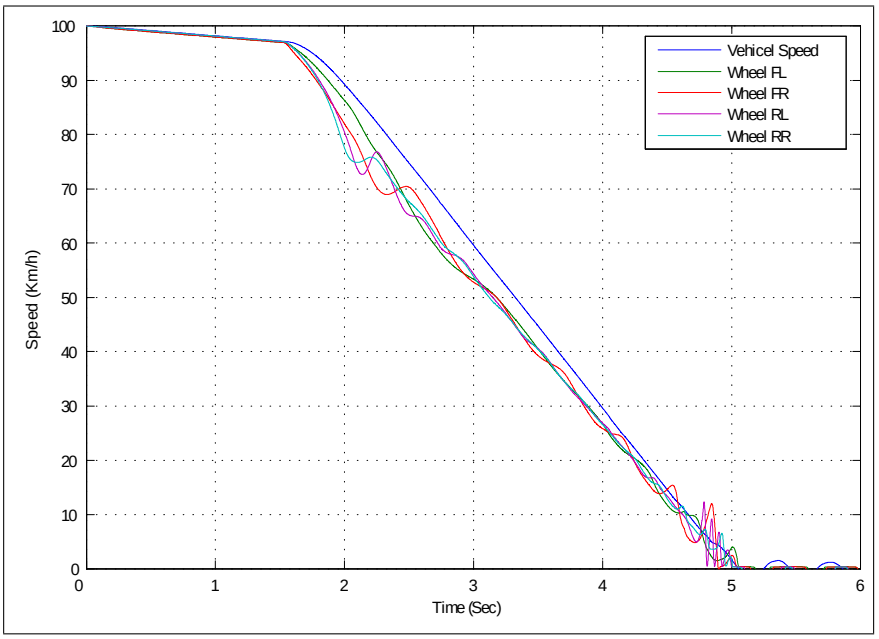


Figure 4.12: Plot of vehicle speed and wheel speeds on a high friction surface ($\mu = 0.85$) with with p value of 0.4

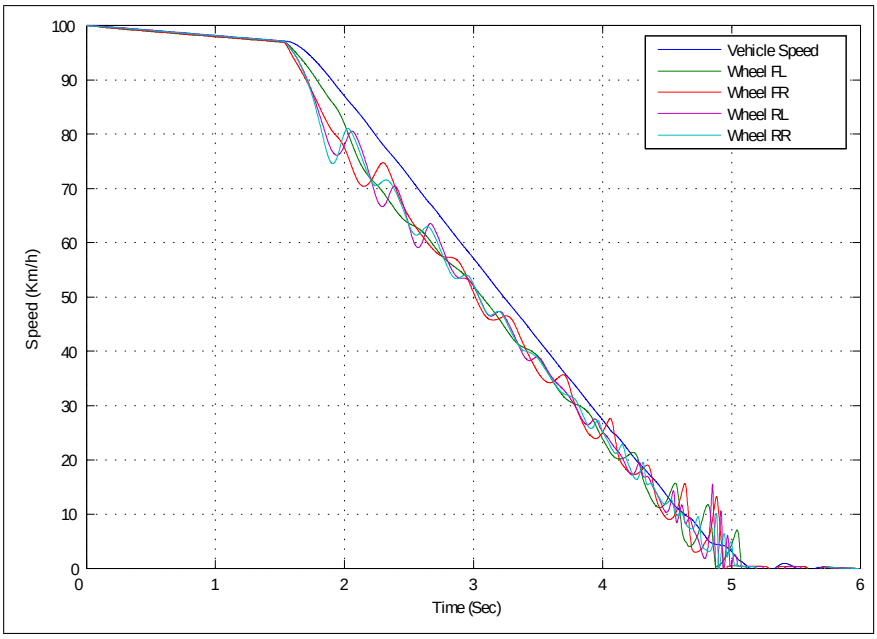


Figure 4.13: Plot of vehicle speed and wheel speeds on a high friction surface ($\mu = 0.85$) with p value of 0.8

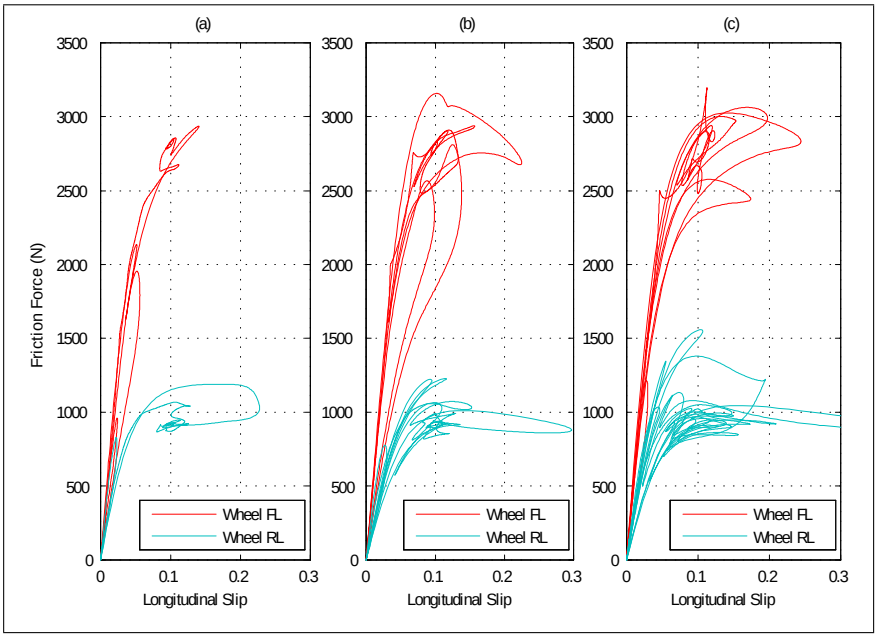


Figure 4.14: Plot of tyre friction forces vs longitudinal slips on high friction surface ($\mu = 0.85$) : (a) $p = 0.2$ (b) $p = 0.4$ (c) $p = 0.8$

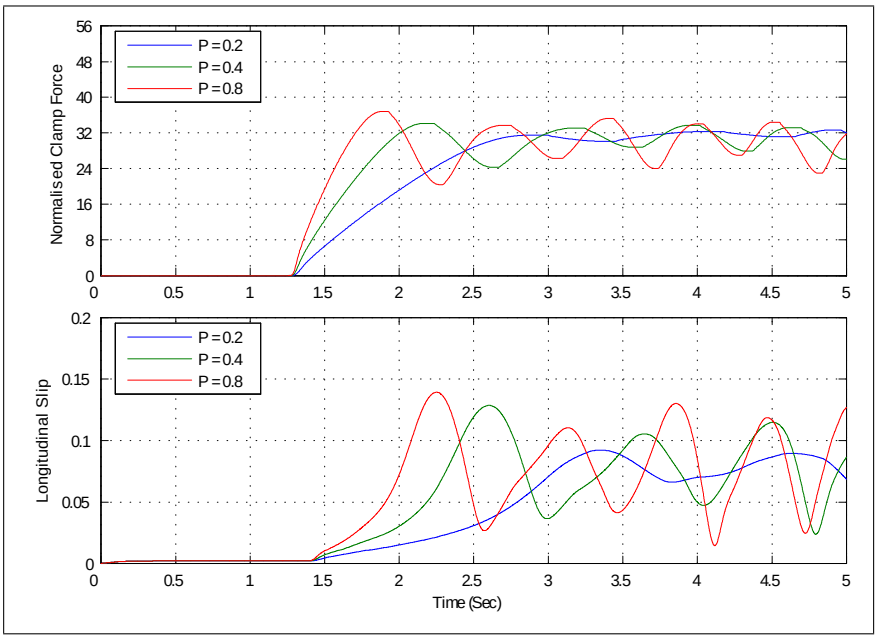


Figure 4.15: Plot of commanded clamp forces (top) and longitudinal slip response of a front left wheel (bottom) on a medium friction surface ($\mu = 0.5$) with varying value of parameter p

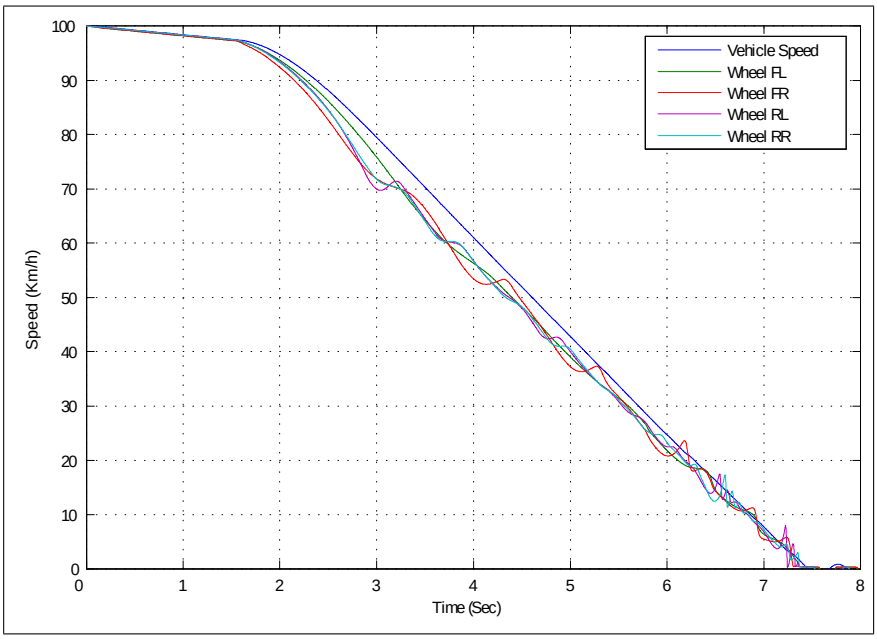


Figure 4.16: Plot of vehicle and wheel speeds on a medium friction surface ($\mu = 0.5$) with with p value of 0.2

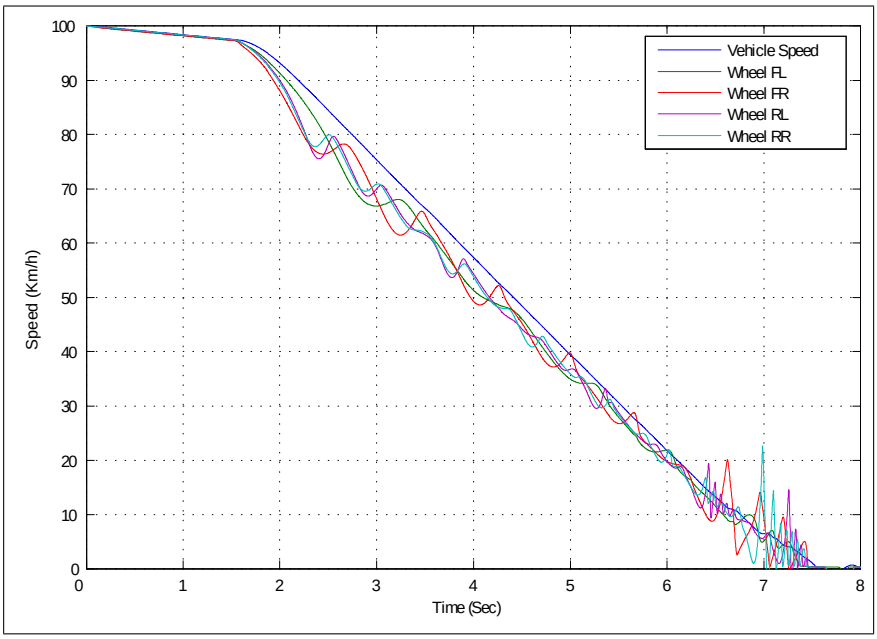


Figure 4.17: Plot of vehicle and wheel speeds on a medium friction surface ($\mu = 0.5$) with p value of 0.4

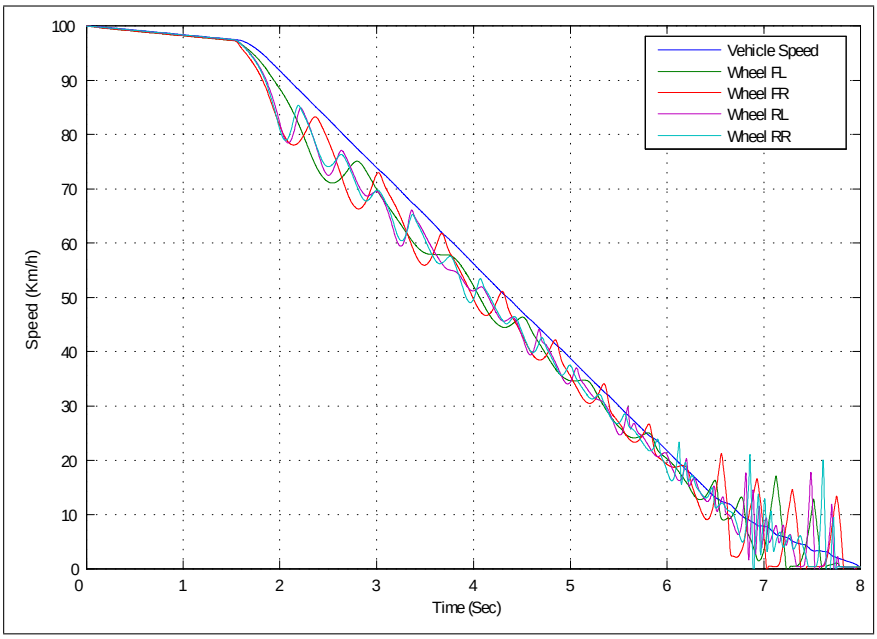


Figure 4.18: Plot of vehicle and wheel speeds on a medium friction surface ($\mu = 0.5$) with p value of 0.8

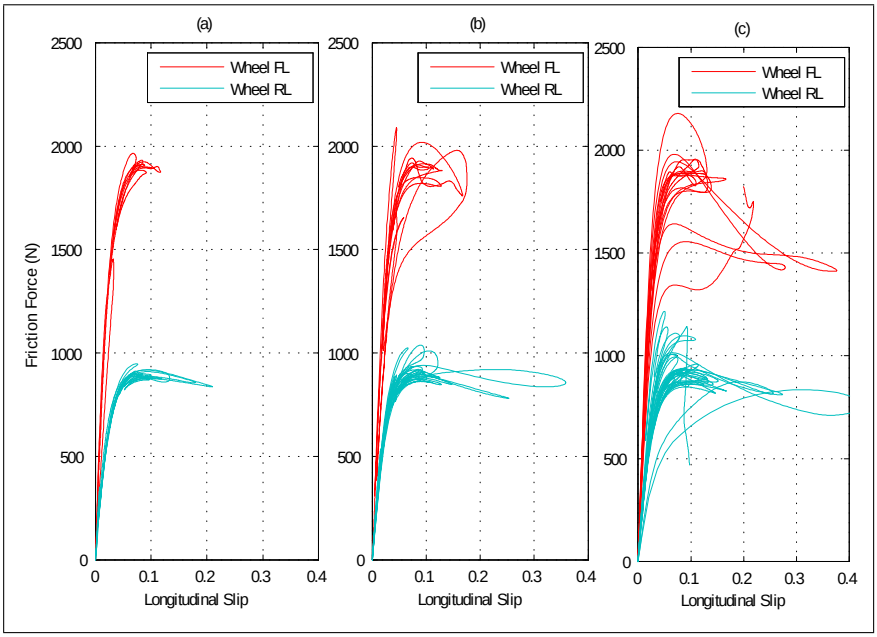


Figure 4.19: Plot of tyre friction forces vs longitudinal slips on a medium friction surface ($\mu = 0.5$) : (a) $p = 0.2$ (b) $p = 0.4$ (c) $p = 0.8$

4.4 Antilock brake performance of wheel slip control

4.4.1 Case A : high friction surface ($\mu = 0.85$)

An emergency braking manoeuvre is performed on high friction surface ($\mu = 0.85$) with an initial vehicle speed of 100 km/h. In this case, a step braking input is applied at 1.5 sec to emulate an urgent driver request and the supervisory logic in Section 3.2 is implemented in the controller to detect unstable slip levels of all wheels. Figure 4.20 shows the vehicle and wheel speed values during the performed braking manoeuvre. Figure 4.21 shows a good performance of regulating the longitudinal slip values with respect to the slip setpoints (dashed line) for both front and rear wheels. Figure also shows the effectiveness of the supervisory logic to inhibit a further increase of braking force, when the instantaneous slip level is exceeded a threshold value. Figure 4.22 shows the set of applied clamp force inputs by the slip controller and a step braking input (Driver cmd). Based on the observation made in the previous section, a design parameter p is chosen to be 0.3. An detailed view of the EMB model responses to the command force inputs are shown in Figure 4.23. For the period of locking in the reversal regions of the clamp force, this is mainly caused by the dynamic friction and the inertia of the internal components of the EMB. A plot of longitudinal slip vs tyre friction force is shown in Figure 4.24. Overall slip performances are satisfactory with

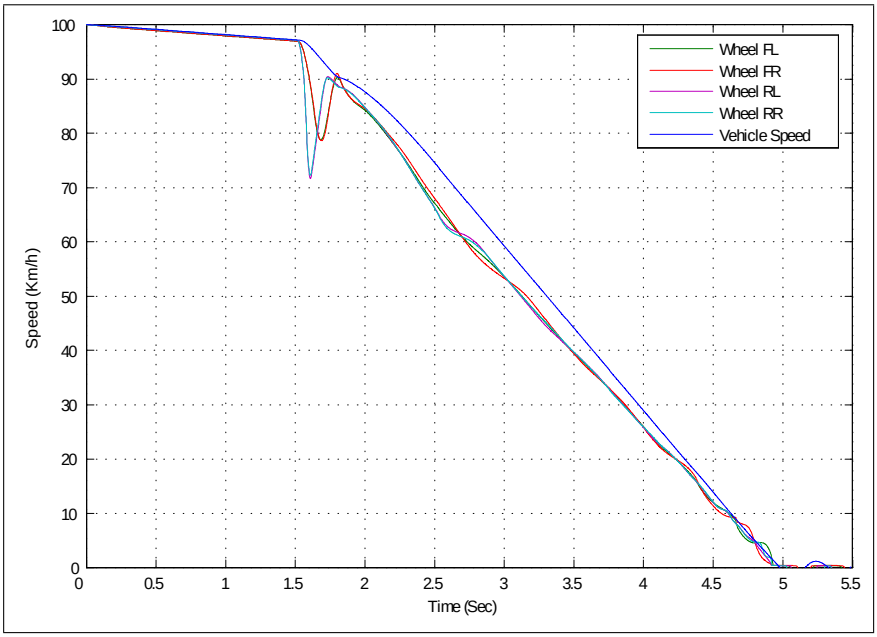


Figure 4.20: Case A : plot of vehicle and wheel speeds during an emergency braking manoeuvre: a spike braking is applied at 1.5 sec, and a parameter p is tuned at 0.45

tyre friction forces being maintained around the peak of friction curve.

4.4.2 Case B : medium friction surface ($\mu = 0.5$)

Identical straight line braking manoeuvre in Case A is performed on a medium friction surface ($\mu = 0.5$). The simulated vehicle and wheel speeds responses from a nonlinear vehicle model is shown in Figure 4.25. Figure 4.26 shows the oscillatory responses in longitudinal slip towards a lower speed region and evidently indicates that slip set-point needs to be lowered to prevent a significant variability in slip response in the low speed region. Figure 4.27 shows both clamp force commanded from the driver model and clamp force generated by the EMB model. Amplified view of the clamp force is shown in Figure 4.28. It can be observed that response of EMB caliper model to a commanded input is generally well responded and less delay in response of clamp force by the electromechanical brake caliper model can be noted which is due to decreased amount of actuation, hence there is less dynamic frictions and inertias to overcome. In Figure 4.29 shows the effectiveness of the slip controller, where most of the attained tyre friction forces are kept within the peak of friction curve.

4.4.3 Case C : low friction surface ($\mu = 0.2$)

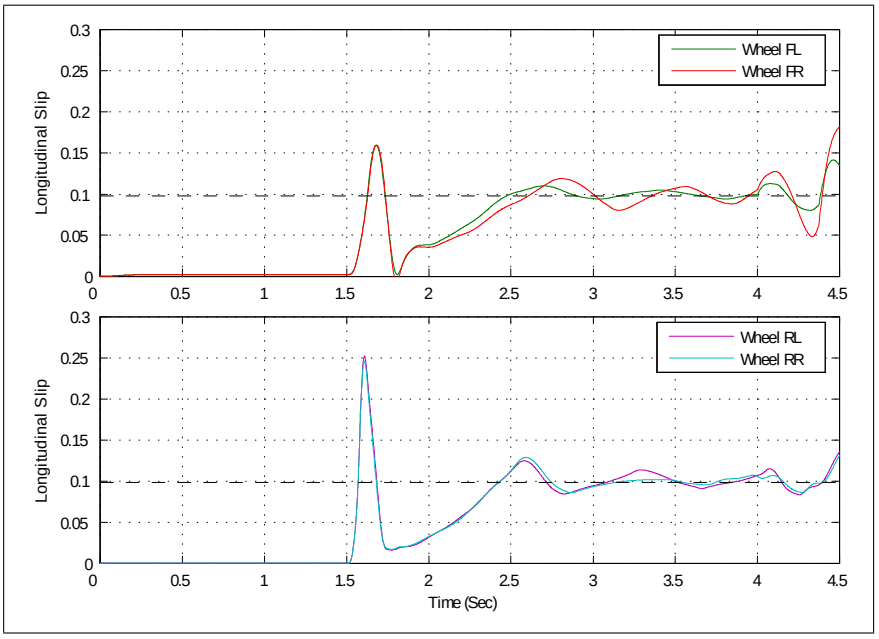


Figure 4.21: Case A : longitudinal slip responses of front and rear wheels with respect to the slip setpoints (dashed line) ; ($p = 0.45$)

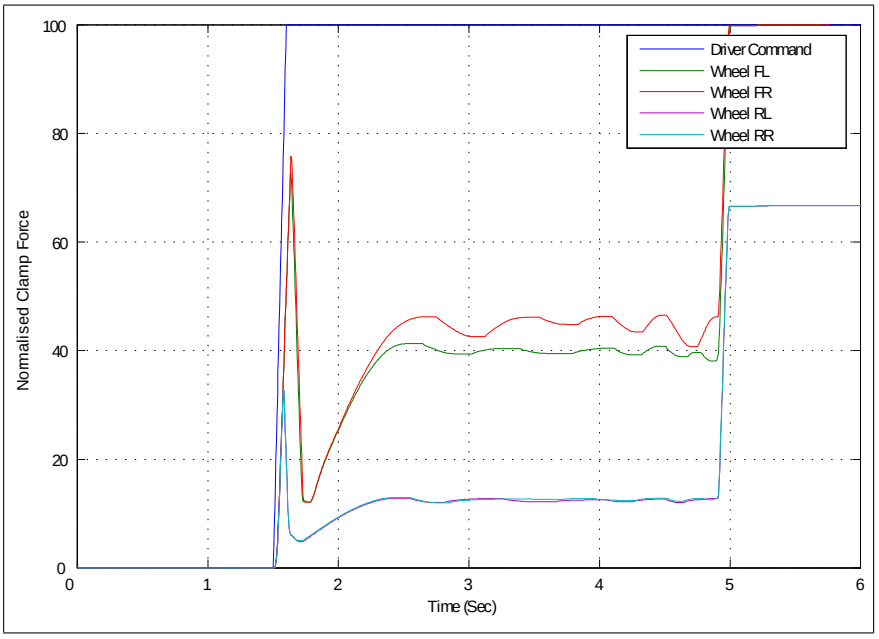


Figure 4.22: Case A : Applied clamp forces by the controller during an emergency braking manoevure (1.5 sec -5 sec). A spike braking input is applied by the driver model (driver cmd) at 1.5 sec: ($p = 0.45$)

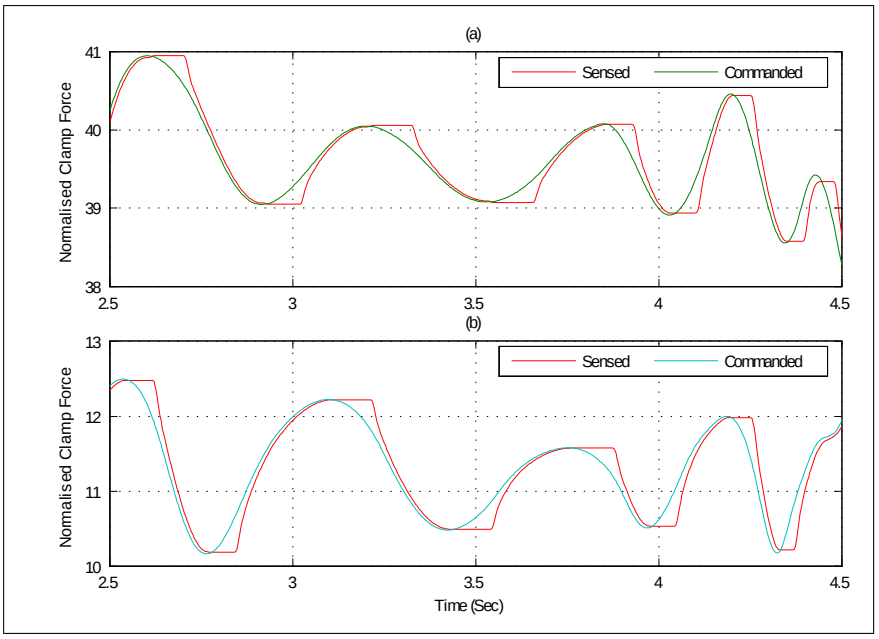


Figure 4.23: Case A: Amplified view of EMB simulation model responses (Sensed) to wheel slip controller force command inputs (Commanded) : (a) front left brake caliper; (b) rear left brake caliper : High friction surface ($\mu = 0.85$)

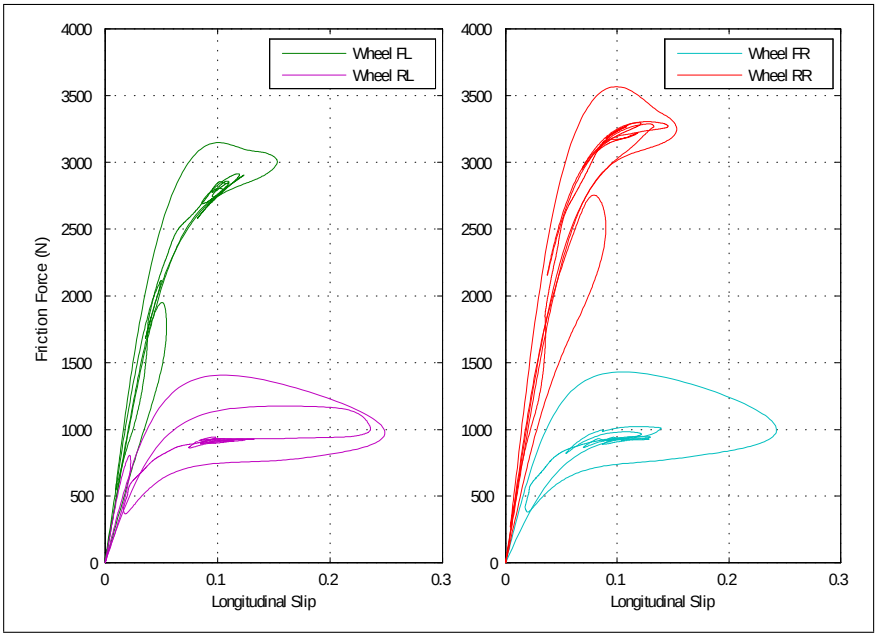


Figure 4.24: Characteristic friction curve for a high friction surface ($\mu = 0.85$) during an ABS braking manoeuvre

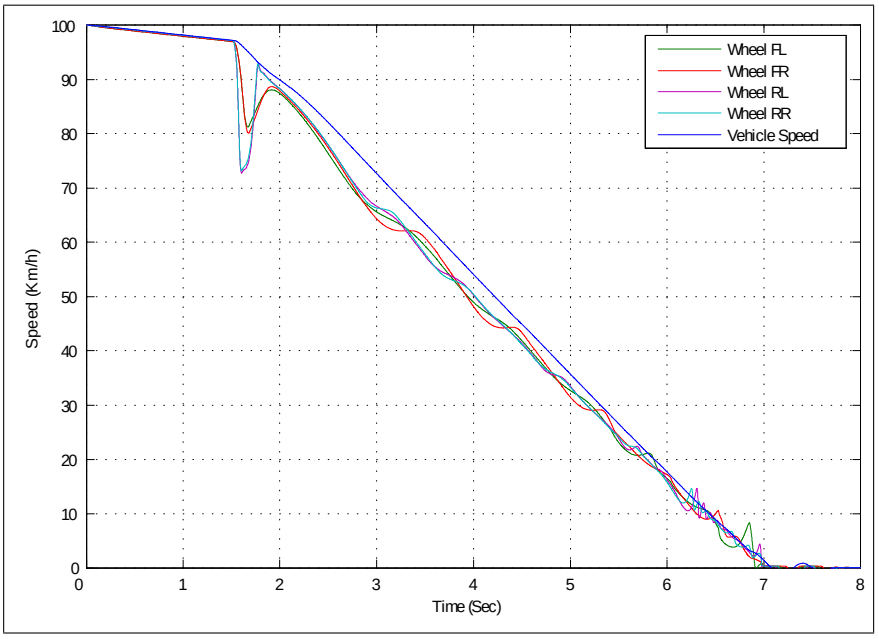


Figure 4.25: Plot of vehicle and wheel speeds during an ABS braking manoeuvre on a medium friction surface ($\mu = 0.5$): a step braking is applied at 1.5 sec ; ($p = 0.25$)

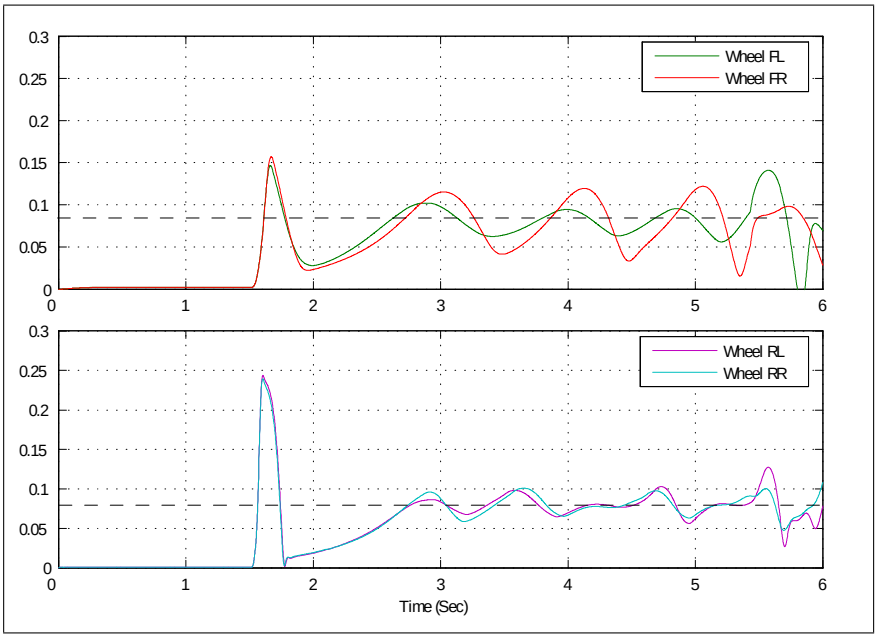


Figure 4.26: Longitudinal slip response of front and rear wheel with respect to the slip setpoint (dashed line) for an ABS braking on a medium friction surface ($\mu = 0.5$); ($p = 0.25$)

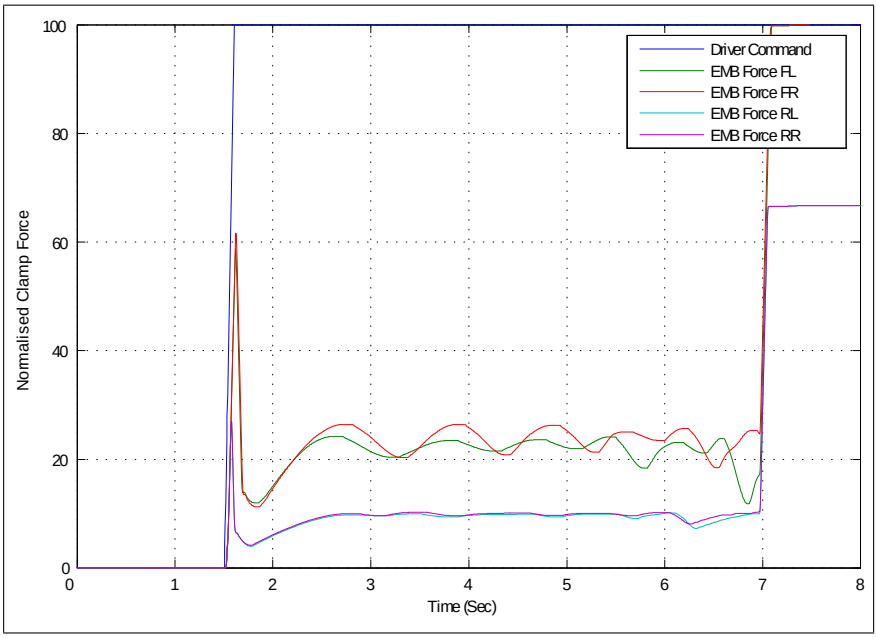


Figure 4.27: Plot of clamp force inputs by the slip controller during an ABS braking manoevure and the spike braking input from the driver model (driver cmd) at 1.5 sec: Medium friction surface ($\mu = 0.5$); ($p = 0.25$)

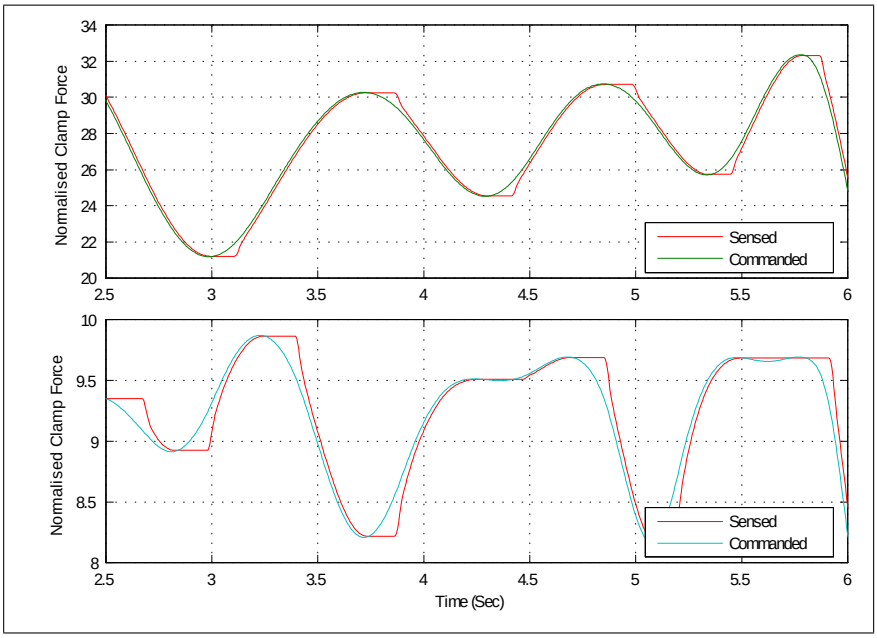


Figure 4.28: Amplified view of responses by the EMB caliper model(Sensed) to the commanded clamp force(Commanded) from the wheel slip controller : (a) front left brake caliper; (b) rear left brake caliper :Medium friction surface ($\mu = 0.5$);($p = 0.25$)

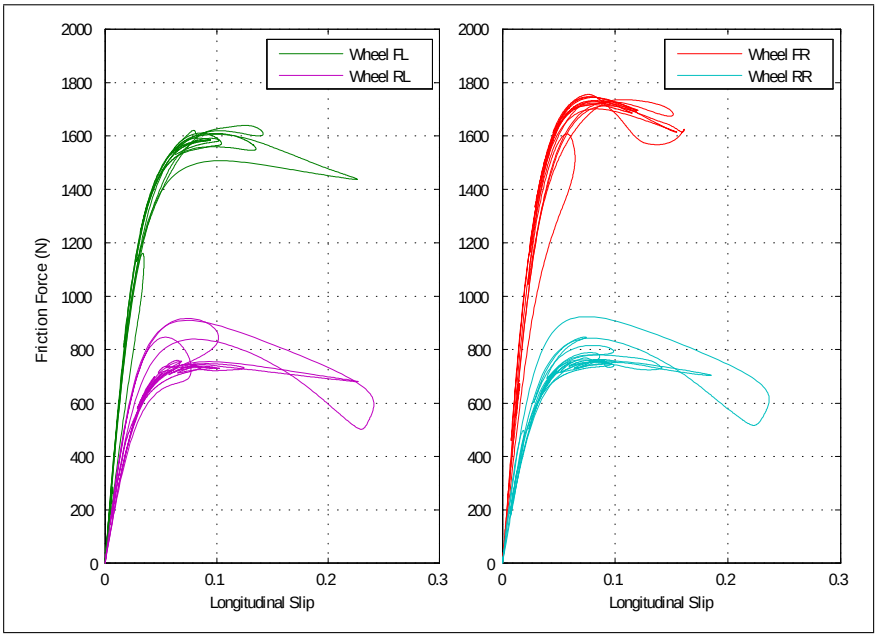


Figure 4.29: Characteristic friction curve for a medium friction surface ($\mu = 0.5$) during an ABS braking manoevure

For a low friction surface such as ice or snow road, considerably less traction force is available between the tyre and the road. Preventing the wheel lock-up becomes an increasingly difficult task. In order to overcome this difficulty, an activation of the control algorithm is done earlier by lowering the slip threshold value. Vehicle speed and wheel speeds during the performed braking manoeuvre are shown in Figure 4.30. In a low friction surface, it is harder to maintain the slip levels at constant or to stabilise it around the optimal level. Figure 4.31 illustrates the more oscillatory longitudinal slip responses. However the overall slip level is maintained within the stable region, and noticeably no major wheel lockup occurred. Figure 4.32 shows the generated input from the wheel slip control algorithm and responded clamp force by the EMB model. Figure 4.33 gives the detailed view of fine modulation of commanded clamp force applied by the controller and the quality of response by the electromechanical brake caliper model. Inspection of Figure 4.34, suggests that venturing into the unstable region is more frequent.

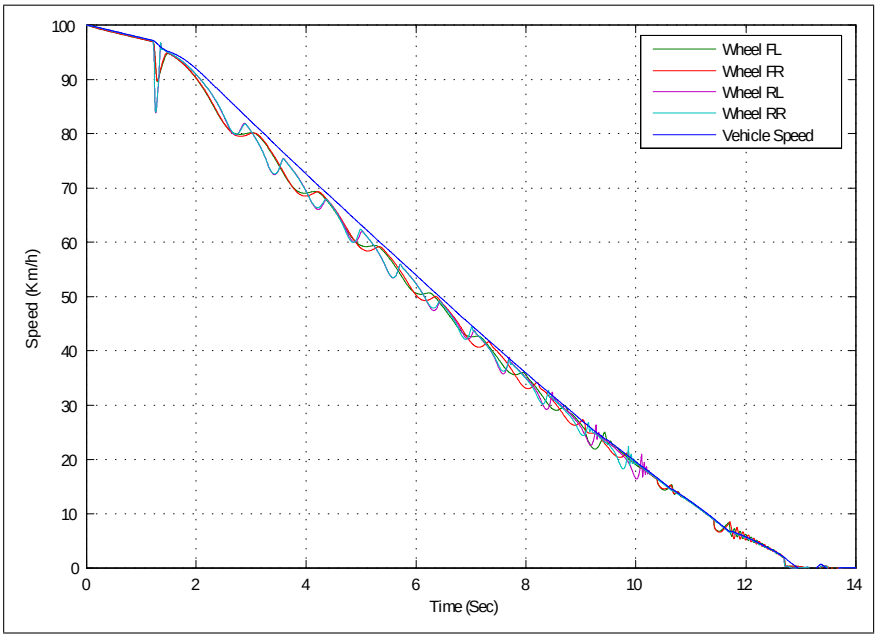


Figure 4.30: Plot of vehicle and wheel speeds during an ABS braking manoeuvre on a low friction surface ($\mu = 0.2$): a step braking is applied at 1.5 sec ; ($p = 0.2$)

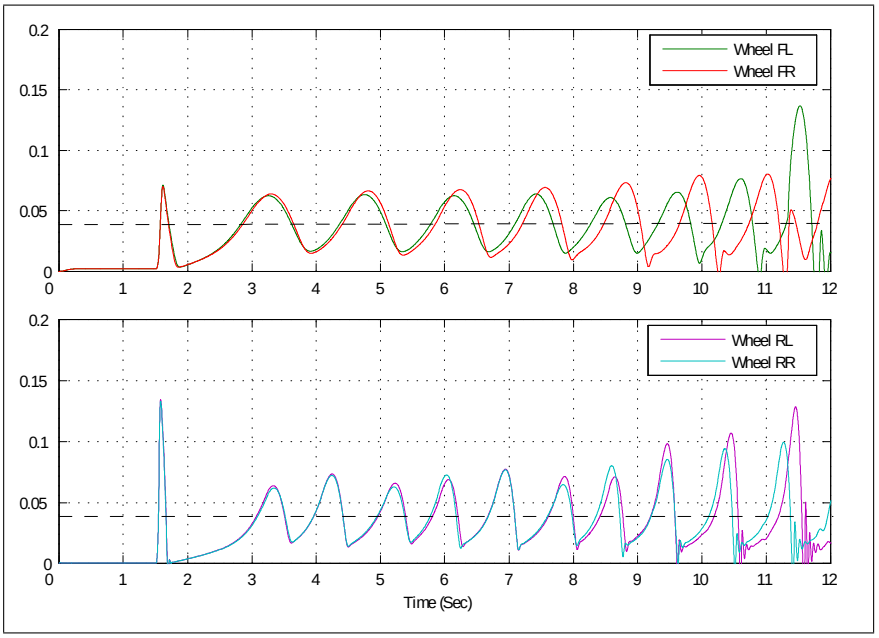


Figure 4.31: Longitudinal slip response of front and rear wheel with respect to the slip setpoint (dashed line) during an ABS braking on a low friction surface ($\mu = 0.2$); ($p = 0.25$)

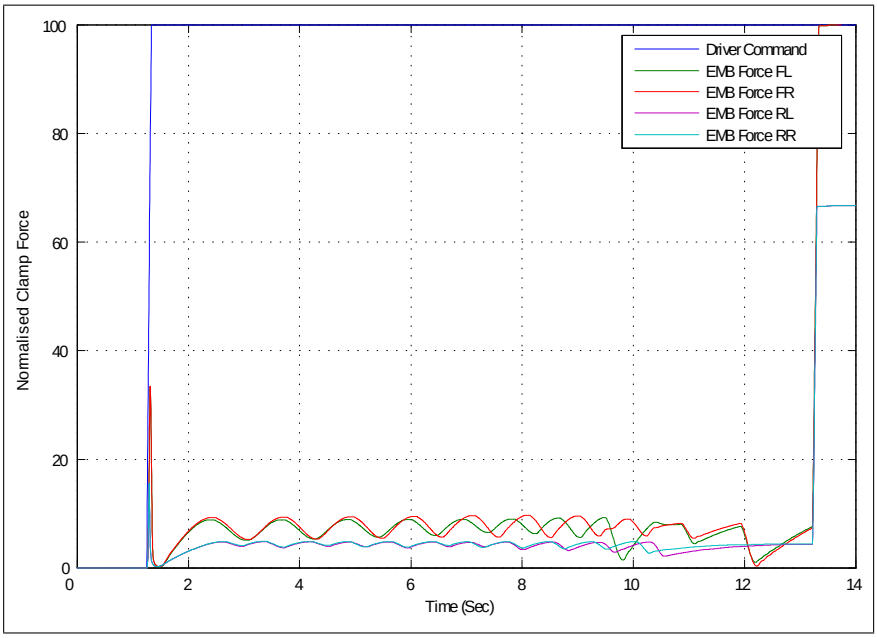


Figure 4.32: Plot of clamp force inputs by the slip controller during an ABS braking manoevure and the spike braking input from the driver model (driver cmd) at 1.5 sec: Medium friction surface ($\mu = 0.2$); ($p = 0.2$)

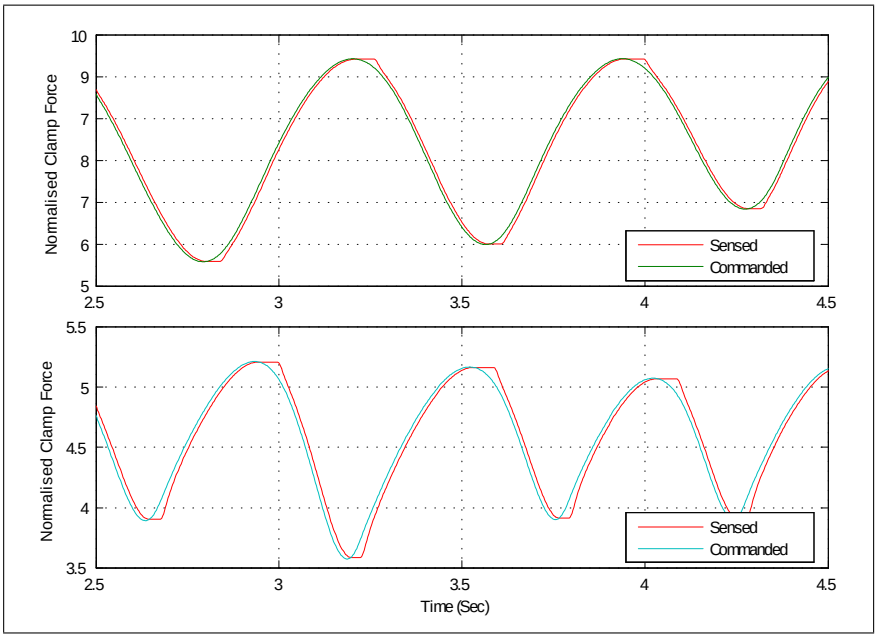


Figure 4.33: Amplified view of responses by the EMB caliper model(sensed) to the commanded clamp force(commanded) from the wheel slip controller : (a) front left brake caliper; (b) rear left brake caliper :Low friction surface ($\mu = 0.2$) , ($p = 0.25$)

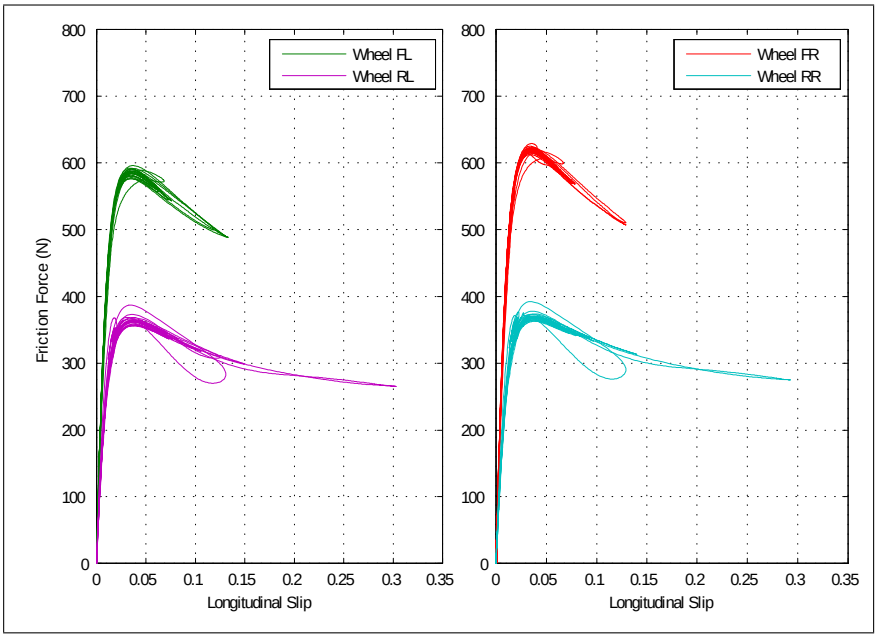


Figure 4.34: Characteristic friction curve for a low friction surface ($\mu = 0.2$) during an ABS braking manoevure

Chapter 5

Comparison of Control Methods for Wheel Slip Control System

In this chapter, comparative antilock brake performances between the PID based control scheme, similar to (Emereole, 2004) and (Solyom, 2004), and the proposed MPC based control scheme is performed in a nonlinear vehicle simulation environment. The PID wheel slip control algorithm used in this work is provided by PBR Automotive Pty Ltd, which contains the commercially protected property. Therefore any detailed design of the algorithm can not be disclosed. however it is assured that it has been tuned for the same EMB system and the evaluation of results are made with the respect to the slip performance criteria.

5.1 Simulation Environment

Simulation testing is conducted under the multibody vehicle simulation environment of Carsim. This vehicle simulation model has been frequently used in the industry for simulating and analyzing the dynamic braking and handling behaviour of vehicles under a variety of conditions. Table 5.1 shows the nominal vehicle parameters used in the simulation.

Unsprung vehicle mass	1000 kg
Wheel mass	26 kg
Wheel inertia	0.78 kgm^2
Unloaded wheel radius	0.205 m

Table 5.1 : CarSim vehicle parameters

5.2 PID Control vs. MPC Control

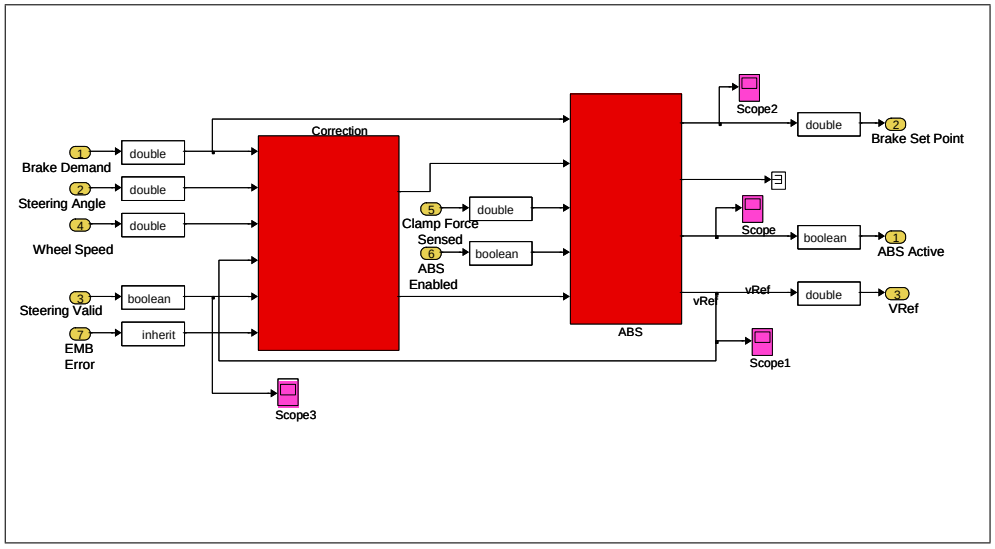


Figure 5.1: Simulink representation of PID control algorithm. **Inputs**: (1). Brake Demand from the Driver input. (2) Steering Angle input from the Driver input (it is assumed to be zero.) (3) Wheel speed input (4) System error : Steering Valid & EMB error. (5) Clamp force sensed: actual clamp force applied at the brake. (6) ABS Enabled. **Outputs**: (1) Brake set point: computed brake clamp force from the wheel slip control algorithm. (2) ABS active (3) VRef: vehicle reference speed calculated by the ABS subsystem based on the wheel speed input.

5.2.1 Overview of PID control algorithm

The PID control algorithm works by switching between two states, that is a real feedback control algorithm during excessive wheel slip and a time dependent clamping force rise during the rest of time. It also compares desired lateral acceleration to the longitudinal deceleration to find a compromise between steering and braking when the driver demands more braking and steering power than available. When the wheel is estimated to be unstable (i.e. value of wheel slip being greater than the optimum slip value), PID controller is activated to reduce the wheel slip value to the desired set-point value and the anti-windup algorithm is used to handle the actuation limit of EMB calipers. Once the wheel slip is estimated to be stable, the set clamping force is rised with time which the gradient of rise in clamping force is calculated by the internal algorithm. Figure 5.1 shows the Simulink model of the PID control algorithm. Due to the unavailability of the PID gains for the low friction surface. Comparisons of PID and MPC based wheel slip control systems are made on high and medium friction surfaces only.

5.2.2 Comparison of controller performance on dry road surface

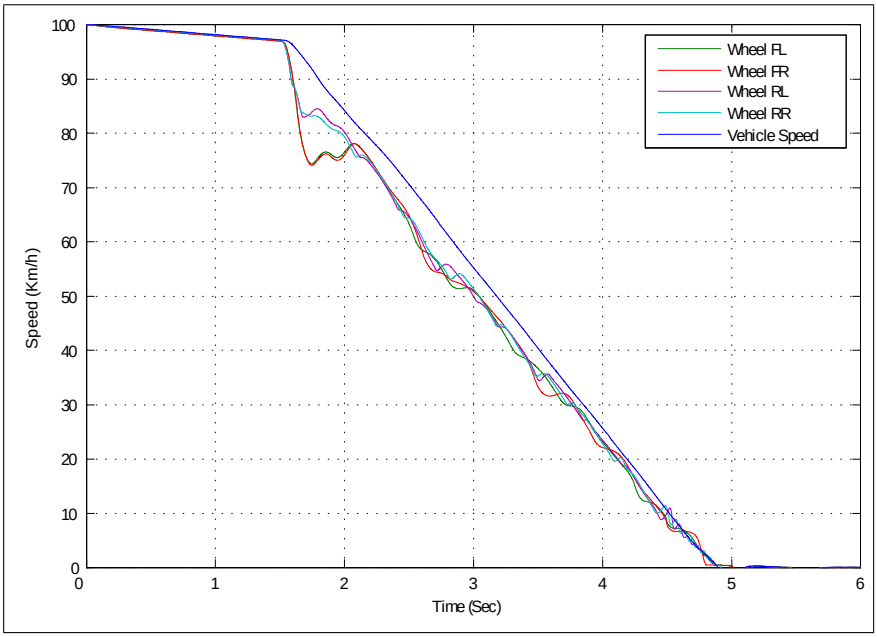


Figure 5.2: Plot of vehicle and wheels speed on a high friction surface ($\mu = 0.85$). Spike braking is applied at 1.5 sec: PID control

In this comparison, values of MPC parameters (N, T_p and p) are chosen to be same as in Section 4.4.1. Identical straight braking manoeuvre is performed on dry road surface (high friction surface, $\mu = 0.85$) where the friction coefficients of a surface is assumed to be constant. Figure 5.2 and 5.3 shows the response of the vehicle and the wheel speed values from the PID and MPC wheel slip control system respectively. The outcome of an individual wheel slip levels of two controllers are shown in Figure 5.4 and 5.5. As can be seen, MPC controller outperforms the PID controller for regulating the slip levels around the optimal point (dashed line). This is due to a more smoother control inputs applied by the MPC controller. Figure 5.6 and 5.7 highlight the applied clamp forces by the two controllers.

5.2.3 Comparison of controller performance on wet road surface

ABS braking manoeuvre on wet surface ($\mu = 0.5$) by the PID and MPC controller is shown respectively in Figure 5.8 and 5.9. Design parameters of MPC controller in Section 4.4.2 are used in this comparison. The rear wheel slip responses of the MPC controller in Figure 5.11 indicates that slip level are retained at higher and more closer to the optimal slip level ($\lambda = 0.8$) than the PID controller.

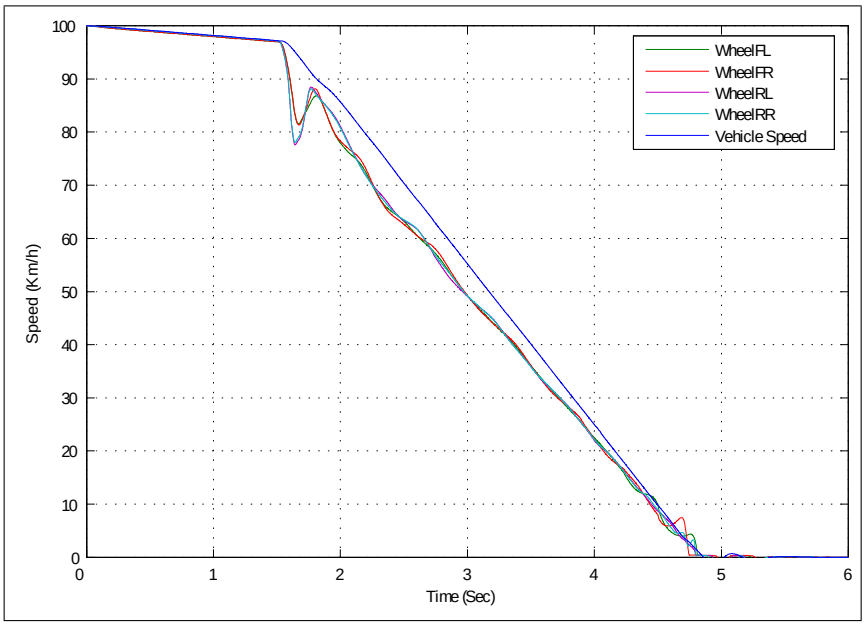


Figure 5.3: Plot of vehicle and wheels speed on a high friction surface ($\mu = 0.85$). Spike braking is applied at 1.5 sec: Model Predictive Control

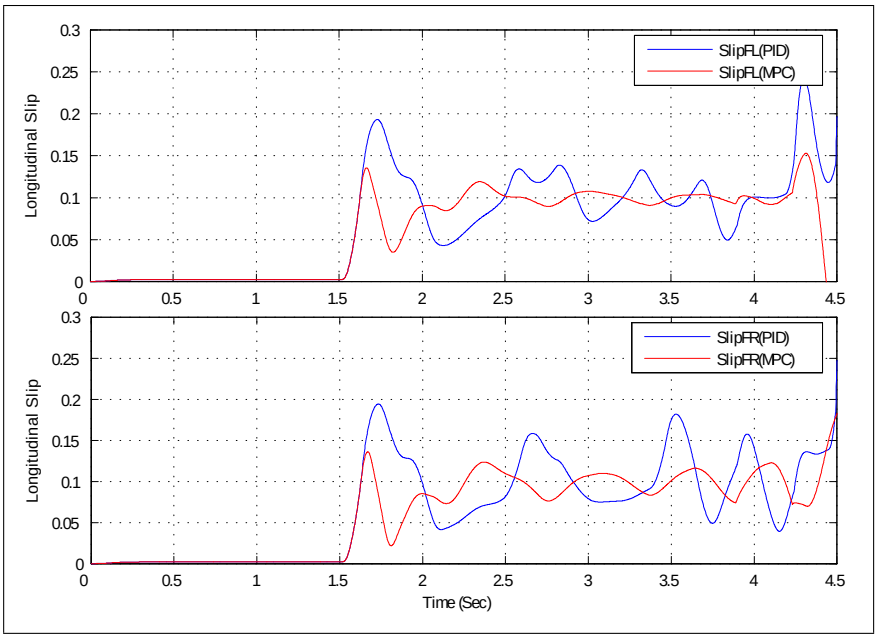


Figure 5.4: Comparison of front wheel slip response w.r.t optimal slip level (dashed line) of 0.1 on a high friction surface ($\mu = 0.85$)

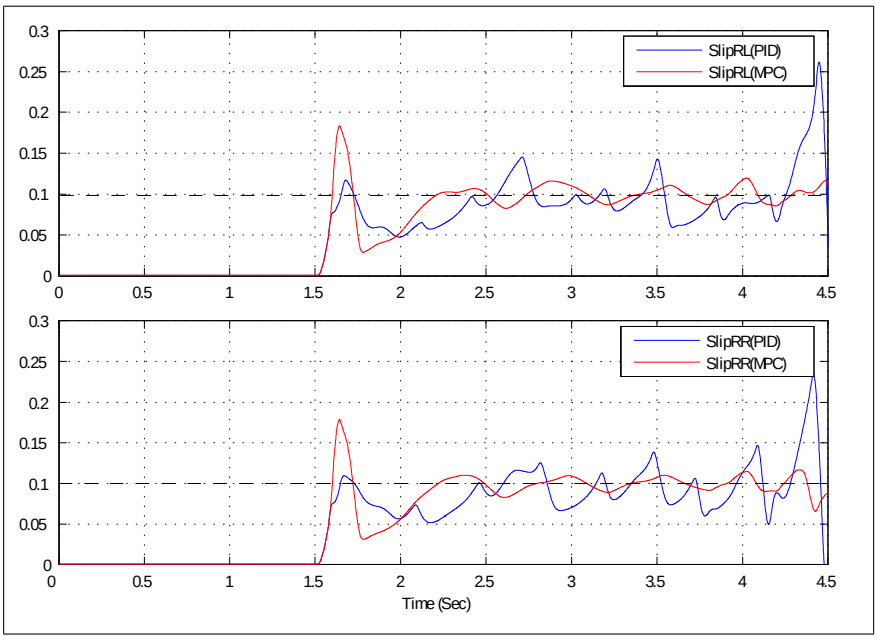


Figure 5.5: Comparison of rear wheel slip response w.r.t optimal slip level (dashed line) of 0.1 on a high friction surface ($\mu = 0.85$)

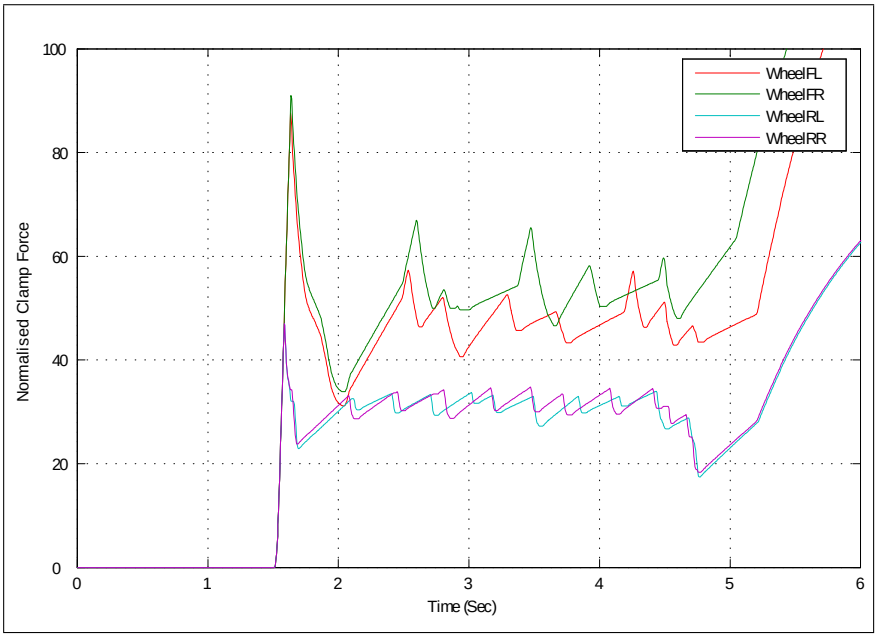


Figure 5.6: Response of EMB caliper model to force command signal generated by the PID controller, ($\mu = 0.85$): When the slip value exceeds the threshold (see Figure 5.4 and 5.5) after 1.5 sec, the controller is turned on to reduce the clamp force to stabilise the slip value. Period between 2 and 5 sec, the brake clamp force are modulated by the PID controller to stabilise the slip at the optimal point. After 5 seconds. when the vehicle came to a full stop, the controller is turned off..

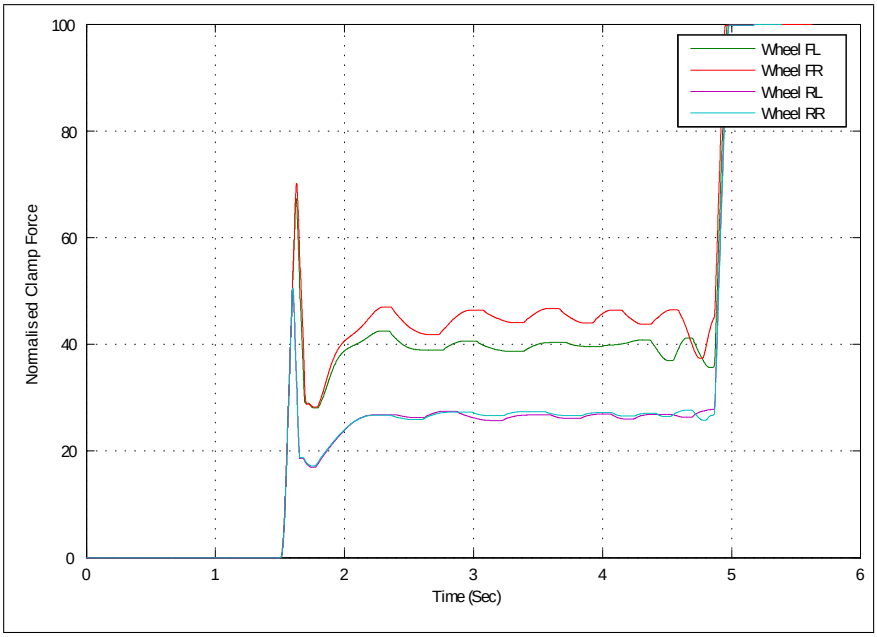


Figure 5.7: EMB model response to force command signal generated by the model predictive controller, ($\mu = 0.85$): The controller is turned on when the slip value exceeds the threshold (see Figure 5.4 and 5.5) after 1.5 sec. MPC controller generates a more smooth control signal and the slip is than the PID controller. Clamp force to stabilise the slip value. Period between 2 and 5 sec, the brake clamp force are modulated by the PID controller to stabilise the slip at the optimal point. After 5 seconds. when the vehicle came to a full stop, the controller is turned off..

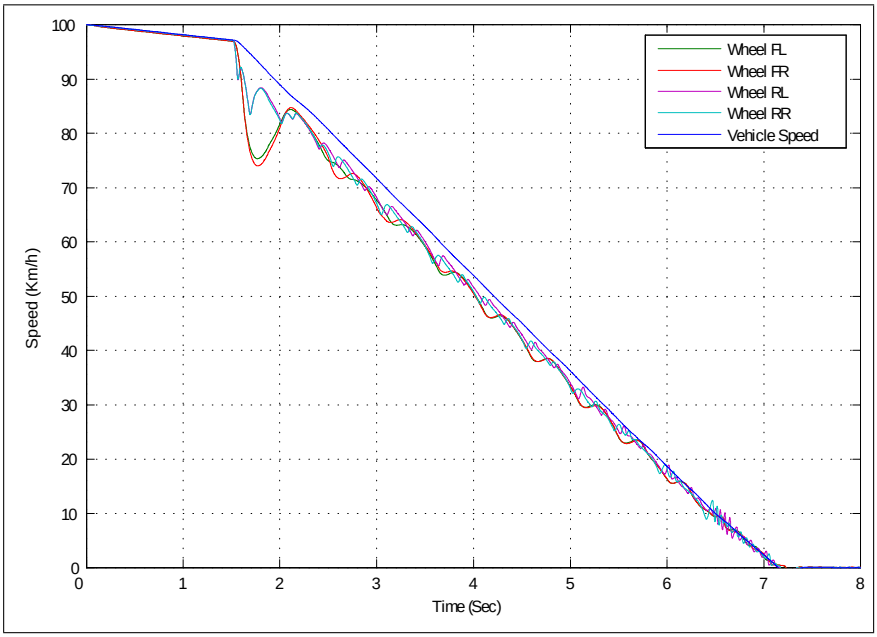


Figure 5.8: Plot of vehicle and wheels speed on medium friction surface ($\mu = 0.5$) : PID control

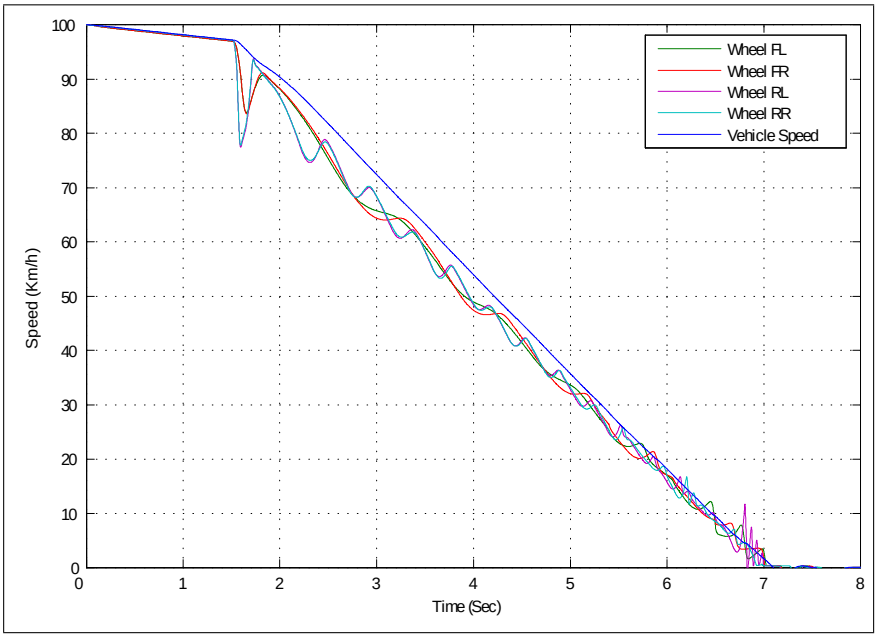


Figure 5.9: Plot of vehicle and wheels speed on medium friction surface ($\mu = 0.5$) : MPC control

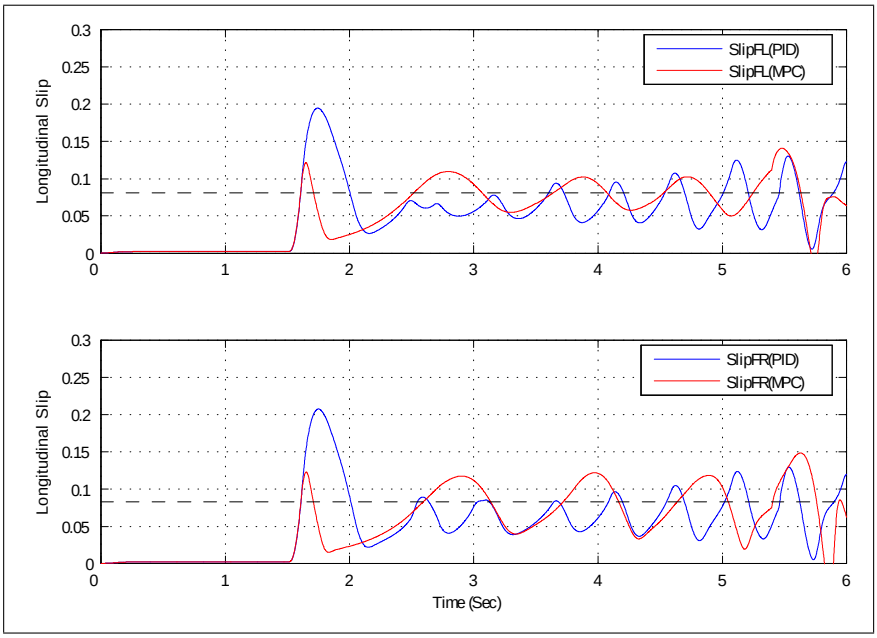


Figure 5.10: Comparison of front wheel slip response w.r.t optimal slip level (dashed line) of 0.08 on high friction surface ($\mu = 0.5$)

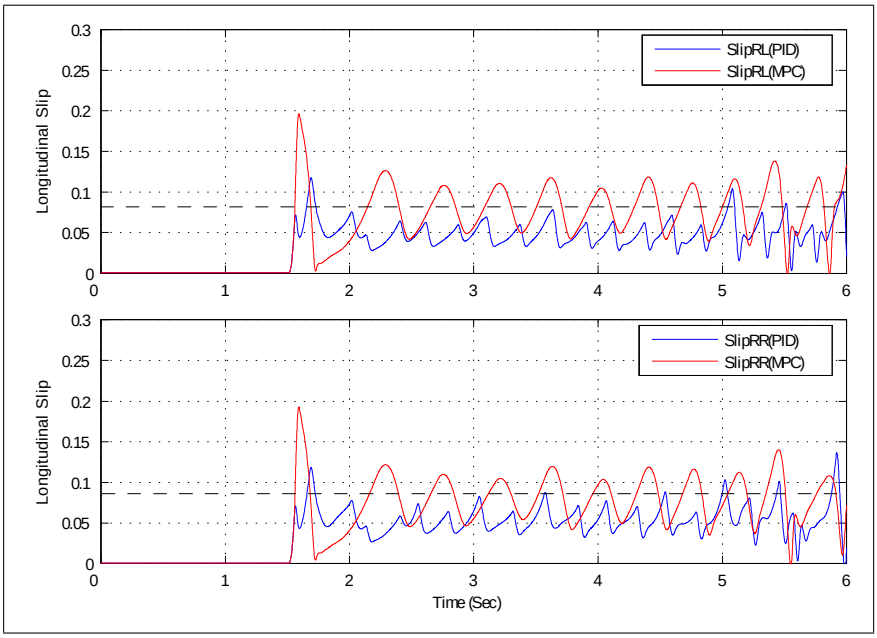


Figure 5.11: Comparison of rear wheel slip response w.r.t optimal slip level (dashed line) of 0.08 on high friction surface ($\mu = 0.5$)

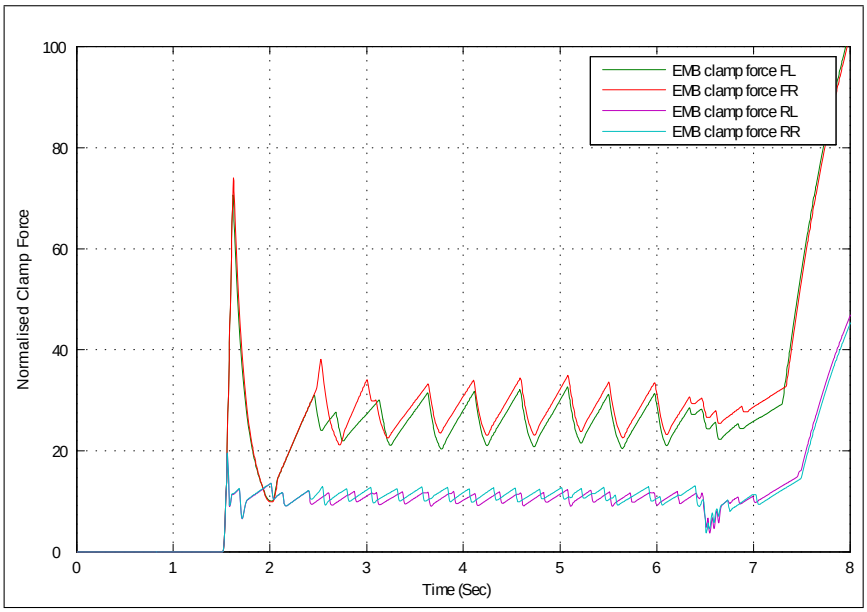


Figure 5.12: Response of EMB caliper model to force command signal by the PID controller, ($\mu = 0.5$): When the slip value exceeds the threshold ($\lambda = 0.06$) (see Figure 5.10 and 5.11), the controller is turned on to reduce the clamp force to stabilise the slip value. Period between 2 and 6 sec, the high frequent brake force modulation is generated by the PID controller to stabilise the slip at the optimal point ($\lambda = 0.08$). After 7 seconds. when the vehicle came to a full stop, the controller is turned off..

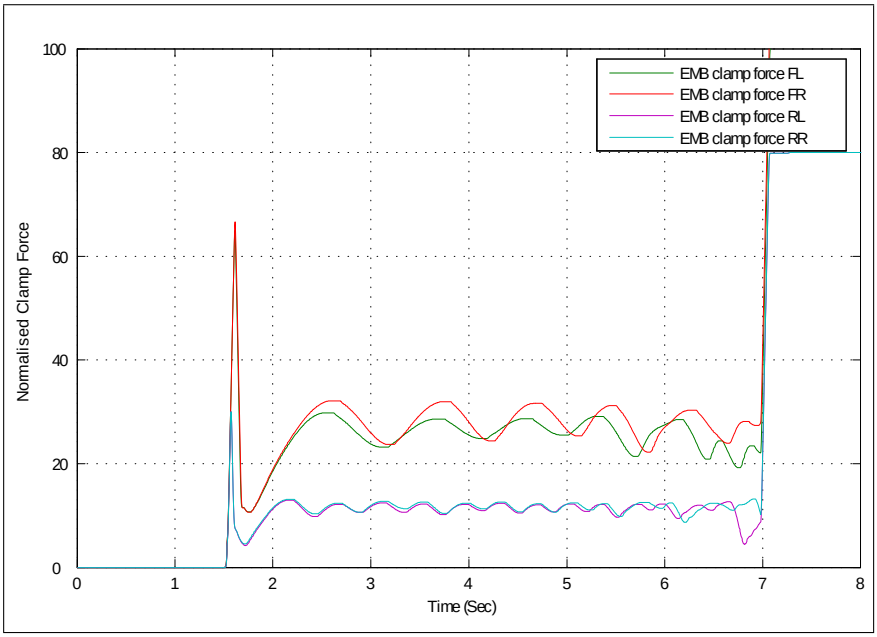


Figure 5.13: EMB caliper model response to a force command signal by the MPC controller, ($\mu = 0.5$): When the slip value exceeds the threshold ($\lambda = 0.06$) (see Figure 5.10 and 5.11), the controller is turned on to reduce the clamp force to stabilise the slip value. Comparatively the more smooth brake force modulation is generated by the MPC controller to stabilise the slip at the optimal point ($\lambda = 0.08$). After 7 seconds. when the vehicle came to a full stop, the controller is turned off..

Chapter 6

Hardware-In-the-Loop (HiL) Simulation

Complexity of a vehicle dynamic control system has made it difficult to devise a new control algorithm using the conventional trial and error method. This has prompted a computer aided simulation method, as described in Chapter 5, to be employed to verify the implementation code in the simulation environment that consists of mathematical model representation of the plant and the environment. In this chapter, to further validate and evaluate the performance of the proposed wheel slip control system in real time, the real hardware/actuator is incorporated in the HiL simulation loop (i.e. Hardware-in-the-Loop (HiL)). More specifically the actuator model used in SiL simulation is replaced by real EMB brake calipers to investigate the performance of the proposed wheel slip control algorithm in a more realistic brake-by-wire system. Section 6.1 presents the framework of proposed HiL system with a detailed description of the hardware architecture. This system has been constructed based on the configuration of the distributed simulation cluster in Chapter 5 and the real-time implementation of the wheel slip control algorithm. Sections 6.2 and 6.3 present the validated performance of the wheel slip controller in a brake by wire environment. Similar tuning procedures used in Chapter 5 is carried out in the hardware in the loop simulation environment. Again the same scenarios of ABS braking manoeuvre are performed.

6.1 HiL simulation framework

The essence of HiL simulation approach is to incorporate an actuator of primary concern in the simulation environment to investigate the characteristics of a true physical

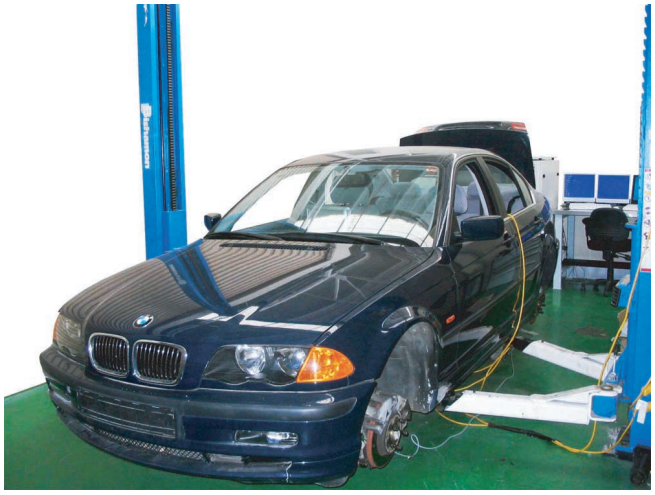


Figure 6.1: Front view of prototype brake-by-wire vehicle in HiL simulation set-up

system and validation of the controller's performance in a more realistic environment. As a result, it is increasingly recognised as an essential tool in many applications as a rapid prototyping stage in the development cycle. However, the majority of existing HiL designs are constructed in an ad-hoc fashion where designs are intended for a particular application. These shortfalls of a existing HiL design approach are highlighted by (Bacic, 2005) and (Stasko *et al.* , 1998), and suggest a systematic way of constructing the HiL system. With the importance of modularity and reusability of the simulation system. a flexible "Distributed HiL system" is proposed based on the principles of distributed simulation described in Section 5.1. The proposed system extends the concept of a standard monolithic processor based simulation system to multiprocessor simulation system where it combines actual operational equipment with realistic representation of operational environments (see Figure 6.1, 6.2 and 6.3).

6.1.1 Hardware Configuration

The hardware part of the system consists of a multi-processor simulation cluster, a prototype brake-by-wire vehicle and a CAN communication bus. Figure 6.4 shows the overall structure of the set-up where the central control unit (ECCU) provides the gateway communication between the real-time simulation cluster and the BBW system. Informations such as commanded brake clamp forces generated by the wheel slip controller are transmitted from the real-time simulation cluster to the ECCU via CAN bus, which is then distributed to individual WBCU using the internal communication channel (i.e. Time-Triggered Protocol (TTP)). Each Wheel Brake Control

Figure 6.2: Rear view of prototype brake-by-wire vehicle in HiL simulation set-up

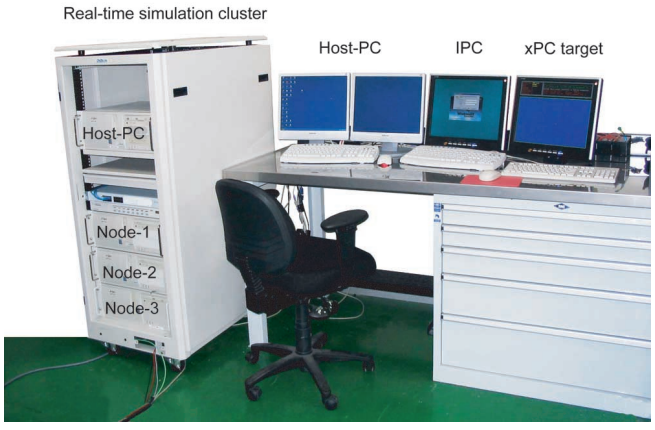


Figure 6.3: Picture of real-time simulation cluster with test bench

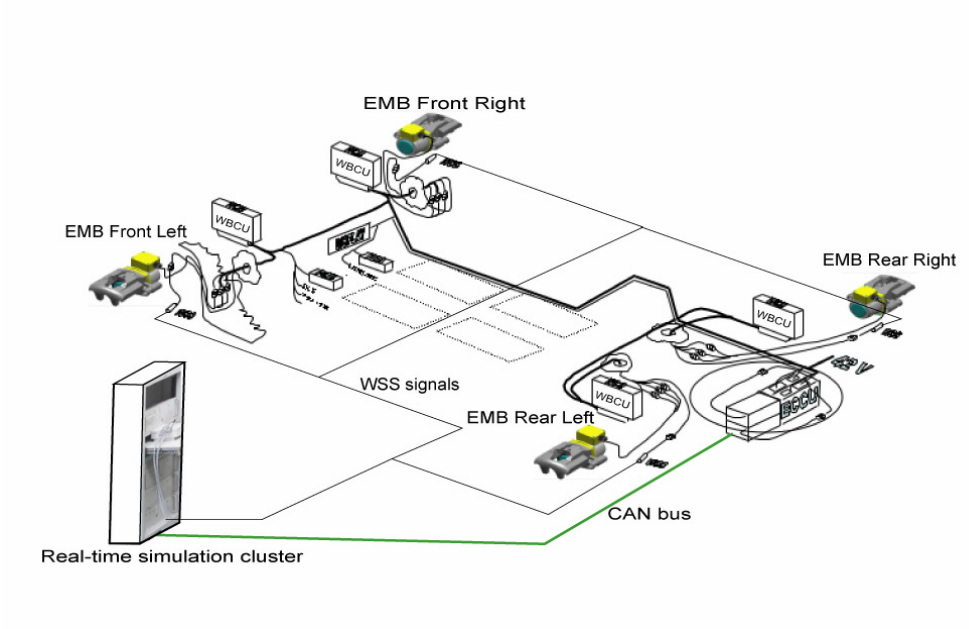


Figure 6.4: An overview of hardware-in-the-loop simulation set-up

Unit (WBCU) controls the EMB caliper, as described in Chapter 1, to generate the required brake clamp force. The internal clamp force sensor of EMB is sensed by the WBCU, and the measured brake clamp force is then transmitted back to the ECCU which is also shared with the real time simulation cluster for computing the control input by the wheel slip control algorithm.
The real-time simulation cluster is the core component in the system where it provides the real-time computation of a nonlinear vehicle simulation model, and the wheel slip predictive control algorithm as well as providing transmission medium between the test vehicle and the simulation cluster. The simulation cluster contains three real-time simulation units and a host computer as shown in Figure 6.3. Each of the real-time simulation unit is constructed in a similar manner and comprises of high speed real time processors for calculating the dynamic models, algorithms and I/O cards for receiving and transmitting CAN messages. Each unit also contains reflective memory network cards. To realise the maximal benefit of the multi-processor architecture, the real-time tasks are partitioned as below,

Node configuration

- **Simulation Node-1**

  A real time execution of the wheel slip control algorithm is performed in simulation node-1, see Figure 6.3. It also handles the CAN communication messages

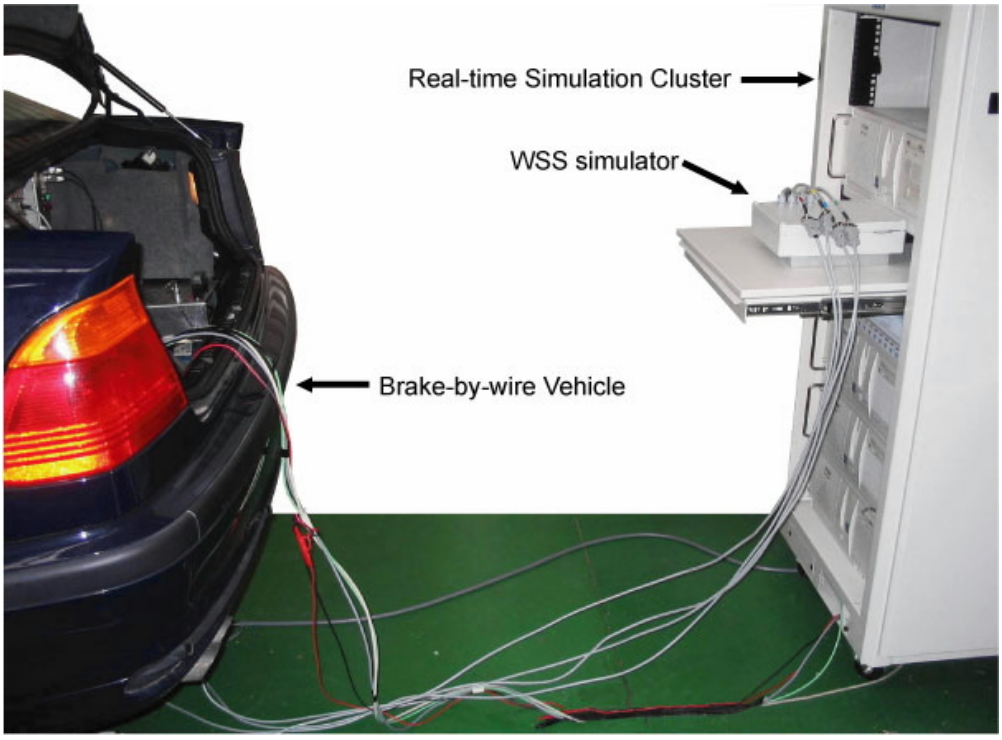


Figure 6.5: Picture of interconnection between the brake-by-wire vehicle, real-time simulation cluster and wheel speed simulator (WSS simulator)

between the BBW vehicle, in order to transmit the calculated brake force command signals to the test vehicle and to receive the measured clamp forces from the internal force sensors in the EMB calipers.

- **Simulation Node-2 and 3**

  It is expected that the real-time simulation of vehicle model would require a substantial computational power and for this reason, the vehicle model is decomposed and distributed over simulation node 2 and 3.

- **Host computer**

  The host PC is used both to develop models and to configure and download a compiled model to a targeted real-time simulation unit. The host PC contains a reflective memory card for logging of vehicle parameters and provides user interface.

6.1.2 Software Configuration

As the credibility of HIL simulation results is largely dependent on the accuracy of the real-time simulation model, a real-time implementation of ADVANCE vehicle model is used in this work which contains the same set of validated realistic parameters of a medium size vehicle as described in Section 5.3.

Real-time simulation of the nonlinear vehicle model and the computation of the wheel slip control algorithm are performed using the MathWorks^{TM} xPC target toolbox. As shown in Figure 6.6, reflective memory blockset (orange) provides input from the real-time vehicle simulation model such as instantaneous slip, vehicle speed and measured clamp force from the EMB calipers. The wheel slip control algorithm block (green) contains the proposed wheel slip predictive control algorithm. Calculated brake forces are then transmitted to the prototype brake-by-wire system via CAN communication bus at a predefined rate.

The following section describes the steps involved in converting a numerical simulation model in the Matlab environment to real-time simulation model in the xPC target environment.

- A Simulink model of the ADVANCE vehicle model is compiled with the fixed-time step solver using the real-time workshop to generate the C code for real-time simulation.

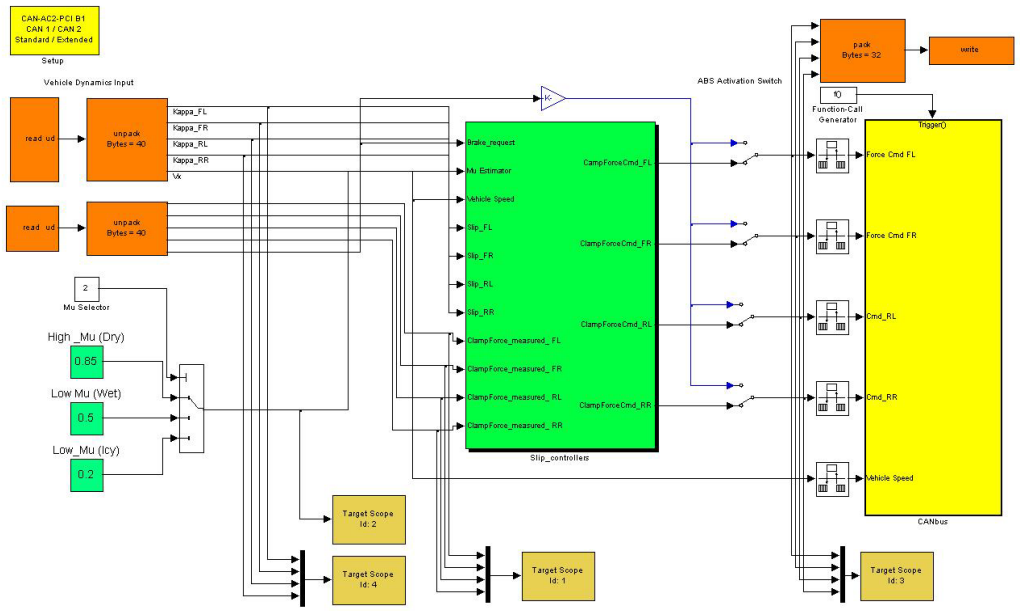


Figure 6.6: Real-time implementation wheel slip control model: reflective memory blockset (orange), CAN communication blockset (yellow) and wheel slip control algorithm block (green)

- The generated code is then downloaded to a targeted real time processing unit over Ethernet.

For visualisation and data logging, the host computer is fitted with reflective memory where it collects the relevant inputs to the VRML world for 3-D visualisation of the vehicle. Standard VRML tool provided by the TNO automotive is used to represent the response of vehicle in 3-D graphical format. Customised code is written for mapping of physical reflective memory into host computer's internal memory. This enables the use of existing reflective memory library blockset with custom driver software, which significantly eases of logging and monitoring of the variables in the distributed units.

6.2 Experimental results of tuned controller

In the following results of tuned controller, no supervisory logic is implemented, and the activation of the wheel slip control algorithm is only realised by the external command.

6.2.1 Case A : High friction surface ($\mu = 0.85$)

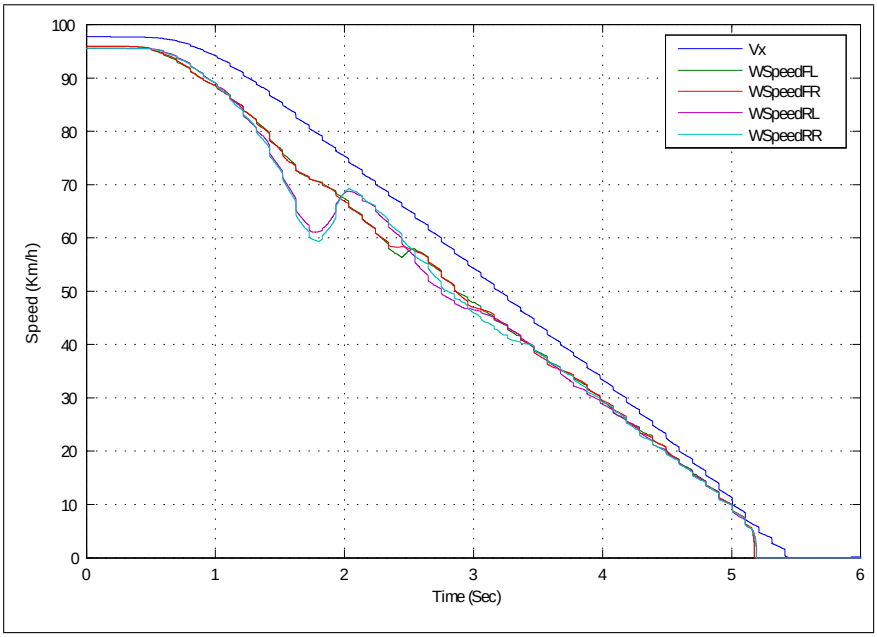


Figure 6.7: Plot of simulated vehicle speed (Vx) and wheel speeds (WSpeed) on a high friction surface ($\mu = 0.85$) with the parameter p tuned at 0.35

A similar tuning procedure as described in Section 5.3 is carried out to obtain a desirable braking response in a hardware-in-the-loop simulation. As a result, parameter p is tuned at 0.35. Figure 6.7 shows the plot of vehicle and wheel speed values on a high friction road surface ($\mu = 0.85$). Initial brake input is applied by the controller after 0.5 sec until the vehicle comes to a full stop after 5 sec. Generality of responses indicates that results are closely matched with results of software-in-the-loop (SiL) in section. Figure 6.8 shows the longitudinal slip levels of each wheel for the same stop. It can be seen that the controller gives a satisfactory performance of regulating the slip levels at the setpoint (dashed line). An overshoot in initial response of rear wheel slip is caused by pitch dynamic. Measured clamp forces from the actual EMB caliper is shown in Figure 6.9. A smooth and continuous control signal trajectory is again observed and a detailed view of measured clamp force from the EMB caliper to the commanded input demonstrates that an EMB caliper responds appropriately to the force modulation signal, see Figure 6.10. Plotted response of longitudinal slip vs. tyre friction force in Figure 6.2.1 shows the good performance of keeping the slip level within the optimal region.

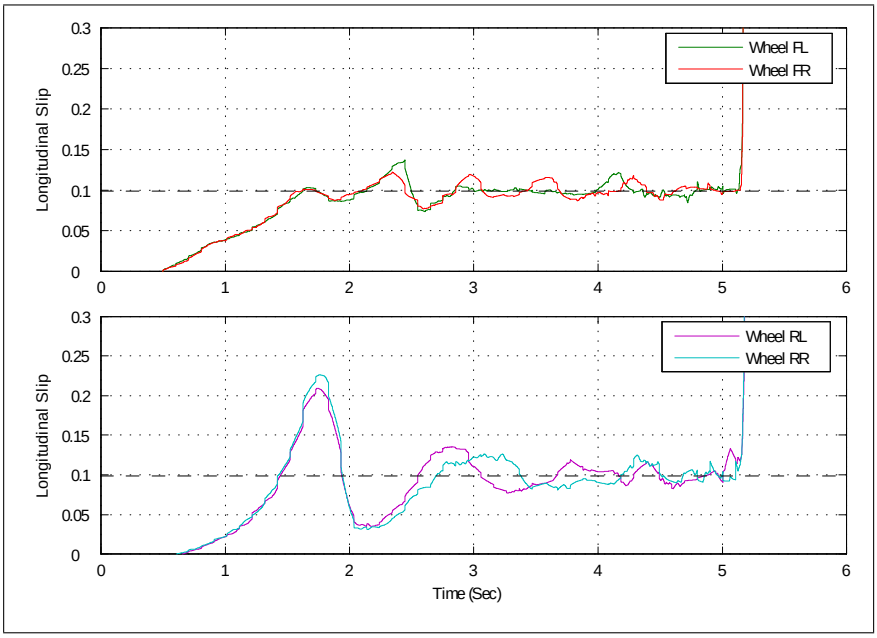


Figure 6.8: Simulated longitudinal slip responses of front wheels (FL,RL) and rear wheels (RL,RR) w.r.t slip setpoint of 0.1 (dashed line) on a high friction surface ($\mu = 0.85$).

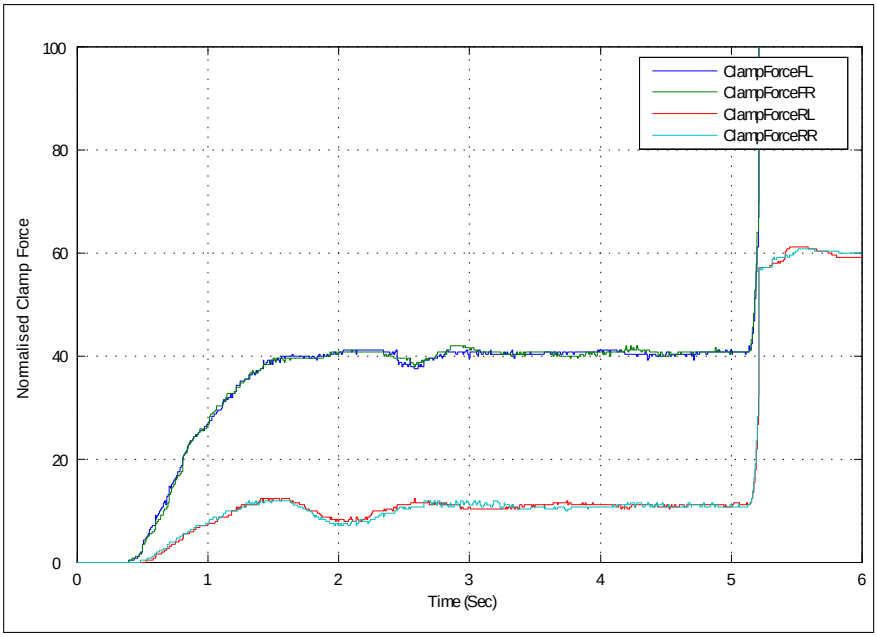


Figure 6.9: Measured clamp forces by the EMB calipers for a high friction surface ($\mu = 0.85$) with the parameter p tuned at 0.35 : Initial braking is applied around 0.5 sec

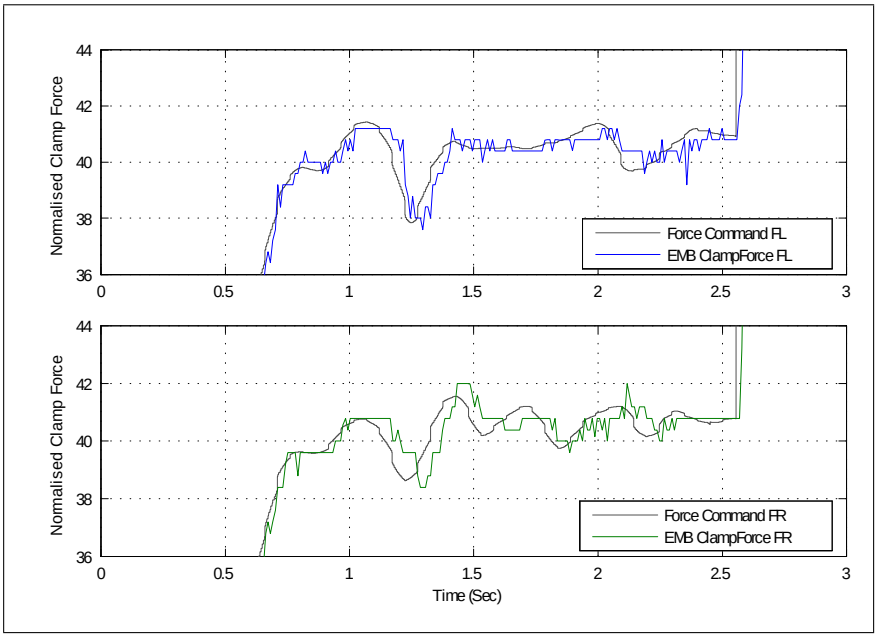
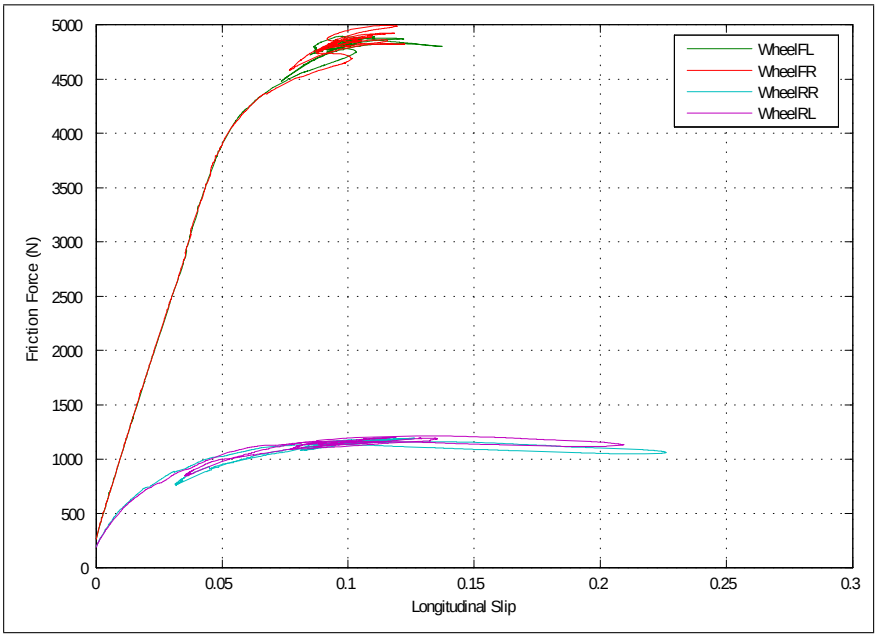


Figure 6.10: Amplified view of measured clamp forces by the front EMB calipers for a high friction surface ($\mu = 0.85$) with the parameter p tuned at 0.35



Plot of longitudinal slip vs. tyre friction force on high friction surface ($\mu = 0.85$) under the closed loop MPC control

6.2.2 Case B : Medium friction surface ($\mu = 0.5$)

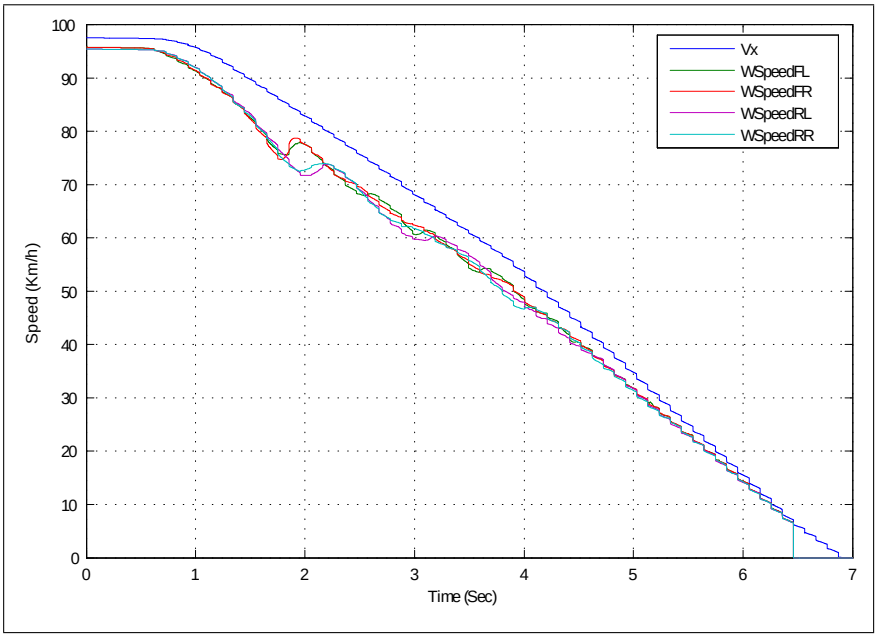


Figure 6.11: Plot of simulated vehicle speed (Vx) and wheel speeds (WSpeed) on a high friction surface ($\mu = 0.5$): Initial vehicle speed of 100 km/h

Performance of the tuned controller on a medium friction surface is shown in Figure 6.11. A significant variability of the slip responses are found in a lower speed region, hence to avoid this unstable responses, the slip set point is adjusted by the controller according to the vehicle speed. This property is clearly demonstrated in Figure 6.12. It can be seen that the slip setpoint is lowered to 0.06 from 0.08 of initial set point (dashed line) after 5 Sec. Consequence of lowering the slip setpoint is that less traction forces are exerted on the tyre, which in turn may increase the overall braking distance. However the unstable slip response promotes lateral instability and the loss of steerability of the vehicle, hence the controller performance is compromised. Applied clamp forces by the EMB calipers are shown in Figure 6.13. Initial braking is applied after 0.5 sec and generally a smooth control signal is observed.

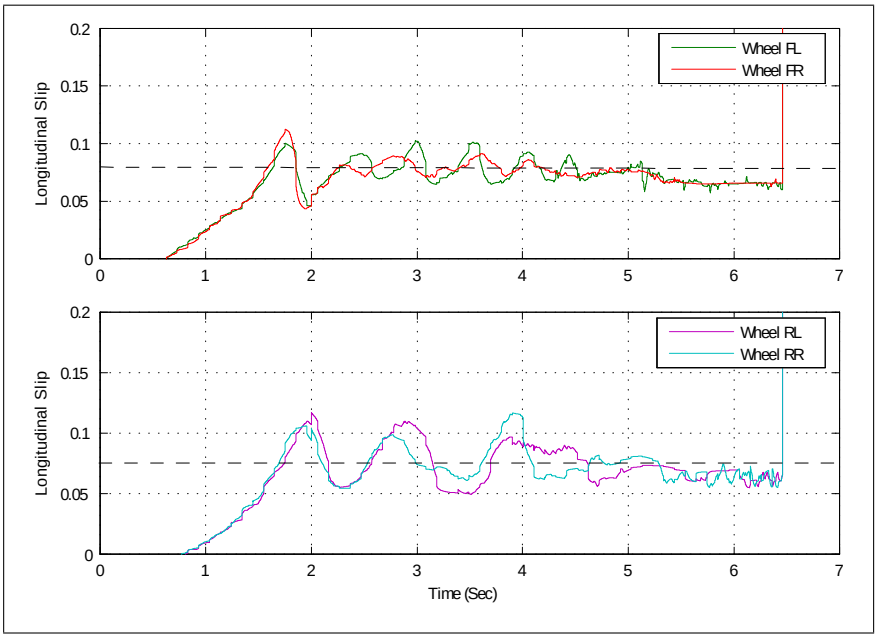


Figure 6.12: Simulated longitudinal slip responses of front wheels (FL,RL) and rear wheels (RL,RR) on a high friction surface ($\mu = 0.5$) w.r.t slip set point of 0.08 (dashed line)

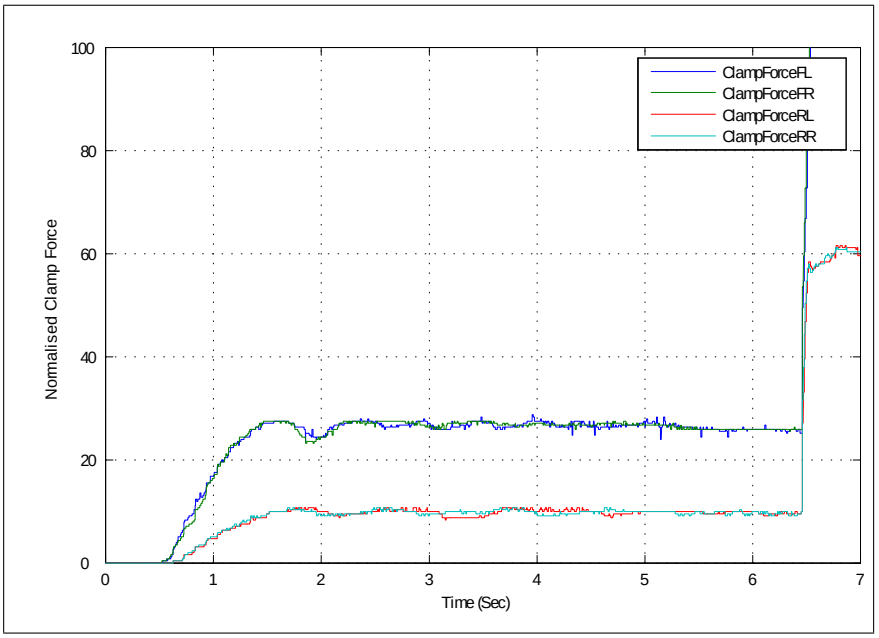


Figure 6.13: Measured clamp force by the EMB caliper for a high friction surface ($\mu = 0.5$)

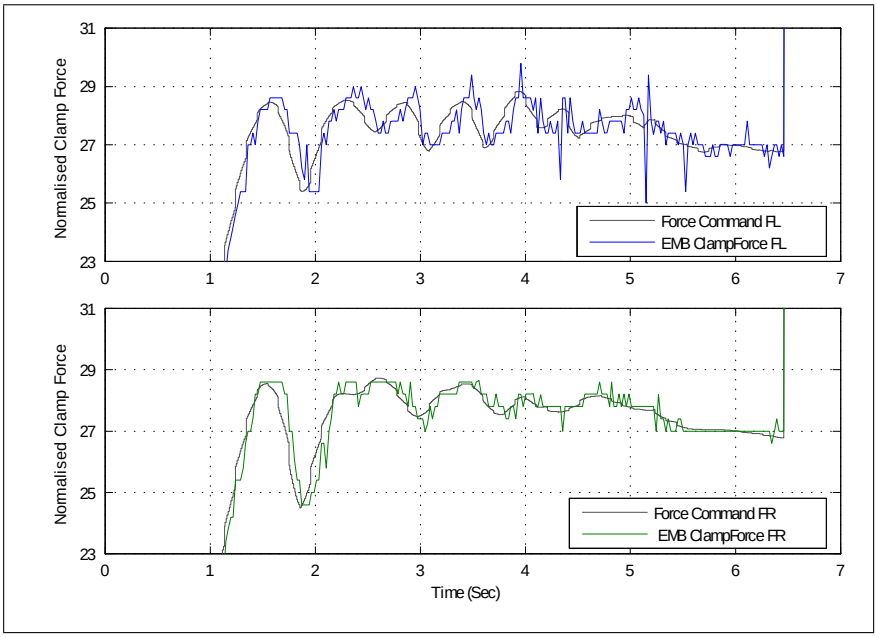
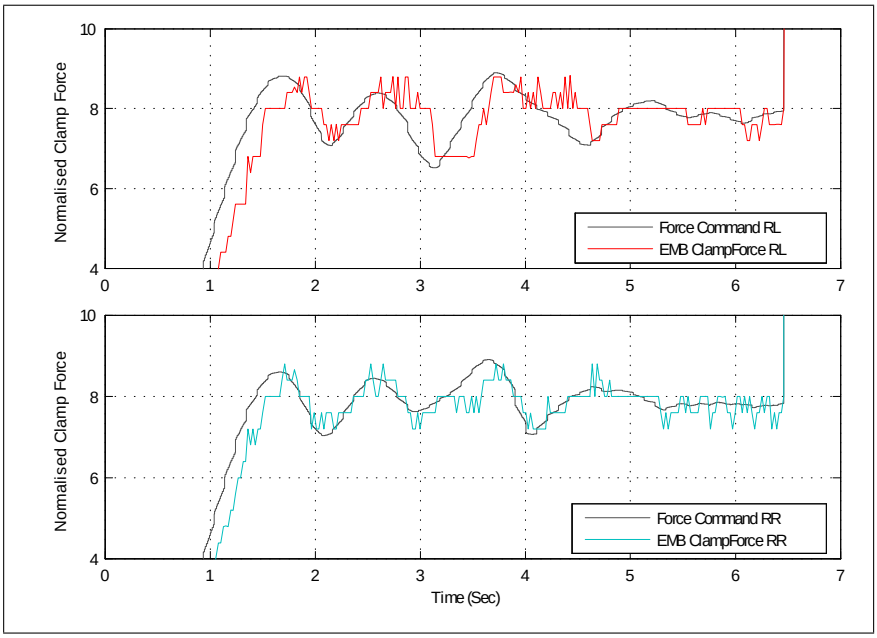


Figure 6.14: Amplified view of measured clamp force responses by the front EMB calipers to the force command input



Amplified view of measured clamp force responses from the rear EMB calipers to the force command input

6.2.3 Case C : Low friction surface ($\mu = 0.2$)

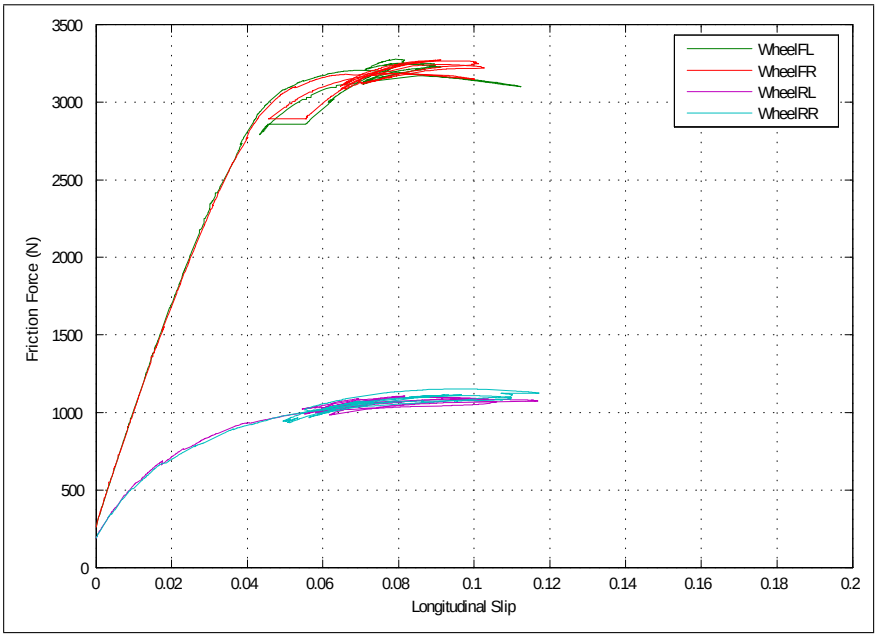


Figure 6.15: Plot of longitudinal slip vs. tyre friction force on high friction surface ($\mu = 0.5$)

Braking manoeuvre performed on low friction surface and corresponding responses of vehicle and wheel speeds are shown in Figure 6.16. Longitudinal slip responses on low friction surface is shown in Figure 6.17. At such low wheel slip and clamp force level, the control system becomes sensitive towards disturbances in actuator inaccuracies. Figure 6.18,6.19 and 6.20 illustrates the response of EMB caliper to the force command input.

6.3 Experimental Results of Wheel Slip Control

6.3.1 Case A : High friction surface ($\mu = 0.85$)

Figure 6.22 shows the logged values of vehicle and wheel speeds during an emergency braking manoeuvre on a high friction surface ($\mu = 0.85$). Results shown here indicates that after sudden braking input, rear wheels are becoming increasingly harder to avoid high slip when compared to the results of software simulation. This is mainly due to the slower response of the actual EMB caliper than the EMB simulation model. Figure 6.23 shows front and rear slip values of the same braking stop. Front wheels slip levels are maintained well around the optimal slip value of 0.1 (dashed line), however a larger overshoot in initial of rear wheels slip levels are noticed and slip levels are difficulty in

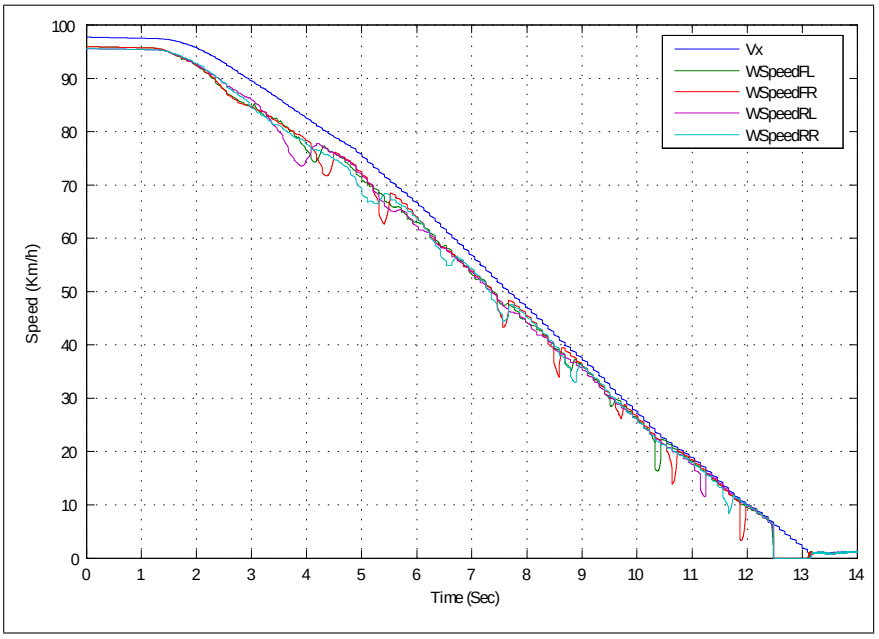


Figure 6.16: Plot of simulated vehicle speed (Vx) and wheel speeds (WSpeed) on a low friction surface ($\mu = 0.2$): Initial vehicle speed of 100 km

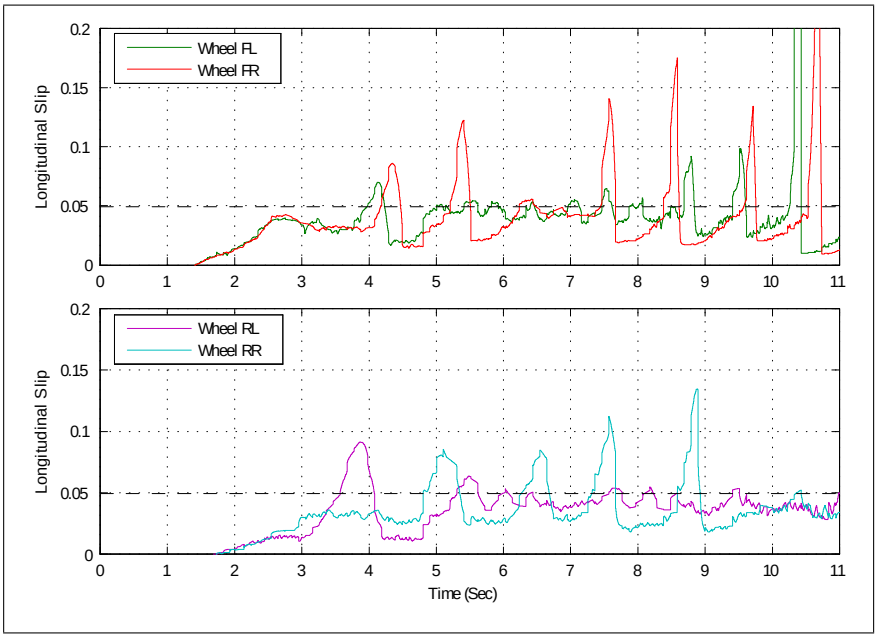


Figure 6.17: Simulated longitudinal slip response of front wheels (FL,RL) and rear wheels (RL,RR) on a high friction surface ($\mu = 0.2$) : Slip set point of 0.05 (dashed line)

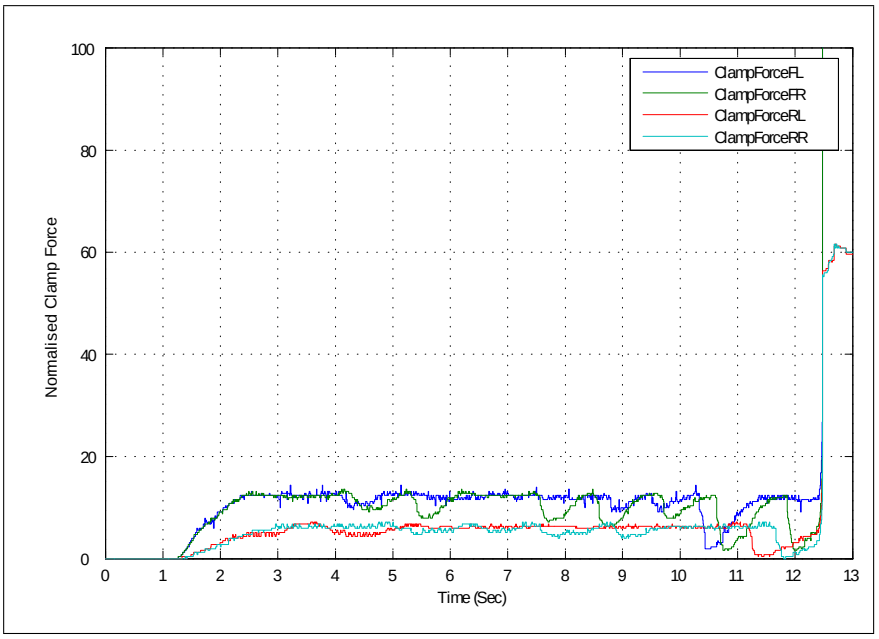


Figure 6.18: Measured clamp force by the EMB caliper for a low friction surface ($\mu = 0.2$)

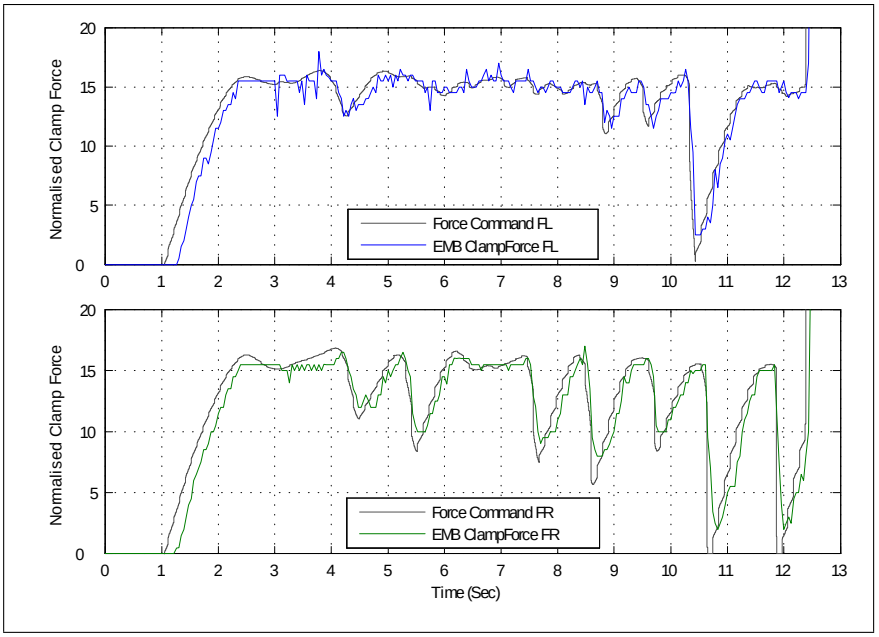


Figure 6.19: Amplified view of front EMB caliper response to the force command input

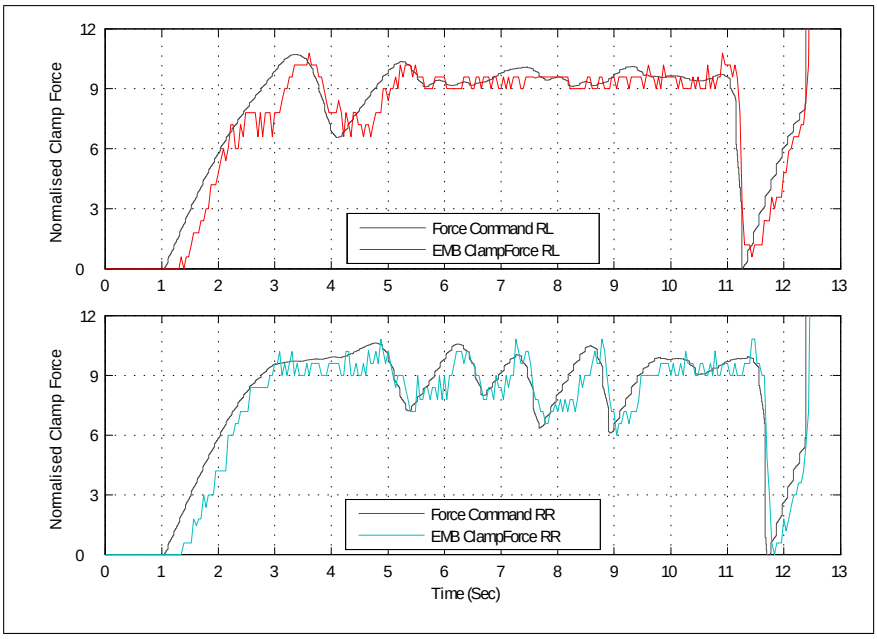


Figure 6.20: Amplified view of rear EMB clamp force response to the force command input

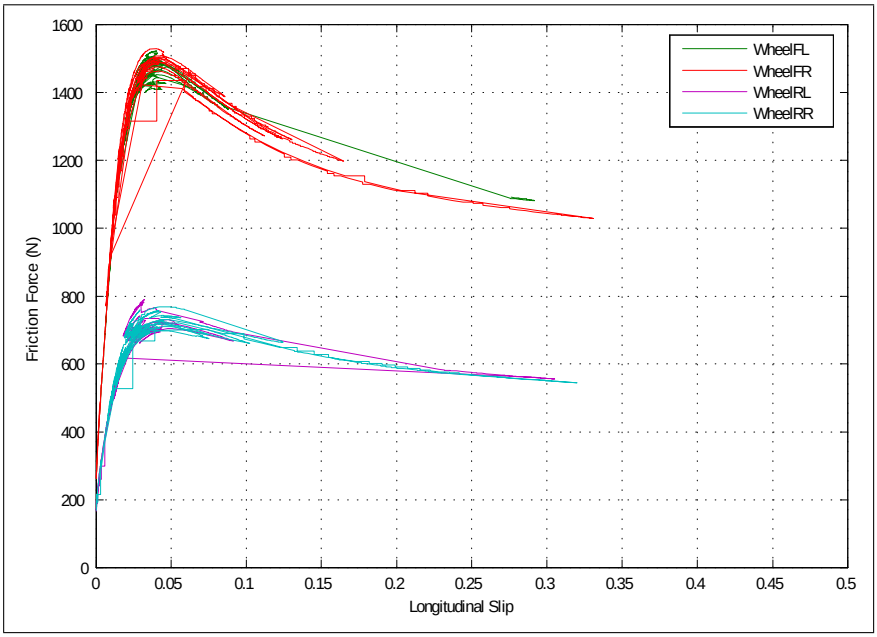


Figure 6.21: Plot of longitudinal slip vs. tyre friction force on low friction surface ($\mu = 0.2$)

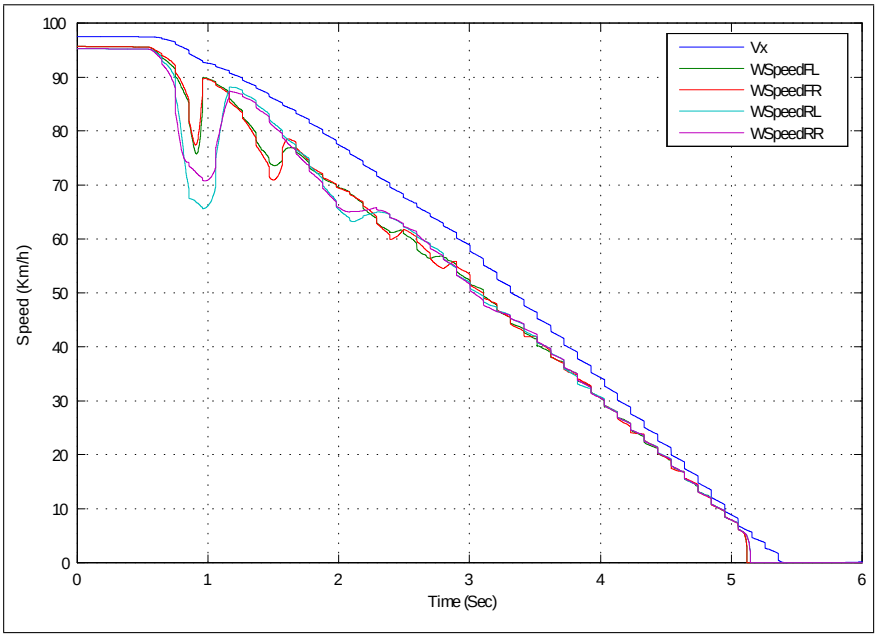


Figure 6.22: Plot of simulated vehicle (Vx) and wheel speeds (WSpeed) on a high friction surface ($\mu = 0.85$) for ABS braking manouevure.

maintaining at optimal level. Measured EMB caliper responses to the force command input by the driver (Force Demand) is seen in Figure 6.13. Plot shows the part of region where clamp forces of the EMB caliper no longer follows the driver command, hence indicates that unstable slip threshold is detected and the controller reduces the clamp forces as quickly as possible to avoid further venturing into unstable slip region. From that point, controller takes in charge of braking control until the vehicle comes to a full stop after 5 sec, which then releases the braking control to manually follow the driver's request. Figure 6.25 shows the good performance of regulating both tyre friction forces and longitudinal slip level within the peak regions of friction curve. Again, comparatively a larger oscillation of rear wheel slip responses are shown.

6.3.2 Case B : Medium friction surface ($\mu = 0.5$)

An ABS type braking manoeuvre is performed on a medium friction surface ($\mu = 0.5$). The driver presses the brake pedal abruptly after 1.5 sec. Values of the vehicle (Vx) and wheel speed (WSpeed FL,FR,RL,RR) from the HiL simulation is plotted in Figure 6.26. Analysing the longitudinal slip responses of each wheel, shown in Figure 6.27, it can be observed that the detection of unstable slip level is harder when compared to the high friction surface and this is mainly due to faster responses of the tyre's longitudinal slip dynamics with less available friction force. Plot of measured clamp

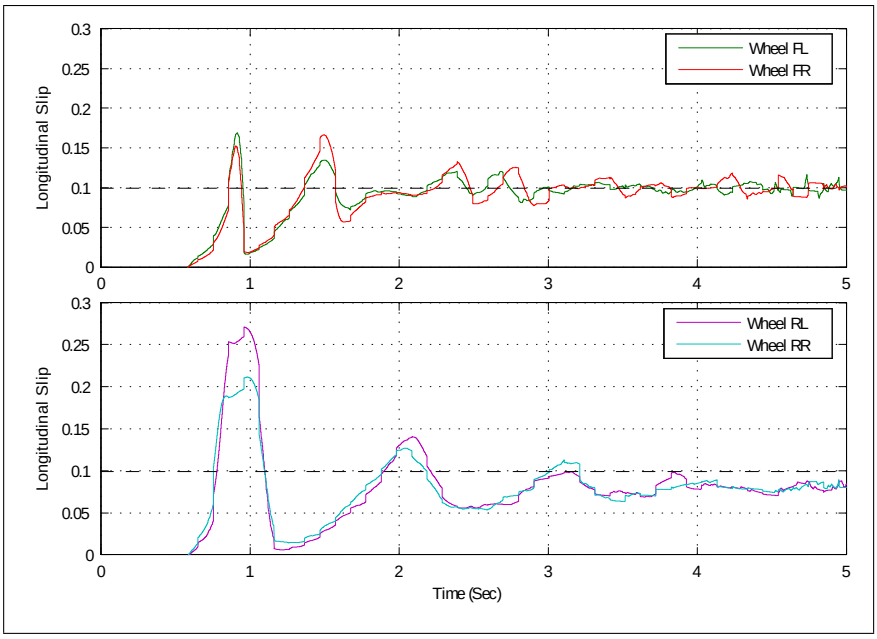


Figure 6.23: Simulated longitudinal slip responses of front and rear wheels w.r.t optimal slip setpoint of 0.1 (dashed line) on a high friction surface ($\mu = 0.85$) for ABS braking manoeuver:

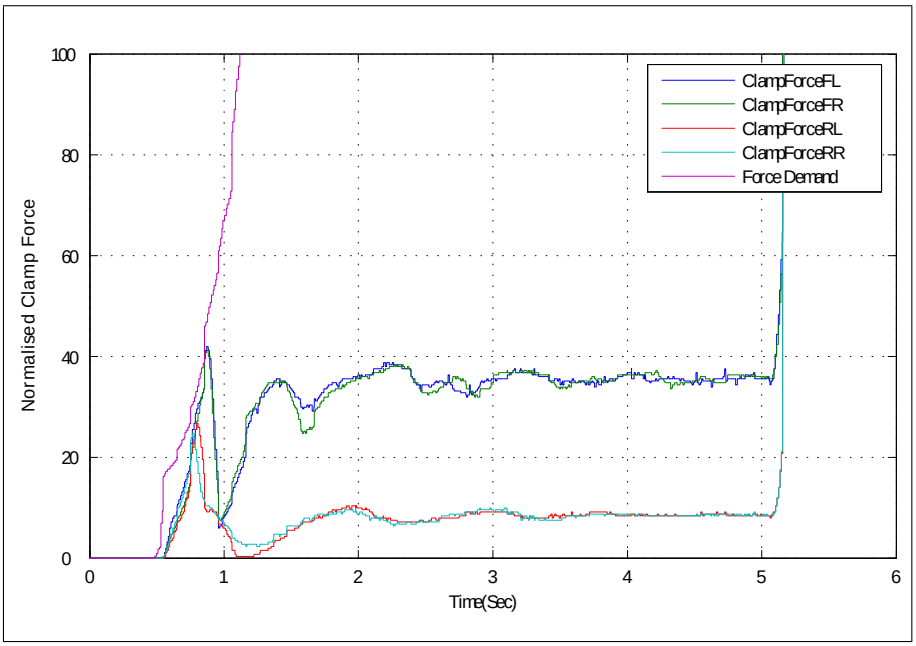


Figure 6.24: Plot of measured clamp forces by EMB brake calipers (ClampForce FL,FR,RL,RR) and driver brake request (Force Demand) during ABS braking for a high friction surface ($\mu = 0.85$)

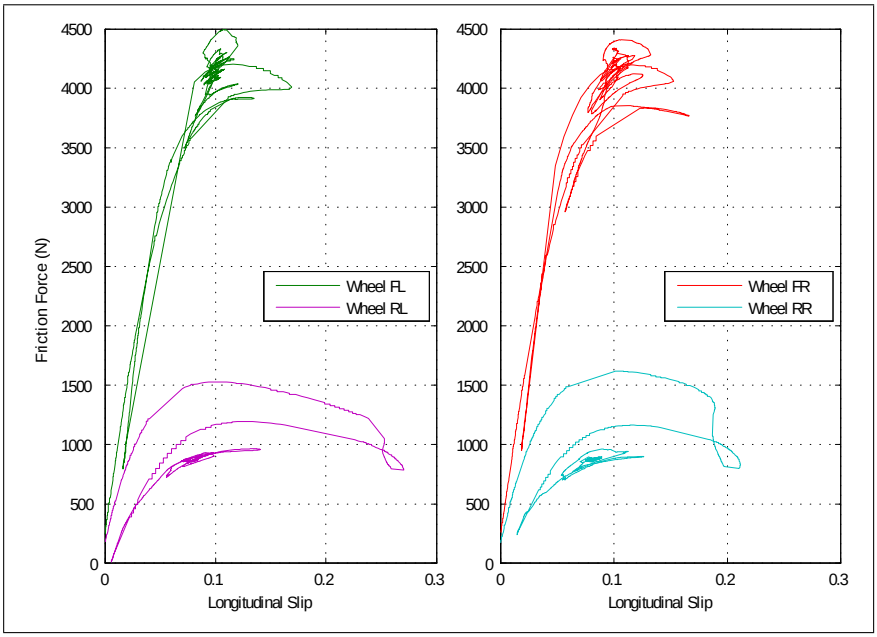


Figure 6.25: Characteristic friction force curve of front and rear wheels for ABS braking stop on high friction surface ($\mu = 0.85$)

forces from the EMB caliper is shown in Figure 6.28 where the period after the 7.5 sec, controller switches to the "off" mode allowing the maximum clamp forces requested from the driver to be applied at the wheels. Generality of slip performances shown in Figure 6.29.

6.3.3 Case C : Low friction surface ($\mu = 0.2$)

Antilock brake performance of the slip controller on a low friction surface is shown Figures 6.32, 6.30, 6.31, and 6.33. Responses are a more oscillatory compared to the SiL results. This is mainly due to the sensitivity of the slip responses towards the actuator inaccuracies and the induced pitch torque during the vehicle deceleration is much more evident. Setpoint is again lowered to avoid the unstable slip responses and lock-up of wheels in a lower speed region.

6.4 Concluding Remark

In this chapter it is shown that the proposed MPC wheel slip controller can successfully control the real BBW system. The dynamic response of EMB actuators are found to be very similar to the simulation model that was used in Chapter 4 and 5. A high accurate setting of EMB actuators and real-time framework of BBW system have

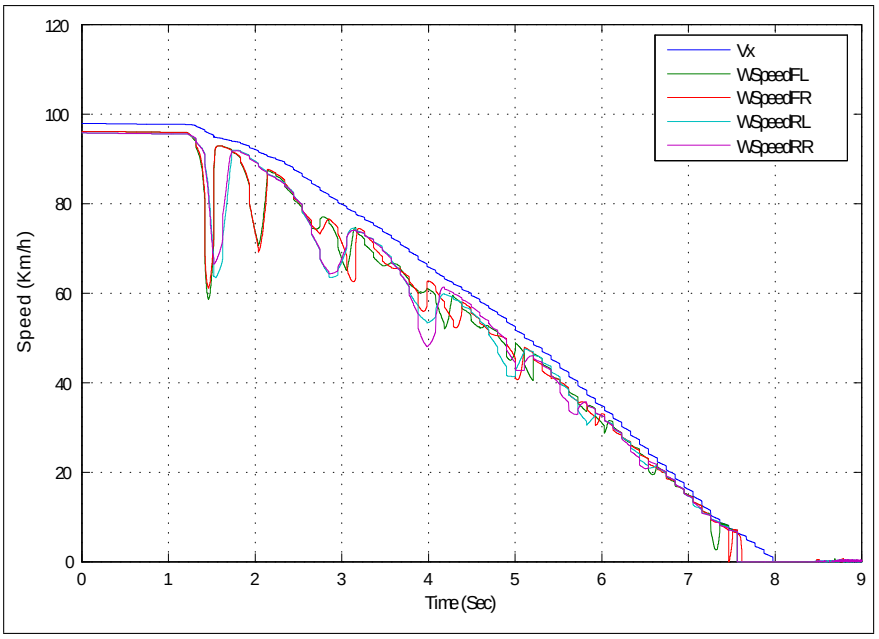


Figure 6.26: Plot of simulated vehicle (Vx) and wheel speeds (WSpeed) on a medium friction surface ($\mu = 0.5$) for ABS braking manouevure.

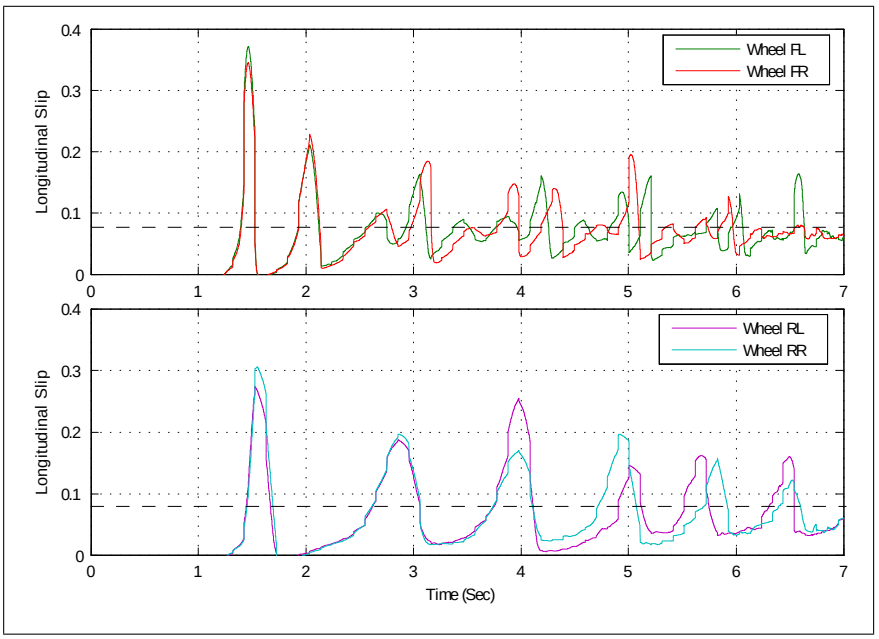


Figure 6.27: Simulated longitudinal slip responses of front and rear wheels w.r.t optimal slip setpoint of 0.08 (dashed line) on a medium friction surface ($\mu = 0.5$) for ABS braking manoeuver:

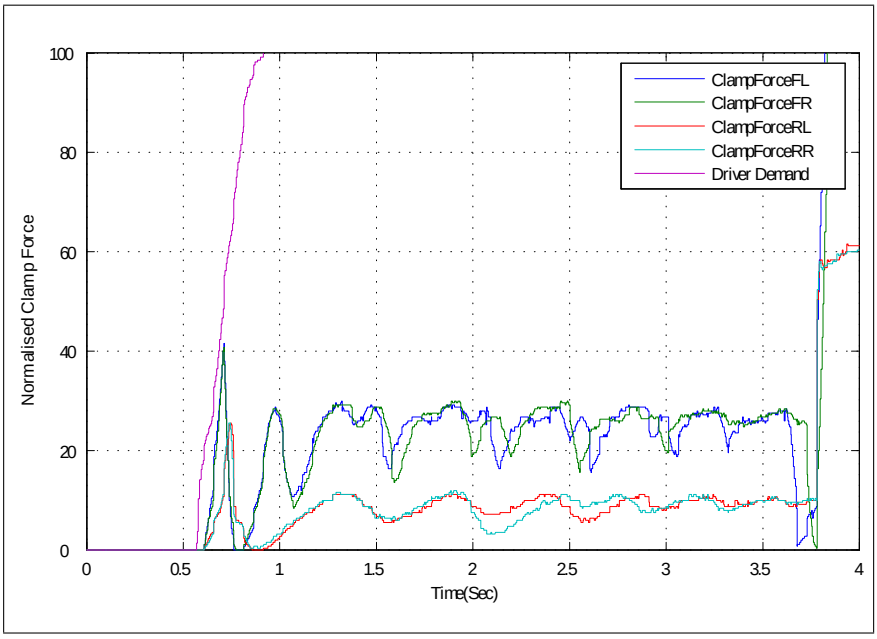


Figure 6.28: Plot of measured clamp forces by EMB brake calipers (ClampForce FL,FR,RL,RR) and driver brake request (Force Demand) during ABS braking for a medium friction surface ($\mu = 0.5$)

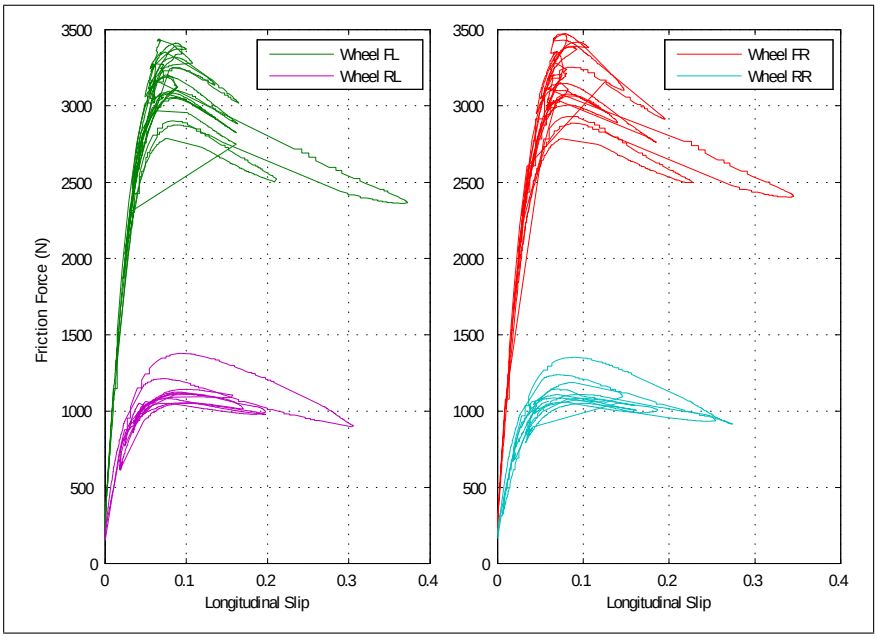


Figure 6.29: Characteristic friction force curve of front and rear wheels for ABS braking stop on medium friction surface ($\mu = 0.5$)

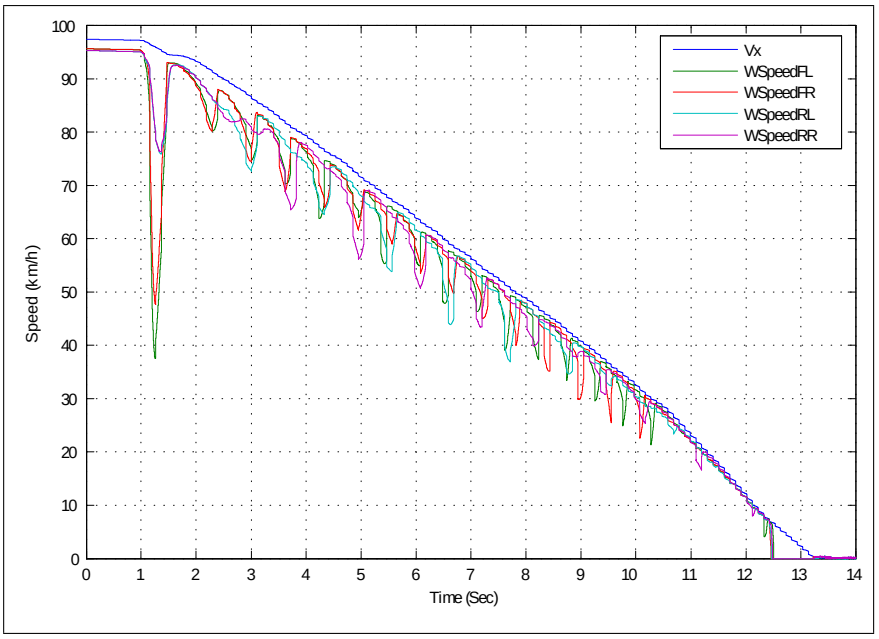


Figure 6.30: Plot of simulated vehicle (Vx) and wheel speeds (WSpeed) on a low friction surface ($\mu = 0.2$) for ABS braking manouevure.

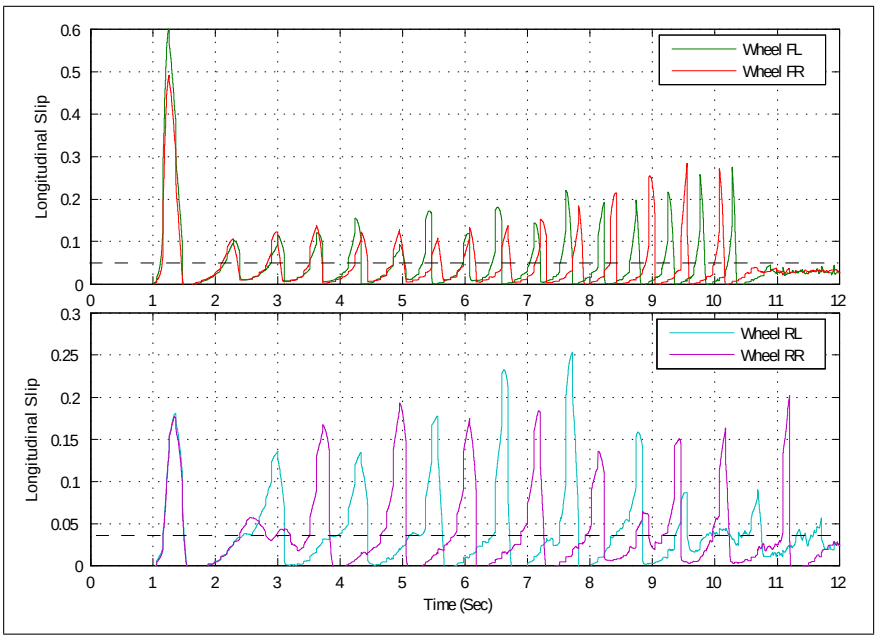


Figure 6.31: Simulated longitudinal slip responses of front and rear wheels w.r.t optimal slip setpoint of 0.05 (dashed line) on a low friction surface ($\mu = 0.2$) for ABS braking manoeuver:

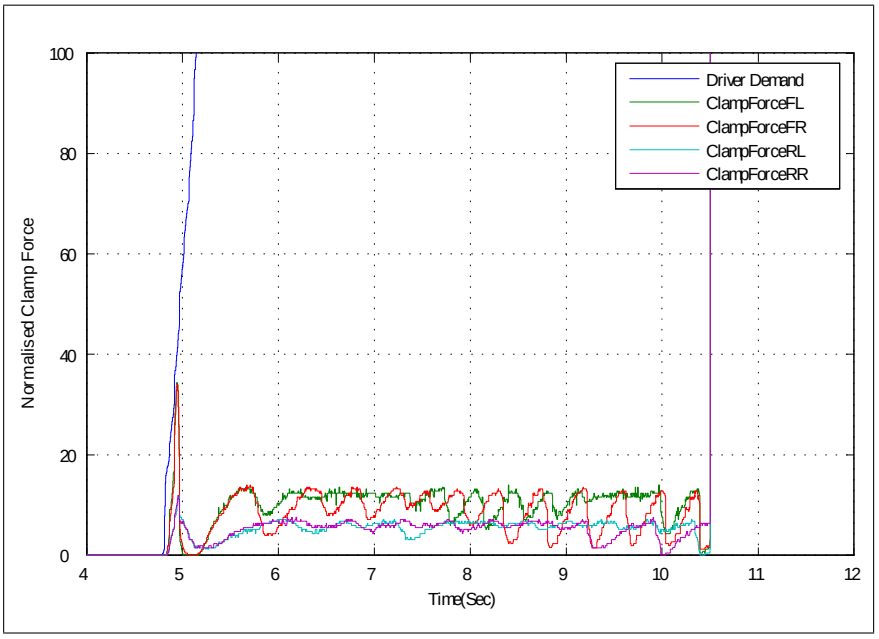


Figure 6.32: Plot of measured clamp forces by EMB brake calipers (ClampForce FL,FR,RL,RR) and driver brake request (Force Demand) during ABS braking for a low friction surface ($\mu = 0.2$)

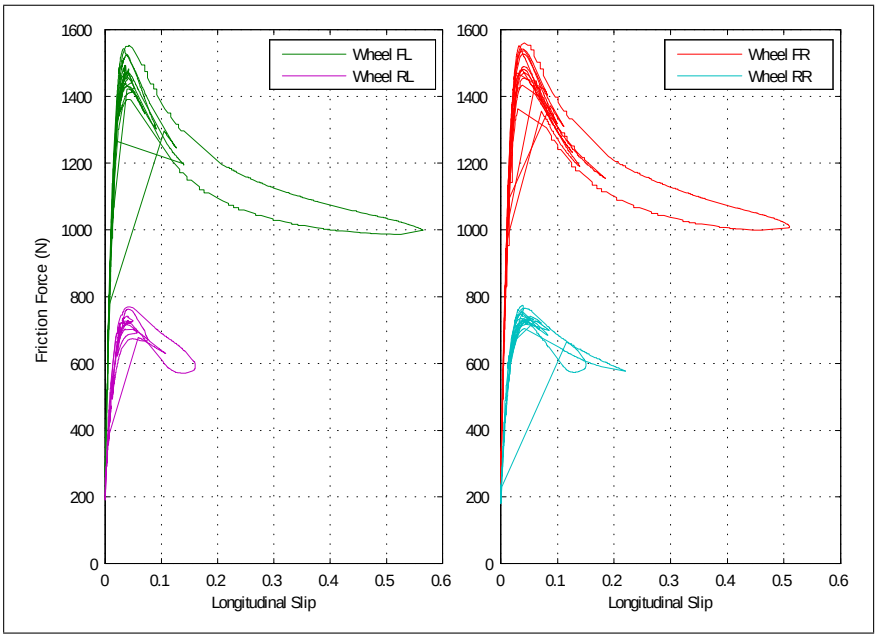


Figure 6.33: Characteristic friction force curve of front and rear wheels for ABS braking stop on a low friction surface ($\mu = 0.2$)

allowed satisfactorily to regulate the longitudinal slip level at the optimum point and tuning parameters obtained in Chapter 4 proved to be a good starting point, and only needed a minor adjustment.

Chapter 7

Conclusions

The central idea of this thesis has been the modelling, design and implementation of a model based wheel slip controller. A linear wheel slip model was built which included tyre transient behaviour and actuator dynamics. Based on a linear wheel slip model, a model based wheel slip controller is designed using a generic continuous model predictive control algorithm and also the supervisory logic was developed to comply with the safety aspects of the wheel slip control system. The complete system is evaluated through a Software-In-the-Loop (SIL) simulation on a multiprocessor architecture. This provided the topology required to be able to run the system as a combination of software and hardware-in-the-loop simulation(HIL), the EMB caliper. The simulation results from SIL and combined HIL proved to be a good match. MPC showed to be robust for a change in actuator dynamics and outperformed a PID controller.

7.1 Future work

1. To implement the controller in a digital environment and test it on a real vehicle.
2. Lateral stability controller can be easily devised based on the implemented control architecture.
3. In order to fully validate the proposed controller, a more extended test procedure is required such as a mu transition and split mu condition
4. Asynchronous distributed simulation algorithm can be developed to increase the simulation throughput and to provide a even greater flexibility for the hardware in the loop simulation.

Appendix A

Matlab Code

A.1 Distributed Simulation Algorithm

A.1.1 Local task scheduler

```
/* S-function implementation of a local task scheduler*/

#define         S_FUNCTION_LEVEL        2
#undef          S_FUNCTION_NAME
#define         S_FUNCTION_NAME         Chassis_tskScheduler

#include        <stddef.h>
#include        <stdlib.h>
#include        <math.h>
#include        <string.h>
#include        "simstruc.h"

#ifdef          MATLAB_MEX_FILE
#include        "mex.h"
#endif

#ifndef         MATLAB_MEX_FILE
#include        <windows.h>
#include        "io_xpcimport.h"
#include        "pci_xpcimport.h"
#endif

/* Input Arguments */

#define NUMBER_OF_ARGS      (1)
#define SLOT_ARG            ssGetSFcnParam(S, 0)

#define SAMP_TIME_IND       (0)
```

```
#define BASE_ADDR_IND       (0)

#define NO_I_WORKS          (5)
#define BASE_ADDR_RFM_CONT  (0)
#define INTERRUPT_NUMBER    (1)
#define TARGET_NODE         (2)
#define TASK1_COUNTER       (3)
#define TASK2_COUNTER       (4)

#define NO_R_WORKS          (0)
#define NO_P_WORKS          (0)


//  Macros for command group of board registers that controls
//  the reflective memory
//  Note - requires rfmcontrolregisters to be defined
//  as a pointer to bytes (char *)

#define RFM_BRV    (*((uint8_T *)(rfmcontrolregisters+0x00)))
#define RFM_BID    (*((uint8_T *)(rfmcontrolregisters+0x01)))
#define RFM_NID    (*((uint8_T *)(rfmcontrolregisters+0x04)))
#define RFM_LCSR1  (*((uint32_T *)(rfmcontrolregisters+0x08)))
#define RFM_LISR   (*((uint32_T *)(rfmcontrolregisters+0x10)))

#define RFM_NTD    (*((uint32_T *)(rfmcontrolregisters+0x18)))
#define RFM_NTN    (*((uint8_T *)(rfmcontrolregisters+0x1C)))
#define RFM_NIC    (*((uint8_T *)(rfmcontrolregisters+0x1D)))
#define RFM_SID1   (*((uint8_T *)(rfmcontrolregisters+0x24)))
#define RFM_SID2   (*((uint8_T *)(rfmcontrolregisters+0x2C)))
#define RFM_SID3   (*((uint8_T *)(rfmcontrolregisters+0x34)))
#define RFM_INITD  (*((uint8_T *)(rfmcontrolregisters+0x3C)))

static char_T msg[256];

static void mdlInitializeSizes(SimStruct *S)
{

 #ifndef MATLAB_MEX_FILE
 #include "io_xpcimport.c"
 #include "pci_xpcimport.c"
 #endif

 ssSetNumSFcnParams(S, NUMBER_OF_ARGS);

 if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
```

```
      sprintf(msg,"Wrong number of input arguments passed.\n"
                "%d arguments are expected\n",NUMBER_OF_ARGS);
      ssSetErrorStatus(S,msg);
      return;
 }
    ssSetNumOutputPorts(S,4);
    ssSetNumContStates(S,0);
    ssSetNumDiscStates(S,0);

    ssSetOutputPortWidth(S,0,3);

    ssSetOutputPortDataType(S,1,SS_UINT8);
    ssSetOutputPortWidth(S,1,1);

    ssSetOutputPortDataType(S,2,SS_UINT8);
    ssSetOutputPortWidth(S,2,1);

    ssSetOutputPortDataType(S,3,SS_UINT8);
    ssSetOutputPortWidth(S,3,1);

    ssSetNumSampleTimes(S,1);

    ssSetNumRWork(S,NO_R_WORKS);
    ssSetNumIWork(S,NO_I_WORKS);
    ssSetNumPWork(S,NO_P_WORKS);

    ssSetNumModes(S,0);

    ssSetOptions(S,SS_OPTION_EXCEPTION_FREE_CODE|SS_OPTION_PLACE_ASAP);
    return;
 }

// Configures the output port of the block to be

static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S,0,0.01);
    ssSetOffsetTime(S,0,0);

    ssSetExplicitFCSSCtrl(S,1);
    ssSetCallSystemOutput(S,0);
    ssSetCallSystemOutput(S,1);
    ssSetCallSystemOutput(S,2);
}
```

```
#define MDL_START
static void mdlStart(SimStruct *S)
{

#ifndef MATLAB_MEX_FILE

 PCIDeviceInfo pciinfo;
 void *Physical1;
 void *Virtual1;
 volatile unsigned long *ioaddress32;
 volatile uint8_T  *rfmcontrolregisters;

// Base Address 2 (define as byte-wide to simplify offset calculation)
 int32_T interrupt_number,target_node;
 const char *device_name = "VMIC 5565";
 const unsigned short vendor_id = 0x114A;
 const unsigned short device_id = 0x5565;

 if ((int_T)mxGetPr(SLOT_ARG)[0]<0) {

        /* Look for the PCI-Device */
 if (rl32eGetPCIInfo(vendor_id,device_id,&pciinfo)) {
   sprintf(msg,"%s: board not present", device_name);
   ssSetErrorStatus(S,msg);
   return;
   }
 }
 else {
   int_T bus;
   int_T slot;

   if (mxGetN(SLOT_ARG) == 1) {
       bus = 0;
       slot = (int_T)mxGetPr(SLOT_ARG)[0];
   }
   else {
       bus = (int_T)mxGetPr(SLOT_ARG)[0];
       slot = (int_T)mxGetPr(SLOT_ARG)[1];
   }

   // look for the PCI-Device
  if (rl32eGetPCIInfoAtSlot(vendor_id,device_id,(slot & 0xff)
          | ((bus & 0xff)<< 8),&pciinfo)) {
    sprintf(msg,"%s (bus %d, slot %d)
```

```
            : board not present",device_name, bus, slot );
    ssSetErrorStatus(S,msg);
    return;
    }
  }
    Physical1=(void *)pciinfo.BaseAddress[2];
    Virtual1 = rl32eGetDevicePtr(Physical1, 0x40, RT_PG_USERREADWRITE);
    rfmcontrolregisters=(uint8_T *)Physical1;
    ssSetIWorkValue(S, BASE_ADDR_RFM_CONT,(uint_T)rfmcontrolregisters);
    ssSetIWorkValue(S, TASK1_COUNTER,0);

    RFM_LISR = 0x0000;
    RFM_SID1 = 0x0;
    RFM_SID2 = 0x0;
    RFM_SID3 = 0x0;
    RFM_INITD = 0x0;

#endif
}

static void mdlOutputs(SimStruct *S, int_T tid)
{

#ifndef MATLAB_MEX_FILE

uint8_T  *y3 =  (uint8_T *)ssGetOutputPortSignal(S,3);
uint8_T  *y1 =  (uint8_T *)ssGetOutputPortSignal(S,1);
uint8_T  *y2 =  (uint8_T *)ssGetOutputPortSignal(S,2);
uint8_T  task1_counter = ssGetIWorkValue(S,TASK1_COUNTER);
uint8_T  task2_counter = ssGetIWorkValue(S,TASK2_COUNTER);

volatile uint8_T *rfmcontrolregisters =
                  ssGetIWorkValue(S,BASE_ADDR_RFM_CONT);

UNUSED_ARG(tid);


if (RFM_LISR == 0x0001)
 {
    *y1 = RFM_SID1;
    RFM_SID1 = 0x0;
    RFM_LISR = 0x0000;
    if (!ssCallSystemWithTid(S,1,tid)) {
    /* Error occurred which will be reported by Simulink */
       return;}
```

```
    RFM_NTN = 0x11;
    RFM_NIC = 0x1;
  }
  else if (RFM_LISR == 0x0004)
  {
    *y1 = RFM_SID3;
    RFM_SID3 = 0x0;
    RFM_LISR = 0x0000;

    if (!ssCallSystemWithTid(S,2,tid)) {
      /* Error occurred which will be reported by Simulink */
            return;}

    RFM_NTN = 0x11;
    RFM_NIC = 0x3;
  }
  else if (RFM_LISR == 0x0080)
  {
    *y1 = RFM_INITD;
    RFM_INITD = 0x0;
    RFM_LISR = 0x0000;

    if (!ssCallSystemWithTid(S,0,tid)) {
    /* Error occurred which will be reported by Simulink */
       return;}

  }
  else
  {
     *y1 = RFM_SID2;
      RFM_SID2 = 0x0;
  }

#endif
}


#ifdef MATLAB_MEX_FILE  /* Is this file being compiled as a MEX-file?*/
#include "simulink.c"   /* Mex glue */
#else
#include "cg_sfun.h"    /* Code generation glue */
#endif
```

A.1.2 Global task scheduler

```
/* mem5565netboardcast.c - S-Func for generation of
 *    network interrupts with the VMIC 5565 Shared memory
 *    board
 *
 * See also:
 * mem5565netwrite.c,mem5565netread.c, meme5565netinit.c
 * and ..\src\xpcvmic5565.c
 */

/* Copyright 1994-2003 The MathWorks, Inc.
 * $Revision: 1.1 $ $Date: 2003/10/10 14:35:30 $
 */


#define         S_FUNCTION_LEVEL        2
#undef          S_FUNCTION_NAME
#define         S_FUNCTION_NAME         Global_tskScheduler

#include        <stddef.h>
#include        <stdlib.h>

#include        "simstruc.h"

#ifdef          MATLAB_MEX_FILE
#include        "mex.h"
#endif

#ifndef         MATLAB_MEX_FILE
#include        <windows.h>
#include        "io_xpcimport.h"
#include        "pci_xpcimport.h"
#endif

/* Input Arguments */
#define NUMBER_OF_ARGS          (2)
#define SLOT_ARG          ssGetSFcnParam(S, 0)
#define SAMPLETIME_ARG    ssGetSFcnParam(S, 1)

#define SAMP_TIME_IND           (0)
#define BASE_ADDR_IND           (0)

#define NO_I_WORKS              (3)
#define BASE_ADDR_RFM_CONT      (0)
#define INTERRUPT_NUMBER        (1)
```

```
#define TARGET_NODE             (2)

#define NO_R_WORKS              (6)
#define RM_LISR                 (0)
#define RM_SENDER_ID            (1)
#define TASK_ID                 (2)
#define TASK_ACTIVE             (3)
#define ACTIVE_NODE             (4)
#define CYCLE                   (5)

#define NO_P_WORKS              (0)


// Macros for command group of board registers that control
// shared memory
//   Note - requires rfmcontrolregisters to be defined
//   as a pointer to bytes (char *)

#define RFM_BRV    (*((uint8_T *)(rfmcontrolregisters+0x00)))
#define RFM_BID    (*((uint8_T *)(rfmcontrolregisters+0x01)))
#define RFM_NID    (*((uint8_T *)(rfmcontrolregisters+0x04)))
#define RFM_LCSR1  (*((uint32_T *)(rfmcontrolregisters+0x08)))
#define RFM_LISR   (*((uint32_T *)(rfmcontrolregisters+0x10)))

#define RFM_NTD    (*((uint32_T *)(rfmcontrolregisters+0x18)))
#define RFM_NTN    (*((uint8_T *)(rfmcontrolregisters+0x1C)))
#define RFM_NIC    (*((uint8_T *)(rfmcontrolregisters+0x1D)))
#define RFM_SID1    (*((uint8_T *)(rfmcontrolregisters+0x24)))
#define RFM_SID2    (*((uint8_T *)(rfmcontrolregisters+0x2C)))
#define RFM_SID3   (*((uint8_T *)(rfmcontrolregisters+0x34)))
#define RFM_INITD   (*((uint8_T *)(rfmcontrolregisters+0x3C)))

static char_T msg[256];


static void mdlInitializeSizes(SimStruct *S)
{


 #ifndef MATLAB_MEX_FILE
 #include "io_xpcimport.c"
 #include "pci_xpcimport.c"
 #endif

 ssSetNumSFcnParams(S, NUMBER_OF_ARGS);
```

```
 if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
     sprintf(msg,"Wrong number of input arguments passed.\n"
           "%d arguments are expected\n",NUMBER_OF_ARGS);
     ssSetErrorStatus(S,msg);
     return;
 }
ssSetNumContStates(S, 0);
ssSetNumDiscStates(S, 0);

ssSetNumOutputPorts(S, 4);
ssSetOutputPortDataType(S, 0, SS_UINT8);
ssSetOutputPortWidth(S, 0, 1);

ssSetOutputPortDataType(S, 1, SS_UINT8);
ssSetOutputPortWidth(S, 1, 1);

ssSetOutputPortDataType(S, 2, SS_UINT8);
ssSetOutputPortWidth(S, 2, 1);

ssSetOutputPortDataType(S, 3, SS_UINT8);
ssSetOutputPortWidth(S, 3, 1);

ssSetNumSampleTimes(S, 1);

ssSetNumRWork(S, NO_R_WORKS);
ssSetNumIWork(S, NO_I_WORKS);
ssSetNumPWork(S, NO_P_WORKS);


ssSetNumModes(S, 0);
ssSetNumNonsampledZCs(S, 0);

ssSetOptions(S,SS_OPTION_EXCEPTION_FREE_CODE|SS_OPTION_PLACE_ASAP);
return;
}


static void mdlInitializeSampleTimes(SimStruct *S)
{

 if (mxGetPr(SAMPLETIME_ARG)[0]==-1.0) {
    ssSetSampleTime(S, 0, 0.01);
    ssSetOffsetTime(S, 0, 0);
 }
 else {
```

```
    ssSetSampleTime(S, 0, mxGetPr(SAMPLETIME_ARG)[0]);
    ssSetOffsetTime(S, 0, 0.0);
 }
}

#define MDL_START
static void mdlStart(SimStruct *S)
{

#ifndef MATLAB_MEX_FILE

 PCIDeviceInfo pciinfo;
 void *Physical1;
 void *Virtual1;
 volatile unsigned long *ioaddress32;
 volatile uint8_T  *rfmcontrolregisters;
 int32_T interrupt_number,target_node;

 const char *device_name = "VMIC 5565";
 const unsigned short vendor_id = 0x114A;
 const unsigned short device_id = 0x5565;

 if ((int_T)mxGetPr(SLOT_ARG)[0]<0) {
     /* look for the PCI-Device */
     if (rl32eGetPCIInfo(vendor_id,device_id,&pciinfo)) {
          sprintf(msg,"%s: board not present", device_name);
         ssSetErrorStatus(S,msg);
         return;
     }
 }
 else {

   int_T bus;
   int_T slot;
   if (mxGetN(SLOT_ARG) == 1) {
        bus = 0;
        slot = (int_T)mxGetPr(SLOT_ARG)[0];
    } else {
        bus = (int_T)mxGetPr(SLOT_ARG)[0];
        slot = (int_T)mxGetPr(SLOT_ARG)[1];
    }

   if (rl32eGetPCIInfoAtSlot(vendor_id,device_id,(slot & 0xff)
                     |((bus & 0xff)<< 8),&pciinfo)) {
   sprintf(msg,"%s (bus %d, slot %d)
```

```
                : board not present",device_name, bus, slot);
        ssSetErrorStatus(S,msg);
        return;
        }
    }

 Physical1=(void *)pciinfo.BaseAddress[2];
 Virtual1 = rl32eGetDevicePtr(Physical1, 0x40, RT_PG_USERREADWRITE);
 rfmcontrolregisters=(uint8_T *)Physical1;

 ssSetIWorkValue(S, BASE_ADDR_RFM_CONT,(uint_T)rfmcontrolregisters);
 ssSetRWorkValue(S, TASK_ID,0);
 ssSetRWorkValue(S, TASK_ACTIVE,0);
 ssSetRWorkValue(S, ACTIVE_NODE,0);
 ssSetRWorkValue(S, CYCLE,0);

 RFM_SID1 = 0x0;
 RFM_SID2 = 0x0;
 RFM_SID3 = 0x0;
 RFM_INITD = 0x0;
 RFM_LISR = 0x0000;

 RFM_NIC = 0x8 | 0x7;

#endif

}

static void mdlOutputs(SimStruct *S, int_T tid)
{


#ifndef MATLAB_MEX_FILE

volatile uint8_T *rfmcontrolregisters =
                  ssGetIWorkValue(S,BASE_ADDR_RFM_CONT);
uint8_T  *y1 =  (uint8_T *)ssGetOutputPortSignal(S,0);
uint8_T  *y2 =  (uint8_T *)ssGetOutputPortSignal(S,1);
uint8_T  *y3 =  (uint8_T *)ssGetOutputPortSignal(S,2);
uint8_T  *y4 =  (uint8_T *)ssGetOutputPortSignal(S,3);

real_T task_ID = ssGetRWorkValue(S, TASK_ID);
real_T task_active = ssGetRWorkValue(S,TASK_ACTIVE);
real_T active_node = ssGetRWorkValue(S,ACTIVE_NODE);
real_T active_cycle = ssGetRWorkValue(S,CYCLE);
```

```
uint8_T node_no = 0;

*y3 = task_ID;
*y2 = active_cycle;

if(task_ID ==0)
{
 if(task_active == 0)
 {
    if(RFM_LISR == 0)
     {
        // Body Task
        ssSetRWorkValue(S, TASK_ID,1);
        ssSetRWorkValue(S, TASK_ACTIVE,1);
        RFM_NTN = 0x6;
        RFM_NIC = 0x1;
     }
 }
}
else
{
  if(task_active == 1)
  {
    if(RFM_LISR == 1)
    {
      if(task_ID ==1)
      {
        active_cycle = active_cycle + 1;
        ssSetRWorkValue(S,CYCLE,active_cycle);
        ssSetRWorkValue(S, TASK_ID,2);
        ssSetRWorkValue(S, ACTIVE_NODE,4);
        *y2 = RFM_SID1;
        RFM_NIC = 0x8 | 0x2;
      }
    }
    else if(RFM_LISR == 2)
    {
      if(task_ID == 2)
      {
        node_no = RFM_SID2;
        switch (node_no) {

          case 2:
          active_node = active_node - 1;
```

```
          ssSetRWorkValue(S, ACTIVE_NODE,active_node);
          *y1 = node_no;
          break;

          case 3:
          active_node = active_node - 1;
          ssSetRWorkValue(S, ACTIVE_NODE,active_node);
          *y2 = node_no;
          break;

          case 32:
          active_node = active_node - 1;
          ssSetRWorkValue(S, ACTIVE_NODE,active_node);
          *y3 = node_no;
          break;

          case 33:
          active_node = active_node - 1;
          ssSetRWorkValue(S, ACTIVE_NODE,active_node);
          *y4 = node_no;
          break;
      }
    }
}
else if(RFM_LISR == 4)
{
  if(task_ID == 3)
  {
    *y2 = RFM_SID3;
    ssSetRWorkValue(S, TASK_ID,1);
    RFM_NTN = 0x6;
    RFM_NIC = 0x1;
  }
}
else
{
   if(task_ID == 2)
   {
     if(active_node == 0)
     {
       *y2 = RFM_SID2;
       RFM_SID2 = 0x0;
       RFM_LISR = 0x0000;
       ssSetRWorkValue(S, TASK_ID,3);
       RFM_NTN = 0x6;
```

```
      RFM_NIC = 0x3;
    }
  }
}
}
}

#endif

}


#ifdef MATLAB_MEX_FILE  /* Is this file being compiled as a MEX-file?*/
#include "simulink.c"   /* Mex glue */
#else
#include "cg_sfun.h"    /* Code generation glue */
#endif
```

A.2 Reflective Memory Driver Source Code

A.2.1 Read block for reflective memory

```
/******Read block for Reflective Memory in Window xp *******/


#define         S_FUNCTION_LEVEL        2
#undef          S_FUNCTION_NAME
#define         S_FUNCTION_NAME         win5565read


#include "simstruc.h"

#include "mex.h"



#include <windows.h>
#include <stdio.h>
#include <string.h>
#include <windows.h>

#include "rfm2g_api.h"
```

```
/*************************Input Arguments ********************

partition(1).Internal.Address  -
Contains specified Offset Address, is set by
internally by the completepartitionstruct command

partition(1).Internal.NDwords  -
Contains the no of DWORDS to be read, Is set
Internally by the compltetepartitionstruct command

ts _ sampling time
pci - PCI slot - default would be 1,
will change if more than 1 reflective memory
cards are installed

errport - To log error information

*****************************************************************/


// Get the supplied parameters

#define NUMBER_OF_ARGS          (5)
//Offset Address
#define ADDRESS_ARG             ssGetSFcnParam(S, 0)
//No of DWORDS
#define NDWORDS_ARG            ssGetSFcnParam(S, 1)
#define SAMPLETIME_ARG          ssGetSFcnParam(S, 2)

//SLOT_ARG : 1 is the default value
/(Would change if 2 cards are there on same machine)
#define SLOT_ARG ssGetSFcnParam(S, 3)


//#define N_PAGES ssGetSFcnParam(S, 4)

#define ERROR_STATUS_ARG        ssGetSFcnParam(S, 4)



#define SAMP_TIME_IND           (0)
#define BASE_ADDR_IND           (0)

// IWorks storage
#define NO_I_WORKS              (2)
```

```
#define BASE_ADDR_I_MEMORY \qquad (0)
#define RFM_HANDLE \qquad (1)

#define NO_R_WORKS              (0)
#define NO_P_WORKS              (0)

#define DEVICE_PREFIX "\\\\.\\rfm2g"

static char_T msg[256];


static void mdlInitializeSizes(SimStruct *S)
{

int_T status_port;


 ssSetNumSFcnParams(S, NUMBER_OF_ARGS);
 if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
     sprintf(msg,"Wrong number of input arguments passed.\n"
               "%d arguments are expected\n",NUMBER_OF_ARGS);
     ssSetErrorStatus(S,msg);
     return;
}

ssSetNumContStates(S, 0);
ssSetNumDiscStates(S, 0);


ssSetNumInputPorts(S, 0);



 // Grow if necessary
 status_port = (int_T)*mxGetPr(ERROR_STATUS_ARG);
 if(status_port) {
    if (!ssSetNumOutputPorts(S,2)) return;
     ssSetOutputPortWidth(S, 1, 1);
     ssSetOutputPortDataType(S, 1, SS_UINT32);
     ssSetNumSampleTimes(S, 1);
 }
 else {
    ssSetNumOutputPorts(S, 1);
 }
```

```
  // set type of data port (required)
  ssSetOutputPortWidth(S, 0, (int_T)mxGetPr(NDWORDS_ARG)[0]);
  ssSetOutputPortDataType(S, 0, SS_UINT32);
  ssSetNumSampleTimes(S, 1);

  ssSetNumRWork(S, NO_R_WORKS);
  ssSetNumIWork(S, NO_I_WORKS);
  ssSetNumPWork(S, NO_P_WORKS);

  ssSetNumModes(S, 0);
  ssSetNumNonsampledZCs(S, 0);

  ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE|SS_OPTION_PLACE_ASAP);

}


static void mdlInitializeSampleTimes(SimStruct *S)
{
 if (mxGetPr(SAMPLETIME_ARG)[0]==-1.0) {
     ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
     ssSetOffsetTime(S, 0, FIXED_IN_MINOR_STEP_OFFSET);
 } else {
     ssSetSampleTime(S, 0, mxGetPr(SAMPLETIME_ARG)[0]);
     ssSetOffsetTime(S, 0, 0.0);
 }
}


#define MDL_START
static void mdlStart(SimStruct *S)
{

 RFM2G_STATUS result;\qquad /* To hold return values from API calls */
/* the following code will be replaced by the dynamic buffer
   to accomodate the packed variables */

RFM2GHANDLE Handle=0;
RFM2G_INT8 device_name[40];

volatile unsigned long *ioaddress32=NULL;

/* The following code will be an argument in the s-funtion */

//  uint32_T mem_size;
```

```
sprintf(device_name,"%s%d",DEVICE_PREFIX,mxGetN(SLOT_ARG));


result=RFM2gOpen(device_name,&Handle);

if(result!=RFM2G_SUCCESS)
 {
 RFM2gErrorMsg(result);//  - Return error status with text
 printf("\nError: RFM2gOpen() failed.\n");
 printf("Error: %s\n",RFM2gErrorMsg(result));
 return(-1);
}

 /*Get the first valid Offset Value */

result=
  RFM2gUserMemory(Handle,(unsigned long**)&ioaddress32,0x00,100000);

/* The Last Parameter(pages) would be an user input to the block */
/* Details available from device specific Manual*/

 if(result!=RFM2G_SUCCESS)
 {
 //RFM2gErrorMsg()  - Return error status with text
 printf("\nError: RFM2gUserMemory() failed.\n");
  printf("Error: %s\n",RFM2gErrorMsg(result));
  return(-1);
 }

ssSetIWorkValue(S, RFM_HANDLE, (uint_T)Handle);
ssSetIWorkValue(S, BASE_ADDR_I_MEMORY, (uint_T)ioaddress32);

}


static void mdlOutputs(SimStruct *S, int_T tid)
{

uint_T base = ssGetIWorkValue(S, BASE_ADDR_I_MEMORY);
uint_T Handle = ssGetIWorkValue(S, RFM_HANDLE);

volatile unsigned long *ioaddress32 =(void *) base;
uint32_T  *y =  (uint32_T *)ssGetOutputPortSignal(S,0);
```

```
uint32_T  i;
uint32_T no_words;
uint32_T n_offset;

no_words= (uint32_T)mxGetPr(NDWORDS_ARG)[0];
for(i=0;i<no_words;i++) {
       y[i]=ioaddress32[((uint32_T)mxGetPr(ADDRESS_ARG)[0]/4)+i];
printf("\nValue read from location %u is:%f\n"
                         ,ioaddress32,ioaddress32[5376/4];
    }

if( ssGetNumOutputPorts(S) > 1) {
          y= (uint32_T *)ssGetOutputPortSignal(S,1);
       }
}

static void mdlTerminate(SimStruct *S)
{

uint_T Handle = ssGetIWorkValue(S, RFM_HANDLE);

RFM2gClose( &Handle );
}


#ifdef MATLAB_MEX_FILE /* Is this file being compiled as a MEX-file? */
#include "simulink.c"  /* Mex glue */
#endif
```

A.2.2 Write block for Reflective Memory

```
/*******Write block for Reflective Memory in Window xp *********/


#define         S_FUNCTION_LEVEL        2
#undef          S_FUNCTION_NAME
#define         S_FUNCTION_NAME         win5565write


#include        "simstruc.h"

#include        "mex.h"


#include        <windows.h>
```

```
#include <stdio.h>
#include <string.h>
#include <windows.h>

#include "rfm2g_api.h"



/******************Input Arguments ********************

partition(1).Internal.Address  -
Contains specified Offset Address, is set by
internally by the completepartitionstruct command

partition(1).Internal.NDwords  -
Contains the no of DWORDS to be read, Is set
Internally by the compltetepartitionstruct command

ts _ sampling time
pci - PCI slot - default would be 1,
will change if more than 1 reflective memory
cards are installed

errport - To log error information



// Get the supplied parameters

#define NUMBER_OF_ARGS          (5)
//Offset Address
#define ADDRESS_ARG             ssGetSFcnParam(S, 0)
//No of DWORDS
#define NDWORDS_ARG             ssGetSFcnParam(S, 1)
#define SAMPLETIME_ARG          ssGetSFcnParam(S, 2)

//SLOT_ARG : 1 is the default value
(Would change if 2 cards are there on same machine)
#define SLOT_ARG ssGetSFcnParam(S, 3)

// default is 100 pages, according the requirement of memory
and specific device, the page size could be changed

//#define N_PAGES ssGetSFcnParam(S, 4)
```

```
#define ERROR_STATUS_ARG   ssGetSFcnParam(S, 4)



#define SAMP_TIME_IND           (0)
#define BASE_ADDR_IND           (0)

// IWorks storage
#define NO_I_WORKS              (2)
#define BASE_ADDR_I_MEMORY      (0)
#define RFM_HANDLE              (1)

#define NO_R_WORKS              (0)
#define NO_P_WORKS              (0)

#define DEVICE_PREFIX "\\\\.\\rfm2g"

static char_T msg[256];


static void mdlInitializeSizes(SimStruct *S)
{

int_T status_port;


ssSetNumSFcnParams(S, NUMBER_OF_ARGS);
if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
sprintf(msg,"Wrong number of input arguments passed.\n"
       "%d arguments are expected\n",NUMBER_OF_ARGS);
ssSetErrorStatus(S,msg);
return;
}

    ssSetNumContStates(S, 0);
    ssSetNumDiscStates(S, 0);

    // Grow if necessary
    status_port = (int_T)*mxGetPr(ERROR_STATUS_ARG);
    if(status_port) {
        if (!ssSetNumOutputPorts(S,1)) return;
        ssSetOutputPortWidth(S, 0, 1);
        ssSetOutputPortDataType(S, 0, SS_UINT32);
        ssSetNumSampleTimes(S, 1);
    }
```

```
    else {
        ssSetNumOutputPorts(S, 0);
    }

 // set type of data port (required)
    ssSetNumInputPorts(S, 1);
    ssSetInputPortWidth(S, 0, (int_T)mxGetPr(NDWORDS_ARG)[0]);
    ssSetInputPortDataType(S, 0, SS_UINT32);
    ssSetInputPortDirectFeedThrough(S, 0, 1);
    ssSetInputPortRequiredContiguous(S, 0, 1);


    ssSetNumSampleTimes(S, 1);


    ssSetNumRWork(S, NO_R_WORKS);
    ssSetNumIWork(S, NO_I_WORKS);
    ssSetNumPWork(S, NO_P_WORKS);

    ssSetNumModes(S, 0);
    ssSetNumNonsampledZCs(S, 0);

    ssSetOptions(S,SS_OPTION_EXCEPTION_FREE_CODE|SS_OPTION_PLACE_ASAP);


}


static void mdlInitializeSampleTimes(SimStruct *S)
{
    if (mxGetPr(SAMPLETIME_ARG)[0]==-1.0) {
        ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
        ssSetOffsetTime(S, 0, FIXED_IN_MINOR_STEP_OFFSET);
    } else {
        ssSetSampleTime(S, 0, mxGetPr(SAMPLETIME_ARG)[0]);
        ssSetOffsetTime(S, 0, 0.0);
    }
}


#define MDL_START
static void mdlStart(SimStruct *S)
{
```

```
 RFM2G_STATUS result; //To hold return values from API calls */
 /* the following code will be replaced by the dynamic buffer
 to accomodate the packed variables */

 RFM2GHANDLE Handle=0;
 RFM2G_INT8 device_name[40];

 volatile unsigned long *ioaddress32=NULL;

 /* The following code will be an argument in the s-funtion */

 sprintf(device_name,"%s%d",DEVICE_PREFIX,mxGetN(SLOT_ARG));


 result=RFM2gOpen(device_name,&Handle);

 if(result!=RFM2G_SUCCESS)
 {
 //RFM2gErrorMsg()  - Return error status with text
 printf("\nError: RFM2gOpen() failed.\n");
 printf("Error: %s\n",RFM2gErrorMsg(result));
 return(-1);
 }

 /*Get the first valid Offset Value */

result=RFM2gUserMemory(Handle,(unsigned long**)&ioaddress32,0x00,100);
/* The Last Parameter(pages) would be an user input to the block */
/* Details available from device specific Manual*/

 if(result!=RFM2G_SUCCESS)
 {
 //RFM2gErrorMsg()  - Return error status with text
 printf("\nError: RFM2gUserMemory() failed.\n");
 printf("Error: %s\n",RFM2gErrorMsg(result));
 return(-1);
 }

ssSetIWorkValue(S, RFM_HANDLE, (uint_T)Handle);
ssSetIWorkValue(S, BASE_ADDR_I_MEMORY, (uint_T)ioaddress32);

}


static void mdlOutputs(SimStruct *S, int_T tid)
```

```
{

uint_T base = ssGetIWorkValue(S, BASE_ADDR_I_MEMORY);
uint_T Handle = ssGetIWorkValue(S, RFM_HANDLE);


volatile unsigned long *ioaddress32 =(void *) base;
uint32_T  *u =  (uint32_T *)ssGetInputPortSignal(S,0);
int32_T  i;
uint32_T no_words;
uint32_T n_offset;

no_words= (uint32_T)mxGetPr(NDWORDS_ARG)[0];

for(i=0;i<no_words;i++) {
ioaddress32[((uint32_T)mxGetPr(ADDRESS_ARG)[0]/4)+i]=u[i];
printf("\nValue read from location %u is:
                  %f\n",ioaddress32,ioaddress32[5376/4]);
}

if( ssGetNumOutputPorts(S) > 0) {
   u= (uint32_T *)ssGetOutputPortSignal(S,0);
}

}
static void mdlTerminate(SimStruct *S)
{

uint_T Handle = ssGetIWorkValue(S, RFM_HANDLE);

RFM2gClose( &Handle );
}
#ifdef MATLAB_MEX_FILE  /* Is this file being compiled as a MEX-file?*/
#include "simulink.c"   /* Mex glue */
#endif
```

References

Anwar, S., & Ashrafi, B. 2002. A Predictive Control Algorithm for an Anti-Lock Braking System. SAE Technical Paper 2002-01-0302.

Ayoubi, M., Demmeler, T., Leffler, H., & Kohn, P. 2004. X-by-Wire Functionality, Performance and Infrastructure. SAE Technical Paper 2004-21-0043.

Bacic, M. 2005. On Hardware-in-the-Loop Simulation. Proceedings of the 44th IEEE Conferences on Decision and Control , and European Control Conference.

Bannatyne, R.T. 1998. Advances and Challenges in Electronic Braking Control Technology. SAE Technical Paper 982244.

Basset, M., Aimmer, C., & Gissinger, G.L. 1997. Fuzzy Approach to the Real Time Longitudinal Velocity Estimation of a FWD Car in Critical Situations. *Vehicle System Dynamics*, **27**, 477–489.

Bosch. 1999. *Driving-Safety Systems.* Robert Bosch GmbH.

Daiss, A., & Kiencke, U. 1995. Estimation of Vehicle Speed : Fuzzy-Estimation in Comparison with Kalman Filtering. Proceedings of the IEEE Conference on Control Applications.

Drakunov, S., Ozguner, U., Dix, P., & Ashrafi, B. 1995. ABS Control Using Optimum Search Via Sliding Modes. *IEEE Trans. Control Systems Technology 3(1),79-85.*

Emereole, Okwuchi Chigoziri. 2004. *Antilock Performance Comparison Between Hydraulic and Electromechanical Brake Systems.* Ph.D. thesis, University of Melbourne.

Fujimoto, R.M. 1990. Optimistic Approaches to Parallel Discrete Event Simulation. *SCS Trans*, **7**, 153–191.

Garcia, C. E., Prett, D. M., & Morari, M. 1989. Model Predictive Control: Theory and Practice-a Survey. *Automatica*, **25**(3), 335–348.

Gawthrop, P.J., Demircioglu, H., & Siller-Alcala, I. 1998. Multivariable Continuous Time Generalised Predictive Control : A State Space Approach to Linear and Nonlinear Systems. *Proc. IEE Pt.D: Control Theory and Applications.*

Gustafsson, F. 1997. Slip Based Tire Road Friction Estimation. *Automatica*, **33**, 1087–99.

Hadri, A. El, Cadiou, J., M'sirdi, K., & Delanne, Y. 2001. Wheel Slip Regulation Based on Sliding Mode Approach. SAE Technical Paper.

Hedenetz, B., & Belschner, R. 1998. Brake-by-Wire Without Mechanical Backup by Using a TTP-Communication Network. SAE Technical paper 981109.

Jonner, W.-D, Winner, H., Dreilich, L., & Schunck, E. 1996. Electrohydraulic Brake System - The First Approach to Brake-by-Wire Technology. SAE Technical Paper 960991.

Kelling, N., & Leteinturier, P. 2003. X-by-Wire : Opportunities, Challenges and Trends. SAE Technical Paper 2003-01-0113.

Kohl, S., & Jegminat, D. 2005. How to Do Hardware-in-the-Loop Simulation Right. SAE Technical Paper 2005-01-1657.

Kopetz, H., & Grunsteidl, G. 1994. TTP - A Protocol for Fault-Tolerant Real-Time Systems. *IEEE Computer (27(1)), 14-23.*

Kouvaritakis, B., Cannon, M., & Rossiter, J. A. 1999. Recent Developments in Generalized Predictive Control for Continuous-Time Systems. *International Journal of Control*, **72**(2), 164–173.

Line, C, Manzie, C, & Good, M. 2004. Control of an Electromechanical Brake for Automotive Brake-by-Wire with an Adapted Motion Control Architecture. SAE.

Maciejowski, J. M. 2002. *Predictive control : with constraints.* Prentice Hall.

Maron, C., Dieckmann, T., Hauck, S., & Prinzler, H. 1997. Electromechanical Brake System: Actuator Control Development System. SAE Technical Paper 970814.

Mayne, D. Q., Rawlings, J. B., Rao, C. V., & Scokaert, P. O. M. 2000. Constrained Model Predictive Control: Stability and Optimality. *Automatica*, **36**, 789–814.

Morari, M., & Lee, J. H. 1999. Model Predictive Control: Past, Present and Future. *Computers and Chemical Engineering*, **23**, 667–682.

Nabi, S., Balike, M., Allen, J., & Rzemien, K. 2004. An Overview of Hardware-in-the-Loop Testing System at Visteon. SAE Technical Paper 2004-01-1240.

Petersen, I. 2003. *Wheel Slip Control in ABS Brakes Using Gain Scheduled Optimal Control with Constraints.* Ph.D. thesis, Norwegian university of science and technology.

Ploger, M., Sauer, J., Budenbender, M., Held, J., Costanzo, F., Manes, M. De, Mare, G. Di, Ferrara, F., & Montieri, A. 2004. Testing Networked ECUs in a Virtual Car Environment. SAE Technical Paper 2004-01-1724.

Pollini, L., & Innocenti, M. 2000. A Synthetic Environment for Dynamic Systems Control and Distributed Simulation. *IEEE Control Systems Magazine*, 49–61.

Rao, C. V., Wright, S. J., & Rawlings, J. B. 1998. Application of Interior-Point Methods to Model Predictive Control. *Journal of Optimization Theory and Applications*, **99**, 723–757.

Rawlings, J. B. 2000. Tutorial Overview of Model Predictive Control. *IEEE Control Systems Magazine*, **20**(Jun), 38–52.

Rossiter, J. A. 2003. *Model-Based Predictive Control: A Practical Approach.* CRC Press.

Schwarz, R., Isermann, R., Bohm, J., Nell, J., & Reith, P. 1990. Clamping Force Estimation for a Brake-by-Wire Actuator. SAE Technical paper 1999-01-0482.

Schwarz, R., Isermann, R., Bohm, J., Nell, J., & Rieth, P. 1998. Modelling and Control of an Electromechanical Disk Brake. SAE Technical Paper 980600.

Semmler, S., Isermann, R., Schwarz, R., & Rieth, P. 2003. Wheel Slip Control for Antilock Braking Systems Using Brake-by-Wire Actuators. SAE Technical Paper 2003-01-0325.

Solyom, S. 2004. *Control of Systems with Limited Capacity.* Ph.D. thesis, Lund Institue of Technology.

Stasko, J.C., Crandell, R.L., Dunn, M.T., Sureshbabu, N., & Weber, W. 1998. The Versatile Hardware-in-the-Loop Laboratory : Beyond the Ad Hoc Fixture. Proceedings of the American Control Conference.

Unsal, C., & Kachroo, P. 1999. Sliding Mode Measurement Feedback Control for Antilock Braking Systems. *IEEE Trans. Control Systems Technology*, **7**.

Wang, L. 2001. Continuous Time Model Predictive Control Design using Orthonormal Functions. *International Journal of Control*, **74**(16), 1588–1600.

Wang, L. 2002 (Sep). A Tutorial on Model Predictive Control. *Pages 1394–1399 of: The 4th Asian Control Conference.*

Wright, S. J. 1997. Applying New Optimization Algorithms to Model Predictive Control. *Chemical Process Control-V, CACHE, AIChE Symposium Series*, **93**(316), 147–155.