DEVELOPMENT OF A VEHICLE DYNAMICS CONTROLLER FOR OBSTACLE AVOIDANCE

Geraint Paul Bevan

A thesis for the degree of Doctor of Philosophy (PhD)

Submitted to the Faculty of Engineering,
University of Glasgow

January 2008

Abstract

As roads become busier and automotive technology improves, there is considerable potential for driver assistance systems to improve the safety of road users. Longitudinal collision warning and collision avoidance systems are starting to appear on production cars to assist drivers when required to stop in an emergency. Many luxury cars are also equipped with stability augmentation systems that prevent the car from spinning out of control during aggressive lateral manoeuvres. Combining these concepts, there is a natural progression to systems that could assist in aiding or performing lateral collision avoidance manoeuvres.

A successful automatic lateral collision avoidance system would require convergent development of many fields of technology, from sensors and instrumentation to aid environmental awareness through to improvements in driver vehicle interfaces so that a degree of control can be smoothly and safely transferred between the driver and vehicle computer. A fundamental requirement of any collision avoidance system is determination of a feasible path that avoids obstacles and a means of causing the vehicle to follow that trajectory.

This research focuses on feasible trajectory generation and development of an automatic obstacle avoidance controller that integrates steering and braking action.

A controller is developed to cause a specially modified car (a Mercedes ‘S’ class with steer-by-wire and brake-by-wire capability) to perform an ISO 3888-2 emergency obstacle avoidance manoeuvre.

A nonlinear two-track vehicle model is developed and used to derive optimal controller parameters using a series of simulations. Feedforward and feedback control is used to track a feasible reference trajectory. The feedforward control loops use inverse models of the vehicle dynamics. The feedback control loops are implemented as linear proportional controllers with a force allocation matrix used to apportion braking effort between redundant actuators.

Two trajectory generation routines are developed: a geometric method, for steering a vehicle at its physical limits; and an optimal method, which integrates steering and braking action to make full use of available traction. The optimal trajectory is obtained using a multi-stage convex optimisation procedure.

The overall controller performance is validated by simulation using a complex proprietary model of the vehicle that is reported to have been validated and calibrated against experimental data over several years of use in an industrial environment.

Acknowledgements

Vielen Dank an mein Doktorvater **Henrik Gollee**! Your guidance has been exquisite. Many thanks also to **Dominic Diston** and **Peter Gawthrop** who have taught me so much about modelling and simulation.

My colleagues have been a joy to work with throughout this research project, particularly **Simon O'Neill**, with whom I had the great pleasure of sharing an office for several years, and **John O'Reilly**, whose story-telling is routinely interspersed with gems of wisdom. I must also give particular thanks to **Jens Kalkkuhl** and **Avshalom Suissa** of Daimler who have given their time freely and offered invaluable insight into the field of automotive engineering and the more specific problem of collision avoidance.

The Centre for Rehabilitation Engineering might be considered an unlikely setting for research into automotive control, yet it seems to work. Thanks, of course, to **Ken Hunt** for all his efforts to provide a stimulating research environment. I wish to pay tribute to my friends and colleagues who have passed, or are passing, through the CRE and who have made the last few years such an enjoyable experience. I must also thank the secretarial and IT support staff who have been so helpful, particularly **Elaine McNamara** and **Kenny Stevenson**.

Finally, I would like to thank the examiners who gave this thesis a thorough examination, **Warren Manning** and **Euan McGookin**.

This research was supported by CEMACS – Complex Embedded Automotive Control Systems – a specific targeted research project funded through the European Union's Sixth Framework Programme, under Contract 004175.

Nomenclature

Symbols used within this thesis are shown below. The following typefaces are used: $scalar$; $\mathbf{vector}$; $\underline{\mathrm{matrix}}$; and $\mathrm{function}$. Calligraphic script is used to denote a grid $\mathcal{G}$ (vector of vectors).

Symbol	Type	Description	Units
$\underline{\mathrm{A}}$	$\in \mathbb{R}^{3\times3}$	Linearised state matrix	
$\underline{\mathrm{B}}_{\delta}$	$\in \mathbb{R}^{3\times1}$	Linearised input matrix	
$\underline{\mathrm{B}}_{f}$	$\in \mathbb{R}^{3\times4}$	Linearised input matrix	
$\underline{\mathrm{B}}_{f}^{\dagger}$	$\in \mathbb{R}^{4\times3}$	Pseudo-inverted input matrix	
B	$\in \mathbb{R}$	Magic formula tyre model coefficient	-
C	$\in \mathbb{R}$	Magic formula tyre model coefficient	-
C_{α}	$\in \mathbb{R}$	Tyre cornering stiffness	N/rad
D	$\in \mathbb{R}$	Magic formula tyre model coefficient	-
D_{Y}	$\in \mathbb{R}$	Aerodynamic drag force	N
D_{ψ}	$\in \mathbb{R}$	Aerodynamic drag moment	N m
E	$\in \mathbb{R}$	Magic formula tyre model coefficient	-
F	$\in \mathbb{R}$	Tyre force (body axis)	N
$\mathcal{G}$	$\in \mathbb{R}^{6\times(L+1)}$	Direct transcription grid	-
$\mathbf{G}$	$\in \mathbb{R}^{6}$	Velocity and position vector	m/s, rad/s, m, rad
$\underline{\mathrm{I}}$	$\in \mathbb{R}^{n\times n}$	Identity matrix	-
J	$\in \mathbb{R}$	Optimal trajectory cost function	rad/s^2
J_{ZZ}	$\in \mathbb{R}$	Vehicle moment of inertia	kg m^2
$\underline{\mathrm{K}}_{\delta}$	$\in \mathbb{R}$	Controller gain	-
$K_{\delta,Y^{\oplus}}$	$\in \mathbb{R}$	Controller gain	-
$K_{\delta,\psi}$	$\in \mathbb{R}$	Controller gain	-
$K_{\delta,\dot{\psi}}$	$\in \mathbb{R}$	Controller gain	-
$\underline{\mathrm{K}}_{f}$	$\in \mathbb{R}$	Controller gain	-
$K_{f,\dot{Y}}$	$\in \mathbb{R}$	Controller gain	-
$K_{f,\dot{\psi}}$	$\in \mathbb{R}$	Controller gain	-
L	$\in \mathbb{N}$	Number of grid points	-

$\underline{\mathrm{M}}$		Generic matrix	-
M	$\in \mathbb{R}$	Moment about centre of mass	N m
O	$\in \mathbb{R}^2$	Centre of circle	(m,m)
P	$\in \mathbb{R}^2$	Waypoint position	(m,m)
R	$\in \mathbb{R}$	Radius of curvature	m
T	$\in \mathbb{R}$	Period	s
$W^{\oplus}$	$\in \mathbb{R}^4$	Lateral wheel position relative to CG	m
$X^{\oplus}$	$\in \mathbb{R}$	Longitudinal position (fixed Earth axis)	m
$Y^{\oplus}$	$\in \mathbb{R}$	Lateral position (fixed Earth axis)	m
$Z^{\oplus}$	$\in \mathbb{R}$	Vertical position (fixed Earth axis)	m
X	$\in \mathbb{R}$	Forward position (body axis)	m
Y	$\in \mathbb{R}$	Lateral position (body axis)	m
Z	$\in \mathbb{R}$	Vertical position (body axis)	m
$\mathbf{b}^{\oplus}$	$\in \mathbb{R}^{L+1}$	Boundary position	m
$\mathbf{e}_f$	$\in \mathbb{R}$	Brake loop error signal	m/s, m/s, rad/s
f	$\in \mathbb{R}$	Tyre force	N
$\mathbf{f}$	$\in \mathbb{R}^{4\times1}$	Brake force vector	N
g	$\in \mathbb{R}$	Gravitational acceleration	m/s^2
h	$\in \mathbb{R}^3$	Non-linear plant model	
l	$\in \mathbb{R}$	Moment arm (from centre of mass to wheel)	m
m	$\in \mathbb{R}$	Vehicle mass	kg
n	$\in \mathbb{N}$	Vector size	-
r	$\in \mathbb{R}$	Tyre radius	m
t	$\in \mathbb{R}$	Time	s
$\mathbf{u}_f$	$\in \mathbb{R}^3$	Brake loop control signal	-
u	$\in \mathbb{R}$	Forward speed	m/s
v	$\in \mathbb{R}$	Wheel speed	m/s
x	$\in \mathbb{R}$	Longitudinal position (wheel axis)	m
y	$\in \mathbb{R}$	Lateral position (wheel axis)	m
z	$\in \mathbb{R}$	Vertical position (wheel axis)	m
$\mathbf{v}$	$\in \mathbb{R}^{4\times1}$	Velocity vector	m/s, rad/s
$\mathbf{w}$	$\in \mathbb{R}^{4\times1}$	Acceleration vector	m/s^2, rad/s^2
Δ	$\in \mathbb{R}$	Grid spacing	m
Γ	$\in \mathbb{R}^{3\times3}$	Rotation matrix	
$\mathbf{\Sigma}$	$\in \mathbb{R}^n$	Singular values	

α	$\in \mathbb{R}$	Wheel slip angle	rad
δ	$\in \mathbb{R}$	Steering angle	rad
η	$\in \mathbb{R}$	Construction angle	rad
κ	$\in \mathbb{R}$	Wheel slip (generic)	-
λ	$\in \mathbb{R}$	Wheel slip ratio	-
μ	$\in \mathbb{R}$	Friction coefficient	-
ν	$\in \mathbb{R}$	Noise parameter	-
ω	$\in \mathbb{R}$	Wheel speed (angular)	rad/s
ϕ	$\in \mathbb{R}$	Trajectory construction angle	rad
$\boldsymbol{\rho}$	$\in \mathbb{R}^{4\times 1}$	Scheduling vector	m/s, rad
θ	$\in \mathbb{R}$	Construction angle	rad
σ	$\in \mathbb{R}$	Singular value	-
$\bar{\sigma}$	$\in \mathbb{R}$	Standard deviation	-
τ	$\in \mathbb{R}$	Singular value tolerance	-
ξ	$\in \mathbb{R}$	Manoeuvre constant	-
χ	$\in \mathbb{R}$	Generic parameter	-
ψ	$\in \mathbb{R}$	Yaw angle	rad
$\dot{\psi}$	$\in \mathbb{R}$	Yaw rate	rad/s

Subscript modifiers

Symbol	**type**	**Description**
0	-	Initial
I	-	First optimisation pass
II	-	Second optimisation pass
III	-	Third optimisation pass
ff	-	Feed-forward
i	$\in \{1, 2, 3, 4\}$	Wheel number
j	$\in \{0, \ldots, L\}$	Grid point
l	-	Lower
r	-	Reference
u	-	Upper
x	-	Longitudinal
y	-	Lateral
z	-	Vertical

Contents

List of Tables

List of Figures

Figure 1 has been removed due to Copyright restrictions.

The illustration appeared in New Scientist Magazine Issue 2611, 4^{th} July 2007:
"The programmable robot of ancient Greece" by Noel Sharkey.

The image is available online at

http://technology.newscientist.com/data/images/archive/2611/26111601.jpg

Figure 1: Two thousand years ago, Hero of Alexandria designed a steer-by-wire robot capable of performing a double lane-change manoeuvre using feed-forward control (Sharkey 2007).

Chapter 1

Introduction

The programmable self-propelled machine might even go back as far as the 8th century BC.

— Noel Sharkey [2007]

1.1 Background

Two thousand years ago, Hero of Alexandria designed a steer-by-wire robot capable of performing a double lane change manoeuvre using feed-forward control (Sharkey (2007), see Figure 1). At that time, the fastest vehicles on the roads would have been chariots, with the horses controlled directly by skilled handlers. The most powerful controllable forces that could be exerted for propulsion in those ancient times were derived from muscle power or the wind. In his Pneumatics (c. 300 CE, translated by Woodcroft (1851)), Hero of Alexandria described many inventions, including an early steam engine (aeolipile) which could be used to rotate a pivoted ball above a cauldron. Although an amusing toy, the engine does not feature a control system and there is no indication that it was ever used to produce useful work.

By the time of the Renaissance, more than a millennium after Hero, there had been little progress towards useful self-propelled vehicles. Compressionless engines were described by the likes of Leonardo da Vinci and Christian Huygens, but these were not practical designs. Leonardo da Vinci designed a clockwork tricycle, (Codex Atlanticus (c. 1519 CE), folio 812 recto), a modern reconstruction of which is exhibited at the Museum of Science History in Florence. The clockwork mechanism stores small amounts of energy and further springs provide steering action, but the design was not a challenge to traditional transportation methods.

A couple of centuries later, the industrial revolution brought about the exercise of controllable

power far greater than had been seen previously. Steam engines could power factories or locomotives, while ingenious devices such as the Watt governor (which had previously been used in windmills) provided automatic control. With the advent of the modern internal combustion engine in the 19^{th} century, came the appearance of road vehicles that were more powerful than the horses and carriages that had preceded them. Along with the enhanced mobility that cars could provide, came new dangers for other road users.

In Britain, early attempts to reduce road fatalities included the legal requirement for a man with a red flag to walk in front of a motor vehicle. New infrastructure, such as traffic lights and roundabouts, were introduced to prevent collisions. The last century has seen rapid advances in automotive technology. Modern cars, equipped with efficient engines, are able to travel at very high speeds while offering unprecedented levels of protection to occupants. Improvements in tyre technology, coupled with anti-lock braking systems (ABS), have helped drivers to routinely make good use of available traction when stopping a vehicle.

While passengers have become comfortable in recent years with the concept of fly-by-wire aircraft being flown by autopilots, most people would be uncomfortable with the notion of cars exhibiting such a degree of autonomy. Instinctively, we tend to trust people more than machines. In the case of cars which are not subject to the rigorous maintenance inspections that are seen in the aerospace industry, this is an entirely rational perspective. Nevertheless, there is evidence to suggest that the performance of modern cars has outstripped the ability of average drivers to control them.

Analysing the contributory factors to a quarter of road accidents (those with a Contributory Factor record) in Great Britain from 1999 to 2002, Mosedale, Purdy & Clarkson (2004) found that the most frequently recorded factor was *failed to avoid vehicle or object in carriageway* (28% of all accidents) followed by *loss of control of vehicle* (19%). Meanwhile, the main precipitating factor in fatal and serious accidents was *loss of control of vehicle*, accounting for 43% of fatalities and 29% of serious accidents. The inability of drivers to adequately control road vehicles motivates the investigation of technologies that may intervene to improve safety.

1.2 Motivation

As roads become busier and automotive technology improves, there is considerable potential for driver assistance systems to improve the safety of road users. It is becoming increasingly common for luxury cars to be fitted with longitudinal collision avoidance systems, in which cruise control functions are integrated with forward looking obstacle detection sensors to assist deceleration of the car when necessary. Such devices are a valuable aid if an impending rear-end collision between cars travelling in the same lane is due to driver inattention and the vehicles are separated sufficiently in space and time for the aft vehicle to brake. However, longitudinal collision avoidance systems are of limited benefit for preventing head-on collisions or avoiding obstacles which appear suddenly in front of a moving vehicle. In these circumstances, aggressive lateral manoeuvres are more appropriate; as well as altering the path of the vehicle to move it out of danger, the manoeuvre can be completed in a

shorter distance than that required to stop the vehicle.

Generally, drivers would not appreciate an automatic collision avoidance system that restricts their options in an emergency. Nor is it likely that drivers would be comfortable with a computer taking control of the vehicle unnecessarily. An automatic collision avoidance system must therefore give the driver every opportunity to take whatever evasive action they deem appropriate until the very last moment at which the obstacle can still be avoided. Consequently, such systems would be expected to operate the vehicle at its physical limits: far from equilibrium conditions and in parts of the dynamic envelope at which actuators, such as tyres, exhibit highly nonlinear behaviour.

This research was undertaken as part of the CEMACS project: Complex Embedded Automotive Control Systems (`http://www.hamilton.ie/cemacs`). CEMACS is a specific targeted research programme of the European Union's Framework 6, bringing together researchers from DaimlerChrysler Research and Technology (Germany), SINTEF (Norway), Lund Institute of Technology (Sweden), Hamilton Institute (Ireland) and University of Glasgow (Scotland). The overall project aims are to develop active safety technology for road vehicles while researching appropriate control design and analysis techniques.

The CEMACS project comprises six work packages: (1) active safety: (i) rollover prevention; and (ii) collision avoidance; (2) integrated chassis control; (3) control design: (i) classical multi-variable control analysis and design; (ii) hybrid control systems; (iii) multi-variable control systems with time delay; and (iv) non-linear and adaptive control; (4) vehicle state observation; (5) experimental; and (6) management and dissemination.

The research described in this thesis is particularly concerned with the development of an automatic obstacle avoidance controller for a passenger car and was undertaken under the auspices of Work Packages 1.2 (active safety: collision avoidance) and Work Package 3 (control design). It builds upon the existing state of the art in automotive control technology.

1.3 State of the art

This section reviews some of the key technologies that are applicable to automatic control of a vehicle for collision avoidance.

In November 2007, the US Defense Advanced Research Projects Agency hosts teams competing for prizes of up to $2 million in its third Grand Challenge: Urban Challenge (DARPA 2007). Following previous Grand Challenges that saw autonomous ground vehicles travelling through the desert, the Urban Challenge will require competing vehicles to navigate a sixty mile (100 [km]) course over a period of six hours in a mock urban environment. The vehicles must obey traffic laws while merging with traffic and avoiding moving obstacles. With teams spending approximately $1 million per vehicle, it is likely to be several years before much of the technology on display finds its way into production vehicles. Even so, with an anticipated average speed of 10 mph (16 [km/hr]), the vehicles should be operating comfortably within their physical limits. It is the sensors and decision-making that are likely to present the greatest challenges to contestants.

At higher speeds, Volkswagen have unveiled a self-driving Golf GTi that can navigate a test-track and accelerate independently to speeds of up to 150 [mph] (240 [km/hr]), as reported by the Daily Mail (Massey 2006). Radar and laser sensors in the grille are used to identify obstacles and a satellite navigation system is used to track its position. After learning the course, the car is then able to navigate it automatically.

More limited autonomy is already beginning to appear on high-end production vehicles. Electronic control of brake systems, in the form of anti-lock brakes and, more recently, traction control systems, over-rides the driver's pedal inputs to maintain controllability of the car. Meanwhile, forward-looking sensors are being combined with automatic cruise control functions to either warn the driver, or to actually assist in the task of deceleration, when a collision is imminent. Each of these systems tackles a different aspect of the same essential problem. A vehicle travelling at speed has significant momentum which, because of traction limits, cannot always be adjusted as quickly as a driver would desire.

1.3.1 Tyre models

Tyres have a significant influence on vehicle dynamics. An extensive investigation of tyre dynamics was undertaken by Sakai and appeared in four parts (Sakai 1981*a*,*b*,*c*, 1982). More recently, several tyre models have been published; most notably the dynamic model of Canudas de Wit & Tsiotras (1999), which is based on the physical LuGre friction model of Canudas de Wit, Olsson, Åström & Lischinsky (1995), and the steady-state empirical *Magic Formula* tyre model of Pacejka & Bakker (1993), which defines longitudinal or lateral tyre forces as a function of longitudinal or lateral slip parameters. Combined slip (longitudinal and lateral) can be accommodated in the Magic Formula by introducing additional terms (e.g. to represent hysteresis) into the slip parameters and altering the model coefficients. However, it is generally expensive to obtain the test data needed to generate the coefficients for combined slip because of the large number of experiments required. A method of obtaining the necessary coefficients from limited test data is given by Sharp & Bettella (2003).

Hansen, Murray-Smith & Johansen (2005) propose a further abstraction, using a Gaussian Process model to identify and capture the essence of a tyre's friction curve. Applying the method to design a robust wheel slip controller, they demonstrate that accurate friction curves can be generated from sparse training data. The authors comment that nonparametric Gaussian Process models provide not only mean predictions and uncertainty estimates, but also mean and uncertainty estimates of local linearisations to the curves.

Forces generated by the tyres are non-linear functions of vehicle and wheel speed. Several authors (Heinzl, Lugner & Plöchl 2002, Schinkel & Hunt 2002, Solyom & Rantzer 2003) have observed that the friction curve of a typical tyre may be approximated by modelling it as two linear segments, one representing a stable region, where the gradient of the force-slip curve is positive, and another representing an unstable region where the gradient is negative.

1.3.2 Brake-by-wire

Electronic control of the brakes is becoming increasingly common as manufacturers attempt to make maximum use of the traction available to the tyres in dangerous situations.

Anti-lock braking systems (ABS) have been available on vehicles for several decades and have become standard on production vehicles in recent years (Emig, Goebels & Schramm 1990). If too much pressure is applied to the brakes, usually by the driver through the pedal and hydraulic system, the wheels will decelerate so much that the tyres are forced to operate in the unstable region. This reduces the braking force between the tyre and road and risks the onset of wheel-lock. An ABS intervenes electronically to reduce the brake pressure intermittently, thus allowing the wheels to accelerate, bringing the tyre slip closer to the point of maximum traction.

The concept of ABS was developed by Bosch and the mechanical details of an ABS are published in Bosch's Automotive Handbook (Bosch 2000). However, as noted by Jiang & Gao (2001), Schuller, Brangs, Rothfuss, Lutz & Breit (2002) and Wu & Shih (2003), the precise proprietary algorithms used within these controllers are trade secrets and tuning is subject to a considerable degree of manual refinement by test engineers using prototype vehicles. Furthermore, satisfactory tuning of parameters is dependent on the test engineers having reasonable insight into the parameterisation of the controllers (Schuller et al. 2002).

Braking systems are now being extended to include traction control and overall vehicle stability systems (Austin & Morrey 2000) such as DaimlerChrysler's Electronic Stability Programme (ESP) and BMW's Dynamic Stability Control (DSC). Whereas an ABS reduces the brake pressure from that demanded by the driver, traction control and stability systems apply the brakes automatically, without any intervention from the driver. This is done to prevent the wheels from being accelerated into an unstable operating region if too much power is applied through the drive-train. Such systems thereby allow a vehicle to accelerate on low friction surfaces or maintain directional stability during tight cornering.

Braking actuators

Johansen, Kalkkuhl, Lüdemann & Petersen (2001) explain that the use of cheap and simple valves has traditionally been an important factor in the design and cost of automotive braking systems. Several researchers have however investigated the use of advanced actuators, in particular actuators that permit continuous application of control (rather than the discrete control of solenoid valves) and the effect that these can have on the implementation of anti-lock braking systems.

Austin & Morrey (2000) cite investigations into the use of a spring and motor brake system to provide smooth rather than pulsed response and the use of electro-magnetic valves to supplement the braking force applied by conventional friction brakes.

The benefits of servo-valves are mentioned by Chamaillard, Gissinger, Perronne & Renner (1994) who note the excellent delay and bandwidth characteristics of a proportional servo-valve controlled by a rotary electro-magnetic motor. The use of servo-valves is also endorsed by Kazemi & Zaviyeh

(2001) who report results from simulations using servo-valves and a dynamic surface control scheme in comparison with a conventional ABS. In particular, they report almost a 20% reduction in stopping distance on a slippery road surface and a factor of three reduction in the time taken for the system to reach a steady state after a reduction in friction.

Gissinger, Menard & Constans (2003) describe an "intelligent braking system" which is also reported to reduce stopping distances. This system makes use of a completely re-designed brake mechanism including full contact brakes and proportional hydraulic servo-valves. Results from testing of a prototype system demonstrate a reduction of a few percent in stopping distance compared to a conventional ABS. They report that investigations into the use of commercially available (and hence cheaper) electro-magnetic valves have proved promising and that studies into the development of an electric actuator are underway.

Emig et al. (1990) describe how the introduction of electric brakes would permit "drive-by-wire" to be introduced to vehicles, with the controller regulating the braking force under partial braking conditions as well as during the critical situations that are handled by ABS and traction control systems.

A more radical suggestion is the use of electro-rheostatic actuation presented by Choi, Bang, Cho & Lee (2002). Replacement of standard hydraulic fluid with an electro-rheostatic fluid, i.e. a fluid in which the viscosity may be altered by application of an electric field, permits the pressure within the wheel brake cylinder to be manipulated reversibly, instantaneously and in a continuous manner. The authors report an improvement in stability and steerability over conventional ABS and note that electro-rheostatic valves can be effective at preventing chatter in the control signal.

Jiang & Gao (2001) note that trucks commonly use pneumatic rather than hydraulic brake systems and that this adds complexity to the control problem (aside from the obvious differences due to modified vehicle dynamics in the presence of a trailer). These pneumatic systems tend to be slower in response than hydraulic systems and have a high degree of hysteresis.

Brake control strategies

Anti-lock braking is based on the principle that for a given set of running conditions, friction between the tyre and road is maximised for a certain level of wheel slip (Austin & Morrey 2000, Bosch 2000). As wheel slip is increased from zero to this maximum during braking, the retarding force generated on the wheel increases accordingly. Beyond the point of maximum friction, the retarding force tends to diminish as the wheel slip is increased, although this does depend on the type of tyre used and the road conditions. Momentary relaxation of the brake pressure when the peak of the friction curve has been passed allows the wheel to accelerate back towards the maximum friction point. The aim of ABS controllers then is to keep the tyres operating at or near the peak of the friction curve in order to use the available grip as effectively as possible.

Traction control systems operate when a vehicle is accelerating rather than braking, however they have a similar aim (Austin & Morrey 2000, Emig et al. 1990). Again the controller must try to

make best use of the available grip by managing the amount of wheel slip. This may be achieved either by changing the total amount of power delivered to the wheels by the engine or by altering the distribution of power between the wheels using selective application of brakes.

The most significant difficulties that must be overcome by controllers are the (generally uncertain) nonlinear characteristics of the friction versus wheel slip curve and uncertainty in plant parameters, particularly the peak friction coefficient between tyre and road and the amount of slip exhibited by each wheel (Germann, Würtenberger & Daiß 1994).

Given the close correlation between wheel slip and friction coefficient, many ABS controllers are designed on the premise that it is possible to keep the level of wheel slip near to the point that affords the greatest possible traction. Unfortunately, the precise degree of slip that will maximise traction is generally unknown as this is a function of both the type and condition and of the tyre and road surface. The road surface may of course change instantaneously. Another important difficulty that is faced when implementing slip control is that of measuring the actual slip that is present at any given time. Although the speed of each wheel may be known, the vehicle speed is generally not, and as noted by Yi, Alvarez, Claeys & Horowitz (2003), knowledge of wheel speeds is not sufficient for estimation of ground speed and slip as the system is almost unobservable.

A further complication to wheel slip control occurs when integrated into an overall vehicle dynamics controller. Solyom & Rantzer (2003) note that when cornering and braking occur simultaneously it is necessary that the wheels be allowed to turn at different speeds. This has implications for the target slip of each wheel.

Canudas de Wit & Tsiotras (1999) explain that extremum-seeking control strategies require *a priori* knowledge of the optimal target slip and this problem is not adequately dealt with in the current literature. Austin & Morrey (2000) opine that a fixed slip target is not appropriate because of the complexity of estimating the optimal slip in real-time under varying conditions. However, many researchers have chosen to select such a fixed target slip that is suitable for a particular analysis or is suitable for nominal conditions. Values in the range of 10% to 20% are common for conditions in which there is no lateral wheel slip α, i.e. $\alpha = 0$ (e.g. Jiang & Gao (2000): 20%, Kazemi & Zaviyeh (2001): 12.5%, Schinkel (2002): 10%). Kraft & Leffler (1990) note that there are instances in which it is desirable to allow greater wheel slip to occur, for instance when using snow chains.

Evaluation of the optimal slip ratio to achieve maximum braking or traction is a particularly complex topic. Gustafsson (1998) and Lee & Żak (2002) show that the optimum slip ratio can alter significantly between different road surfaces, such as dry asphalt and ice, and explain that a single slip ratio is not appropriate for all conditions. Nouillant, Assadian, Moreau & Oustaloup (2002) propose to use the acceleration at the start of braking (i.e. before activation of the ABS) as an approximation of the optimal acceleration available. The slip may then be controlled to maintain this near-optimal acceleration. However, as Gustafsson (1998) also notes, conditions may change instantaneously and unevenly and it is therefore necessary for any extremum-seeking controller to continuously monitor the changing conditions and respond accordingly. Lee & Żak (2002) propose using fuzzy logic (tuned with a genetic algorithm) which acts on acceleration data. Yi, Alvarez, Horowitz & de Wit (2000)

note that slip estimation becomes more difficult at low speed but that placing an upper limit on the value of slip ratio can eliminate problems that this would otherwise cause.

Brake torque control

Chamaillard et al. (1994) note that velocity estimation based on wheel speed sensors is inadequate to control slip ratio and describe classical ABS as merely an acceptable compromise. They therefore choose to control brake torque rather than the more usual wheel slip within an inner control loop.

The authors note that brake torque is very closely related to the grip between the tyre and the road. By placing strain gauges in a Wheatstone Bridge formation on the brake calliper support structure, and with the aid of some modelling of the suspension structure, they are able to demonstrate a very strong correlation between measured brake torque and longitudinal acceleration (as measured by on-board accelerometers). They also note that the signal is resistant to noise.

Unlike many control schemes, they do not attempt to measure precisely where on the tyre/road friction curve the wheel is operating. Indeed they acknowledge that they cannot. However, they are able to identify whether the wheel is operating in the stable or unstable part of the curve and are thus able to reduce the brake pressure when the maximum friction point has been passed.

Describing an "intelligent braking system", Gissinger et al. (2003) compare torque and slip control. They find that a torque loop is well suited to accounting for parameter variation within the brake's mechanical parameters while a slip loop is able to keep the tyre operating near the maximum of the friction curve. They recommend the use of an inner torque loop and an outer slip loop on each wheel.

Brake torque control requires some means of controlling the torque on multiple wheels independently. This may mean independent control of the torque on each wheel. Alternatively, just two (front) wheels can be controlled independently while the wheels on the other (rear) axle are controlled as one unit, as in the differential braking scheme described by Kraft & Leffler (1990).

Differential braking

Kraft & Leffler (1990) describe the principles behind differential braking as a traction control measure and as a means of ensuring stability when the surface friction available to each side of the car differs (split-μ) in a similar manner to that of a limited slip differential.

A differential acts to equalise the torque on each wheel and thus allows wheels to spin at different speeds, which is necessary for smooth turning (yaw) of the vehicle. If one wheel is on a low friction surface however, this wheel is unable to generate much grip and the action of an open differential prevents the wheel on the high friction surface from generating a higher torque than that on the spinning wheel. Off-road vehicles may get around this problem by using (part- or full-time) limited slip differentials or by providing the means to lock the differential entirely, thus equalising wheel speed rather than torque. Such mechanisms are however complex, heavy and expensive relative to simple open differentials. Application of torque to a wheel on a low friction surface increases the

torque in the differential and allows the wheel on a high friction surface to use more of the available adhesion. Such a torque can be generated by application of brake pressure, which is known by people who partially apply the handbrake to rear-wheel drive vehicles in order to escape from low friction surfaces.

Independent control of the brakes allows torque to be applied to only the wheel which is spinning (more closely approximating the behaviour of a fully locked axle) while also providing the capability for the control system to maintain directional stability which is an important part of traction control (Jung, Kwak & Park 2002).

Kraft and Leffler describe how only the front wheels of a production vehicle (BMW 850i, 1990) are controlled independently, the rear wheels being subject to control as a single unit. For this particular car, the front/rear brake proportion is 72/28, so any waste of adhesion available to the rear wheels is of minor consequence. For this vehicle, differential braking is combined with engine torque control. At the onset of significant slip on a wheel, the throttle is reduced (in combination with retardation of the ignition to improve engine sensitivity) and brake pressure is applied to the spinning wheel. As the wheel slip decreases, the brake pressure is gradually decreased while the engine is gradually throttled back to its normal level.

Engine torque control

Control of engine torque is an integral part of traction control schemes (Schuller et al. 2002); a reduction in drive torque is necessary if all the driven wheels begin to spin (Austin & Morrey 2000). In cases where wheels on only one side of the car are spinning, reduction of engine torque can effectively increase the amount of grip available to the tyres, although as noted by Kraft and Leffler this does not provide access to the full adhesion that is available on a split-μ surface. For that, some means of providing different torque to each side of the car is required.

Eren & Göktan (2001) propose the use of engine torque control, or the application of torque by an electric motor, to re-accelerate a locking wheel as a supplement to friction braking and claim improvements in stopping distance performance of 10% are possible on low friction surfaces. The authors do note however that inappropriate application of torque can cause the wheels to overspeed, resulting in longer stopping distances, and that accurate velocity estimation is essential for the system to be effective.

1.3.3 Steer-by-wire

Notwithstanding the availability of steer-by-wire augmentation and electronic stability programmes that could enable lateral collision avoidance manoeuvres to be initiated and performed under the guidance of a vehicle management computer, there is relatively little work reported on lateral emergency collision avoidance (Vahidi & Eskandarian 2004). Of particular interest, however, is the work of Shiller & Sundar (1998), who describe how vehicles are not always able to avoid a collision solely by braking; evasive steering manoeuvres are often required, particularly at higher speeds.

There has been research into the use of steer-by-wire to perform automatic lane changing manoeuvres such as that performed under the auspices of the California PATH project. However, this has tended to concentrate on gentle manoeuvres undertaken by autonomous vehicles travelling on intelligent highway systems (Rajamani, Tan, Law & Zhang 2000) or systems which are reliant on inter-vehicle communication (e.g. Kaneko & Shimamura (1997) and Swaroop & Yoon (1999)).

Following successful introduction of longitudinal collision warning and/or avoidance (CW/CA) systems on vehicles, some manufacturers are starting to introduce lane departure warnings, in which the car tries to attract the driver's attention to possible risks (Connolly 2007). Autonomous steering control can be implemented to perform lane-keeping for the driver but, as noted by Eidehall, Pohl, Gustafsson & Ekmark (2007), there are dangers if drivers come to rely on such systems as a form of autopilot.

Burgio & Zegelaar (2006) identify some of the difficulties related to integration of brake and steering controls. The problem is multi-input, multi-output (MIMO), intrinsically non-linear due to inherent tyre characteristics and suffers from a high degree of plant uncertainty due to variation in parameters. They propose a method of state feedback linearisation to produce a globally stable controller that uses the brakes only in critical cases.

1.3.4 Vertical dynamics

Automatic braking and steering are primarily concerned with controlling the longitudinal and lateral dynamics of the vehicle. Active suspension systems provide control over vertical dynamics. Describing the introduction of an active suspension system on the 1989 Toyota Celica, Aburaya, Kawanishi, Kondo & Hamada (1990) explain that the systems serve two purposes, namely improving ride quality (passenger comfort) and improving vehicle handling (controllability and stability). More recent research has seen active body control used for vehicle emulation and generic prototyping, e.g. Akar, Kalkkuhl & Suissa (2007) and Villegas, Leith, Shorten & Kalkkuhl (2007).

Ride quality and emulation are of little direct interest in an emergency situation. However, coupling between vertical and lateral dynamics is relevant. The effect of coupling can be particularly acute for high-sided vehicles which are at risk of rollover if excessively aggressive lateral manoeuvres are performed (Schofield, Hägglund & Rantzer 2006). As lateral dynamics affect vehicle roll, so the converse is true. Roll and pitch dynamics alter the normal load acting on each tyre, thus affecting the ability to generate longitudinal and lateral forces (Shim & Margolis 2005).

1.3.5 Path planning and construction

If a vehicle is to generate lateral forces, steering autonomously, a path must be determined for the vehicle to follow. In an emergency, two factors are of particular importance when choosing a path; it must be safe and feasible. A safe path is one which avoids obstacles and a feasible path is one that the vehicle is capable of following, given its actuator limits. In a non-emergency situation, passenger comfort and fuel economy might also be factors to be considered.

Fraichard (1991) explores possible collision-free trajectories for non-holonomic vehicles subject to minimum radius turn constraints for following polygonal splines. Paths are constructed from series of straight lines and circular arcs. A Curvature Centres Space is built and then searched to find the shortest unobstructed path. Later, Fraichard & Ahuactzin (2001) noted that paths must be created with smooth transitions between lines and arcs if the car is to follow continuously. They introduced clothoid arcs to transition between segments. The construction of Continuous Curvature paths in the absence of obstacles is described in detail by Fraichard & Scheuer (2004).

Lamiraux & Laumond (2001) note that clothoids do not have a closed form, making control of their shapes difficult and dangerous in the presence of obstacles. They propose a four-dimensional consideration of path generation from a kinematic point of view. The generation is performed in multiple stages, with non-holonomic constraints neglected in the first instance.

Pei & Horng (1998) consider path generation for navigating a car into a parking space - an obstacle avoidance problem at low speed. They suggest that most models are oversimplified and do not capture the intricacies of path planning in the real world. They propose a model in which a car can adopt one of eight orientations. Including vehicle orientation increases the complexity of the optimisation process but produces more realistic parking strategies than algorithms that do not. However, there does not appear to be any consideration of vehicle dynamic constraints in the method, which may reduce its usefulness for vehicles operating with high momentum.

Noting the computational expense of optimisation, Durali, Javid & Kasaiezadeh (2006) propose using sinusoidal or exponential trajectories for obstacle avoidance manoeuvres, which can be calculated easily in real time.

In the field of robotics, potential methods are commonly used for finding obstacle-free paths. Quinlan & Khatib (1993) introduced the idea of elastic bands for global path planning and control. Gehrig & Stein (2001) adopted this energy minimisation technique to navigate a vehicle through regular traffic. Their simulations have been validated with real world experimental results from a demonstration vehicle. Sattel & Brandt (2005) also suggest that modified elastic bands can be used to derive trajectories for autonomous vehicles.

Myers, Nöel, Parent & Vlacic (2005) observe that most researchers working on collision avoidance work with obstacles that are known *a priori*. They emphasise the need for algorithms that generate trajectories in real time, suggesting a Gradient Velocity algorithm, which their simulations show can be made to work for robots travelling up to 25 [km/hr], which they describe as “high speed”.

1.3.6 Measurement and estimation

Vehicle ground speed is usually measured by counting wheel revolutions using an inductive sensor attached to the wheel. A typical example of such a sensor is described by Austin & Morrey (2000). Although measurements obtained by these devices are perfectly adequate for normal driving, in the presence of longitudinal or side slip of the wheels such sensors prove to be inaccurate, as noted by Kobayashi, Cheok & Watanabe (1995). Operating at a vehicle’s physical limits, it is under conditions

such as these that a collision avoidance controller is required to perform in an emergency.

Kobayashi et al. (1995) explain that on a rear wheel drive car it is possible to generate accurate velocity data from the front wheel sensors provided that only the handbrake (which operates on the rear wheels) is used to reduce velocity. A similar principle is used for obtaining experimental data in the method described by Jiang & Gao (2000); they use a "fifth wheel" which is a non-driven bicycle wheel attached to the vehicle that can work as an effective tachometer. Obviously these methods may work on a test ground, but are not to be recommended for normal driving. Braking only the rear wheels is particularly dangerous because of the reduction in vehicle stability that this causes, as demonstrated by "handbrake turns".

A proprietary ABS estimates vehicle speed during a braking manoeuvre by periodically releasing the rear brakes and allowing those wheels to accelerate. In the absence of both a driving or braking torque, the wheels are accelerated by the road to a low slip condition, thus improving the accuracy of the induction sensors. The loss of braking traction that this entails is not considered to be a problem as the front wheels carry the greater braking load. There is also little risk of causing yaw instability with this method.

Many authors note that accurate, direct and non-contact means of obtaining vehicle ground speed exist, such as optical correlation or spatial filtering methods (Jiang & Gao 2000). Yi, Alvarez, Claeys, Horowitz & de Wit (2001) mention that monopulse radar can be used to estimate velocity on vehicles that are equipped with automatic cruise control. Such methods are very expensive to implement however and whilst they may be feasible during controller design, it is not practical to implement them on most production cars (Daiß & Kiencke 1995). On production cars, vehicle velocity estimation methods are likely to be used; accurate measurement of vehicle velocity is a significant problem for vehicle dynamics controller design.

Vehicle state estimation

Measurement of velocity is a significant problem in the design of vehicle dynamics controllers. Although measurement of the wheel angular velocity is easily obtained using inductive sensors, in the presence of wheel slip, the angular velocities measured do not accurately correspond with the motion of the vehicle body.

Tyres are not rigid and deform in the presence of a load. This deformation causes a larger area of the tread pattern to be in contact with the road than would be the case for a rigid wheel. As the wheel turns, the part of the tread pattern in contact with the road deforms further, so that its speed relative to the road does not precisely match that of the vehicle. This is the phenomenon known as wheel slip, various mathematical definitions of which are described by Milliken & Milliken (1995). With both the vehicle speed and wheel slip as unknown parameters, measurements of angular velocity from inductive sensors cannot give accurate estimates of vehicle speed.

Daiß & Kiencke (1995) estimate vehicle velocity by fusing measurement data from six sensors: inductive sensors on each wheel, a longitudinal accelerometer and a yaw rate gyroscope. The wheel

sensors are sufficiently accurate under normal driving (no-slip) conditions whereas the accelerometer and gyroscope are particularly suited to provide velocity measurements during short transient accelerations. They compare two methods of weighting the data to estimate a true vehicle speed: fuzzy logic and Kalman filtering. Both methods are shown to give excellent results. The authors note however that it is not possible to implement the Kalman filter in real time and that the fuzzy logic is the better choice. They also express the opinion that the fuzzy logic method is easier to design.

Imsland, Johansen, Fossen, Grip, Kalkkuhl & Suissa (2006) also note that Extended Kalman filters are troublesome for real time applications because of the need to solve Riccati differential equations. They propose an observer that fuses data from a sensor suite that adds measurements of lateral acceleration and wheel steering angles to the sensors used by Daiß and Kiencke. A nonlinear tyre-road friction model is used by the observer to fully exploit lateral acceleration measurements but the authors note that this does require that the coefficient of friction between the tyre and road is known or assumed.

Jiang & Gao (2000) present an interesting approach to velocity estimation that does not require the use of any accelerometers. Adaptive filtering is used based on the observation that a vehicle that is braking cannot go faster than its wheel and the assumption that the peak velocity of a wheel is instantaneously close to the true vehicle velocity. Good results from field trials are reported with the only significant estimation error occurring just at the start of a braking manoeuvre.

Yi et al. (2001) also attempt to estimate vehicle velocity using only data from wheel speed sensors. They use a model-based observer together with a dynamic modified LuGre tyre model. The authors show that their controller is able to bring a quarter car model to a halt at close to the maximum deceleration. Unfortunately, despite making a number of simplifying assumptions regarding the vehicle dynamics, the authors acknowledge that the velocity estimation fails to converge to the true value during simulations. They do assert, however, that although their method does not give an accurate estimation of velocity, this is not a major problem as vehicles have other means of determining velocity; radar and human perception are given as examples.

As the electronics on board cars continue to become more sophisticated, it may be the case that more accurate measures of velocity become available without the need to perform complex estimation on insufficient and noisy data. Liu, Lu, Shi & An (2001) comment on the synergies between anti-lock braking, traction control and automatic cruise control systems in terms of the sensors and actuators that are required. The widespread introduction of automatic cruise control in the future may be accompanied by speed measurement and/or command devices relying on communication with roadside beacons.

Friction estimation

The traction available to a tyre is directly proportional to the friction coefficient μ between the rubber and the road (Milliken & Milliken 1995). The coefficient of friction cannot be measured directly but there are a number of estimation methods reported in the literature.

The problem of estimating friction is closely coupled to that of determining the type of road surface on which the vehicle is travelling. If the surface is known, perhaps from optical or acoustic methods, it is possible to make a reasonable estimate (Andersson, Bruzelius, Casselgren, Gäfvert, Hjort, Hultén, Håbring, Klomp, Olsson, Sjödahl, Svendenius, Woxneryd & Wälivaara 2007). In the absence of such information, friction can only be estimated from the effects that frictional forces have on wheel or vehicle motion. These effects are only observed when the vehicle is braking or accelerating. A quiescent steady system in which the wheels are rolling at near constant velocity provides no information from which μ can be estimated.

If the braking torque is known, as in the case of the intelligent braking system of Gissinger et al. (2003), together with the vertical load and the wheel speed, then the frictional force at the wheel can easily be determined by differentiating the angular velocity of the wheel with respect to time. Most vehicles are not fitted with the appropriate sensors to generate these data but some commercial ABS sensors provide friction estimates as a result of wheel acceleration in response to brake pressures, which are assumed to be proportional to the braking forces.

Many attempts have been made to estimate μ in real time. Germann et al. (1994) have developed a friction monitoring system that computes wheel loads and longitudinal tyre forces on the basis of models of friction-slip characteristics. Meanwhile, Pasterkamp & Pacejka (1997) use neural networks to estimate friction on-line. Gustafsson (1997) attempts to detect abrupt changes in friction from slip measurements. Yi, Woo, Kim & Lee (1999) use friction estimates to determine safe spacing distances for vehicles as part of a collision warning/avoidance system.

Samadi, Kazemi, Nikravesh & Kabganian (2001) note that despite the existence of many tyre models, significant differences between models and real road behaviour are found. Rather than using a physical model, they propose the use of an extended Kalman Filter for estimating friction parameters. All of these methods have great difficulty when the system is quiescent and, even under ideal circumstances, are unable to predict the friction conditions that will be found in the road ahead. Despite significant research effort, μ remains a highly uncertain parameter for automotive control applications.

Yi et al. (2003) note that even when friction parameters are relatively well known, it is not sufficient to consider only wheel velocities to predict braking performance and vehicle speed because the dynamic system is almost unobservable using just those measurements. They therefore use the pressure within the master brake cylinder to estimate the forces involved.

Choi et al. (2002) estimate friction using a sliding mode observer with a fuzzy logic algorithm. Liu et al. (2001) propose to determine the surface type on the basis of the maximum acceleration achieved by the vehicle body. An appropriate friction coefficient is then selected based on the estimated surface type. This technique is equivalent to considering the system as being composed of multiple models, a topic that is discussed further in Section 1.3.7.

The addition of a change detector to the friction estimator developed by Gustafsson (1997) enables detection of a change in road surface within four samples (0.8 [s]). A number of change detectors are compared and a cumulative summation (CUSUM) detector recommended, primarily because of its

simplicity.

Despite the great effort of many researchers, real-time friction estimation remains a challenging problem, introducing parametric uncertainty that complicates the design of vehicle controllers.

1.3.7 Control techniques

A wide variety of control techniques have been explored for automotive control, ranging from classical linear techniques to modern nonlinear methods.

Multiple plant models

Kalkkuhl, Johansen & Lüdemann (2002) use a finite set of parameter models with an estimator resetting routine to switch between models. The various models represent different conditions and the decision to switch between them is taken on the basis of guaranteed reduction in a Lyapanov function. Obviously, given the uncertainty in the plant, this guarantee can only be given if some bounds are placed on the uncertainties. Solyom & Rantzer (2003) use the method of Kalkkuhl et al. (2002) together with cone-bounded uncertainties in the plant to develop a gain-sheduled pair of PID controllers. Two controllers are scheduled because the friction curve is approximated as two linear segments representing a stable and an unstable region. In this case the authors recommend placing the integral gain inside the numerical integration to smooth transitions.

Simultaneous stabilisation

An alternative method of handling variation in the plant parameters is to develop a robust controller that is able to handle all variation of those parameters. Hunt, Wang, Schinkel & Schmitt-Hartmann (2003) describe a method of solving the (strong) simultaneous stabilisation problem (S/SSP) which requires a single controller to place the closed loop poles of multiple plants within a specified D-region of the complex plane. The development of a single control law to handle all variation of friction coefficient would allow a brake controller to be implemented without using friction observers which, as noted by Schinkel & Hunt (2002), are well known to have poor properties. Unlike conventional pole placement methods, this technique does not attempt to place the poles arbitrarily (standard optimisation routines are used to find (locally) optimal solutions) so full state feedback is not required.

Feedforward and robust control

Nouillant et al. (2002) advocate the use of a feedforward and robust feedback controller as a solution to the problem of uncertainty in the friction between the tyre and road. The feedforward controller contains an inverse model of Pacejka's magic formula; the feedback controller is a CRONE (Commande robuste d'ordre non entir: robust control of non-integer order) controller designed for use with a hydro-pneumatic (i.e. gas spring) suspension system (Oustaloup, Moreau & Nouillant 1996). The control strategy is to maintain near-optimal acceleration (by means of slip control) where

the target acceleration is determined for the road surface at the start of braking. Stopping distance improvements of up to 30% are claimed on icy surfaces from simulation results but this control is dependent upon the use of proportional servo-valves which are capable of applying continuous control. The obvious weakness of such a strategy is that it will not cope well if the road surface changes suddenly during braking.

Hybrid (finite state) methods

There are two senses in which vehicle controllers may be said to be hybrid. Virtually all electronic brake control systems may be said to be hybrid in the sense that digital controllers operate at discrete intervals whereas the plants operate continuously. The controllers may also be hybrid in the sense that they include some form of moding or finite state machine within the algorithm (Johansen et al. 2001). These two types of hybridity are neither mutually exclusive nor dependent.

The simplest finite state control is probably that known as “bang-bang” control in which actuators are active until a certain condition is met at which point they become inactive. Control methods such as this are used in the brake system of some vehicles based on the state of the wheels (e.g. locked or rolling) according to Jiang & Gao (2001), however the references that they cite date to the late 1980s and may perhaps no longer be valid.

Finite state methods may be used in vehicles to alter the slip target based on road type. Combined with friction estimators, finite state machines can be used to select appropriate control laws based on the road conditions as described by Liu et al. (2001) who select between multiple fixed accelerations to control a vehicle under automatic cruise control.

PID control

One of the great advantages of Proportional + Integral + Derivative (PID) controllers is the ease with which they may be tuned. Jiang & Gao (2001) highlight the importance of controllers being tunable and testable by field engineers. They decry the example of loop-shaping controllers based on linear models which work well in simulation but cannot be tuned in an industrial setting, and note the similar difficulties arising from controllers based on more advanced control strategies such as fuzzy logic control, model reference control and neural networks. Similar comments about the simplicity of PID design compared to model-based controllers are made by Solyom & Ingimundarson (2002).

There are numerous well established tuning methods for designing PID controllers which, as noted by Panagopoulos, Åström & Hägglund (2002), range from those which are very easy to tune and require relatively little plant knowledge to those which require greater insight into the plant behaviour. A number of authors present methods of tuning PID controllers. An important theme in many of these is the means by which model uncertainty is dealt with, particularly the use of constrained optimisation methods. Solyom & Ingimundarson (2002) use a cone-bounded optimisation method for tuning PI and PID controllers in which boundaries for nonlinearities and uncertainties within the plant model are ascertained and used to provide constraints; for PI controllers, the authors highlight the benefits

of a graphical approach to finding starting values for the tuning process, while for PID controllers, where the parameter space may not be so easy to visualise, a numerical optimisation routine is used.

Kristiansson & Lennartson (2002) emphasise the necessity of including any filter dynamics early in the design phase when presenting their tuning rules for robust and near-optimal PID control and stress that it is necessary that all available parameters are free to be designed.

Jiang & Gao (2001) find conventional linear PID controllers to be capable of only limited performance. To overcome this, they propose a nonlinear PID controller to implement a smooth form of gain scheduling by mapping the error signal χ to a function of the form $\chi^{\alpha} : \alpha \in [0, 1]$ which progressively reduces the gain as the error signal is increased. A linear region around the origin improves numerical stability for very small errors.

One of the disadvantages of traditional tuning methods for PID controllers is that they only work on stable plants. Paraskevopoulos, Pasgianos & Arvanitis (2004) describe several tuning methods for pseudo-derivative feedback controllers which can be used to stabilise unstable first order plants with dead time and avoid the need for integral control terms which can lead to excessive overshoot in the closed loop response.

Sliding mode control

The use of sliding mode control for braking applications is investigated by very many authors, for example: Choi et al. (2002), Canudas de Wit & Tsiotras (1999), Heinzl et al. (2002), Jung et al. (2002), Kazemi & Zaviyeh (2001), Wu & Shih (2003) and Yu & Ozguner (2002). The popularity of this technique is because of the robustness of sliding mode control.

Schinkel & Hunt (2002) assert that it is not possible to design a continuous feedback controller to achieve maximum deceleration and therefore investigate a sliding mode controller. The assertion depends on the assumption that a wheel may not be accelerated while braking, which may not be strictly true as it may be accelerated either by the road or an external torque (such as that proposed by Eren & Göktan (2001)).

A sliding mode brake controller is developed by Yi & Chung (2001) as part of a collision warning/collision avoidance (CW/CA) system. Sliding mode control is used because it can account for the significant uncertainties in the non-linear dynamics of brake actuators.

Pole placement

Both the linear and nonlinear observers developed by Kiencke & Daiß (1997) were developed using pole placement techniques, with good results reported. Chamaillard et al. (1994) state that it is advisable to start with well known control techniques before starting to evaluate more sophisticated methods. They used pole placement as a starting technique for controller design because it is a method with which the authors are familiar. An introduction to the method of designing controllers by solving the Diophantine equation is given by Åström & Wittenmark (1997).

1.3.8 Modelling tools

A Mathworks Automotive Advisory Body (MAAB) issues guidelines (Donald 2001) for the use of Matlab, Simulink and Stateflow for automotive applications and Matlab Simulink is described as the industry standard tool for controller design by Schuller et al. (2002). However, it is noteworthy that they use other tools for the detailed modelling of physical components, such as hydraulic elements of the brake system. Various Matlab toolboxes are also mentioned in the literature, for example, the change detection toolbox (Gustafsson 1997), the neural net toolbox (Pasterkamp & Pacejka 1997) and the optimisation toolbox (Hunt et al. 2003).

Bond graphs represent power transactions between subsystems and can provide useful information to control engineers (Gawthrop & Bevan 2007). The use of bond graphs for modelling mechanical systems, particularly suspension elements, is expounded in a number of French papers, for instance by Chamaillard et al. (1994) and Gissinger, Chamaillard & Stemmelen (1995). Konik (2002), meanwhile, uses Matrix$_X$ to model vehicle dynamics and perform rapid prototyping in the development of a "Dynamic Drive" active suspension system. The TruckSim tool was used by Jiang & Gao (2001) and used with Matlab in the form of C-Mex files.

Jansen, Zegelaar & Pacejka (1999) describe a rigid ring tyre model that can be linked to Simulink; the tyre model was created using Madymo and links to a Fortran body model.

The use of ADAMS for detailed kinematic modelling with many degrees of freedom is mentioned by numerous authors, e.g. Lidner (1993), Heinzl et al. (2002) and Pauwelussen, Gootjes, Schröder, Köhne, Jansen & Schmeitz (2003).

1.3.9 Standard manoeuvres

Lidner (1993) recommends the use of ISO TR 8725 which is used by Volvo as a good test for steering properties under normal conditions and for stability under high lateral acceleration. A number of ISO standards (many also incorporated as British Standards) exist describing test procedures for passenger cars (ICS 43.100), road vehicle systems (ICS 40.040) and road vehicles in general (ICS 43.020). Most of these standards are primarily designed to be used to set up conditions for conducting tests on real vehicles, however they may potentially be of some use for developing controller evaluation criteria.

ISO 3888-1:1999 Passenger cars – Test track for a severe lane-change manoeuvre – Part 1: Double lane-change

ISO 3888-2:2002 Passenger cars – Test track for a severe lane-change manoeuvre – Part 2: Obstacle avoidance

ISO 4138:1996 Passenger cars – Steady-state circular driving behaviour – Open-loop test procedure

ISO 6597:2002 Road vehicles – Motor vehicles with hydraulic braking systems with and without antilock device – Measurement of braking performance

ISO 7401:2003 Road vehicles – Lateral transient response test methods – Open-loop test methods

ISO 7975:1996 Passenger cars – Braking in a turn – Open-loop test procedure

ISO/TR 8725:1988 Road vehicles – Transient open-loop response test method with one period of

sinusoidal input

ISO 8726:1988 Road vehicles – Transient open-loop response test method with pseudo-random steering input

ISO 9815:2003 Road vehicles – Passenger-car and trailer combinations – Lateral stability test

ISO 9816:1993 Passenger cars – Power-off reactions of a vehicle in a turn – Open-loop test method

ISO 11835:2002 Road vehicles – Motor vehicles with antilock braking systems (ABS) – Measurement of braking performance

ISO 12021-1:1996 Road vehicles – Sensitivity to lateral wind – Part 1: Open-loop test method using wind generator input

ISO 13674-1:2003 Road vehicles – Test method for the quantification of on-centre handling – Part 1: Weave test

ISO 14512:1999 Passenger cars – Straight-ahead braking on surfaces with split coefficient of friction – Open-loop test procedure

ISO 15037-1:1998 Road vehicles – Vehicle dynamics test methods – Part 1: General conditions for passenger cars

ISO 15037-1:1998/Cor 1:2001

ISO 17288-1:2002 Passenger cars – Free-steer behaviour – Part 1: Steering-release open-loop test method

ISO/TS 20119:200 Road vehicles – Test method for the quantification of on-centre handling – Determination of dispersion metrics for straight-line driving

1.3.10 Measures of performance

Determining an objective set of criteria for evaluating controllers is an important part of the optimisation process as well as a necessary prerequisite for determining when a design is complete.

Pauwelussen et al. (2003) measure the time required to bring a vehicle to a halt in their comparison of tyre models. Perhaps a more directly relevant measure in an emergency situation is the stopping distance achieved by the controller under examination. Stopping distance improvements in response to brake control designs are cited by numerous authors, for example Eren & Göktan (2001) claim a 10% reduction in stopping distance is possible using their external applied torque, Gissinger et al. (2003) report improvements of a few percent for their intelligent braking compared to conventional ABS. In general, authors reporting results of simulations tend to claim greater improvements than are claimed by those reporting test track data. It is advisable to be somewhat cautious of simulation results, at least until the models used have been calibrated and validated against actual test data.

The performance of individual components may be measured as well as that of the overall vehicle/controller combination. Jiang & Gao (2001) use the 2-norm error of wheel velocity as a criterion in their report on nonlinear PID control. Improvement in velocity estimation is likely to improve controller performance regardless of the method used and it would seem to be worthwhile to measure such parameters independently if possible.

Schuller, Schaeffer, Neukum & Krueger (1999) describe a ratings module which attempts to predict subjective assessment of driver workload from objective measures derived from simulation. The concept of driver workload could provide a useful criterion for filtering out poor controller designs as it is to be expected that good controller combinations will not require excessive work load from the actuation systems.

1.4 Critique

The dominant theme emerging from the review of the state-of-the art is the importance of non-linear tyre behaviour on the overall vehicle dynamics. Sustained research into the traction generated by tyres has led to several alternative models, which fall broadly into four categories, classified either as dynamic or steady state; and empirical or physics-based. The more detailed models are shown to better represent observed tyre behaviour, but there are two important caveats. Firstly, these detailed models require large sets of experimental data for calibration and parameterisation; data which are very expensive to obtain and not readily available. Secondly, all models are highly dependent upon a significantly uncertain parameter, namely the level of friction between the tyre and road surface.

Considerable effort has been expended by researchers investigating methods of observing or inferring the nature of the surface on which a vehicle is travelling. Methods relying on optical correlation or analysis of vibration appear promising for identifying the type of road surface, while analysis of wheel acceleration data allows more direct inference of the available friction. Nevertheless, the friction coefficient must still be regarded as highly uncertain. Consequently, there is no justification for using an excessively detailed model that captures dynamic behaviour orders of magnitude less significant than the parametric uncertainty in the system. The Magic Formula tyre model is predominant in the published literature for vehicle dynamics control and appears to be an excellent compromise between the need to accommodate highly non-linear characteristics while respecting the inherent parametric uncertainty and difficulty of obtaining experimental test data.

Several brake control strategies are presented. By avoiding dependence on detailed knowledge of the tyre characteristics, Gissinger's Intelligent Braking System has highly attractive properties. Unfortunately, it entails extensive redesign of the braking and instrumentation systems and is not therefore easily adaptable to existing vehicles. Among the control strategies described for more conventional braking systems, PID and sliding mode control are pre-eminent. Sliding mode control appears to be favoured by academics, because of its robustness in the face of matched uncertainties. However, PID control features heavily in the literature that is more closely associated with industrial practitioners. This is unsurprising given the widespread use of PID control in industry generally. It is clear that any practical controller must be implemented in such a way that it can be easily adapted by test engineers without requiring adjustment to detailed mathematical models of the system behaviour.

With little published research into steer-by-wire vehicles, it is unsurprising that there is little guidance to be offered into the design of integrated steering and braking controller design or trajectory generation. The most relevant related fields are those of aerospace and robotics research. However,

each of these differs significantly from road vehicle dynamics. Multi-input multi-output (MIMO) control systems are common in aerospace control, but for aircraft there is usually a surplus of actuators that enable control actions to be separately allocated to some extent, for example, engines can control speed while aerodynamic surfaces control attitude. The main lesson that can be learnt from the field is the importance of ensuring that individual control loops interact well with each other when operating in parallel. Non-holonomic wheeled robots are more similar to road vehicles in the sense that all propulsive forces are generally generated through the wheels. However, with 25 [km/hr] considered to be a high-speed in the field, there are substantial differences to automotive control problems, where the momentum and kinetic energy - and hence stopping distances - involved are significantly higher. Within automotive control, the bulk of research effort has been focused on gentle manoeuvres performed by vehicles operating comfortably within their limits either as platoons, interacting with other vehicles, or following markers in the road. Generation of feasible trajectories for emergency lateral manoeuvres therefore remains an outstanding problem.

Finally, it is necessary to comment on the greater performance improvements claimed by researchers working with simulated data, compared to their experimental counterparts. It is reasonable to suppose that researchers relying on simulated data may be dealing with more exotic designs than those constrained by current technology and cost limitations. However, it is also likely that the use of vehicle models that do not accurately capture all the intricacies of vehicle dynamics lead to optimistic predictions. This would reinforce the need to ensure that any practical design based on simulation should be capable of being adapted with relative ease when experimental data become available.

1.5 Aims and objectives

An obstacle avoidance capability is an essential prerequisite for any automatic lateral collision avoidance system that is intended to safely navigate a vehicle in the presence of obstructions. The aim of this research is to develop a vehicle dynamics controller for a car equipped with steer-by-wire and brake-by-wire systems. The controller is required to perform an obstacle avoidance manoeuvre with the car operating at its physical limits.

The development of a practical lateral collision avoidance system is a long term goal that requires maturation of many key technologies. The work presented here is focused on the control of vehicle dynamics rather than the environment in which a vehicle operates. As such, the scope is restricted to the development of an automatic controller. Sensing technology, data fusion and driver-vehicle interaction issues are specifically excluded.

The aim, then, is to develop an integrated steering and braking controller that is capable of causing a passenger vehicle to perform an emergency lateral obstacle avoidance manoeuvre. Specific objectives, identified from the preceding literature review, are as follows:

1. development of a vehicle model that captures significant dynamic effects, while remaining sufficiently robust in the face of large parametric uncertainty, particularly in relation to the

friction coefficient;

2. development of a trajectory generation algorithm for identifying a feasible course through a specified obstacle course; and
3. development of an integrated steering and braking controller to perform the specified manoeuvre automatically, i.e. in the absence of human interaction.

If a vehicle faces an obstacle in the lane in which it is travelling, it would usually be appropriate to move to an adjacent lane, if it is free of obstructions. Thus a single lane-change is the basic element of a lateral obstacle avoidance manoeuvre. After performing a single lane-change, unless it is required to take further evasive action, the vehicle may remain in its new lane or return to the old one. On a dual carriageway or multi-lane road, if there is no further danger, it would be sensible for an automatic controller to take the least action necessary, allowing the driver to determine the best course of action after regaining full control of the vehicle.

In the early stages of this research, it was therefore decided that the objective should be to cause a passenger car to perform a single lane-change while operating at its physical limits, i.e. subject to traction and actuator constraints.

As the research progressed, it was decided to make the exercise more challenging and to require the vehicle to instead perform a double lane-change, a task which places tighter constraints on the vehicle's acquisition of the new lane and thus requires greater control to be exerted over the vehicle. This more demanding problem would be an appropriate manoeuvre in circumstances where, upon acquiring a new lane, it is not safe to continue travelling in it, perhaps because of further obstacles.

Satisfaction of the aims and objectives is demonstrated by simulation.

1.6 Contribution

This thesis makes the following contributions to the state of the art.

1. A non-linear vehicle model which is capable of producing velocity-based linearisations at non-equilibrium conditions has been developed and implemented in a form suitable for simulation, either as standalone code or within Matlab or GNU Octave.

2. A new trajectory generation method has been developed. Convex formulations of an emergency obstacle avoidance problem are used during a series of optimisations to generate and refine a feasible trajectory. The method is suitable for application to more general automotive trajectory generation problems.

3. A new controller design strategy has been developed based on simulation using the aforementioned model. The method allows effective integration of steering and braking controllers despite the highly non-linear nature of the interaction between the two sub-systems. The controller is of a form that is amenable to efficient tuning by engineers in a test environment and therefore practical for implementation on production hardware.

In total, these contributions provide an essential element of a collision avoidance system, providing a means for a vehicle to navigate around an obstacle in its path. The following journal publications and conference presentations have so far arisen as a direct result of this research:

- Bevan, Gollee & O'Reilly (2007*a*). Automatic lateral emergency collision avoidance for a passenger car. *International Journal of Control*, 80(11):1751–1762, November 2007;
- Bevan, Gollee & O'Reilly (2007*b*). Trajectory generation for road vehicle obstacle avoidance using convex optimisation. Submitted to *Vehicle Systems Dynamics*, June 2007;
- Bevan, O'Neill, Gollee & O'Reilly (2007). Performance comparison of collision avoidance controller designs. *IEEE Intelligent Vehicles Symposium*, Istanbul, June 2007;
- Bevan, Gollee, and O'Reilly. Automatic lateral collision avoidance for a passenger car. *UKACC International Control Conference / EPSRC graduate workshop*, Glasgow, August 2006;
- Bevan, Gollee, and O'Reilly. Automatic lateral collision avoidance for a passenger car. *CEmACS/HyCon workshop*, Lund, June 2006.

1.7 Conclusion

The research topic has been introduced and the case for considering the introduction of emergency lateral collision avoidance systems in passenger cars has been made on grounds of safety. The state of the art of key technologies related to automotive dynamics in emergency situations has been reviewed. The published literature has been found wanting in regard to trajectory generation for emergency obstacle avoidance manoeuvres. It has also been found that a high degree of parametric uncertainty must be expected in relation to the friction coefficient between tyres and the road and that this parameter has a significant effect on overall vehicle dynamics.

The contributions and structure of this thesis have been detailed and a general problem specification has been outlined. Project aims and objectives have been specified, namely to develop a method for determining a feasible trajectory through an obstacle course; to develop an emergency lateral obstacle avoidance controller, integrating steering and braking subsystems, that is capable of causing a target vehicle to follow that trajectory; and to demonstrate this by simulation.

The remainder of this thesis is arranged as follows.

The focus of Chapter 2 is on modelling vehicle dynamics. The basic equations of motion and common tyre models are described and a highly complex proprietary vehicle model is introduced. The development of a non-linear model for controller design is then described. The model includes a velocity-based linearisation of the system, derived symbolically, which is used during subsequent controller design.

In Chapter 3, two trajectory generation methods are described. The first method, based on circular arcs connected by straight lines, is similar to path planning algorithms used for robotic control. It is used to calculate feasible trajectories at constant speed that would cause a following car to approach the traction limits of its tyres. The second method generates trajectories by means of a series of convex optimisations. Trajectories can be generated for a more general range of manoeuvres, but a

vehicle following a trajectory generated by this method would be required to decelerate while turning, thus requiring integrated steering and braking control.

The design of an integrated steering and braking controller for is described in Chapter 4. The design method uses the controller-design model of the preceding chapter to produce charts that enable controller parameters to be tuned.

Evaluation of results is presented in Chapter 5. The two feasible trajectory generation methods of Chapter 3 are first compared, using the controller-design model to demonstrate qualitative differences between the trajectories that each produces. The performance of the obstacle avoidance controller developed in Chapter 4 is then evaluated by simulation, using both the controller-design model and complex proprietary model introduced in Chapter 2.

Conclusions are presented in Chapter 6, followed by code listings for the trajectory generation routines and controller-design model, which appear in the Appendix.

In the next chapter, a description of the target vehicle for this research will be presented together with vehicle models that capture its essential dynamics during an aggressive lateral manoeuvre.

Chapter 2

Vehicle models

The purpose of computing is insight, not numbers.

— Richard W. Hamming [1973]

The motivation for considering lateral collision avoidance systems for passenger cars is outlined in Chapter 1 in conjunction with a review of the state of the art. In this chapter, details of the target vehicle are specified, along with its governing equations and models derived therefrom. A controller design based on these models is presented in Chapter 4.

Two modelling requirements arise for this research. Firstly, a model is needed to facilitate controller design. Secondly, a model is required for evaluating the performance of the controller. These requirements need not be satisfied by the same model. For controller design, the model must capture the essential characteristics of the system – particularly the dynamics – but remain sufficiently simple to be amenable to analysis and guide decision making. For controller evaluation, the model should ideally offer a very close approximation to reality.

In the following sections, the target vehicle for the research is introduced, followed by a description of a complex proprietary model of that vehicle. Subsequently, a simpler non-linear model of the vehicle dynamics is developed for controller design purposes. After an introduction to velocity-based linearisation, the technique is used to derive linear models from the non-linear controller design model. Finally, the model is verified by simulating various manoeuvres, the outcome of which can be readily calculated from first principles.

2.1 Vehicle and actuators

The target vehicle for this research is a Mercedes 'S' Class, "TS" CEMACS (2005), that has been modified by DaimlerChrysler Research and Technology, Intelligent Systems Group in Stuttgart. The vehicle is equipped with a CAN bus (ISO 11898) to which several on-board computers have been connected, together with an extensive array of sensors. These include wheel speed sensors, a gyroscopic inertial navigation system (INS) and global positioning system (GPS) receiver. The INS can provide accurate and frequent (100 Hz) body acceleration measurements, which may be filtered and integrated to provide velocity and position estimates. Lower frequency (10 Hz) GPS measurements can be used to correct for sensor drift in the INS. An on-board observer collates the various sensor data and can provide timely best estimates of the vehicle states.

The car is equipped with a four-wheel independent brake-by-wire system and a front-wheel steer-by-wire system. In production vehicles, an Electronic Stability Programme (ESP) automatically applies differential braking forces to induce a yawing moment which acts to stabilise the vehicle yaw rate during aggressive turns. However, commercial ESPs tend to be fairly conservative and act to maintain a safety margin of the order of 10% from the theoretical limits. The original ESP on the car has been modified to allow the standard production algorithms to be bypassed and new commands to be injected into the anti-lock braking system (ABS). It is therefore possible to command the ABS to take the vehicle closer to its physical limits. As long as the wheels remain unlocked, the ABS is able to produce a longitudinal force at each wheel on demand. Lateral tyre forces can be produced by steering the front wheels. However, there is no facility for demanding precise lateral forces; instead it is necessary to consider how the lateral slip angle α induces lateral forces and to steer the wheels accordingly.

Table 2.1: Steer-by-wire sensor and actuator constraints.

Sensor/actuator limitation	**Value**
Steering rate limit	160 [rad/s]
Sample rate	100 [Hz]
Delay	40 [ms]

Table 2.2: Brake-by-wire sensor and actuator constraints.

Sensor/actuator limitation	**Value**
Bandwidth	15 to 18 [rad/s]
Sample rate	50 [Hz]
Delay	20 [ms]
Maximum rate (pressure rise)	0.5 [kbar/s]
Maximum rate (pressure drop)	2 [kbar/s]

The steer-by-wire and brake-by-wire systems each have inherent sensor and actuator limitations, namely communication delays, sample rate limits and actuator rate limits. Additionally, the brake system is only able to satisfy the demand for a precise longitudinal force if the wheel is not locked,

but lock detection is not instantaneous. These sensor and actuator limits are specified in Tables 2.1 and 2.2.

2.2 Proprietary nonlinear model: CASCaDE

A proprietary vehicle model, known as *CASCaDE* (Rauh 2003) has been provided by Daimler-Chrysler. CASCaDE has been developed over several years. It offers a simulation harness for integrating modules that may be switched in or out as required (Ammon, Gipser, Rauh & Wimmer 1997). The model is parameterised so that it can represent a range of vehicles. It has been used extensively by DaimlerChrysler for predicting behaviour before experimenting with real vehicles on the test track. Having been subject to a high degree of validation, it is reported to give an accurate and realistic representation of actual vehicle performance.

CASCaDE has been provided as a "black box" model, implemented as a Simulink S-function that uses an encrypted, pre-compiled Fortran library and offers interfaces to representations of the sensors and actuators that are potentially available on the car. The version made available for this research is parameterised to represent the target vehicle: "TS", a Mercedes 'S' Class research vehicle. The car is rear wheel drive with front wheel steer-by-wire and all-wheel independent brake-by-wire capability. In addition to translational and rotational body dynamics, the vehicle's anti-lock braking system (ABS) and electronic stability programme (ESP) are modelled, together with tyre models based on data gathered for the tyres fitted to the car.

The CASCaDE model offers a useful means of evaluating the likely overall response of the vehicle to a set of control inputs; and hence the effectiveness of any controller design. Significant features that are represented in CASCaDE include:

- engine and powertrain dynamics;
- anti-lock braking system dynamics, including an optional electronic stability programme;
- vertical translation (suspension) dynamics, including optional active body control;
- pitch dynamics;
- independent movement of the body relative to the chassis;
- detailed actuator and sensor characteristics;
- sensor locations and correction algorithms;
- disturbances, such as wind-fields;
- detailed mass distribution;
- variable friction surfaces; and
- automatic controllers and closed loops for various systems, such as drive torque, gear changing and speed control.

The interaction of subsystems that are not directly relevant to the design task can inhibit analysis of salient behaviour. Furthermore, as a "black box" model, parameterised for a specific research vehicle, the inner workings are largely unknown. CASCaDE is too complex to be used in the design task itself, so it is necessary to develop a simpler vehicle model for controller design.

2.3 Axis systems

Conflicting standards exist for the orientation of axis sets for automotive applications. ISO-8855:1991 and SAE-J1594:1989 both define right-hand orthogonal axis systems with the forward direction defined as positive. However, whereas the Society of Automotive Engineers defines axes to the right and down to be positive – consistent with aerospace and nautical conventions – the International Organization for Standardization defines positive axes to the left and up. Following the textbooks by Milliken & Milliken (1995) and Gillespie (1992), the SAE-J1594:1989 convention is used throughout this work. Positive directions are: forward, right and down.

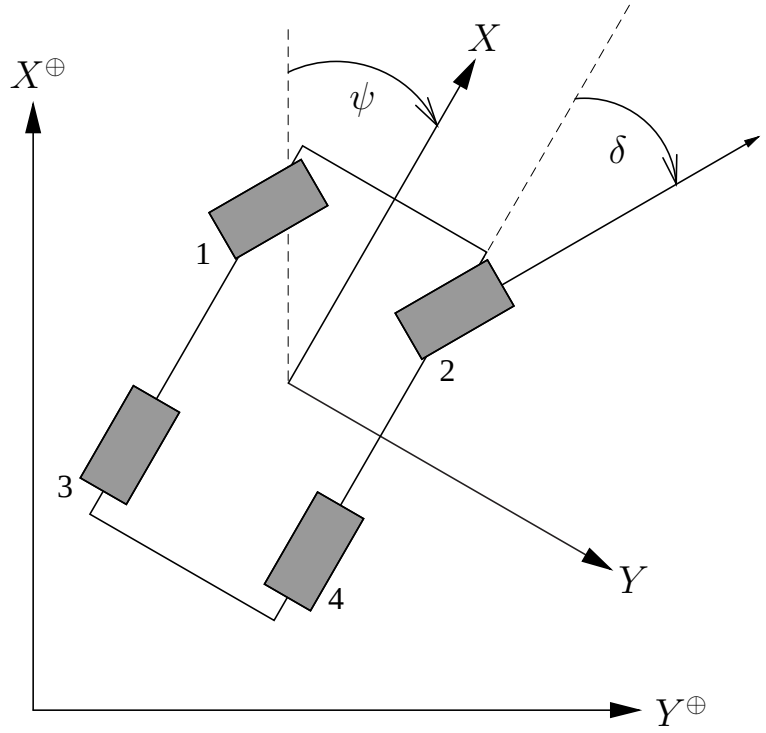
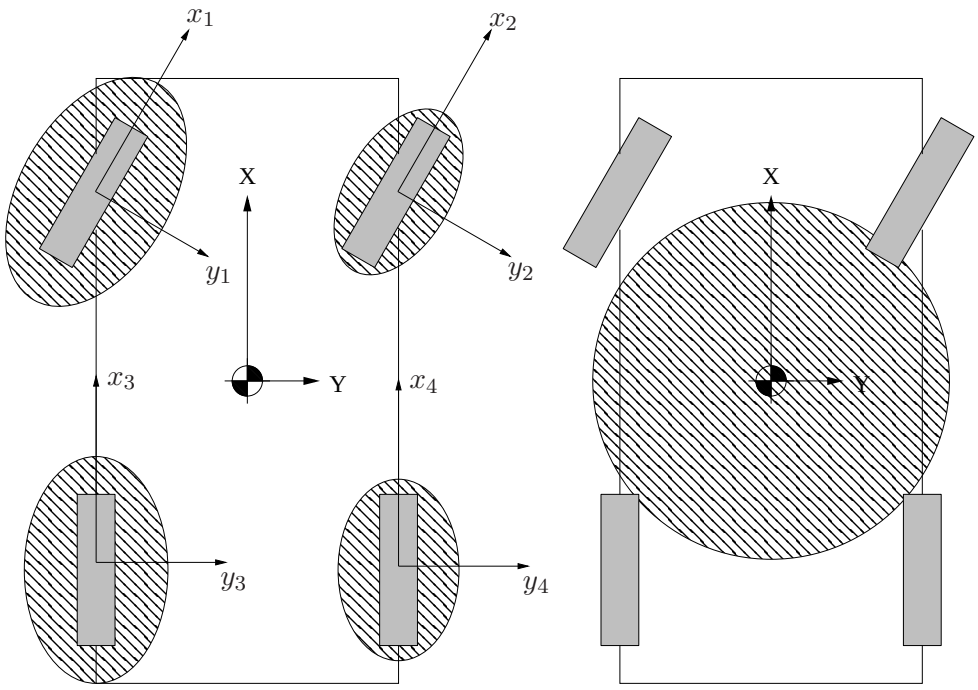


Figure 2.1: Fixed Earth and body axis systems.

Two axis systems are shown in Figure 2.1. The vehicle body axis system is a right-hand orthogonal axis set (X, Y, Z) centred on the vehicle centre of mass with X defined positive forwards along the centre-line of the vehicle and Y defined positive to the right. This axis set can be used to describe the geometry of the car and the vehicle velocity vector $(\dot{X}, \dot{Y}, \dot{Z})$. Since this axis system moves with the vehicle, it is not useful for measuring vehicle position relative to the ground. Therefore, a fixed Earth axis system $(X^{\oplus}, Y^{\oplus}, Z^{\oplus})$ is defined to be co-located and aligned with the vehicle axis at some point before the start of any manoeuvre but does not subsequently move with the vehicle. The horizontal angle of rotation between these axis systems is the vehicle heading angle, ψ. In the absence of pitch and yaw rotation, velocity and acceleration vectors in the vehicle body axis system can be converted into the fixed Earth axis system by rotating the vectors through ψ radians

$$\dot{X}^{\oplus} = \dot{X}\cos\psi - \dot{Y}\sin\psi \qquad \dot{Y}^{\oplus} = \dot{X}\sin\psi + \dot{Y}\cos\psi \qquad \dot{Z}^{\oplus} = \dot{Z} \tag{2.1}$$



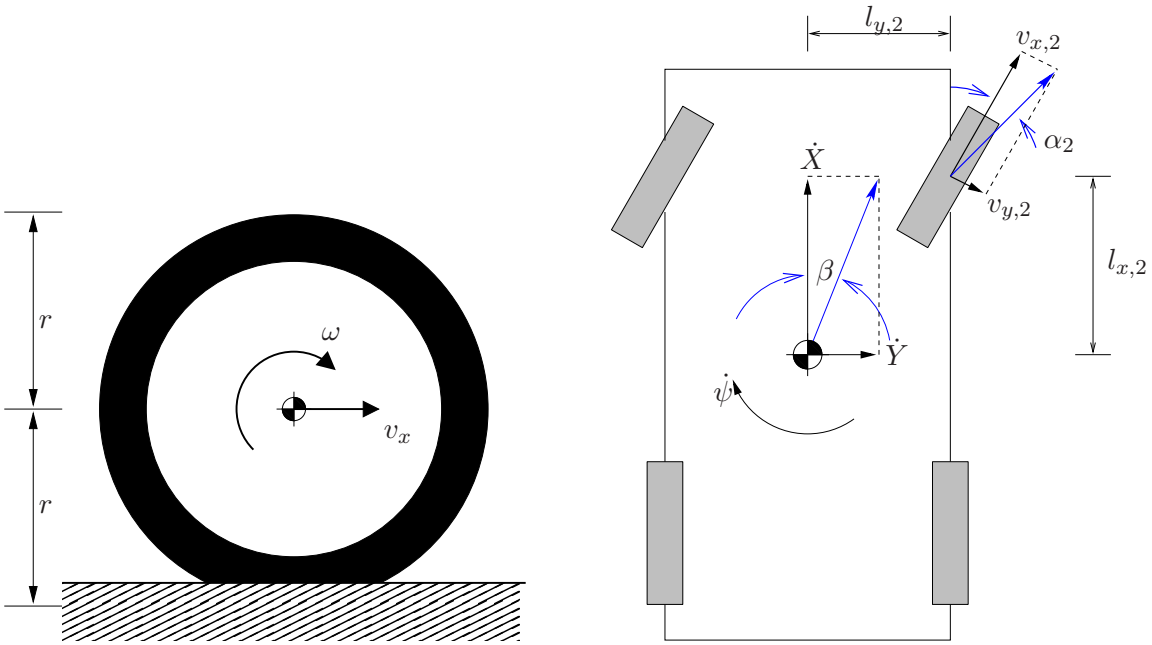
(a) Friction ellipses for each tyre. (b) Friction circle for the vehicle.

Figure 2.2: Friction ellipses. The maximum achievable force is a function of the friction coefficient μ and the vertical load. It is usually greatest in the wheel's longitudinal direction. Combining the friction ellipses from each tyre, a friction circle of radius μmg [N] approximately describes the maximum force achievable by the vehicle in any direction. .

Four further axis systems (not shown) are defined; one for each wheel. These axes (x_i, y_i, z_i) $\forall i \in [1, 4]$ are aligned with, and centred upon, each of the wheels, which are labelled as: 1) front left; 2) front right; 3) rear left; and 4) rear right. For the front wheels, steered through an angle δ [rad], these axes are rotated δ [rad] relative to the vehicle body axis. The rear wheels do not steer and hence there is no rotation of the rear wheel axes relative to the vehicle body axis.

2.4 Tyre forces

The most important actuators on the vehicle are the four tyres. In the absence of aerodynamic control surfaces or propulsive devices, the only controllable forces that may be used to accelerate a vehicle are the frictional forces generated in the small contact areas where tyre rubber meets the road surface. The frictional force that can be generated is constrained to be less than or equal to the product of the coefficient of friction μ and the normal load f_z acting at the point of contact. Traction saturation is thus a significant physical phenomenon when considering vehicle limitations. The value of μ can vary considerably depending on the road surface material and prevailing conditions, ranging from less than 0.05 for ice to approximately 1.0 for dry asphalt (Germann et al. 1994). Neglecting aerodynamic lift forces acting on the body of the car, the load acting over the total contact area must be equal to the vehicle weight and so the maximum acceleration that can be achieved is μg [m/s^2] where g is the acceleration due to gravitation.



(a) Longitudinal wheel slip. Deformation of the tyre means that forward speed is not necessarily proportional to angular velocity. The discrepancy between v_x and $r\omega$ is known as longitudinal wheel slip.

(b) Lateral wheel slip α and side slip β. Lateral wheel slip is the angle of the wheel's velocity vector relative to the wheel axis. Side slip is the angle of the vehicle's velocity vector relative to the body axis.

Figure 2.3: Wheel slip.

The maximum achievable lateral force for a wheel is usually slightly less than the maximum longitudinal force, so the limiting value in any direction is described by a tyre-dependent friction ellipse (Figure 2.2). However, given the high uncertainty in μ and the ability for the front wheels to be steered, a reasonable approximation for the tractive force available for the vehicle as a whole, rather than for any particular tyre, is a friction circle.

Tyre deformation, due to vertical loading and strain at the tyre/road interface, means that the speed at which the tread pattern moves across the road does not correspond exactly to the product of the wheel's angular velocity and nominal radius. The discrepancy between these speeds is known as longitudinal slip (Figure 2.3a). The longitudinal slip ratio for each wheel is defined as

$$\lambda = \frac{r\omega - v_x}{v_x} \tag{2.2}$$

where v_x is the vehicle speed in the direction that the tyre is rolling, r is the effective tyre radius and ω is the wheel rotational velocity. Steering the wheels, or applying differential braking, causes them to point in a different direction to the vehicle velocity vector. The result is side slip (Figure 2.3b). The wheel lateral slip angle α is a the angle that the velocity vector of the wheel makes with the wheel's forward axis

$$\alpha = \arctan \frac{v_y}{v_x} \tag{2.3}$$

where v_y is the lateral speed of the tyre. The wheel speeds can be defined in terms of the vehicle

velocities $(\dot{X}, \dot{Y}, \dot{\psi})$ and the steering angle δ for each wheel i.

$$v_{x,i} = + \left(\dot{X} - l_{Y,i}\,\dot{\psi}\right) \cos\delta_i + \left(\dot{Y} + l_{X,i}\,\dot{\psi}\right) \sin\delta_i \tag{2.4}$$

$$v_{y,i} = - \left(\dot{X} - l_{Y,i}\,\dot{\psi}\right) \sin\delta_i + \left(\dot{Y} + l_{X,i}\,\dot{\psi}\right) \cos\delta_i \tag{2.5}$$

where $l_{X,i}$ and $l_{Y,i}$ are the longitudinal and lateral moment arms of each wheel relative to the vehicle centre of mass.

The Magic Formula (Pacejka & Bakker 1993) is widely used to describe the force f generated by a tyre as a function of a slip parameter κ (representing λ or α).

$$f(\kappa) = D \sin\left(C \arctan\left(B\kappa - E\left(B\kappa - \arctan(B\kappa)\right)\right)\right) \tag{2.6}$$

The parameters B, C, D and E are dependent on the type and condition of the tyres fitted to the vehicle. They are also functions of the vertical load, wheel camber angle and the coefficient of friction between the rubber and the road. The Magic Formula can be used to calculate forces resulting from pure longitudinal slip (arising from acceleration or braking without cornering) or lateral slip (arising from cornering without braking) by substituting the appropriate slip parameter into the equation. However, the coefficients B, C, D and E will generally be different in each case.

Tyre forces are likely to saturate during an aggressive manoeuvre. However, for the purposes of controller design it is not necessary to use a highly detailed model of the tyre dynamics. Furthermore, given the wide variation in tyre characteristics and uncertainty in tyre condition at any time, it would be inadvisable for a vehicle dynamics controller to rely too heavily on detailed models of tyre behaviour.

Lateral force is a nearly linear function of α for small slip angles (Gillespie 1992). Thus, as long as the total tractive limit is adhered to, it is often reasonable to approximate the lateral force as

$$f_y \approx \mu C_\alpha \alpha f_z \tag{2.7}$$

where f_z is the vertical load on the wheel and C_α is a tyre-specific parameter known as the lateral stiffness. Load sensitivity will cause a slight reduction in the magnitude of C_α as the vertical load increases. In cases where traction saturation occurs and the tyres operate outside their linear region, it is necessary to obtain a more accurate representation of the tyre characteristics.

The CASCaDE model of the target vehicle, provided by DaimlerChrysler, calculates tyre forces using sets of test data embedded within look-up tables. By running simulations with a range of steering inputs, force-slip data can be generated against which a curve may be fitted. Following the Magic Formula parameter descriptions of Pacejka & Bakker (1993), the coefficients in Table 2.3 were found to give excellent agreement to the lateral slip characteristics of the car. The resulting function is shown in Figure 2.4.

Table 2.3: Magic Formula coefficients obtained by fitting a curve against simulation-generated data of lateral tyre characteristics.

Coefficient	B	C	D	E
Value	18.0	1.0	0.9	-1.0

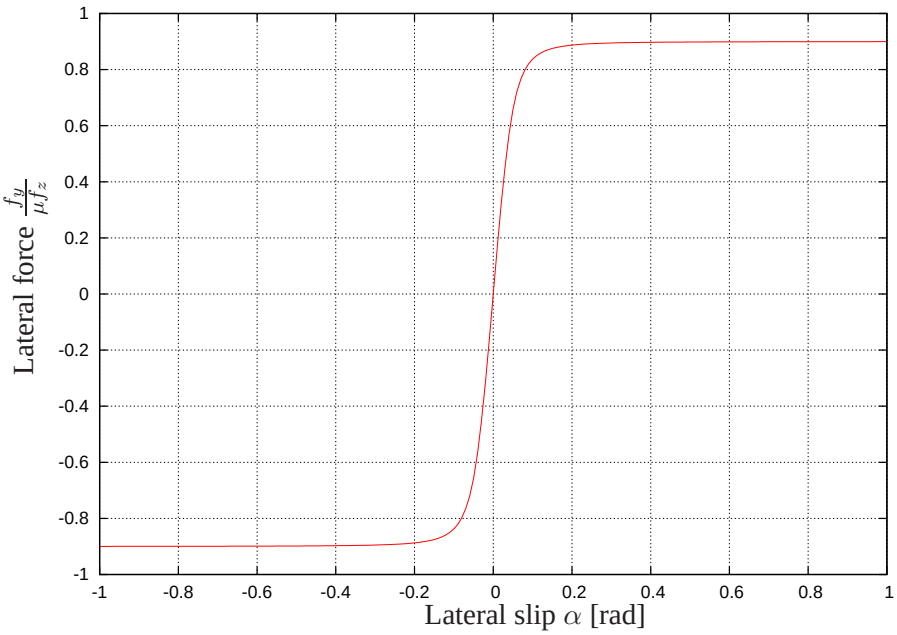


Figure 2.4: Lateral force-slip angle function derived by fitting the Magic Formula to data obtained from simulations with the CASCaDE model.

2.5 Vehicle dynamics

The primary purpose of constructing a design model is to gain insight into the most important aspects of system behaviour. Highly complex models, such as CASCaDE, can make useful predictions, but do not always aid the user in understanding the reasons for the results produced. Understanding is enhanced through simplicity, where appropriate.

2.5.1 Bicycle model

The simplest possible representation of a vehicle is a point mass with forces acting upon it to induce translation. A point mass model cannot account for rotation and is therefore not useful for considering lateral dynamics which are dependent upon yawing motion (rotation about the vertical axis) of the vehicle.

The simplest reasonable model of the vehicle lateral dynamics is the two degree-of-freedom, single-track *bicycle* model, as presented by Milliken & Milliken (1995) among others. This model describes the response of the vehicle lateral velocity $\dot{Y}$ and yaw rate $\dot{\psi}$ to front and rear lateral forces,

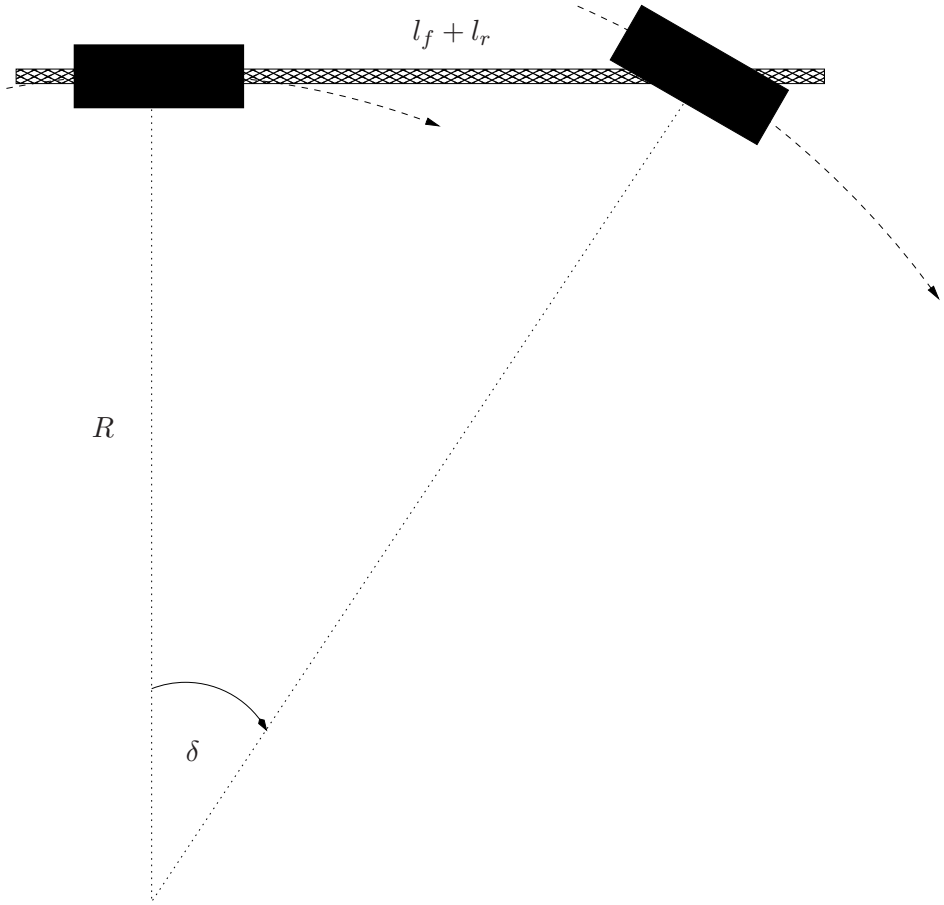


Figure 2.5: Ackermann steering angle. In the absence of lateral slip, the curvature of the vehicle's path is directly related to the front wheel steering angle (Equation (2.9)).

$F_{Y,f}$ and $F_{Y,r}$

$$\begin{bmatrix} m & 0 \\ 0 & J_{ZZ} \end{bmatrix} \begin{pmatrix} \ddot{Y} \\ \ddot{\psi} \end{pmatrix} = \begin{bmatrix} 1 & 1 \\ l_f & l_b \end{bmatrix} \begin{pmatrix} F_{Y,f} \\ F_{Y,r} \end{pmatrix} - \begin{pmatrix} D_Y \\ D_\psi \end{pmatrix} \tag{2.8}$$

where m and J_{ZZ} are the vehicle mass and moment of inertia about the vertical Z-axis; l_f and l_r are the moment arms from the centre of mass to the front and rear axles; $F_{Y,f}$ and $F_{Y,r}$ are the lateral forces, in the body axis system, generated by the front and rear tyres; and D_y and D_ψ are drag terms. The drag terms are frequently neglected. This model considers only the lateral and yaw acceleration; the vehicle longitudinal speed, upon which the tyre forces depend, is usually a constant parameter of the model which therefore represents a vehicle that is cornering without braking.

The bicycle model is particularly useful for understanding the steady-state behaviour of a cornering vehicle. At low speeds, in the absence of slip, it gives rise to the *Ackermann steering angle* which indicates the angle through which the front wheels must be steered if the vehicle is to follow a circular arc with a specified radius of curvature R. From Figure 2.5 it can be seen that the steady-state steering angle is

$$\delta = \arctan \frac{l_f + l_r}{R} \tag{2.9}$$

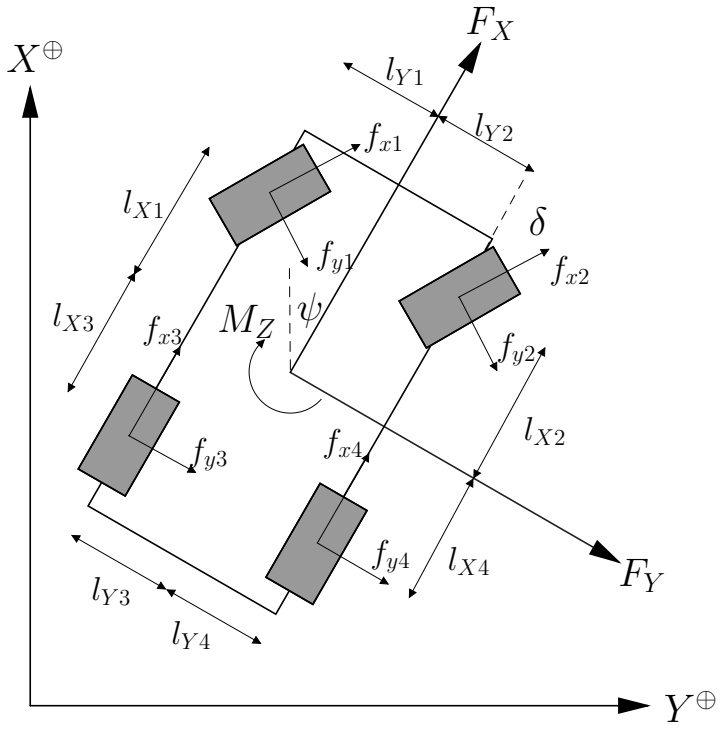


Figure 2.6: Forces and distances in the body and wheel axis systems.

For circular motion, the radius of curvature is the constant of proportionality that relates tangential speed $\dot{X}$ to angular velocity which, if the vehicle is to remain tangential to its path, must equal yaw rate $\dot{\psi}$. The Ackermann steering angle can therefore be expressed as a function of the longitudinal and yaw velocities of the vehicle

$$\delta = \arctan \frac{(l_f + l_r)\,\dot{\psi}}{\dot{X}} \tag{2.10}$$

The bicycle model is commonly formulated using vehicle side-slip $\beta = \arctan(\dot{Y}/\dot{X}) \approx \dot{Y}/\dot{X}$ instead of lateral velocity. Used in conjunction with a linear tyre model, in which lateral tyre force is proportional to slip angle α, this leads to a linear model of the vehicle dynamics.

Although useful for understanding the basic behaviour of a turning vehicle, the bicycle model and Ackermann steering angle cannot capture the effects of differential braking which can contribute to the yawing moment acting upon the vehicle. To represent independent brake operation, it is necessary to consider a two-track model.

2.5.2 Two-track model

For each wheel, $i \in [1, 4]$, the longitudinal and lateral forces $f_{x,i}$ and $f_{y,i}$ in the wheel axis system (Figure 2.6) can be resolved into contributions to the longitudinal and lateral forces acting on the vehicle in the vehicle body axis system

$$\begin{pmatrix} F_{X,i} \\ F_{Y,i} \end{pmatrix} = \begin{pmatrix} +\cos\delta_i & , & -\sin\delta_i \\ +\sin\delta_i & , & +\cos\delta_i \end{pmatrix} \begin{pmatrix} f_{x,i} \\ f_{y,i} \end{pmatrix} \tag{2.11}$$

where $f_{x,i}$ and $f_{y,i}$ are the longitudinal and lateral forces in the tyre axis system and δ_i is the wheel steering angle. The vehicle has no rear wheel steering and the steer-by-wire system permits only a single steering angle δ to be set for both front wheels. Thus the steering angles for each wheel are $\delta_1 = \delta_2 = \delta$ [rad] for the front wheels and $\delta_3 = \delta_4 = 0$ for the rear. The body-oriented forces $F_{X,i}$ and $F_{Y,i}$ can in turn be transformed into component forces $F_{X^{\oplus},i}$ and $F_{Y^{\oplus},i}$ in the fixed Earth axis system and contributions to the yawing moment about the vehicle centre of mass $M_{Z,i}$

$$\begin{pmatrix} F_{X^{\oplus},i} \\ F_{Y^{\oplus},i} \\ M_{Z,i} \end{pmatrix} = \begin{pmatrix} +\cos\psi & , & -\sin\psi \\ +\sin\psi & , & +\cos\psi \\ -l_{Y,i} & , & +l_{X,i} \end{pmatrix} \begin{pmatrix} F_{X,i} \\ F_{Y,i} \end{pmatrix} \tag{2.12}$$

where $l_{X,i}$ and $l_{Y,i}$ are the co-ordinates of the wheel in the body axis system. Applying Newton's Second Law (1687) the vehicle body longitudinal, lateral and yaw accelerations are

$$\ddot{X} = \frac{1}{m}\sum_{i=1}^{4} F_{X,i} \qquad \ddot{Y} = \frac{1}{m}\sum_{i=1}^{4} F_{Y,i} \qquad \ddot{\psi} = \frac{1}{J_{ZZ}}\sum_{i=1}^{4} M_{z_i} \tag{2.13}$$

in the moving body axis system, and

$$\ddot{X}^{\oplus} = \frac{1}{m}\sum_{i=1}^{4} F_{X^{\oplus},i} \qquad \ddot{Y}^{\oplus} = \frac{1}{m}\sum_{i=1}^{4} F_{Y^{\oplus},i} \qquad \ddot{\psi}^{\oplus} = \frac{1}{J_{ZZ}}\sum_{i=1}^{4} M_{z_i} \tag{2.14}$$

in the fixed Earth axis system. Substituting the resultant forces and moments from Equations (2.11) and (2.12) then yields the vehicle accelerations as functions of the individual tyre forces, steering angles and, for the fixed Earth axis, vehicle heading angle.

Vertical dynamics

The traction available to each tyre is proportional to the normal, i.e. vertical, load upon the wheel. Hence the vertical dynamics of the vehicle cannot be neglected entirely when considering lateral vehicle dynamics. Redistribution of vehicle weight affects the maximum forces that can be generated by each tyre.

When lateral tyre forces are used to induce a yawing moment on a car, they also produce a rolling moment. Similarly, longitudinal forces, used to induce longitudinal acceleration, also produce a pitching moment. On a sprung body, these moments and forces cause roll, pitch and heave acceleration of the sprung mass. Pitch and roll of the body can be modelled by considering the torques acting about roll and pitch axes where the vehicle is attached to the chassis (Gillespie 1992). The rotation and heave of the vehicle are damped by the suspension system and the vertical dynamics of the pneumatic tyres; complex systems that are highly vehicle dependent.

However, for analysing lateral dynamics, it is not necessary to know the precise orientation of the car body relative to the chassis; all that matters is the distribution of weight across the wheels. This

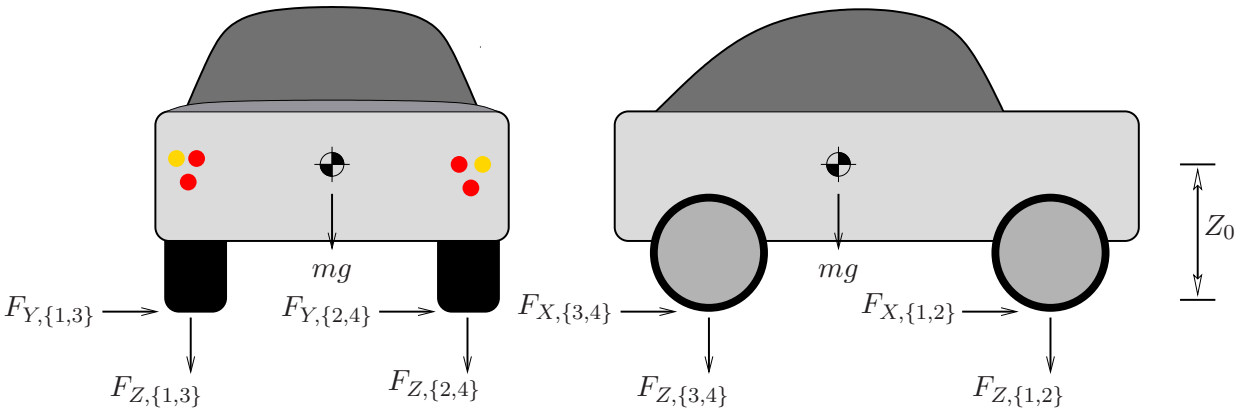


Figure 2.7: Rolling and pitching moments.

can be obtained by considering a quasi-static system of forces. Figure 2.7 shows the longitudinal, lateral and vertical forces acting on the tyres of a rigid, unsprung car body. The height of the centre of mass is Z_0 [m] above the ground and the weight of the car acts through it. The rolling and pitching moments about the centre of mass are

$$M_X = \sum_{i=1}^{4} -Z_0 F_{Y,i} - Y_i F_{Z,i} \qquad M_Y = \sum_{i=1}^{4} -Z_0 F_{X,i} - X_i F_{Z,i} \tag{2.15}$$

Meanwhile, the net vertical force acting on the vehicle is

$$m\ddot{Z} = mg - \sum_{i=1}^{4} F_{Z,i} \tag{2.16}$$

In equilibrium, the terms on either side of Equations (2.15) and (2.16) sum to zero, as the moments and forces are counter-balanced by weight redistribution. While all four wheels remain firmly in contact with the ground, the vertical loading of the wheels is a statically-indeterminate problem. Weight distribution characteristics may be fine-tuned for a particular vehicle, through the use of anti-roll bars or active body control. However, a least norm solution will capture the general behaviour of the vehicle, hence the vector of vertical forces $\mathbf{F}_Z$ can be obtained as a function of the net longitudinal and lateral forces acting on the vehicle and its basic geometry.

$$\mathbf{F}_Z = \min_{\mathbf{F}_Z} \left\| \begin{pmatrix} 1 & 1 & 1 & 1 \\ Y_1 & Y_2 & Y_3 & Y_4 \\ X_1 & X_2 & X_3 & X_4 \end{pmatrix} \mathbf{F}_Z - \begin{pmatrix} mg \\ -Z_0 F_Y \\ -Z_0 F_X \end{pmatrix} \right\|_2 \tag{2.17}$$

This weight distribution can then be used by the tyre model to refine longitudinal and/or lateral force calculations, for example, by substituting the elements of $\mathbf{F}_Z$ corresponding to each wheel for f_z in Equation (2.7).

2.6 Vehicle-specific model

The governing equations described in the previous section are generic; they may be applied to any conventional car. Vehicle longitudinal, lateral and yaw accelerations (Equations (2.13) and (2.14)) are defined in terms of resultant forces and moments (Equations (2.11) and (2.12)) which are themselves functions of vehicle geometry, tyre forces, wheel steering angles and the vehicle heading angle. The tyre forces are dependent upon vertical load, a quasi-static model for which is defined in Equation (2.17), and complex nonlinear functions of wheel and vehicle velocities, as outlined in Section 2.4.

For the target vehicle – "TS" – there are five controllable inputs: front wheel steering angle, which is constrained to be identical on each side; and four independent longitudinal brake forces, which are produced on demand by the anti-lock braking system.

Applying the steering angle constraint ($\delta_1 = \delta_2 = \delta, \delta_3 = \delta_4 = 0$), Equation (2.11) becomes

$$\begin{pmatrix} F_{X,i} \\ F_{Y,i} \end{pmatrix} = \begin{cases} \begin{pmatrix} +\cos\delta & , & -\sin\delta \\ +\sin\delta & , & +\cos\delta \end{pmatrix} \begin{pmatrix} f_{x,i} \\ f_{y,i} \end{pmatrix} & \text{for the front wheels,} \quad i = \{1,2\} \\ \begin{pmatrix} f_{x,i} \\ f_{y,i} \end{pmatrix} & \text{for the rear wheels,} \quad i = \{3,4\} \end{cases} \tag{2.18}$$

As the steering angle and longitudinal forces are controllable, only the lateral tyre forces require further calculation; for these, Pacejka's Magic Formula (Equation (2.6)) is used with the coefficients specified in Table 2.3. Using the Magic Formula, the lateral slip angle α (Equation (2.3)) enters the model. This is a function of the wheel speeds which are themselves functions of the vehicle velocities (Equations (2.4) and (2.5)). Hence the vehicle accelerations are functions of: the vehicle longitudinal, lateral and yaw velocities; and the five controllable inputs: front wheel steering angle and the four brake forces.

2.7 Velocity-based linearisation

The vehicle model described in Section 2.6 is highly nonlinear. It includes several trigonometric terms and products of system inputs.

For control design, it is often useful to linearise models to obtain representations that are amenable to traditional techniques of linear control analysis and design. Linearised models are usually obtained at particular equilibrium conditions (operating points) and predict the approximate behaviour of the system in response to small perturbations. In most cases, such models provide an adequate representation of the system behaviour close to these operating points while the system inputs and disturbances remain sufficiently small. For systems that operate close to a manifold of equilibria, it is often possible to generate a model of sufficient validity across the entire manifold by combining a family of linearised models; a technique which lends itself to the gain-scheduling method of control

(Rugh 1991, Shamma & Athans 1990).

A car performing an emergency evasive manoeuvre close to its physical limits will necessarily operate far from equilibrium and be subject to large steering and braking forces. As such, linear models appropriate for small perturbations about equilibria are likely to be of questionable validity and have little predictive power (Johansen, Hunt, Gawthrop & Fritz 1998).

The technique of velocity-based linearisation (Leith & Leithead 1998*a*,*b*) can provide a means of generating linear models with global validity. The technique relies on partial differentiation of the state vector with respect to time to obtain a local linearisation. Crucially, this differentiation is not restricted to operating conditions that lie on manifolds of equilibria. The suitability of the method for generating useful models of high-performance vehicles performing aggressive manoeuvres has been demonstrated with an example using an agile missile (Leith, Tsourdos, White & Leithead 2001).

The two-track nonlinear model developed in Section 2.5.2 (Equations (2.11) to (2.13)) describes the vehicle acceleration as a nonlinear function of velocity, steering input and brake inputs, which can be expressed as

$$\dot{\mathbf{v}} = \mathrm{h}(\mathbf{v}, \delta, \mathbf{f}) \tag{2.19}$$

where vector $\mathbf{v}$ is the velocity vector $(\dot{X}, \dot{Y}, \dot{\psi})^T$, δ is the front wheel steering angle and $\mathbf{f}$ is the vector of brake forces. Differentiating the acceleration vector with respect to time yields a linear model

$$\ddot{\mathbf{v}} \approx \frac{\partial \mathrm{h}}{\partial \mathbf{v}}\dot{\mathbf{v}} + \frac{\partial \mathrm{h}}{\partial \delta}\dot{\delta} + \frac{\partial \mathrm{h}}{\partial \mathbf{f}}\dot{\mathbf{f}} \tag{2.20}$$

Each of the partial derivative terms in Equation (2.20) are functions of the vehicle velocity vector $\mathbf{v}$ and the front wheel steering angle δ. Defining a scheduling vector $\boldsymbol{\rho}$ comprising these terms, a scheduled family of models can therefore be defined in terms of an acceleration vector $\mathbf{w} = \dot{\mathbf{v}}(\boldsymbol{\rho})$ by

$$\dot{\mathbf{w}} = \underline{\mathrm{A}}(\boldsymbol{\rho})\,\mathbf{w} + \underline{\mathrm{B}}_{\delta}(\boldsymbol{\rho})\,\dot{\delta} + \underline{\mathrm{B}}_{f}(\boldsymbol{\rho})\,\dot{\mathbf{f}} \tag{2.21}$$

where $\underline{\mathrm{A}}(\boldsymbol{\rho}) = \frac{\partial \mathrm{h}}{\partial \mathbf{v}}$, $\underline{\mathrm{B}}_{\delta}(\boldsymbol{\rho}) = \frac{\partial \mathrm{h}}{\partial \delta}$, $\underline{\mathrm{B}}_{f}(\boldsymbol{\rho}) = \frac{\partial \mathrm{h}}{\partial \mathbf{f}}$. In other words, the linearised state and input matrices are the partial derivatives of the acceleration vector with respect to the vehicle velocities and inputs. Written explicitly, these matrices are

$$\underline{\mathrm{A}} = \begin{pmatrix} \frac{\partial \ddot{X}}{\partial \dot{X}} & \frac{\partial \ddot{X}}{\partial \dot{Y}} & \frac{\partial \ddot{X}}{\partial \dot{\psi}} \\ \frac{\partial \ddot{Y}}{\partial \dot{X}} & \frac{\partial \ddot{Y}}{\partial \dot{Y}} & \frac{\partial \ddot{Y}}{\partial \dot{\psi}} \\ \frac{\partial \ddot{\psi}}{\partial \dot{X}} & \frac{\partial \ddot{\psi}}{\partial \dot{Y}} & \frac{\partial \ddot{\psi}}{\partial \dot{\psi}} \end{pmatrix} \qquad \underline{\mathrm{B}}_{f} = \begin{pmatrix} \frac{\partial \ddot{X}}{\partial f_{x,1}} & \frac{\partial \ddot{X}}{\partial f_{x,2}} & \frac{\partial \ddot{X}}{\partial f_{x,3}} & \frac{\partial \ddot{X}}{\partial f_{x,4}} \\ \frac{\partial \ddot{Y}}{\partial f_{x,1}} & \frac{\partial \ddot{Y}}{\partial f_{x,2}} & \frac{\partial \ddot{Y}}{\partial f_{x,3}} & \frac{\partial \ddot{Y}}{\partial f_{x,4}} \\ \frac{\partial \ddot{\psi}}{\partial f_{x,1}} & \frac{\partial \ddot{\psi}}{\partial f_{x,2}} & \frac{\partial \ddot{\psi}}{\partial f_{x,3}} & \frac{\partial \ddot{\psi}}{\partial f_{x,4}} \end{pmatrix} \qquad \underline{\mathrm{B}}_{\delta} = \begin{pmatrix} \frac{\partial \ddot{X}}{\partial \delta} \\ \frac{\partial \ddot{Y}}{\partial \delta} \\ \frac{\partial \ddot{\psi}}{\partial \delta} \end{pmatrix} \tag{2.22}$$

To make use of this model, it is necessary to expand the terms in each of the three matrices.

Starting with Equation (2.13) and the chain rule, the elements of $\underline{\mathrm{A}}$ can be expressed as

$$\frac{\partial \ddot{X}}{\partial \dot{X}} = \sum_{i=1}^{4} \frac{\partial \ddot{X}}{\partial f_{y,i}} \frac{\partial f_{y,i}}{\partial \alpha_i} \left(\frac{\partial \alpha_i}{\partial v_{x,i}} \frac{\partial v_{x,i}}{\partial \dot{X}} + \frac{\partial \alpha_i}{\partial v_{y,i}} \frac{\partial v_{y,i}}{\partial \dot{X}} \right) \tag{2.23}$$

$$\frac{\partial \ddot{X}}{\partial \dot{Y}} = \sum_{i=1}^{4} \frac{\partial \ddot{X}}{\partial f_{y,i}} \frac{\partial f_{y,i}}{\partial \alpha_i} \left(\frac{\partial \alpha_i}{\partial v_{x,i}} \frac{\partial v_{x,i}}{\partial \dot{Y}} + \frac{\partial \alpha_i}{\partial v_{y,i}} \frac{\partial v_{y,i}}{\partial \dot{Y}} \right) \tag{2.24}$$

$$\frac{\partial \ddot{X}}{\partial \dot{\psi}} = \sum_{i=1}^{4} \frac{\partial \ddot{X}}{\partial f_{y,i}} \frac{\partial f_{y,i}}{\partial \alpha_i} \left(\frac{\partial \alpha_i}{\partial v_{x,i}} \frac{\partial v_{x,i}}{\partial \dot{\psi}} + \frac{\partial \alpha_i}{\partial v_{y,i}} \frac{\partial v_{y,i}}{\partial \dot{\psi}} \right) \tag{2.25}$$

$$\frac{\partial \ddot{Y}}{\partial \dot{X}} = \sum_{i=1}^{4} \frac{\partial \ddot{Y}}{\partial f_{y,i}} \frac{\partial f_{y,i}}{\partial \alpha_i} \left(\frac{\partial \alpha_i}{\partial v_{x,i}} \frac{\partial v_{x,i}}{\partial \dot{X}} + \frac{\partial \alpha_i}{\partial v_{y,i}} \frac{\partial v_{y,i}}{\partial \dot{X}} \right) \tag{2.26}$$

$$\frac{\partial \ddot{Y}}{\partial \dot{Y}} = \sum_{i=1}^{4} \frac{\partial \ddot{Y}}{\partial f_{y,i}} \frac{\partial f_{y,i}}{\partial \alpha_i} \left(\frac{\partial \alpha_i}{\partial v_{x,i}} \frac{\partial v_{x,i}}{\partial \dot{Y}} + \frac{\partial \alpha_i}{\partial v_{y,i}} \frac{\partial v_{y,i}}{\partial \dot{Y}} \right) \tag{2.27}$$

$$\frac{\partial \ddot{Y}}{\partial \dot{\psi}} = \sum_{i=1}^{4} \frac{\partial \ddot{Y}}{\partial f_{y,i}} \frac{\partial f_{y,i}}{\partial \alpha_i} \left(\frac{\partial \alpha_i}{\partial v_{x,i}} \frac{\partial v_{x,i}}{\partial \dot{\psi}} + \frac{\partial \alpha_i}{\partial v_{y,i}} \frac{\partial v_{y,i}}{\partial \dot{\psi}} \right) \tag{2.28}$$

$$\frac{\partial \ddot{\psi}}{\partial \dot{X}} = \sum_{i=1}^{4} \frac{\partial \ddot{\psi}}{\partial f_{y,i}} \frac{\partial f_{y,i}}{\partial \alpha_i} \left(\frac{\partial \alpha_i}{\partial v_{x,i}} \frac{\partial v_{x,i}}{\partial \dot{X}} + \frac{\partial \alpha_i}{\partial v_{y,i}} \frac{\partial v_{y,i}}{\partial \dot{X}} \right) \tag{2.29}$$

$$\frac{\partial \ddot{\psi}}{\partial \dot{Y}} = \sum_{i=1}^{4} \frac{\partial \ddot{\psi}}{\partial f_{y,i}} \frac{\partial f_{y,i}}{\partial \alpha_i} \left(\frac{\partial \alpha_i}{\partial v_{x,i}} \frac{\partial v_{x,i}}{\partial \dot{Y}} + \frac{\partial \alpha_i}{\partial v_{y,i}} \frac{\partial v_{y,i}}{\partial \dot{Y}} \right) \tag{2.30}$$

$$\frac{\partial \ddot{\psi}}{\partial \dot{\psi}} = \sum_{i=1}^{4} \frac{\partial \ddot{\psi}}{\partial f_{y,i}} \frac{\partial f_{y,i}}{\partial \alpha_i} \left(\frac{\partial \alpha_i}{\partial v_{x,i}} \frac{\partial v_{x,i}}{\partial \dot{\psi}} + \frac{\partial \alpha_i}{\partial v_{y,i}} \frac{\partial v_{y,i}}{\partial \dot{\psi}} \right) \tag{2.31}$$

Similarly the elements of $\underline{\mathrm{B}}_f$ can be expanded as

$$\frac{\partial \ddot{X}}{\partial f_{x,i}} = \frac{1}{m} \cos \delta_i \tag{2.32}$$

$$\frac{\partial \ddot{Y}}{\partial f_{x,i}} = \frac{1}{m} \sin \delta_i \tag{2.33}$$

$$\frac{\partial \ddot{\psi}}{\partial f_{x,i}} = \frac{1}{J_{zz}} \left(-Y_i \cos \delta_i + X_i \sin \delta_i \right) \tag{2.34}$$

and the elements of $\underline{\mathrm{B}}_\delta$ can be written as

$$\frac{\partial \ddot{X}}{\partial \delta_i} = \frac{\partial \ddot{X}}{\partial f_{y,i}} \frac{\partial f_{y,i}}{\partial \alpha_i} \left(\frac{\partial \alpha_i}{\partial v_{x,i}} \frac{\partial v_{x,i}}{\partial \delta_i} + \frac{\partial \alpha_i}{\partial v_{y,i}} \frac{\partial v_{y,i}}{\partial \delta_i} \right) \tag{2.35}$$

$$\frac{\partial \ddot{Y}}{\partial \delta_i} = \frac{\partial \ddot{Y}}{\partial f_{y,i}} \frac{\partial f_{y,i}}{\partial \alpha_i} \left(\frac{\partial \alpha_i}{\partial v_{x,i}} \frac{\partial v_{x,i}}{\partial \delta_i} + \frac{\partial \alpha_i}{\partial v_{y,i}} \frac{\partial v_{y,i}}{\partial \delta_i} \right) \tag{2.36}$$

$$\frac{\partial \ddot{\psi}}{\partial \delta_i} = \frac{\partial \ddot{\psi}}{\partial f_{y,i}} \frac{\partial f_{y,i}}{\partial \alpha_i} \left(\frac{\partial \alpha_i}{\partial v_{x,i}} \frac{\partial v_{x,i}}{\partial \delta_i} + \frac{\partial \alpha_i}{\partial v_{y,i}} \frac{\partial v_{y,i}}{\partial \delta_i} \right) \tag{2.37}$$

Each of the remaining partial derivatives on the right hand side require further expansion. The dependence of body accelerations on lateral tyre forces can be expanded, using the equations of Section 2.5.2, to become

$$\frac{\partial \ddot{X}}{\partial f_{y,i}} = \frac{-1}{m} \sin \delta_i \tag{2.38}$$

$$\frac{\partial \ddot{Y}}{\partial f_{y,i}} = \frac{1}{m} \cos \delta_i \tag{2.39}$$

$$\frac{\partial \ddot{\psi}}{\partial f_{y,i}} = \frac{1}{J_{zz}} \left(X_i \cos \delta_i + Y_i \sin \delta_i \right) \tag{2.40}$$

By differentiating the Magic Formula, the dependence of lateral tyre forces on wheel side-slip are

$$\frac{\partial f_{y,i}}{\partial \alpha_i} = -B_y D_y \frac{\partial \chi_3}{\partial \chi_1} \cos \chi_3 \tag{2.41}$$

where

$$\chi_1 = B_y \alpha_i \qquad \chi_2 = E_y \left(\chi_1 - \arctan \chi_1 \right) \qquad \chi_3 = C_y \arctan \left(\chi_1 - \chi_2 \right)$$

$$\frac{\partial \chi_2}{\partial \chi_1} = E_y \left(1 - \frac{1}{1 + \chi_1^2} \right) \qquad \frac{\partial \chi_3}{\partial \chi_1} = \frac{C_y}{1 + \left(\chi_1 - \chi_2 \right)^2} \left(1 - \frac{\partial \chi_2}{\partial \chi_1} \right)$$

Partial differentiation of Equation (2.3) gives the dependence of wheel side-slip on wheel speeds

$$\frac{\partial \alpha_i}{\partial v_{x,i}} = \frac{-v_{y,i}}{v_{x,i}^2 + v_{y,i}^2} \tag{2.42}$$

$$\frac{\partial \alpha_i}{\partial v_{y,i}} = \frac{+v_{x,i}}{v_{x,i}^2 + v_{y,i}^2} \tag{2.43}$$

Dependence of wheel speeds on body velocity is obtained by partially differentiating Equations (2.4) and (2.5))

$$\frac{\partial v_{x,i}}{\partial \dot{X}} = + \cos \delta_i \tag{2.44}$$

$$\frac{\partial v_{y,i}}{\partial \dot{X}} = - \sin \delta_i \tag{2.45}$$

$$\frac{\partial v_{x,i}}{\partial \dot{Y}} = + \sin \delta_i \tag{2.46}$$

$$\frac{\partial v_{y,i}}{\partial \dot{Y}} = + \cos \delta_i \tag{2.47}$$

$$\frac{\partial v_{x,i}}{\partial \dot{\psi}} = -Y_i \cos \delta_i + X_i \sin \delta_i \tag{2.48}$$

$$\frac{\partial v_{y,i}}{\partial \dot{\psi}} = +X_i \cos \delta_i + Y_i \sin \delta_i \tag{2.49}$$

and the dependence of wheel speeds on steering angle is obtained in the same way

$$\frac{\partial v_{x,i}}{\partial \delta_i} = + \left(\dot{Y} + X_i\dot{\psi}\right)\cos\delta_i - \left(\dot{X} - Y_i\dot{\psi}\right)\sin\delta_i \tag{2.50}$$

$$\frac{\partial v_{y,i}}{\partial \delta_i} = - \left(\dot{X} - Y_i\dot{\psi}\right)\cos\delta_i - \left(\dot{Y} + X_i\dot{\psi}\right)\sin\delta_i \tag{2.51}$$

2.8 Design model implementation: MexCar

The preceding sections describe non-linear and linearised models of the target vehicle dynamics. These models were implemented in software to aid analysis and provide a basis for simulating the vehicle behaviour. The following functional software requirements were identified for this controller-design model:

- to implement the nonlinear model of Section 2.5.2;
- to perform velocity-based linearisation at any operating condition;
- to be capable of standalone simulation; and
- to provide interfaces for Matlab (Moler 1988) and GNU Octave (Eaton 2002).

The following non-functional requirements were also identified as being desirable:

- to be fast to run; and
- to be maintainable.

An object-oriented model was coded in C++. The base class, *Car*, implements the nonlinear model. Physical properties of the car, e.g. mass, moment of inertia, geometry, etc., are hard-coded in the model. The *Car* object allows the following parameters to be initialised or updated at run-time by the user:

- body velocity $(\dot{X}, \dot{Y}, \dot{\psi})$ and acceleration $(\ddot{X}, \ddot{Y}, \ddot{\psi})$;
- friction coefficient μ;
- longitudinal, lateral and vertical tyre forces $f_{\{x,y,z\},i} \qquad \forall i \in [1,4]$;
- front wheel steering angle δ;
- wheel slip angles $\alpha_i \qquad \forall i \in [1,4]$; and
- wheel speeds $v_{x,i} \qquad \forall i \in [1,4]$.

A forward Euler integration routine allows the model to be exercised as a simulation, with the *Car* object storing and updating the vehicle states. The following outputs are available, in addition to the inputs:

- body displacement (X, Y, ψ); and
- lateral tyre speeds $v_{y,i} \qquad \forall i \in [1,4]$.

Symbolic partial differentiation (see Section 2.7) of the nonlinear model was undertaken manually and checked using the symbolic algebra tools Reduce (Hearn 1982) and Maxima (Max 2007). The resulting expressions were coded in a derived class, *LinearisableCar*, which calculates the velocity-based linearisation of the *Car* in its current state and return the linearised state and input matrices (Section 2.7).

Wrapper functions were created to allow the model to be compiled as dynamic link libraries for use in Matlab (as a "mex" file) and GNU Octave (as a "DLD" file) using one of two interfaces: *MexCar()* or *OctCar()*. This model is henceforth referred to as the MexCar model. The model implementation is shown in Appendix A, Section A.1.

2.9 Verification of MexCar

Each of the elements of the MexCar model was tested in isolation during development (unit testing). Verification of the overall model was performed by using the model to simulate manoeuvres for which the output response could be readily predicted.

A mission was defined using piecewise constant inputs, as follows:

accelerate from rest: apply an accelerating tyre force of magnitude $f_{x,i} = mg$ [N] on each wheel for 5 seconds with all wheels pointing straight ahead ($\delta = 0$ [rad]);

describe a circle: remove the longitudinal tyre forces and steer the front wheels $\pi/16$ [rad] (11.25 [deg]) to the right for 5 seconds;

brake in a straight line: remove the steering input ($\delta = 0$) and apply braking tyre forces of magnitude $f_{x,i} = -mg/4$ [N] on each wheel for 5 seconds;

brake in a turn: without changing the braking forces, again steer the front wheels $\pi/16$ [rad] to the right for 5 seconds; and

brake in a straight line: re-centre the wheels ($\delta = 0$) while continuing to apply the braking forces for a further 10 seconds.

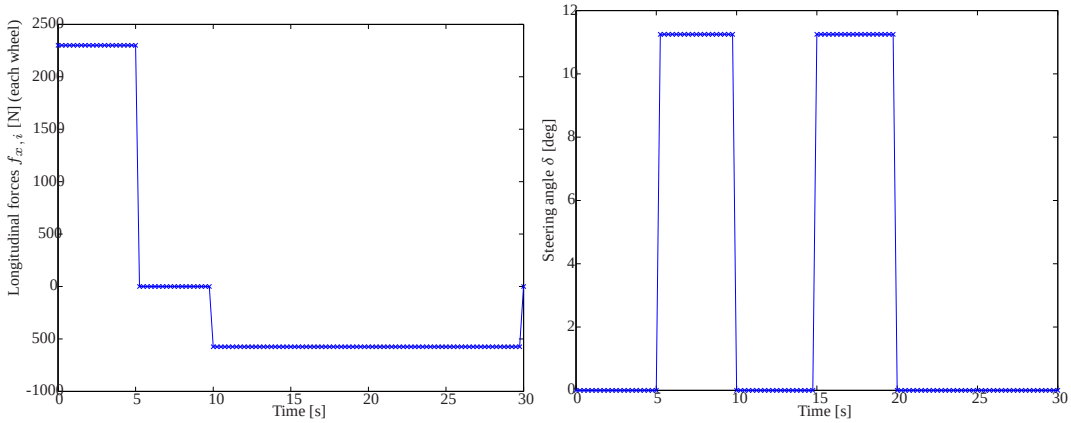
The mission is summarised in Table 2.4.

Table 2.4: Verification mission summary

Time [s]	$f_{x,i}$ *[N]*	δ *[rad]*	*expected behaviour*
0 – 5	$+mg$	0	Car accelerates at 4 [m/s^2] to reach 20 [m/s]
5 – 10	0	$+\pi/16$	Car describes a circle (constant speed and yaw rate)
10 – 15	$-mg/4$	0	Car decelerates at 1 [m/s^2] to reach 15 [m/s]
15 – 20	$-mg/4$	$+\pi/16$	Car decelerates at 1 [m/s^2] to reach 10 [m/s] with varying yaw rate
20 – 30	$-mg/4$	0	Car decelerates at 1 [m/s^2] to rest.

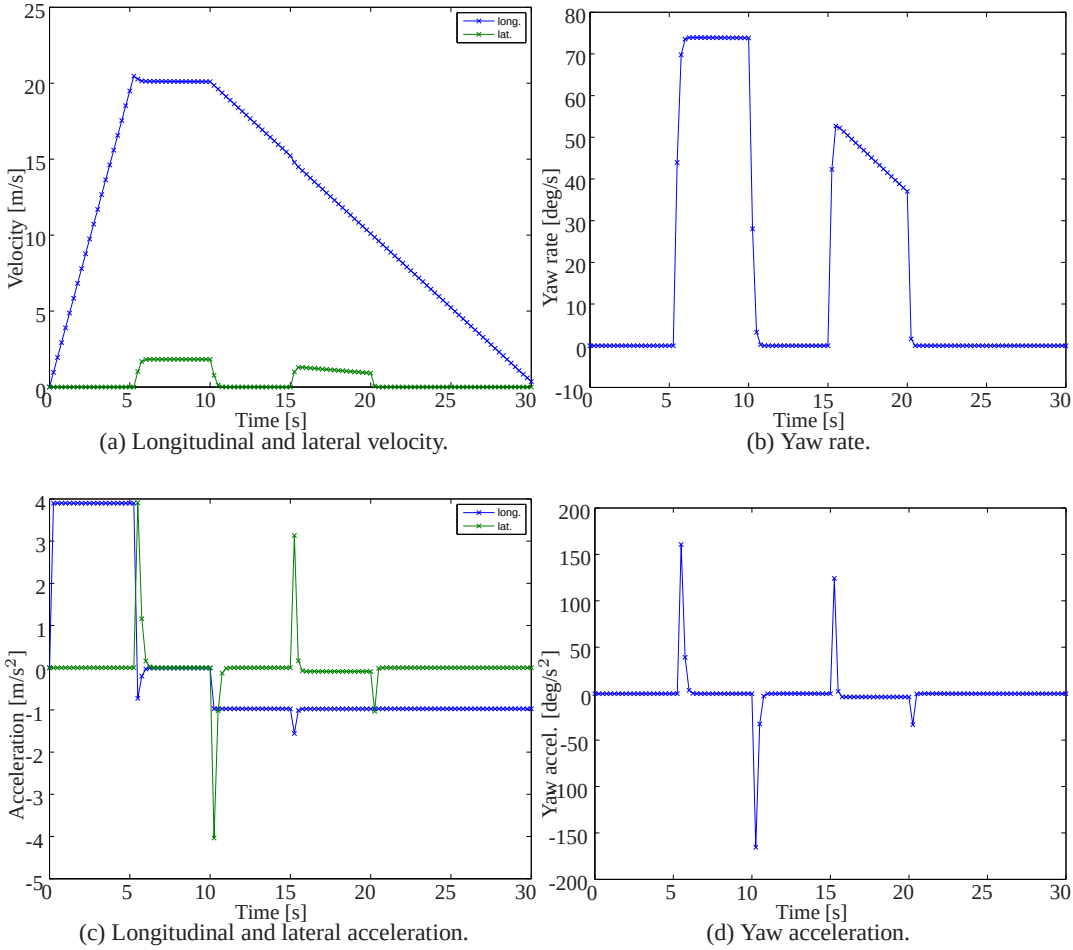
The MexCar model was used within Matlab with a simulation step length of 0.25 [s]. The inputs are shown in Figure 2.8. The output response is shown in Figures 2.9 (velocities and accelerations) and 2.10 (trajectory).

The model is seen to behave as predicted. For the first 5 seconds, the car accelerates with constant acceleration $\ddot{X} = 4$ [m/s^2] reaching a top speed of 20 [m/s]. During the next 5 seconds, the acceleration drops to 0 and the speed remains constant while the vehicle describes a circle (constant yaw rate). From thereon in, the car slows at a constant rate, except for negligible blips when the steering angle changes at $t = 15$ [s] and $t = 20$ [s], until it reaches rest after a total drive-time of 30 seconds.



(a) Longitudinal tyre force f_x, i (each wheel). (b) Front wheel steering angle δ.

Figure 2.8: MexCar verification. Piecewise constant longitudinal tyre forces and steering angles were applied to the MexCar model.



(a) Longitudinal and lateral velocity. (b) Yaw rate.

(c) Longitudinal and lateral acceleration. (d) Yaw acceleration.

Figure 2.9: MexCar verification. The model was verified by simulating a series of manoeuvres. Here the vehicle is accelerated in a straight line from rest, follows a circular trajectory at constant speed with constant steering wheel angle, reduces speed with the front wheels pointing straight ahead, brakes during a second turn and then decelerates to rest in a final straight.

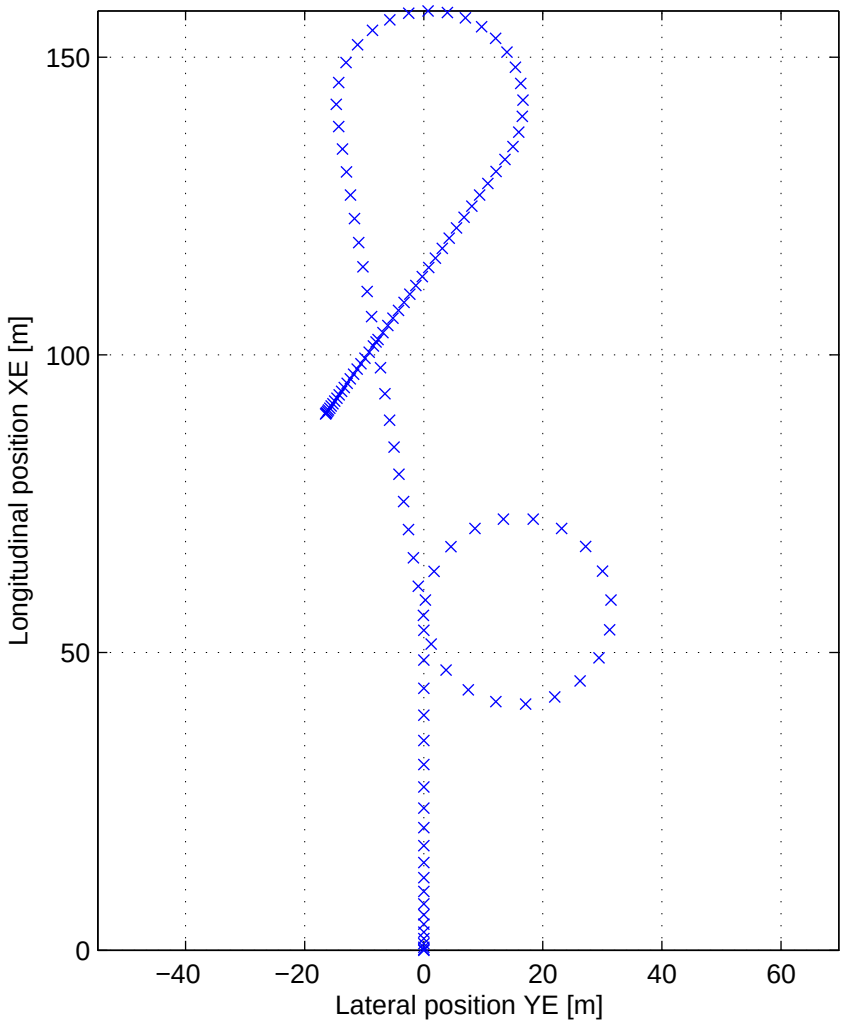


Figure 2.10: MexCar verification. The trajectory described by the simulated vehicle is as expected for the system inputs.

2.10 Conclusion

Equations of motion for the longitudinal, lateral and yaw dynamics of a car have been presented along with a tyre model. From these, a two-track non-linear model has been developed. The model specifies the vehicle accelerations as functions of the vehicle velocities and controllable inputs: the front wheel steering angle and four brake forces. A velocity-based linearisation of the non-linear model has been obtained using symbolic differentiation.

The non-linear design model has been implemented in software, known as MexCar. This model is capable of performing velocity-based linearisations at any operating condition, whether or not in equilibrium, to obtain locally-valid state and input matrices. Interfaces to matrix algebra tools Matlab and GNU Octave allow the model to be used within simulation environments that permit analysis and visualisation of the system behaviour.

The MexCar model forms an essential component of the controller design process, described in Chapter 4.

Chapter 3

Feasible trajectory generation

μή μου τούς κύκλουζ τάραττε.
(*noli turbare circulos meos).*

— Archimedes

Models of the vehicle dynamics have been introduced in Chapter 2. This chapter develops methods for obtaining reference trajectories through an obstacle course such as the ISO 3888 test track described below. The trajectories are used by a controller developed in Chapter 4 to cause the target vehicle to perform specified manoeuvres.

Trajectory generation is a well-studied problem in the fields of aerospace and robotic engineering (e.g. Betts (1998), Chakravarthy & Ghose (1998), Dubins (1957), Oberle (1990), Van Nieuwstadt & Murray (1998)). However, each of these applications differs significantly from automotive considerations. Unlike aircraft, cars operate in very cluttered environments where trajectories are tightly constrained. Cars are also frequently driven close to their physical limits, which is often the reason that aggressive evasive manoeuvres are necessary.

For robotic trajectory planning, the dynamics of the robot itself are usually not a significant factor. Planning is frequently a problem of finding an efficient unblocked route to a target rather than a consideration of robot dynamic equations. In contrast, cars routinely travel at high speed in tightly constrained environments. The stopping distance is generally large compared to the dimensions of the vehicle, while the channels in which the car is constrained to remain are usually little wider than the breadth of the vehicle and substantially narrower than its length. Thus the orientation of a car is an integral part of generating a suitable trajectory and the vehicle dynamics strongly influence the feasibility of following any path.

Two methods of calculating a feasible trajectory for the vehicle to follow are described in this chapter. The first method, previously outlined by Bevan, Gollee & O'Reilly (2007*a*) and described

in Section 3.2, produces a trajectory by application of simple geometry using circular arcs with the minimum radius of curvature achievable by the car. The second method, by the same authors (2007*b*) and described in Section 3.3, uses convex optimisation to find an optimal trajectory that minimises yaw acceleration. By formulating a convex specification for the trajectory generation problem, it is possible to use specialised, highly efficient convex solution algorithms which require fewer computational resources than more general optimisation solvers.

3.1 Manoeuvre specification

International Standard ISO-3888:1991,2002 specifies two test-track layouts for performing lateral manoeuvres with a passenger car. Part 1 (ISO-3888-1:1999) specifies a track layout for performing a double lane change manoeuvre. Part 2 (ISO-3888-2:2002) specifies a layout for an obstacle avoidance double lane change manoeuvre; this is similar to the Part 1 specification but the manoeuvre limits are more tightly constrained. The car must travel further to the side in a shorter distance, thus increasing the acceleration that the vehicle must undergo if it is to successfully navigate the course. In both cases, the standard recommends that the manoeuvre be performed with an initial forward speed of 80 ± 3 [km/hr] ($22.\dot{2} \pm 0.8\dot{3}$ [m/s]). The general shape of the test-track layout is shown in Figure 3.1 and the dimensions for each of the manoeuvres are given in Table 3.1.

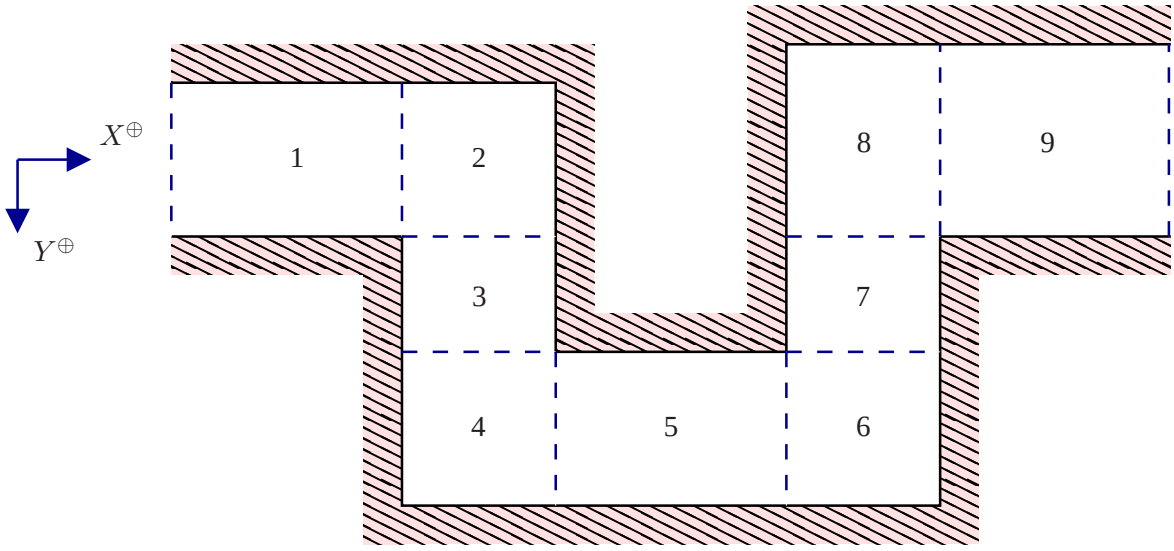


Figure 3.1: Test track layout for a double lane change manoeuvre.

The standard is intended to be used to assess the handling characteristics of vehicles by drivers, but the specified test-tracks form suitable obstacle courses for evaluating the performance of an emergency obstacle controller. In an emergency situation, it may be sensible for a vehicle to remain in its new lane after avoiding collision with an obstacle, rather than automatically returning to its previous lane. Appropriate single lane-change manoeuvres can be obtained by considering only the first five sections of each of the specified test-track layouts.

Table 3.1: Test track dimensions for a double lane change manoeuvre derived from ISO 3888 Parts 1 and 2.

ISO 3888	*Part 1*		*Part 2*	
Section	*Length [m]*	*Width [m]*	*Length [m]*	*Width [m]*
1	15.0	1.1 × car + 0.25	12.0	1.1 × car + 0.25
2	30.0		13.5	
3		3.5 - (1.1 × car + 0.25)		1.0
4		1.2 × car + 0.25		1.0 × car + 1.00
5	25.0		11.0	
6	25.0		12.5	
7		3.5 - (1.1 × car + 0.25)		1.0
8		1.3 × car + 0.25		1.3 × car + 0.25
9	30.0		12.0	but ≥ 3.0

3.2 Geometric method

Finding feasible paths through an obstacle course has long been of interest to robotics researchers. Dubins (1957) showed that, for a particle that does not reverse, the shortest paths are geodesic, consisting of circular arcs and straight line segments. This section describes the construction of such paths suitable for the target vehicle to perform specified lateral obstacle avoidance manoeuvres.

3.2.1 Vehicle dynamic constraints

Given the traction limits described in Section 1.3.1, it is necessary to determine a trajectory that will respect the acceleration limits of the car. Traction saturation leads to a conflict between steering and braking; between lateral and longitudinal acceleration. Redirecting the car's considerable forward momentum by pointing it in a different direction will allow a lateral shift to be performed far faster than attempting to reduce speed while increasing lateral momentum.

One strategy that might therefore be expected to generate a good reference trajectory for a lateral emergency collision avoidance manoeuvre is to change lanes in the following manner. Turn the car as quickly as possible at the start of the manoeuvre, use the vehicle's forward speed to move swiftly into the adjacent lane, then aggressively redirect its momentum in the direction of the new lane. It should be noted that this is significantly different from the more gentle lane-changing manoeuvres investigated by other researchers for vehicles on autonomous highway systems where passenger comfort is of greater importance.

The vehicle is capable of a maximum acceleration of μg [m/s^2] (Section 2.4) and, if steering is to be preferred over braking, it is sensible to direct the acceleration vector perpendicular to the forward speed of the car. This will result in a circular trajectory with radius R

$$R = \dot{X}^2 / (\mu g) \,[\mathrm{m}] \tag{3.1}$$

where $\dot{X}$ is the (constant) tangential speed of the vehicle.

3.2.2 Trajectory construction for a single lane-change

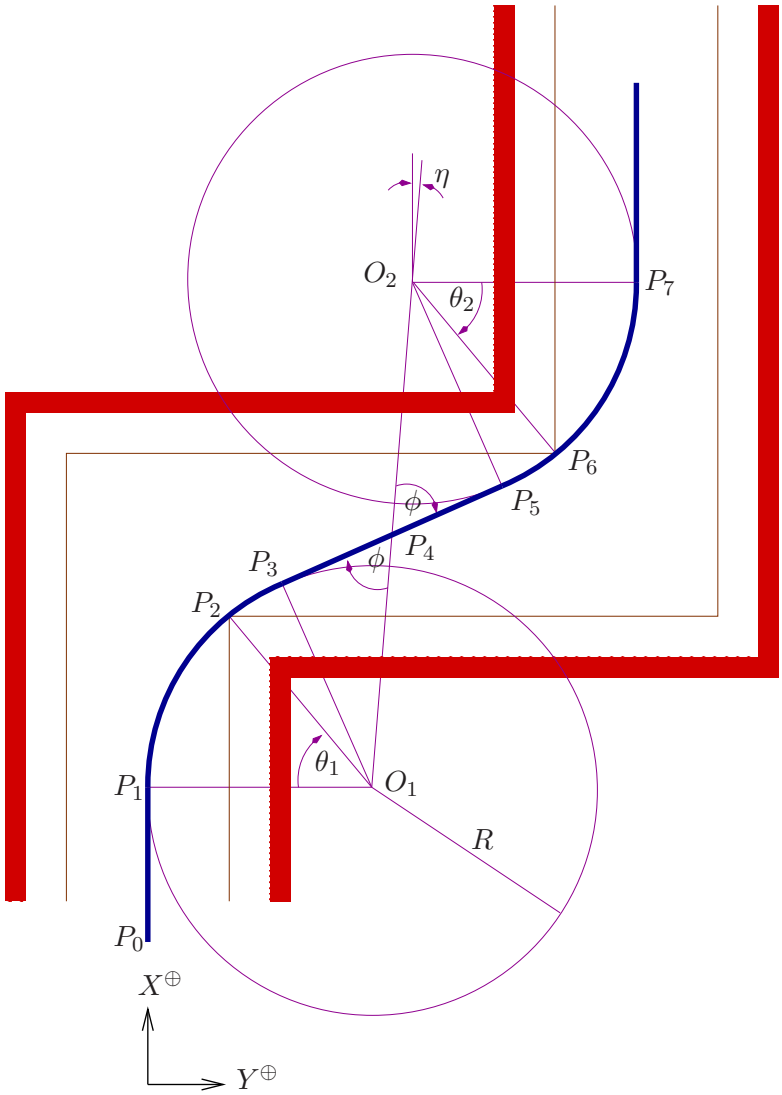


Figure 3.2: Geometric construction of reference trajectory. The trajectory is shown in blue. The limits of the manoeuvre space are shown in red. Brown lines, set half a car width inside the red boundary, show the area within which the vehicle centre-line must remain. Construction lines are depicted in magenta.

Figure 3.2 shows the construction of a trajectory consisting of straight lines and circular arcs for a single lane change. Figure 3.3 shows certain details of the construction in isolation. Passing through position $P_o = \left(X^{\oplus}_{P_0}, Y^{\oplus}_{P_0}\right)$, the trajectory follows the centre of the first lane until reaching $P_1 = \left(X^{\oplus}_{P_1}, Y^{\oplus}_{P_1}\right)$, the beginning of a maximum acceleration turn to the right. Continuing the turn through $P_2 = \left(X^{\oplus}_{P_2}, Y^{\oplus}_{P_2}\right)$, the point of closest approach to the boundary, the trajectory reaches $P_3 = \left(X^{\oplus}_{P_3}, Y^{\oplus}_{P_3}\right)$, from where it follows a straight path through $P_4 = \left(X^{\oplus}_{P_4}, Y^{\oplus}_{P_4}\right)$ to $P_5 = \left(X^{\oplus}_{P_5}, Y^{\oplus}_{P_5}\right)$. A maximum acceleration turn to the left, through $P_6 = \left(X^{\oplus}_{P_6}, Y^{\oplus}_{P_6}\right)$ and $P_7 = \left(X^{\oplus}_{P_7}, Y^{\oplus}_{P_7}\right)$, brings the trajectory to the centre of the destination lane.

The key to calculating the trajectory is identification of the centres of circles with the minimum

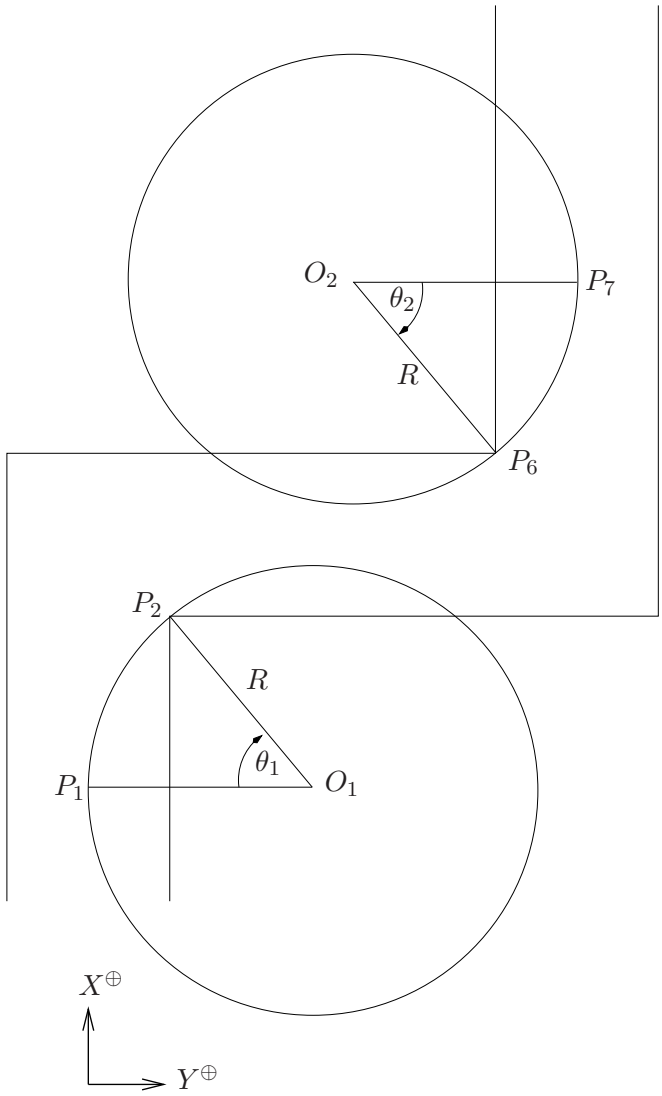


Figure 3.3: Detail of trajectory construction. Elements of Figure 3.2 are shown in isolation to show the placement of circles with minimum radius of curvature and relationships between certain points.

radius of curvature that describe the most aggressive circular trajectories that the car can follow, co-ordinates $O_1 = \left(X^{\oplus}_{O_1}, Y^{\oplus}_{O_1}\right)$ and $O_2 = \left(X^{\oplus}_{O_2}, Y^{\oplus}_{O_2}\right)$. At the start and end of the manoeuvre, the centres of the lanes are tangential to the circles. The lateral position of the centres is therefore simply offset from the lane centres by a distance R [m] (Equation (3.1)) in the appropriate direction. The longitudinal positions of the circles are constrained by the manoeuvre boundary. The first circle meets the boundary at position P_2 while the second circle meets the boundary at position P_6. The points P_2 and P_6 are the points of closest approach of the vehicle to the boundary, defined to be offset longitudinally and laterally by half the width of the car from the vertices of the obstacle boundary.

Considering the co-ordinates of P_1 and P_2, only $X^{\oplus}_{P_1}$ is unknown. The arc $\widehat{P_1P_2}$ subtends an angle $\theta_1 = \angle P_1O_1P_2$ at O_1, which lies $R\cos\theta_1$ [m] to the right of P_2 and R [m] to the right of P_1.

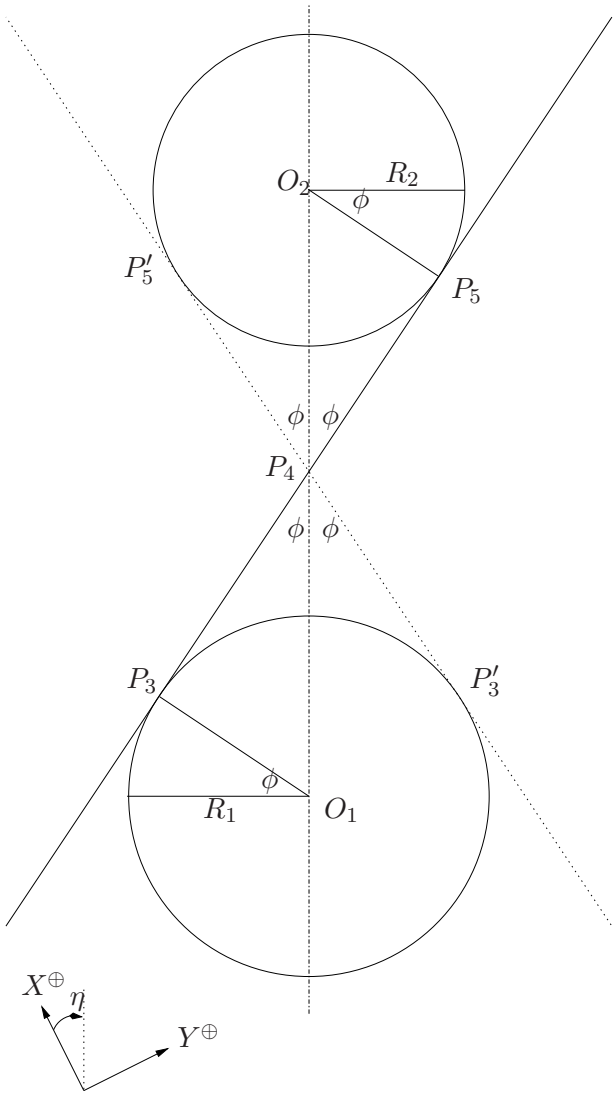


Figure 3.4: Construction of tangents. Any two non-intersecting co-planar circles have four common tangents, two of which cross the centre-line between them

Thus $R(1-\cos\theta_1) = Y^{\oplus}_{P_2} - Y^{\oplus}_{P_1}$, giving

$$\theta_1 = \arccos\left(1 - \frac{Y^{\oplus}_{P_2} - Y^{\oplus}_{P_1}}{R}\right) \tag{3.2}$$

The longitudinal position of P_1, $X^{\oplus}_{P_1}$ can then be calculated as $X^{\oplus}_{P_1} = X^{\oplus}_{P_2} - R\sin\theta_1$. Thus the co-ordinates of O_1 are

$$O_1 = \left(X^{\oplus}_{O_1},\, Y^{\oplus}_{O_1}\right) = \left(X^{\oplus}_{P_2} - R\sin\theta_1,\, Y^{\oplus}_{O_1} = Y^{\oplus}_{P_1} + R\right) \tag{3.3}$$

Similarly, positions P_6 and P_7 may be used to obtain angle θ_2

$$\theta_2 = \arccos\left(1 - \frac{Y^{\oplus}_{P_7} - Y^{\oplus}_{P_6}}{R}\right) \tag{3.4}$$

and hence the co-ordinates of O_2 are

$$X^{\oplus}_{O_2} = \left(X^{\oplus}_{O_2},\, Y^{\oplus}_{O_2}\right) = \left(X^{\oplus}_{P_6} + R\sin\theta_2,\, Y^{\oplus}_{O_2} = Y^{\oplus}_{P_7} - R\right) \tag{3.5}$$

All that remains is to find the line $\overrightarrow{P_3P_5}$ which is tangential to both circles and does not cross the boundary. There are two such lines for any two non-intersecting co-planar circles (or ellipses), symmetric about the centre-line, as shown in Figure 3.4. The angle between each of the tangents and the centre-line is

$$\phi = \arcsin\frac{R_1 + R_2}{\left|\overrightarrow{O_1O_2}\right|} = \arcsin\frac{2R}{\sqrt{\left(X^{\oplus}_{O_2} - X^{\oplus}_{O_1}\right)^2 + \left(Y^{\oplus}_{O_2} - Y^{\oplus}_{O_1}\right)^2}} \tag{3.6}$$

where R_1 and R_2 are the radii of each circle, which in this case are both equal to R. The centre-line $\overrightarrow{O_1O_2}$ is rotated from the $X^{\oplus}$ axis by an angle

$$\eta = \arctan\frac{Y^{\oplus}_{O_2} - Y^{\oplus}_{O_1}}{X^{\oplus}_{O_2} - X^{\oplus}_{O_1}} \tag{3.7}$$

The gradients of the tangents are therefore $\tan(\eta + \phi)$, for line $\overrightarrow{P_3P_5}$, and $\tan(\eta - \phi)$, for its mirror $\overrightarrow{P_3'P_5'}$. For circles of equal radius, the tangents cross half way along the centre-line, at

$$P_4 = \frac{1}{2}\left(X^{\oplus}_{O_1} + X^{\oplus}_{O_2},\, Y^{\oplus}_{O_1} + Y^{\oplus}_{O_2}\right) \tag{3.8}$$

The co-ordinates of the points where the tangents meet the circles are

$$P_3 = \left(X_{O_1} + R\sin(\phi+\eta),\, Y_{O_1} - R\cos(\phi+\eta)\right) \tag{3.9}$$

$$P_5 = \left(X_{O_2} - R\sin(\phi+\eta),\, Y_{O_2} + R\cos(\phi+\eta)\right) \tag{3.10}$$

$$P_3' = \left(X_{O_1} + R\sin(\phi-\eta),\, Y_{O_1} + R\cos(\phi-\eta)\right) \tag{3.11}$$

$$P_5' = \left(X_{O_2} - R\sin(\phi-\eta),\, Y_{O_2} - R\cos(\phi-\eta)\right) \tag{3.12}$$

3.2.3 Summary of waypoints

Seven waypoints have been defined that describe completely the path for the vehicle. They are summarised below.

P_0 The vehicle's initial position.

P_1 The start of the vehicle's first turn, to follow the arc centred at O_1.

P_2 The point of closest approach to the outer boundary.

P_3 The point at which the vehicle stops turning and begins following a straight line segment.

P_4 The midpoint of the line segment, halfway between the end of the first turn and the start of the

second.

P_5 The start of the second turn, to follow the arc centred on O_2.

P_6 The point of closest approach to the inner boundary.

P_7 The end of the second turn, after which the vehicle follows a straight course.

3.2.4 Method limitations

For a double lane-change, two further circles must be defined, centred at O_3 and O_4 (Figure 3.5. These circles are placed in a similar manner to O_1 and O_2. The result of applying this geometric technique to generate a feasible trajectory through the ISO 3888-2 emergency obstacle avoidance manoeuvre for a vehicle travelling at a constant forward speed of 60 [km/hr] is shown in Figures 3.6 and 3.7. The method works well but has two shortcomings. The decision to perform each turn at the maximum possible rate is highly appropriate when the vehicle is required to operate at its physical limits. However, as noted by Bevan, O'Neill, Gollee & O'Reilly (2007), such aggressive turns lead to unnecessarily high lateral accelerations when performing manoeuvres that could be navigated more sedately, such as when travelling at lower speeds. Although passenger comfort is necessarily a secondary consideration when performing emergency evasive manoeuvres, it would be desirable for a general trajectory generation method to be capable of finding less severe paths when appropriate.

A more important limitation of the method arises from the assumption of constant forward velocity. The minimum radius of curvature for a circular path is proportional to the square of vehicle speed. As vehicle speed increases, the radius of each circle increases accordingly. For tight manoeuvres at high speed, the radii may be sufficiently large that the circles cannot be placed without intersecting, which means that no feasible trajectory can be found. The solution to this problem is to reduce vehicle speed during the manoeuvre, thus reducing the minimum radius of curvature during later stages. However, the cost of reducing speed is that, due to traction saturation, the minimum radius of curvature increases while longitudinal braking forces are applied. Thus a trade-off exists between the desire to turn the car as fast as possible and the desire to reduce vehicle speed to allow faster turns later in the manoeuvre. The existence of this trade-off suggests that it should be possible to find an optimal trajectory to balance these conflicting requirements.

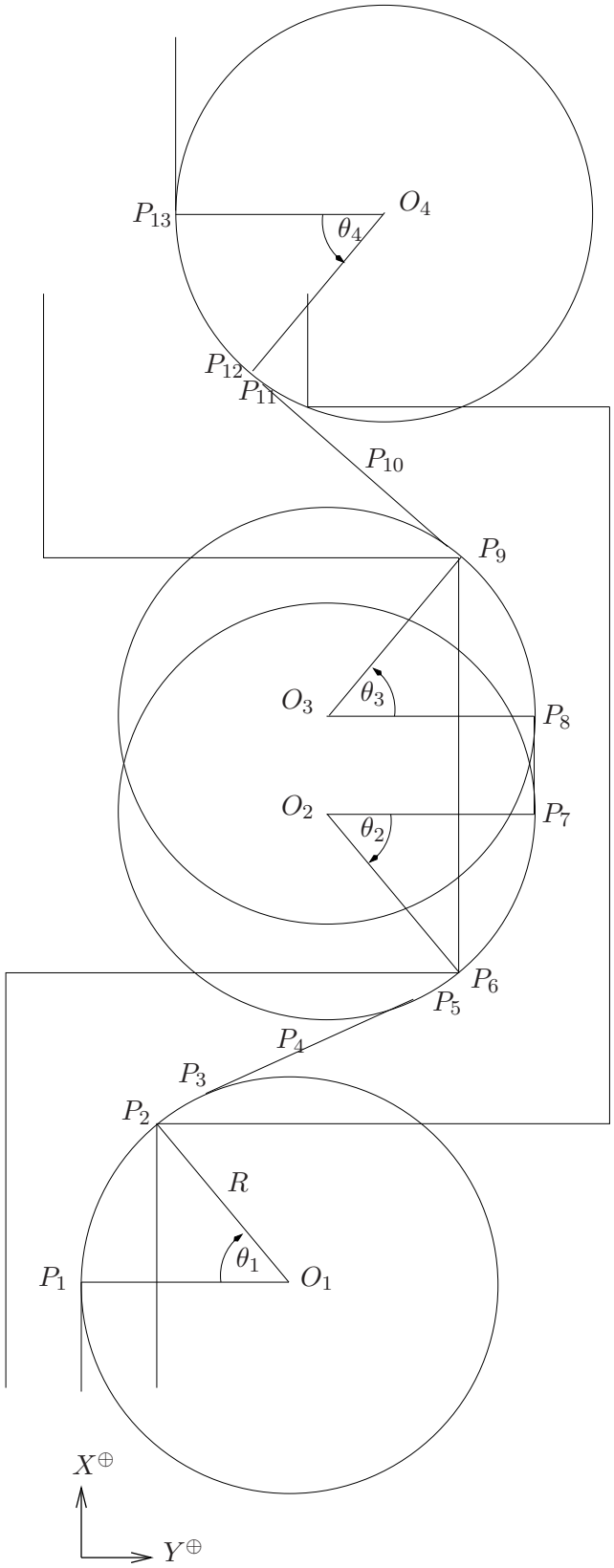


Figure 3.5: Geometric method for a double lane-change. Two further circles, centred on O_3 and O_4, must be placed to generate the path for a double lane change manoeuvre, adding six waypoints (P_8 to P_{13}). These are placed in an identical matter to the two circles centred on O_1 and O_2.

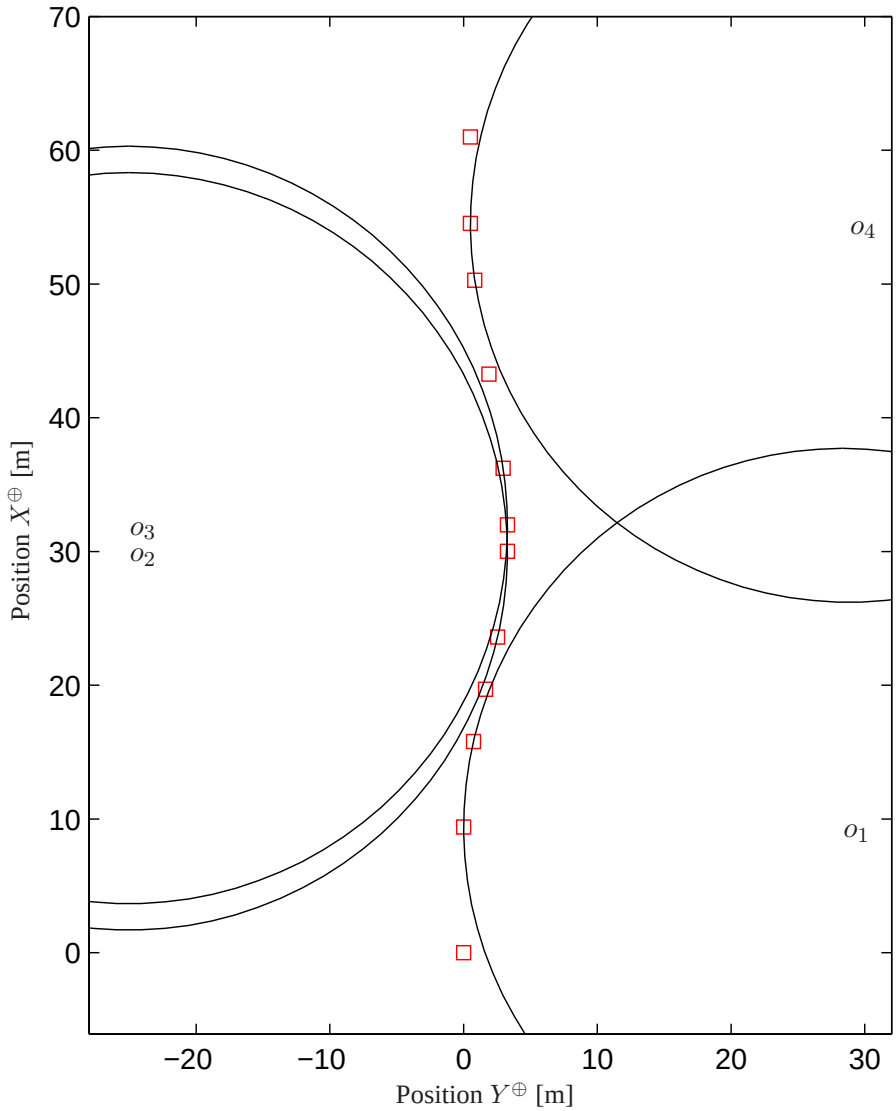


Figure 3.6: Geometrically-placed waypoints for the ISO 3888-2 double lane-change manoeuvre are depicted as red squares. The circles indicate turns of minimum radius for a vehicle speed of 60 [km/hr].

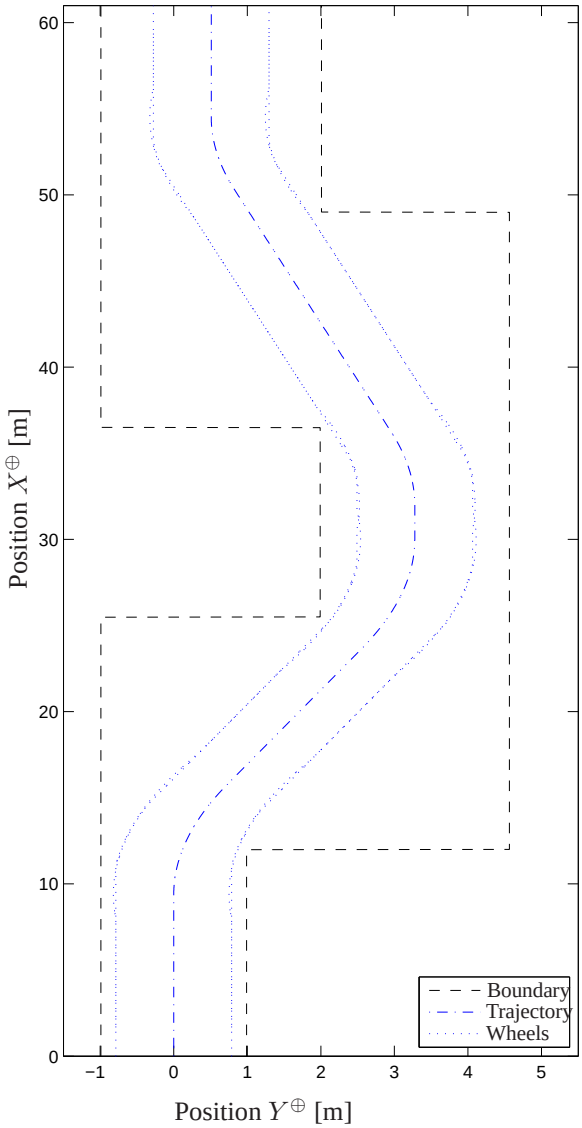


Figure 3.7: The trajectory resulting from the waypoints of Figure 3.6 is shown as a chained line. Dotted lines indicate the wheel positions, assuming that the vehicle orientation remains tangential to its path. The boundary of the manoeuvre is depicted by dashed lines.

3.3 Optimisation

It is desired to find an optimal trajectory that balances competing demands upon the available traction: the demand for longitudinal forces, to slow the car; and for lateral forces, to steer the car. An optimal balance between braking and steering can be found using numerical optimisation.

Optimisation is a complex and well-studied art, closely related to the solution of differential-algebraic equations (DAE). General purpose DAE solvers such as DASSL and LSODI, which rely on backwards differentiation formulae (BDF), have been applied to Trajectory Prescribed Path Control (TPPC) aerospace problems (Brenan, Campbell & Petzold 1996). However, for these problems the path is known *a priori* and the problem is to find the required control inputs. Even here, the authors report numerical difficulties. Such codes are adept at solving initial value problems of index 1, but substantial difficulties arise when higher order DAEs are encountered, as occurs when the constraints are not continuously differentiable. Index reduction, whereby constraints are differentiated until smooth can improve reliability. However, index reduction is often difficult in practice and the solution of the reduced problem need not exactly meet the original constraints. This could be problematic where the constraints are physical barriers, as in the case of an obstacle avoidance manoeuvre.

Problem-specific techniques are often more appropriate than general purpose methods. A method that works well for one trajectory optimisation problem may be totally inappropriate for others (Betts 1998).

In recent years, it has been recognised that efficient methods exist for solving convex optimisation problems and that these arise frequently in the context of engineering (Boyd & Vandenberghe 2004). A convex optimisation problem is one in which it is desired to minimise a convex objective function subject to convex constraints.

Obtaining an optimal solution is essentially a problem of finding a tangent to the set of active constraints in the problem space. For general nonlinear optimisation problems, a substantial difficulty for solvers is that of finding a global minimum without getting trapped by local minima. However, when the problem can be expressed in convex form, any local minimum is also a global minimum, thus allowing very efficient solution algorithms to be used.

Hattori, Ono & Hosoe (2006) note that determination of an optimal trajectory generally requires a large amount of calculation. They show how convex optimisation can be used to generate an obstacle avoidance trajectory by considering the vehicle as a non-rotating point mass and performing a convex optimisation in the vehicle's body axis system. Their method neglects yawing of the vehicle and does not therefore take account of rotation of the vehicle axis system relative to the Earth. It is necessary to extend the work if the constraints are specified in the fixed Earth axis system.

To illustrate the importance of considering rotation, suppose that we wish the vehicle to follow a trajectory $Y^{\oplus} = \cos(\xi X^{\oplus}) - 1$, where ξ is a constant, at constant forward speed u [m/s]. If the vehicle is considered to be a point mass and rotation of the axis is neglected, the necessary equations

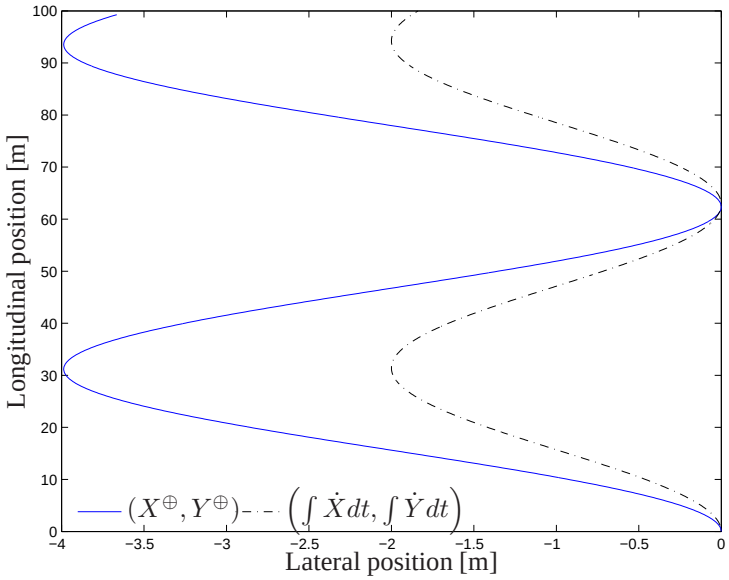


Figure 3.8: Effect of axis rotation due to yaw on the trajectory for a vehicle following the trajectory $Y = \cos(\xi X) - 1$ at forward speed 10 m/s with $\xi = 0.1$. The motion measured in the body axis system, if yaw is neglected, is shown by the dashed line. The solid line shows the actual motion of the vehicle in the fixed Earth axis system.

of motion for a vehicle starting from the origin would be simply

$$\dot{X}(t) = u \qquad\qquad \dot{Y}(t) = -\xi u \sin(\xi u t) \tag{3.13}$$

where t denotes time. However, in reality the car would yaw while following such a trajectory. If it is assumed that there is little lateral slip and that the vehicle heading angle is therefore tangential to the direction of motion, i.e. $\psi = \arctan \frac{dY}{dX}$, then the velocity in the fixed Earth axis would be

$$\begin{pmatrix} \dot{X}^{\oplus} \\ \dot{Y}^{\oplus} \end{pmatrix} = \Gamma \begin{pmatrix} \dot{X} \\ \dot{Y} \end{pmatrix} \text{ where } \Gamma = \begin{pmatrix} +\cos\arctan\frac{dY}{dX} & , & -\sin\arctan\frac{dY}{dX} \\ +\sin\arctan\frac{dY}{dX} & , & +\cos\arctan\frac{dY}{dX} \end{pmatrix} \tag{3.14}$$

Noting that $\sin\arctan\chi \equiv \frac{\chi}{\sqrt{1+\chi^2}}$ and $\cos\arctan\chi \equiv \frac{1}{\sqrt{1+\chi^2}}$ the rotation matrix becomes

$$\Gamma = \frac{1}{\sqrt{1+\frac{dY}{dX}^2}} \begin{pmatrix} +1 & , & -\frac{dY}{dX} \\ +\frac{dY}{dX} & , & +1 \end{pmatrix} \tag{3.15}$$

The trajectory derivative is $\frac{dY}{dX} = -\xi\sin(\xi X) = -\xi\sin(\xi u t)$ and thus the actual velocity that would be seen in the fixed Earth axis is

$$\dot{X}^{\oplus}(t) = \frac{u\left(1-\xi^2\sin^2(\xi u t)\right)}{\sqrt{1+\xi^2\sin^2(\xi u t)}} \qquad\qquad \dot{Y}^{\oplus}(t) = \frac{-2u\xi\sin(\xi u t)}{\sqrt{1+\xi^2\sin^2(\xi u t)}} \tag{3.16}$$

Figure 3.8 shows the effect of axis rotation due to yaw on the trajectory: at any point in the manoeuvre, the lateral distance traversed by the vehicle relative to its starting position in the fixed

Earth axis would be twice that measured in the vehicle axis system. Clearly, if a trajectory is required to avoid obstacles specified in the fixed Earth axes, this axis rotation must be considered during the trajectory generation process.

3.3.1 Optimisation objective

Selection of an appropriate objective is an important part of any optimisation. Minimising the time or distance of a manoeuvre are reasonable approaches that can be used for normal or emergency lane changes, as demonstrated on the California PATH project (Godbole, Hagenmeyer, Sengupta & Swaroop 1997). However, these criteria are not of particular importance if the obstacle to be avoided is in a fixed position or if its position throughout the manoeuvre can be constrained to a definite region. If the vehicle is to continue travelling at high speed throughout the manoeuvre, perhaps to merge into a new lane without causing a collision with other fast moving traffic, then it may be more appropriate to seek a trajectory that is in some sense smooth and that minimises control effort (i.e. steering and braking forces) while respecting the constraints. Sledge Jr. & Marshek (1998) observe that the characteristics of such a trajectory are analogous to the natural bending of a beam. They find an analytical solution for a single lane change by minimising the mean-square curvature of the path. However, their solution relies on the vehicle travelling at constant speed, which precludes use of the brakes and limits the manoeuvre to vehicles travelling below a critical speed. Meanwhile, Blank & Margolis (2000) show that minimising path curvature is beneficial for assisting the driver if both the steering and braking inputs are saturated, which does account for changing speed but does not encompass the general case in the absence of saturation.

With the assumption that the vehicle heading remains tangential to its path, i.e. that lateral slip is negligible, minimising the instantaneous path curvature for a given speed is equivalent to minimising the yaw acceleration of the vehicle. Thus minimising the norm of the yaw acceleration over the length of the manoeuvre should produce a desirable trajectory that intuitively can be expected not to waste control effort. In this context, wasted effort is that which needlessly reduces the available control authority of the system. For a vehicle to accurately follow any chosen trajectory, it is necessary for its controller to provide corrective action. Thus a good trajectory should not waste traction that could be better used for corrective action later.

There are secondary objectives which may be considered to be desirable characteristics of a good trajectory, but which are not explicitly accounted for in the optimisation procedure. Firstly, it should be possible to calculate a feasible trajectory that will allow the car to move to safety when travelling at high speed; the higher the initial speed for which a trajectory can be obtained, the greater the usefulness of the method. Secondly, traction saturation should not be induced unnecessarily so that additional control inputs may be applied to compensate for any deviation of the vehicle from its trajectory. Thirdly, it may be desired that the vehicle should exit the manoeuvre with a forward speed that is either: a) as low as possible to assist the driver in making an emergency stop, or b) as close as possible to the speed of other traffic to enable the vehicle to merge safely.

3.3.2 Grid generation in manoeuvre space

A naive optimisation strategy might involve repeatedly running a time-based simulation to determine the full vehicle trajectory resulting from potential control strategies. However, it is not desirable for the optimisation routine to run a computationally-demanding simulation every time its cost function is evaluated. It is better to operate simultaneously on a full description of the entire system. Direct transcription (Betts 2001) offers an appropriate means of representing the full system.

A grid is established, comprising the system states (vehicle position and velocity) at discrete points throughout the manoeuvre space. Numerical integration of the equations of motion is then achieved by converting an appropriate quadrature function into a set of constraints (Equation (3.24) below).

The manoeuvre boundary is specified as a function of longitudinal distance in the fixed Earth axis system (Section 3.1). It is therefore convenient to generate the grid with longitudinal distance $X^{\oplus}$ as the independent variable. Choosing any other parameter, such as time, would result in a non-constant set of boundary constraints and a significant increase in computational complexity.

Considering an initial position $X_0^{\oplus}$ and a further set of L points along the $X^{\oplus}$ axis, with equi-distant spacing Δ, then the position of the j^{th} point is

$$X_j^{\oplus} = X_0^{\oplus} + j \times \Delta \qquad \forall j \in [0, L] \tag{3.17}$$

The grid $\mathcal{G}$ is then defined as

$$\mathcal{G} = (\mathbf{G_0}, \cdots, \mathbf{G_L}) \in \mathbb{R}^{6\times(L+1)} \tag{3.18}$$

where

$$\mathbf{G_j} = \mathbf{G}(X_j^{\oplus}) \qquad \forall j \in [0, L] \tag{3.19}$$

and

$$\mathbf{G}(X^{\oplus}) = \left(X^{\oplus}, Y^{\oplus}, \psi, \dot{X}, \dot{Y}, \dot{\psi}\right)^T \in \mathbb{R}^6 \tag{3.20}$$

The trajectory generation problem is not convex but certain simplifying assumptions enable the formulation of a convex approximation to the system of equations. It is thereby possible to take advantage of the power of convex optimisation algorithms. The optimisation is performed using the CVX (2005) Matlab package which implements the Disciplined Convex Optimisation modelling framework of Grant, Boyd & Ye (2006).

3.3.3 Optimisation problem specification

Objective The optimisation objective is to minimise the yaw acceleration of the vehicle throughout the length of the manoeuvre.

$$\text{Minimise J} = \|\ddot{\psi}\| \tag{3.21}$$

Grid spacing

The grid spacing is arbitrarily set to $\Delta = 1$ [m], a length which provides sufficient resolution for the trajectory to take shape without requiring excessive computation.

Initial conditions

The Earth axis is fixed at the starting position of the vehicle which is initially moving straight ahead with a forward speed of 22.2 [m/s] (80 [km/hr]) and has no lateral or yaw component of velocity.

$$\begin{aligned} X_0^{\oplus} &= 0 \text{ [m]} & Y_0^{\oplus} &= 0 \text{ [m]} & \psi_0 &= 0 \text{ [rad]} \\ \dot{X}_0 &= 22.2 \text{ [m/s]} & \dot{Y}_0 &= 0 \text{ [m/s]} & \dot{\psi}_0 &= 0 \text{ [rad/s]} \end{aligned} \tag{3.22}$$

Terminal conditions

At the manoeuvre terminus, it is desired that the vehicle should perform lane-keeping and maintain a steady heading along the centre-line of the lane in which it is travelling, which is located approximately half a metre to the right of its initial position. No longitudinal speed is specified.

$$Y_L^{\oplus} = 0.5093 \text{ [m]} \qquad \psi_L = 0 \text{ [rad]} \qquad \dot{\psi}_L = 0 \text{ [rad/s]} \tag{3.23}$$

Quadrature

The vector $\mathbf{G}$ (Equation (3.18)) is evaluated at each grid point by performing a forward Euler integration with the time T that the vehicle takes to cover the distance between each grid point used as the integration step length.

$$\mathbf{G_{j+1}} = \mathbf{G_j} + \dot{\mathbf{G}}_\mathbf{j} \times T \qquad \forall j \in [0, L] \tag{3.24}$$

Acceleration limits

Traction saturation, in the form of a nominal friction circle, is expressed as a limit on the yaw acceleration. Two further limits are imposed: on the longitudinal velocity, to ensure that the vehicle does not move backwards at any time; and on the longitudinal acceleration, to ensure that the vehicle does not increase its speed.

$$\dot{X} \geq 0 \qquad \ddot{X} \leq 0 \qquad \ddot{\psi}^2 \leq \left(\frac{ml_f}{J_{ZZ}}\right)^2 \left((\mu g)^2 - \ddot{X}^2\right) \tag{3.25}$$

Course boundary

The requirement that the vehicle remain within the defined track is expressed as a constraint on the positions of the wheels, which are limited by a lower boundary $b_l^{\oplus}$ and an upper boundary $b_u^{\oplus}$, representing the left and right hand limits of the track respectively. The lateral position of the i^{th}

wheel, in the fixed Earth axis system, relative to the vehicle centre of mass is a function of vehicle orientation, and denoted by $W_i^{\oplus}$.

$$b_l^{\oplus} \leq Y^{\oplus} + W_i^{\oplus} \leq b_u^{\oplus} \qquad \forall i \in [1,4] \tag{3.26}$$

Non-convex constraints

There are several constraints that are incompatible with a convex problem formulation, because they involve trigonometric functions of a vector to be optimised and/or the product or quotient of two such vectors. Various terms in each of the following constraint equations are replaced in each of the optimisation runs so that the problem can be specified in a form suitable for solution by a convex algorithm. The problematic constraints are

$$\text{Axis rotation} \qquad \begin{cases} \dot{X}^{\oplus} &= \dot{X}\cos\psi - \dot{Y}\sin\psi \\ \dot{Y}^{\oplus} &= \dot{X}\sin\psi + \dot{Y}\cos\psi \end{cases} \tag{3.27}$$

$$\text{Wheel positions} \qquad \left\{ W_i^{\oplus} = l_{x,i}\sin\psi + l_{y,i}\cos\psi \qquad \forall i \in [1,4] \right. \tag{3.28}$$

$$\text{Time step} \qquad \left\{ T \quad = \frac{dX^{\oplus}}{d\dot{X}^{\oplus}} \right. \tag{3.29}$$

The integration step length (Equation (3.29)) presents a problem if the speed is allowed to vary. The vehicle dynamic equations are expressed as rates in the time domain whereas the grid is specified as a function of distance. If the speed were constant, multiplication by a fixed constant would allow rates to be expressed in terms of distance. However, this is not possible when the speed varies. For quadrature evaluation during the optimisation, nominal fixed time-steps of length T [s] are chosen to represent the time taken for the vehicle to travel between each grid point. Inconsistencies between distance, speed and time are then reconciled during post-processing.

Axis rotation leads to a set of non-convex constraints due to the presence of trigonometric terms and the multiplication of vectors (Equation (3.27)). Inclusion of vehicle orientation for determination of wheel positions (Equation (3.28)) leads to similar problems. One solution that can often be applied to robotic trajectory planning is to consider a circle of sufficient diameter to enclose the entire vehicle, in which case the orientation does not matter. However, the length of a car is generally significantly longer than its width. In this case, such an encompassing circle would exceed the boundaries, which are defined in terms of the vehicle width. Thus it is necessary to include the vehicle orientation. However, if it is assumed that the vehicle heading angle is small, a first-order Taylor expansion of these trigonometric functions leads to an affine formulation.

In the constraint equations that follow, these non-convex equations are replaced with approximations in which only the vectors denoted with an over-line can vary during the optimisation, i.e: $\overline{\dot{X}}$, $\overline{\dot{X}^{\oplus}}$, $\overline{\dot{Y}^{\oplus}}$, $\overline{\psi}$ and $\overline{W_i^{\oplus}}$. All other parameters and vectors are held constant during optimisation, but may be altered during post-processing.

3.3.4 First pass

The optimisation is performed in three stages. To formulate a convex problem, the first stage optimisation requires several assumptions and approximations that affect the suitability of the solution. The second and third stages make use of earlier results to relax some of these assumptions, thus enabling closer convergence with the true solution.

The first pass determines a feasible path, the locus of which has an appropriate shape to respect the boundary constraints and which is attainable within the traction limits of the tyres. Several assumptions and approximations are made to render the system in a convex form. In particular, it is assumed that: the manoeuvre is performed at constant speed (Equation (3.32)); there is no lateral slip (Equation (3.31)); and the heading angle remains small (Equation (3.30)). The resulting trajectory will not obey the boundary limits when mapped into the real fixed Earth axis system but provides a useful starting point for refinement in subsequent stages.

Having identified an approximate solution, the trajectory is post-processed. The tangent to the trajectory is calculated throughout the manoeuvre to determine the heading angle, still assuming no lateral slip. This heading angle is then used to rotate the velocity vector and calculate the path that the vehicle would actually have followed. This procedure effectively removes the small angle approximation from the result.

The first pass can be summarised as follows:

Approximations I

$$\text{Small angle} \quad \begin{cases} \cos\psi & \leftarrow 1 \\ \sin\psi & \leftarrow \psi \end{cases} \tag{3.30}$$

$$\text{No lateral slip} \quad \left\{ \ddot{Y} \leftarrow 0 \right. \tag{3.31}$$

$$\text{Constant speed} \quad \begin{cases} \ddot{X} & \leftarrow 0 \\ T & \leftarrow \Delta / \dot{X}_0^{\oplus} \end{cases} \tag{3.32}$$

Convex constraints I

$$\text{Axis rotation} \quad \begin{cases} \dot{X}^{\oplus} & = \dot{X}_0 \\ \overline{\dot{Y}^{\oplus}} & = \dot{X}_0 \overline{\psi} \end{cases} \tag{3.33}$$

$$\text{Wheel positions} \quad \left\{ \overline{W_i^{\oplus}} = l_{x,i}\overline{\psi} + l_{y,i} \qquad \forall i \in [1,4] \right. \tag{3.34}$$

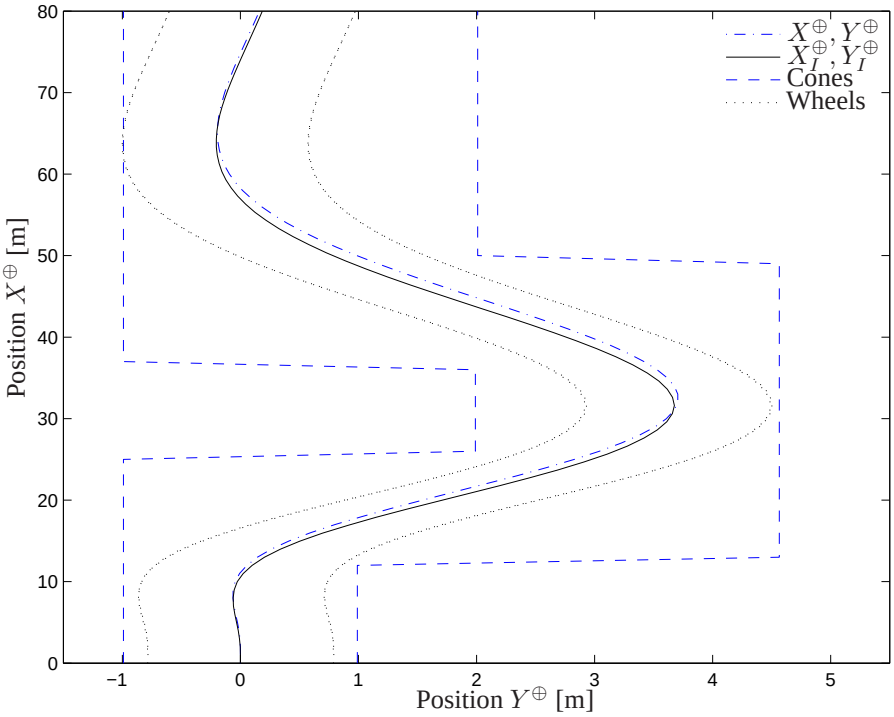


Figure 3.9: First pass. The chained line shows the trajectory produced during the first stage optimisation. The solid line shows the corrected trajectory after first stage post-processing. Dotted lines indicate the positions of the wheels, assuming that the vehicle's orientation remains tangential to its corrected path, and dashed lines indicate the manoeuvre boundary.

Post-processing I

Following the optimisation, the vehicle position at each point is re-evaluated using the calculated heading angle ψ_I instead of the small angle approximation

$$X_{I,j}^{\oplus} \leftarrow X_0^{\oplus} + \int_0^{t_j} \dot{X}_I \cos\psi_I - \dot{Y}_I \sin\psi_I dt \qquad \forall j \in [0, L] \tag{3.35}$$

$$Y_{I,j}^{\oplus} \leftarrow Y_0^{\oplus} + \int_0^{t_j} \dot{X}_I \sin\psi_I + \dot{Y}_I \cos\psi_I dt \qquad \forall j \in [0, L] \tag{3.36}$$

where the subscript I denotes the final values following completion of the optimisation and $t_j = j \times T$ denotes the time at which each grid point j is reached. The heading profile is then rescaled so that it corresponds to the specified grid positions $X_j^{\oplus}$ rather than the longitudinal positions $X_{I,j}^{\oplus}$ actually attained by the vehicle at each point.

$$\psi_I(X_j^{\oplus}) \leftarrow \psi_I(X_{I,j}^{\oplus}) \qquad \forall j \in [0, L] \tag{3.37}$$

3.3.5 Second pass

Starting with the result of the first pass, a second optimisation then allows the speed to vary, holding constant the yaw acceleration profile, as a function of longitudinal distance, under the assumption

that the shape of the optimal trajectory will be similar to that found in the first optimisation pass. During this second optimisation, it is assumed that the longitudinal position at each time coincides precisely with the initial grid spacing. Thus it is assumed that the vehicle covers a distance Δ in each integration step no matter what its velocity (Equation (3.41)).

By pre-calculating $\cos\psi_I$ and $\sin\psi_I$ using the heading profile ψ_I from the preceding optimisation, it is possible to introduce these trigonometric expressions into the constraint equations as constants, allowing an affine/convex formulation of the vehicle trajectory in the fixed Earth axis system and partially dispensing with the small heading angle approximation (Equation (3.39)).

The second pass can be summarised as follows:

Approximations II

$$\text{Small angle} \qquad \left\{\begin{array}{ll}\cos\psi & \leftarrow 1\\ \sin\psi & \leftarrow \psi\end{array}\right\} \text{ for axis rotation} \tag{3.38}$$

$$\text{Fixed heading profile} \qquad \left\{\begin{array}{ll}\cos\psi & \leftarrow \cos\psi_I\\ \sin\psi & \leftarrow \cos\psi_I\end{array}\right\} \text{ for wheel positions} \tag{3.39}$$

$$\text{No lateral slip} \qquad \left\{\ddot{Y} \leftarrow 0\right. \tag{3.40}$$

$$\text{Constant speed} \qquad \left\{\begin{array}{ll}T & \leftarrow \Delta/\dot{X}_0^{\oplus}\\ \dot{X} & \leftarrow \dot{X}_0\end{array}\right\} \text{ for axis rotation } \dot{Y}^{\oplus} \text{ only)} \tag{3.41}$$

Convex constraints II

$$\text{Axis rotation} \qquad \left\{\begin{array}{ll}\overline{\dot{X}^{\oplus}} & = \overline{\dot{X}}\\ \overline{\dot{Y}^{\oplus}} & = \dot{X}_0\overline{\psi}\end{array}\right. \tag{3.42}$$

$$\text{Wheel positions} \qquad \left\{ W_i^{\oplus} = l_{x,i}\sin\psi_I + l_{y,i}\cos\psi_I \qquad \forall i \in [1,4]\right. \tag{3.43}$$

Post-processing II

Following the second optimisation, the resulting velocity profile is used to calculate the true longitudinal position of the vehicle at each instant. Reduction in vehicle speed during the manoeuvre reduces the distance covered. Consequently, the vehicle path will impinge on the boundary constraints because the car turns too early. The trajectory is therefore recalibrated (stretched) to compensate for this deficiency.

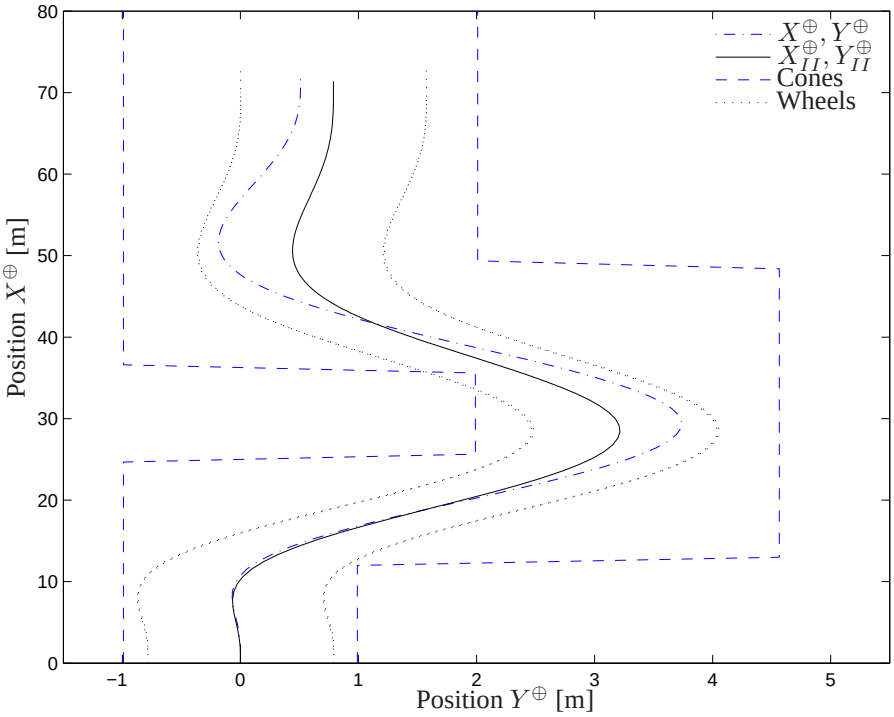


Figure 3.10: Second pass. The chained line shows the trajectory produced during the second stage optimisation. The solid line shows the corrected trajectory after second stage post-processing. Dotted lines indicate the positions of the wheels, assuming that the vehicle's orientation remains tangential to its corrected path, and dashed lines indicate the manoeuvre boundary.

The actual vehicle position at each instant is calculated

$$X^{\oplus}_{II,j} \leftarrow X^{\oplus}_0 + \int_0^{t_j} \dot{X}_{II} \cos\psi_{II} - \dot{Y}_{II} \sin\psi_{II} dt \qquad \forall j \in [0, L] \tag{3.44}$$

$$Y^{\oplus}_{II,j} \leftarrow Y^{\oplus}_0 + \int_0^{t_j} \dot{X}_{II} \sin\psi_{II} + \dot{Y}_{II} \cos\psi_{II} dt \qquad \forall j \in [0, L] \tag{3.45}$$

and the heading angle profile is recalibrated to match the specified grid positions

$$\psi_{II}(X^{\oplus}_j) \leftarrow \psi_{II}(X^{\oplus}_{II,j}) \qquad \forall j \in [0, L] \tag{3.46}$$

The subscript II here indicates the values obtained from the second pass.

3.3.6 Third pass

A third optimisation pass is then performed. As before, the values from the previous run can be used to insert non-convex expressions into the problem specification by holding them constant (Equation (3.48)). In this final optimisation, the heading angle (from the previous step) is included in the calculation of longitudinal position (Equation (3.51)). The longitudinal velocity profile of the previous (recalibrated) trajectory is also used when calculating lateral position, instead of assuming that the vehicle remains at its initial speed (Equation (3.50)). The result of this pass corresponds

closely with the vehicle's behaviour in the fixed Earth axes and is the solution sought.

The third pass can be summarised as follows:

Approximations III

$$\text{Small angle} \qquad \left\{ \sin\psi \quad \leftarrow \psi \right\} \text{ for axis rotation} \tag{3.47}$$

$$\text{Fixed heading profile} \qquad \left\{ \begin{matrix} \cos\psi & \leftarrow \cos\psi_{II} \\ \sin\psi & \leftarrow \sin\psi_{II} \end{matrix} \right\} \text{ for wheel positions} \tag{3.48}$$

$$\text{No lateral slip} \qquad \left\{ \ddot{Y} \quad \leftarrow 0 \right. \tag{3.49}$$

$$\text{Constant speed} \qquad \left\{ \begin{matrix} T & \leftarrow \Delta / \dot{X}_0^{\oplus} \\ \dot{X} & \leftarrow \dot{X}_{II} \end{matrix} \right\} \text{ for axis rotation } (\dot{Y}^{\oplus} \text{ only}) \tag{3.50}$$

Convex constraints III

$$\text{Axis rotation} \qquad \begin{cases} \overline{\dot{X}^{\oplus}} & = \overline{\dot{X}} \cos\psi_{II} \\ \overline{\dot{Y}^{\oplus}} & = \dot{X}_{II} \overline{\psi} \end{cases} \tag{3.51}$$

$$\text{Wheel positions} \qquad \left\{ W_i^{\oplus} = l_{x,i} \sin\psi_{II} + l_{y,i} \cos\psi_{II} \qquad \forall i \in [1,4] \right. \tag{3.52}$$

3.3.7 Optimisation results

Figures 3.9 to 3.11 show the evolution of the trajectory as the optimisation procedure runs through each of the three stages. Figure 3.9 shows that the first pass optimisation successfully determines a trajectory that remains within the specified boundaries. However, it should be noted that this trajectory is dependent upon the assumptions under which it was calculated. In particular, it is assumed that the forward speed remains constant.

In Figure 3.10, it can be seen that the second optimisation pass successfully manages to replicate the shape of the manoeuvre from the first pass while accounting for variation in speed. However, the effect of speed reduction, neglected in the first pass, can be clearly seen: a manoeuvre that would have avoided the boundaries at constant speed does in fact cross the boundary when the speed change is taken into account because the vehicle starts its second lane change too early.

After the trajectory has been recalibrated to account for the change in speed, the third pass successfully achieves a trajectory that respects the limits, while relying on fewer assumptions. The trajectory is shown in Figure 3.11.

The CVX programme, running in Matlab on an Intel Pentium IV personal computer with an Ubuntu GNU/Linux operating system performs the entire multi-stage optimisation in less than a minute.

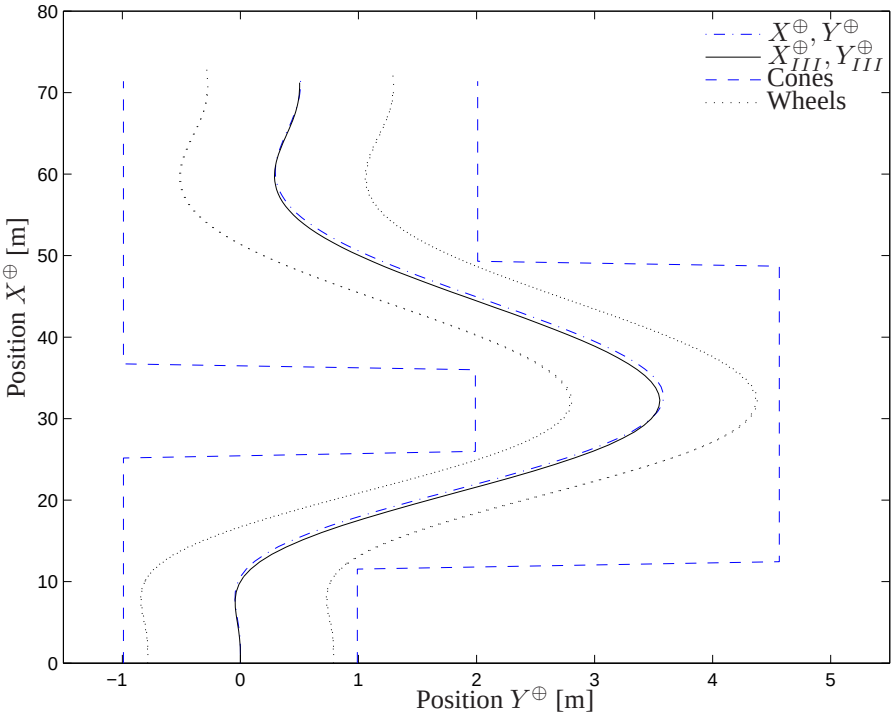


Figure 3.11: Third pass. The chained line shows the trajectory produced during the third stage optimisation. The solid line shows the corrected trajectory after third stage post-processing. Dotted lines indicate the positions of the wheels, assuming that the vehicle's orientation remains tangential to its corrected path, and dashed lines indicate the manoeuvre boundary.

3.4 Conclusion

Two trajectory generation methods have been developed: a geometric method and an optimal method.

The geometric method relies on placement of circles which are then connected by straight lines. The resulting trajectories are designed for a vehicle travelling at constant forward speed and lead to the vehicle being taken to its physical limits.

The optimal method trades braking against steering to minimise yaw acceleration throughout the manoeuvre. Manoeuvre boundaries, axis rotation and limitations on vehicle dynamics are expressed as constraints in a series of convex optimisations. Convexity enables the use of a powerful solution algorithm which solves the optimisation problem in a very short time.

The two trajectory generation methods are assessed and compared in Chapter 5 in the context of the overall obstacle avoidance problem.

Chapter 4

Controller design

Make everything as simple as possible, but not simpler.

— Albert Einstein

Methods for calculating a reference trajectory have been developed in Chapter 3. This chapter develops an automatic controller that causes the target vehicle to perform a specified manoeuvre(ISO 3888) by providing control inputs for the steering and braking subsystems.

R. W. Hamming opined that “in the ideal situation the simulation grows into the design, and that in turn flows into the evaluation of the system; it is wrong to separate the three phases” (Hamming 1973, §43). That is broadly the approach used in this work. The controller design model (MexCar, developed in Chapter 2) is used in simulations to develop and refine the controller design. Simulations using both MexCar and the proprietary CASCaDE model are used for evaluation of the resulting system. Consequently, it is not possible to separate entirely all evaluation of simulation results from explanation of the design method. Simulation results that are of importance for making design decisions are presented here. Further simulation results, used to evaluate the performance of the controller, are presented in Chapter 5 with a more detailed discussion.

In the sections below, an analysis of the problem specification is used to develop an architecture for the controller that enables the steering and braking subsystems to be controlled simultaneously. Consideration of the available actuators and their relative merits for controlling aspects of the vehicle’s behaviour then leads to the development of an approach for coping with redundant actuators and a detailed controller structure which makes use of parallel feedforward and feedback control loops. With the control structure in place, controller parameters are obtained by performing simulations and analysing the results, leading to a full description of a control law capable of achieving the desired objectives.

4.1 Architecture and problem decomposition

The aim of the controller is to enable the vehicle to automatically avoid an obstacle. This requires that the vehicle body remain entirely within the manoeuvre space, leading to requirements on lateral position and yaw angle that must be satisfied throughout the exercise. Vehicle position relative to the Earth is not directly controllable by the actuators so the controller will regulate vehicle velocity and acceleration to achieve the desired positional control. This control will be implemented using parallel feedforward and feedback control loops to cause the vehicle to follow a reference trajectory.

The specification in Section 3.1 defines the test-track boundary for the entire length of the manoeuvre. Given this information, it is advisable to check that a feasible trajectory exists before attempting to design control laws. Furthermore, having identified a suitable trajectory, the result may be used as part of the overall control strategy to derive reference profiles for system states and inputs, i.e. feedforward control.

The manoeuvre is specified in terms of position within the fixed Earth axis system whereas the equations of motion for the vehicle are most naturally expressed in terms of velocities measured in the body axis system; translation in space does not directly affect the vehicle dynamics. Although it would be possible to implement a simple controller that acts only according to a pre-defined reference position and measured error, such a design would lead to somewhat arbitrary control of the vehicle velocity. Given that the actuators act most directly on the body dynamics, which are expressed as velocities and accelerations, it is better to use knowledge of the system to explicitly determine the velocity profile that the vehicle is desired to follow. It follows that the controller architecture must map the fixed Earth positional requirements into vehicle velocity requirements. In the absence of disturbances, sensor noise and parametric uncertainty, this mapping could be entirely formulated by defining reference velocities for the vehicle at each point in the manoeuvre. In reality, the vehicle will deviate from any pre-determined velocity profile and it is therefore necessary to include compensation for errors, i.e. feedback control.

Two sub-systems are available to control the vehicle, namely steering and braking. Normal use of these controls by human drivers offers guidance pertaining to the controller structure. Although both the steering and braking systems affect vehicle velocity (speed and direction), the steering system is designed primarily to give the driver control over the vehicle orientation, a positional parameter. In contrast, the brakes are usually used to reduce vehicle speed, entirely independent of position. Intuitively, it may be expected that the steering system will offer better positional control; and the brakes will provide better speed control. Consideration of normal driving behaviour also suggests use of the steering system as the primary means of navigating a route, with the brakes used primarily to ensure that the car can be steered safely.

For a car equipped with a brake-by-wire system, it is possible for the controller to use differential braking, that is application of different forces on each side of the vehicle, or single tyre braking, to provide a level of yaw control beyond that available to human drivers. This is seen increasingly commonly on production vehicles in the working of electronic stability programmes which control

the brakes independently to induce a stabilising yawing moment.

Both the steering and braking systems ultimately operate by generating tyre forces. The forces generated by each system are not independent. The contribution of braking forces to the overall yawing moment depends on the steering angle of the wheels. Traction saturation also couples the systems. Use of steering reduces the traction available for braking and *vice versa*. Thus, in an evasive manoeuvre, there is a trade-off between braking - the only way of removing energy from the system - and steering to avoid an obstacle. As well as a common set of actuators, i.e. the tyres, both actuation subsystems have similar latencies and sample rates. A cascade controller structure is therefore unlikely to be efficient and parallel loops are likely to be more effective. With such a design, it is imperative that the control loops should be well integrated and not conflict with each other. This suggests that it may be beneficial if they co-operate to achieve a common objective, reinforcing the case for using a pre-defined reference trajectory.

The steering and braking inputs are used to control three sets of outputs: the longitudinal, lateral and yaw dynamics of the vehicle. As well as being multi-input multi-output (MIMO), the design of the controller is complicated by virtue of the system having redundant actuators. The use of four brakes to control, at most, three independent vehicle velocities and/or accelerations (longitudinal, lateral and yaw) means that the system is non-square and under-determined. There exists no unique solution for obtaining a specified output. Nor is it possible to assign any single actuator to have primary responsibility for any controlled output.

Classical linear design techniques, which focus on stability of isolated control loops, offer little benefit when designing controllers for highly nonlinear subsystems which exhibit such a degree of interaction between the control inputs. Individual Channel Analysis and Design (ICAD) (O'Reilly & Leithead 1991) does explicitly consider the interaction of parallel loops in a MIMO system. Using an ICAD framework, analysis is usually performed while parallel loops are closed, initially using nominal controllers. The ICAD method does, however, rely on manipulating square matrices, a luxury that does not apply in this case. The technique has been applied to non-square systems (Dudgeon & Gribble 1998, Liceaga, Liceaga & Amézquita 2005) but these are over-determined systems which the authors decompose into series of square subsystems, represented by adequate linear models. Nevertheless, the basic premise of ICAD – that analysis and design of control loops should be performed in a manner that fully considers the effects of, and on, interacting controllers – is entirely sound.

4.1.1 Longitudinal, lateral and yaw control by steering

The steering system provides direct control over vehicle yaw rate. Equation (2.10) specifies a directly proportional relationship for a one-track vehicle executing a steady turn. While the relationship is slightly more complex for a two-track vehicle undergoing braking, the power of the steering system to turn the vehicle is increased, not diminished.

If traction saturation occurs, lateral weight transfer can reduce the total traction available.

Although all load transfered from one side is transfered across the vehicle, traction saturation prevents all the traction lost by the inside wheels from being recovered by those on the outside, which are more heavily loaded during a turn. Longitudinal weight transfer also has a significant effect. For a braking vehicle, weight transfers to the front (steering) wheels, causing the front of the vehicle body to pitch down. This increases the traction available to the steered wheels, at the expense of those at the rear of the car.

Steering the vehicle does not directly affect the vehicle longitudinal and lateral speed measured in the body axis system, other than creating a small increase in drag; a negligible effect. However, the effect of steering the front wheels does have a significant effect on the translational velocity, and hence position, measured relative to the fixed Earth axis. This arises as a consequence of the effect on the vehicle heading angle (Equation (2.1)). The steering system also impacts on control of longitudinal speed due to interaction with the braking system. There are two sources of this coupling. Firstly, because of traction saturation, steering the wheels affects the traction available to the brakes to control the vehicle. Secondly, by altering the direction of the front wheels, the steering system changes the line of action of the braking forces.

4.1.2 Longitudinal, lateral and yaw control by braking

The braking system provides direct control of longitudinal velocity – its primary purpose. At small steering angles, the braking system has negligible effect on the vehicle lateral velocity. At larger steering angles, when the longitudinal wheel forces are more closely aligned with the lateral axis of the vehicle, there is potential for the brakes to have a more direct effect on vehicle side slip.

The braking system can also be used to control yaw rate, but this is subject to limitations. When operating comfortably within the physical limits of the vehicle, braking the wheels on the inside track can be used to increase the magnitude of yaw acceleration and help the vehicle to turn; this is how tanks and other caterpillar-tracked vehicles manoeuvre. However, as the lateral acceleration increases, weight transfers from the inner wheels to those on the outer track (Equation (2.17)). This reduces the braking force that may be applied by the inner track, hence limiting the scope for increasing yaw rate through braking. Conversely, the increased load on the outer wheels enhances the ability to apply a retarding moment and hence decrease yaw rate. Thus the brakes can potentially be used in a servo-actuation role to track yaw rate under benign conditions, but are more suited to stabilisation of yaw rate close to the vehicle's physical limits. During extreme manoeuvres, Electronic Stability Programmes use brakes in this manner to stabilise yaw rate and prevent vehicles from spinning out of control.

Brake force allocation

To solve the problem of allocating control effort between each of the four brakes, a force allocation matrix is used. This is derived from the velocity-based linearisation of the vehicle model (Equation (2.21)). The linearised input matrix $\underline{\mathrm{B}}_f(\boldsymbol{\rho})$ relates the vehicle accelerations to the brake

forces. Inverting the relationship gives a force allocation matrix. $\underline{\mathrm{B}}_f$ is not square, so cannot be inverted directly. Instead, the pseudo-inverse $\underline{\mathrm{B}}_f^{\dagger}(\rho)$ is computed.

A complication arises from the inclusion of the lateral acceleration row in $\underline{\mathrm{B}}_f$. For small steering angles, longitudinal wheel forces have negligible effect on lateral acceleration. When $\underline{\mathrm{B}}_f$ is (pseudo-)inverted, the very small elements associated with these lateral dynamics become very large, dominating the matrix and reducing its usefulness for allocating forces. At low steering angles it would be desirable to neglect the lateral acceleration row completely. However, for larger steering angles, when the front wheels turn towards the lateral axis of the body, the braking forces do have an important effect on lateral acceleration; an effect that cannot reasonably be neglected from the vehicle dynamics. Thus the problem is to eliminate the unwanted dominance of the inverted lateral dynamics terms when steering angles are small, while retaining them at higher angles where they become more important.

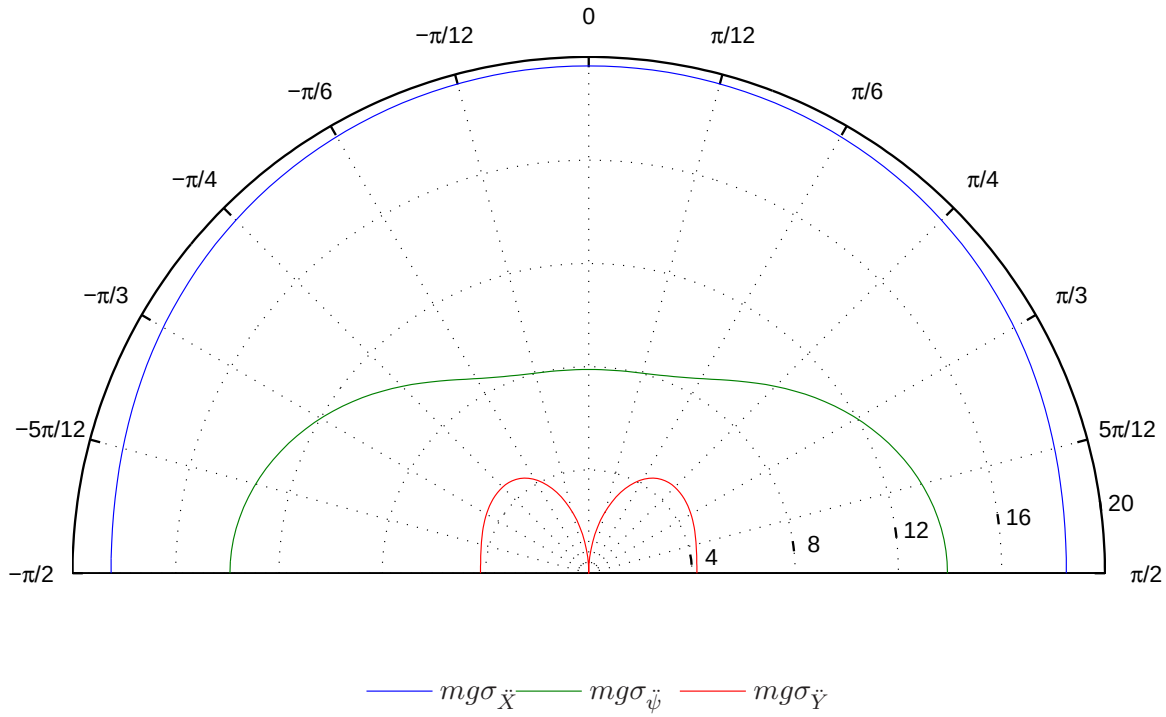


Figure 4.1: Singular values of $\underline{\mathrm{B}}_f$ plotted as functions of steering angle δ [rad]. The singular values are scaled by the weight of the car mg to improve clarity. The blue line shows the singular value $\sigma_{\ddot{X}}$ associated with $\partial\ddot{X}/\partial\mathbf{f}_x$; the green line shows the singular value $\sigma_{\ddot{\psi}}$ relating to $\partial\ddot{\psi}/\partial\mathbf{f}_x$; and the red line shows the singular value $\sigma_{\ddot{Y}}$ linked with $\partial\ddot{Y}/\partial\mathbf{f}_x$.

The solution is to consider a singular value decomposition of the matrix and the effect of singular values on pseudo-inversion. Figure 4.1 shows how the singular values of $\underline{\mathrm{B}}_f$ vary as the steering angle changes. The red line shows how the singular value $\sigma_{\ddot{Y}}$ associated with the lateral dynamics tends to zero for small steering angles and stabilises above $\frac{4}{mg}$ for steering angles greater than $\frac{\pi}{6}$ [rad]. Note that the other two singular values remain far larger than this throughout the entire range of steering angle.

When pseudo-inverting a matrix, a tolerance τ must be specified. Singular values less than τ are equated to zero during the pseudo-inversion procedure. By using τ to eliminate singular values below the chosen value $\frac{4}{mg}$, lateral dynamics terms can be suppressed until the effects become important and the numerical properties of the matrix will not be compromised by their inclusion. It should be noted that this tolerance is many orders of magnitude greater than the default values used in matrix algebra tools such as Matlab and GNU Octave, in which values comparable to the machine precision are typical.

It will be useful to consider the effect that specifying this tolerance has on the (spectral) matrix norm of the force allocation matrix. The matrix norm of the input matrix $\underline{\mathrm{B}}_f$ can be expressed as

$$\|\underline{\mathrm{B}}_f\|_2 = \max \sqrt{\mathrm{eig}\, \underline{\mathrm{B}}_f^H \underline{\mathrm{B}}_f} = \max \boldsymbol{\Sigma} \tag{4.1}$$

where $\boldsymbol{\Sigma} = \sqrt[+]{\mathrm{eig}\, \underline{\mathrm{B}}_f^H \underline{\mathrm{B}}_f}$ is the vector of singular values and the superscript H denotes the (Hermitian) conjugate transpose. The singular values of the pseudo-inverse $\underline{\mathrm{B}}_f^\dagger$ are the reciprocals of each element in $\boldsymbol{\Sigma}$. The matrix norm of the inverse is therefore equal to the reciprocal of the smallest singular value of $\underline{\mathrm{B}}_f$. By eliminating the smallest singular values, the tolerance τ thus places an upper bound on the matrix norm of the pseudo-inversion

$$\|\underline{\mathrm{B}}_f^\dagger(\boldsymbol{\rho})\|_2 = \frac{1}{\sigma} \leq \frac{1}{\tau} = \frac{mg}{4} \tag{4.2}$$

where $\sigma = \min \Sigma : \sigma \geq \tau$.

4.2 Simulations

Description of the controller design process is aided by inclusion of simulation results, for illustration and parameter selection. All simulations in this chapter use the non-linear MexCar model, described in Section 2.8, within Matlab 7 (R2006a) on a 2.4 [GHz] Intel Pentium IV personal computer running Ubuntu GNU/Linux. The simulations are performed using forward Euler integration with a time-step $T = 0.005$ [s]. Data are logged and the simulations controlled by "m-file" scripts. These provide a means of specifying the manoeuvre to be performed, selecting the trajectory generation method to be used and setting parameter values such as controller gains, initial speed, friction coefficient and noise parameters. The scripts also implement the actuator limits and delays specified in Tables 2.1 and 2.2.

Results from simulations of two manoeuvres are detailed here, namely the double lane-changes of ISO 3888 Part 1 and Part 2. The simulations are conducted with an initial forward speed of 80 [km/hr] on a surface with friction coefficient $\mu = 1.0$. Simulations of the Part 1 manoeuvre are used to illustrate the effect of using only feedforward control, and hence to determine design objectives for other elements of the controller. The reference trajectories for these simulations are calculated using both the geometric and optimal trajectory generation methods described in Chapter 3. Simulations of the Part 2 manoeuvre are used for controller tuning and use only the optimal trajectory generation

method.

4.3 Controller structure

Five inputs are available for controlling the vehicle: the front wheel steering angle δ and braking forces $\mathbf{f}_x$ on each wheel. Three independent outputs are potentially able to be controlled directly: longitudinal, lateral and yaw acceleration, or time integrals and derivatives thereof. For successful navigation of the obstacle course, it is also necessary to account for vehicle position relative to the fixed Earth axes.

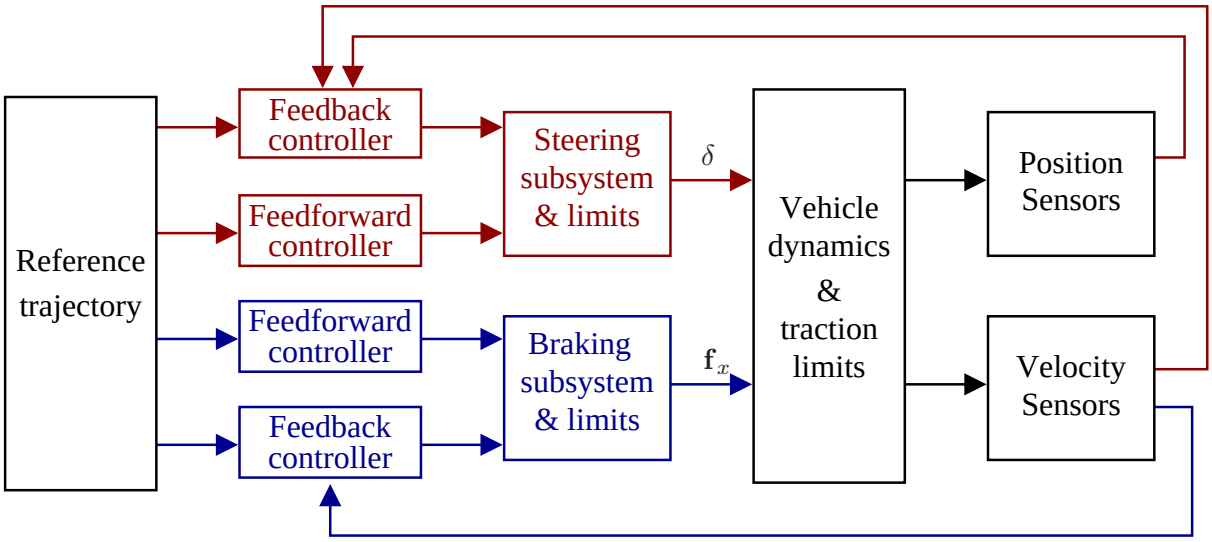


Figure 4.2: Controller structure. The controller uses feedforward and feedback control of the steer-by-wire and brake-by-wire systems to cause the vehicle to follow a reference trajectory.

Figure 4.2 shows the controller structure that is used to generate the input signals for the steering and braking systems. It comprises five significant components, namely: a generator of feasible reference trajectories; a feedforward steering controller; a feedback steering controller; a feedforward braking controller; and a feedback braking controller.

The feedforward controllers both make use of inverse models. The feedforward steering controller calculates a front wheel steering angle to follow a reference yaw rate profile. The feedforward brake controller calculates brake forces to follow a reference longitudinal acceleration profile.

The control signals from each of the feedforward controllers are augmented by the outputs of proportional feedback controllers. The feedback steering controller acts upon position and velocity error signals. The feedback braking controller acts only on velocity errors.

4.4 Feasible trajectory generation

The most important element of the control strategy is reference trajectory generation. If the trajectory is not feasible then the obstacle will not be navigated successfully. Two trajectory generation methods

are described in the previous chapter: a geometric method in Section 3.2; and an optimal method in Section 3.3.

The geometric method produces a reference profile for lateral position $Y_r^{\oplus}(X^{\oplus})$ as a function of longitudinal position. A full set of reference profiles - positions, velocities and accelerations - can be obtained by assuming that:

1. the forward speed is constant;
2. there is zero lateral slip - resulting in zero lateral velocity; and
3. the vehicle heading remains tangential to its path.

The first two assumptions lead directly to reference profiles for the longitudinal and lateral velocities, and hence accelerations. Requiring that the vehicle heading remains tangential to the trajectory produces a yaw angle reference profile $\psi_r(X^{\oplus})$. The geometric trajectory generator calculates a reference yaw rate by forming the product of the forward speed $\dot{X}$, which is assumed to be constant, and the the derivative of the yaw angle with respect to distance, giving $\dot{\psi}_r = \dot{X}\frac{d\psi_r}{dX^{\oplus}}$. Yaw acceleration is derived similarly: $\ddot{\psi}_r = \dot{X}\frac{d\dot{\psi}_r}{dX^{\oplus}}$.

The optimal generation method yields reference positions, velocities and accelerations directly as a result of the optimisation procedure.

4.5 Feedforward steering and braking control

After the feasible trajectory generator, the next most important element of the controller is feedforward control based on inverse models. Feedforward steering is used to produce a steering angle demand based on the reference yaw rate; feedforward braking is used to cause the vehicle to follow the reference longitudinal acceleration profile.

Each of the actuation subsystems is subject to a communication delay: 40 [ms] for steering and 20 [ms] for braking. In the time it takes for the steering system to respond to a control input, a car with a speed of 80 [km/hr] travels almost a metre – a significant distance in a tightly constrained space. To counteract these delays, predicted positions $X_{40}^{\oplus}$ and $X_{20}^{\oplus}$, calculated by assuming constant forward speed, are used for generating the reference inputs to the feedforward loops

$$X_{40}^{\oplus}(t) = X^{\oplus}(t) + 0.04\dot{X}(t) \approx X^{\oplus}(t + 0.04)$$
$$X_{20}^{\oplus}(t) = X^{\oplus}(t) + 0.02\dot{X}(t) \approx X^{\oplus}(t + 0.02)$$

The Ackermann steering angle is a function of the longitudinal and yaw velocity of the vehicle and the length of its wheelbase (Equation (2.10)). As a purely geometric construction, it is independent of the tyre characteristics, which are highly variable and, in most cases, uncertain. It is eminently suitable as a feedforward element for the controller. The steering controller, shown in Figure 4.3, calculates a feedforward steering angle δ_{ff} based on the Ackermann steering angle for a reference yaw rate. The radius of curvature, used to obtain the tangent of δ_{ff}, is calculated using the reference yaw rate and the initial speed of the vehicle. The brake force allocation matrix, developed in

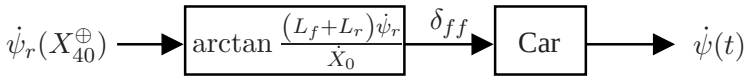


Figure 4.3: A feedforward steering controller uses an inverse bicycle model to calculate the feedforward steering angle δ_{ff} [rad/s] based on the reference yaw rate and initial speed of the vehicle.

Section 4.1.2, relates the vehicle acceleration to the brake forces. The brake controller, shown in

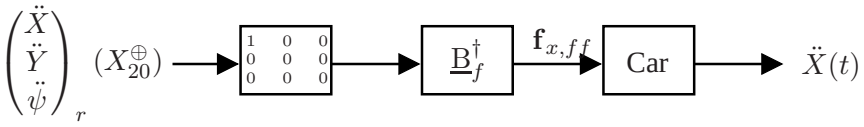


Figure 4.4: A feedforward braking controller uses a pseudo-inverted force allocation matrix to calculate feedforward brake forces $\mathbf{f}_{x,ff}$ [N] based on the reference longitudinal acceleration.

Figure 4.4, calculates feedforward braking forces $\mathbf{f}_{x,ff}$ using the force allocation matrix and reference longitudinal acceleration.

Before proceeding to refine the controller design, it is important to identify the areas that require improvement. Figure 4.5 shows simulation results from the application of the feedforward steering and braking inputs shown above for a car performing the ISO 3888-1 double lane-change. The reference trajectory is obtained using the geometric generation method.

The format of simulation results presented in the sequel follow a common pattern. In each figure:

subfigure (a) shows the trajectory of the vehicle centre of mass (solid blue line) within the manoeuvre boundary (dashed black lines). The positions of the wheels are shown as dotted blue lines. The reference profile is depicted using a black chain line for the reference centre of mass and black dotted lines to show the reference wheel positions;

subfigure (b) shows the brake forces applied to each wheel: front left (blue), front right (green), back left (red), back right (cyan);

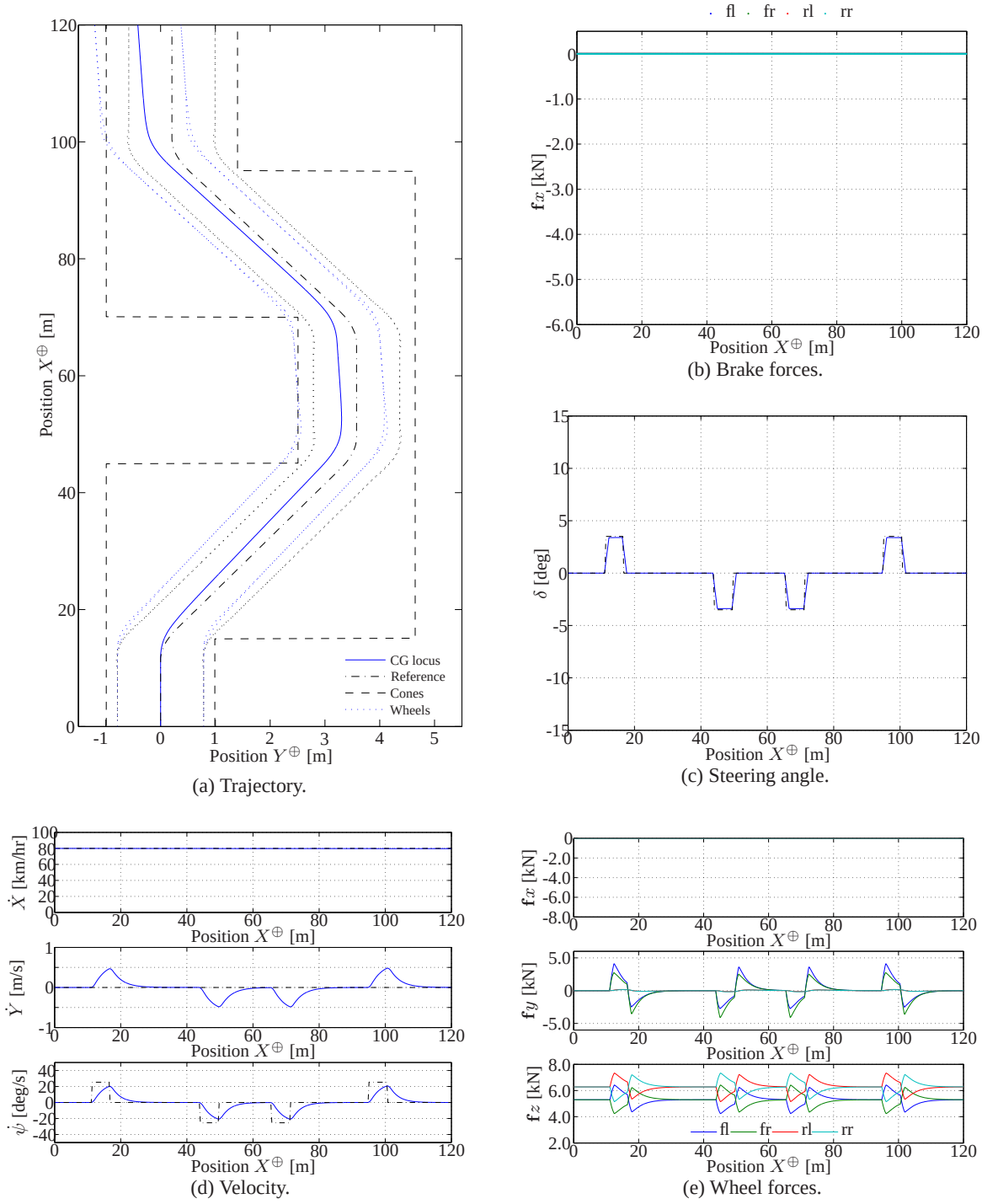
subfigure (c) shows the applied (solid blue) and reference (dashed black) steering angle;

subfigure (d) depicts the actual (solid blue) and reference velocities (longitudinal, lateral and yaw); and

subfigure (e) depicts the longitudinal, lateral and vertical tyre forces, with the same colours as subfigure (b).

Beneath each figure there is a tabulated summary of the simulation to which it pertains, describing: the model used (MexCar or CASCaDE); the manoeuvre (ISO 3888 part 1 or 2); the forward speed [km/hr]; the trajectory generator (geometric or optimal); the surface type and friction coefficient; and whether the signals include disturbances (clean or noisy).

Figure 4.6 shows the results of a similar simulation in which the reference trajectory is generated using the optimal method. The vehicle model is MexCar, the nonlinear two-track model of Chapter 2, with delays and rate-limits imposed between the controller and actuator outputs. It can be seen that the simulation ends while the vehicle is still inside the manoeuvre boundary - the car stops because of the braking action performed during the manoeuvre.



(a) Trajectory.

(b) Brake forces.

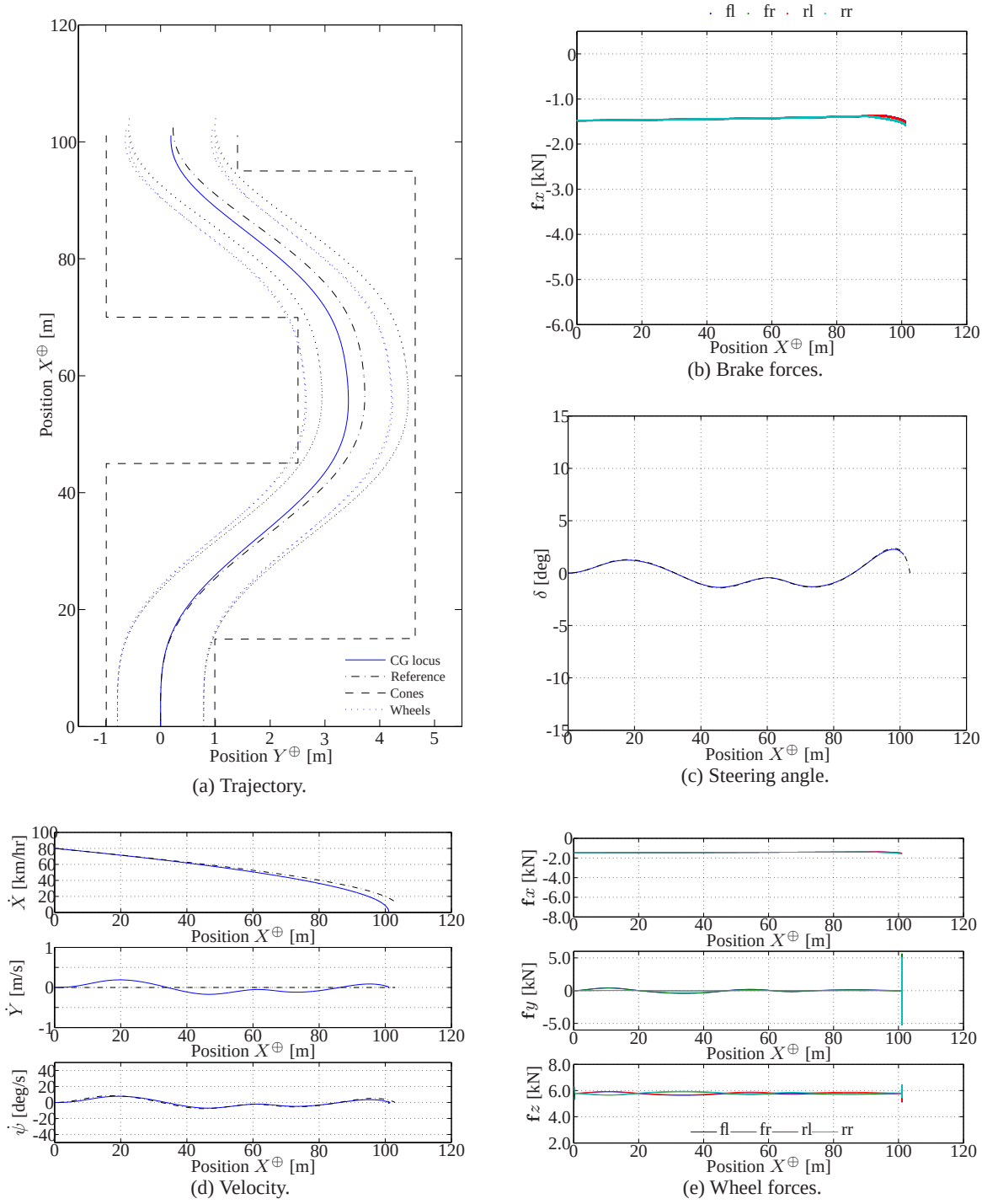
(c) Steering angle.

(d) Velocity.

(e) Wheel forces.

Model:	*MexCar*	**Manoeuvre**:	*ISO 3888-1*	**Speed**:	*80 [km/hr]*
Trajectory:	*geometric*	**Surface**:	*dry asphalt:* $\mu = 1$	**Signal**:	*clean*

Figure 4.5: Simulation of a double lane change manoeuvre using only feedforward control to follow a trajectory produced by the geometric generation method. Reference profiles are shown as black chain lines; simulation outputs as coloured solid lines.



(a) Trajectory.

(b) Brake forces.

(c) Steering angle.

(d) Velocity.

(e) Wheel forces.

Model:	*MexCar*	**Manoeuvre**:	*ISO 3888-1*	**Speed**:	*80 [km/hr]*
Trajectory:	*optimal*	**Surface**:	*dry asphalt:* $\mu = 1$	**Signal**:	*clean*

Figure 4.6: Simulation of a double lane change manoeuvre using only feedforward control to follow a trajectory produced by the optimal generation method. Reference profiles are shown as black chain lines; simulation outputs as coloured solid lines.

The effect of inertia is evident in the yaw response (Figure 4.5(d)) when the geometric reference trajectory generator is used. Combined with short sharp control demands, inertia prevents the car from achieving the desired yaw rate throughout the manoeuvre which leads to significant drift in the final heading angle. For the optimal trajectory, which uses a gentle yaw rate reference profile, inertia poses less of a problem (Figure 4.6(d)).

The controller causes the vehicle to perform a double lane-change whichever reference trajectory generator is used, but the track limits are violated. Additional compensation is required:

1. the yaw tracking response must be enhanced to accommodate inertia and bring the trajectory back towards the reference during the lane-changing part of the manoeuvre; and

2. lateral position and heading angle error must be eliminated during the final lane-keeping phase of the manoeuvre.

4.6 Feedback steering control

Under normal conditions, drivers have no difficulty acquiring and keeping lanes using only the steering wheels, without recourse to differential braking. There is no reason why the single input of the steering wheel angle cannot be used to control multiple outputs. Three are of particular interest for lane changing and lane keeping: lateral position $Y^{\oplus}$, yaw angle ψ and yaw rate $\dot{\psi}$.

The feedback steering loop comprises three parallel controllers, each acting on the front wheel steering angle in concert with the feedforward steering controller. Reference profiles for each of the three controlled outputs are defined as functions of longitudinal position $X^{\oplus}$.

Classical loop-shaping was initially performed, using the velocity-based linearisation of the two-track model, to design loops for each of the three outputs. The attempt was entirely fruitless, with none of the resulting controllers having any redeeming qualities whatsoever. The difficulty appears to have arisen because of the highly nonlinear nature of the system, particularly the interaction between the steering angle and longitudinal forces. Equation (2.11) includes the product of brake forces and trigonometric functions of the steering angle, but this product is not retained in the linearised steering dynamics, where the equations of motion are partially differentiated with respect to the brake forces.

Therefore it was decided to design the controller with the aid of simulation, making direct use of the MexCar model to enable these important nonlinear effects to be adequately accommodated. Using simulations in this manner is somewhat similar to tuning controllers on real hardware. For a successful design strategy, it makes sense to choose a controller type that is amenable to such tuning. With this in mind, PID control is the obvious choice, being significantly easier to tune than more mathematically sophisticated controller forms - hence its widespread use in industry.

However, the outputs are not independent. Yaw rate and yaw angle are, respectively, the derivative and integral of each other with respect to time. Although independent error signals are available for each, a proportional gain on yaw rate serves much the same purpose as a derivative controller on yaw angle. Similarly, a proportional gain on yaw angle is largely equivalent to an integral gain on yaw

rate. Thus simple proportional control on each loop would seem sufficient to get most of the benefit of full PID control on each loop, while being much simpler to tune.

The third output, lateral position, is less closely related to the vehicle dynamics than the yaw parameters. Its inclusion is primarily to assist with accurate lane-acquisition and lane-keeping. The error depends on the history of the vehicle's rotation in the fixed Earth co-ordinate system, so the derivative and integral with respect to time have little relevance to the problem. Consequently, proportional control would also seem appropriate for this parameter.

Proportional controllers are inserted into each of the channels: $K_{\delta,\dot{\psi}}$ from yaw rate error to the steering angle command; $K_{\delta,\psi}$ from yaw angle error to the steering angle command; and $K_{\delta,Y^{\oplus}}$ from lateral position error to the steering angle command. The sum of the individual channels is then added to the feedforward steering angle input. It is desired to obtain a set of three sympathetic gains that will improve overall trajectory-tracking performance without exhibiting significant adverse interaction between the loops.

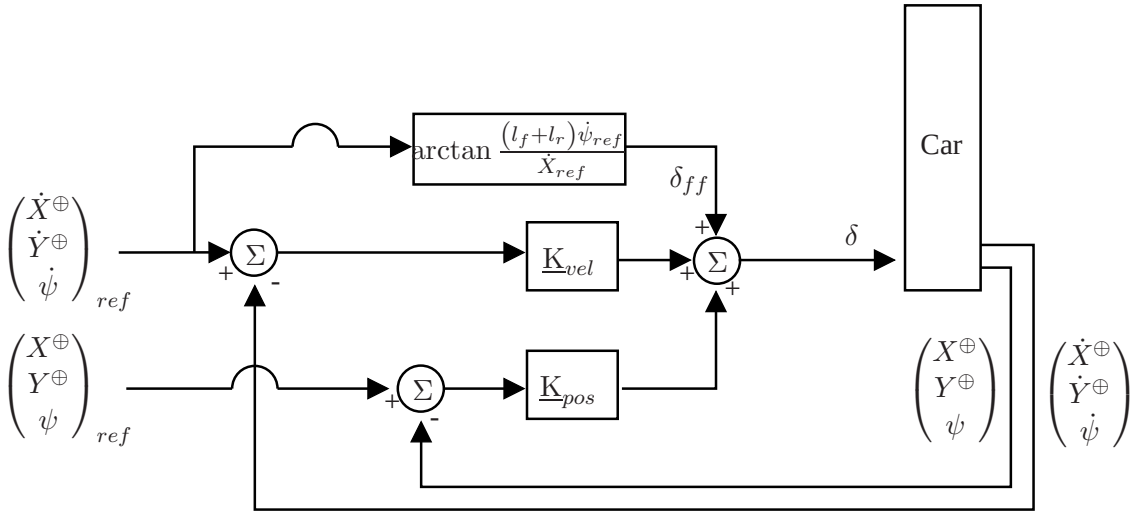


Figure 4.7: Controller (steering elements) with feedforward steering control of yaw rate and feedback steering control of yaw rate, yaw angle and lateral position. The Ackermann steering angle δ_{ff} is obtained from an inverse bicycle model. The feedback control loops have proportional gain matrices $\underline{\mathrm{K}}_{pos} = \left(0, K_{\delta,Y^{\oplus}}, K_{\delta,\psi}\right)$ and $\underline{\mathrm{K}}_{vel} = \left(0, 0, K_{\delta,\dot{\psi}}\right)$.

The steering elements of the controller structure are shown in Figure 4.7.

4.6.1 Gain tuning: simulation-based optimisation

A small number of trial runs are performed with the MexCar model to identify the approximate parameter space in which the controller gains should lie. For each gain, five decades is ample, ranging from small gains which have no effect to excessively large gains that degrade performance. To ensure sufficient resolution within the parameter space, the five decades are divided into 251 exponentially-spaced points.

Five blocks of simulations are performed, each comprising 251 runs. The first three blocks are

used to identify initial gains. A subsequent two blocks are used to refine these values. The total computational time required is less than 7 hours. Each simulation uses the nonlinear MexCar model to perform an ISO 3888-2 double lane change manoeuvre following an optimal trajectory. The controller in each case uses feedback steering control with the feedforward steering and braking loops described in Section 4.5. Within each block, two of the gains are held constant while the third is assigned values from a range spanning five decades. Data from each simulation are used to calculate error vectors: yaw rate error $\mathbf{e}_{\dot{\psi}}(t) = \dot{\boldsymbol{\psi}}_{\mathbf{r}}(\mathbf{X}^{\oplus}(t, K_{\chi})) - \dot{\boldsymbol{\psi}}(t)$; yaw angle error $\mathbf{e}_{\psi}(t) = \boldsymbol{\psi}_{r}(\mathbf{X}^{\oplus}(t, K_{\chi})) - \boldsymbol{\psi}(t)$; and lateral position error $\mathbf{e}_{Y^{\oplus}}(t) = \mathbf{Y}_{r}^{\oplus}(\mathbf{X}^{\oplus}(t, K_{\chi})) - \mathbf{Y}^{\oplus}(t)$, where K_{χ} represents the varying gain. To quantify overall performance for the duration of each simulation, error norms are calculated: $\|\mathbf{e}_{\dot{\psi}}(t, K_{\chi})\|_2$, $\|\mathbf{e}_{\psi}(t, K_{\chi})\|_2$ and $\|\mathbf{e}_{Y^{\oplus}}(t, K_{\chi})\|_2$.

Table 4.1: Gain variation for investigating the effect of the three proportional feedback steering control loops.

	Block 1 ($K_{\delta,\dot{\psi}}$)	Block 2 ($K_{\delta,\psi}$)	Block 3 ($K_{\delta,Y^{\oplus}}$)
$K_{\delta,\dot{\psi}}$	$10^{\{-3.00,-2.98,\cdots,+1.98,+2.00\}}$	0	0
$K_{\delta,\psi}$	0	$10^{\{-3.00,-2.98,\cdots,+1.98,+2.00\}}$	0
$K_{\delta,Y^{\oplus}}$	0	0	$10^{\{-4.00,-3.98,\cdots,+0.98,+1.00\}}$

Blocks 1 to 3 are used to obtain an initial understanding of the effect of each gain on system behaviour. The constant gains are set to zero (Table 4.1) so that the effect of the individual controllers can be seen.

Error responses are plotted in: Figure 4.8, for varying $K_{\delta,\dot{\psi}}$; Figure 4.9, for varying $K_{\delta,\psi}$; and Figure 4.10, for varying $K_{\delta,Y^{\oplus}}$. The graphs show how each of the loops in isolation affects the three errors measurements.

The plots show that in each case the five decades span the full range of interest, from very low gains that produce negligible effect, to very high gains that degrade tracking performance. Distinct global minima are evident in all error responses when $K_{\delta,\dot{\psi}}$ and $K_{\delta,\psi}$ vary, making selection of suitable gains a straightforward matter. The error responses to variation in $K_{\delta,Y^{\oplus}}$ are more ambiguous; tracking displacement in the fixed Earth axis system in which the vehicle rotates while moving adds complexity to the task. These results suggest that the gains relating to yaw response should be tuned first, with the lateral position gain addressed afterwards to fine-tune the system behaviour close to the controller design point.

Determination of a criterion for optimal selection of gains requires the relative importance of each error measurement to be considered. Of the three measurements, lateral position is the most important for ensuring that the vehicle stays within the track limits. Indeed, as long as good tracking of lateral position does not cause excessive degradation in yaw response, which might lead to the front or rear of the vehicle crossing the boundary, it is reasonable to consider only lateral position error. Figures 4.8 and 4.9 support this approach, indicating that little degradation in yaw response would arise as a result of selecting $K_{\delta,\dot{\psi}}$ and $K_{\delta,\psi}$ such that $\|\mathbf{e}_{Y^{\oplus}}\|_2$ is minimised.

The first three simulation blocks reveal the effect of altering the gain in each loop when acting in isolation. Interaction between the loops leads to different optimum values when they are all operating

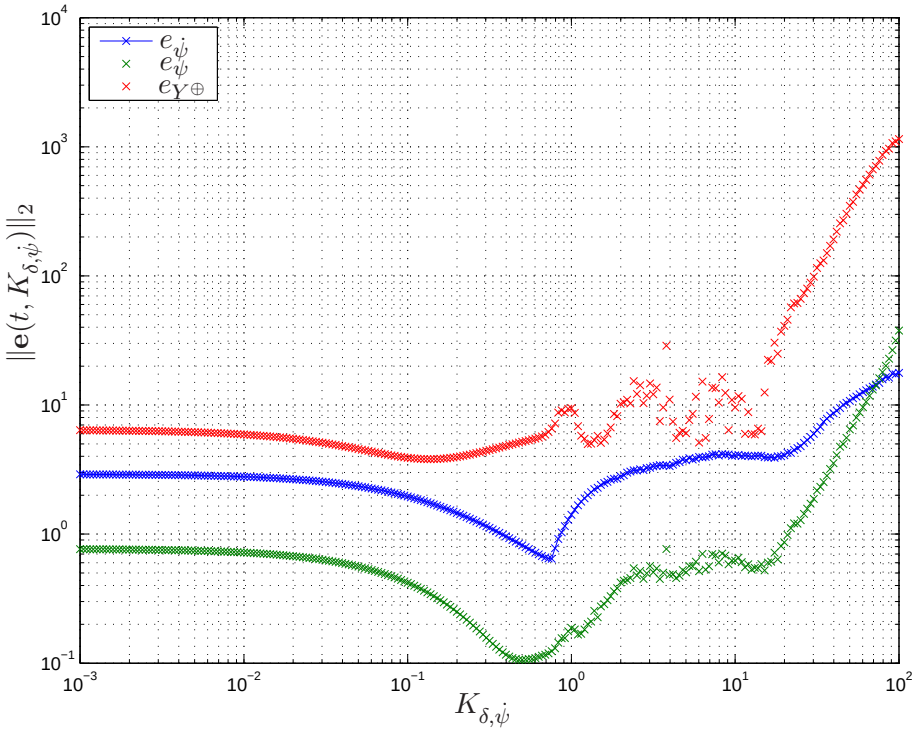


Figure 4.8: Block 1. The effect of changing the steering loop yaw rate gain $K_{\delta,\dot{\psi}}$ on each of the error responses: yaw rate (blue); yaw angle (green); and lateral position (red). The manoeuvre is the ISO-3888-2 double lane-change performed at 80 km/hr following an optimal trajectory, with friction coefficient $\mu = 1$ and controller gains $K_{\delta,\psi} = 0$, and $K_{\delta,Y^{\oplus}} = 0$.

in parallel. To converge upon a (locally) optimal solution, each loop is activated in turn, with the most recently activated loop tuned to accommodate the other active loops. It is arbitrarily decided to activate the yaw rate loop first, using the gain identified previously.

Table 4.2: Gain variation for tuning the proportional feedback steering control loops.

	Block 4 ($K_{\delta,\psi}$)	Block 5 ($K_{\delta,Y^{\oplus}}$)
$K_{\delta,\dot{\psi}}$	0.12	0.12
$K_{\delta,\psi}$	$10^{\{-3.00,-2.98,\cdots,+1.98,+2.00\}}$	0.36
$K_{\delta,Y^{\oplus}}$	0	$10^{\{-4.00,-3.98,\cdots,+0.98,+1.00\}}$

The yaw angle and lateral position loops are refined in succession using a further two blocks of simulations (Blocks 4 and 5, Table 4.2). From Figure 4.8 a gain $K_{\delta,\dot{\psi}} = 0.12$ is chosen to minimise lateral position error $\|\mathbf{e}_{Y^{\oplus}}\|_2$. For Block 4, the lateral position gain $K_{\delta,Y^{\oplus}}$ is set to zero and the yaw angle gain $K_{\delta,\psi}$ is varied. The resulting error responses are shown in Figure 4.11. From this plot, a yaw angle gain $K_{\delta,\psi} = 0.36$ is selected to further reduce lateral position error.

The exercise is then repeated with Block 5 to identify a suitable lateral position gain. The resulting error responses, shown in Figure 4.12, are well behaved (c.f. Figure 4.10) and indicate that selection of a gain in the range $0.2 \leq K_{\delta,Y^{\oplus}} \leq 1.0$ will improve tracking performance, with higher gains having the most impact on the error. Selecting the optimal gain $K_{\delta,\dot{\psi}} = 1.0$ that results in the minimum error $\|\mathbf{e}_{Y^{\oplus}}\|_2$ gives a controller with excellent trajectory-tracking performance (Figure 4.13).

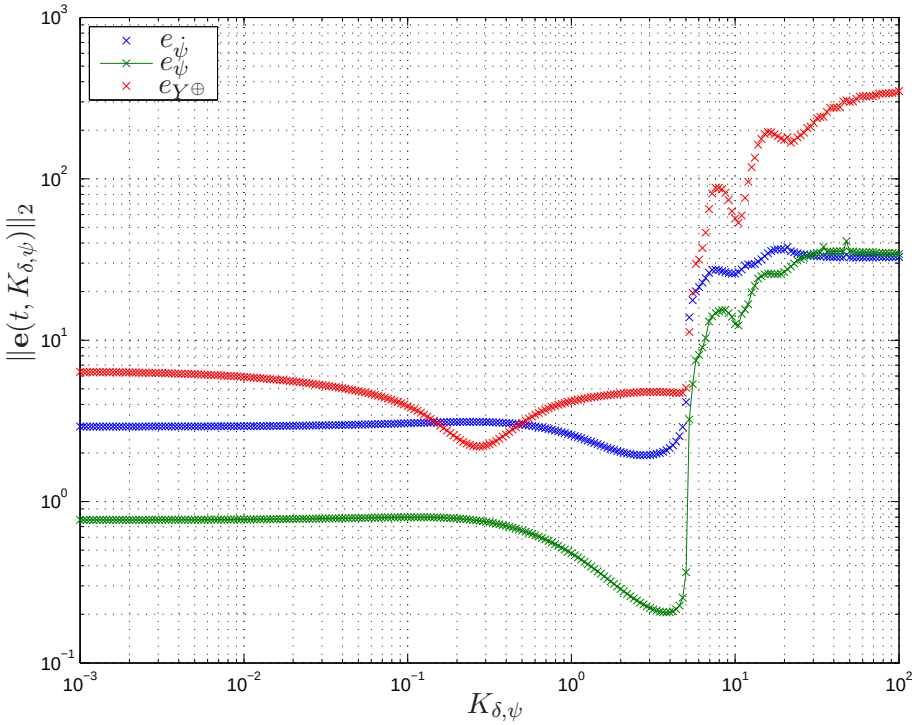


Figure 4.9: Block 2. The effect of changing the steering loop yaw angle gain $K_{\delta,\psi}$ on each of the error responses: yaw rate (blue); yaw angle (green); and lateral position (red). The manoeuvre is the ISO-3888-2 double lane-change performed at 80 km/hr following an optimal trajectory, with friction coefficient $\mu = 1$ and controller gains $K_{\delta,\dot{\psi}} = 0$, and $K_{\delta,Y\oplus} = 0$.

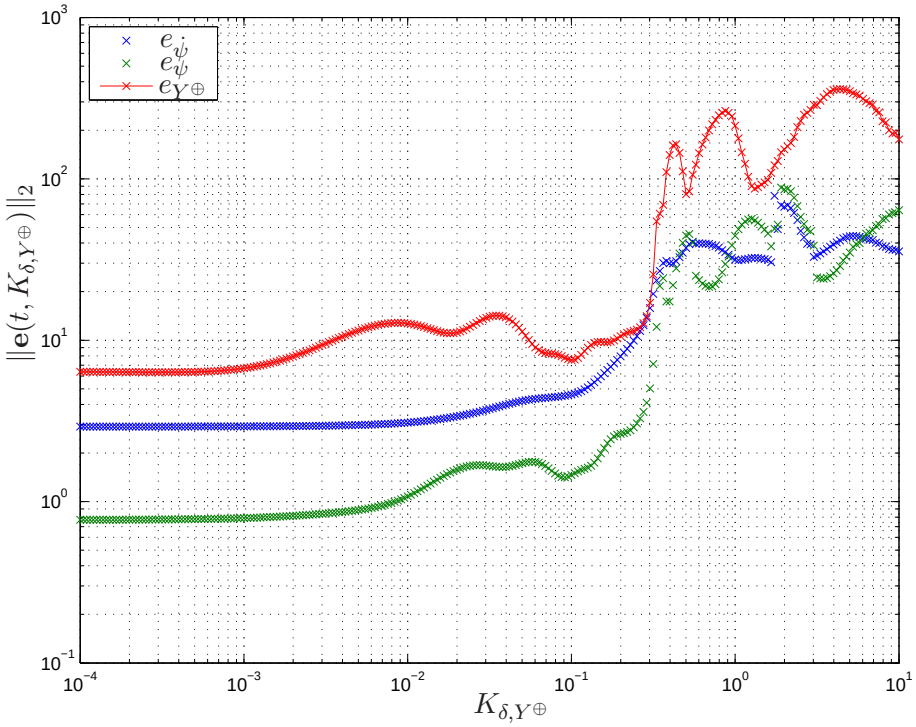


Figure 4.10: Block 3. The effect of changing the steering loop lateral position gain $K_{\delta,Y\oplus}$ on each of the error responses: yaw rate (blue); yaw angle (green); and lateral position (red). The manoeuvre is the ISO-3888-2 double lane-change performed at 80 km/hr following an optimal trajectory, with friction coefficient $\mu = 1$ and controller gains $K_{\delta,\dot{\psi}} = 0$, and $K_{\delta,\psi} = 0$.

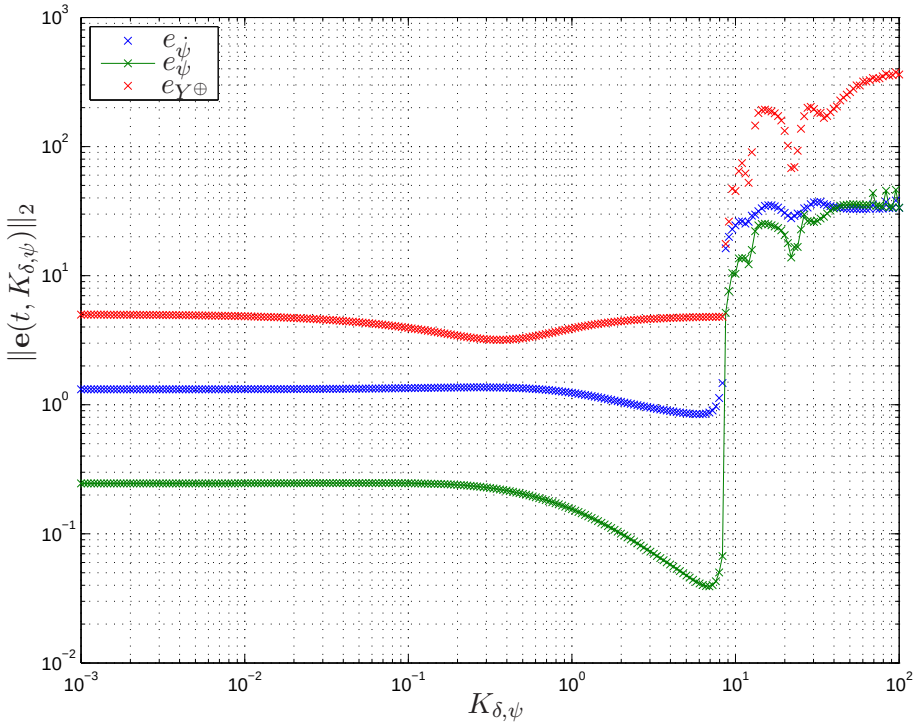


Figure 4.11: Block 4. The effect of changing the steering loop yaw angle gain $K_{\delta,\psi}$ on each of the error responses: yaw rate (blue); yaw angle (green); and lateral position (red). The manoeuvre is the ISO-3888-2 double lane-change performed at 80 km/hr following an optimal trajectory, with friction coefficient $\mu = 1$ and controller gains $K_{\delta,\dot{\psi}} = 0.12$, and $K_{\delta,Y^{\oplus}} = 0$.

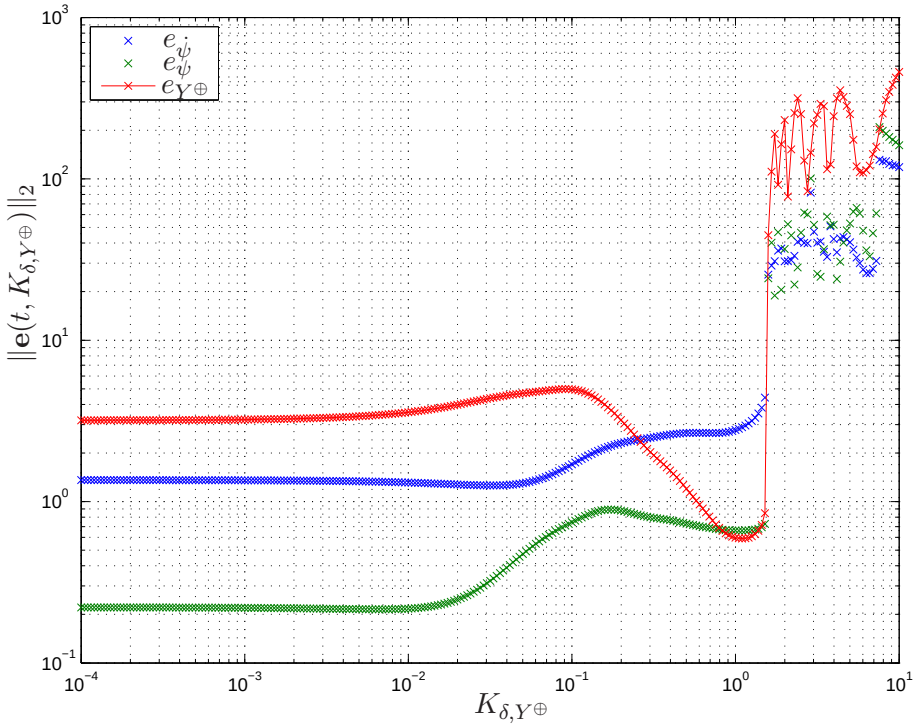
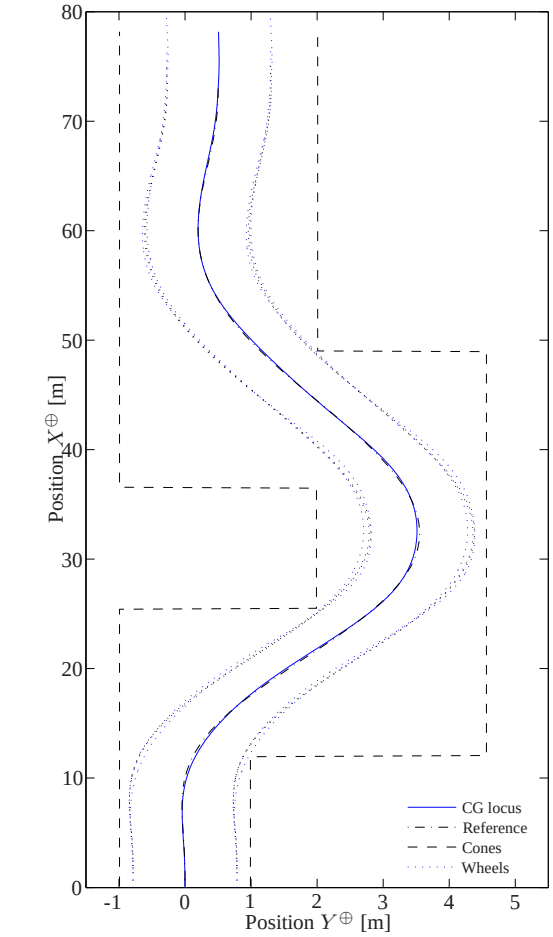
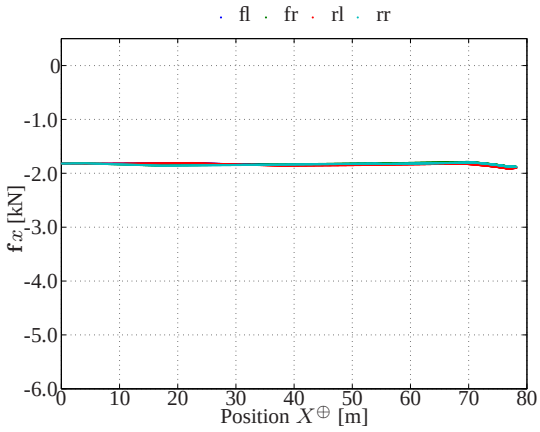


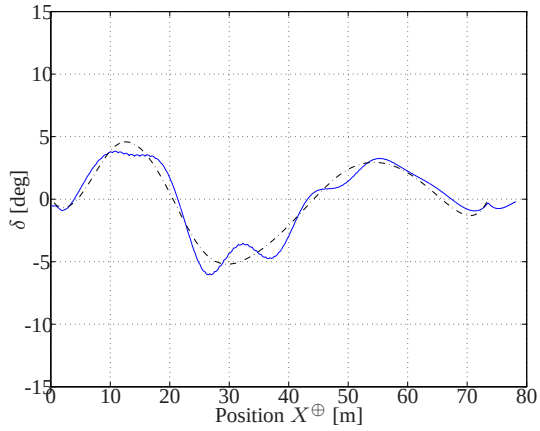
Figure 4.12: Block 5. The effect of changing the steering loop lateral position gain $K_{\delta,Y^{\oplus}}$ on each of the outputs: yaw rate (blue); yaw angle (green); and lateral position (red). The manoeuvre is the ISO-3888-2 double lane-change performed at 80 km/hr following an optimal trajectory, with friction coefficient $\mu = 1$ and controller gains $K_{\delta,\dot{\psi}} = 0.12$, and $K_{\delta,\psi} = 0.36$.



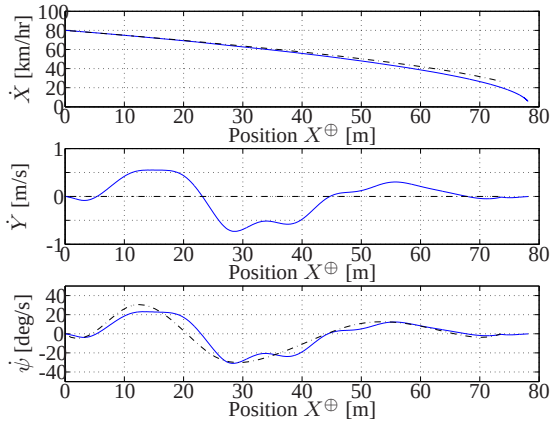
(a) Trajectory.



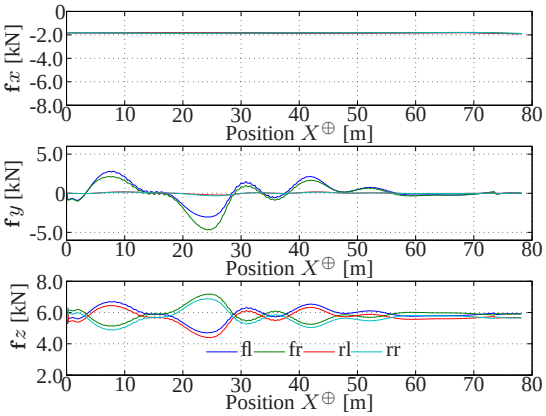
(b) Brake forces.



(c) Steering angle.



(d) Velocity.



(e) Wheel forces.

Model:	*MexCar*	**Manoeuvre**:	*ISO 3888-2*	**Speed**:	*80 [km/hr]*
Trajectory:	*optimal*	**Surface**:	*dry asphalt:* $\mu = 1$	**Signal**:	*clean*

Figure 4.13: Simulation of a double lane change manoeuvre using only feedforward and feedback control to follow a trajectory produced by the optimal generation method. Reference profiles are shown as black chain lines; simulation outputs as coloured solid lines. Controller parameters are $K_{\delta,\dot{\psi}} = 0.12$, $K_{\delta,\psi} = 0.36$ and $K_{\delta,Y^{\oplus}} = 1.0$.

4.6.2 Noise and disturbances

The signals in the previous simulations are clean, i.e. there are no noise or disturbances present in the control and measurement outputs. It is essential that a controller be robust in the presence of the noise and disturbances that it will encounter.

For linear feedback control systems, the effects of disturbances (noise) are usually analysed in terms of (complementary) sensitivity functions. The concept of sensitivity functions has been extended to nonlinear systems (Seron & Goodwin 1996). However, in light of the difficulties encountered when attempting to design the controller by loop-shaping, it is questionable that any such analysis would produce meaningful results. Interaction between control inputs plays a dominant role in the system behaviour; signal errors will interact similarly. Noise and disturbances are therefore considered in the design by way of simulation with deliberate errors injected into signals.

The vehicle measurement data for the real target vehicle are to be provided by an observer (still under development at the time of this research) that makes use of multiple sensors, applies filters and makes corrections to provide best estimates. The precise nature of these corrections are not specified but it is reasonable to suppose that the eventual signals supplied to the controller will be subject to errors following a normal distribution with a variance related to that of the normal signal range.

A noise model is therefore constructed as follows. For each measurement and control signal, the standard deviation $\bar{\sigma}$ is calculated from the data recorded during a nominal manoeuvre undertaken with clean signals (Figure 4.13). For the brake force vector, the mean of the four standard deviations is used. The results in Table 4.3 are obtained. The simulation is then re-run with noise and disturbances

Table 4.3: Standard deviation of clean signals obtained during simulation of the emergency double lane change, ISO 3888-2.

Signal	*Standard deviation* $\bar{\sigma}$	*Units*
$X^{\oplus}$	23.08	[m]
$Y^{\oplus}$	1.09	[m]
ψ	0.10	[rad]
$\dot{X}$	5.95	[m/s]
$\dot{Y}$	0.29	[m/s]
$\dot{\psi}$	0.21	[rad/s]
$\ddot{X}$	0.11	[m/s^2]
$\ddot{Y}$	0.93	[m/s^2]
$\ddot{\psi}$	0.69	[rad/s^2]
$\mathbf{f}_x$	102.21	[N]
δ	0.04	[rad]

applied to each signal $\chi(t)$ such that

$$\chi(t) = \chi(t) + \nu\,\bar{\sigma}\ \mathrm{randn} \tag{4.3}$$

where ν is a scaling factor, arbitrarily chosen as 0.5%, and randn is a normally distributed random

number with zero mean and unity variance calculated using Matlab's default method: Marsaglia's Ziggurat algorithm. Before simulation, the state of the random number generator is set to zero to ensure that results are repeatable. Including noise and disturbance model in the simulation reveals that the controller is over-tuned (Figure 4.14). The gains are too specific to the exact conditions under which they were tuned and the overall controller is therefore insufficiently robust. The steering controller over-reacts to small lateral position errors, leading to wild steering inputs in the region $20 < X^{\oplus} < 60$ [m]. The solution is to reduce the gain $K_{\delta,Y^{\oplus}}$.

Returning to Figure 4.12, a region of degradation is apparent at $K_{\delta,Y^{\oplus}} \geq 1$, where the error norms rise dramatically and oscillate erratically as the gain increases. Meanwhile, it can be seen that the error norms for yaw rate $\|\mathbf{e}_{\dot{\psi}}\|_2$ and lateral position $\|\mathbf{e}_{Y^{\oplus}}\|_2$ cross at $K_{\delta,Y^{\oplus}} = 0.26$. It would seem appropriate to select this point which provides an adequate margin to the region of degradation while balancing the two error measurements.

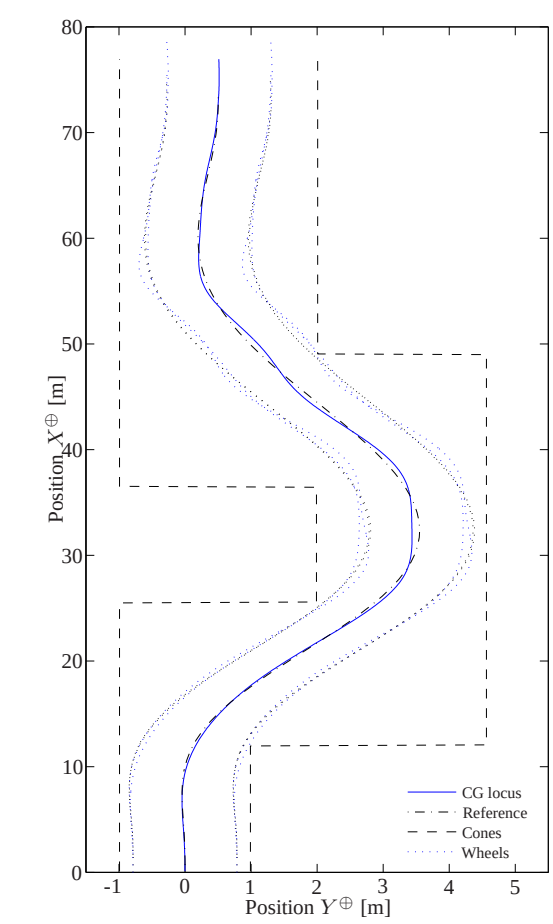
Figure 4.15 demonstrates that relaxing the lateral position gain eliminates the undesirable steering behaviour.

4.7 Feedback braking control

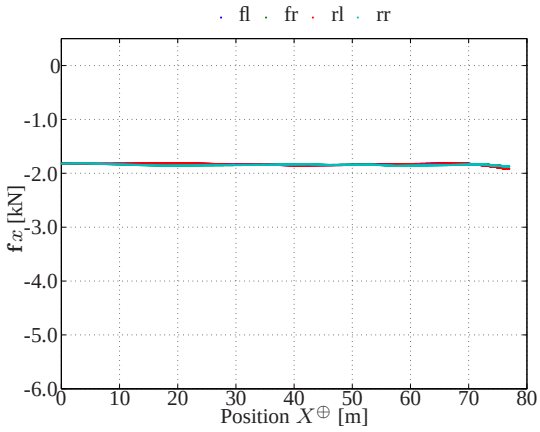
With the steering wheels being used to control yaw and lateral position, it is not possible for them to control yaw rate fully and independently, so there is a possibility of yaw instability occurring if the steering should become too aggressive. Electronic stability programmes are starting to appear on production cars and one is installed on the target vehicle – indeed, it is through the ESP that brake control is achieved. It would therefore be possible to allow the ESP to guard against the vehicle spinning out of control, acting in parallel with the collision avoidance controller. However, it would seem beneficial to integrate the two systems to achieve proper co-ordination rather than leaving the interaction of two systems to chance. Integrating ESP functionality into the collision avoidance controller also allows the conservative production system to be disabled, potentially allowing the car to be taken closer to its physical limits.

The essential characteristics of an integrated yaw stabilisation system are that it should prevent yaw instability and that adverse effects on the performance of the collision avoidance controller should be minimised. Because yaw stabilisation prevents the car from yawing to the full extent demanded by the driver (or automatic control system), it is not possible to entirely eliminate all negative impact on the trajectory tracking task. Maintaining the ability to closely control yaw dynamics comes at the price of sacrificing a degree of path control.

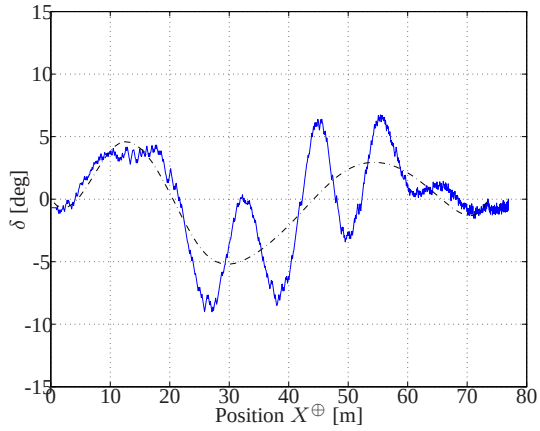
As well as limiting yaw acceleration, conventional yaw stabilisation systems attempt to limit the lateral slip experienced by the wheels. It is unnecessary to do that in this case because the vehicle lateral position and yaw angle are controlled directly by the steering controller. Lateral slip is therefore necessarily constrained implicitly.



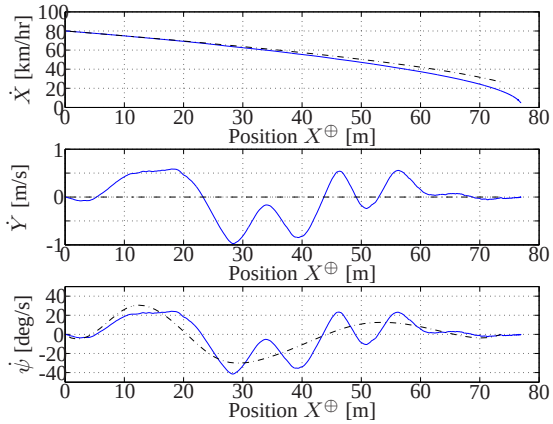
(a) Trajectory.



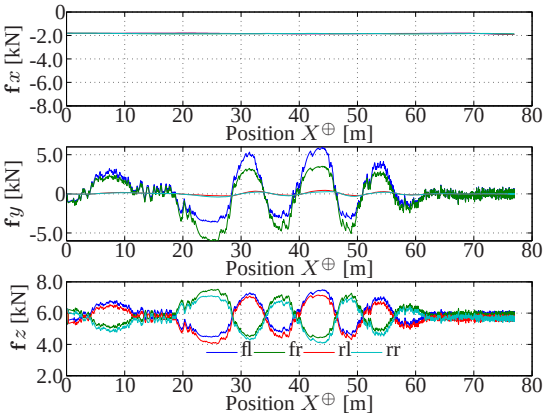
(b) Brake forces.



(c) Steering angle.



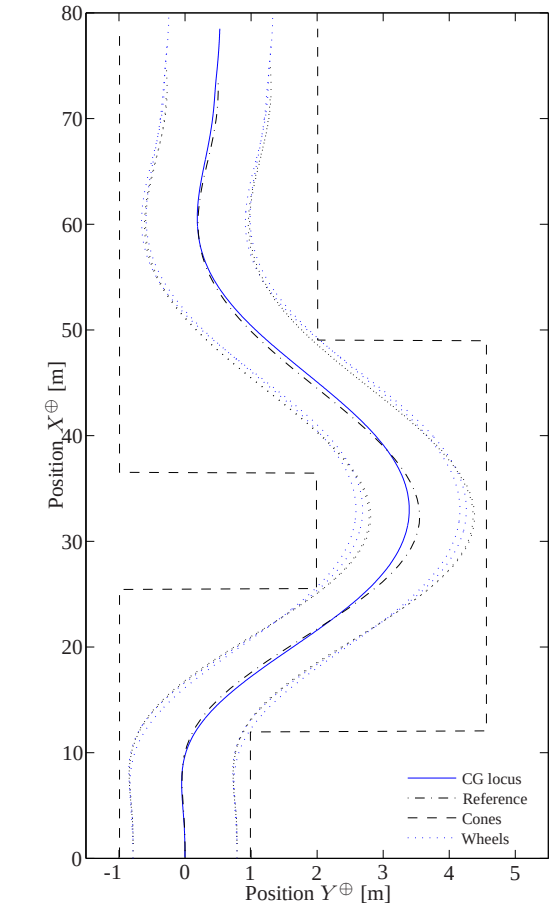
(d) Velocity.



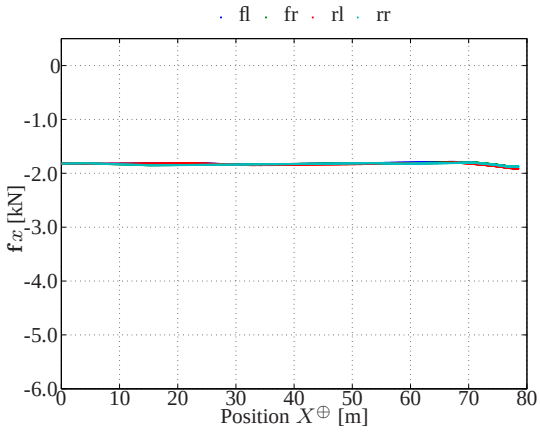
(e) Wheel forces.

Model:	*MexCar*	**Manoeuvre**:	*ISO 3888-2*	**Speed**:	*80 [km/hr]*
Trajectory:	*optimal*	**Surface**:	*dry asphalt:* $\mu = 1$	**Signal**:	*noisy:* $\nu = 0.5\%$

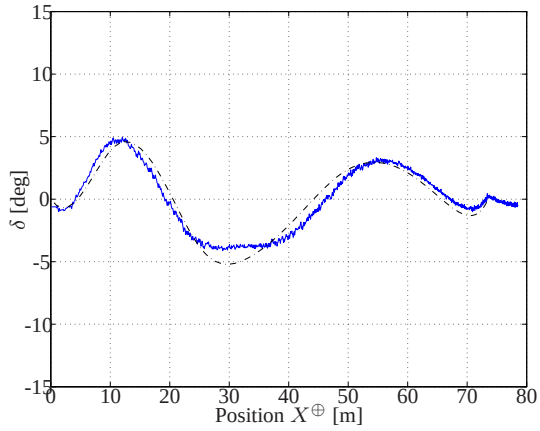
Figure 4.14: Simulation of a double lane change manoeuvre using only feedforward and feedback control to follow a trajectory produced by the optimal generation method. Reference profiles are shown as black chain lines; simulation outputs as coloured solid lines. Controller parameters are $K_{\delta,\dot{\psi}} = 0.12$, $K_{\delta,\psi} = 0.36$ and $K_{\delta,Y^{\oplus}} = 1.0$.



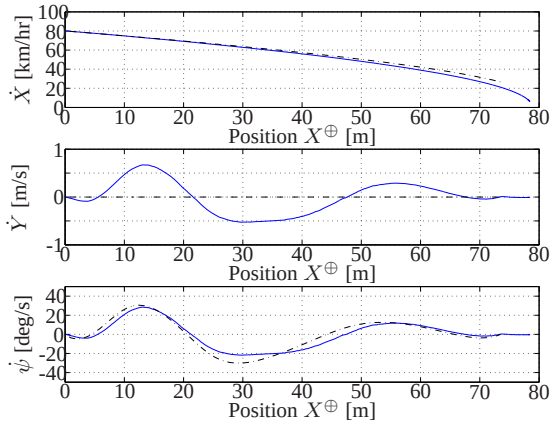
(a) Trajectory.



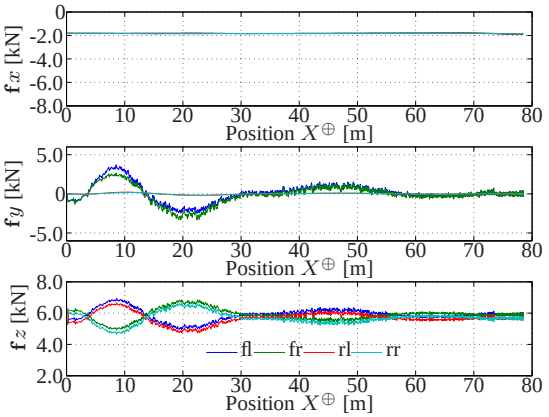
(b) Brake forces.



(c) Steering angle.



(d) Velocity.



(e) Wheel forces.

Model:	*MexCar*	**Manoeuvre**:	*ISO 3888-2*	**Speed**:	*80 [km/hr]*
Trajectory:	*optimal*	**Surface**:	*dry asphalt:* $\mu = 1$	**Signal**:	*noisy:* $\nu = 0.5\%$

Figure 4.15: Simulation of a double lane change manoeuvre using only feedforward and feedback control to follow a trajectory produced by the optimal generation method. Reference profiles are shown as black chain lines; simulation outputs as coloured solid lines. Controller parameters are $K_{\delta,\dot{\psi}} = 0.12$, $K_{\delta,\psi} = 0.36$ and $K_{\delta,Y^{\oplus}} = 0.26$.

The stabilisation loop can potentially act on either an acceleration or velocity signal. Attempting to control yaw acceleration would undoubtedly interfere with the yaw rate and angle tracking performance of the steering control loops. It is therefore preferable to operate on velocity so that the steering and braking controllers can operate on the same yaw rate error signal and complement each other rather than conflict. For improving velocity tracking, the simplest reasonable control law $\mathbf{u}_f$ is a proportional controller acting on the yaw rate error signal

$$\mathbf{u}_f = \underline{\mathrm{K}}_f \times e_f = K_{f,\dot{\psi}} \times e_{\dot{\psi}} \tag{4.4}$$

where $\underline{\mathrm{K}}_f$ is a diagonal gain matrix with non-zero element $K_{f,\dot{\psi}}$ and $\mathbf{e}_f$ is a vector of velocity errors with non-zero element $e_{\dot{\psi}} = \dot{\psi}_r(X^{\oplus}(t)) - \dot{\psi}(t)$, the yaw rate error.

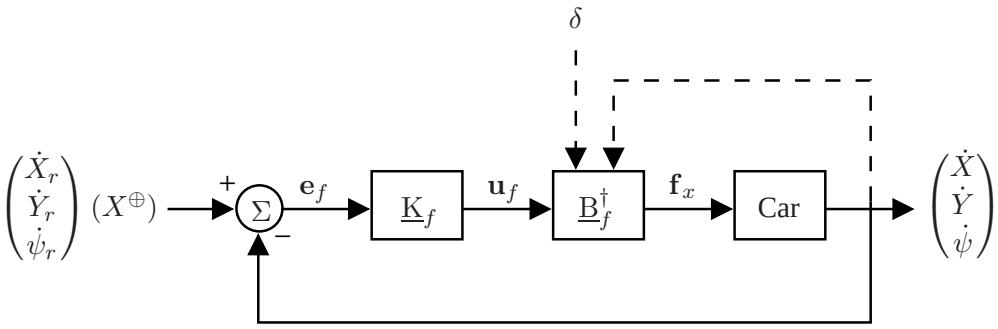


Figure 4.16: Brake loop. A proportional yaw rate feedback controller $\underline{\mathrm{K}}_f = \mathrm{diag}(0, 0, K_{f,\dot{\psi}})$ in series with a force allocation matrix $\underline{\mathrm{B}}_f^{\dagger}$.

As with the feedforward braking loop, the pseudo-inverted input matrix $\underline{\mathrm{B}}_f^{\dagger}$ is used for force allocation. The loop is shown in block diagram form in Figure 4.16. Physical insight simplifies parameter selection for this loop. The car is not equipped with an electronic torque vectoring system so it is not possible to demand positive longitudinal forces from the controller. Nor is there any braking action from a driver which must be eliminated. Hence a negative gain would be meaningless. The minimum gain, zero, contributes nothing towards tracking performance but would have no adverse effects. On the other hand, an extremely high gain would lead to saturation of the brake forces. This would impair the effectiveness of the steering controller, or place greater demands upon it, but would not lead to instability. The ABS prevents the brakes from operating in the unstable slip-traction regime. Even in the absence of an ABS, brakes are inherently stabilising, removing energy from the system whether or not the wheels are locked. The main risk from excessive braking is disruption to the steering control loop. Thus any positive gain is potentially acceptable and it is necessary only to find one that interacts well with the steering system and offers good driving characteristics; a goal which is somewhat subjective.

A good controller should make full use of the available braking power in extreme conditions, but should not apply excessive braking that would interfere with normal operation of the collision avoidance controller. In particular, the brake forces should not saturate excessively, which would prevent the steering controller from operating well. A balance can be struck by considering the

traction limits to which the vehicle is subject. The precise degree of force that would lead to saturation is a complex and generally unknown function of the wheel steering angle. However, in all cases it will be less than or equal to the longitudinal force that would lead to saturation if the wheels are pointing straight ahead. The total longitudinal force is the maximum absolute column sum norm of the brake forces and must be less than or equal to the available traction

$$\|\mathbf{f}_x\|_1 = \sum_{i=1}^{4} |f_{x,i}| \leq \mu\, m\, g \tag{4.5}$$

The maximum longitudinal force on any wheel is the corresponding absolute row sum norm and, assuming that the weight of the car is evenly distributed, must be less than or equal to a quarter of the total available traction

$$\|\mathbf{f}_x\|_\infty = \max_i |f_{x,i}| \leq \frac{\mu\, m\, g}{4} \tag{4.6}$$

The product of these norms is greater than or equal to the square of the spectral norm (Weisstein 2004) hence an upper bound can be placed on the spectral norm

$$\begin{aligned} \|\mathbf{f}_x\|_2^2 &\leq \|\mathbf{f}_x\|_1 \times \|\mathbf{f}_x\|_\infty \\ &\leq \frac{\mu\, m\, g}{4} \times \mu\, m\, g \\ \implies \|\mathbf{f}_x\|_2 &\leq \frac{\mu\, m\, g}{2} \end{aligned} \tag{4.7}$$

Now, the brake forces are the product of the control signal $\mathbf{u}_f$ and the force allocation matrix $\underline{\mathrm{B}}_f^\dagger$, so

$$\begin{aligned} \|\mathbf{f}_x\|_2 &= \|\underline{\mathrm{B}}_f^\dagger \quad \times \quad \mathbf{u}_f\|_2 \\ &\leq \|\underline{\mathrm{B}}_f^\dagger\|_2 \times \|\mathbf{u}_f\|_2 \\ &\leq \|\underline{\mathrm{B}}_f^\dagger\|_2 \times \|K_{f,\dot{\psi}} \ \times \ \mathbf{e}_{\dot{\psi}}\|_2 \end{aligned} \tag{4.8}$$

The requirement to prevent traction saturation, in the absence of steering inputs, therefore translates into a sufficient (but not necessary) upper bound on the gain matrix

$$\begin{aligned} \|\underline{\mathrm{B}}_f^\dagger\|_2 \times K_{f,\dot{\psi}} \times e_{\dot{\psi}} &\leq \frac{\mu\, m\, g}{2} \\ \implies K_{f,\dot{\psi}} &\leq \frac{\mu\, m\, g}{2 \times \|\underline{\mathrm{B}}_f^\dagger\|_2 \times e_{\dot{\psi}}} \end{aligned} \tag{4.9}$$

Recalling Equation (4.2), the norm of the force allocation matrix can be replaced by the expression

$$\|B_f^\dagger\|_2 = \frac{1}{\sigma} = \frac{\tau}{\sigma} \times \frac{1}{\tau} \tag{4.10}$$

With a tolerance $\tau = \frac{\mu m g}{4}$, this gives

$$K_{f,\dot{\psi}} \leq \frac{\sigma}{\tau} \times \frac{\mu m g}{2\tau e_{\dot{\psi}}} = \frac{\sigma}{\tau} \times \frac{2\mu}{e_{\dot{\psi}}} \tag{4.11}$$

The fraction σ/τ can be calculated for any vehicle state, but has a lower bound of unity. It is necessary to estimate the maximum yaw rate error that is likely to occur during normal operation. Running a simulation using the MexCar model for the vehicle following an optimal trajectory through the ISO 3888-2 manoeuvre, with an initial speed of 80 [km/hr] and a friction coefficient of 1, the yaw rate error can be measured. It is found that the maximum error encountered during the simulation of Figure 4.15 is $\|e_{\dot{\psi}}\|_\infty = 0.1366$ [rad/s]. Using Equation (4.11) yields a gain of $K_{f,\dot{\psi}} = 15$.

This gain is added to the controller and the simulation is re-run. The results (Figure 4.17) show that differential braking is employed to counteract yaw rate errors. At their peak ($X^\oplus \approx 30$ [m]), the brakes are just applied to their full extent (Figure 4.17(b)). Thus it is seen that the calculated gain for the brake feedback control loop does cause the brakes to operate precisely as intended. Nevertheless, this operation does still interfere slightly with the steering action. Figure 4.17(d) shows that the yaw rate error is relatively large at $X^\oplus = 30$ [m]. This in turn leads to the vehicle just clipping the final corner of the test-track (Figure 4.17). Yaw stabilisation should therefore only be used if necessary.

4.8 Control law

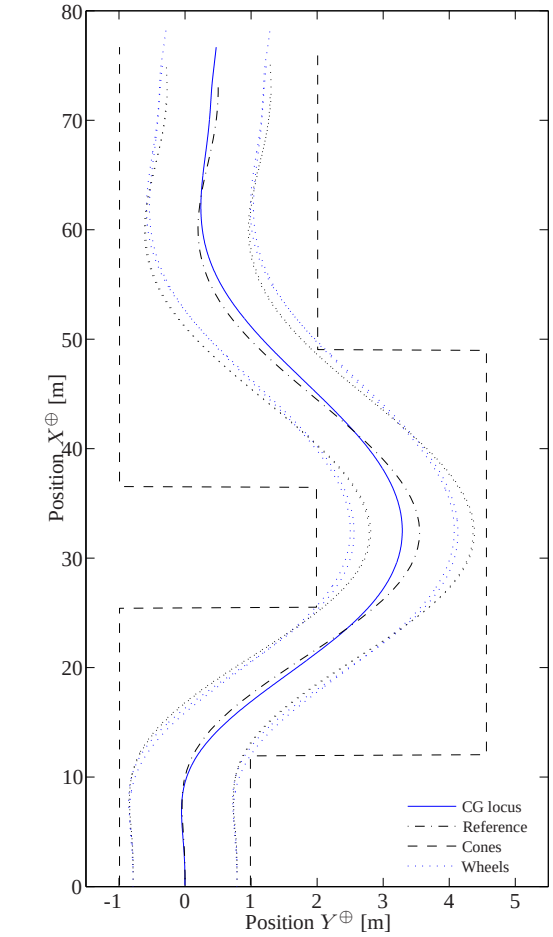
$$\delta_{ff} = \arctan\left(\frac{l_f + l_r}{\dot{X}_0} \times \dot{\psi}_r(X^\oplus_{40})\right) \tag{4.12}$$

$$\delta = \delta_{ff} + 0.12 \times e_{\dot{\psi}} + 0.36 \times e_\psi + 0.26 \times e_{Y^\oplus} \tag{4.13}$$

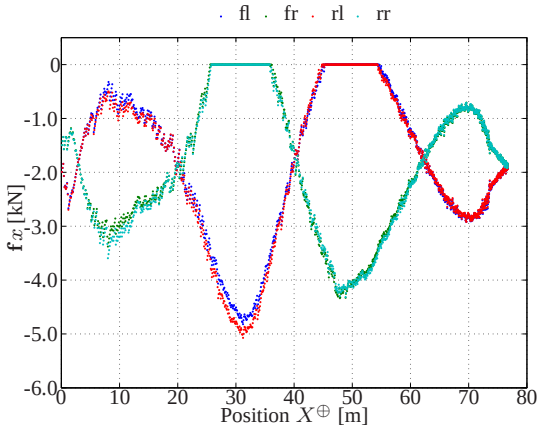
$$\mathbf{f}_x = \underline{\mathrm{B}}_f^\dagger \times \ddot{X}_r(X^\oplus_{20}) \tag{4.14}$$

$$+ 15 \times \underline{\mathrm{B}}_f^\dagger \times e_{\dot{\psi}} \qquad \text{if yaw stabilisation is required} \tag{4.15}$$

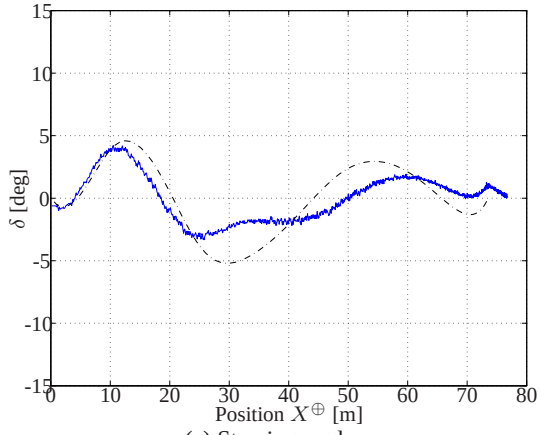
Combining the feedforward steering and braking controllers, the feedback steering controller and the yaw stabilisation feedback braking controller yields the final control law. The controller is shown schematically in Figure 4.18.



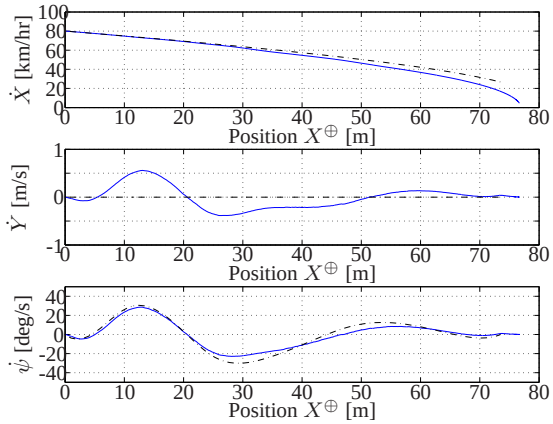
(a) Trajectory.



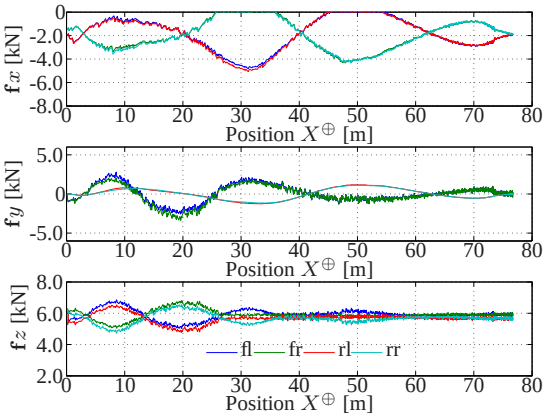
(b) Brake forces.



(c) Steering angle.



(d) Velocity.



(e) Wheel forces.

Model:	*MexCar*	**Manoeuvre**:	*ISO 3888-2*	**Speed**:	*80 [km/hr]*
Trajectory:	*optimal*	**Surface**:	*dry asphalt:* $\mu = 1$	**Signal**:	*noisy:* $\nu = 0.5\%$

Figure 4.17: Simulation of a double lane change manoeuvre using feedforward and feedback control to follow a trajectory produced by the optimal generation method. Reference profiles are shown as black chain lines; simulation outputs as coloured solid lines. Controller parameters are $K_{\delta,\dot{\psi}} = 0.12$, $K_{\delta,\psi} = 0.36$, $K_{\delta,Y^{\oplus}} = 0.26$ and $K_{f,\dot{\psi}} = 15$.

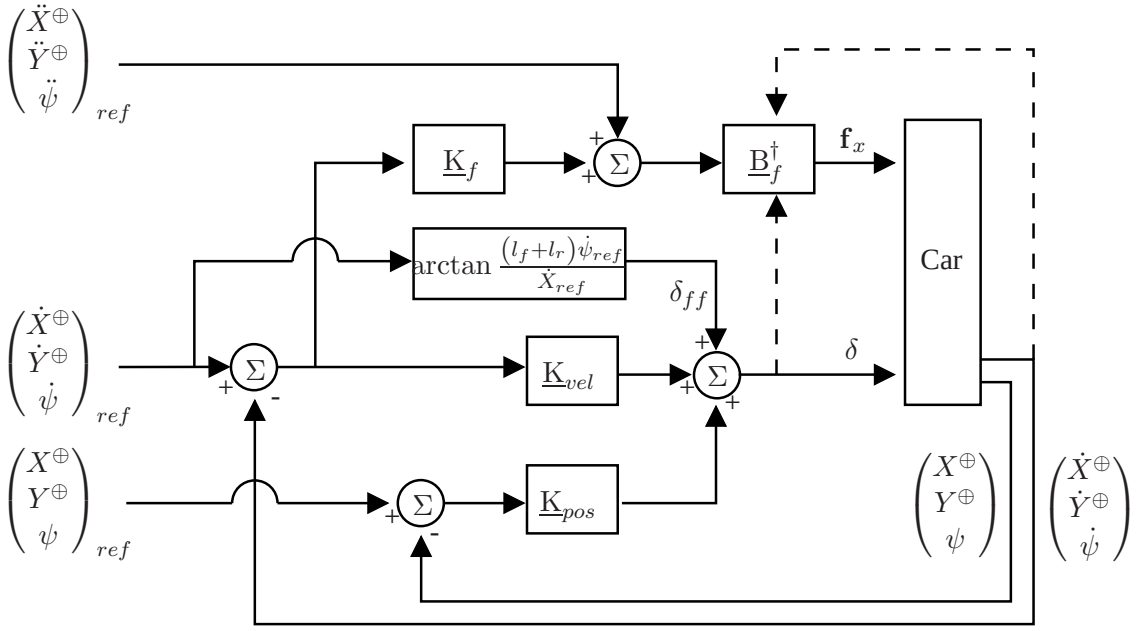


Figure 4.18: Controller with feedforward steering control of yaw rate; feedforward braking control of longitudinal acceleration; and feedback steering control of yaw rate, yaw angle and lateral position. The Ackermann steering angle δ_{ff} is obtained from an inverse bicycle model. The force allocation matrix $\underline{\mathrm{B}}_f^{\dagger}$ is the pseudo-inverse of the (velocity-based) linearised two-track input matrix $\underline{\mathrm{B}}_f$, with tolerance $4/(mg)$ used during the pseudo-inversion. $\underline{\mathrm{B}}_f^{\dagger}$ is updated according to the vehicle state and control inputs. The proportional feedback control loops have gain matrices $\underline{\mathrm{K}}_f = \mathrm{diag}\left(0, 0, K_{f,\dot{\psi}}\right)$, $\underline{\mathrm{K}}_{pos} = \left(0, K_{\delta,Y^{\oplus}}, K_{\delta,\psi}\right)$ and $\underline{\mathrm{K}}_{vel} = \left(0, 0, K_{\delta,\dot{\psi}}\right)$ where the gains have values $K_{f,\dot{\psi}} = 0$ or 15, $K_{\delta,Y^{\oplus}} = 0.26$, $K_{\delta,\psi} = 0.36$ and $K_{\delta,\dot{\psi}} = 0.12$.

4.9 Conclusion

An emergency obstacle avoidance controller has been developed. Reference position, velocity and acceleration profiles are obtained for trajectories generated using either of the methods (geometric or optimal) described in the previous chapter. Feedforward and feedback control are used to integrate steer-by-wire and brake-by-wire functions to control the vehicle, causing it to follow the reference trajectory.

The steering loop comprises: a feedforward element, derived from an inverse bicycle model, to calculate a nominal front wheel steering angle based on a reference yaw rate profile; and feedback elements, consisting of proportional controllers acting on three error signals - yaw rate, yaw angle and lateral position.

A simulation-based method of tuning has been presented for tuning the steering controller parameters. Each of the three control loops is first tuned independently to identify the general behaviour of each and to obtain an initial set of gains. The loops are then activated and tuned in succession to accommodate each other. This method accounts for the highly nonlinear nature of the system and interaction between parallel control loops.

The brake loop comprises: a feedforward element, used to cause the vehicle to follow a reference longitudinal acceleration profile; and a feedback element acting on a yaw rate error signal. Both the feedforward and feedback loops act through a force allocation matrix, constructed by pseudo-inverting an input matrix derived from a velocity-based linearisation of a nonlinear two track vehicle model.

Consideration of the singular values of the brake force allocation matrix and the tolerance used during the pseudo-inversion procedure allows the unwanted dominance of lateral dynamic terms to be neglected when the front wheel steering angle is small. A method for calculating an upper bound on the spectral norm of the force allocation matrix has been described. This is used to define a gain for the brake feedback loop to prevent traction saturation.

A noise model has been developed to inject normally distributed random disturbances and noise into the simulated system. This noise model is based on the standard deviation of normal signal values. Injection of noise during the simulation is used to refine the controller parameters to make the final controller more robust.

Evaluation of the performance of this controller is presented in Chapter 5.

Chapter 5

Controller evaluation

Eppur si muove.

— Galileo Galilei

Two feasible trajectory generation methods have been developed in Chapter 3 and a design for an automatic controller was developed in Chapter 4. In this chapter the trajectory generation methods are compared and the performance of the controller for following these trajectories is evaluated. Conclusions follow in Chapter 6.

5.1 Verification and validation

There are two essential steps for testing any system: verification and validation. Verification demonstrates correct implementation of the specified design. Validation demonstrates that the designed system satisfies its requirements when measured against the real world.

Systems may be validated by experiment or by simulation. Ideally, validation is performed by measurement and comparison against real world experimental data. However, where that is impractical, it is normal to validate against an independent model, preferably one which has itself been validated by experiment.

The MexCar model developed in Chapter 2 can be used to *verify* that the controller functions as designed. It is also suitable for *validating* those aspects of the controller which are designed independently of it, such as the trajectory generators and feedforward control. However, it is not sufficiently independent for validating the complete design because it is used directly to tune the feedback control loops and calculate the force allocation matrix.

In this chapter, DaimlerChrysler's CASCaDE model serves the role of an independent, validated model against which the overall system design may be tested. CASCaDE has been validated over

several years against experimental data obtained from research vehicles operating under controlled conditions on test tracks.

All MexCar simulations presented here are run within Matlab on an Intel Pentium IV personal computer running Ubuntu GNU/Linux using forward Euler integration with a time-step $T = 0.005$ [s], as in the previous chapter. Various auxiliary scripts are used to control the simulations, set parameters and implement delays and actuator limits.

Simulations that use CASCaDE are run on the same platform. The output data rate is the same as MexCar (200 Hz) but the internal integration step size is reported to be $T = 0.001$ [s] and the model uses its own numerical integration routine. A Simulink interface and various parameter files perform equivalent functions to the auxiliary MexCar script files.

5.2 Comparison of trajectory generation methods

Two methods of calculating a feasible reference trajectory are described in Chapter 3: a geometric method and an optimal method. The first step in assessing these methods is to verify that the trajectories they generate are in fact feasible: that paths are found which avoid the test track boundaries and respect the specified vehicle limits.

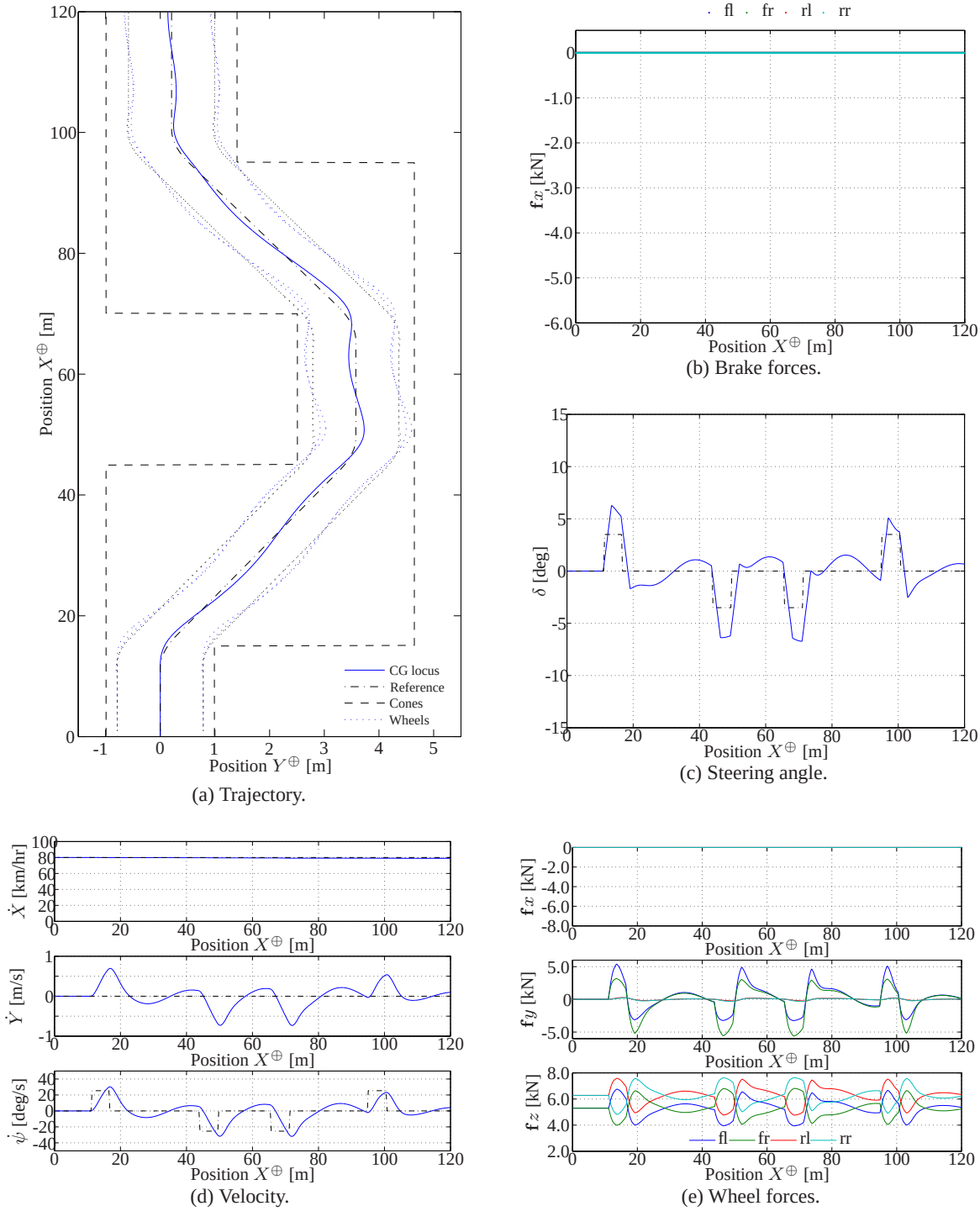
To establish feasibility, trajectories are generated (using both generators - geometric and optimal) for both of the double lane change manoeuvres(ISO 3888 Part 1 and Part 2) at 80 [km/hr]. Following successful trajectory generation, the MexCar model is used to demonstrate that the vehicle would be capable of following each trajectory while subject to the constraints included in the model.

During each simulation, the controller developed in the previous chapter is used to cause the model car to follow the reference trajectory. It is not the aim of this section to evaluate the controller itself. Nevertheless, certain characteristics of the trajectory generation methods are revealed by the control inputs and these are discussed.

5.2.1 ISO 3888-1 double lane-change

In the previous chapter, reference trajectories for the ISO 3888-1 double lane change manoeuvre were generated to evaluate the performance of feedforward controllers acting without feedback compensation. By repeating those simulations using the full controller, it can be shown that the generated trajectories are indeed feasible for a vehicle that is subject to the specified limitations. The results are shown in Figure 5.1 for the geometric trajectory and Figure 5.2 for the optimal one. It can be seen that the car successfully follows both trajectories without violating the test track boundaries.

The controller is identical in both cases, but the very different nature of the reference trajectories leads to markedly different control inputs, and hence significantly different vehicle behaviour. The geometric trajectory demands short, sharp control inputs whereas the optimal trajectory requires prolonged, gentle control. This was also evident in the simulations using feedforward-only controllers (Figures 4.5 and 4.6).



(a) Trajectory.

(b) Brake forces.

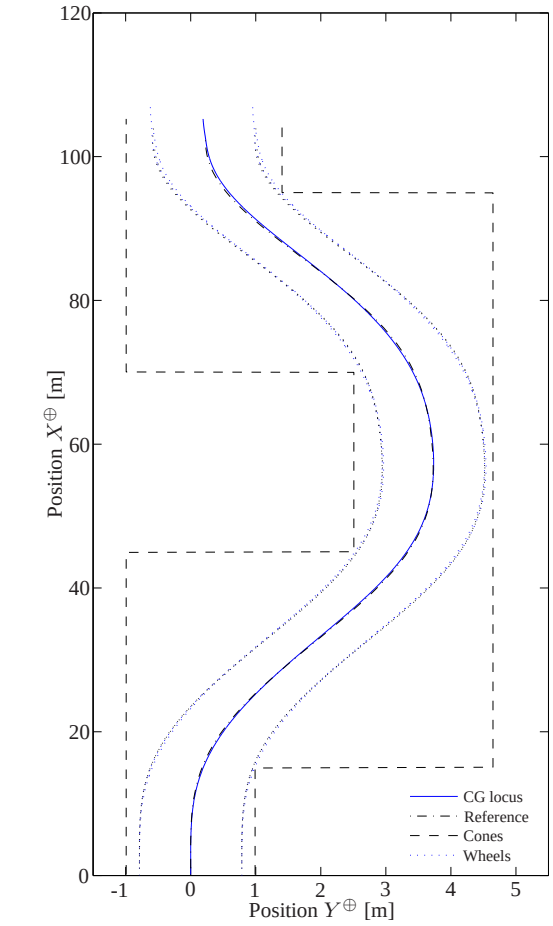
(c) Steering angle.

(d) Velocity.

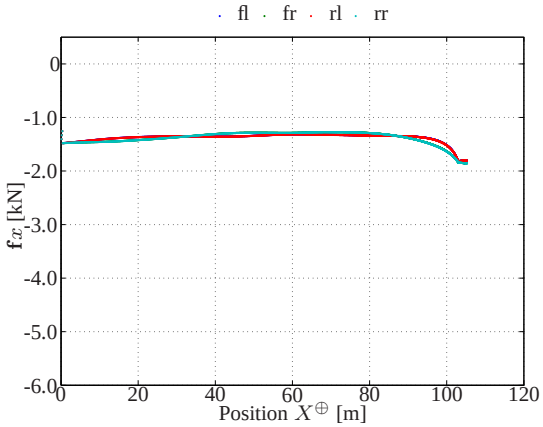
(e) Wheel forces.

Model:	*MexCar*	**Manoeuvre**:	*ISO 3888-1*	**Speed**:	*80 [km/hr]*
Trajectory:	*geometric*	**Surface**:	*dry asphalt*	**Signal**:	*clean*

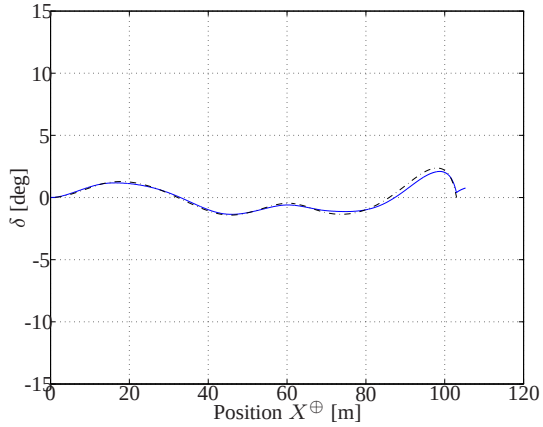
Figure 5.1: Simulation of a double lane change manoeuvre using the feedforward and feedback control to follow a trajectory produced by the geometric generation method. Reference profiles in subfigures (a) and (d) are shown as black chain lines; simulation outputs as coloured solid lines. In subfigure (c), the black chain line shows the feedforward steering profile.



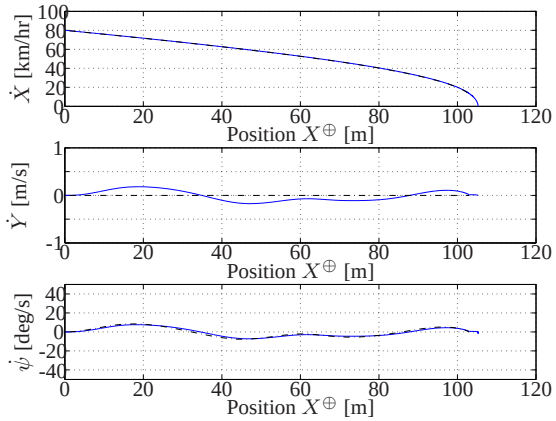
(a) Trajectory.



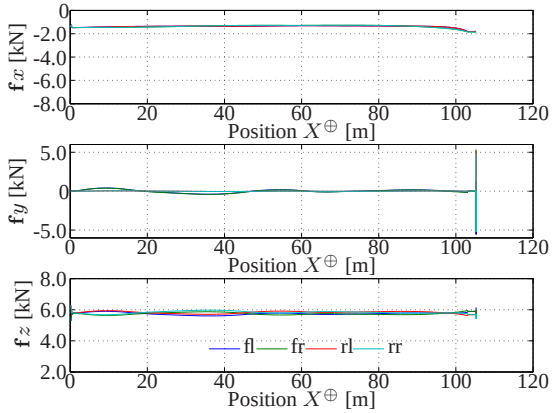
(b) Brake forces.



(c) Steering angle.



(d) Velocity.



(e) Wheel forces.

Model:	*MexCar*	**Manoeuvre**:	*ISO 3888-1*	**Speed**:	*80 [km/hr]*
Trajectory:	*optimal*	**Surface**:	*dry asphalt*	**Signal**:	*clean*

Figure 5.2: Simulation of a double lane change manoeuvre using feedforward and feedback control to follow a trajectory produced by the optimal generation method. Reference profiles in subfigures (a) and (d) are shown as black chain lines; simulation outputs as coloured solid lines. In subfigure (c), the black chain line shows the feedforward steering profile.

Perhaps the most significant difference between the two manoeuvres is the speed at which they are performed. Following the geometrically-derived trajectory, the vehicle exits the obstacle course with its speed virtually unchanged (Figure 5.1(d)). In contrast, the vehicle halts within the test track while following the optimal trajectory. This difference in behaviour arises because of the different braking strategies that are employed. For the geometric trajectory, calculated using a constant-speed assumption, there is no feedforward component to the brake control input (Figure 5.1(b)). Contradistinctly, the optimal trajectory specifies braking throughout the duration of the exercise (Figure 5.2(b)).

When following the optimal trajectory, the steering profile closely matches the feedforward reference steering angle; relatively little feedback correction is required (Figure 5.2(c)). This is not the case for the geometric trajectory; large deviations from the feedforward profile are indicative of the difficulty that the car has in following such an aggressive path (Figure 5.1(c)). The effect is mirrored in the yaw rate response (Figure 5.1(d). High steering angle inputs are required to keep the vehicle close to its reference position. Nevertheless, the controller does manage to achieve this, thus demonstrating the feasibility of the trajectory.

It is also noteworthy that far greater lateral weight transfer occurs during the more aggressive turns associated with the geometric trajectory; a difference of almost 4 [kN] is seen between the wheel load on either side of the car (Figure 5.1(e)) as opposed to 0.5 [kN] for the optimal case (Figure 5.2(e)). This weight transfer is due to the far higher lateral acceleration, and hence rolling moment, experienced by the vehicle when following the geometric trajectory. The difference in lateral acceleration for each case can be inferred from the lateral tyre forces in Figures 5.1(e) and 5.2(e) respectively.

Yaw rate stabilisation

The differing nature of the reference trajectories produced by the two generation methods also leads to differences in the operation of the optional yaw rate stabilisation brake controller. Figures 5.3 and 5.4 show the result of repeating the previous two simulations with the feedback brake control loop activated.

The feedback brake loop has little effect on the vehicle when following the optimal trajectory. The only indication that it is active is the differential braking visible in Figure 5.4(b). Small errors in yaw rate lead to increased brake forces on one side of the car. These are counterbalanced by decreased forces on the other side.

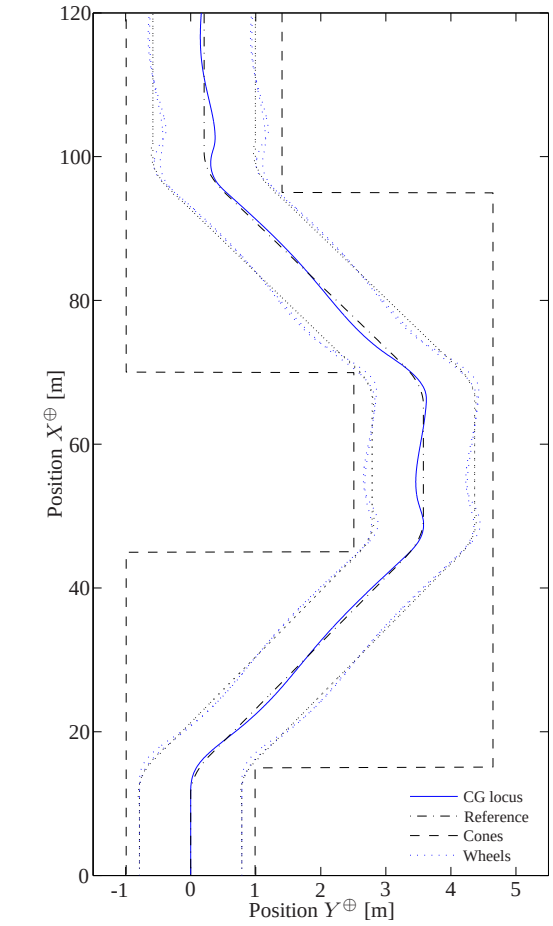
In contrast, feedback braking has a dramatic effect when the vehicle follows the geometric profile. Instead of exiting the course almost as fast as its entry speed, the car slows to less than 40 [km/hr] (Figure 5.3(d)). This is a consequence of large yaw error rates that occur briefly throughout the manoeuvre. Large brake forces applied on one side of the car cannot be counterbalanced by reduced forces on the other (Figure 5.3(b)); the mean brake force is zero before these forces are applied and positive longitudinal control forces cannot be generated without using the engine and differential to

apply drive torque to specific wheels, i.e. torque vectoring. These unbalanced retarding forces lead to the observed deceleration in vehicle speed. This deceleration is exacerbated by the rate limits on the rate of change of brake pressure, which prevent the brakes from releasing as quickly as they are applied. It can be seen that large brake forces continue to be applied on one side of the car while the brakes are being applied on the other to cause the vehicle to turn in the opposite direction. This rate limit effect is not noticeable when following the optimal trajectory (Figure 5.4(b)), where smaller brake forces are observed.

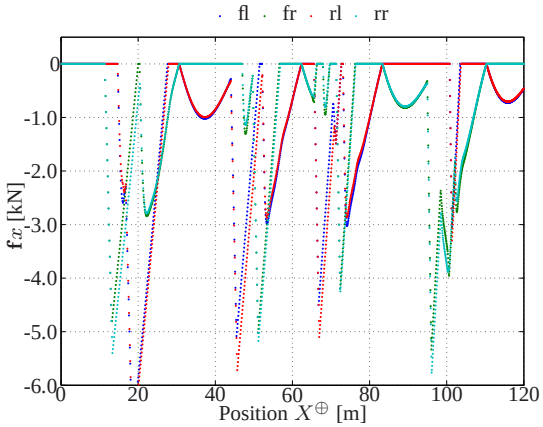
The ISO 3888-1 double lane-change on a dry asphalt surface ($\mu = 1$) is not so demanding that all available tyre traction must be used throughout the manoeuvre. Hence, excessive deceleration during turns does not prevent the vehicle from navigating this course successfully. However, in a more tightly constrained manoeuvre, braking instead of steering can cause failure (c.f. Figure 4.17).

A further feature of the brake controller is visible when following the geometric trajectory but not so readily apparent when following the optimal profile. That is the effect of the force allocation matrix $\underline{\mathrm{B}}_f^{\dagger}$ on the split of front/rear braking. The front wheel steering angle δ appears in the terms of $\underline{\mathrm{B}}_f^{\dagger}$. While following the geometric reference profile, relatively large steering angles ($|\delta| > 5$ [deg]) are generated (Figure 5.3(c)). At these times, the corresponding brake forces (Figure 5.3(b)) exhibit differences between the front and rear wheels. The brake forces are higher on the rear wheels than the front wheels, which are not pointing straight ahead. Although a small effect, this is a potentially useful feature that arises automatically from the use of the force allocation matrix. Shifting the braking burden to the rear wheels leads to a consequent reduction in interference with the steering action of the front wheels. However, there is a limit on how much of the braking effort can be transferred to the rear of the car because weight transfers to the front wheels during deceleration, reducing the traction available to the back tyres. This effect is not seen in the simulations because MexCar does not include pitch dynamics.

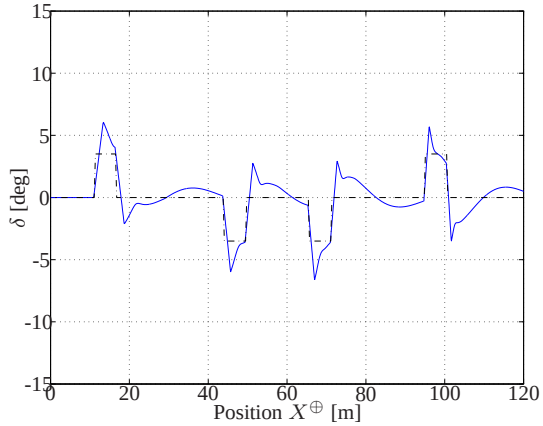
It should be noted that the combined lateral and longitudinal forces exhibited in Figure 5.3 are greater than would actually be achievable by the car. The MexCar model does not explicitly account for the reduction in traction that arises from combined slip conditions. Nevertheless, the simulation does demonstrate the qualitative behaviour of a controller attempting to follow the geometric trajectory.



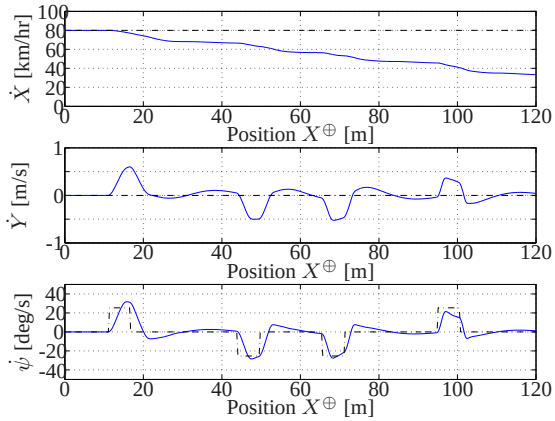
(a) Trajectory.



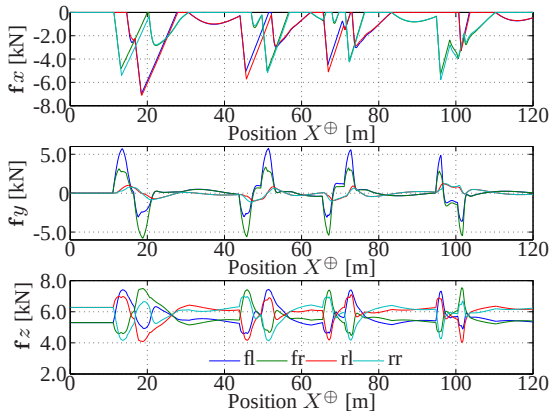
(b) Brake forces.



(c) Steering angle.



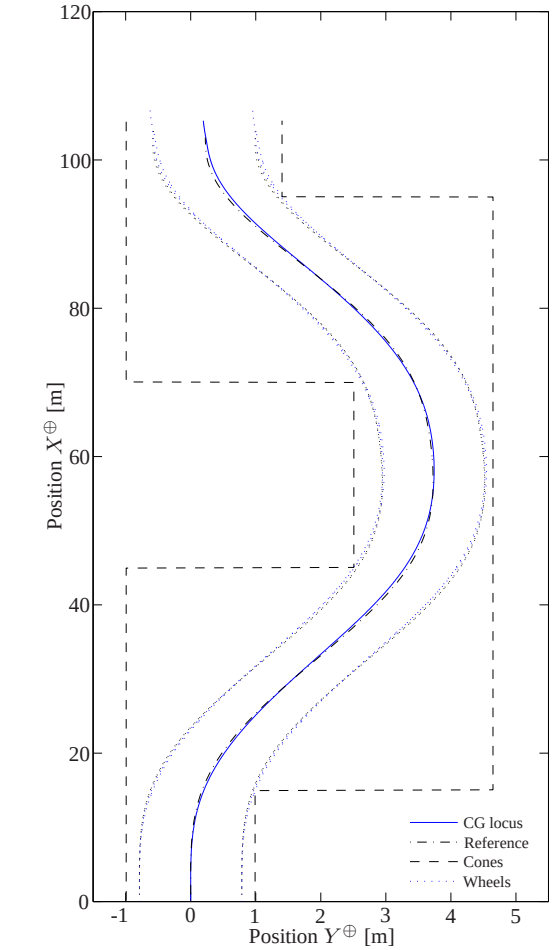
(d) Velocity.



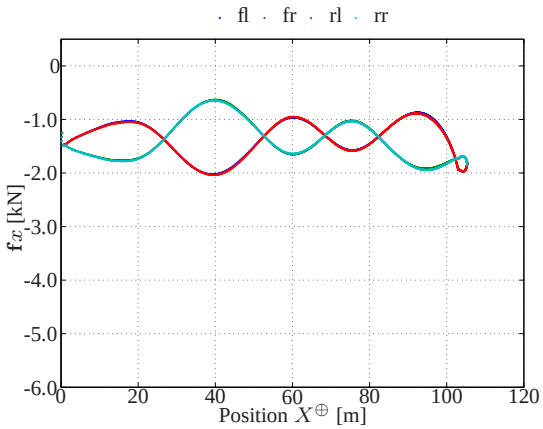
(e) Wheel forces.

Model:	*MexCar*	**Manoeuvre**:	*ISO 3888-1*	**Speed**:	*80 [km/hr]*
Trajectory:	*geometric*	**Surface**:	*dry asphalt*	**Signal**:	*clean*

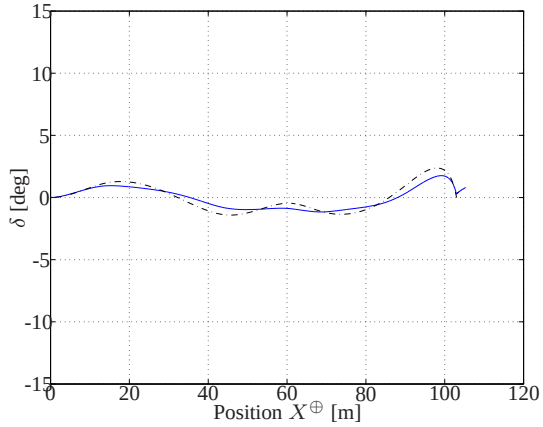
Figure 5.3: Simulation of a double lane change manoeuvre using the full controller with yaw rate stabilisation to follow a trajectory produced by the geometric generation method. Reference profiles in subfigures (a) and (d) are shown as black chain lines; simulation outputs as coloured solid lines. In subfigure (c), the black chain line shows the feedforward steering profile.



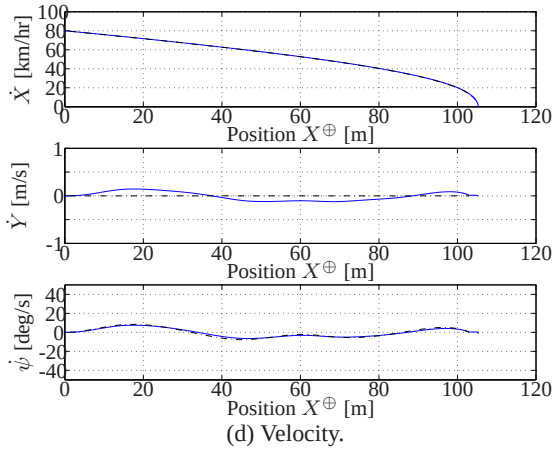
(a) Trajectory.



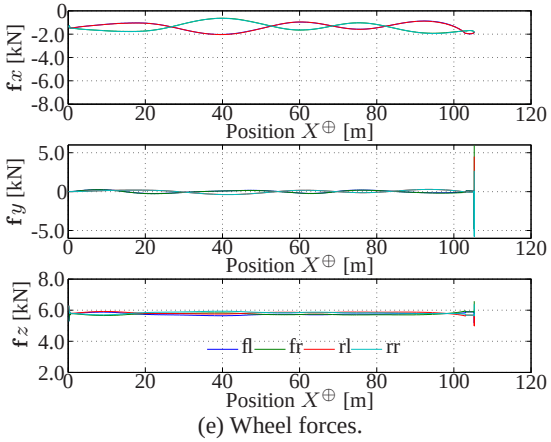
(b) Brake forces.



(c) Steering angle.



(d) Velocity.



(e) Wheel forces.

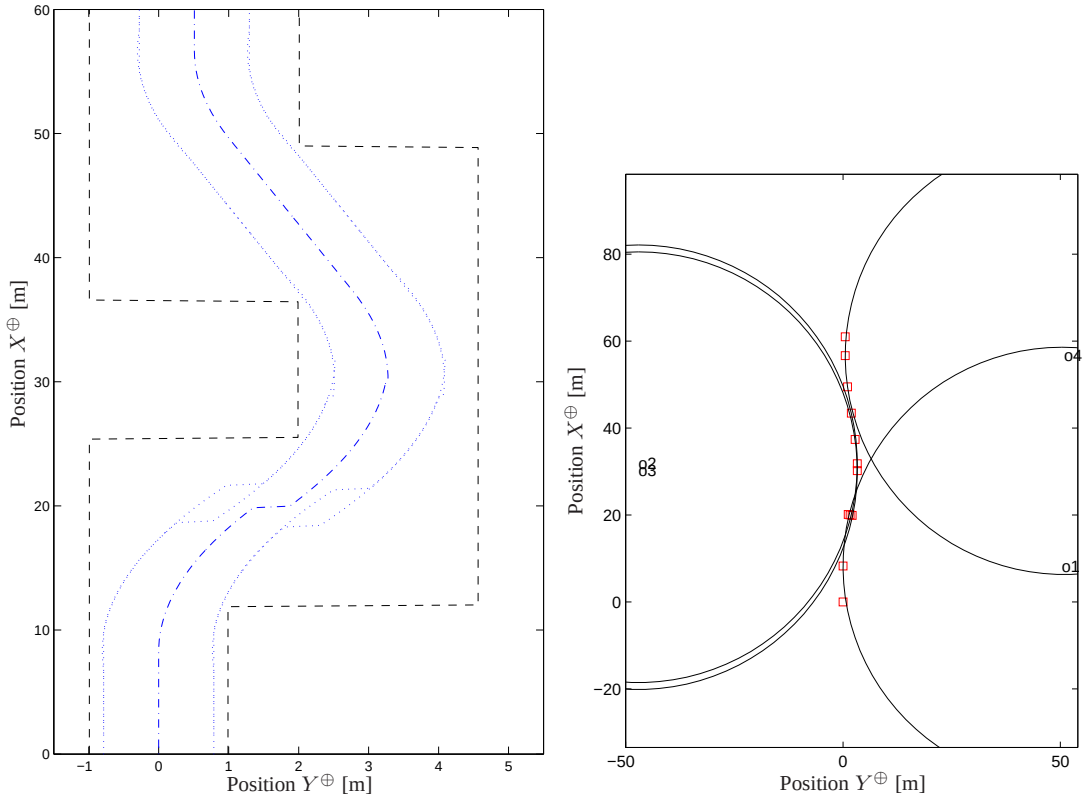
Model:	*MexCar*	**Manoeuvre**:	*ISO 3888-1*	**Speed**:	*80 [km/hr]*
Trajectory:	*optimal*	**Surface**:	*dry asphalt*	**Signal**:	*clean*

Figure 5.4: Simulation of a double lane change manoeuvre using the full controller with yaw rate stabilisation to follow a trajectory produced by the optimal generation method. Reference profiles in subfigures (a) and (d) are shown as black chain lines; simulation outputs as coloured solid lines. In subfigure (c), the black chain line shows the feedforward steering profile.

5.2.2 ISO 3888-2 emergency double lane-change

The Part 2 emergency double lane-change is more challenging than its non-emergency counterpart because of far tighter constraints. Both trajectory generation methods have been used to determine reference profiles for navigating this test track.

In Chapter 3, the optimal method was explained with an example trajectory for a vehicle travelling at 80 [km/hr]. Meanwhile, the geometric method was used to calculate a trajectory for a vehicle travelling more slowly at 60 [km/hr]. The reason for this disparity is that the geometric method is incapable of finding a feasible path at the higher speed. The original design goal of the geometric method was to perform a *single* lane change (Bevan, Gollee & O'Reilly 2007*a*) at 80 [km/hr] while deliberately taking the vehicle to its physical limits. Figure 5.5 shows the result of attempting to use the geometric method to generate a reference trajectory for a vehicle travelling at 80 [km/hr] through the emergency course. The radius of each circle used to generate the waypoints is too large for the manoeuvre space. The circles intersect, with the centre O_3 being placed before O_2. This creates a discontinuity at longitudinal distance $X^{\oplus} \approx 20$ [m]. The method cannot produce a feasible reference trajectory without modification. In contrast, the optimal method continues to work well even for



(a) Geometric reference trajectory.

(b) Waypoints.

Figure 5.5: Reference trajectory for an ISO 3888-2 emergency double lane-change manoeuvre generated using the geometric method for a vehicle travelling at 80 [km/hr] and the waypoints that produce it (c.f. Figure 3.6). The circles cannot be placed without intersecting, which causes a discontinuity in the generated trajectory. This trajectory is not feasible.

this more difficult task, as evidenced by successful trajectory generation during the controller design process outlined in the previous chapter (Figure 4.17).

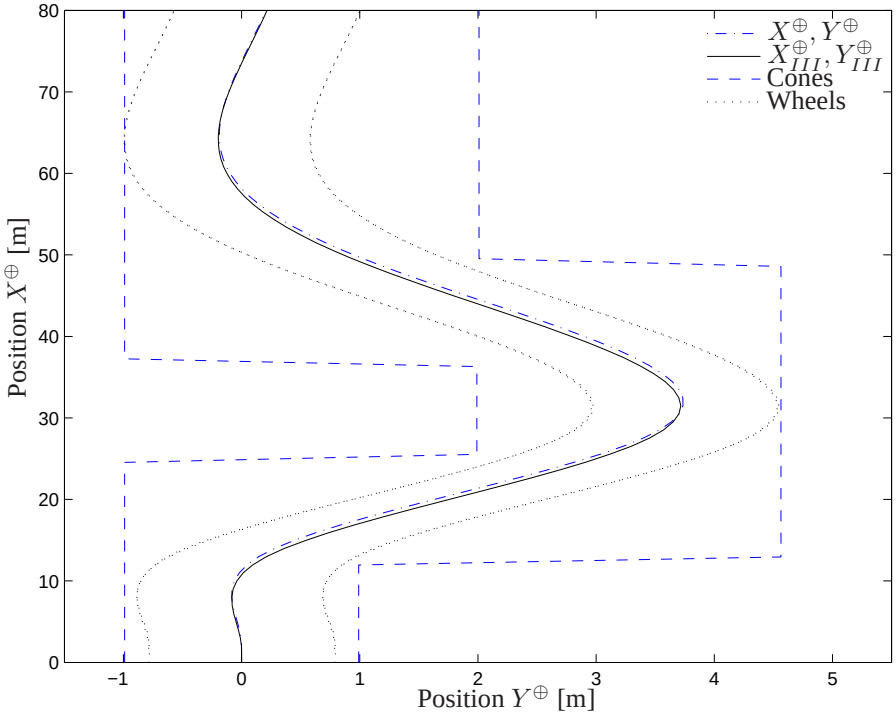


Figure 5.6: Optimal reference trajectory for the ISO 3888-2 emergency double lane change manoeuvre at 157 [km/hr].

The optimal trajectory generator is in fact capable of calculating trajectories for speeds as high as 157 [km/hr] (Figure 5.6) without exceeding the track limits. However, to make use of such a trajectory, the controller would have to be re-tuned to operate at twice its design speed and cope with excessive traction saturation. This high speed trajectory uses the full width of the road. A prominent feature is the initial turn to the left to create more space for later turns. Like a good driver, the optimal trajectory generator takes account of the road ahead.

5.3 Verification of controller performance using MexCar

The aim of the controller is to cause the vehicle to perform an ISO 3888-2 emergency double lane change manoeuvre with an initial speed of 80 [km/hr] on dry, smooth asphalt with state measurements provided by an on-board observer making use of various sensors.

Verification of the controller performance is undertaken using the MexCar model. It is first demonstrated that the controller performs as required under ideal conditions and the results are used to highlight aspects of how the controller works. It is then shown that the controller copes adequately in the presence of noise, disturbances and parameter uncertainties.

5.3.1 Operation of feedback control loops

In the previous chapter, Figures 4.5 and 4.6 show that sole use of feedforward control as implemented is insufficient to perform even the relatively undemanding double lane-change specified in ISO 3888-1. Section 4.5 identifies two specific requirements for the feedback control loops: improvement of yaw tracking during lane-changing and correction of lateral and yaw position for lane-keeping. To verify that the inclusion of feedback control does indeed correct these problems, the simulations are repeated using the full controller. Figures 5.1 and 5.2 demonstrate that both lane-changing and lane-keeping functions are improved by the addition of feedback control. Improvement in yaw rate tracking is seen in the yaw rate response for the geometric trajectory (Figure 5.1(d)) when feedback is included (c.f. Figure 4.5(d)) The peak yaw rate for the geometric trajectory attains – and slightly exceeds – the demanded reference value. As a result, the vehicle turns sufficiently to track the reference path and the trajectory is thus followed far more precisely.

Also in Chapter 4, the overall performance of the controller for tracking the more demanding ISO 3888-2 emergency double lane change manoeuvre is shown in Figure 4.15. The controller configuration uses feedforward steering and braking and feedback steering. The feedback braking loop for yaw rate stabilisation is inactive ($K_{f,\dot{\psi}} = 0$). A noise and disturbance level $\nu = 0.5\%$ represents disturbances and noise, using the model of Equation (4.3). Figure 4.15(a) shows that the vehicle completes the manoeuvre successfully, avoiding the test-track limits. The path is smooth and the vehicle follows it closely. Thus the controller does perform as required.

A high correlation is seen between lateral velocity and yaw rate in Figure 4.15(d). It is possible for a car to experience either lateral slip without yawing, such as when driving in a cross-wind, or to yaw without experiencing lateral slip, such as when turning at low speed. However, at high speeds both effects are usually experienced together. It is difficult to force a car to yaw without slipping laterally, which would require exact cancellation of the front and rear lateral forces. That these forces do not generally cancel is seen in the graph of $\mathbf{f}_y$ in Figure 4.15(e). Because the controller does not attempt to track lateral velocity, the correlation with yaw rate does not cause any problem; but if the controller were to be modified to do so, it would be necessary to ensure consistency between the lateral and yaw velocity reference profiles.

Figure 4.15(e) shows that the weight transfers to the outside wheels during each turn. This accords with the behaviour of real vehicles, instilling some confidence in the correctness of the simple weight distribution model of Equation (2.17).

5.3.2 Effect of increasing noise and disturbances

The robustness of the controller with respect to noise and disturbances is investigated by altering the noise level ν applied during simulation.

Figure 5.7 shows the effect of a five-fold increase in the noise level, to $\nu = 2.5\%$. Figures 5.8 and 5.9 show two further doublings of the noise level, to $\nu = 5\%$ and $\nu = 10\%$ respectively. In all cases, despite significant disturbances, the controller is seen to perform well, tracking the reference

trajectory without any signs of instability in the controller response. However, the vehicle is seen to brush against the final corner, clipping it by a couple of centimetres for the most severe noise level of 10% (Figure 5.9).

If noise and disturbances of such severity are encountered by the controller, it would be prudent to include a small margin when generating the reference trajectory; a soft limit a few centimetres inside the true boundary to accommodate random perturbations. Provided that the trajectory is not at the limit of what is feasible, this can be accomplished simply by increasing the virtual width of the car or narrowing the track dimensions within the trajectory generator.

A further doubling of the disturbance level ($\nu = 20\%$, Figure 5.10) shows that there are limits to how much the controller can withstand. The vehicle starts to depart significantly from the reference trajectory at $X^{\oplus} = 20$ [m]. If such severe disruption were believed likely, then it would be necessary to perform additional filtration of measurement and error signals before use.

5.3.3 Effect of loss of traction

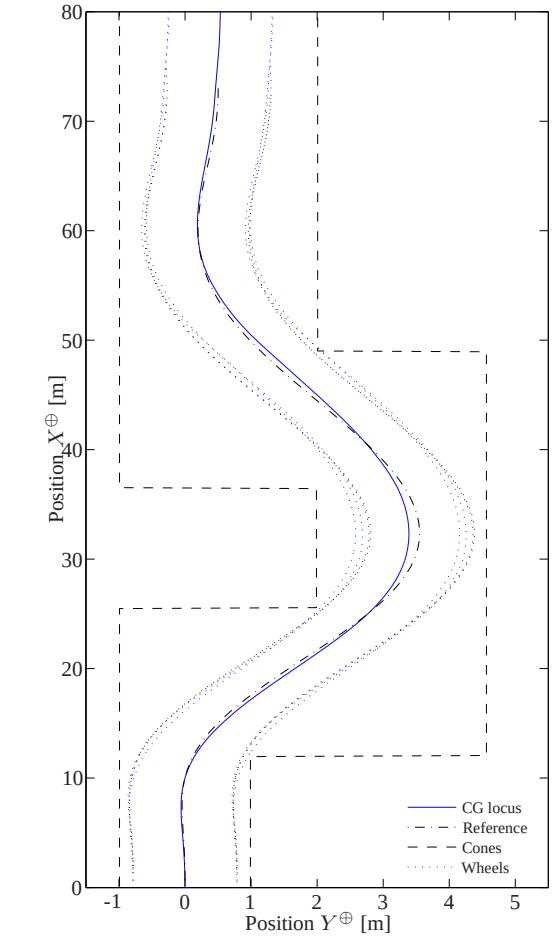
No reports have been found in the literature of experiments to ascertain a minimum friction limit for a car and driver to navigate the ISO 3888-2 emergency obstacle course. ISO 3888 double lane change manoeuvres were simulated (Mancosu & Arosio 2005) and performed (Dodd & Gothié 2005) on snow and ice as part of the VERTEC project, but using only the more benign Part 1 test-track (Dodd 2007).

The lowest friction coefficient for which a feasible trajectory can be generated for the ISO 3888-2 emergency double lane-change with the target car, using the optimal generation method, is $\mu = 0.25$. The controller is designed to operate in conditions with a friction coefficient of $\mu \approx 1$; not for wet or icy conditions, or for driving on gravel or loose surfaces. Nevertheless, it is of interest to see how the controller behaves away from its design conditions.

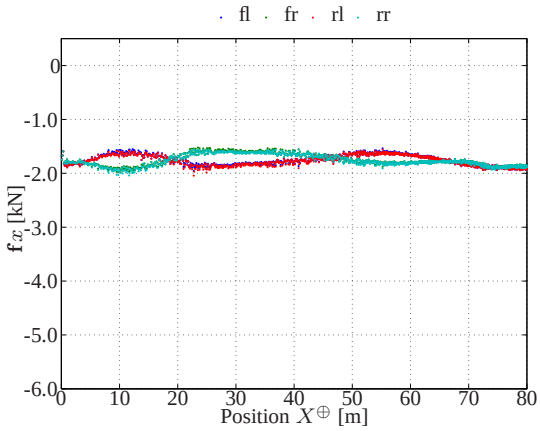
To test performance of the controller on a surface with reduced friction, the Part 2 emergency double lane-change is simulated on wet asphalt ($\mu = 0.7$) which provides 30% less traction for the tyres than a dry road. The results are shown in Figure 5.11. The (unmodified) controller copes perfectly well. Slightly larger steering angles are employed than for a dry road, but there is otherwise little difference in behaviour.

Similar simulations are performed with the friction reduced further. It is seen that a 5% reduction in friction coefficient prevents the controller from accomplishing its objective (Figure 5.12). The vehicle remains stable but exceeds the test-track limits at $X^{\oplus} = 30$ [m].

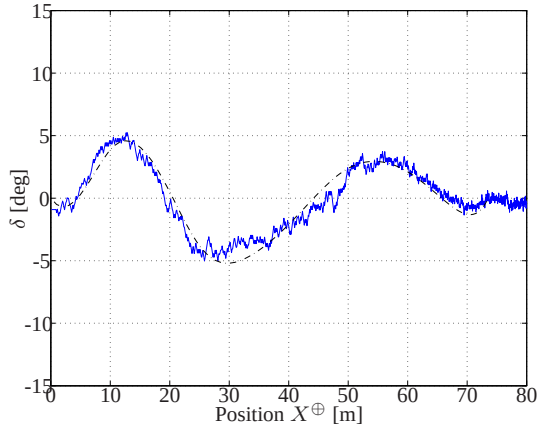
If the controller is to work on more slippery services, it is evidently necessary to generate a trajectory more suited to lower friction conditions. This can be accomplished readily by changing the friction coefficient in the trajectory generator. The optimum balance between steering and braking forces may be rather different for a low-friction trajectory, requiring a different set of controller parameters for the new surface/trajectory combination.



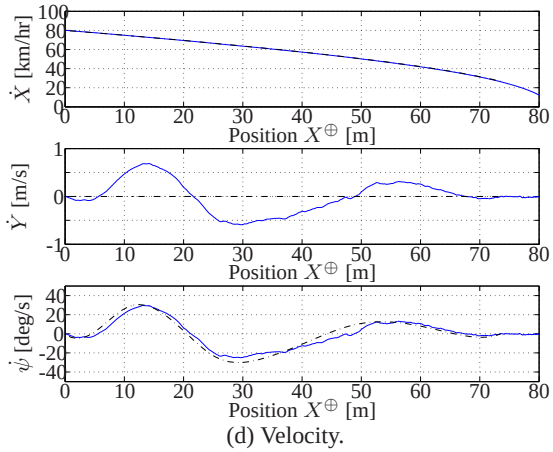
(a) Trajectory.



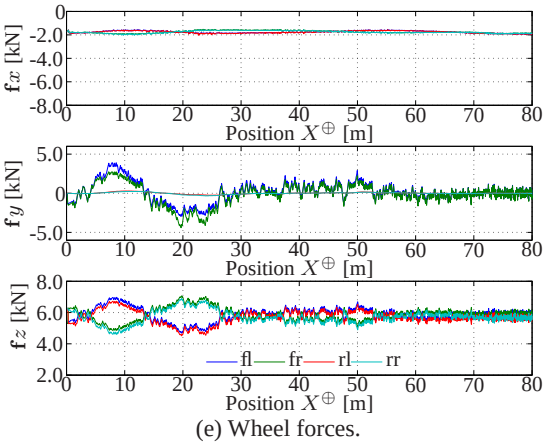
(b) Brake forces.



(c) Steering angle.



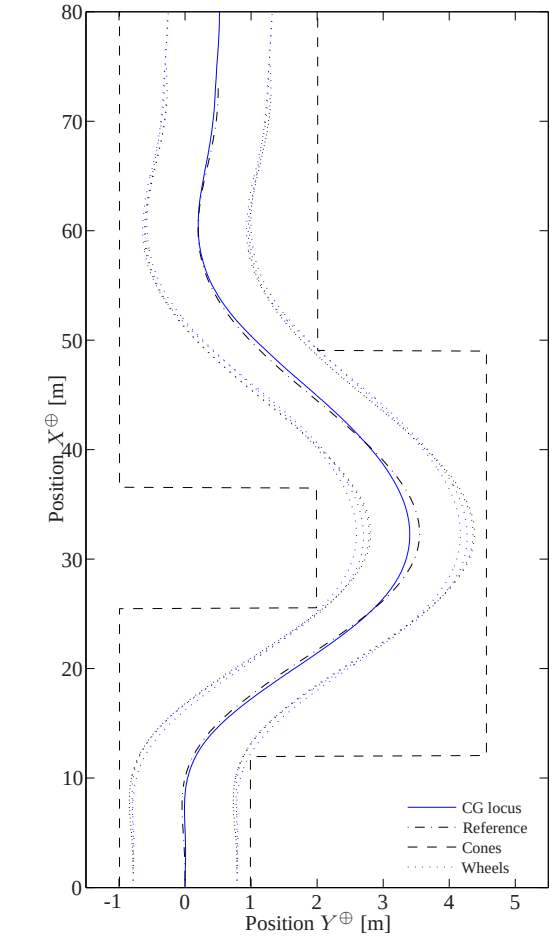
(d) Velocity.



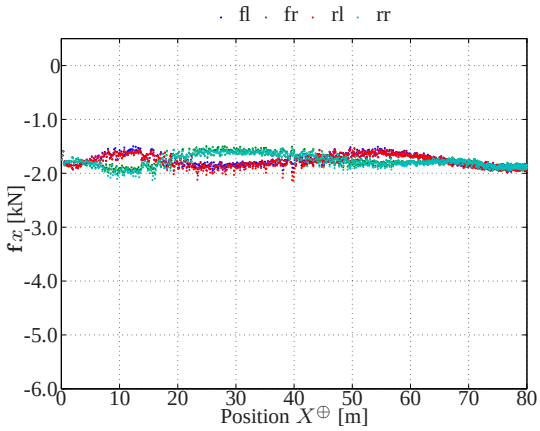
(e) Wheel forces.

Model:	*MexCar*	**Manoeuvre**:	*ISO 3888-2*	**Speed**:	*80 [km/hr]*
Trajectory:	*optimal*	**Surface**:	*dry asphalt:* $\mu = 1$	**Signal**:	*noisy:* $\nu = 2.5\%$

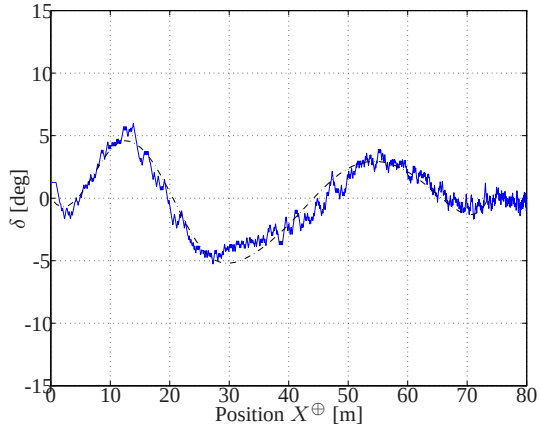
Figure 5.7: Simulation of an emergency double lane-change with a noise and disturbance level of $\nu = 2.5\%$.



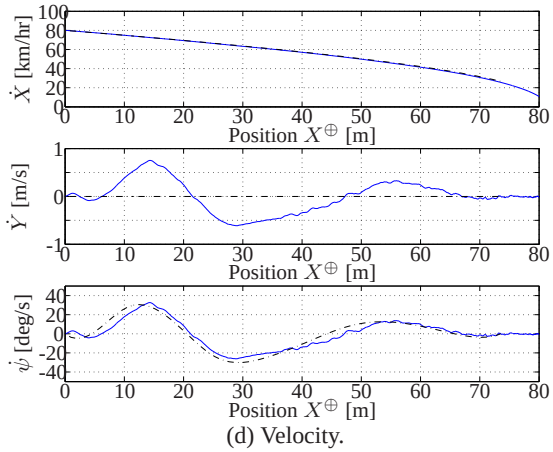
(a) Trajectory.



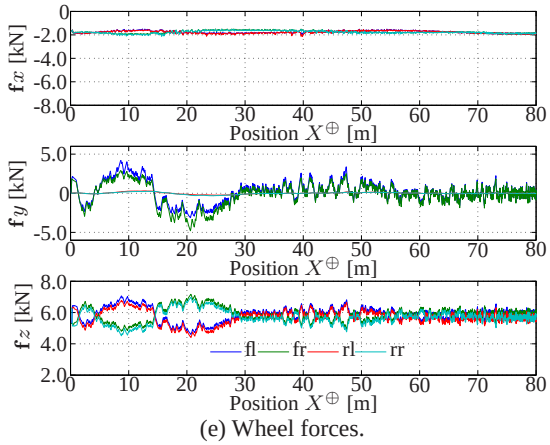
(b) Brake forces.



(c) Steering angle.



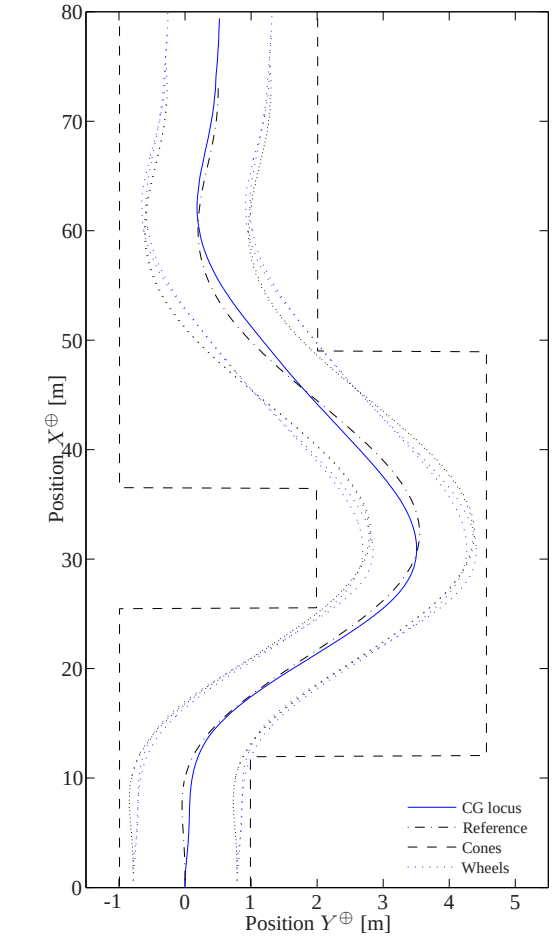
(d) Velocity.



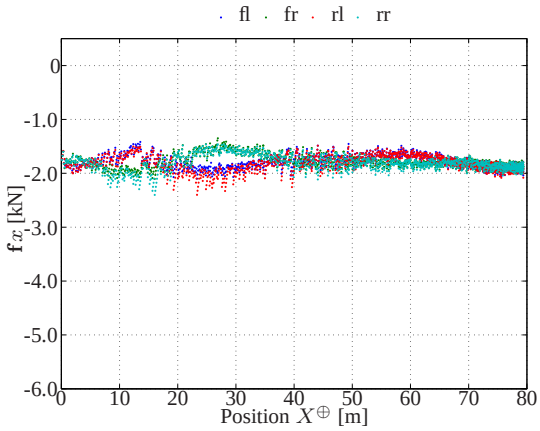
(e) Wheel forces.

Model:	*MexCar*	**Manoeuvre**:	*ISO 3888-2*	**Speed**:	*80 [km/hr]*
Trajectory:	*optimal*	**Surface**:	*dry asphalt:* $\mu = 1$	**Signal**:	*noisy:* $\nu = 5\%$

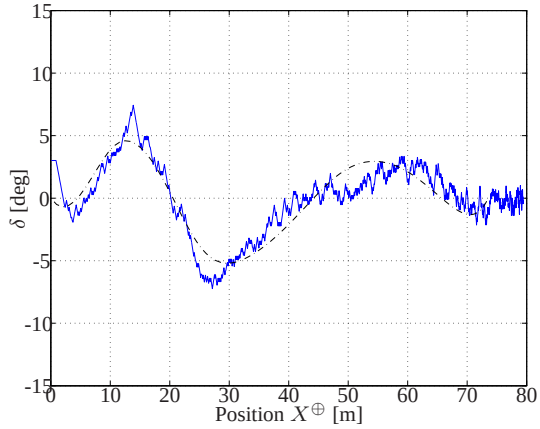
Figure 5.8: Simulation of an emergency double lane-change with a noise and disturbance level of $\nu = 5\%$.



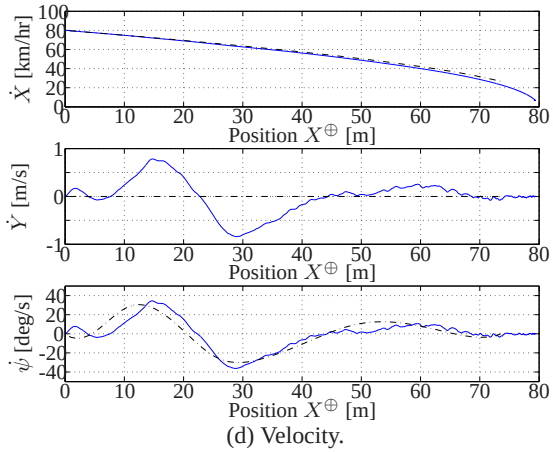
(a) Trajectory.



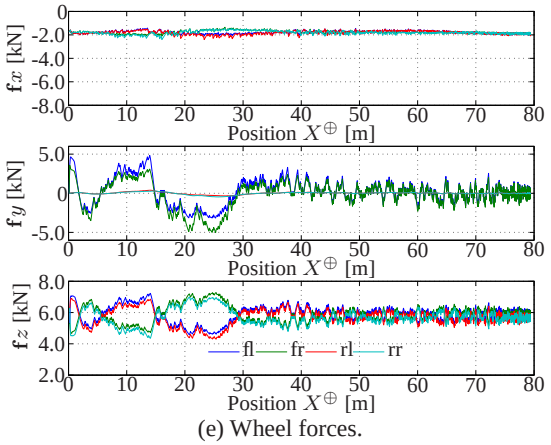
(b) Brake forces.



(c) Steering angle.



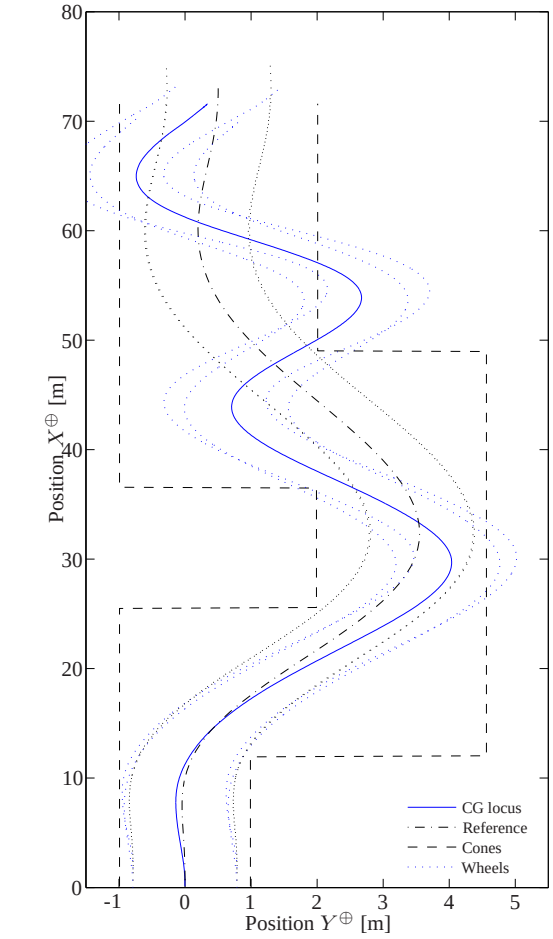
(d) Velocity.



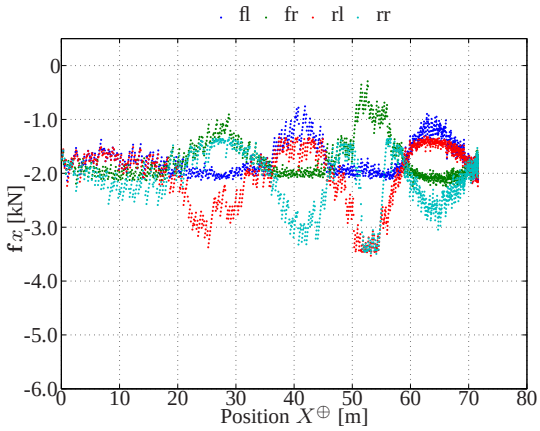
(e) Wheel forces.

Model:	*MexCar*	**Manoeuvre**:	*ISO 3888-2*	**Speed**:	*80 [km/hr]*
Trajectory:	*optimal*	**Surface**:	*dry asphalt:* $\mu = 1$	**Signal**:	*noisy:* $\nu = 10\%$

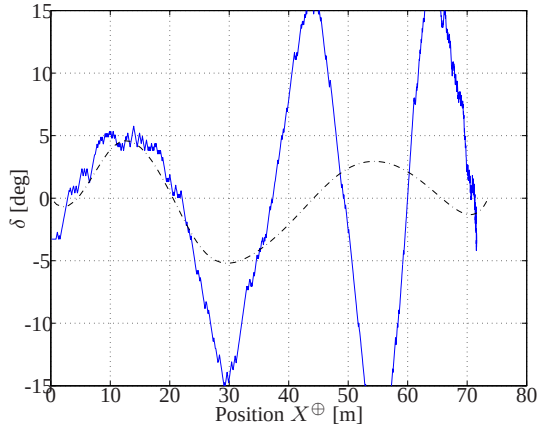
Figure 5.9: Simulation of an emergency double lane-change with a noise and disturbance level of $\nu = 10\%$.



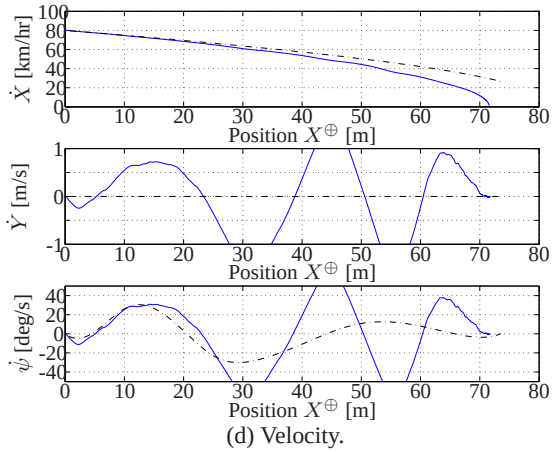
(a) Trajectory.



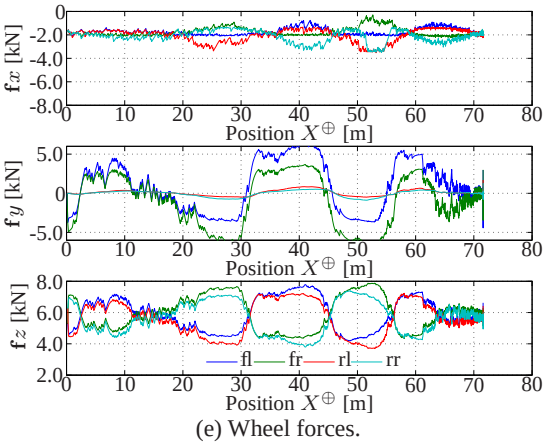
(b) Brake forces.



(c) Steering angle.



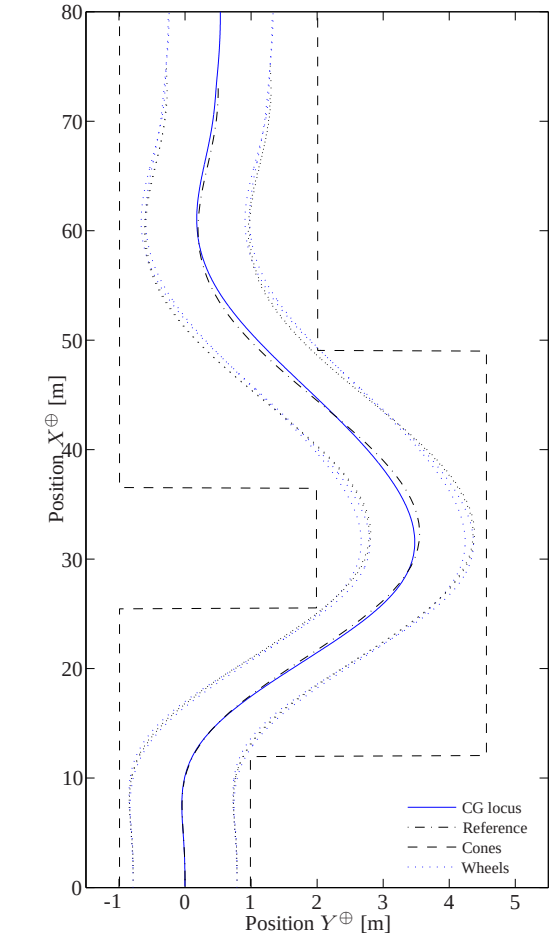
(d) Velocity.



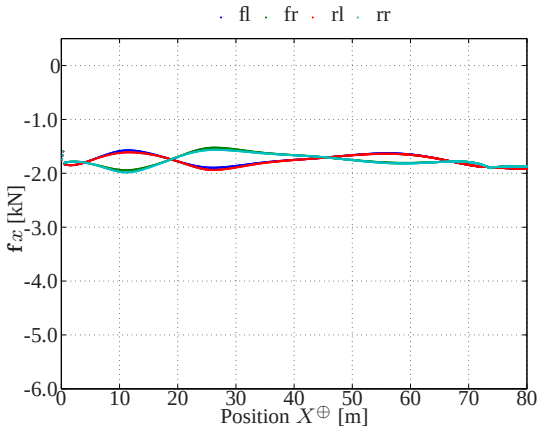
(e) Wheel forces.

Model:	*MexCar*	**Manoeuvre**:	*ISO 3888-2*	**Speed**:	*80 [km/hr]*
Trajectory:	*optimal*	**Surface**:	*dry asphalt:* $\mu = 1$	**Signal**:	*noisy:* $\nu = 20\%$

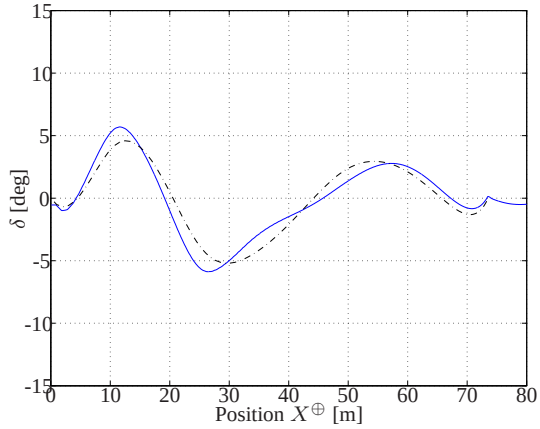
Figure 5.10: Simulation of an emergency double lane-change with a noise and disturbance level of $\nu = 20\%$.



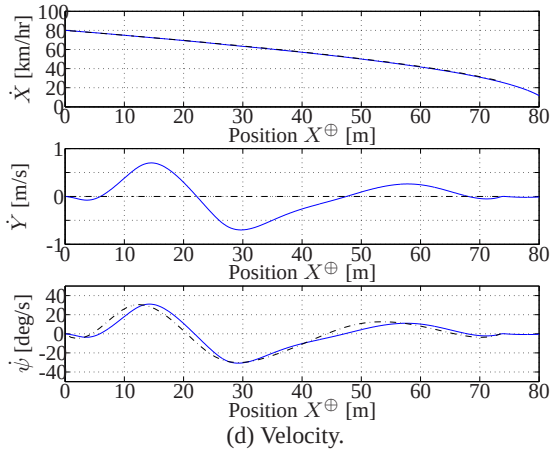
(a) Trajectory.



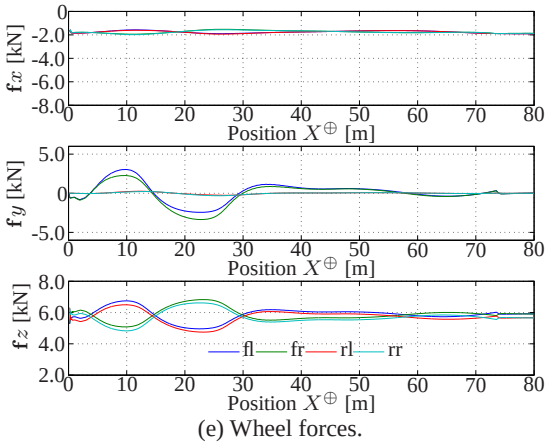
(b) Brake forces.



(c) Steering angle.



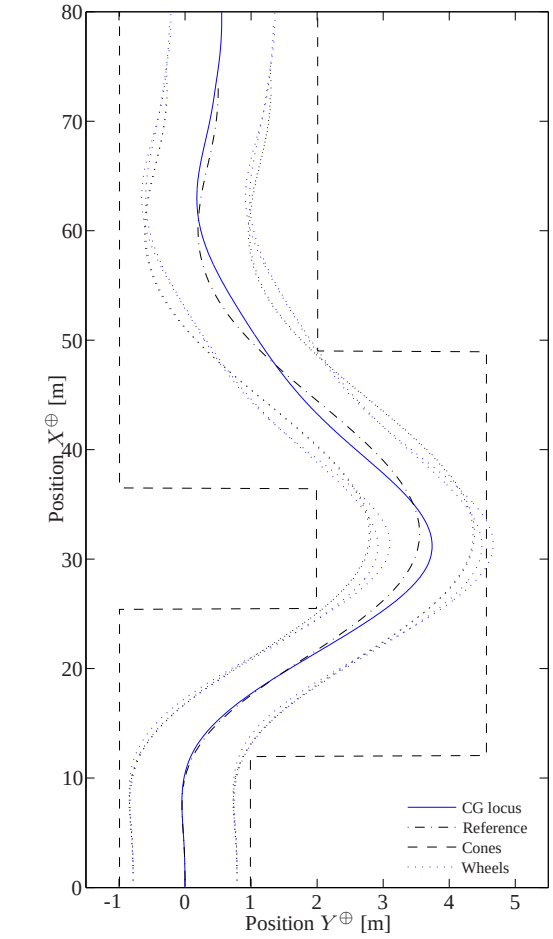
(d) Velocity.



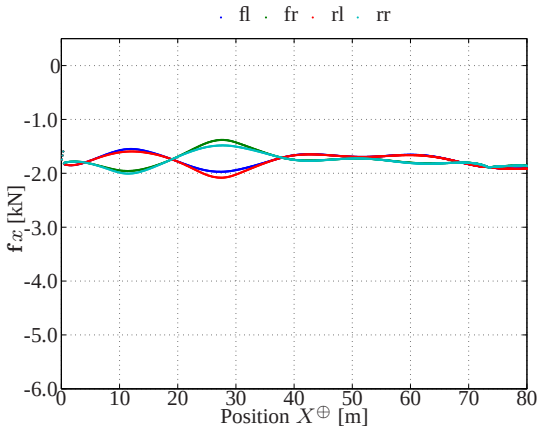
(e) Wheel forces.

Model:	*MexCar*	**Manoeuvre**:	*ISO 3888-2*	**Speed**:	*80 [km/hr]*
Trajectory:	*optimal*	**Surface**:	*wet asphalt:* $\mu = 0.7$	**Signal**:	*clean*

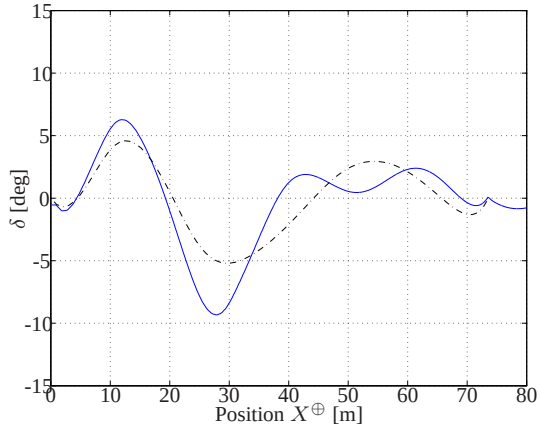
Figure 5.11: Simulation of an emergency double lane-change with a friction coefficient of $\mu = 0.7$.



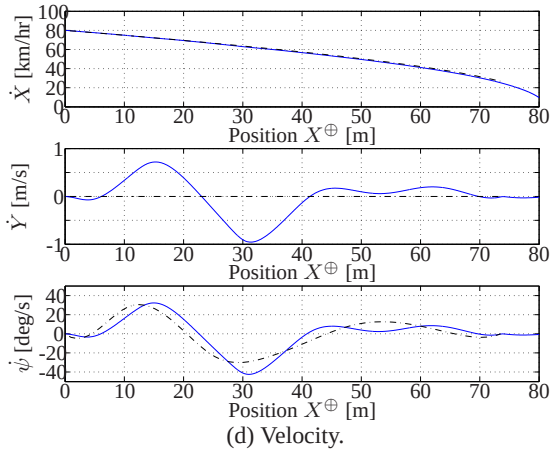
(a) Trajectory.



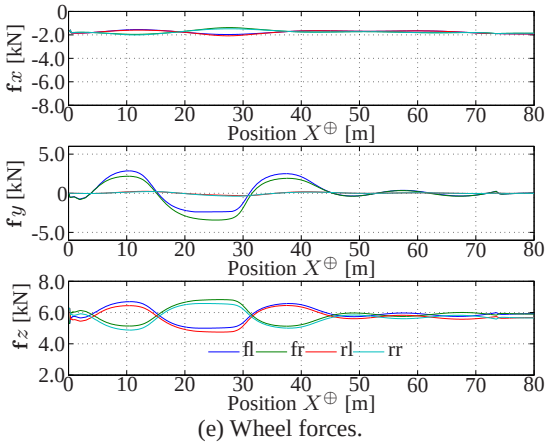
(b) Brake forces.



(c) Steering angle.



(d) Velocity.



(e) Wheel forces.

Model:	*MexCar*	**Manoeuvre**:	*ISO 3888-2*	**Speed**:	*80 [km/hr]*
Trajectory:	*optimal*	**Surface**:	*wet asphalt:* $\mu = 0.6$	**Signal**:	*clean*

Figure 5.12: Simulation of an emergency double lane-change with a friction coefficient of $\mu = 0.6$.

A trajectory is generated for a surface which produces a friction coefficient $\mu = 0.6$ with the vehicle tyres. After trajectory generation, the controller is re-tuned using batch simulations as described in the previous chapter. Figures 5.13 to 5.15 show the error responses of yaw rate, yaw angle and lateral position arising from variation of the controller gains, resulting from batch simulations as described in Section 4.6.1. Figure 5.13 suggests a yaw rate gain $K_{\delta,\dot{\psi}} = 1.0$ to minimise lateral position error – indeed, to minimise all of the error norms; Figure 5.14 suggests a yaw angle gain $K_{\delta,\psi} = 2.0$; and Figure 5.15 suggests a range for the lateral position gain $0.4 \leq K_{\delta,Y^{\oplus}} \leq 3$. A value of $K_{\delta,Y^{\oplus}} = 1.0$ is approximately the midpoint of the range on a logarithmic scale and is selected to provide a degree of robustness.

Each of the gains selected for the low friction case is significantly higher than for the standard scenario, although none differ by an order of magnitude. The yaw rate gain has increased from 0.12, by a factor of 8.3; the yaw angle gain has increased from 0.26, by a factor of 5.6; and the lateral position gain has increased from 0.26, by a factor of 3.8. These increases suggest that much tighter control is required on a low friction surface, which would accord with intuition. With all the gains increasing, the balance between the control loops does not differ greatly.

A simulation is then run with the new configuration. Figure 5.16 shows that the new combination of trajectory and controller parameters work for the lower friction surface.

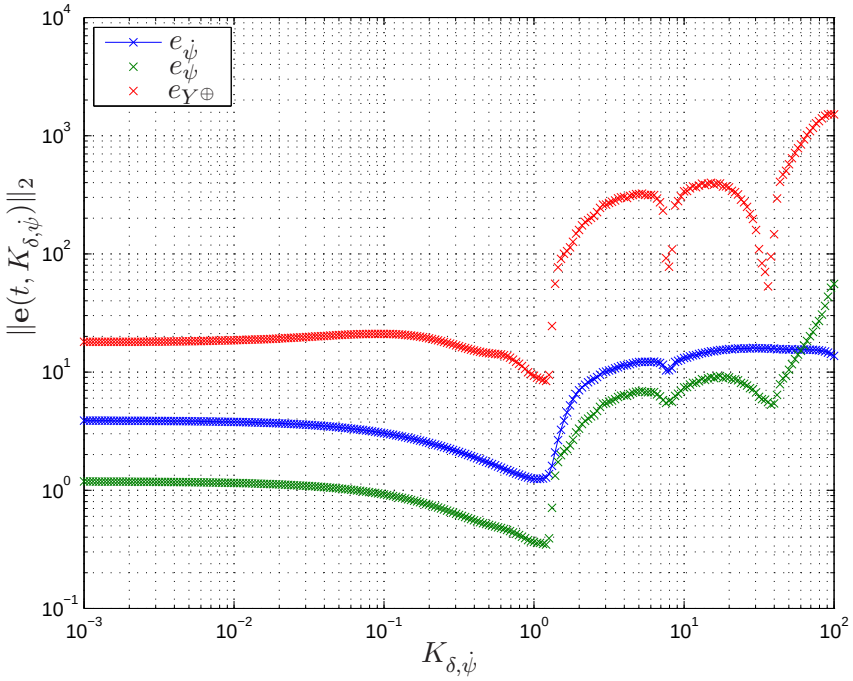


Figure 5.13: Re-tuning $K_{\delta,\dot{\psi}}$ for low friction. The effect of changing the steering loop yaw rate gain on each of the outputs: yaw rate (blue); yaw angle (green); and lateral position (red). The manoeuvre is the ISO-3888-2 emergency double lane-change performed at 80 km/hr following an optimal trajectory, with friction coefficient $\mu = 0.6$ and controller gains $K_{\delta,\psi} = 0$, $K_{\delta,Y^{\oplus}} = 0$ and $K_f = 0$.

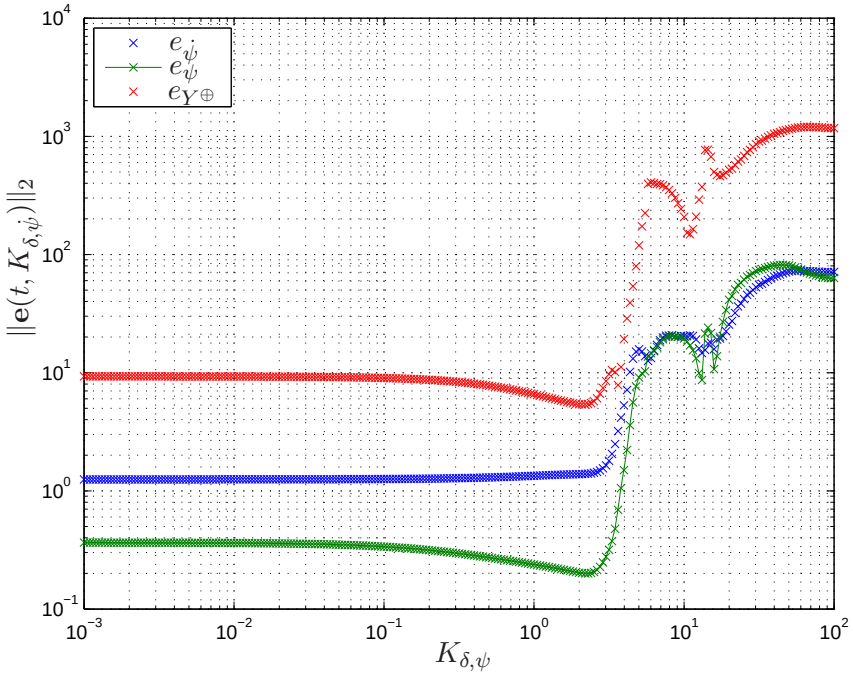


Figure 5.14: Re-tuning $K_{\delta,\psi}$ for low friction. The effect of changing the steering loop yaw angle gain on each of the outputs: yaw rate (blue); yaw angle (green); and lateral position (red). The manoeuvre is the ISO-3888-2 emergency double lane-change performed at 80 km/hr following an optimal trajectory, with friction coefficient $\mu = 0.6$ and controller gains $K_{\delta,\dot{\psi}} = 1$, $K_{\delta,Y\oplus} = 0$ and $K_f = 0$.

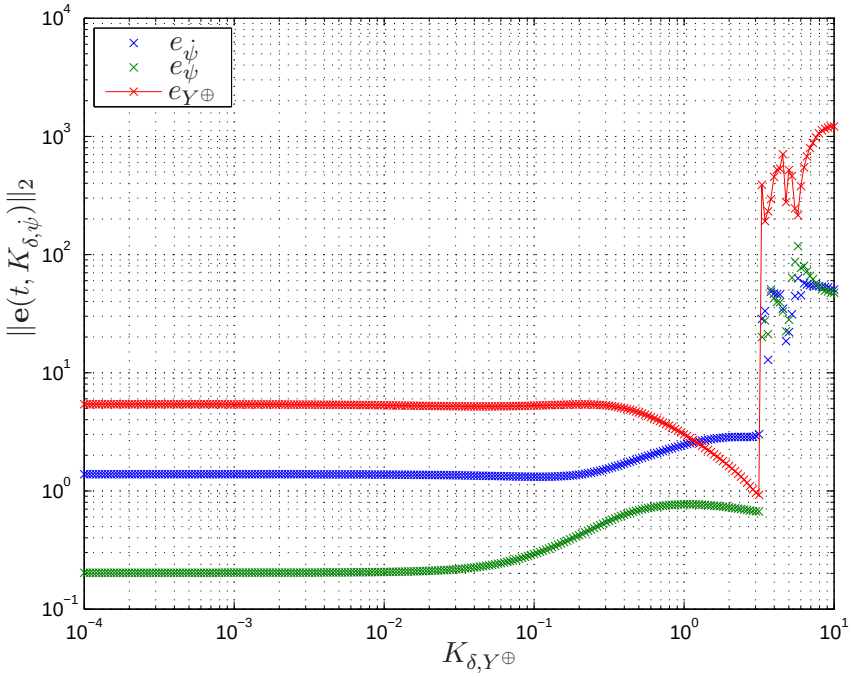
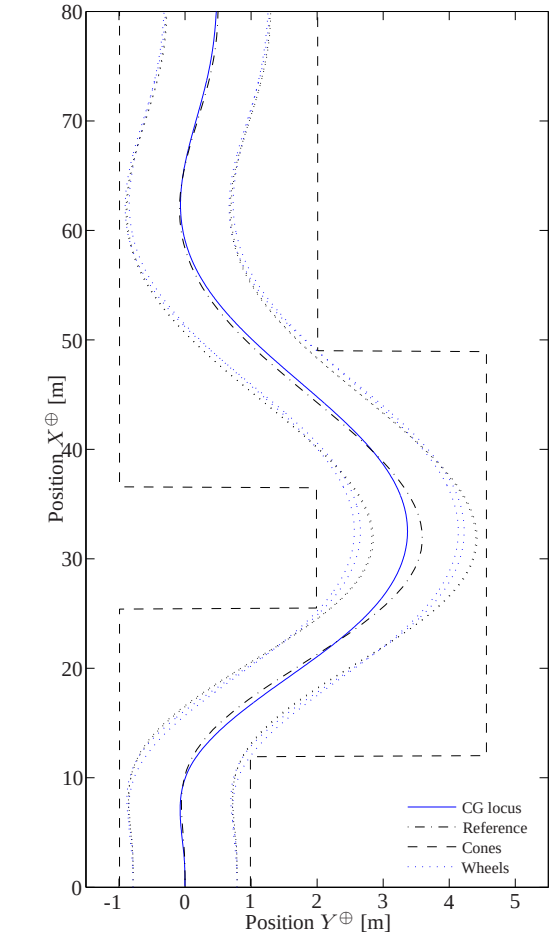
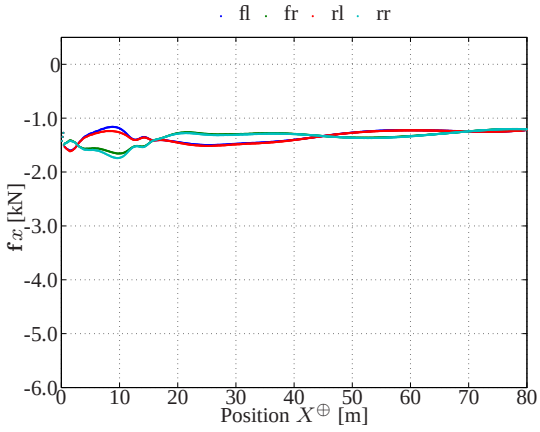


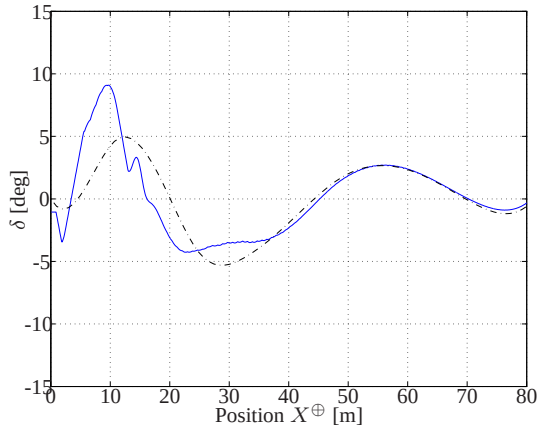
Figure 5.15: Re-tuning $K_{\delta,Y\oplus}$ for low friction. The effect of changing the steering loop yaw angle gain on each of the outputs: yaw rate (blue); yaw angle (green); and lateral position (red). The manoeuvre is the ISO-3888-2 emergency double lane-change performed at 80 km/hr following an optimal trajectory, with friction coefficient $\mu = 0.6$ and controller gains $K_{\delta,\dot{\psi}} = 1$, $K_{\delta,\psi} = 2$ and $K_f = 0$.



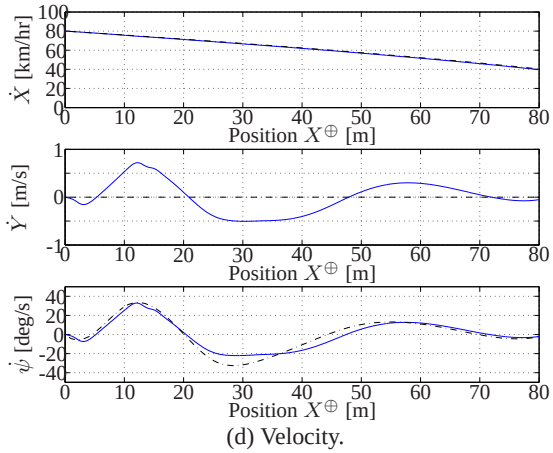
(a) Trajectory.



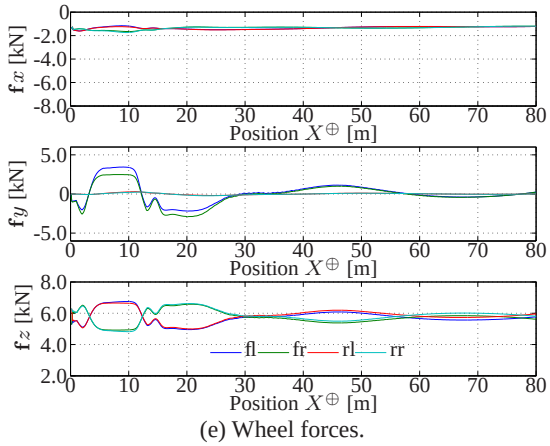
(b) Brake forces.



(c) Steering angle.



(d) Velocity.



(e) Wheel forces.

Model:	*MexCar*	**Manoeuvre**:	*ISO 3888-2*	**Speed**:	*80 [km/hr]*
Trajectory:	*optimal*	**Surface**:	*wet asphalt:* $\mu = 0.6$	**Signal**:	*clean*

Figure 5.16: Simulation of an emergency double lane-change with a friction coefficient of $\mu = 0.6$ and re-tuned controller parameters: $K_{\delta,\dot{\psi}} = 1$, $K_{\delta,\psi} = 2$ and $K_{\delta,Y^{\oplus}} = 1$.

5.4 Comparison of vehicle models

Having verified with the MexCar model that the controller accomplishes its design task and performs as required, the CASCaDE model is used for independent validation.

CASCaDE is essentially a "black box" model. It has been refined and validated against reality over many years. The model is of a high order, with many degrees of freedom. However, it includes significant features, including embedded closed-loop controllers, which are of little interest for the task at hand but lead to complex behaviour that inhibits analysis of the dynamic response to inputs (see Section 2.2).

Due to its complexity, the CASCaDE model is more fragile than MexCar. For example, it is not possible to perform the verification manoeuvre described in Section 2.9. An attempt to do so results in the simulation aborting prematurely because of errors associated with an inverse model of vertical dynamics used by its active body controller, even if this module is not active during the manoeuvre.

Nevertheless, it is useful to compare the output of both models before using CASCaDE to validate the controller. A similar manoeuvre is defined to the verification exercise, but the steering inputs are reduced by a factor of five, which is sufficient to prevent CASCaDE from aborting during the simulation. Using both models, MexCar and CASCaDE, the following manoeuvre is therefore performed:

accelerate from rest: apply an accelerating tyre force of magnitude $f_{x,i} = mg$ [N] on each wheel for 5 seconds with all wheels pointing straight ahead ($\delta = 0$ [rad]);

describe an arc: remove the longitudinal tyre forces and steer the front wheels $\pi/80$ [rad] (1.4 [deg]) to the right for 5 seconds;

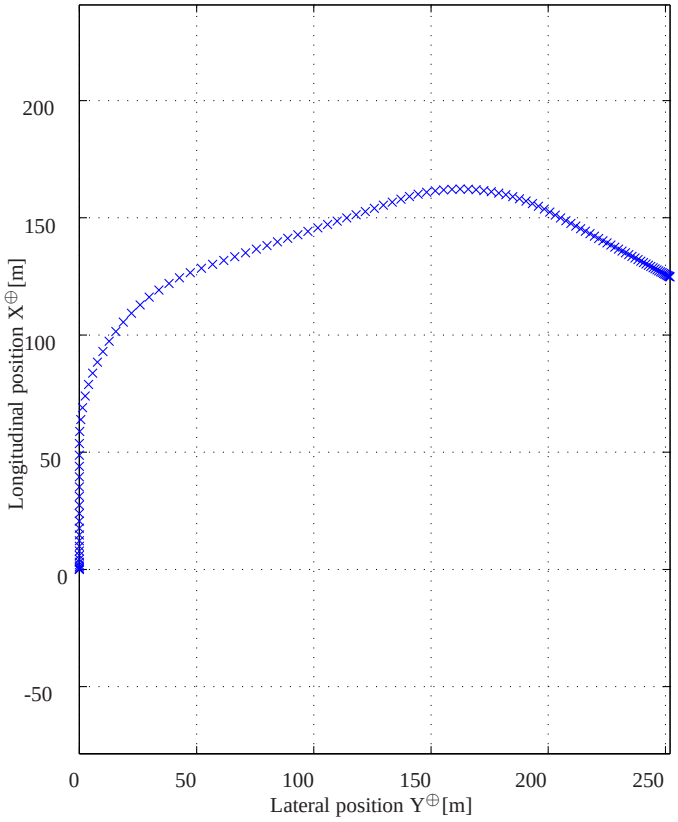
brake in a straight line: remove the steering input ($\delta = 0$) and apply braking tyre forces of magnitude $f_{x,i} = -mg/4$ [N] on each wheel for 5 seconds;

brake in a turn: without changing the braking forces, again steer the front wheels $\pi/80$ [rad] to the right for 5 seconds; and

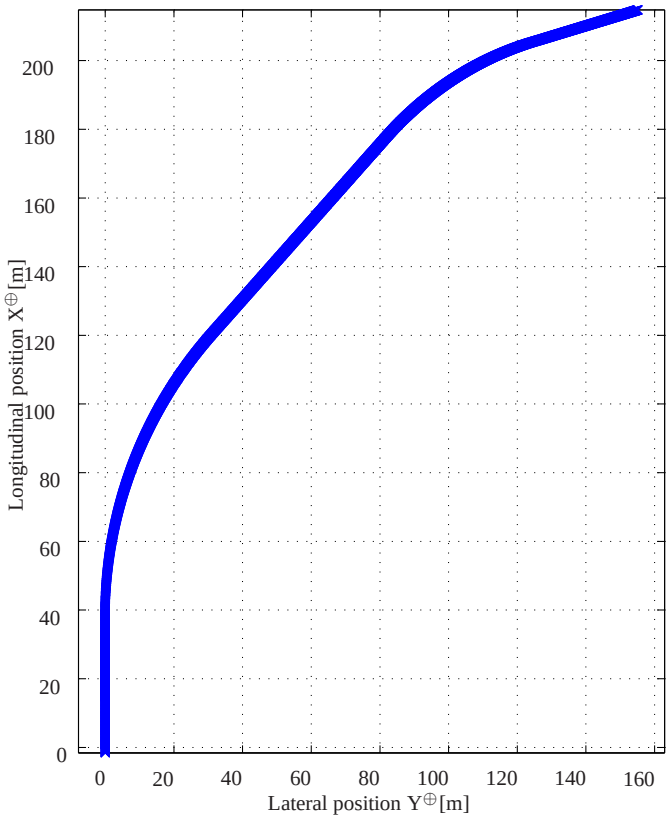
brake in a straight line: re-centre the wheels ($\delta = 0$) while continuing to apply the braking forces for a further 10 seconds.

Figure 5.17 shows the trajectory followed by both models, MexCar and CASCaDE, while Figure 5.18 shows the calculated longitudinal, lateral and yaw velocities. Visual inspection shows that the basic behaviour of the vehicle is similar in both cases, but there are clear differences between the two sets of results. In particular, CASCaDE is seen to exhibit significant oscillation in its yaw response whereas MexCar does not.

It is also noticeable that the maximum velocity attained during the CASCaDE simulation (approximately 18 [m/s]) falls short of the 20 [m/s] attained by MexCar and predicted by consideration of Newton's second law applied to a rigid body. The reason for this is that during the CASCaDE simulation, the initial application of longitudinal tyre forces does not cause the vehicle to start accelerating immediately. Instead, for the first couple of seconds, the vehicle remains almost stationary while internal body dynamics settle down.

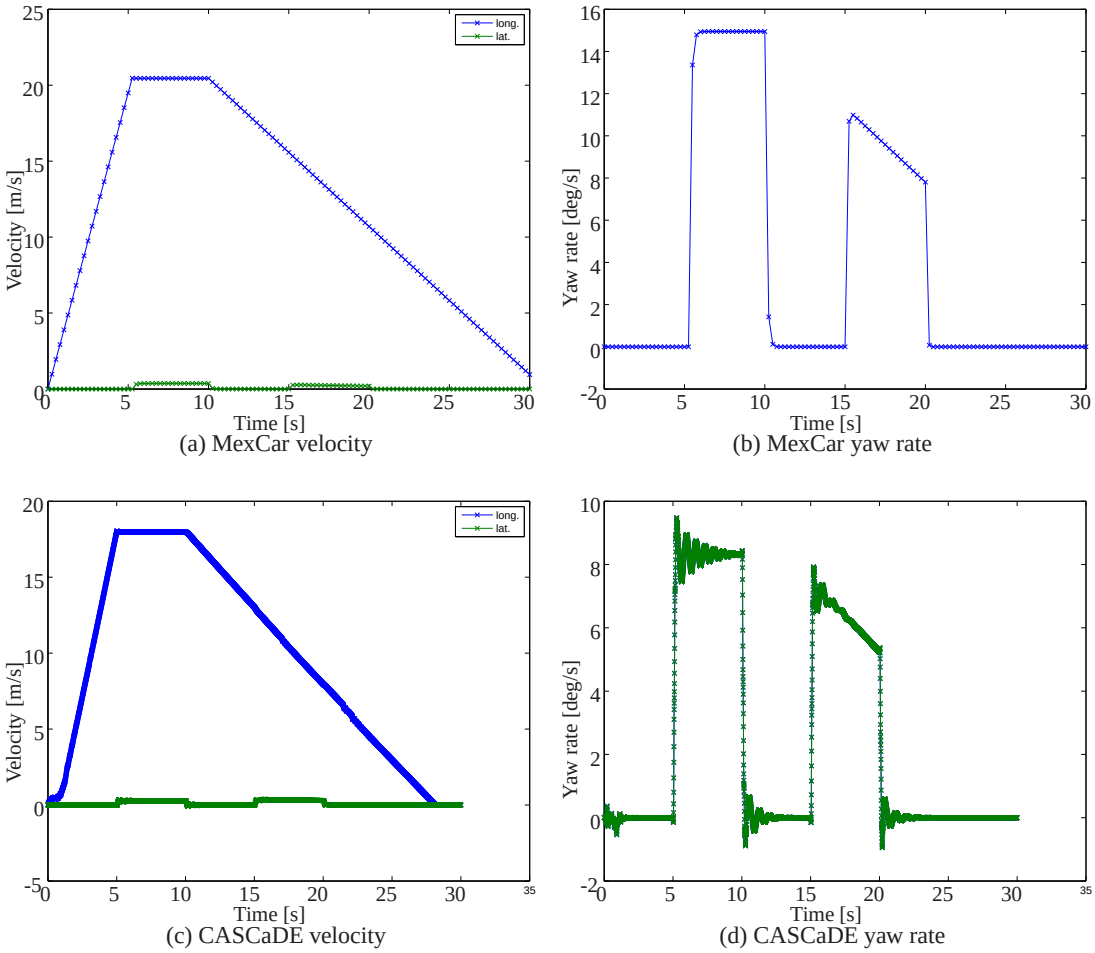


(a) MexCar trajectory



(b) CASCaDE trajectory

Figure 5.17: MexCar and CASCaDE: Trajectory comparison



(a) MexCar velocity (b) MexCar yaw rate

(c) CASCaDE velocity (d) CASCaDE yaw rate

Figure 5.18: MexCar and CASCaDE: longitudinal, lateral and yaw velocity

Just as the peak longitudinal velocity is lower in the CASCaDE simulation, so is the peak yaw rate, which is significantly lower (less than 10 [deg/s] for CASCaDE, compared to 15 [deg/s] for MexCar) because of oscillation in the yaw response. Consequently, the CASCaDE simulation predicts that the vehicle would not travel as far, or turn as sharply, as the MexCar simulation would suggest.

The oscillation predicted by CASCaDE arises as a result of the vehicle's vertical and internal dynamics. Unlike MexCar, which uses a quasi-static load distribution model, CASCaDE includes a full model of the suspension system and body-chassis interaction. The results from CASCaDE indicate that vertical dynamics have a greater effect on longitudinal, lateral and yaw dynamics than would be inferred from MexCar.

Without experimental data, which are not available, it is not possible to say for certain which model more accurately predicts the vehicle response. However, engineers at DaimlerChrysler claim that CASCaDE does predict vehicle behaviour very well. It is also a prerequisite of installing any controller on their research vehicles that the controller is first tested with CASCaDE. For the purposes of this research, it is therefore assumed that CASCaDE does provide an accurate representation of the relevant vehicle dynamics.

5.5 Validation of controller performance using CASCaDE

Simulations of the ISO 3888-2 emergency double lane-change are performed using the CASCaDE model and the controller developed previously. CASCaDE simulations do not generally start from a steady state. The vehicle lateral and yaw velocities, front wheel steering angle and wheel brake forces are initialised to zero and the longitudinal velocity is set slightly higher than the required forward speed. With these conditions, it takes a short while for the car to reach equilibrium. To allow initial dynamics to settle down, an initial run-up of 30 [m] is included before the start of the obstacle course. During the run-up, the controller acts only to keep the vehicle travelling straight ahead with as little speed loss as possible. If the speed at entry to the obstacle course is not correct, the simulation is repeated with slightly different starting speeds until this condition is met.

The controller parameters, derived in Chapter 4, are $K_{\delta,\dot{\psi}} = 0.12$, $K_{\delta,\psi} = 0.36$, $K_{\delta,Y^{\oplus}} = 0.26$ and $K_{f,\dot{\psi}} = 0$ or 15. Figure 5.19 shows simulation results when yaw rate stabilisation is not used ($K_{f,\dot{\psi}} = 0$) and Figure 5.20 shows the corresponding results when it is ($K_{f,\dot{\psi}} = 15$).

Without yaw rate stabilisation, the vehicle fails to attain the necessary yaw rate at the first turn, cuts across the first corner, fails to correct in time for the second turn and exits the manoeuvre space through the boundary. With yaw rate stabilisation, the result is similar, but the vehicle does follow the reference path slightly more closely, as evidenced by the narrower wheel tracks; the front and rear wheels follow more similar paths.

In both cases, the manoeuvre begins to go wrong shortly after 40 [m], a distance of 10 [m] into the start of the obstacle course. The reason can be seen in Figures 5.19(e) and 5.20(e). The vertical load on the right rear tyre drops to zero; the tyre leaves the road and is unable to contribute to controlling the vehicle. Similar behaviour is not exhibited by the MexCar model (Figure 4.15 and 4.17).

There are two noteworthy aspects to the discrepancy between MexCar and CASCaDE in this respect. Firstly, it is seen that only the rear wheel lifts from the ground. A considerable load remains on the front right wheel. This longitudinal weight transfer is not modelled within MexCar which does not consider the effect of pitching motion. The other significant difference between the models that contributes to this difference of behaviour is the roll model. MexCar employs a quasi-static roll model (Equation (2.17)) whereas CASCaDE allows the body to move relative to the chassis and integrates the dynamic roll equations that result from forces within the suspension system. Consequently, MexCar is seen to underestimate the roll experienced by the car during an aggressive manoeuvre.

There are several possible approaches to resolving the discrepancy. It would be possible to add these dynamics to MexCar, at the expense of making the model more complex. Alternatively, it would be to impose additional constraints on the trajectory generator, which can be done simply. However, in either case, the outcome would be predictable. The fundamental problem is that the car is required to perform a manoeuvre that is too aggressive. The severity of the turn must be reduced. If the boundary remains fixed, this must be achieved by reducing speed earlier in the manoeuvre.

Repeating the design exercise, including optimisation, would be expected to increase early braking, by the minimum amount necessary. However, it is not clear that this is actually the most

desirable behaviour for a collision avoidance system. Although this research does not focus on human factors, it is useful to briefly consider how such a system would be implemented on a production vehicle.

An aggressive lateral collision avoidance manoeuvre is never something to be undertaken lightly. The driver must be given every opportunity to decide whether such action should be taken, or whether it would instead be preferable to crash into an obstruction. For example, in the presence of pedestrians, undetected by the vehicle's sensors, it may well be preferable to crash into a car in front, sustaining damage to vehicles and mild injury to occupants, than to cause the death of unprotected bystanders.

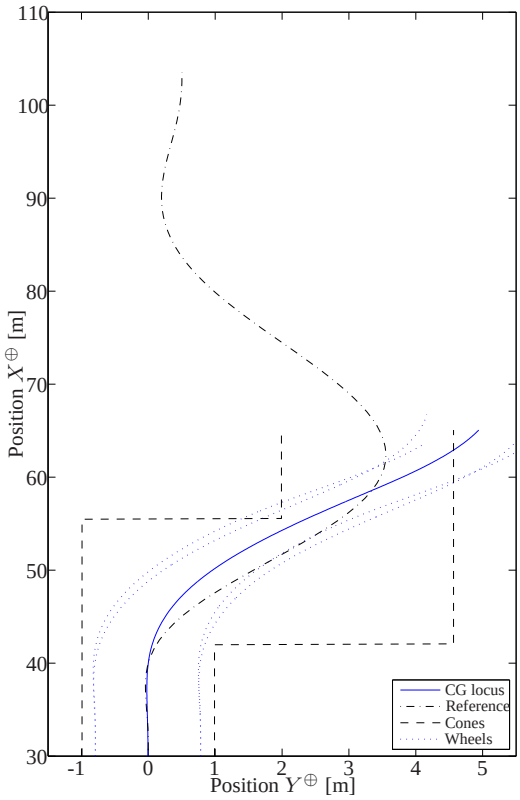
In a realistic scenario, the vehicle computer would continually anticipate possible collisions and plan how to avoid them should the need arise, much like a human driver. As the risk of collision increases, it is likely that the vehicle would perform the following sequence of actions:

1. warn the driver about the presence of danger;
2. increase the alert level and warn the driver that automatic action is imminent;
3. increase the alert level further while initiating braking (without steering) to give the driver time to act and mitigate the effects of any impact, while indicating to other vehicles that evasive action is imminent;
4. execute a last minute lateral collision avoidance manoeuvre - the focus of this research.

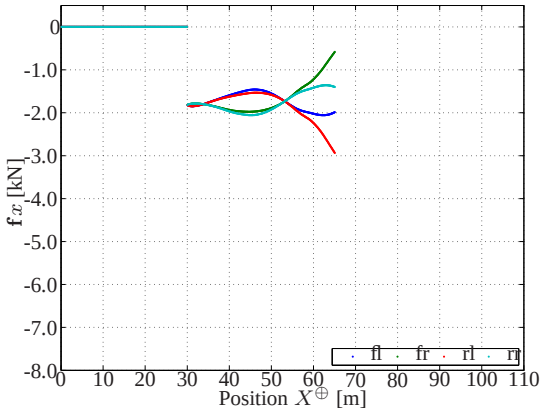
On the roads, braking is generally considered to be a safe manoeuvre. Although there may be instances where braking would be foolish, such as halfway across a level crossing, it is usually the responsibility of other drivers to drive in a manner that ensures they would be able to cope with an emergency stop by another vehicle. Consequently, it is likely that a lateral collision avoidance system, which entails severe risk, would be used only after longitudinal avoidance or mitigation.

From this perspective, the initial sections of the ISO 3888 manoeuvres, during which the vehicle is not permitted to turn, can be seen as the period during which the vehicle performs only longitudinal collision avoidance and warns other vehicles of impending action, without leaving the vehicle's lane. In this context, it is desirable that the vehicle should brake hard during this phase of the manoeuvre. To continue driving at full speed through this region without performing any control action wastes a large proportion of the traction available within the course. Application of the brakes in this region will not have any adverse impact on the ability of the tyre to generate lateral forces for turning later. Indeed, using this phase of the manoeuvre to remove kinetic energy from the system would be very sensible behaviour.

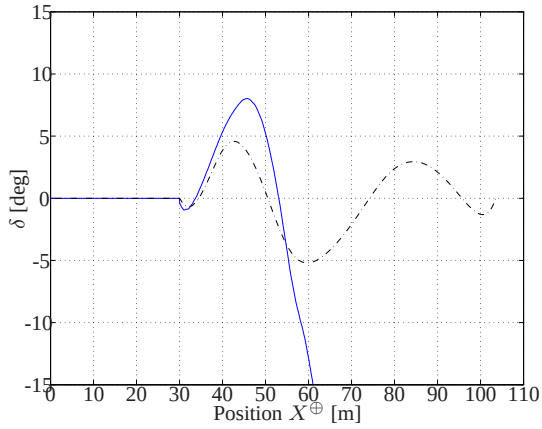
The controller is modified to include a pre-braking element. The control loops and parameters remain as before, as does the steering profile, but the reference longitudinal acceleration profile is altered. Upon entering the test-track, the reference acceleration $\ddot{X}$ is set to $-\mu g$ [m/s^2] for the first 12 [m]. The results of this modification to the control law are shown in Figures 5.21 and 5.22 for a controller with and without yaw rate stabilisation respectively.



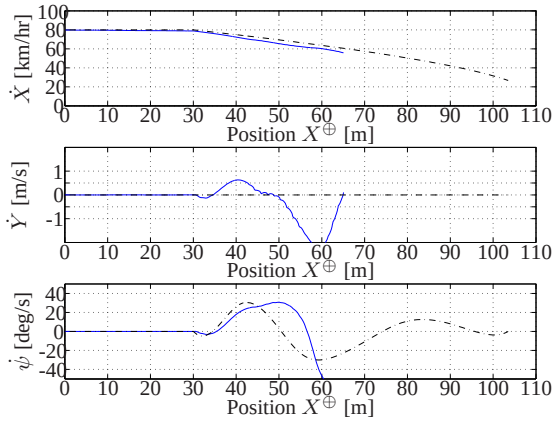
(a) Trajectory.



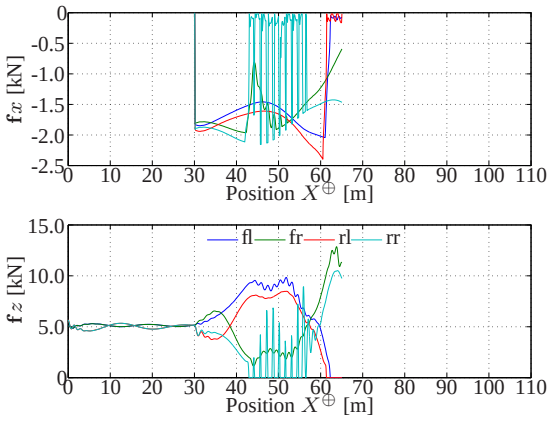
(b) Brake forces.



(c) Steering angle.



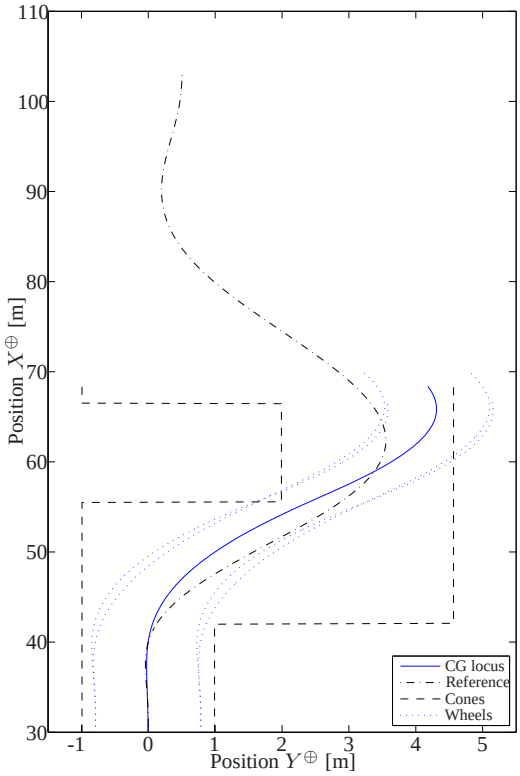
(d) Velocity.



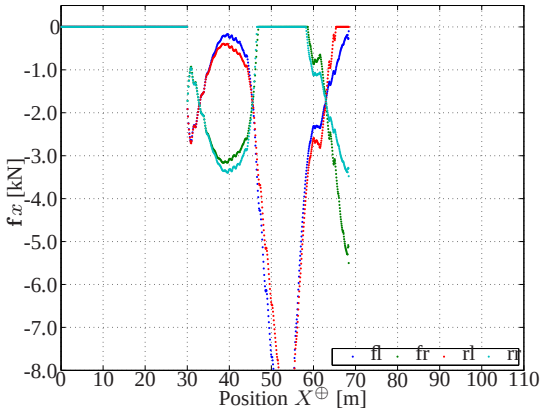
(e) Wheel forces.

Model:	*CASCaDE*	**Manoeuvre**:	*ISO 3888-2*	**Speed**:	*80 [km/hr]*
Trajectory:	*optimal*	**Surface**:	*dry asphalt:* $\mu = 1$	**Signal**:	*clean*

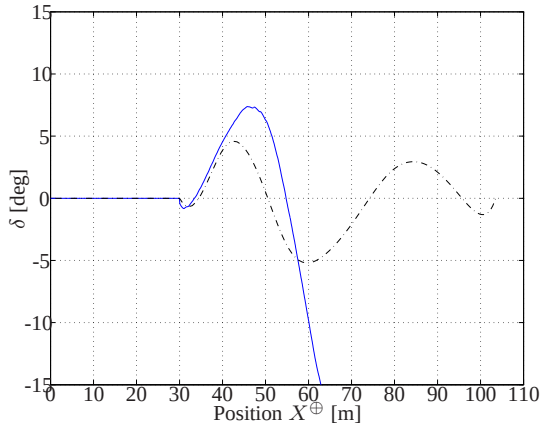
Figure 5.19: Simulation of an emergency double lane-change using the CASCaDE model. Without pre-braking. Without yaw stabilisation.



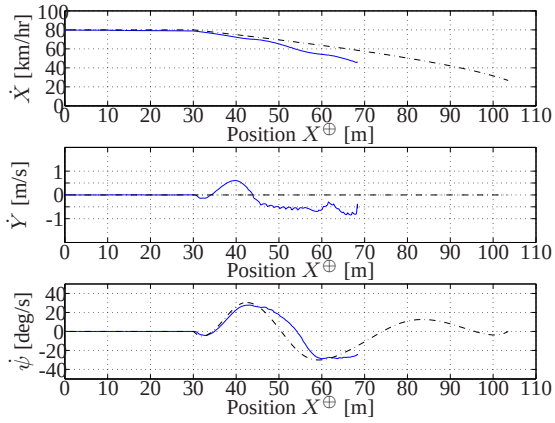
(a) Trajectory.



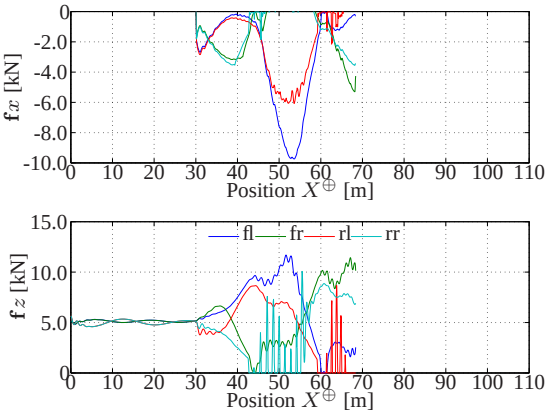
(b) Brake forces.



(c) Steering angle.



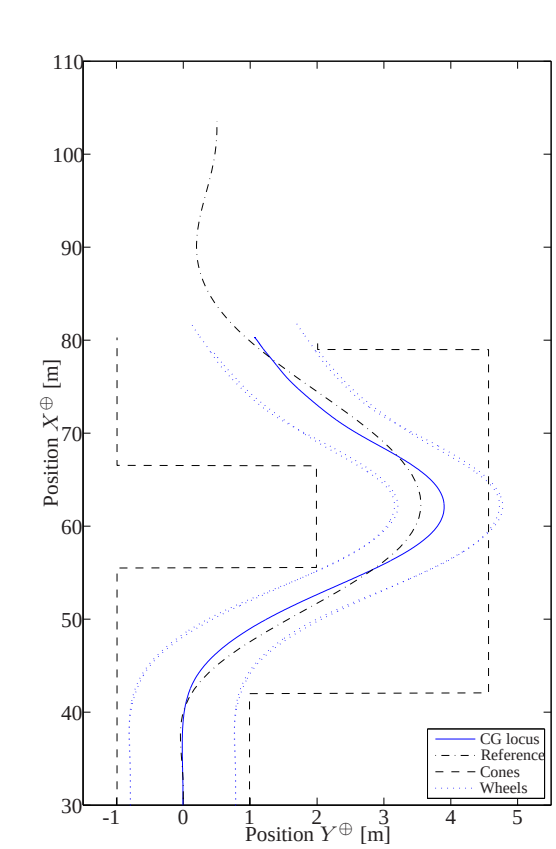
(d) Velocity.



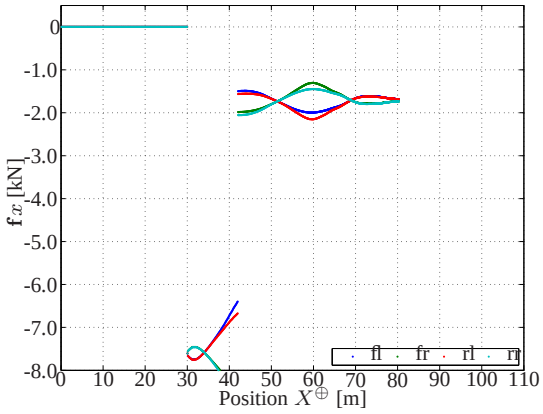
(e) Wheel forces.

Model:	*CASCaDE*	**Manoeuvre**:	*ISO 3888-2*	**Speed**:	*80 [km/hr]*
Trajectory:	*optimal*	**Surface**:	*dry asphalt:* $\mu = 1$	**Signal**:	*clean*

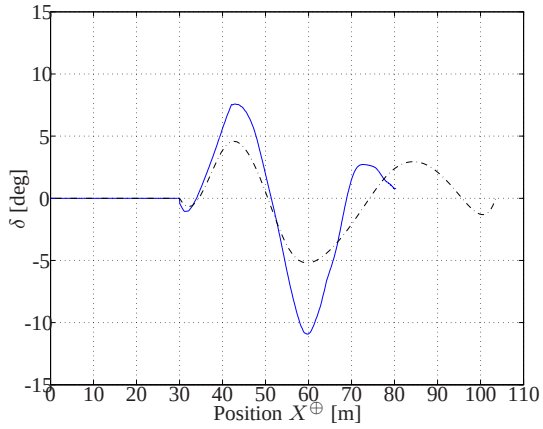
Figure 5.20: Simulation of an emergency double lane-change using the CASCaDE model. Without pre-braking. With yaw stabilisation.



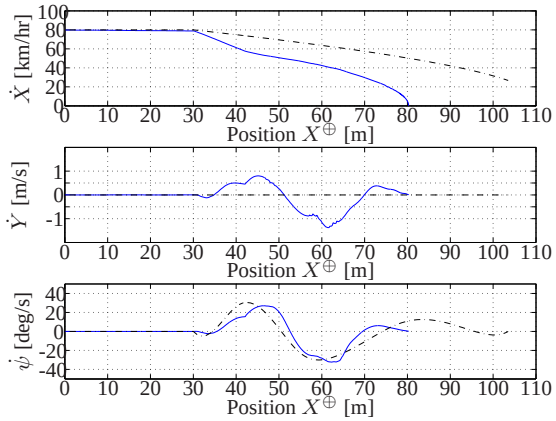
(a) Trajectory.



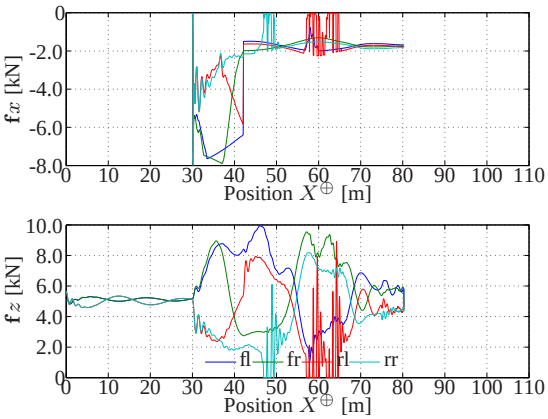
(b) Brake forces.



(c) Steering angle.



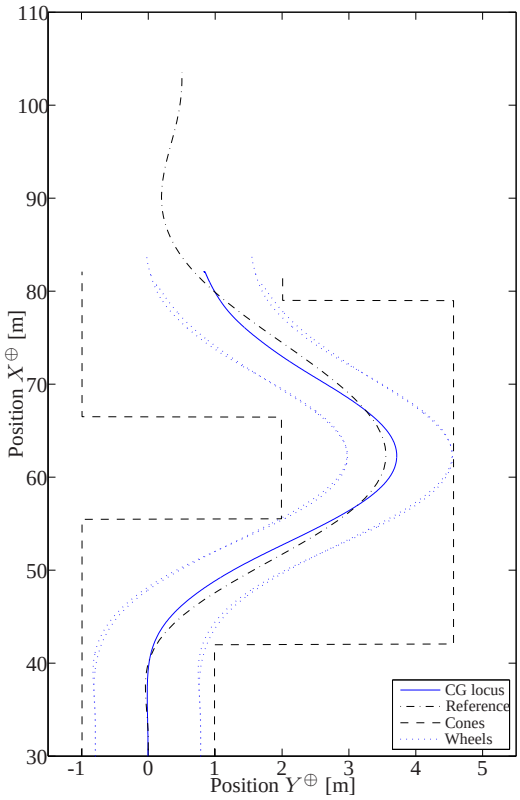
(d) Velocity.



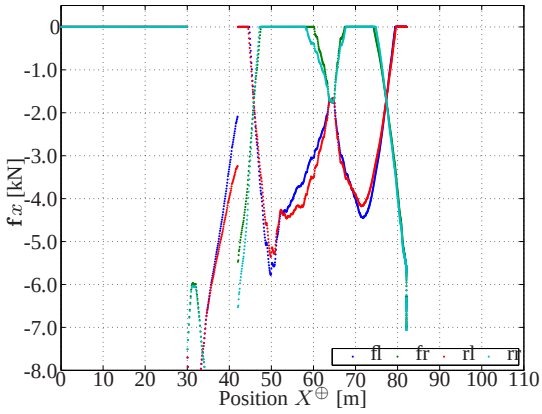
(e) Wheel forces.

Model:	*CASCaDE*	**Manoeuvre**:	*ISO 3888-2*	**Speed**:	*80 [km/hr]*
Trajectory:	*optimal*	**Surface**:	*dry asphalt:* $\mu = 1$	**Signal**:	*clean*

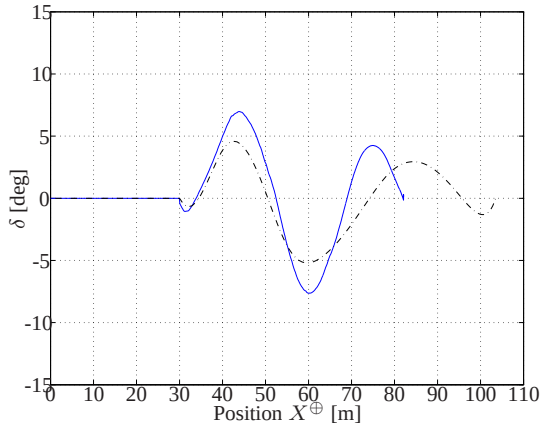
Figure 5.21: Simulation of an emergency double lane-change using the CASCaDE model. With pre-braking. Without yaw stabilisation.



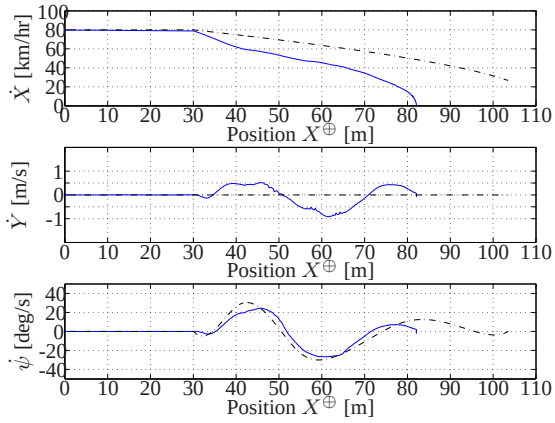
(a) Trajectory.



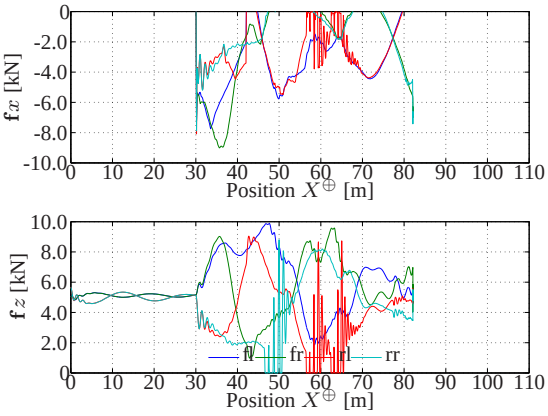
(b) Brake forces.



(c) Steering angle.



(d) Velocity.



(e) Wheel forces.

Model:	*CASCaDE*	**Manoeuvre**:	*ISO 3888-2*	**Speed**:	*80 [km/hr]*
Trajectory:	*optimal*	**Surface**:	*dry asphalt:* $\mu = 1$	**Signal**:	*clean*

Figure 5.22: Simulation of an emergency double lane-change using the CASCaDE model. With pre-braking. With yaw stabilisation.

Without yaw rate stabilisation, the vehicle behaviour is much improved. Its path follows the reference trajectory closely through the first turn without hitting the boundary. Nevertheless, the vehicle is not fully on the reference trajectory when the second turn starts. Consequently, the vehicle is unable to make the second turn and crashes through the boundary. However, with yaw rate stabilisation included in the control law, the vehicle is able to make the second turn without violating the boundary. The vehicle is thus able to complete the manoeuvre successfully, stopping within the test-track as predicted by MexCar.

5.6 Discussion

In this research, a simulation-based design process is developed. The use of simulation allows the effects of highly complex, nonlinear dynamic interactions and transformation to be considered and accommodated. Two complementary models are critical to the success of the approach: the formulation of the trajectory generator as a convex optimisation problem and the MexCar dynamic model.

Neither model exactly matches CASCaDE which, for the purposes of the project, is deemed to be an entirely accurate representation of reality. Nevertheless, they are sufficient to allow the design of a good controller that does meet the design objectives, as evidenced by Figure 5.22. Both models - MexCar and the trajectory generator - neglect vertical dynamics to some extent, not accounting for the lifting of a wheel from the road surface during an aggressive turn. There is therefore scope for improvement of the design process, either by including this behaviour in the models, or altering the vehicle to suppress this unfavourable lifting; the emergence of Active Body Control (ABC) in which actuators apply vertical forces makes it feasible to consider maintaining weight on the inside wheels.

At first sight, the method of batch-simulation appears to be a somewhat brute-force approach to optimisation. However, the principle of running multiple experiments and measuring the output is well established in control engineering; it is after all the underlying basis for classical frequency-based design methods and analysis resulting from system identification.

Unlike optimisation methods that produce a single optimal solution, the simulations produce a useful tool for later refinement of the design. The tuning maps of error responses against gain parameter variation are ideal for understanding how each control loop affects the overall system response and show the ranges through which each gain may be varied. Thus they would be invaluable for field engineers wishing to make small refinements to the controller parameters after conducting experiments with real hardware. Retaining the benefits of PID control advocated by Jiang & Gao (2001), the design method should prove useful in an industrial setting.

Moreover, the final design has only four proportional gains that require tuning. Generation of reference profiles in conjunction with feedforward control relieves much of the burden from the feedback loops. The entire structure is such that each stage of the design greatly simplifies the task for remaining stages.

The choice of objective or cost function is a critical component of any optimisation procedure. For

generating feasible trajectories, minimisation of yaw acceleration is highly appropriate. The selection of lateral position error as the objective for controller parameter selection is slightly less satisfactory. It leads to appropriate gains, but the lateral position error does not directly measure the true objective, which is to ensure that the vehicle remains within the test-track boundary at all times.

5.7 Conclusion

A controller has been developed to cause a Mercedes 'S' class, "TS", to perform an ISO 3888-2 emergency obstacle avoidance double lane change manoeuvre. The performance of the controller has been validated in simulations using a highly complex proprietary model provided by DaimlerChrysler.

A simulation-based approach to the design relies on calculation of a feasible reference trajectory, which is produced using a convex formulation of the test-track and vehicle dynamic specifications. A two track vehicle model is used to identify appropriate controller parameters to cause the car to follow the reference path.

The resulting controller integrates steering and braking action and is of a form that is amenable to further refinement by field engineers.

Chapter 6

Conclusions and future work

If we knew what it was we were doing, it would not be called research, would it?

— Albert Einstein

Chapter 5 contains a comparison of the two trajectory generation methods that are developed in Chapter 3. It also features a presentation of simulation results that are used for evaluating the performance of the controller developed in Chapter 4. In this chapter, conclusions are drawn on the basis of results and discussion in the preceding chapters.

6.1 Conclusions

Successful execution of an ISO-3888-2:2002 emergency obstacle avoidance manoeuvre has been demonstrated by simulation (Chapter 5) with a proprietary vehicle model that has previously been extensively calibrated and validated against real world experimental data.

A new vehicle model has been developed. A non-linear two-track model, it is capable of generating velocity-based linearisations at non-equilibrium conditions. This model is described in Chapter 2.

A new trajectory generation method has been developed. Using a series of convex optimisations, it produces a feasible trajectory through a demanding obstacle course. This optimal trajectory generation method is described in Chapter 3.

A controller design strategy has been developed to produce a controller that integrates steering and braking actions. The strategy results in a controller that is amenable to tuning by test engineers, making it practical for use in an industrial setting. Controller design is described in Chapter 4.

6.1.1 Vehicle model

A nonlinear two-track vehicle model has been developed and implemented in C++ with interfaces to Matlab and GNU Octave. The model, which is designed for studying lateral dynamics, includes the effects of steering and braking on the vehicle longitudinal, lateral and yaw velocities.

Linear state and input matrices can be obtained from the model at any operating condition, whether or not the point lies on a manifold of equilibria. The matrices are obtained using velocity-based linearisations that have been derived symbolically from the non-linear model and account explicitly for its states and inputs.

A simple lateral weight distribution model based on a least norm solution has been used to model the vertical load on each side of the car. The general behaviour of weight transfer is correct, but lack of pitch dynamics and movement of the body relative to the chassis means that the model does not predict the lifting of wheels from the road surface.

6.1.2 Trajectory generation

Two trajectory generation methods have been used. The first, a geometric method similar to previous methods of generating paths for robots, has been used to develop a trajectory that would take a car to its physical limits while manoeuvring through an ISO-3888-1:1999 double lane-change obstacle course at constant speed. This method was found to be unsuitable for finding a feasible path through the more demanding ISO-3888-2:2002 emergency obstacle avoidance course.

The second trajectory generation method is new, and uses convex optimisation to balance use of brakes against steering. By performing the optimisation in multiple stages, a highly non-linear problem has been formulated as a series of convex approximations that have been solved using powerful off-the-shelf optimisation software. The method has been shown to be capable of finding a feasible path through the more demanding ISO-3888-2:2002 course at speeds that far exceed the requirements of the standard.

6.1.3 Controller design

The optimal feasible trajectory generator has been used as the first stage in the controller design, to generate reference profiles and feedforward control signals, with the aid of inverse models, for generating steering and braking inputs.

Parallel proportional feedback control has been used to augment the feedforward control to correct errors that arise while the vehicle follows the reference trajectory.

Simulations have been used for tuning proportional gains in the steering feedback loops. It has been found that graphical display of three error norms provides useful insight for controller parameter selection.

Consideration of the singular values of the pseudo-inverted linearised input matrix, combined with estimation of yaw errors by means of simulation, has been used to determine a suitable gain for

a proportional feedback brake controller which is used for yaw stabilisation.

Overall, the velocity-based model, optimal trajectory generation routine and controller design strategy lead to a controller that successfully achieves the desired aim: to cause a vehicle to automatically execute an emergency lateral obstacle avoidance manoeuvre at high speed.

6.2 Future work

There is scope for improvement of both the MexCar vehicle model and optimal trajectory generation routine. Under-estimation of weight transfer, and neglection of pitching dynamics, leads to over-estimates of the traction available to the vehicle. Refinement of the weight distribution model would be expected to improve predictive power. An improved model should, in turn, improve the performance of controllers designed using its predictions.

There is wide public interest in active safety technology for vehicles. Many requests for further information arose from an article about this research in the Sunday Telegraph (Gray 2007). However, the system is not yet ready for use on public roads. There has been an underlying assumption throughout this work that the vehicle computer would have full control of the steering and braking inputs during any manoeuvre. In reality, many drivers would be extremely reluctant to surrender complete control of the vehicle during automatic operation, a point made repeatedly during several recent BBC radio interviews (BBC Radio 5 Live 2007*a*,*b*, BBC Radio WM (West Midlands) 2008, BBC World Service 2008). If the driver is to retain some degree of control, several questions arise which would form a useful basis for future research. The area of driver/computer interaction will be particularly important.

When, precisely, should a vehicle computer initiate an emergency lateral manoeuvre? Waiting until the last moment is inherently more risky than earlier action, but drivers would not take kindly to a vehicle that continually pre-empts them.

How and when should the vehicle computer warn the driver and other road users about its intended action? The earlier information can be given to the driver, the more chance he or she has to act, but false alarms could be extremely annoying and distracting, hence increasing danger.

If, during an automatic manoeuvre, vehicle stability is dependent upon individual wheel braking, which cannot be accomplished using the driver's brake pedal, should control of the brakes be returned to the driver? Or should the braking system be redesigned to be more integrated with the steering system for routine driving?

If, during an automatic manoeuvre, the driver moves the steering wheel, should the computer ignore this, attempt to interpret the driver's wishes, or return full control to the driver?

How could an automatic collision avoidance system be implemented such that drivers remain attentive and do not come to rely on the computer at the expense of taking responsibility for their own actions?

The best answer to all these questions may be that the system should not be fully automatic. Perhaps, like existing electronic stability programmes, the computer should only attempt to augment

the driver's actions. In this case, significant work is required to determine how the computer should interpret the drivers intention and how typical drivers will respond to any such intervention.

6.2.1 Wider issues

There are several issues that remain to be addressed before it would be practical to attempt to implement a lateral collision avoidance system on a production vehicle, particularly relating to sensing of the external environment and the manner in which control would pass from the driver to the vehicle computer; then back again. The work presented here focuses purely on control of the vehicle dynamics to perform a fully automatic emergency collision avoidance manoeuvre under the assumptions that there is a clear adjacent lane into which the vehicle may safely move and a decision to proceed has already been taken either by a vehicle management computer or a driver-initiated action.

Performing aggressive lateral manoeuvres at high speed is not a step to be undertaken lightly. Even if it is certain that the car can be controlled satisfactorily, there are other dangers to consider. Other cars may be relatively easy to detect. Pedestrians, particularly children, and cyclists never will be.

Appendix A

Code

A.1 Vehicle model code

A.1.1 Car.hh

```
// -*-c++-*-

/*************************************************************
 * Car.hh declares Car - a non-linear vehicle model.
 *
 * Geraint Paul Bevan <g.bevan@mech.gla.ac.uk>
 * Initial Revision : <2005-06-21>
 * Latest Time-stamp: <2007-08-05 21:18:12 geraint>
 *
 * $Id: Car.hh,v 1.1 2008-01-09 14:21:13 gbevan Exp $
 *
 *************************************************************/

#ifndef _CAR_HH_
#define _CAR_HH_

/// nonlinear two-track model of a car.
/**
 * Car implements a non-linear two-track model of vehicle
 * dynamics. The model is intended primarily for lateral
 * dynamics analysis.
 *
 * During simulation, the continuous states are the
 * longitudinal, lateral and yaw velocity of the vehicle
 * body and the rates are the corresponding accelerations.
```

```
 *
 * The inputs are the wheel longitudinal forces (in their
 * own axis system) and the steering angle. Wheel speeds and
 * forces are retained, thus acting as discrete states.
 */
class Car
{
public:

  Car();

  /// axis is a structure containing three vector elements.
  /**
   * The axis elements are used for containing longitudinal,
   * lateral and yaw components of position, velocity and
   * acceleration.
   */
  struct axis {
    /** longitudinal component */
    double X;
    /** lateral component */
    double Y;
    /** yaw component */
    double Psi;
  };
  /** a structure to contain vector components. */
  typedef struct axis axis;

  /// wheel is an enumeration for indexing wheels.
  /**
   * - FL: Front left;
   * - FR: Front right;
   * - RL: Rear left;
   * - RR: Rear right.
   *
   * A prefix increment operator is defined so that it is
   * possible to loop over each of the wheels with a "for"
   * construct: for (i = FL; i <= RR; ++i) { ... }
   */
  enum wheel { FL = 0, FR, RL, RR };
  /**
   * an enumeration to provide clear and unambiguous
   * reference to each of the vehicle's wheels.
   */
  typedef enum wheel wheel;

  axis get_position(void) const;
  axis get_velocity(void) const;
  axis get_acceleration(void) const;

  double get_friction_coefficient(void) const;
  double get_wheel_lateral_force(const wheel i) const;
  double get_wheel_lateral_speed(const wheel i) const;
  double get_wheel_longitudinal_force(const wheel i) const;
  double get_wheel_longitudinal_speed(const wheel i) const;
  double get_wheel_slip_angle(const wheel i) const;
  double get_wheel_steering_angle(const wheel i) const;
  double get_wheel_vertical_force(const wheel i) const;

  virtual void integrate_euler(double dt);

  void set_acceleration(const axis &accel);
  void set_friction_coefficient(const double friction_coefft);
  void set_velocity(const axis &vel);
  void set_wheel_lateral_force(const wheel i, const double force);
  void set_wheel_longitudinal_force(const wheel i, const double force);
  void set_wheel_slip_angle(const wheel i, const double angle);
  void set_wheel_speed(const wheel i, const double speed);
  void set_wheel_steering_angle(const double angle);
  void set_wheel_vertical_force(const wheel i, const double force);

  void write_parameters(void) const;

protected:
```

```
void update_acceleration(void);
void update_tyre_forces(void);
void update_weight_distribution(void);
void update_wheel_speeds(void);

/// stores the vehicle position (fixed Earth).
/**
 * The position of the centre of gravity of the Car is
 * measured relative to the Earth axis system.
 */
axis position;

/// stores the vehicle velocity.
/**
 * The velocity of the centre of gravity of the Car is
 * measured relative to its body axis system.
 */
axis velocity;

/// stores the vehicle acceleration.
/**
 * The acceleration of the centre of gravity of the Car is
 * measured relative to its body axis system.
 */
axis acceleration;


// parameters

/** vehicle mass (kilogrammes) */
double m;

/** moment of inertia (kilogrammes metres squared)*/
double Izz;

/** longitudinal moment arm to wheels (metres) */
double X[4];

/** lateral moment arm to wheels (metres) */
double Y[4];

/** height of centre of mass (metres) */
double Z0;

/** tyre lateral parameter (non-dimensional, Pacejka) */
double By, Cy, Dy, Ey;

/** gravitational constant (metres per second squared)*/
double g;

/** friction coefficient (-) */
double mu;

// inputs

/** wheel steering angle (radians) */
double delta[4];

/** tyre longitudinal force (Newtons) */
double fx[4];


// intermediate variables

/** tyre slip angles (radians) */
double alpha[4];

/** tyre lateral forces (Newtons) */
double fy[4];

/** wheel longitudinal speeds (metres per second) */
double vx[4];

/** wheel lateral speeds (metres per second) */
double vy[4];
```

```
  /** tyre vertical load (Newtons) */
  double fz[4];
};

Car::wheel &operator++(Car::wheel &w);

#endif // _CAR_HH_
```

A.1.2 Crash.hh

```
// -*-c++-*-

/*************************************************************
 * Crash.hh
 *
 * Geraint Paul Bevan <g.bevan@mech.gla.ac.uk>
 * Initial Revision : <2005-06-23>
 * Latest Time-stamp: <2007-08-05 21:59:13 geraint>
 *
 * $Id: Crash.hh,v 1.1 2008-01-09 14:21:13 gbevan Exp $
 *
 *************************************************************/

#ifndef _CRASH_H_
#define _CRASH_H_

/// An exception that can be thrown.
class Crash
{
public:

  /** can be called with a message which can be read by the
   * error handling routine.
   */
  Crash(const char *message) {
    std::cerr << message << std::endl;
  }
};

#endif // _CRASH_H_
```

A.1.3 Car.cc

```
// -*-c++-*-

/*************************************************************
 * Car.cc defines Car - a non-linear vehicle model.
 *
 * Geraint Paul Bevan <g.bevan@mech.gla.ac.uk>
 * Initial Revision : <2005-06-21>
 * Latest Time-stamp: <2007-08-05 21:20:10 geraint>
 *
 * $Id: Car.cc,v 1.1 2008-01-09 14:21:13 gbevan Exp $
 *
 *************************************************************/

#include <iostream>
#include <cmath>
#include "Crash.hh"
#include "Car.hh"

/// initialises the Car.
/**
 * The mass and moment of inertia are defined along with the
 * moment arms to each of the wheels. Parameters are set to
 * define the lateral tyre characteristics.  The steering
 * and slip angles of each wheel are set to zero, as are the
 * forces and speeds of each wheel.
 */
Car::Car(void) {
  // initial conditions
  position.X = 0.0;
  position.Y = 0.0;
  position.Psi = 0.0;

  velocity.X = 1.0;
  velocity.Y = 0.0;
```

```
  velocity.Psi = 0.0;

  acceleration.X = 0.0;
  acceleration.Y = 0.0;
  acceleration.Psi = 0.0;

  // mass and moment of inertia
  m = 2360;                    // standard: 1900; v220ts
  Izz = 4700.0;   // standard 2870.0; v220ts from kfzpw220.h

  // gravitational constant
  g = 9.81;

  // friction coefficient
  mu = 1.0;

  // moment arms to wheels
  X[FL] = 1.67;
  X[FR] = X[FL];

  X[RL] = -1.41;
  X[RR] = X[RL];

  Y[FL] = -0.80;
  Y[FR] = -Y[FL];

  Y[RL] = Y[FL];
  Y[RR] = Y[FR];

  // height of centre of mass
  Z0    = 0.5;

  // tyre lateral parameters
  By =  18.0;
  Cy =  1.0;
  Dy =  0.9;
  Ey = -1.0;

  // wheel initialisation
  for (wheel i = FL; i <= RR; ++i) {
    alpha[i] = 0.0;
    delta[i] = 0.0;
    fx[i] = 0.0;
    fy[i] = 0.0;
    vx[i] = 0.0;
    vy[i] = 0.0;
    fz[i] = m * g / 4.0;
  }
}

/// performs one Euler integration step.
/**
 * The vehicle acceleration is updated and then integrated
 * over a period of dt seconds to obtain new vehicle
 * velocities; these are integrated in turn to obtain a new
 * vehicle position.
 */
void
Car::integrate_euler(double dt) {
  update_acceleration();

  velocity.X += dt * acceleration.X;
  velocity.Y += dt * acceleration.Y;
  velocity.Psi += dt * acceleration.Psi;

  position.X += dt * (+ velocity.X * cos(position.Psi)
                      - velocity.Y * sin(position.Psi));

  position.Y += dt * (+ velocity.X * sin(position.Psi)
                      + velocity.Y * cos(position.Psi));

  position.Psi += dt * velocity.Psi;
};

/// returns the acceleration of the Car's centre of gravity.
/**
 * The acceleration components are returned in SI units:
```

```
 * metres per second squared and radians per second squared,
 * for translational and rotation units respectively.  These
 * acceleration components are defined relative to the Car's
 * body axis system.
 */
Car::axis
Car::get_acceleration(void) const {
  return acceleration;
}

/// returns the friction coefficient between the tyres and road.
/**
 * A coefficient of 1.0 corresponds to dry asphalt; 0.6 is
 * reasonable for wet roads; for ice it may be 0.05.
 */
double
Car::get_friction_coefficient(void) const {
  return mu;
}

/// returns the position of the Car's centre of gravity.
/**
 * The position components are returned in SI units: metres
 * and radians, for translational and rotation units
 * respectively.  These position components are defined
 * relative to the Earth axis system in which the Car was
 * initialised.
 */
Car::axis
Car::get_position(void) const {
  return position;
}

/// returns the velocity of the Car's centre of gravity.
 /**
 * The velocity components are returned in SI units: metres
 * per second and radians per second, for translational and
 * rotational units respectively.  These velocity components
 * are defined relative to the Car's body axis system.
 */
Car::axis
Car::get_velocity(void) const {
  return velocity;
}

/// returns the lateral force between a wheel and the road.
/**
 * The force is returned in SI units: Newtons.  This force
 * component is defined relative to the wheel's axis system
 * and is therefore perpendicular to the rolling motion of
 * the wheel.
 */
double
Car::get_wheel_lateral_force(const wheel i) const {
  return fy[i];
}

/// returns the lateral speed of a wheel.
/**
 * The speed is returned in SI units: metres per second.
 * This velocity component is defined relative to the
 * wheel's axis system and is therefore perpendicular to the
 * rolling motion of the wheel.
 */
double
Car::get_wheel_lateral_speed(const wheel i) const {
  return vy[i];
}

/// returns the longitudinal force between a wheel and the road.
/**
 * The force is returned in SI units: Newtons.  This force
 * component is defined relative to the wheel's axis system
```

```
 * and is therefore co-linear with the rolling motion of the
 * wheel.
 */
double
Car::get_wheel_longitudinal_force(const wheel i) const {
  return fx[i];
}

/// returns the longitudinal speed of a wheel.
/**
 * The speed is returned in SI units: metres per second.
 * This velocity component is defined relative to the
 * wheel's axis system and is therefore co-linear with the
 * rolling motion of the wheel.
 */
double
Car::get_wheel_longitudinal_speed(const wheel i) const {
  return vx[i];
}

/// returns the wheel slip angle of a wheel relative to the road.
/**
 * The angle is returned in SI units: radians.  The slip
 * angle is the angular difference between the wheel's
 * orientation and its direction of motion.
 */
double
Car::get_wheel_slip_angle(const wheel i) const {
  return alpha[i];
}

/// returns the angle at which a wheel is oriented.
/**
 * The angle is returned in SI units: radians.  The steering
 * angle is defined relative to the Car's body axis system.
 */
double
Car::get_wheel_steering_angle(const wheel i) const {
  return delta[i];
}

/// returns the vertical load on a wheel.
/**
 * The force is returned in SI units: Newtons.
 */
double
Car::get_wheel_vertical_force(const wheel i) const {
  return fz[i];
}

/// sets the acceleration of the Car's centre of gravity.
/**
 * The acceleration components are specified in SI units:
 * metres per second squared and radians per second squared,
 * for translational and rotation units respectively.  These
 * acceleration components are defined relative to the Car's
 * body axis system.
 */
void
Car::set_acceleration(const axis &accel) {
  acceleration = accel;
}

/// sets the friction coefficient between the tyres and road.
/**
 * A coefficient of 1.0 corresponds to dry asphalt; 0.6 is
 * reasonable for wet roads; for ice it may be as 0.05.
 */
void
Car::set_friction_coefficient(const double friction_coefft) {
  mu = friction_coefft;
}

/// sets the velocity of the Car's centre of gravity.
```

```
/**
 * The velocity components are specified in SI units: metres
 * per second and radians per second, for translational and
 * rotational units respectively.  These velocity components
 * are defined relative to the Car's body axis system.  The
 * wheel speeds are updated accordingly.
 */
void
Car::set_velocity(const axis &vel) {
  velocity = vel;
  update_wheel_speeds();
}

/// sets the lateral force between a wheel and the road.
/**
 * The force is specified in SI units: Newtons.  This force
 * component is defined relative to the wheel's axis system
 * and is therefore perpendicular to the rolling motion of
 * the wheel.
 */
void
Car::set_wheel_lateral_force(const wheel i, const double force) {
  fy[i] = force;
}

/// sets the longitudinal force between a wheel and the road.
/**
 * The force is specified in SI units: Newtons.  This force
 * component is defined relative to the wheel's axis system
 * and is therefore co-linear with the rolling motion of the
 * wheel.
 */
void
Car::set_wheel_longitudinal_force(const wheel i, const double force) {
  fx[i] = force;
}

/// sets the wheel slip angle of a wheel relative to the road.
/**
 * The angle is specified in SI units: radians.  The slip
 * angle is the angular difference between the wheel's
 * orientation and its direction of motion.
 */
void
Car::set_wheel_slip_angle(const wheel i, const double angle) {
  alpha[i] = angle;
}

/// sets the longitudinal speed of a wheel relative to the road.
/**
 * The speed is specified in SI units: metres per second.
 * This velocity component is defined relative to the
 * wheel's axis system and is therefore co-linear with the
 * rolling motion of the wheel.
 */
void
Car::set_wheel_speed(const wheel i, const double speed) {
  vx[i] = speed;
}

/// sets the angle at which the front wheels are oriented.
/**
 * The front wheels are constrained to have the same
 * steering angle. The rear wheels remain fixed straight
 * ahead.  The steering angle is defined relative to the
 * Car's body axis system.
 */
void
Car::set_wheel_steering_angle(const double angle) {
  delta[FL] = angle;
  delta[FR] = angle;
  delta[RL] = 0.0;
  delta[RR] = 0.0;
```

```
}

/// sets the vertical load on a wheel.
/**
 * The force is specified in SI units: Newtons.
 */
void
Car::set_wheel_vertical_force(const wheel i, const double force) {
  fz[i] = force;
}

/// updates the vehicle acceleration.
/**
 * Components of acceleration are calculated from the forces
 * generated between the tyres and road at each of the
 * wheels.
 */
void
Car::update_acceleration(void) {
  update_weight_distribution();
  update_tyre_forces();

  axis F = {0.0, 0.0, 0.0};

  for (wheel i = FL; i <= RR; ++i) {

    // traction saturation
    double ft = sqrt(fx[i]*fx[i] + fy[i]*fy[i]);
    double fmax = mu * fz[i];
    if (ft > fmax) {
      std::clog << "Traction saturation" << std::endl;
      fx[i] *= fmax/ft;
      fy[i] *= fmax/ft;
    }

    // wheel rotation
    double FX = + fx[i]*cos(delta[i]) - fy[i]*sin(delta[i]);
    double FY = + fx[i]*sin(delta[i]) + fy[i]*cos(delta[i]);
    double MZ = -Y[i]*FX + X[i]*FY;
    F.X += FX;
    F.Y += FY;
    F.Psi += MZ;
  }

  // Newton II
  acceleration.X = F.X / m;
  acceleration.Y = F.Y / m;
  acceleration.Psi = F.Psi / Izz;
}

/// updates the lateral tyre forces.
/**
 * Lateral tyre forces are calculated using Pacejka's magic
 * formula and the slip angles at each of the wheels.
 */
void
Car::update_tyre_forces(void) {
  update_wheel_speeds();
  double phi;
  for (wheel i = FL; i <= RR; ++i) {
    if (vx[i] != 0.0) {
      alpha[i] = atan(vy[i] / vx[i]);
    } else if (vy[i] == 0.0) {
      alpha[i] = 0.0;
    } else {
      alpha[i] = 0.5 * M_PI;
      if (vy[i] < 0.0) {
        alpha[i] = - alpha[i];
      }
    }

    // Pacejka's magic formula
    phi = By * alpha[i];
    fy[i] = - Dy * sin(Cy * atan(phi - Ey * (phi - atan(phi))));

    fy[i] *= mu * fz[i];
```

```
  }
};

/// dgelss (LAPACK) is used for pseudoinversion
/**
 * See netlib.org for more details about LAPACK
 * http://www.netlib.org/lapack/index.html
 */
extern "C" {
  void dgelss_(int *m, int *n, int *nrhs,
               double *a, int *lda, double *b, int *ldb,
               double *s, double *rcond, int *rank,
               double *w, int* lwork, int *info);
}

/// updates the weight distribution.
/**
 * The vehicle is assumed to be rigid so the moments from
 * vertical loads balance those from lateral and
 * longitudinal forces.
 *
 *       ||  1  1  1  1           m*g  ||
 *  min  || Y1 Y2 Y3 Y4 * Fz  -  -Z0*Fy ||
 *   Fz  || X1 X2 X3 X4          -Z0*Fx ||2
 *
 * dgelss (LAPACK) is used to calculate the pseudoinverse
 */
void
Car::update_weight_distribution(void) {
  double Fx = m*acceleration.X;
  double Fy = m*acceleration.Y;

  int nrow=3, ncol=4, nrhs=4;
  double rcond = 0.0;

  double A[3][4] = {            // A will be pseudoinverted
    {  1.0 ,  1.0 ,  1.0 ,  1.0  },
    { Y[FL], Y[FR], Y[RL], Y[RR] },
    { X[FL], X[FR], X[RL], X[RR] }
  };
  double B[4][4] = {            // B will become the answer
    { 1.0, 0.0, 0.0, 0.0 },
    { 0.0, 1.0, 0.0, 0.0 },
    { 0.0, 0.0, 1.0, 0.0 },
    { 0.0, 0.0, 0.0, 1.0 }
  };
  double S[3][4] = {            // singular values
    { 0.0, 0.0, 0.0, 0.0 },
    { 0.0, 0.0, 0.0, 0.0 },
    { 0.0, 0.0, 0.0, 0.0 }
  };

  // Fortran arrays are the transpose of C arrays
  double AT[ncol*nrow], BT[nrhs*nrhs], ST[ncol*nrow];
  for (int i = 0; i < ncol; ++i) {
    for (int j = 0; j < nrow; ++j) {
      AT[j + nrow*i] = A[j][i];
      ST[j + nrow*i] = S[j][i];
    }
  }
  for (int i = 0; i < nrhs; ++i) {
    for (int j = 0; j < nrhs; ++j) {
      BT[j + nrhs*i] = B[j][i];
    }
  }

  // LAPACK: solves min_x || Ax - b ||
  int lda=nrow, ldb=ncol, lwork=16;
  int rank=0, info=0;
  double work[lwork];
  dgelss_(&nrow, &ncol, &nrhs,
          AT, &lda, BT, &ldb, ST, &rcond, &rank,
          work, &lwork, &info);

  for (int i = 0; i < nrhs; ++i) {
    for (int j = 0; j < nrhs; ++j) {
```

```
      B[j][i] = BT[j + nrhs*i];
    }
  } // B contains pseudoinverse of A

  for (wheel i = FL; i <= RR; ++i) {
    fz[i] = B[i][0] * m*g
      + B[i][1] * (-Z0*Fy)
      + B[i][2] * (-Z0*Fx);
  }

#ifdef DEBUG
  for (int i = 0; i < ncol; ++i) {
    for (int j = 0; j < nrow; ++j) {
      A[j][i] = AT[j + nrow*i];
      S[j][i] = ST[j + nrow*i];
    }
  }
#endif
}

/// updates the velocity at each wheel.
/**
 * The velocity of each wheel is calculated from the Car's
 * body velocity and consideration of geometry.
 */
void
Car::update_wheel_speeds(void) {
  for (wheel i = FL; i <= RR; ++i) {
    vx[i] =
      + (velocity.X - Y[i]*velocity.Psi) * cos(delta[i])
      + (velocity.Y + X[i]*velocity.Psi) * sin(delta[i]);
    vy[i] =
      - (velocity.X - Y[i]*velocity.Psi) * sin(delta[i])
      + (velocity.Y + X[i]*velocity.Psi) * cos(delta[i]);
  }
}

/// writes vehicle parameters to standard output.
void
Car::write_parameters(void) const {
  std::cout << "=== Vehicle parameters ===" << std::endl
            << "m\t" << m << std::endl
            << "Izz\t" << Izz << std::endl
            << "mu\t" << mu << std::endl
            << "X[FL]\t" << X[FL] << std::endl
            << "X[FR]\t" << X[FR] << std::endl
            << "X[RL]\t" << X[RL] << std::endl
            << "X[RR]\t" << X[RR] << std::endl
            << "Y[FL]\t" << Y[FL] << std::endl
            << "Y[FR]\t" << Y[FR] << std::endl
            << "Y[RL]\t" << Y[RL] << std::endl
            << "Y[RR]\t" << Y[RR] << std::endl
            << "By\t" << By << std::endl
            << "Cy\t" << Cy << std::endl
            << "Dy\t" << Dy << std::endl
            << "Ey\t" << Ey << std::endl
            << "==========================" << std::endl;
}

/// allows "for" constructs to loop over the wheels.
/**
 * Implementation of the prefix increment operator allows
 * constructs such as:
 *
 * for (wheel i = FL; i < RR; ++i) {
 *    do_something_to_wheel(i);
 * }
*/
Car::wheel &
operator++(Car::wheel &w) {
  int i = w;
  w = static_cast<Car::wheel>(i + 1);
  return w;
}
```

A.1.4 DldCar_usage.hh

```
/** \file DldCar_usage.hh
 *
 * contains the usage strings for the functions
 * which are exported via the Matlab and
 * GNU Octave dynamic link library interfaces.
 */

#ifndef DLDCAR_HH
#define DLDCAR_HH
#endif

#if (! defined (OCTAVE) & ! defined (MATLAB))
#error Define OCTAVE or MATLAB
#endif

#include <map>
#include <string>

/** serves to prevent errors from occurring if
 * usage_text is called by a function for which
 * a description of the usage has not been yet
 * been written.
 */
const char *
usage_text(std::string function,
           std::string arguments,
           std::string null)
{return "";};

/** returns information about the usage of functions. */
const char *
usage_text(std::string function)
{
  static std::map<std::string,std::string> arguments;
  static std::map<std::string,std::string> description;

  static bool is_initialised = false;
  if (! is_initialised) {
    is_initialised = true;

    arguments["get_A"] = "";
    description["get_A"] =
      "Return the state matrix of the linear\n"
      "velocity-based vehicle model.\n";

    arguments["get_B"] = "";
    description["get_B"] =
      "Return the input matrix of the linear\n"
      "velocity-based vehicle model.\n";

    arguments["get_friction_coefficient"] = "";
    description["get_friction_coefficient"] =
      "Return the friction coefficient\n"
      "between the tyres and the road.\n";

    arguments["get_position"] = "";
    description["get_position"] =
      "Return the position of the vehicle\n"
      "(in the Earth axis system).\n";

    arguments["get_velocity"] = "";
    description["get_velocity"] =
      "Return the velocity of the vehicle\n"
      "(in its body axis system).\n";

    arguments["get_acceleration"] = "";
    description["get_acceleration"] =
      "Return the acceleration of the vehicle\n"
      "(in its body axis system).\n";

    arguments["get_wheel_lateral_forces"] = "";
    description["get_wheel_lateral_forces"] =
      "Return the lateral force of each wheel\n"
      "as a column vector [FL; FR; RL; RR].\n";
```

```
arguments["get_wheel_lateral_speeds"] = "";
description["get_wheel_lateral_speeds"] =
  "Return the longitudinal speed of each wheel\n"
  "as a column vector [FL; FR; RL; RR].\n";

arguments["get_wheel_longitudinal_forces"] = "";
description["get_wheel_longitudinal_forces"] =
  "Return the longitudinal force of each wheel\n"
  "as a column vector [FL; FR; RL; RR].\n";

arguments["get_wheel_longitudinal_speeds"] = "";
description["get_wheel_longitudinal_speeds"] =
  "Return the longitudinal speed of each wheel\n"
  "as a column vector [FL; FR; RL; RR].\n";

arguments["get_wheel_slip_angles"] = "";
description["get_wheel_slip_angles"] =
  "Return the slip angle of each wheel\n"
  "as a column vector [FL; FR; RL; RR].\n";

arguments["get_wheel_steering_angles"] = "";
description["get_wheel_steering_angles"] =
  "Return the steering angle of each wheel\n"
  "as a column vector [FL; FR; RL; RR].\n";

arguments["get_wheel_vertical_forces"] = "";
description["get_wheel_vertical_forces"] =
  "Return the vertical load on each wheel\n"
  "as a column vector [FL; FR; RL; RR].\n";

arguments["integrate_euler_linear"] = "DT, N";
description["integrate_euler_linear"] =
  "Perform N steps of Euler integration with\n"
  "timestep DT seconds using the derived linear\n"
  "model of the vehicle.\n";

arguments["integrate_euler_nonlinear"] = "DT, N";
description["integrate_euler_nonlinear"] =
  "Perform N steps of Euler integration with\n"
  "timestep DT seconds using the underlying\n"
  "nonlinear model of the vehicle.\n";

arguments["set_acceleration"] = "A";
description["set_acceleration"] =
  "Set the vehicle acceleration with the\n"
  "column vector A [longitudinal; lateral; yaw]\n"
  "(in the body axis system).\n";

arguments["set_friction_coefficient"] = "mu";
description["set_friction_coefficient"] =
  "Set the friction coefficient\n"
  "between the tyres and the road.\n";

arguments["set_velocity"] = "V";
description["set_velocity"] =
  "Set the vehicle velocity with the\n"
  "column vector V [longitudinal; lateral; yaw]\n"
  "(in the body axis system).\n";

arguments["set_wheel_longitudinal_forces"] = "F";
description["set_wheel_longitudinal_forces"] =
  "Set the longitudinal tyre forces with the\n"
  "column vector F [FL; FR; RL; RR]\n"
  "(in the wheel axis systems).\n";

arguments["set_wheel_slip_angles"] = "ALPHA";
description["set_wheel_slip_angles"] =
  "Set the slip angles (radians) with the\n"
  "column vector ALPHA [FL; FR; RL; RR].\n";

arguments["set_wheel_speeds"] = "VX";
description["set_wheel_speeds"] =
  "Set the wheel speeds (m/s) with the\n"
  "column vector VX [FL; FR; RL; RR]\n"
  "(in the wheel axis systems).\n";

arguments["set_wheel_steering_angle"] = "DELTA";
description["set_wheel_steering_angle"] =
```

```
      "Set the steering angle of the front wheels\n"
      "to DELTA radians.\n";

    arguments["set_wheel_vertical_forces"] = "F";
    description["set_wheel_vertical_forces"] =
      "Set the vertical wheel loads with the\n"
      "column vector F [FL; FR; RL; RR]\n"
      "(in the wheel axis systems).\n";

    arguments["write_parameters"] = "";
    description["write_parameters"] =
      "Writes internal parameters.\n"
      "If no car is initialised,\n"
      " a new one is created.\n";
  }

  std::string s("");
#if OCTAVE
  s += "OctCar_" + function + "(" + arguments[function] + ")\n";
#endif
#if MATLAB
  s += "MexCar('" + function + "'";
  if (arguments[function] != "") {
    s += ", " + arguments[function];
  }
  s += ")\n";
#endif
  s = s + "\n" + description[function];
  return s.c_str();
}
```

A.1.5 MexCar.cc

```
// -*-c++-*-

/*****************************************************************
 * MexCar.cc
 *
 * Geraint Paul Bevan <g.bevan@mech.gla.ac.uk>
 * Initial Revision : <2005-08-10>
 * Latest Time-stamp: <2007-08-06 16:40:53 geraint>
 *****************************************************************/

/** \file MexCar.cc
 * implements an interface to the LinearisableCar class.
 * An instance of the model can be created
 * and initialised by calling the function
 * "MexCar" from the matlab prompt.
 *
 * Functions of LinearisableCar for which a handle has
 * been created can then be called by passing the
 * function name as an argument.
 *
 * For example, after initialising the model with
 * matlab> MexCar();
 * the state matrix can be obtained by:
 * matlab> MexCar('get_A');
 */

#include <iostream>
#include <string>
#include <map>

#include <mex.h>
#include <LinearisableCar.hh>
#include <DldCar_usage.hh>

#ifdef DEBUG
#define DEBUGPRINT printf
#else
#define DEBUGPRINT //
#endif

static LinearisableCar *car;

/// function list.
```

```
/**
 * Calling this function presents a list of functions which
 * may be called from the Matlab interpreter after the
 * LinearisableCar has been initialised.
 */
const char *usage_MexCar =
"MexCar()\n"
"\n"
"A Matlab interface to a linearisable car model:\n"
"Initialises a velocity-based model of a car\n"
"and displays the vehicle parameters.\n"
"\n"
"Related function:\n"
" MexCar_get_A\n"
" MexCar_get_B\n"
" MexCar_get_position\n"
" MexCar_get_velocity\n"
" MexCar_get_acceleration\n"
" MexCar_get_friction_coefficient\n"
" MexCar_get_wheel_lateral_forces\n"
" MexCar_get_wheel_lateral_speeds\n"
" MexCar_get_wheel_longitudinal_forces\n"
" MexCar_get_wheel_longitudinal_speeds\n"
" MexCar_get_wheel_slip_angles\n"
" MexCar_get_wheel_steering_angles\n"
" MexCar_get_wheel_vertical_forces\n"
" MexCar_integrate_euler_nonlinear\n"
" MexCar_integrate_euler_linear\n"
" MexCar_set_acceleration\n"
" MexCar_set_friction_coefficient\n"
" MexCar_set_velocity\n"
" MexCar_set_wheel_longitudinal_forces\n"
" MexCar_set_wheel_slip_angles\n"
" MexCar_set_wheel_speeds\n"
" MexCar_set_wheel_steering_angle\n"
" MexCar_set_wheel_vertical_forces\n"
" MexCar_write_parameters\n";

/** function calling routine */
void CallMexSubfunction(const char *subfunction,
                        int nlhs, mxArray *plhs[],
                        int nrhs, const mxArray *prhs[]);

/** initialises the LinearisableCar. */
void mexFunction(int nlhs, mxArray *plhs[],
                 int nrhs, const mxArray *prhs[])
{
  if (nrhs == 0) {
    if (car != NULL) {
      delete(car);
    }
    std::cout << usage_MexCar << std::endl;
    car = new LinearisableCar();
    car->write_parameters();
  } else {
    char buf[128];
    mxGetString(prhs[0], buf, sizeof(buf)-1);
    std::string s(buf);
    CallMexSubfunction(buf, nlhs, plhs, nrhs-1, prhs+1);
  }
  return;
}

void
MexCar_get_A(int nlhs, mxArray *plhs[],
             int nrhs, const mxArray *prhs[])
{
  if (nrhs != 0) {
    mexWarnMsgTxt(usage_text("get_A"));
    mexWarnMsgTxt("ignoring extra arguments");
  }

  double *A_array;
  plhs[0] = mxCreateDoubleMatrix(LinearisableCar::NX,LinearisableCar::NX,mxREAL);
  A_array = mxGetPr(plhs[0]);
  car->get_A(A_array);
```

```
  return;
}

void
MexCar_get_B(int nlhs, mxArray *plhs[],
             int nrhs, const mxArray *prhs[])
{
  if (nrhs != 0) {
    mexWarnMsgTxt(usage_text("get_B"));
    mexWarnMsgTxt("ignoring extra arguments");
  }

  double *B_array;
  plhs[0] = mxCreateDoubleMatrix(LinearisableCar::NX,LinearisableCar::NU,mxREAL);
  B_array = mxGetPr(plhs[0]);
  car->get_B(B_array);
  return;
}

void
MexCar_get_position(int nlhs, mxArray *plhs[],
                    int nrhs, const mxArray *prhs[])
{
  if (nrhs != 0) {
    mexWarnMsgTxt(usage_text("get_position"));
    mexWarnMsgTxt("ignoring extra arguments");
  }

  Car::axis position;
  position = car->get_position();

  double *P_array;
  plhs[0] = mxCreateDoubleMatrix(3,1,mxREAL);
  P_array = mxGetPr(plhs[0]);
  P_array[0] = position.X;
  P_array[1] = position.Y;
  P_array[2] = position.Psi;
  return;
}

void
MexCar_get_velocity(int nlhs, mxArray *plhs[],
                    int nrhs, const mxArray *prhs[])
{
  if (nrhs != 0) {
    mexWarnMsgTxt(usage_text("get_velocity"));
    mexWarnMsgTxt("ignoring extra arguments");
  }

  Car::axis velocity;
  velocity = car->get_velocity();

  double *V_array;
  plhs[0] = mxCreateDoubleMatrix(3,1,mxREAL);
  V_array = mxGetPr(plhs[0]);
  V_array[0] = velocity.X;
  V_array[1] = velocity.Y;
  V_array[2] = velocity.Psi;
  return;
}

void
MexCar_get_acceleration(int nlhs, mxArray *plhs[],
                        int nrhs, const mxArray *prhs[])
{
  if (nrhs != 0) {
    mexWarnMsgTxt(usage_text("get_acceleration"));
    mexWarnMsgTxt("ignoring extra arguments");
  }

  Car::axis acceleration;
  acceleration = car->get_acceleration();

  double *A_array;
  plhs[0] = mxCreateDoubleMatrix(3,1,mxREAL);
  A_array = mxGetPr(plhs[0]);
  A_array[0] = acceleration.X;
  A_array[1] = acceleration.Y;
```

```
  A_array[2] = acceleration.Psi;
  return;
}

void
MexCar_get_friction_coefficient(int nlhs, mxArray *plhs[],
                                int nrhs, const mxArray *prhs[])
{
  if (nrhs != 0) {
    mexWarnMsgTxt(usage_text("get_friction_coefficient"));
    mexWarnMsgTxt("ignoring extra arguments");
  }

  double *mu;
  plhs[0] = mxCreateDoubleMatrix(1,1,mxREAL);
  mu = mxGetPr(plhs[0]);
  mu[0] = car->get_friction_coefficient();;
  return;
}

void
MexCar_get_wheel_lateral_forces(int nlhs, mxArray *plhs[],
                                int nrhs, const mxArray *prhs[])
{
  if (nrhs != 0) {
    mexWarnMsgTxt(usage_text("get_wheel_lateral_forces"));
    mexWarnMsgTxt("ignoring extra arguments");
  }

  double *Fy;
  plhs[0] = mxCreateDoubleMatrix(4,1,mxREAL);
  Fy = mxGetPr(plhs[0]);
  Fy[0] = car->get_wheel_lateral_force(Car::FL);
  Fy[1] = car->get_wheel_lateral_force(Car::FR);
  Fy[2] = car->get_wheel_lateral_force(Car::RL);
  Fy[3] = car->get_wheel_lateral_force(Car::RR);
  return;
}

void
MexCar_get_wheel_lateral_speeds(int nlhs, mxArray *plhs[],
                                int nrhs, const mxArray *prhs[])
{
  if (nrhs != 0) {
    mexWarnMsgTxt(usage_text("get_wheel_lateral_speeds"));
    mexWarnMsgTxt("ignoring extra arguments");
  }

  double *vy;
  plhs[0] = mxCreateDoubleMatrix(4,1,mxREAL);
  vy = mxGetPr(plhs[0]);
  vy[0] = car->get_wheel_lateral_speed(Car::FL);
  vy[1] = car->get_wheel_lateral_speed(Car::FR);
  vy[2] = car->get_wheel_lateral_speed(Car::RL);
  vy[3] = car->get_wheel_lateral_speed(Car::RR);
  return;
}

void
MexCar_get_wheel_longitudinal_forces(int nlhs, mxArray *plhs[],
                                     int nrhs, const mxArray *prhs[])
{
  if (nrhs != 0) {
    mexWarnMsgTxt(usage_text("get_wheel_longitudinal_forces"));
    mexWarnMsgTxt("ignoring extra arguments");
  }

  double *Fx;
  plhs[0] = mxCreateDoubleMatrix(4,1,mxREAL);
  Fx = mxGetPr(plhs[0]);
  Fx[0] = car->get_wheel_longitudinal_force(Car::FL);
  Fx[1] = car->get_wheel_longitudinal_force(Car::FR);
  Fx[2] = car->get_wheel_longitudinal_force(Car::RL);
  Fx[3] = car->get_wheel_longitudinal_force(Car::RR);
  return;
}

void
```

```
MexCar_get_wheel_longitudinal_speeds(int nlhs, mxArray *plhs[],
                                     int nrhs, const mxArray *prhs[])
{
  if (nrhs != 0) {
    mexWarnMsgTxt(usage_text("get_wheel_longitudinal_speeds"));
    mexWarnMsgTxt("ignoring extra arguments");
  }

  double *vx;
  plhs[0] = mxCreateDoubleMatrix(4,1,mxREAL);
  vx = mxGetPr(plhs[0]);
  vx[0] = car->get_wheel_longitudinal_speed(Car::FL);
  vx[1] = car->get_wheel_longitudinal_speed(Car::FR);
  vx[2] = car->get_wheel_longitudinal_speed(Car::RL);
  vx[3] = car->get_wheel_longitudinal_speed(Car::RR);
  return;
}

void
MexCar_get_wheel_slip_angles(int nlhs, mxArray *plhs[],
                             int nrhs, const mxArray *prhs[])
{
  if (nrhs != 0) {
    mexWarnMsgTxt(usage_text("get_wheel_slip_angles"));
    mexWarnMsgTxt("ignoring extra arguments");
  }

  double *alpha;
  plhs[0] = mxCreateDoubleMatrix(4,1,mxREAL);
  alpha = mxGetPr(plhs[0]);
  alpha[0] = car->get_wheel_slip_angle(Car::FL);
  alpha[1] = car->get_wheel_slip_angle(Car::FR);
  alpha[2] = car->get_wheel_slip_angle(Car::RL);
  alpha[3] = car->get_wheel_slip_angle(Car::RR);
  return;
}

void
MexCar_get_wheel_steering_angles(int nlhs, mxArray *plhs[],
                                 int nrhs, const mxArray *prhs[])
{
  if (nrhs != 0) {
    mexWarnMsgTxt(usage_text("get_wheel_steering_angles"));
    mexWarnMsgTxt("ignoring extra arguments");
  }

  double *delta;
  plhs[0] = mxCreateDoubleMatrix(4,1,mxREAL);
  delta = mxGetPr(plhs[0]);
  delta[0] = car->get_wheel_steering_angle(Car::FL);
  delta[1] = car->get_wheel_steering_angle(Car::FR);
  delta[2] = car->get_wheel_steering_angle(Car::RL);
  delta[3] = car->get_wheel_steering_angle(Car::RR);
  return;
}

void
MexCar_get_wheel_vertical_forces(int nlhs, mxArray *plhs[],
                                 int nrhs, const mxArray *prhs[])
{
  if (nrhs != 0) {
    mexWarnMsgTxt(usage_text("get_wheel_vertical_forces"));
    mexWarnMsgTxt("ignoring extra arguments");
  }

  double *Fz;
  plhs[0] = mxCreateDoubleMatrix(4,1,mxREAL);
  Fz = mxGetPr(plhs[0]);
  Fz[0] = car->get_wheel_vertical_force(Car::FL);
  Fz[1] = car->get_wheel_vertical_force(Car::FR);
  Fz[2] = car->get_wheel_vertical_force(Car::RL);
  Fz[3] = car->get_wheel_vertical_force(Car::RR);
  return;
}

void
MexCar_integrate_euler_linear(int nlhs, mxArray *plhs[],
                              int nrhs, const mxArray *prhs[])
```

```
{
  if (nrhs != 2) {
    mexWarnMsgTxt(usage_text("integrate_euler_linear"));
    mexErrMsgTxt("incorrect number of arguments");
    return;
  }
  double *arg1;
  double *arg2;
  arg1 = mxGetPr(prhs[0]);
  arg2 = mxGetPr(prhs[1]);
  double dt = arg1[0];
  int n = static_cast<int>(arg2[0]);
  for (int i = 0; i < n; i++) {
    DEBUGPRINT("n = %i\n", n);
    car->integrate_euler(dt);
  }
  return;
}

void
MexCar_integrate_euler_nonlinear(int nlhs, mxArray *plhs[],
                                 int nrhs, const mxArray *prhs[])
{
  if (nrhs != 2) {
    mexWarnMsgTxt(usage_text("integrate_euler_nonlinear"));
    mexErrMsgTxt("incorrect number of arguments");
    return;
  }
  double *args;
  double dt = *mxGetPr(prhs[0]);
  int n = static_cast<int>(*mxGetPr(prhs[1]));
  for (int i = 0; i < n; i++) {
    car->Car::integrate_euler(dt);
  }
  return;
}

void
MexCar_set_acceleration(int nlhs, mxArray *plhs[],
                        int nrhs, const mxArray *prhs[])
{
  if (nrhs != 1) {
    mexWarnMsgTxt(usage_text("set_acceleration"));
    mexErrMsgTxt("incorrect number of arguments");
    return;
  }

  double *A;
  A = mxGetPr(prhs[0]);

  Car::axis acceleration;
  acceleration.X = A[0];
  acceleration.Y = A[1];
  acceleration.Psi = A[2];
  DEBUGPRINT("Setting acceleration = [%f;%f;%f] ... ", A[0], A[1], A[2]);
  car->set_acceleration(acceleration);
  DEBUGPRINT("done.\n");
  return;
}

void
MexCar_set_friction_coefficient(int nlhs, mxArray *plhs[],
                                int nrhs, const mxArray *prhs[])
{
  if (nrhs != 1) {
    mexWarnMsgTxt(usage_text("set_friction_coefficient"));
    mexErrMsgTxt("incorrect number of arguments");
    return;
  }
  double *mu;
  mu = mxGetPr(prhs[0]);
  DEBUGPRINT("Setting mu = %f ... ", *mu);
  car->set_friction_coefficient(*mu);
  DEBUGPRINT("done.\n");
  return;
}

void
```

```
MexCar_set_velocity(int nlhs, mxArray *plhs[],
                    int nrhs, const mxArray *prhs[])
{
  if (nrhs != 1) {
    mexWarnMsgTxt(usage_text("set_velocity"));
    mexErrMsgTxt("incorrect number of arguments");
    return;
  }

  double *V;
  V = mxGetPr(prhs[0]);

  Car::axis velocity;
  velocity.X = V[0];
  velocity.Y = V[1];
  velocity.Psi = V[2];
  DEBUGPRINT("Setting velocity = [%f;%f;%f] ... ", V[0], V[1], V[2]);
  car->set_velocity(velocity);
  DEBUGPRINT("done.\n");
  return;
}

void
MexCar_set_wheel_longitudinal_forces(int nlhs, mxArray *plhs[],
                                     int nrhs, const mxArray *prhs[])
{
  if (nrhs != 1) {
    mexWarnMsgTxt(usage_text("set_wheel_longitudinal_forces"));
    mexErrMsgTxt("incorrect number of arguments");
    return;
  }

  double *Fx;
  Fx = mxGetPr(prhs[0]);
  car->set_wheel_longitudinal_force(Car::FL, Fx[0]);
  car->set_wheel_longitudinal_force(Car::FR, Fx[1]);
  car->set_wheel_longitudinal_force(Car::RL, Fx[2]);
  car->set_wheel_longitudinal_force(Car::RR, Fx[3]);
  return;
}

void
MexCar_set_wheel_slip_angles(int nlhs, mxArray *plhs[],
                             int nrhs, const mxArray *prhs[])
{
  if (nrhs != 1) {
    mexWarnMsgTxt(usage_text("set_wheel_slip_angles"));
    mexErrMsgTxt("incorrect number of arguments");
    return;
  }

  double *alpha;
  alpha = mxGetPr(prhs[0]);
  car->set_wheel_slip_angle(Car::FL, alpha[0]);
  car->set_wheel_slip_angle(Car::FR, alpha[1]);
  car->set_wheel_slip_angle(Car::RL, alpha[2]);
  car->set_wheel_slip_angle(Car::RR, alpha[3]);
  return;
}

void
MexCar_set_wheel_speeds(int nlhs, mxArray *plhs[],
                        int nrhs, const mxArray *prhs[])
{
  if (nrhs != 1) {
    mexWarnMsgTxt(usage_text("set_wheel_speeds"));
    mexErrMsgTxt("incorrect number of arguments");
    return;
  }

  double *vx;
  vx = mxGetPr(prhs[0]);
  car->set_wheel_speed(Car::FL, vx[0]);
  car->set_wheel_speed(Car::FR, vx[1]);
  car->set_wheel_speed(Car::RL, vx[2]);
  car->set_wheel_speed(Car::RR, vx[3]);
  return;
}
```

```
void
MexCar_set_wheel_steering_angle(int nlhs, mxArray *plhs[],
                                int nrhs, const mxArray *prhs[])
{
  if (nrhs != 1) {
    mexWarnMsgTxt(usage_text("set_wheel_steering_angle"));
    mexErrMsgTxt("incorrect number of arguments");
    return;
  }
  double *angle;
  angle = mxGetPr(prhs[0]);
  DEBUGPRINT("Setting delta = %f ... ", *angle);
  car->set_wheel_steering_angle(*angle);
  DEBUGPRINT("done.\n");
  return;
}

void
MexCar_set_wheel_vertical_forces(int nlhs, mxArray *plhs[],
                                 int nrhs, const mxArray *prhs[])
{
  if (nrhs != 1) {
    mexWarnMsgTxt(usage_text("set_wheel_vertical_forces"));
    mexErrMsgTxt("incorrect number of arguments");
    return;
  }

  double *Fz;
  Fz = mxGetPr(prhs[0]);
  car->set_wheel_vertical_force(Car::FL, Fz[0]);
  car->set_wheel_vertical_force(Car::FR, Fz[1]);
  car->set_wheel_vertical_force(Car::RL, Fz[2]);
  car->set_wheel_vertical_force(Car::RR, Fz[3]);
  return;
}

void
MexCar_write_parameters(int nlhs, mxArray *plhs[],
                        int nrhs, const mxArray *phrs[])
{
  if (car == NULL) {
    car = new LinearisableCar();
  }
  car->write_parameters();
  return;
}

typedef void (*pMexF)(int, mxArray*[], int, const mxArray*[]);

void
CallMexSubfunction(const char *subfunction,
                   int nlhs, mxArray *plhs[],
                   int nrhs, const mxArray *prhs[])
{
  pMexF f;
  if (! strcmp(subfunction, "get_A")) {
    f = &MexCar_get_A;
  } else if (! strcmp(subfunction, "get_B")) {
    f = &MexCar_get_B;
  } else if (! strcmp(subfunction, "get_position")) {
    f = &MexCar_get_position;
  } else if (! strcmp(subfunction, "get_velocity")) {
    f = &MexCar_get_velocity;
  } else if (! strcmp(subfunction, "get_acceleration")) {
    f = &MexCar_get_acceleration;
  } else if (! strcmp(subfunction, "get_friction_coefficient")) {
    f = &MexCar_get_friction_coefficient;
  } else if (! strcmp(subfunction, "get_wheel_lateral_forces")) {
    f = &MexCar_get_wheel_lateral_forces;
  } else if (! strcmp(subfunction, "get_wheel_lateral_speeds")) {
    f = &MexCar_get_wheel_lateral_speeds;
  } else if (! strcmp(subfunction, "get_wheel_longitudinal_forces")) {
    f = &MexCar_get_wheel_longitudinal_forces;
  } else if (! strcmp(subfunction, "get_wheel_longitudinal_speeds")) {
    f = &MexCar_get_wheel_longitudinal_speeds;
  } else if (! strcmp(subfunction, "get_wheel_slip_angles")) {
    f = &MexCar_get_wheel_slip_angles;
```

```
  } else if (! strcmp(subfunction, "get_wheel_steering_angles")) {
    f = &MexCar_get_wheel_steering_angles;
  } else if (! strcmp(subfunction, "get_wheel_vertical_forces")) {
    f = &MexCar_get_wheel_vertical_forces;
  } else if (! strcmp(subfunction, "integrate_euler_linear")) {
    f = &MexCar_integrate_euler_linear;
  } else if (! strcmp(subfunction, "integrate_euler_nonlinear")) {
    f = &MexCar_integrate_euler_nonlinear;
  } else if (! strcmp(subfunction, "set_acceleration")) {
    f = &MexCar_set_acceleration;
  } else if (! strcmp(subfunction, "set_friction_coefficient")) {
    f = &MexCar_set_friction_coefficient;
  } else if (! strcmp(subfunction, "set_velocity")) {
    f = &MexCar_set_velocity;
  } else if (! strcmp(subfunction, "set_wheel_longitudinal_forces")) {
    f = &MexCar_set_wheel_longitudinal_forces;
  } else if (! strcmp(subfunction, "set_wheel_slip_angles")) {
    f = &MexCar_set_wheel_slip_angles;
  } else if (! strcmp(subfunction, "set_wheel_speeds")) {
    f = &MexCar_set_wheel_speeds;
  } else if (! strcmp(subfunction, "set_wheel_steering_angle")) {
    f = &MexCar_set_wheel_steering_angle;
  } else if (! strcmp(subfunction, "set_wheel_vertical_forces")) {
    f = &MexCar_set_wheel_vertical_forces;
  } else if (! strcmp(subfunction, "write_parameters")) {
    f = &MexCar_write_parameters;
  } else {
    mexErrMsgTxt("Unknown subfunction");
  }
  f(nlhs, plhs, nrhs, prhs);
  return;
}
```

A.1.6 OctCar.cc

```
// -*-c++-*-

/*************************************************************
 * OctCar.cc
 *
 * Geraint Paul Bevan <g.bevan@mech.gla.ac.uk>
 * Initial Revision : <2005-06-24>
 * Latest Time-stamp: <2007-08-05 21:31:17 geraint>
 *
 *************************************************************/

/** \file OctCar.cc implements an interface to the
 * LinearisableCar class.  An instance of the model can be
 * created and initialised by calling the function "OctCar"
 * from the octave prompt.
 *
 * Functions of LinearisableCar for which a handle has been
 * created can then be called by appending the function name
 * to the prefix OctCar_.
 *
 * For example, after initialising the model with
 * - octave> OctCar();
 * the state matrix can be obtained by:
 * - octave> OctCar_get_A();
 *
 * A list of available functions can be obtained by typing
 * OctCar and pressing the tab key.
 */

/** \file MexCar.m
 * implements a simple wrapper so that the OctCar
 * functions defined in OctCar.cc can be called
 * with the same names and arguments as the functions
 * defined in MexCar.cc.
 * Thus common scripts can be written to work identically
 * in either GNU Octave or Matlab.
 *
 * Example: to call the function OctCar_get_B() using the
 * Mex-style interface, type MexCar('get_B').
 */
```

```
#include <octave/oct.h>
#include <LinearisableCar.hh>
#include <DldCar_usage.hh>

static LinearisableCar *car;

/// function list.
/**
 * Calling this function presents a list of functions which
 * may be called from the Octave interpreter after the
 * LinearisableCar has been initialised.
 */
const char *usage_OctCar =
"OctCar()\n"
"\n"
"An Octave interface to a linearisable car model:\n"
"Initialises a velocity-based model of a car\n"
"and displays the vehicle parameters.\n"
"\n"
"Related functions:\n"
" OctCar_get_A\n"
" OctCar_get_B\n"
" OctCar_get_position\n"
" OctCar_get_velocity\n"
" OctCar_get_acceleration\n"
" OctCar_get_friction_coefficient\n"
" OctCar_get_wheel_lateral_forces\n"
" OctCar_get_wheel_lateral_speeds\n"
" OctCar_get_wheel_longitudinal_forces\n"
" OctCar_get_wheel_longitudinal_speeds\n"
" OctCar_get_wheel_slip_angles\n"
" OctCar_get_wheel_steering_angles\n"
" OctCar_get_wheel_vertical_forces\n"
" OctCar_integrate_euler_nonlinear\n"
" OctCar_integrate_euler_linear\n"
" OctCar_set_acceleration\n"
" OctCar_set_friction_coefficient\n"
" OctCar_set_velocity\n"
" OctCar_set_wheel_longitudinal_forces\n"
" OctCar_set_wheel_slip_angles\n"
" OctCar_set_wheel_speeds\n"
" OctCar_set_wheel_steering_angle\n"
" OctCar_set_wheel_vertical_forces\n"
" OctCar_write_parameters\n";

/** initialises the LinearisableCar. */
DEFUN_DLD(OctCar, args, ,
          usage_OctCar)
{
  octave_value_list retval;
  if (args.length() != 0) {
    usage(usage_OctCar);
  }
  if (car != NULL) {
    delete(car);
  }
  car = new LinearisableCar();
  car->write_parameters();
  retval(0) = true;
  return retval;
}

DEFUN_DLD(OctCar_get_A, args, ,
          usage_text("get_A"))
{
  octave_value_list retval;
  if (args.length() != 0) {
    usage(usage_text("get_A"));
  }
  double A_array[LinearisableCar::NX*LinearisableCar::NX];
  car->get_A(A_array);
  Matrix m(LinearisableCar::NX,LinearisableCar::NX);
  int counter = 0;
  for (int i = 0; i < LinearisableCar::NX; i++) {
    for (int j = 0; j < LinearisableCar::NX; j++) {
```

```
      m(j,i) = A_array[counter++];
    }
  }
  retval(0) = m;
  return retval;
}

DEFUN_DLD(OctCar_get_B, args, ,
          usage_text("get_B"))
{
  octave_value_list retval;
  if (args.length() != 0) {
    usage(usage_text("get_B"));
  }
  double B_array[LinearisableCar::NX*LinearisableCar::NU];
  car->get_B(B_array);
  Matrix m(LinearisableCar::NX,LinearisableCar::NU);
  int counter = 0;
  for (int i = 0; i < LinearisableCar::NU; i++) {
    for (int j = 0; j < LinearisableCar::NX; j++) {
      m(j,i) = B_array[counter++];
    }
  }
  retval(0) = m;
  return retval;
}

DEFUN_DLD(OctCar_get_position, args, ,
          usage_text("get_position"))
{
  octave_value_list retval;
  if (args.length() != 0) {
    usage(usage_text("get_position"));
  }
  ColumnVector p(3);
  Car::axis position;
  position = car->get_position();
  p(0) = position.X;
  p(1) = position.Y;
  p(2) = position.Psi;
  retval(0) = p;
  return retval;
}

DEFUN_DLD(OctCar_get_velocity, args, ,
          usage_text("get_velocity"))
{
  octave_value_list retval;
  if (args.length() != 0) {
    usage(usage_text("get_velocity"));
  }
  ColumnVector v(3);
  Car::axis velocity;
  velocity = car->get_velocity();
  v(0) = velocity.X;
  v(1) = velocity.Y;
  v(2) = velocity.Psi;
  retval(0) = v;
  return retval;
}

DEFUN_DLD(OctCar_get_acceleration, args, ,
          usage_text("get_acceleration"))
{
  octave_value_list retval;
  if (args.length() != 0) {
    usage(usage_text("get_acceleration"));
  }
  ColumnVector a(3);
  Car::axis acceleration;
  acceleration = car->get_acceleration();
  a(0) = acceleration.X;
  a(1) = acceleration.Y;
  a(2) = acceleration.Psi;
  retval(0) = a;
  return retval;
}
```

```
DEFUN_DLD(OctCar_get_friction_coefficient, args, ,
          usage_text("get_friction_coefficient"))
{
  octave_value_list retval;
  if (args.length() != 0) {
    usage(usage_text("get_friction_coefficient"));
  }
  double mu;
  mu = car->get_friction_coefficient();
  retval(0) = mu;
  return retval;
}

DEFUN_DLD(OctCar_get_wheel_lateral_forces, args, ,
          usage_text("get_wheel_lateral_forces"))
{
  octave_value_list retval;
  if (args.length() != 0) {
    usage(usage_text("get_wheel_lateral_forces"));
  }
  ColumnVector fy(4);
  fy(0) = car->get_wheel_lateral_force(Car::FL);
  fy(1) = car->get_wheel_lateral_force(Car::FR);
  fy(2) = car->get_wheel_lateral_force(Car::RL);
  fy(3) = car->get_wheel_lateral_force(Car::RR);
  retval(0) = fy;
  return retval;
}

DEFUN_DLD(OctCar_get_wheel_lateral_speeds, args, ,
          usage_text("get_wheel_lateral_speeds"))
{
  octave_value_list retval;
  if (args.length() != 0) {
    usage(usage_text("get_wheel_lateral_forces"));
  }
  ColumnVector vy(4);
  vy(0) = car->get_wheel_lateral_speed(Car::FL);
  vy(1) = car->get_wheel_lateral_speed(Car::FR);
  vy(2) = car->get_wheel_lateral_speed(Car::RL);
  vy(3) = car->get_wheel_lateral_speed(Car::RR);
  retval(0) = vy;
  return retval;
}

DEFUN_DLD(OctCar_get_wheel_longitudinal_forces, args, ,
          usage_text("get_wheel_longitudinal_forces"))
{
  octave_value_list retval;
  if (args.length() != 0) {
    usage(usage_text("get_wheel_longitudinal_forces"));
  }
  ColumnVector fx(4);
  fx(0) = car->get_wheel_longitudinal_force(Car::FL);
  fx(1) = car->get_wheel_longitudinal_force(Car::FR);
  fx(2) = car->get_wheel_longitudinal_force(Car::RL);
  fx(3) = car->get_wheel_longitudinal_force(Car::RR);
  retval(0) = fx;
  return retval;
}

DEFUN_DLD(OctCar_get_wheel_longitudinal_speeds, args, ,
          usage_text("get_wheel_longitudinal_speeds"))
{
  octave_value_list retval;
  if (args.length() != 0) {
    usage(usage_text("get_wheel_longitudinal_speeds"));
  }
  ColumnVector vx(4);
  vx(0) = car->get_wheel_longitudinal_speed(Car::FL);
  vx(1) = car->get_wheel_longitudinal_speed(Car::FR);
  vx(2) = car->get_wheel_longitudinal_speed(Car::RL);
  vx(3) = car->get_wheel_longitudinal_speed(Car::RR);
  retval(0) = vx;
  return retval;
}
```

```
DEFUN_DLD(OctCar_get_wheel_slip_angles, args, ,
          usage_text("get_wheel_slip_angles"))
{
  octave_value_list retval;
  if (args.length() != 0) {
    usage(usage_text("get_wheel_slip_angles"));
  }
  ColumnVector alpha(4);
  alpha(0) = car->get_wheel_slip_angle(Car::FL);
  alpha(1) = car->get_wheel_slip_angle(Car::FR);
  alpha(2) = car->get_wheel_slip_angle(Car::RL);
  alpha(3) = car->get_wheel_slip_angle(Car::RR);
  retval(0) = alpha;
  return retval;
}

DEFUN_DLD(OctCar_get_wheel_steering_angles, args, ,
          usage_text("get_wheel_steering_angles"))
{
  octave_value_list retval;
  if (args.length() != 0) {
    usage(usage_text("get_wheel_steering_angles"));
  }
  ColumnVector delta(4);
  delta(0) = car->get_wheel_steering_angle(Car::FL);
  delta(1) = car->get_wheel_steering_angle(Car::FR);
  delta(2) = car->get_wheel_steering_angle(Car::RL);
  delta(3) = car->get_wheel_steering_angle(Car::RR);
  retval(0) = delta;
  return retval;
}

DEFUN_DLD(OctCar_get_wheel_vertical_forces, args, ,
          usage_text("get_wheel_vertical_forces"))
{
  octave_value_list retval;
  if (args.length() != 0) {
    usage(usage_text("get_wheel_vertical_forces"));
  }
  ColumnVector fz(4);
  fz(0) = car->get_wheel_vertical_force(Car::FL);
  fz(1) = car->get_wheel_vertical_force(Car::FR);
  fz(2) = car->get_wheel_vertical_force(Car::RL);
  fz(3) = car->get_wheel_vertical_force(Car::RR);
  retval(0) = fz;
  return retval;
}

DEFUN_DLD(OctCar_integrate_euler_linear, args, ,
          usage_text("integrate_euler_linear"))
{
  octave_value_list retval;
  if (args.length() != 2) {
    usage(usage_text("integrate_euler_linear"));
    error("incorrect number of arguments");
    retval(0) = false;
    return retval;
  }
  double dt = args(0).double_value();
  int n = static_cast<int>(args(1).double_value());
  for (int i = 0; i < n; i++) {
    car->integrate_euler(dt);
  }
  retval(0) = true;
  return retval;
}

DEFUN_DLD(OctCar_integrate_euler_nonlinear, args, ,
          usage_text("integrate_euler_nonlinear"))
{
  octave_value_list retval;
  if (args.length() != 2) {
    usage(usage_text("integrate_euler_nonlinear"));
    error("incorrect number of arguments");
    retval(0) = false;
```

```
    return retval;
  }
  double dt = args(0).double_value();
  int n = static_cast<int>(args(1).double_value());
  for (int i = 0; i < n; i++) {
    car->Car::integrate_euler(dt);
  }
  retval(0) = true;
  return retval;
}

DEFUN_DLD(OctCar_set_accleration, args, ,
          usage_text("set_acceleration"))
{
  octave_value_list retval;
  if (args.length() != 1) {
    usage(usage_text("set_acceleration"));
    error("incorrect number of arguments");
    retval(0) = false;
    return retval;
  }
  ColumnVector a(3, 0.0);
  a = args(0).column_vector_value();
  Car::axis acceleration;
  acceleration.X = a(0);
  acceleration.Y = a(1);
  acceleration.Psi = a(2);
  car->set_acceleration(acceleration);
  retval(0) = true;
  return retval;
}

DEFUN_DLD(OctCar_set_friction_coefficient, args, ,
          usage_text("set_friction_coefficient"))
{
  octave_value_list retval;
  if (args.length() != 1) {
    usage(usage_text("set_friction_coefficient"));
    error("incorrect number of arguments");
    retval(0) = false;
    return retval;
  }
  double mu = args(0).double_value();
  car->set_friction_coefficient(mu);
  retval(0) = true;
  return retval;
}

DEFUN_DLD(OctCar_set_velocity, args, ,
          usage_text("set_velocity"))
{
  octave_value_list retval;
  if (args.length() != 1) {
    usage(usage_text("set_velocity"));
    error("incorrect number of arguments");
    retval(0) = false;
    return retval;
  }
  ColumnVector v(3, 0.0);
  v = args(0).column_vector_value();
  Car::axis velocity;
  velocity.X = v(0);
  velocity.Y = v(1);
  velocity.Psi = v(2);
  car->set_velocity(velocity);
  retval(0) = true;
  return retval;
}


DEFUN_DLD(OctCar_set_wheel_lateral_forces, args, ,
          usage_text("set_wheel_lateral_forces"))
{
  octave_value_list retval;
  if (args.length() != 1) {
    usage(usage_text("set_wheel_lateral_forces"));
    error("incorrect number of arguments");
```

```
    retval(0) = false;
    return retval;
  }
  ColumnVector f(4, 0.0);
  f = args(0).column_vector_value();
  car->set_wheel_lateral_force(Car::FL, f(0));
  car->set_wheel_lateral_force(Car::FR, f(1));
  car->set_wheel_lateral_force(Car::RL, f(2));
  car->set_wheel_lateral_force(Car::RR, f(3));
  retval(0) = true;
  return retval;
}

DEFUN_DLD(OctCar_set_wheel_longitudinal_forces, args, ,
          usage_text("set_wheel_longitudinal_forces"))
{
  octave_value_list retval;
  if (args.length() != 1) {
    usage(usage_text("set_wheel_longitudinal_forces"));
    error("incorrect number of arguments");
    retval(0) = false;
    return retval;
  }
  ColumnVector f(4, 0.0);
  f = args(0).column_vector_value();
  car->set_wheel_longitudinal_force(Car::FL, f(0));
  car->set_wheel_longitudinal_force(Car::FR, f(1));
  car->set_wheel_longitudinal_force(Car::RL, f(2));
  car->set_wheel_longitudinal_force(Car::RR, f(3));
  retval(0) = true;
  return retval;
}

DEFUN_DLD(OctCar_set_wheel_slip_angles, args, ,
          usage_text("set_wheel_slip_angles"))
{
  octave_value_list retval;
  if (args.length() != 1) {
    usage(usage_text("set_wheel_slip_angles"));
    error("incorrect number of arguments");
    retval(0) = false;
    return retval;
  }
  ColumnVector alpha(4, 0.0);
  alpha = args(0).column_vector_value();
  car->set_wheel_slip_angle(Car::FL, alpha(0));
  car->set_wheel_slip_angle(Car::FR, alpha(1));
  car->set_wheel_slip_angle(Car::RL, alpha(2));
  car->set_wheel_slip_angle(Car::RR, alpha(3));
  retval(0) = true;
  return retval;
}

DEFUN_DLD(OctCar_set_wheel_speeds, args, ,
          usage_text("set_wheel_speeds"))
{
  octave_value_list retval;
  if (args.length() != 1) {
    usage(usage_text("set_wheel_speeds"));
    //    error("incorrect number of arguments");
    retval(0) = false;
    return retval;
  }
  ColumnVector vx(4, 0.0);
  vx = args(0).column_vector_value();
  car->set_wheel_speed(Car::FL, vx(0));
  car->set_wheel_speed(Car::FR, vx(1));
  car->set_wheel_speed(Car::RL, vx(2));
  car->set_wheel_speed(Car::RR, vx(3));
  retval(0) = true;
  return retval;
}

DEFUN_DLD(OctCar_set_wheel_steering_angle, args, ,
          usage_text("set_wheel_steering_angle"))
{
  octave_value_list retval;
```

```
  if (args.length() != 1) {
    usage(usage_text("set_wheel_steering_angle"));
    error("incorrect number of arguments");
    retval(0) = false;
    return retval;
  }
  double angle = args(0).double_value();
  car->set_wheel_steering_angle(angle);
  retval(0) = true;
  return retval;
}

DEFUN_DLD(OctCar_set_wheel_vertical_forces, args, ,
          usage_text("set_wheel_vertical_forces"))
{
  octave_value_list retval;
  if (args.length() != 1) {
    usage(usage_text("set_wheel_vertical_forces"));
    error("incorrect number of arguments");
    retval(0) = false;
    return retval;
  }
  ColumnVector f(4, 0.0);
  f = args(0).column_vector_value();
  car->set_wheel_vertical_force(Car::FL, f(0));
  car->set_wheel_vertical_force(Car::FR, f(1));
  car->set_wheel_vertical_force(Car::RL, f(2));
  car->set_wheel_vertical_force(Car::RR, f(3));
  retval(0) = true;
  return retval;
}

DEFUN_DLD(OctCar_write_parameters, args, ,
          usage_text("write_parameters"))
{
  octave_value_list retval;
  if (car == NULL) {
    car = new LinearisableCar();
  }
  car->write_parameters();
  retval(0) = true;
  return retval;
}
```

A.1.7 MexCar.m

```
function retval = MexCar(subfunc, varargin)
  try
    s = ["OctCar_", subfunc];
    f = str2func(s);
    retval = f(all_va_args);
  catch
    subfunc, varargin, s, f
    error('something is broken')
  end_try_catch
endfunction
```

A.2 Trajectory generation code

A.2.1 create_reference_profiles.m

```
%% -*- matlab -*-

function [ref,man] = create_reference_profiles(parameters)

USE_OPTIMISATION = true;

%% operating conditions
mu      = parameters.mu;
spec    = parameters.spec;
Vx      = parameters.Vx;

%% vehicle parameters
v220_par_050710;
lf = fazP_kafi_Fzg_lv;                  % m
lr = fazP_kafi_Fzg_lh;                  % m
```

```
w  = 2*fazP_kafi_Fzg_spb;              % m

%% manoeuvre specification
manoeuvre = define_manoeuvre(spec, w);

X0 = manoeuvre.l0;                     % m
X1 = manoeuvre.l1 + X0;                % m
X2 = manoeuvre.l2 + X1;                % m
X3 = manoeuvre.l3 + X2;                % m
X4 = manoeuvre.l4 + X3;                % m
X5 = manoeuvre.l5 + X4;                % m

disp('[X0,X1,X2,X3,X4,X5]'); [X0,X1,X2,X3,X4,X5]

Y1 = 0.0;                              % m
Y3 = Y1 ...
     - manoeuvre.w1/2 ...
     + manoeuvre.w2 ...
     - manoeuvre.w3/2;                 % m
Y5 = Y3 ...
     + manoeuvre.w3/2 ...
     - manoeuvre.w4 ...
     + manoeuvre.w5/2;                 % m

disp('[Y1,Y3,Y5]'); [Y1,Y3,Y5]

%% constants of nature
g       = 9.81;                        % m/s2

if (~ USE_OPTIMISATION)

  a = mu*g;                            % max acceleration
  r = Vx^2/a                           % circular motion

  % p0  starting position
  % p1  end of first straight
  % p2  first approach to first corner
  % p3  end of first turn
  % p4  midpoint of first 2 circles
  % p5  start of second turn
  % p6  second approach to boundary
  % p7  end of second turn
  % p8  start of third turn
  % p9  third approach to boundary
  % p10 end of third turn
  % p11 midpoint between 3rd and 4th circles
  % p12 start of fourth turn
  % p13 fourth approach to boundary
  % p14 end of fourth turn

  % (Xa,Ya) P1  end of straight, start of first arc
  % (Xb,Yb) P3  end of first arc, start of straight
  % (Xc,Yc) P5  end of straight, start of second arc
  % (Xd,Yd) P7  end of second arc, start of straight
  % (Xe,Ye) P8  end of straight, start of third arc
  % (Xf,Yf) P10 end of third arc, start of straight
  % (Xg,Yg) P12 end of straight, start of fourth arc
  % (Xh,Yh) P14 end of fourth arc, start of straight

  % Unknown: Xp1,Xp3,Xp4,Xp5,Xp7,Xp8,Xp10,Xp11,Xp12,Xp14
  Xp0  = 0;
  Xp2  = X1 + w/2;
  Xp6  = X2 - w/2;
  Xp9  = X3 + w/2;
  Xp13 = X4 - w/2;
  Xp15 = X5;

  % Unknown: Yp3,Yp4,Yp5,Yp10,Yp11,Yp12
  Yp0  = Y1;
  Yp1  = Y1;
  Yp2  = Y1 + manoeuvre.w1/2 - w/2;
  Yp6  = Y3 - manoeuvre.w3/2 + w/2;
  Yp7  = Y3;
  Yp8  = Y3;
  Yp9  = Y3 - manoeuvre.w3/2 + w/2;
```

```
Yp13 = Y5 + manoeuvre.w5/2 - w/2;
Yp14 = Y5;
Yp15 = Y5;

theta1 = acos(1-(Yp2-Yp1)/r);
theta2 = acos(1-(Yp7-Yp6)/r);
theta3 = acos(1-(Yp8-Yp9)/r);
theta4 = acos(1-(Yp13-Yp14)/r);

Xp1  = Xp2  - r*sin(theta1);
Xp7  = Xp6  + r*sin(theta2);
Xp8  = Xp9  - r*sin(theta3);
Xp14 = Xp13 + r*sin(theta4);

% circle centres
Xo1 = Xp1;    Yo1 = Yp1  + r;
Xo2 = Xp7;    Yo2 = Yp7  - r;
Xo3 = Xp8;    Yo3 = Yp8  - r;
Xo4 = Xp14;   Yo4 = Yp14 + r;

Xp4  = (Xo1+Xo2)/2; Yp4  = (Yo1+Yo2)/2;
Xp11 = (Xo3+Xo4)/2; Yp11 = (Yo3+Yo4)/2;

% still need (Xp3,Yp3) (Xp5,Yp5)(Xp10,Yp10) (Xp12,Yp12)
phi12 = asin(2*r/sqrt((Xo2-Xo1)^2+(Yo2-Yo1)^2))
phi34 = asin(2*r/sqrt((Xo4-Xo3)^2+(Yo4-Yo3)^2))
sigma12 = atan2((Yo2-Yo1),(Xo2-Xo1))
sigma34 = atan2((Yo4-Yo3),(Xo4-Xo3))

Xp3  = Xo1 + r*sin(phi12+sigma12); Yp3  = Yo1 - r*cos(phi12+sigma12);
Xp5  = Xo2 - r*sin(phi12+sigma12); Yp5  = Yo2 + r*cos(phi12+sigma12);
Xp10 = Xo3 + r*sin(phi34-sigma34); Yp10 = Yo3 + r*cos(phi34-sigma34);
Xp12 = Xo4 - r*sin(phi34-sigma34); Yp12 = Yo4 - r*cos(phi34-sigma34);

[xcirc,ycirc] = pol2cart(linspace(0,2*pi,100),r*ones(1,100));
xcirc1 = xcirc + Xo1; ycirc1 = ycirc + Yo1;
xcirc2 = xcirc + Xo2; ycirc2 = ycirc + Yo2;
xcirc3 = xcirc + Xo3; ycirc3 = ycirc + Yo3;
xcirc4 = xcirc + Xo4; ycirc4 = ycirc + Yo4;

clf; figure(4);
plot(...
    [Xp0;Xp1;Xp3;Xp4;Xp5;Xp7;Xp8;Xp10;Xp11;Xp12;Xp14;Xp15], ...
    [Yp0;Yp1;Yp3;Yp4;Yp5;Yp7;Yp8;Yp10;Yp11;Yp12;Yp14;Yp15], ...
    'sr', ...
    xcirc1, ycirc1, '-k', ...
    xcirc2, ycirc2, '-k', ...
    xcirc3, ycirc3, '-k', ...
    xcirc4, ycirc4, '-k'  ...
    );

text(Xo1,Yo1,'o1');
text(Xo2,Yo2,'o2');
text(Xo3,Yo3,'o3');
text(Xo4,Yo4,'o4');

title([spec, ' at ', num2str(Vx*3.6), ' km/hr']);
xlabel('Position XE [m]'); ylabel('Position YE [m]');
axis('ij');
ymin = floor(-1+min(min(min(Yo1,Yo2),Yo3),Yo4)/2)*2;
ymax = ceil(+1+max(max(max(Yo1,Yo2),Yo3),Yo4)/2)*2;
axis([X0,X5,ymin,ymax])
view([-90,90]);
axis('equal')
print -depsc 'waypoints.eps';
saveas(gcf, 'waypoints.fig');
figure(1);

mp = +tan(sigma12+phi12)
mq = +tan(sigma34-phi34)

Xa = Xp1;  Ya = Yp1;
Xb = Xp3;  Yb = Yp3;
Xc = Xp5;  Yc = Yp5;
Xd = Xp7;  Yd = Yp7;
Xe = Xp8;  Ye = Yp8;
Xf = Xp10; Yf = Yp10;
Xg = Xp12; Yg = Yp12;
```

```
Xh = Xp14; Yh = Yp14;

Xc1 = Xo1; Yc1 = Yo1;
Xc2 = Xo2; Yc2 = Yo2;
Xc3 = Xo3; Yc3 = Yo3;
Xc4 = Xo4; Yc4 = Yo4;

disp('Xa,Xb,Xc,Xd,Xe,Xf,Xg,Xh');
[Xa,Xb,Xc,Xd,Xe,Xf,Xg,Xh]

disp('Xc1,Yc1,Xc2,Yc2,Xc3,Yc3,Xc4,Yc4');
[Xc1,Yc1,Xc2,Yc2,Xc3,Yc3,Xc4,Yc4]

spacing       = (X2-X1) / 100;
X             = [0:spacing:X5];

i00 = find((X<X0));
i11 = find((X>=X0)&(X<=Xa));
i12 = find((X>Xa)&(X<=Xb));
i22 = find((X>Xb)&(X<=Xc));
i23 = find((X>Xc)&(X<=Xd));
if (abs(Y3 - Y5) < eps)
  i33 = find(X>Xd);
  i34 = [];
  i44 = [];
  i45 = [];
  i55 = [];
else
  i33 = find((X>Xd)&(X<=Xe));
  i34 = find((X>Xe)&(X<=Xf));
  i44 = find((X>Xf)&(X<=Xg));
  i45 = find((X>Xg)&(X<=Xh));
  i55 = find((X>Xh));
end

Y(i00) = Y1;
Y(i11) = Y1;
Y(i12) = Yc1 - sqrt(r^2 - (X(i12)-Xc1).^2);
Y(i22) = Yb + mp*(X(i22)-Xb);
Y(i23) = Yc2 + sqrt(r^2 - (X(i23)-Xc2).^2);
Y(i33) = Y3;
Y(i34) = Yc3 + sqrt(r^2 - (X(i34)-Xc3).^2);
Y(i44) = Yf + mq*(X(i44)-Xf);
Y(i45) = Yc4 - sqrt(r^2 - (X(i45)-Xc4).^2);
Y(i55) = Y5;

dY_dX(i00) = 0;
dY_dX(i11) = 0;
dY_dX(i12) = +(X(i12)-Xc1)./sqrt(r^2-(X(i12)-Xc1).^2);
dY_dX(i22) = mp;
dY_dX(i23) = -(X(i23)-Xc2)./sqrt(r^2-(X(i23)-Xc2).^2);
dY_dX(i33) = 0;
dY_dX(i34) = -(X(i34)-Xc3)./sqrt(r^2-(X(i34)-Xc3).^2);
dY_dX(i44) = mq;
dY_dX(i45) = +(X(i45)-Xc4)./sqrt(r^2-(X(i45)-Xc4).^2);
dY_dX(i55) = 0;

d2Y_dX2(i00) = 0;
d2Y_dX2(i11) = 0;
d2Y_dX2(i12) = +r^2./((r^2-(X(i12)-Xc1).^2).^1.5);
d2Y_dX2(i22) = 0;
d2Y_dX2(i23) = -r^2./((r^2-(X(i23)-Xc2).^2).^1.5);
d2Y_dX2(i33) = 0;
d2Y_dX2(i34) = -r^2./((r^2-(X(i34)-Xc3).^2).^1.5);
d2Y_dX2(i44) = 0;
d2Y_dX2(i45) = +r^2./((r^2-(X(i45)-Xc4).^2).^1.5);
d2Y_dX2(i55) = 0;

Psi = atan(dY_dX);
dPsi_dX = d2Y_dX2./(dY_dX.^2+1);
Psidot = dPsi_dX*Vx;

RoT(i00) = 0;
RoT(i11) = 0;
RoT(i12) = +r;
RoT(i22) = 0;
RoT(i23) = -r;
```

```
  RoT(i33) = 0;
  RoT(i34) = -r;
  RoT(i44) = 0;
  RoT(i45) = +r;
  RoT(i55) = 0;

  delta = zeros(size(X));
  nz = find(RoT);
  delta(nz) = atan((lf+lr)./RoT(nz));

  fref  = zeros(size(X));

  ref.Xe        = X;
  ref.Ye        = Y;
  ref.Psi       = Psi;

  ref.Xdot      = Vx*ones(size(X));
  ref.Ydot      = 0*ones(size(X));
  ref.Psidot    = Psidot;

  ref.Xddot     = 0*ones(size(X));
  ref.Yddot     = 0*ones(size(X));
  ref.Psiddot   = 0*ones(size(X));

  ref.delta     = delta;

else                                        % USE_OPTIMISATION

  figure(1); clf

  if isempty(strfind(path,'cvx'))
    addpath('/usr/local/cvx/cvx');
  end
  cvx_clear;

  %%% manoeuvre specification
  dx    = 1;
  xinit = [X0:dx:X5+50]';
  limit = cones(spec, xinit, X0, Y1, w); % calls define_manoeuvre
  ylb   = limit.Y_lhs;
  yub   = limit.Y_rhs;
  L     = length(xinit)-1;

  % clearance
  ylb   = ylb + 0*0.15;
  yub   = yub - 0*0.15;

  % parameters
  a     = mu*g;
  m     = 2364; %2360;
  Izz   = 4488; %4700;

  % terminal conditions
  rL    = 0;
  qL    = 0;
  yL    = (ylb(end)+yub(end))/2;

  % I. First optimisation
  disp('-- Starting run 1 --')

  cvx_begin

  variables x_e(L+1) y_e(L+1) q(L+1)         % I. Earth positions
  variables xedot(L+1) yedot(L+1)            % I. Earth velocities
  variables xdot(L+1) ydot(L+1) qdot(L+1)    % I. Body velocities
  variables xddot(L+1) yddot(L+1) qddot(L+1) % I. Body accelerations
  variables c1(L+1) c2(L+1) c3(L+1) c4(L+1); % I. vehicle geometry

  minimize(norm(qddot));                   % I. Objective
  subject to

  % I. Assumptions

  cosq  = 1;                           % I. small angle
  sinq  = q;

  xddot == 0;                          % I. constant speed
  yddot == 0;                           % I. no lateral slip
```

```
dt    = dx/Vx;

% l. Constraints

x_e(1)        == 0;                    % l. initial conditions
y_e(1)        == 0;
q(1)          == 0;
xdot(1)       == Vx;
ydot(1)       == 0;
qdot(1)       == 0;

y_e(L+1)      == yL;          % l. terminal conditions
q(L+1)        == qL;
qdot(L+1)     == rL;

x_e(2:L+1)    == x_e(1:L) + dt*xedot(1:L); % l. quadrature
y_e(2:L+1)    == y_e(1:L) + dt*yedot(1:L);
q(2:L+1)      == q(1:L) + dt*qdot(1:L);
xdot(2:L+1)   == xdot(1:L) + dt*xddot(1:L);
ydot(2:L+1)   == ydot(1:L) + dt*yddot(1:L);
qdot(2:L+1)   == qdot(1:L) + dt*qddot(1:L);

xedot         == Vx.*1;
yedot         == Vx.*q;

qddot.^2      <= (lf*m/Izz)^2*(mu*g)^2; % l. accel limit

c1    == y_e + lf*sinq - (w/2)*cosq;  % l. geometry
c2    == y_e + lf*sinq + (w/2)*cosq;
c3    == y_e - lr*sinq - (w/2)*cosq;
c4    == y_e - lr*sinq + (w/2)*cosq;

c1    <= yub;                         % l. manoeuvre boundary
c2    <= yub;
c3    <= yub;
c4    <= yub;
c1    >= ylb;
c2    >= ylb;
c3    >= ylb;
c4    >= ylb;

cvx_end;

x     = x_e;
y     = y_e;

cosq  = cos(q(1:L));
sinq  = sin(q(1:L));

xe    = dt*cumtrapz(+Vx.*cos(q)-0.*sin(q));
ye    = dt*cumtrapz(+Vx.*sin(q)+0.*cos(q));
qe    = interp1(x,q,xe);

cosqI = cos(qe);
sinqI = sin(qe);
xdotI = xdot;                         % Vx

corners = [+lf, +lf, -lr, -lr;
           -w/2, +w/2, +w/2, -w/2];
for i = 1:length(x)
  T{i} = [+cos(q(i)), -sin(q(i));
          +sin(q(i)), +cos(q(i))];
  c{i} = T{i}*corners;
  cxe(i,:) = +xe(i) + c{i}(1,:);
  cye(i,:) = +ye(i) + c{i}(2,:);
end
plot (x, y, ..;
      xe, ye, 'r', ...
      cxe, cye, ':b', ...
      xinit, [ylb+w/2,yub-w/2], '--k', ...
      xinit, [ylb,yub], ':k');
legend({'CGv ','CGe ', 'Tyre '});
legend boxoff;
pause(0.01);
plot(x, y, '-.b', xe, ye, '-k', xinit, ylb, '--b', cxe, cye, ':k', xinit, yub,
   '--b');
legend({'(XE,YE)', '(XE I,YE I) ', 'Cones ', 'Wheels '});
legend boxoff;
```

```
xlabel('Position XE [m]');
ylabel('Position YE [m]');
axis([0, 80, -1.5, 5.5]);
axis('ij');
view([-90,90]);
print -depsc 'optimisation1.eps'
saveas(gcf, 'optimisation1.fig');

% II. Second optimisation
disp('-- Starting run 2 --')

xinit = xe;
limit = cones(spec, xinit, X0 ,Y1, w);
ylb   = limit.Y_lhs;
yub   = limit.Y_rhs;

cvx_clear
cvx_begin

variables x_e(L+1) y_e(L+1) q(L+1);    % II. Earth positions
variables xedot(L+1) yedot(L+1);          % II. Earth velocities
variables xdot(L+1) ydot(L+1) qdot(L+1);     % II. Body velocities
variables xddot(L+1) yddot(L+1) qddot(L+1);  % II. Body accelerations
variables c1(L+1) c2(L+1) c3(L+1) c4(L+1);

minimize(norm(qddot))
subject to

% II. Assumptions

cosq  = cosqI;
sinq  = sinqI;

yddot == 0;
dt    = dx/Vx;

% II. Constraints

x_e(1)        == 0;                  % II. initial conditions
y_e(1)        == 0;
q(1)          == 0;
xdot(1)       == Vx;
ydot(1)       == 0;
qdot(1)       == 0;

y_e(L+1)      == yL;                 % II. terminal conditions
q(L+1)        == qL;
qdot(L+1)     == rL;

x_e(2:L+1)    == x_e(1:L) + dt*xedot(1:L); % II. quadrature
y_e(2:L+1)    == y_e(1:L) + dt*yedot(1:L);
q(2:L+1)      == q(1:L) + dt*qdot(1:L);
xdot(2:L+1)   == xdot(1:L) + dt*xddot(1:L);
ydot(2:L+1)   == ydot(1:L) + dt*yddot(1:L);
qdot(2:L+1)   == qdot(1:L) + dt*qddot(1:L);

xedot         == xdot*1;
yedot         == Vx.*q;

qddot.^2      <= (lf*m/Izz)^2*((mu*g)^2-xddot.^2); % II. Accel
xdot          >= 0;                  % no reversing
xddot         <= 0;

c1    == y_e + lf*sinq - (w/2)*cosq;
c2    == y_e + lf*sinq + (w/2)*cosq;
c3    == y_e - lr*sinq - (w/2)*cosq;
c4    == y_e - lr*sinq + (w/2)*cosq;

c1    <= yub;
c2    <= yub;
c3    <= yub;
c4    <= yub;
c1    >= ylb;
c2    >= ylb;
c3    >= ylb;
c4    >= ylb;

cvx_end;
```

```
x     = x_e;
y     = y_e;

xe    = dt*cumtrapz(+xdot.*cos(q)-ydot.*sin(q));
ye    = dt*cumtrapz(+xdot.*sin(q)+ydot.*cos(q));
qe    = interp1(x,q,xe);

cosqII        = cos(qe);
sinqII        = sin(qe);
xdotII        = xdot;

corners = [+lf, +lf, -lr, -lr;
           -w/2, +w/2, +w/2, -w/2];
for i = 1:length(x)
  T{i} = [+cos(q(i)), -sin(q(i));
          +sin(q(i)), +cos(q(i))];
  c{i} = T{i}*corners;
  cxe(i,:) = +xe(i) + c{i}(1,:);
  cye(i,:) = +ye(i) + c{i}(2,:);
end
plot (x, y, ..;
      xe, ye, 'r', ...
      cxe, cye, ':b', ...
      xinit, [ylb+w/2,yub-w/2], '--k', ...
      xinit, [ylb,yub], ':k');
legend({'CGv ','CGe ', 'Tyre '});
legend boxoff;
pause(0.01);
plot(x, y, '-.b', xe, ye, '-k', xinit, ylb, '--b', cxe, cye, ':k', ...
     xinit, yub, '--b');
legend({'(XE,YE) ', '(XE II,YE II) ', 'Cones ', 'Wheels '});
legend boxoff;
xlabel('Position XE [m]');
ylabel('Position YE [m]');
axis([0, 80, -1.5, 5.5]);
axis('ij');
view([-90,90]);
print -depsc 'optimisation2.eps'
saveas(gcf, 'optimisation2.fig');

% III. Third optimisation
disp('-- Starting run 3 --')

xinit = xe;
limit = cones(spec, xinit, X0, Y1, w);
ylb   = limit.Y_lhs;
yub   = limit.Y_rhs;

cvx_clear
cvx_begin

variables x_e(L+1) y_e(L+1) q(L+1)    % III. Earth positions
variables xedot(L+1) yedot(L+1)       % III. Earth velocities
variables xdot(L+1) ydot(L+1) qdot(L+1) % III. Body velocities
variables xddot(L+1) yddot(L+1) qddot(L+1) % III. Body accel
variables c1(L+1) c2(L+1) c3(L+1) c4(L+1);

minimize(norm(qddot))
subject to

% III. Assumptions

cosq  = cosqII;
sinq  = sinqII;
yddot == 0;

% III. Constraints

x_e(1)        == 0;                   % III. initial conditions
y_e(1)        == 0;
q(1)          == 0;
xdot(1)       == Vx;
ydot(1)       == 0;
qdot(1)       == 0;

y_e(L+1)      == yL;                  % III. terminal conditions
q(L+1)        == qL;
```

```
qdot(L+1)     == rL;

x_e(2:L+1)    == x_e(1:L) + dt*xedot(1:L); % III. quadrature
y_e(2:L+1)    == y_e(1:L) + dt*yedot(1:L);
q(2:L+1)      == q(1:L) + dt*qdot(1:L);
xdot(2:L+1)   == xdot(1:L) + dt*xddot(1:L);
ydot(2:L+1)   == ydot(1:L) + dt*yddot(1:L);
qdot(2:L+1)   == qdot(1:L) + dt*qdddot(1:L);

xedot         == xdot.*cosq;
yedot         == xdotII.*q;

qddot.^2      <= (lf*m/Izz)^2*((mu*g)^2-xddot.^2);
xdot          >= 0;                  % no reversing
xddot         <= 0;

c1    == y_e + lf*sinq - (w/2)*cosq;
c2    == y_e + lf*sinq + (w/2)*cosq;
c3    == y_e - lr*sinq - (w/2)*cosq;
c4    == y_e - lr*sinq + (w/2)*cosq;

c1    <= yub;
c2    <= yub;
c3    <= yub;
c4    <= yub;
c1    >= ylb;
c2    >= ylb;
c3    >= ylb;
c4    >= ylb;

cvx_end;

x     = x_e;
y     = y_e;

xe    = dt*cumtrapz(+xdot.*cos(q)-ydot.*sin(q));
ye    = dt*cumtrapz(+xdot.*sin(q)+ydot.*cos(q));
qe    = interp1(x,q,xe);

corners = [+lf, +lf, -lr, -lr;
           -w/2, +w/2, +w/2, -w/2];
for i = 1:length(x)
  T{i} = [+cos(q(i)), -sin(q(i));
          +sin(q(i)), +cos(q(i))];
  c{i} = T{i}*corners;
  cxe(i,:) = +xe(i) + c{i}(1,:);
  cye(i,:) = +ye(i) + c{i}(2,:);
end
plot (x, y, ..;
      xe, ye, 'r', ...
      cxe, cye, ':b', ...
      xinit, [ylb+w/2,yub-w/2], '--k', ...
      xinit, [ylb,yub], ':k');
legend({'CGv ','CGe ', 'Tyre '});
legend boxoff;
pause(0.01);
plot(x, y, '-.b', xe, ye, '-k', xinit, ylb, '--b', cxe, cye, ':k', xinit, yub,
    '--b');
legend({'(XE,YE) ', '(XE III,YE III) ', 'Cones ', 'Wheels '});
legend boxoff;
xlabel('Position XE [m]');
ylabel('Position YE [m]');
axis([0, 80, -1.5, 5.5]);
axis('ij');
view([-90,90]);
print -depsc 'optimisation3.eps'
saveas(gcf, 'optimisation3.fig');

% assign reference and plotting parameters
if max(isnan(x))
  return
end

ref.Xe        = xe';
ref.Ye        = ye';
ref.Psi       = qe';

ref.Xdot      = xdot';
```

```
  ref.Ydot      = ydot';
  ref.Psidot    = qdot';

  ref.Xddot     = xddot';
  ref.Yddot     = yddot';
  ref.Psiddot   = qddot';

  ref.delta     = atan2((lf+lr)*qdot,xdot)';
end

HAVE_TRAJECTORY=1

man.lf  = lf;
man.lr  = lr;
man.w   = w;
man.X0  = X0;
man.X1  = X1;
man.X2  = X2;
man.X3  = X3;
man.X4  = X4;
man.X5  = X5;
man.Y1  = Y1;
man.Y3  = Y3;
man.Y5  = Y5;
```

A.2.2 define_manoeuvre.m

```
%% -*-matlab-*-

function [manoeuvre] = define_manoeuvre(spec, vehicle_width)

switch (spec)
 case 'ISO 3888-1'

  lane_offset = 3.5;

  manoeuvre.w1 = 1.1*vehicle_width + 0.25;
  manoeuvre.w3 = 1.2*vehicle_width + 0.25;
  manoeuvre.w5 = 1.3*vehicle_width + 0.35;

  manoeuvre.w2 = manoeuvre.w3 + lane_offset;
  manoeuvre.w4 = manoeuvre.w2;

  manoeuvre.l0 = 0*100;
  manoeuvre.l1 = 15;
  manoeuvre.l2 = 30;
  manoeuvre.l3 = 25;
  manoeuvre.l4 = 25;
  manoeuvre.l5 = 15 + 15;

 case 'ISO 3888-2'

  lane_offset = 1.0;

  manoeuvre.w1 = 1.1*vehicle_width + 0.25;
  manoeuvre.w3 = 1.0*vehicle_width + 1.00;
  manoeuvre.w5 = 1.3*vehicle_width + 0.25;
  manoeuvre.w5 = max(manoeuvre.w5,3);

  manoeuvre.w2 = manoeuvre.w1 + manoeuvre.w3 + lane_offset;
  manoeuvre.w4 = manoeuvre.w2;

  manoeuvre.l0 = 0*100.0;
  manoeuvre.l1 = 12.0;
  manoeuvre.l2 = 13.5;
  manoeuvre.l3 = 11.0;
  manoeuvre.l4 = 12.5;
  manoeuvre.l5 = 12.0;

 case 'single part 1'

  lane_offset = 3.5;

  manoeuvre.w1 = 1.1*vehicle_width + 0.25;
  manoeuvre.w3 = 1.2*vehicle_width + 0.25;

  manoeuvre.w2 = manoeuvre.w3 + lane_offset;
  manoeuvre.w4 = manoeuvre.w3;
  manoeuvre.w5 = manoeuvre.w3;
```

```
  manoeuvre.l0 = 0*100.0;
  manoeuvre.l1 = 15.0;
  manoeuvre.l2 = 30.0;
  manoeuvre.l3 = 25.0;
  manoeuvre.l4 = 25.5;
  manoeuvre.l5 = 15 + 15;

 case 'single part 2'

  lane_offset = 1.0;

  manoeuvre.w1 = 1.1*vehicle_width + 0.25;
  manoeuvre.w3 = 1.0*vehicle_width + 1.00;

  manoeuvre.w2 = manoeuvre.w1 + manoeuvre.w3 + lane_offset;
  manoeuvre.w4 = manoeuvre.w3;
  manoeuvre.w5 = manoeuvre.w3;

  manoeuvre.l0 = 0*100.0;
  manoeuvre.l1 = 12.0;
  manoeuvre.l2 = 13.5;
  manoeuvre.l3 = 11.0;
  manoeuvre.l4 = 12.5;
  manoeuvre.l5 = 12.0;

 case 'gentle lane change'

  lane_offset = 3.5;

  manoeuvre.w1 = 1.1*vehicle_width + 0.25;
  manoeuvre.w3 = 1.2*vehicle_width + 0.25;

  manoeuvre.w1 = manoeuvre.w1 * 1.50;
  manoeuvre.w3 = manoeuvre.w3 * 1.50;

  manoeuvre.w2 = manoeuvre.w3 + lane_offset;
  manoeuvre.w4 = manoeuvre.w3;
  manoeuvre.w5 = manoeuvre.w3;

  manoeuvre.l0 = 0*100.0;
  manoeuvre.l1 = 15.0;
  manoeuvre.l2 = 30.0 * 1.5;
  manoeuvre.l3 = 25.0;
  manoeuvre.l4 = 25.5;
  manoeuvre.l5 = 15 + 15;

 otherwise

  error('define_manoeuvre: spec is invalid');
end
```

A.2.3 cones.m

```
function limit = cones(spec, x, x0, y0, w)

manoeuvre = define_manoeuvre(spec, w);

x1 = x0 + manoeuvre.l1;
x2 = x1 + manoeuvre.l2;
x3 = x2 + manoeuvre.l3;
x4 = x3 + manoeuvre.l4;
x5 = x4 + manoeuvre.l5;

yl1 = y0-manoeuvre.w1/2;
yr1 = y0+manoeuvre.w1/2;

yl2 = yl1;
yr2 = yl1+manoeuvre.w2;

yr3 = yr2;
yl3 = yr3-manoeuvre.w3;

yr4 = yr3;
yl4 = yr4-manoeuvre.w4;

yl5 = yl4;
yr5 = yl5+manoeuvre.w5;

if (nargin < 1)
```

```
  limit.X     = [ x0; x1; x1; x2; x2; x3; x3; x4; x4; x5];
  limit.Y_lhs = [yl1;yl1;yl2;yl2;yl3;yl3;yl4;yl4;yl5;yl5];
  limit.Y_rhs = [yr1;yr1;yr2;yr2;yr3;yr3;yr4;yr4;yr5;yr5];

  plot(limit.X, [limit.Y_lhs,limit.Y_rhs])

else

  if ((size(x,1) == 1) & (size(x,2) > 1))
    x = x'
  end

  limit.X     = x;

  limit.Y_lhs(:,1) ...
      =             (x <= x1)  .* yl1 ...
      + ((x > x1) & (x <= x2)) .* yl2 ...
      + ((x > x2) & (x <= x3)) .* yl3 ...
      + ((x > x3) & (x <= x4)) .* yl4 ...
      + ((x > x4) & (x <= x5)) .* yl5 ...
      + (x > x5) .* yl5;

  limit.Y_rhs(:,1) ...
      =             (x <= x1)  .* yr1 ...
      + ((x > x1) & (x <= x2)) .* yr2 ...
      + ((x > x2) & (x <= x3)) .* yr3 ...
      + ((x > x3) & (x <= x4)) .* yr4 ...
      + ((x > x4) & (x <= x5)) .* yr5 ...
      +  (x  > x5) .* yr5;
end
```

Bibliography

AAC (2001), *American Control Conference*, Institute of Electrical and Electronics Engineers Inc., Arlington, VA, USA.

AAC (2002), *American Control Conference*, Institute of Electrical and Electronics Engineers Inc., Anchorage, AK, USA.

Aburaya, T., Kawanishi, M., Kondo, H. & Hamada, T. (1990), Development of an electronic control system for active suspension, *in* 'Proceedings of the IEEE Conference on Decision and Control', Vol. 4, IEEE, Institute of Electrical and Electronics Engineers Inc., Honolulu, HI, USA, pp. 2220–2225.

Akar, M., Kalkkuhl, J. C. & Suissa, A. (2007), Vertical dynamics emulation using a vehicle equipped with active suspension, in *2007 IEEE Intelligent Vehicles Symposium* IEE (2007), pp. 866–871.

Ammon, D., Gipser, M., Rauh, J. & Wimmer, J. (1997), 'High performance system dynamics simulation of the entire system tire-suspension-steering-vehicle', *Vehicle System Dynamics* **27**(5–6), 435–455.

Andersson, M., Bruzelius, F., Casselgren, J., Gäfvert, M., Hjort, M., Hultén, J., Håbring, F., Klomp, M., Olsson, G., Sjödahl, M., Svendenius, J., Woxneryd, S. & Wälivaara, B. (2007), Road friction estimation, Technical Report 2004:17750, Intelligent Vehicle Safety Systems (IVSS).

Åström, K. J. & Wittenmark, B. (1997), *Computer controlled systems: theory and design*, Information and system science, third edn, Prentice Hall.

Austin, L. & Morrey, D. (2000), 'Recent advances in antilock braking systems and traction control systems', *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering* **214**(6), 625–638.

BBC Radio 5 Live (2007*a*), 'Drive', Radio broadcast.

BBC Radio 5 Live (2007*b*), 'Weekend News', Radio broadcast.

BBC Radio WM (West Midlands) (2008), 'Danny Kelly', Radio broadcast.

BBC World Service (2008), 'The World Today', Radio broadcast.

Betts, J. T. (1998), 'Survey of numerical methods for trajectory optimization', *Journal of Guidance, Control, and Dynamics* **21**(2), 193–207. Trajectory optimization problem;.

Betts, J. T. (2001), *Practical Methods for Optimal Control using Nonlinear Programming*, SIAM, Philadelphia.

Bevan, G., Gollee, H. & O'Reilly, J. (2007*a*), 'Automatic lateral emergency collision avoidance for a passenger car', *International Journal of Control* **80**(11), 1751–1762.

Bevan, G. P., Gollee, H. & O'Reilly, J. (2007*b*), Trajectory generation for road vehicle obstacle avoidance using convex optimisation, Submitted to Vehicle Systems Dynamics, June 2007.

Bevan, G. P., O'Neill, S. J., Gollee, H. & O'Reilly, J. (2007), Performance comparison of collision avoidance controller designs, in *2007 IEEE Intelligent Vehicles Symposium* IEE (2007), pp. 468–473.

Blank, M. & Margolis, D. L. (2000), 'Minimizing the path radius of curvature for collision avoidance', *Vehicle System Dynamics* **33**(3), 183–201.

Bosch, R. (2000), *Automotive handbook (Bosch)*, fifth edn, Bentley Publishers.

Boyd, S. & Vandenberghe, L. (2004), *Convex Optimization*, Cambridge University Press.

Brenan, K. E., Campbell, S. L. & Petzold, L. R. (1996), *Numerical solution of initial-value problems in differential-algebraic equations*, Classics in applied mathematics, SIAM, Philadelphia.

Burgio, G. & Zegelaar, P. (2006), 'Integrated vehicle control using steering and brakes', *International Journal of Control* **79**(5), 534–541.

Canudas de Wit, C. & Tsiotras, P. (1999), Dynamic tire friction models for vehicle traction control, *in* 'Proceedings of the IEEE Conference on Decision and Control', Vol. 4, IEEE, Institute of Electrical and Electronics Engineers Inc., Phoenix, AZ, USA, pp. 3746–3751.

Canudas de Wit, C., Olsson, H., Åström, K. & Lischinsky, P. (1995), 'A new model for control of systems with friction', *IEEE Transactions on Automatic Control* **40**(3), 419–425.

Chakravarthy, A. & Ghose, D. (1998), 'Obstacle avoidance in a dynamic environment: A collision cone approach', *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans.* **28**(5), 562–574.

Chamaillard, Y., Gissinger, G. L., Perronne, J. M. & Renner, M. (1994), An original braking controller with torque sensor, in *Proceedings of the 1994 IEEE Conference on Control Applications. Part 1 (of 3)* IEE (1994), pp. 619–625.

Choi, S.-B., Bang, J.-H., Cho, M.-S. & Lee, Y.-S. (2002), 'Sliding mode control for anti-lock brake system of passenger vehicles featuring electrorheological valves', *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering* **216**(11), 897–908.

Connolly, C. (2007), 'Collision avoidance technology: from parking sensors to unmanned aircraft', *Sensor Review* **27**(3), 182–188.

CVX (2005), 'A Matlab package for disciplined convex programming'. `http://www.stanford.edu/~boyd/cvx/`.

da Vinci, L. (c. 1519 CE), *Codex Atlanticus*, Digital Library, Biblioteca Ambrosiana. .

Daiß, A. & Kiencke, U. (1995), Estimation of vehicle speed fuzzy-estimation in comparison with kalman-filtering, *in* 'Proceedings of the IEEE Conference on Control Applications', IEEE, Institute of Electrical and Electronics Engineers Inc., Albany, NY, USA, pp. 281–284.

DARPA (2007), '3^{rd} Grand Challenge: Urban Challenge', `http://www.darpa.mil/grandchallenge`.

Dodd, M. (2007), 'Personal correspondence'.

Dodd, M. & Gothié, M. (2005), Vehicle, road, tyre and electronic control system interaction - prediction and validation of handling behaviour, Stuttgart. http://www.vertec.hut.fi deliverable P03.

Donald, H. (2001), *Controller style guidelines for production intent using MATLAB, Simulink and Stateflow Version 1.00*, MathWorks Automotive Advisory Board (MAAB). http://www.mathworks.com/industries/auto/maab.html.

Dubins, L. E. (1957), 'On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents', *American Journal of Mathematics* **79**(3), 497–516.

Dudgeon, G. J. W. & Gribble, J. J. (1998), 'Helicopter translational rate command using individual channel analysis and design', *Control Engineering Practice* **6**(1), 15–23.

Durali, M., Javid, G. A. & Kasaiezadeh, A. (2006), Collision avoidance maneuver for an autonomous vehicle, *in* '9^th IEEE International Workshop on Advanced Motion Control', IEEE, Institute of Electrical and Electronics Engineers Inc., Istanbul, Turkey, pp. 249–254.

Eaton, J. W. (2002), *GNU Octave manual*, Network Theory Limited. http://www.octave.org.

Eidehall, A., Pohl, J., Gustafsson, F. & Ekmark, J. (2007), 'Towards autonomous collision avoidance by steering', *IEEE Transactions on Intelligent Transportation Systems* **8**(1), 84–94.

Emig, R., Goebels, H. & Schramm, H. J. (1990), Antilock braking systems (ABS) for commercial vehicles - status 1990 and future prospects, *in* 'Proceedings of the International Congress on Transportation Electronics', number P-233, IEEE Vehicular Technology Soc., Society of Automotive Engineers, Dearborn, MI, USA, pp. 515–523.

Eren, H. & Göktan, A. G. (2001), 'External torque application on antilock brake systems', *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering* **215**(7), 789–794.

Fraichard, T. (1991), Smooth trajectory planning for a car in a structured world, *in* 'Proceedings - 1991 IEEE International Conference on Robotics and Automation', IEEE, Institute of Electrical and Electronics Engineers Inc., Sacramento, CA, USA, pp. 318–323.

Fraichard, T. & Ahuactzin, J.-M. (2001), Smooth path planning for cars, *in* 'Proceedings - 2001 IEEE International Conference on Robotics and Automation', Vol. 4, IEEE Robotics and Automation Society, Institute of Electrical and Electronics Engineers Inc., Seoul, South Korea, pp. 3722–3727.

Fraichard, T. & Scheuer, A. (2004), 'From Reeds and Shepp's to continuous-curvature paths', *IEEE Transactions on Robotics* **20**(6), 1025–1035.

Gawthrop, P. J. & Bevan, G. P. (2007), 'Bond-graph modeling', *IEEE Control Systems Magazine* **27**(2), 24–45.

Gehrig, S. K. & Stein, F. J. (2001), Elastic bands to enhance vehicle following, *in* 'IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC', ITSC, IEEE, Institute of Electrical and Electronics Engineers Inc., Oakland, CA, USA, pp. 597–602.

Germann, S., Würtenberger, M. & Daiß, A. (1994), Monitoring of the friction coefficient between tyre and road surface, in *Proceedings of the 1994 IEEE Conference on Control Applications. Part 1 (of 3)* IEE (1994), pp. 613–618.

Gillespie, T. D. (1992), *Fundamentals of vehicle dynamics*, Society of Automotive Engineers.

Gissinger, G. L., Chamaillard, Y. & Stemmelen, T. (1995), 'Modelling a motor vehicle and its braking system', *Mathematics and Computers in Simulation* **39**(5-6), 541–548.

Gissinger, G. L., Menard, C. & Constans, A. (2003), 'A mechatronic conception of a new intelligent braking system', *Control Engineering Practice* **11**(2), 163–170.

Godbole, D. N., Hagenmeyer, V., Sengupta, R. & Swaroop, D. (1997), Design of emergency maneuvers for automated highway system: Obstacle avoidance problem, *in* 'Proceedings of the IEEE Conference on Decision and Control', Vol. 5, IEEE, Institute of Electrical and Electronics Engineers Inc., San Diego, CA, USA, pp. 4774–4779.

Grant, M., Boyd, S. & Ye, Y. (2006), *Global Optimization: From Theory to Implementation (Nonconvex Optimization and Its Applications)*, Springer Science + Business Media, New York, USA, chapter Disciplined Convex Programming, pp. 155–210.

Gray, R. (2007), 'Engineers plan crash-proof computer car', *Sunday Telegraph*. **http://www.telegraph.co.uk/**earth/main.jhtml?xml=/earth/2007/12/30/scitech130.xml.

Gustafsson, F. (1997), 'Slip-based tire-road friction estimation', *Automatica* **33**(6), 1087–1099.

Gustafsson, F. (1998), 'Monitoring tire-road friction using the wheel slip', *IEEE Control Systems Magazine* **18**(4), 42–49.

Hamming, R. W. (1973), *Numerical Methods for Scientists and Engineers*, 2nd edn, McGraw-Hill.

Hansen, J., Murray-Smith, R. & Johansen, T. A. (2005), Nonparametric identification of linearizations and uncertainty using gaussian process models - application to robust wheel slip control, in *Proceedings of the IEEE Conference on Decision and Control* IEE (2005), pp. 5083–5088.

Hattori, Y., Ono, E. & Hosoe, S. (2006), 'Optimum vehicle trajectory control for obstacle avoidance problem', *IEEE/ASME Transactions on Mechatronics* **11**(5), 507–512.

Hearn, A. C. (1982), Reduce - a case study in algebra system development, *in* 'Lecture Notes in Computer Science', ACM, Special Interest Group on Symbolic and Algebraic Manipulation, Springer-Verlag, Marseille, Fr., pp. 263–272. `http://www.reduce-algebra.com`.

Heinzl, P., Lugner, P. & Plöchl, M. (2002), 'Stability control of a passenger car by combined additional steering and unilateral braking', *Vehicle System Dynamics* **37**(SUPPL), 221–233.

Hero of Alexandria (c. 300 CE), *Pneumatics*, .

Hunt, K. J., Wang, Y., Schinkel, M. & Schmitt-Hartmann, T. (2003), *Nonlinear and hybrid systems in automotive control*, in Johansson & Rantzer (2003), chapter 6: Controller design for hybrid systems using simultaneous D-stabilisation and its application to anti-lock braking systems (ABS), pp. 97–124.

IEE (1994), *IEEE Conference on Control Applications*, Institute of Electrical and Electronics Engineers Inc., Glasgow, UK.

IEE (2005), 44^{th} *IEEE Conference on Decision and Control*, Institute of Electrical and Electronics Engineers Inc., Seville, Spain.

IEE (2007), *2007 IEEE Intelligent Vehicles Symposium*, Institute of Electrical and Electronics Engineers Inc., Istanbul, Turkey.

Imsland, L., Johansen, T. A., Fossen, T. I., Grip, H. F., Kalkkuhl, J. C. & Suissa, A. (2006), 'Vehicle velocity estimation using nonlinear observers', *Automatica* **42**(12), 2091–2103.

International Organization for Standardization (1991), 'ISO 8855: Terms for road vehicle dynamics and road holding ability'.

International Organization for Standardization (1999), 'ISO 3888: Passenger cars - test track for a severe lane-change manoeuvre – part 1: Double lane-change'.

International Organization for Standardization (1999, 2002*a*), 'ISO 3888: Passenger cars - test track for a severe lane-change manoeuvre'.

International Organization for Standardization (2002*b*), 'ISO 3888: Passenger cars - test track for a severe lane-change manoeuvre – part 2: Obstacle avoidance'.

Jansen, S. T. H., Zegelaar, P. W. A. & Pacejka, H. B. (1999), 'The influence of in-plane tyre dynamics on ABS braking of a quarter vehicle model', *Vehicle System Dynamics* **32**(2), 249–261.

Jiang, F. & Gao, Z. (2000), An adaptive nonlinear filter approach to the vehicle velocity estimation for ABS, *in* 'Proceedings of the IEEE Conference on Control Applications', IEEE, Institute of Electrical and Electronics Engineers Inc., Anchorage, AK, USA, pp. 490–495.

Jiang, F. & Gao, Z. (2001), An application of nonlinear PID control to a class of truck ABS problems, *in* 'Proceedings of the IEEE Conference on Decision and Control', Vol. 1, IEEE, Institute of Electrical and Electronics Engineers Inc., Orlando, FL, USA, pp. 516–521.

Johansen, T. A., Hunt, K. J., Gawthrop, P. J. & Fritz, H. (1998), 'Off-equilibrium linearisation and design of gain scheduled control with application to vehicle speed control', *Control Engineering Practice* **6**(2), 167–180.

Johansen, T. A., Kalkkuhl, J., Lüdemann, J. & Petersen, I. (2001), Hybrid control strategies in ABS, in *Proceedings of the American Control Conference* AAC (2001), pp. 1704–1705.

Johansson, R. & Rantzer, A., eds (2003), *Nonlinear and hybrid systems in automotive control*, Springer-Verlag.

Jung, H., Kwak, B. & Park, Y. (2002), 'Improving the directional stability of a traction control system without additional sensors', *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering* **216**(8), 641–648.

Kalkkuhl, J., Johansen, T. A. & Lüdemann, J. (2002), 'Improved transient performance of nonlinear adaptive backstepping using estimator resetting based on multiple models', *IEEE Transactions on Automatic Control* **47**(1), 136–140.

Kaneko, J. & Shimamura, A. (1997), A design of lane change maneuver for automated vehicles, *in* 'Proceedings of the IEEE Conference on Decision and Control', Vol. 1, IEEE, Institute of Electrical and Electronics Engineers Inc., Tampa, Fl, USA, pp. 1031–1033.

Kazemi, R. & Zaviyeh, K. J. (2001), Development of a new ABS for passenger cars using dynamic surface control method, in *Proceedings of the American Control Conference* AAC (2001), pp. 677–683.

Kiencke, U. & Daiß, A. (1997), 'Observation of lateral vehicle dynamics', *Control Engineering Practice* **5**(8), 1145–1150.

Kobayashi, K., Cheok, K. C. & Watanabe, K. (1995), Estimation of absolute vehicle speed using fuzzy logic rule-based kalman filter, *in* 'Proceedings of the American Control Conference', Vol. 5, AACC, Seattle, WA, USA, pp. 3086–3090.

Konik, D. (2002), 'Development of the dynamic drive for the new 7 series of the BMW group', *International Journal of Vehicle Design* **28**(1-3), 131–149.

Kraft, H. J. & Leffler, H. (1990), 'The integrated brake and stability control system of the new BMW 850i', *SAE Transactions* **99**(sect 6), 291–299.

Kristiansson, B. & Lennartson, B. (2002), 'Robust and optimal tuning of PI and PID controllers', *IEE Proceedings: Control Theory and Applications* **149**(1), 17–25.

Lamiraux, F. & Laumond, J.-P. (2001), 'Smooth motion planning for car-like vehicles', *IEEE Transactions on Robotics and Automation* **17**(4), 498–502.

Lee, Y. & Żak, S. H. (2002), 'Designing a genetic neural fuzzy antilock-brake-system controller', *IEEE Transactions on Evolutionary Computing* **6**(2), 198–211.

Leith, D. J. & Leithead, W. E. (1998*a*), 'Gain-scheduled and nonlinear systems: dynamic analysis by velocity-based linearization families', *International Journal of Control* **70**(2), 289–317.

Leith, D. J. & Leithead, W. E. (1998*b*), 'Gain-scheduled controller design: An analytic framework directly incorporating non-equilibrium plant dynamics', *International Journal of Control* **70**(2), 249–269.

Leith, D. J., Tsourdos, A., White, B. & Leithead, W. (2001), 'Application of velocity-based gain-scheduling to lateral auto-pilot design for an agile missile', *Control Engineering Practice* **9**(10), 1079–1093.

Liceaga, J., Liceaga, E. & Amézquita, L. (2005), Multivariable gyroscope control by individual channel design, *in* 'Proceedings of the IEEE Conference on Control Applications', IEEE, Institution of Electrical Engineers, Toronto, Canada, pp. 785–790.

Lidner, L. (1993), Experience with the magic formula tyre model, *in* Pacejka (1991), pp. 30–46.

Liu, Z.-D., Lu, J., Shi, K.-B. & An, W. (2001), 'Integrated ABS/ASR/ACC system for the car', *Journal of Beijing Institute of Technology (English Edition)* **10**(3), 326–330.

Mancosu, F. & Arosio, D. (2005), Vehicle, road, tyre and electronic control systems interaction: increasing vehicle active safety by means of a fully integrated model for behaviour prediction in potentially dangerous situations, *in* 'Proceedings of the IEEE International Symposium on Industrial Electronics 2005', IEEE, Institute of Electrical and Electronics Engineers Inc., Dubrovnic, Croatia, pp. 337–342.

Massey, R. (2006), 'The self-driving Golf that would give Herbie a run for its money', *Daily Mail*. **http://www.dailymail.co.uk/**pages/live/articles/news/news.html?in_article_id=393401.

Max (2007), *Maxima Reference Manual*, version 5.13.0 edn. http://maxima.sourceforge.net.

Milliken, W. F. & Milliken, D. L. (1995), *Race car vehicle dynamics*, Society of Automotive Engineers.

Moler, C. (1988), Matlab - a mathematical visualization laboratory, *in* 'Compcon Spring '88: Thirty-Third IEEE Computer Society International Conference, Digest of Papers', Institute of Electrical and Electronics Engineers Inc., San Francisco, CA, USA, pp. 480–481. http://www.mathworks.com.

Mosedale, J., Purdy, A. & Clarkson, E. (2004), Contributory factors to road accidents, Technical report, Transport Statistics: Road Safety, Department for Transport.

Myers, T. J., Nöel, T., Parent, M. & Vlacic, L. (2005), Autonomous motion of a driverless vehicle operating among dynamic obstacles, in *Proceedings of the IEEE Conference on Decision and Control* IEE (2005), pp. 5071–5076.

Newton, I. (1687), *in* 'Philosophiae naturalis principia mathematica autore Js. Newton', number Wing / N1049 *in* 'Early English Books, 1641-1700 / 1553:13', Londoni : Jussu Societas Regiae ac typis Josephi Streater, prostant venales apud Sam. Smith ..., MDCLXXXVII [1687], pp. [8], 383, 400–510, [1] p., 1 folded leaf of plates. "Imprimatur S. Pepys, Reg. soc. praeses. Julii 5. 1686."; Reproduction of original in Cambridge University Library. **http://gateway.proquest.com/**openurl?ctx_ver=Z39.88-2003&res_id=xri:eebo&rft_id=xri:eebo:image:100057:13.

Nouillant, C., Assadian, F., Moreau, X. & Oustaloup, A. (2002), 'Feedforward and crone feedback control strategies for automobile ABS', *Vehicle System Dynamics* **38**(4), 293–315.

Oberle, H. J. (1990), 'Numerical computation of singular control functions in trajectory optimization problems', *Journal of Guidance, Control, and Dynamics* **13**(1), 153–159.

O'Reilly, J. & Leithead, W. E. (1991), 'Multivariable control by 'individual channel design", *International Journal of Control* **54**(1), 1–46.

Oustaloup, A., Moreau, X. & Nouillant, M. (1996), 'The CRONE suspension', *Control Engineering Practice* **4**(8), 1101–1108.

Pacejka, H. B. & Bakker, E. (1993), The magic formula tyre model, *in* Pacejka (1991), pp. 1–18.

Pacejka, H. B., ed. (1991), 1^{st} *International Colloquium on Tyre Models for Vehicle Dynamics Analysis*, Vehicle System Dynamics, Swets, Delft, NL.

Panagopoulos, H., Åström, K. J. & Hägglund, T. (2002), 'Design of PID controllers based on constrained optimization', *IEE Proceedings: Control Theory and Applications* **149**(1), 32–40.

Paraskevopoulos, P. N., Pasgianos, G. D. & Arvanitis, K. G. (2004), 'New tuning and identification methods for unstable first order plus dead-time processes based on pseudoderivative feedback control', *IEEE Transactions on Control Systems Technology* **12**(3), 455–464.

Pasterkamp, W. R. & Pacejka, H. B. (1997), 'The tyre as a sensor to estimate friction', *Vehicle System Dynamics* **27**(5-6), 409–422.

Pauwelussen, J. P., Gootjes, L., Schröder, C., Köhne, K. U., Jansen, S. & Schmeitz, A. (2003), 'Full vehicle ABS braking using the SWIFT rigid ring tyre model', *Control Engineering Practice* **11**(2), 199–207.

Pei, S.-C. & Horng, J.-H. (1998), 'Finding the optimal driving path of a car using the modified constrained distance transformation', *IEEE Transactions on Robotics and Automation* **14**(5), 663–670.

Quinlan, S. & Khatib, O. (1993), Elastic bands: connecting path planning and control, *in* 'Proceedings - 1993 IEEE International Conference on Robotics and Automation', Vol. 2, IEEE, Institute of Electrical and Electronics Engineers Inc., Atlanta, GA, USA, pp. 802–807.

Rajamani, R., Tan, H.-S., Law, B. K. & Zhang, W.-B. (2000), 'Demonstration of integrated longitudinal and lateral control for the operation of automated vehicles in platoons', *IEEE Transactions on Control Systems Technology* **8**(4), 695–708.

Rauh, J. (2003), 'Virtual development of ride and handling characteristics for advanced passenger cars', *Vehicle System Dynamics* **40**(1–3), 135–155.

Rugh, W. J. (1991), 'Analytical framework for gain scheduling', *IEEE Control Systems Magazine* **11**(1), 79–84.

Sakai, H. (1981*a*), 'Theoretical and experimental studies on the dynamic properties of tyres, part 1: Review of theories of rubber friction', *International Journal of Vehicle Design* **2**(1), 78–110.

Sakai, H. (1981*b*), 'Theoretical and experimental studies on the dynamic properties of tyres, part 2: Experimental investigation of rubber friction and deformation of a tyre', *International Journal of Vehicle Design* **2**(2), 182–226.

Sakai, H. (1981*c*), 'Theoretical and experimental studies on the dynamic properties of tyres, part 3: Calculation of the six components of force and moment of a tyre.', *International Journal of Vehicle Design* **2**(3), 335–372.

Sakai, H. (1982), 'Theoretical and experimental studies on the dynamic properties of tyres, part 4: Investigations of the influence of running conditions by calculation and experiment', *International Journal of Vehicle Design* **3**(3), 333–375.

Samadi, B., Kazemi, R., Nikravesh, K. Y. & Kabganian, M. (2001), Real-time estimation of vehicle state and tire-road friction forces, in *Proceedings of the American Control Conference* AAC (2001), pp. 3318–3323.

Sattel, T. & Brandt, T. (2005), Ground vehicle guidance along collision-free trajectories using elastic bands, *in* 'Proceedings of the American Control Conference', Vol. 7, AACC, Institute of Electrical and Electronics Engineers Inc., Portland, OR, USA, pp. 4991–4996.

Schinkel, M. (2002), Nondeterministic hybrid dynamical systems, PhD thesis, University of Glasgow, Dept. Mechanical Engineering, Glasgow, Scotland, G12 8QQ.

Schinkel, M. & Hunt, K. (2002), Anti-lock braking control using a sliding mode like approach, in *Proceedings of the American Control Conference* AAC (2002), pp. 2386–2391.

Schofield, B., Hägglund, T. & Rantzer, A. (2006), Vehicle dynamics control and controller allocation for rollover prevention, *in* 'Proceedings of the IEEE Conference on Control Applications', IEEE, Institute of Electrical and Electronics Engineers Inc., Munich, Germany.

Schuller, J., Brangs, P., Rothfuss, R., Lutz, A. & Breit, R. (2002), 'Development methodology for dynamic stability control systems', *International Journal of Vehicle Design* **28**(1-3), 37–56.

Schuller, J., Schaeffer, F., Neukum, A. & Krueger, H.-P. (1999), A driver model as development tool for vehicle handling design, Vol. 33 of *Vehicle Systems Dynamics*, Swets, Pretoria, South Africa, pp. 110–121.

Seron, M. M. & Goodwin, G. C. (1996), 'Sensitivity limitations in nonlinear feedback control', *Systems & Control Letters* **27**(4), 249–54.

Shamma, J. S. & Athans, M. (1990), 'Analysis of gain scheduled control for nonlinear plants', *IEEE Transactions on Automatic Control* **35**(8), 898–907.

Sharkey, N. (2007), 'The programmable robot of ancient greece', *New Scientist* (2611), 32–35.

Sharp, R. S. & Bettella, M. (2003), 'On the construction of a general numerical tyre shear force model from limited data', *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering* **217**(3), 165–172.

Shiller, Z. & Sundar, S. (1998), 'Emergency lane-change manoeuvres of autonomous vehicles', *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME* **120**(1), 37–44.

Shim, T. & Margolis, D. (2005), 'Dynamic normal force control for vehicle stability enhancement', *International Journal of Vehicle Autonomous Systems* **3**(1), 1–14.

Sledge Jr., N. H. & Marshek, K. M. (1998), Development and validation of an optimized emergency lane-change trajectory, *in* 'Vehicle Dynamics and Simulation', number 1361, SAE, Society of Automotive Engineers, Detroit, MI., USA, pp. 103–121.

Society of Automotive Engineers (1989), 'Vehicle aerodynamics terminology'.

Solyom, S. & Ingimundarson, A. (2002), 'Synthesis method for robust PID controllers for a class of uncertain systems', *Asian Journal of Control* **4**(4), 381–387.

Solyom, S. & Rantzer, A. (2003), *Nonlinear and hybrid systems in automotive control*, in Johansson & Rantzer (2003), chapter 5: ABS Control - a design model and control structure, pp. 85–96.

Swaroop, D. & Yoon, S. M. (1999), 'Integrated lateral and longitudinal vehicle control for an emergency lane change manoeuvre design', *International Journal of Vehicle Design* **21**(2), 161–174.

CEMACS (2005), Confidential test vehicle specification, Technical Report D11, CEMACS.

Vahidi, A. & Eskandarian, A. (2004), 'Research advances in intelligent collision avoidance and adaptive cruise control', *IEEE Transactions on Intelligent Transportation Systems* **4**(3), 143–153.

Van Nieuwstadt, M. J. & Murray, R. M. (1998), 'Real-time trajectory generation for differentially flat systems', *International Journal of Robust and Nonlinear Control* **8**(11), 995–1020. Trajectory generation;Differentially flat systems;.

Villegas, C., Leith, D. J., Shorten, R. N. & Kalkkuhl, J. (2007), A disturbance response decoupling controller for emulating vertical dynamics of vehicles, in *2007 IEEE Intelligent Vehicles Symposium* IEE (2007), pp. 877–882.

Weisstein, E. W. (2004), 'Matrix norm', From Mathworld – A Wolfram Web Resource. `http://mathworld.wolfram.com/`MatrixNorm.html.

Woodcroft, B., ed. (1851), *The Pneumatics of Hero of Alexandria from the original greek*, Inventions of the Ancients, Taylor Walton and Maberly, London.

Wu, M.-C. & Shih, M.-C. (2003), 'Simulated and experimental study of hydraulic anti-lock braking system using sliding-mode PWM control', *Mechatronics* **13**(4), 331–351.

Yi, J., Alvarez, L., Claeys, X. & Horowitz, R. (2003), 'Emergency braking control with an observer-based dynamic tire/road friction model and wheel angular velocity measurement', *Vehicle System Dynamics* **39**(2), 81–97.

Yi, J., Alvarez, L., Claeys, X., Horowitz, R. & de Wit, C. C. (2001), Emergency braking control with an observer-based dynamic tire/road friction model and wheel angular velocity information, in *Proceedings of the American Control Conference* AAC (2001), pp. 19–24.

Yi, J., Alvarez, L., Horowitz, R. & de Wit, C. C. (2000), Adaptive emergency braking control using a dynamic tire/road friction model, *in* 'Proceedings of the IEEE Conference on Decision and Control', Vol. 1, IEEE, Institute of Electrical and Electronics Engineers Inc., Sydney, NSW, Australia, pp. 456–461.

Yi, K. & Chung, J. (2001), 'Nonlinear brake control for vehicle CW/CA systems', *IEEE/ASME Transactions on Mechatronics* **6**(1), 17–25.

Yi, K., Woo, M., Kim, S. & Lee, S. (1999), 'Study on a road-adaptive CW/CA algorithm for automobiles using hil simulations', *JSME International Journal, Series C* **42**(1), 163–170.

Yu, H. & Ozguner, U. (2002), Extremum-seeking control strategy for ABS system with time delay, in *Proceedings of the American Control Conference* AAC (2002), pp. 3753–3758.